UNIVERSITÀ
DEGLI STUDI
DI PALERMO

NDS
NETWORKING AND DISTRIBUTED SYSTEMS
RESEARCH GROUP

Tecniche di Machine Learning nei sistemi di rilevamento delle intrusioni

Tesi di Laurea Magistrale in Ingegneria Informatica

Massimiliano Cardillo

Relatore: Prof. Salvatore Gaglio

Correlatore: Prof. Alessandra De Paola

Tecniche di Machine Learning nei sistemi di rilevamento delle intrusioni

TESI DI LAUREA DI
Massimiliano Cardillo

RELATORE
Prof. Salvatore Gaglio

CORRELATORE
Prof.ssa Alessandra De Paola

Sommario

I sistemi di rilevamento delle intrusioni (IDS) rappresentano una componente cruciale nella sicurezza delle reti. Tra le varie tipologie di IDS assumono particolare importanza gli IDS tipo anomaly-based, che sfruttano tecniche di Intelligenza Artificiale per effettuare il rilevamento: in letteratura infatti sono molteplici le applicazioni di tecniche di machine learning negli IDS. In questo lavoro di tesi viene dapprima effettuata una panoramica dei principali metodi di tipo supervisionato, non supervisionato ed ibrido utilizzati in tale ambito. Subito dopo viene effettuata un'analisi sperimentale dei metodi supervisionati e non supervisionati analizzati, su uno dei dataset più recenti in tale dominio ed infine viene proposto un metodo ibrido per il rilevamento di intrusioni che combina l'algoritmo K-means ed un classificatore Random Forest (RF). L'algoritmo K-means svolge qui un duplice ruolo: in primo luogo è applicato per effettuare il clustering a partire dal frammento di dataset in ingresso, quindi è applicato nuovamente a ciascun cluster generato al passo precedente per ottenere un partizionamento. In seguito una tecnica di Feature Extraction basata su misure statistiche è applicata alle partizioni ottenute per generare un dataset con il quale addestrare un classificatore RF. I risultati della valutazione sperimentale hanno dimostrato che il metodo proposto rappresenta una migliore alternativa rispetto ad alcuni dei metodi non supervisionati più comunemente utilizzati.

Ringraziamenti

Un sentito ringraziamento va ai miei relatori, il professore Gaglio e la professoressa De Paola per avermi supportato e guidato nella stesura di questo lavoro con disponibilità e puntualità.

Desidero ringraziare tutti i colleghi universitari conosciuti in questi anni e gli amici con i quali ho condiviso questo percorso di studi.

Infine vorrei ringraziare in particolar modo i miei genitori e mia nonna per essermi stati sempre accanto, per avermi sostenuto, sopportato ed incoraggiato nel raggiungimento di questo traguardo.

Indice

Introduzione 6

1. Sistemi di rilevamento delle intrusioni 8

1.1 Panoramica 8

1.2 Classificazioni dei sistemi di rilevamento delle intrusioni 9

1.2.1 Tipi di implementazione 9

1.2.2 Tipi di architettura 10

1.2.3 Modalità di monitoraggio 11

1.2.4 Metodi di rilevamento 12

1.2.5 Classi di attacchi 12

2. Metodi di Machine Learning applicati per il rilevamento delle intrusioni 14

2.1 Apprendimento automatico 14

2.2 Metodi di apprendimento supervisionato 15

2.2.1 Reti neurali artificiali 16

2.2.2 Alberi di decisione 19

2.2.3 Random Forest 22

2.2.4 K-Nearest Neighbors 24

2.2.5 Algoritmi genetici 26

2.2.6 Naïve Bayes 27

2.2.7 Support Vector Machines 29

2.3 Metodi di apprendimento non supervisionato 31

2.3.1 Clustering gerarchico 32

2.3.2 K-means 34
2.3.3 Expectation-Maximization 36
2.3.4 Self-Organizing Maps 37
2.3.5 DBSCAN 40
2.4 Riduzione della dimensionalità dei dati 43
2.4.1 Selezione delle features basata sulla correlazione 44
2.4.2 Analisi delle componenti principali 45
2.5 Confronto tra metodi supervisionati e non supervisionati 46
2.6 Metodi ibridi 46
3. Valutazione sperimentale delle tecniche di machine learning e analisi delle problematiche 49
3.1 Tool di sviluppo 49
3.2 Dataset CICIDS2017 49
3.3 Metodologia di valutazione 55
3.4 Preprocessing dei dati 55
3.5 Riduzione della dimensionalità 56
3.6 Split del dataset 56
3.7 Misure di prestazioni 57
3.8 Risultati sperimentali dei metodi supervisionati 59
3.9 Risultati sperimentali dei metodi non supervisionati 71
4. Soluzione proposta e valutazione sperimentale 78
4.1 Metodologia **Error! Bookmark not defined.**
4.2 Test del metodo proposto **Error! Bookmark not defined.**
4.3 Risultati sperimentali **Error! Bookmark not defined.**
5. Conclusioni 79
Bibliografia 80

Introduzione

Al giorno d'oggi il tema della sicurezza informatica è più che mai di vitale importanza data la crescita esponenziale della condivisione delle informazioni raggiunta con la diffusione di Internet ed il costante incremento della velocità delle connessioni. In tutto il mondo vi sono sempre più hacker in grado di sferrare sofisticate tipologie di cyber-attacchi ai danni delle organizzazioni, delle infrastrutture degli enti governativi. Tra gli attacchi più insidiosi vi sono quelli che hanno come obiettivo l'accesso non autorizzato ad un sistema informatico o ad una rete, anche noto come intrusione. Con il termine intrusione si intende una qualsiasi sequenza di azioni finalizzata a compromettere l'integrità, la disponibilità o la riservatezza di una data risorsa in un sistema informatico. L'integrità garantisce che i dati non vengano alterati da utenti non autorizzati, la disponibilità assicura agli utenti legittimi del sistema il pronto accesso ai dati ed ai servizi richiesti, mentre la riservatezza garantisce che i dati non possano essere acceduti o visualizzati da parte di soggetti esterni non autorizzati. Un sistema informatico è considerato sicuro quando questi tre requisiti sono soddisfatti.

Le intrusioni possono essere finalizzate ad esempio ad attacchi alle reti informatiche attraverso lo sfruttamento di una vulnerabilità nota di un servizio, a tentativi di accesso agli host della rete tramite l'innalzamento illecito dei privilegi degli utenti, ad accessi non autorizzati ai file privati o sensibili, all'installazione di programmi malevoli come virus, trojan e worm, o al furto di dati o leak, che si traducono in ingenti perdite di denaro e danni di immagine per le compagnie coinvolte. Oggigiorno le organizzazioni, per salvaguardare la sicurezza delle loro infrastrutture di rete, affiancano ai tradizionali strumenti quali firewall e antivirus dei sistemi in grado di rilevare le intrusioni. Nel corso degli ultimi anni la diffusione dell'Intelligenza Artificiale, in particolare lo sviluppo delle discipline legate al Machine Learning, e la crescente disponibilità di dati forniti dalle moderne applicazioni ICT, hanno permesso lo sviluppo di sistemi di rilevamento delle intrusioni in grado di applicare diverse tecniche basate sull'apprendimento automatico per monitorare il traffico in entrata nelle reti e rilevare attività potenzialmente malevole.

Contributo di questa tesi

Il presente lavoro di tesi si concentra sull'analisi delle principali tecniche di machine learning applicate ai sistemi di rilevamento delle intrusioni. Dal momento che in letteratura sono presenti numerosi contributi per quel che concerne l'applicazione di tecniche di apprendimento supervisionato nel contesto del rilevamento delle intrusioni, in questa sede si è deciso di testare preliminarmente, oltre a tali tecniche, alcune delle principali tecniche di tipo non supervisionato, in particolare le tecniche di clustering, e di prendere poi in esame quelle supervisionate nel contesto di un sistema ibrido, in grado di combinare il meglio dei due approcci. In particolare viene proposta e studiata una versione modificata di un sistema di rilevamento ibrido presente in letteratura allo scopo di fornire un'alternativa rispetto ai metodi tradizionali impiegati. Vengono inoltre presentati i risultati ottenuti dalla fase di valutazione sperimentale che hanno anche guidato le diverse scelte progettuali intraprese per la realizzazione della soluzione proposta.

L'elaborato è strutturato nei seguenti cinque capitoli:

- nel capitolo 1 viene presentata una panoramica generale sui sistemi di rilevamento delle intrusioni, sulle loro tipologie e classificazioni;
- nel capitolo 2 vengono presentati i diversi metodi di machine learning adottati nel rilevamento di intrusioni allo stato dell'arte in letteratura;
- nel capitolo 3 viene descritta la fase di valutazione sperimentale, gli strumenti software ed i dataset utilizzati e vengono presentati i risultati ottenuti;
- nel capitolo 4 viene descritta l'architettura della soluzione proposta, vengono forniti i dettagli relativi alla sua realizzazione ed i risultati dei test sperimentali;
- nel capitolo 5 sono presentate le conclusioni complessive sul lavoro svolto e vengono delineati i possibili sviluppi futuri.

1. Sistemi di rilevamento delle intrusioni

In questo capitolo è presentata una breve introduzione dei concetti più importanti inerenti i sistemi di rilevamento delle intrusioni, le loro tipologie e classificazioni.

1.1 Panoramica

Il National Institute of Standards and Technology (NIST) classifica il rilevamento delle intrusioni come "*il processo di monitoraggio degli eventi che si verificano in un sistema informatico o in una rete e di analisi degli stessi alla ricerca di segni di intrusioni, definiti come tentativi di compromettere la riservatezza, l'integrità, la disponibilità o di aggirare i meccanismi di sicurezza di un computer o di una rete*" [36].

Un sistema di rilevamento delle intrusioni o *Intrusion Detection System* (IDS), è uno strumento software o hardware che monitora un sistema o una rete, eseguendo la scansione delle risorse del sistema o del traffico di rete e segnalando eventuali attività potenzialmente malevole o violazioni delle politiche di sicurezza. Gli IDS rappresentano una componente di fondamentale importanza di qualsiasi architettura di network security, e forniscono al sistema un livello di difesa il cui obiettivo è l'identificazione di potenziali minacce e la generazione di report e avvisi tempestivi, in particolar modo nelle prime fasi dell'attacco, in modo da ridurne l'impatto. Se viene rilevata una qualunque attività sospetta o violazione questa viene prontamente segnalata all'amministratore del sistema. Di norma il compito di un IDS è solo quello di rilevare attacchi, non di filtrarli o fermarli, sebbene alcuni dei più avanzati IDS possano effettuare ulteriori azioni in risposta ad essi.

I sistemi di rilevamento delle intrusioni sono tuttavia soggetti alla generazione di falsi allarmi, di conseguenza un aspetto cruciale della loro progettazione è rappresentato dallo step di *fine-tuning* al fine di rendere tali sistemi in grado di riconoscere l'aspetto del normale traffico nella rete rispetto alle attività dannose. È possibile effettuare diverse classificazioni degli IDS in base a più criteri.

1.2 Classificazioni dei sistemi di rilevamento delle intrusioni

In questa sezione sono presentate le diverse classificazioni dei sistemi di rilevamento delle intrusioni in base al tipo di implementazione, all'architettura, al metodo di rilevamento e alla tipologia di apprendimento automatico impiegati.

1.2.1 Tipi di implementazione

Per quanto riguarda il tipo di implementazione, gli IDS possono essere suddivisi, come mostrato nella Figura 1, in:

- *Host-based (HIDS):* un IDS basato su host prevede che su ciascun host all'interno di una rete vi sia un agente locale in grado di monitorare i pacchetti in entrata e in uscita, e di avvisare l'amministratore se viene rilevata una qualsiasi azione sospetta, non autorizzata o contraria alle politiche di sicurezza del sistema. Esempi di caratteristiche che un HIDS potrebbe monitorare, oltre al traffico di rete dell''host, sono i registri di sistema, i processi in esecuzione, l'attività delle applicazioni, l'accesso e la modifica dei file, e le eventuali modifiche ai file di configurazione del sistema e delle applicazioni. Gli IDS basati su host sono più comunemente distribuiti su host critici, come server accessibili pubblicamente e server contenenti informazioni sensibili.
- *Network-based (NIDS):* i NIDS sono degli strumenti, installati in uno specifico punto di una rete, i quali monitorano e analizzano il traffico proveniente da tutti i dispositivi della rete, a livello di pacchetto o a livello di flusso, alla ricerca di anomalie; una volta identificato un attacco o osservato un comportamento anomalo generano un avviso che viene poi inoltrato all'amministratore. Uno dei principali esempi di questa tipologia è Snort, tra i più noti NIDS.
- *Ibridi:* gli IDS implementati sia a livello di host che di rete, combinano i dati dell'agente host o del sistema con le informazioni di rete per sviluppare una visione completa del sistema.

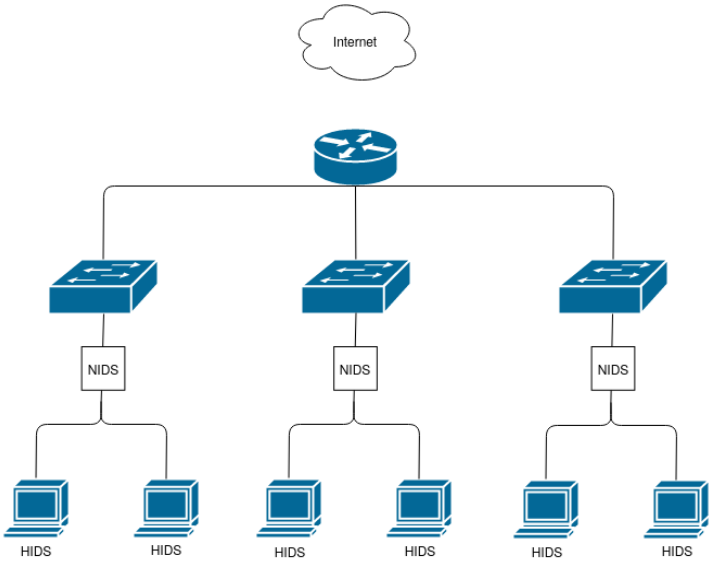

Figura 1: Schema di una struttura di rete che combina gli HIDS ed i NIDS tratto da [26].

1.2.2 Tipi di architettura

Gli IDS possono essere classificati in base alla loro architettura di comunicazione in (si veda la Figura 2):

- *Centralized:* in un IDS centralizzato sono presenti diversi IDS (*HIDS* o *NIDS*) in grado di monitorare il comportamento del rispettivo host o il traffico nella rete. I dati di ciascuno di essi sono condivisi con un'unità di analisi centrale e questi possono includere segnalazioni di un rilevamento su un host o dati estratti dal traffico di rete locale. L'unità di analisi applica algoritmi di correlazione sugli allarmi ricevuti o algoritmi di rilevamento standard sul traffico di rete in entrata. Questa tuttavia rappresenta un collo di bottiglia delle prestazioni del sistema, il quale non è pertanto scalabile, e un singolo punto di vulnerabilità o *Single Point of Failure* (SPoF) del sistema.
- *Decentralized:* gli IDS decentralizzati presentano una struttura gerarchica di punti di monitoraggio o più distribuzioni IDS autonome. Attraverso tale struttura, essi evitano le problematiche degli IDS centralizzati, in quanto ogni nodo all'interno della gerarchia applica un preprocessing e algoritmi di correlazione sui dati monitorati, e

se necessario trasmette un avviso al nodo superiore, che a sua volta può ripetere il processo, fino a quando i dati convergono all'unità di analisi centrale in cima alla gerarchia.

- *Distributed:* in un IDS distribuito ogni dispositivo rappresenta un'unità di analisi, pertanto essi condividono equamente il rilevamento. Si tratta quindi in genere di un'architettura di tipo *Peer-to-Peer* (P2P), in cui i dati monitorati sono correlati, aggregati e analizzati in modo completamente distribuito tra tutti i dispositivi.

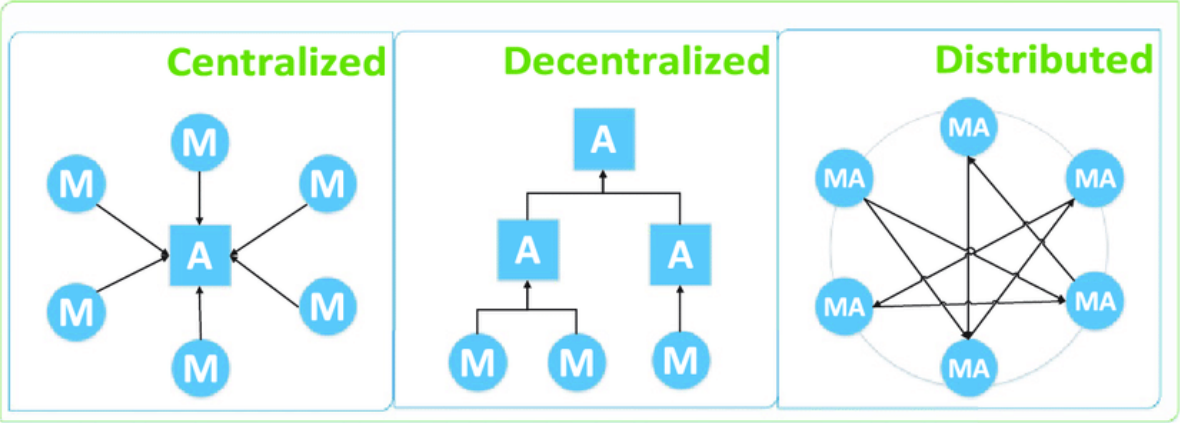

Figura 2: Schema delle diverse architetture degli IDS tratto da [35].

1.2.3 Modalità di monitoraggio

Gli IDS network-based possono essere a loro volta classificati in base al posizionamento all'interno della rete nelle seguenti tipologie:

- *In-line:* un IDS viene posizionato in un punto nella rete in modo che tutto il traffico debba attraversarlo, proprio come avviene con il flusso di traffico associato a un firewall. La motivazione principale di tale implementazione è quella di consentire all'IDS di fermare gli attacchi bloccando il traffico di rete. Gli IDS *in-line* vengono generalmente posizionati dove verrebbero posizionati firewall di rete e altri dispositivi di sicurezza di rete, ovvero in corrispondenza delle divisioni tra le reti, come nelle connessioni con reti esterne e nei confini tra diverse reti interne. Questa tipologia di IDS è detta anche *Intrusion Detection and Prevention System* (*IDPS*);
- *Passive:* l'IDS viene connesso a un router o ad uno switch all'interno della rete in modo che esso possa monitorare una copia del traffico di rete; nessun pacchetto di

traffico effettivo passa attraverso l'IDS. Dal momento che questo opera solo su una copia del traffico sarà impossibilitato a bloccare o filtrare i reali pacchetti in transito nella rete e potrà soltanto segnalare all'amministratore della rete l'eventuale presenza di attività sospette (*passive behavior*).

1.2.4 Metodi di rilevamento

Gli IDS possono essere inoltre classificati in base al metodo di rilevamento adottato. Queste sono principalmente di tipo:

- *Signature-based (o misuse-based):* gli IDS basati sul riconoscimento di firme mantengono un database di firme (*signatures*) predefinite corrispondenti ad attacchi noti ed effettuano il rilevamento confrontandole con il traffico in entrata applicando un *pattern matching*. Essi sono in grado di rilevare efficacemente gli attacchi la cui firma è presente nel sistema producendo un tasso di falsi positivi molto basso, ma non sono in grado di rilevare nuove tipologie di attacchi (*zero-day attacks)*. Le prestazioni sono strettamente legate alla qualità del database: vi è la necessità di mantenerlo aggiornato e, all'aumentare delle sue dimensioni, aumentano anche il tempo di elaborazione ed il costo.
- *Anomaly-based:* gli IDS basati sul rilevamento di anomalie sono applicati ad un dataset e mirano a modellare un comportamento "normale" al fine di identificare anomalie o istanze che non sono conformi ad esso. Sono in grado di rilevare sia attacchi noti che sconosciuti ma producono un alto tasso di falsi positivi. Possono essere impiegati per creare nuove *signatures* utilizzabili con il precedente approccio. Per migliorare le prestazioni degli IDS Anomaly-based numerosi studi recenti hanno applicato ad essi diversi algoritmi di Machine Learning (ML).

- *Ibridi*: le tecniche ibride combinano gli approcci *signature-based* ed *anomaly-based* in modo tale da aumentare i tassi di rilevamento di intrusioni note e diminuire il tasso di falsi positivi per gli attacchi sconosciuti. La maggior parte degli IDS *anomaly-based* presenti in letteratura sono in realtà ibridi.

1.2.5 Classi di attacchi

In letteratura gli attacchi informatici sono classificati in base alle azioni effettuate e agli obiettivi degli attacchi. In particolare all'interno del dataset DARPA 1998 [1] sono

individuate quattro principali categorie di attacchi che un IDS deve essere in grado di rilevare. Queste sono:

- *Denial-of-Service (DoS):* mirano a negare la disponibilità di risorse o servizi di un sistema o di un'infrastruttura di rete agli utenti legittimi attraverso l'invio di un enorme quantità di richieste all'host target;
- *User-to-Root (U2R):* l'attaccante mira ad ottenere l'accesso di root o admin su un dato sistema nel quale possiede solo il livello di accesso utente. Un esempio comune di tale attacco è il buffer overflow;
- *Remote-to-Local (R2L):* ha l'obiettivo di ottenere l'accesso locale al sistema target remoto tramite l'invio di pacchetti sulla stessa rete;
- *Probe (o Scan):* mira ad acquisire informazioni sulle risorse di rete o del sistema target (eventuali porte aperte, servizi in esecuzione sull'host, ..) attraverso scansioni della rete. L'attaccante può in tal modo individuare indirizzi IP validi di dispositivi sui quali testare eventuali vulnerabilità da sfruttare in attacchi successivi.

Tra questi gli attacchi U2R e R2L risultano i più difficili da rilevare poiché essi sono molto simili al traffico normale. Tali classi di attacco risultano tuttavia obsolete e pertanto non riflettono le attuali tendenze di attacco. Una classificazione di attacchi più recente e che meglio riflette gli attuali pattern di attacco è discussa nel paragrafo 3.2.

2. Metodi di Machine Learning applicati per il rilevamento delle intrusioni

Lo scopo di questo capitolo è fornire una panoramica generale dei metodi di machine learning utilizzati per il rilevamento delle intrusioni nei lavori presentati nella letteratura scientifica.

2.1 Apprendimento automatico

Una delle principali definizioni di Machine Learning (ML) si deve a Tom M. Mitchell, il quale afferma che: "*Si dice che un programma apprende dall'esperienza E con riferimento ad alcune classi di compiti T e con misurazione della performance P, se le sue performance nel compito T, come misurato da P, migliorano con l'esperienza E*".

Oggi con il termine *machine learning* o apprendimento automatico si fa riferimento a quella branca dell'intelligenza artificiale che ha come obiettivo la realizzazione di sistemi in grado di apprendere dai dati disponibili e di effettuare operazioni, che in genere includono il riconoscimento di pattern, la classificazione, la regressione o il clustering dei dati, senza essere stati esplicitamente programmati.

Un algoritmo di ML prevede che un modello venga addestrato su un insieme di dati di addestramento o *training set*, generalmente rappresentati tramite un vettore n-dimensionale di caratteristiche o *features,* il quale viene presentato per molte volte al modello in modo tale che questo sia in grado di apprendere dalla struttura intrinseca dei dati e di aggiustare di conseguenza i propri parametri. Ogni presentazione completa dell'insieme di addestramento è definita un'epoca. Dopo un certo numero di epoche il modello viene testato su un insieme di test diverso dall'insieme di addestramento, allo scopo di validare il modello e di testare la sua capacità di generalizzare a partire dagli esempi. Tale processo viene effettuato poiché un modello di ML può incorrere in fenomeni quali l'*overfitting*, ovvero il sovradattamento ai dati presentati, che si traduce in una scarsa capacità di generalizzazione da parte del modello, e l'*underfitting*, cioè l'incapacità del modello di apprendere dagli esempi forniti dovuta alla presentazione di un numero di epoche non adeguato o ad un'eccessiva semplicità del modello.

Tra le modalità di apprendimento adottate dai modelli di ML è possibile distinguere tre paradigmi principali:

- *Apprendimento supervisionato:* il sistema è realizzato in modo da apprendere a partire da degli esempi forniti in input, ossia dati etichettati con un output predefinito, con l'obiettivo di determinare una funzione nascosta o una regola generale che spieghi i dati;
- *Apprendimento non supervisionato:* i dati forniti in ingresso al sistema non sono etichettati e non hanno una struttura definita. Viene quindi effettuato un *pattern recognition* in modo da identificare delle strutture nascoste (pattern) nei dati, in modo da poterli prevedere;
- *Apprendimento per rinforzo:* un agente interagisce con un ambiente dinamico nel tentativo di soddisfare un goal ed in base alle sue azioni esso viene premiato o penalizzato dall'ambiente.

Nel prosieguo ci si focalizzerà sui principali metodi di apprendimento supervisionati e non supervisionati applicati secondo l'attuale stato dell'arte nell'ambito del rilevamento delle intrusioni. In base al tipo di apprendimento utilizzato nelle tecniche di machine learning che essi applicano gli IDS di tipo *anomaly-based* possono essere a loro volta suddivisi in:

- *Supervisionati (o Semi-supervisionati);*
- *Non supervisionati;*
- *Ibridi.*

2.2 Metodi di apprendimento supervisionato

I metodi di apprendimento supervisionato sono largamente utilizzati in letteratura e prevedono che un modello venga addestrato per apprendere da esempi, ossia dati etichettati. Ad ogni nuova istanza, un classificatore tenta di assegnarla a una delle classi predefinite. Secondo [1] e [4] tra le tecniche di tipo supervisionato più comunemente utilizzate nel rilevamento di intrusioni vi sono quelle basate su:

- *Reti neurali artificiali;*

- *Alberi di decisione;*
- *Random forest;*
- *K-nearest neighbors;*
- *Algoritmi genetici;*
- *Naive Bayes;*
- *Support vector machines.*

Nel seguito ciascuno di questi metodi verrà descritto ed analizzato, e verranno riportati alcuni lavori presentati nella letteratura scientifica che li adottano per il rilevamento delle intrusioni.

2.2.1 Reti neurali artificiali

Le reti neurali artificiali o Artificial Neural Network (ANN) sono uno dei metodi di apprendimento automatico più applicati e sono ispirate alla struttura ed al funzionamento biologico del cervello umano, il quale contiene circa 100 miliardi di neuroni interconnessi. Le ANN sono basate sugli studi di McCulloch-Pitts, che nel 1943 realizzarono un modello matematico del comportamento del neurone. Un neurone è composto da un corpo cellulare, da delle ramificazioni dette dendriti e da un lungo ramo, l'assone, il quale si connette ai dendriti di altri neuroni formando le sinapsi. Un neurone riceve in ingresso segnali (impulsi) da altri neuroni attraverso le sinapsi, le quali possono attenuare o amplificare il segnale trasmesso. Se i segnali che arrivano dagli altri neuroni sono superiori ad una certa soglia il neurone emette l'impulso, il quale verrà trasmesso agli altri neuroni.

Il meccanismo delle sinapsi viene modellato matematicamente associando ad ogni connessione un peso, che può essere positivo (contributo eccitatorio) o negativo (contributo inibitorio). Pertanto l'operazione preliminare che si effettua in una rete neurale è la somma pesata degli ingressi, la quale viene poi confrontata con una funzione di attivazione a soglia. Se il valore della somma è superiore alla soglia si dice che il neurone è attivo, altrimenti esso è inattivo. Generalmente si tratta di una funzione non lineare, come la funzione gradino o la funzione segno, che viene spesso approssimata con funzione continua come una sigmoide o una funzione logistica. Questa applicata alla somma pesata degli ingressi restituisce l'uscita del neurone. Nella Figura 3 è mostrato il funzionamento di un neurone artificiale.

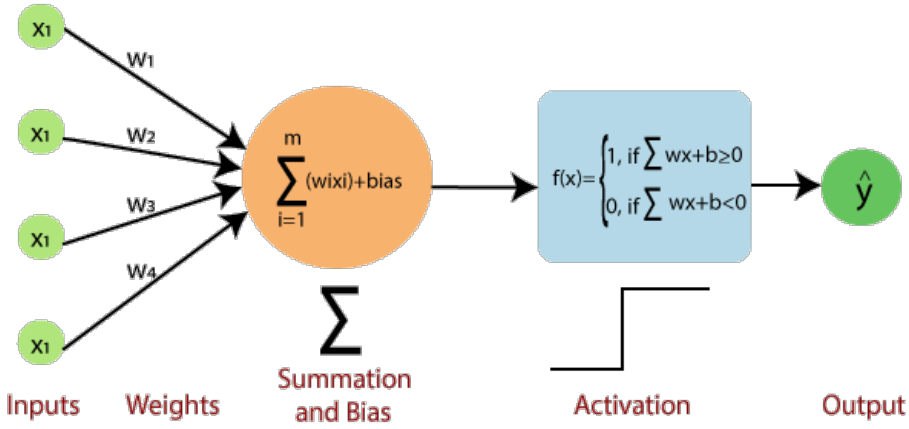

Figura 3: Schema del funzionamento di un neurone tratto da [27].

Più che al singolo neurone, chiamato anche *percettrone*, il principale interesse è rappresentato dalla costruzione di reti di neuroni. Vi sono varie architetture di rete in letteratura, tra le quali le più utilizzate sono le cosiddette *feed-forward networks* con percettroni multi-strato, in inglese *Multilayer Perceptron (MLP)*, ossia percettroni organizzati in strati successivi e nei quali le connessioni sono solo in avanti ed i segnali di ingresso si propagano direttamente verso l'uscita. In una rete neurale, come mostrato nella Figura 4, è possibile distinguere tre parti:

- un layer di input;
- un numero di layer interni nascosti (*hidden*);
- un layer di output.

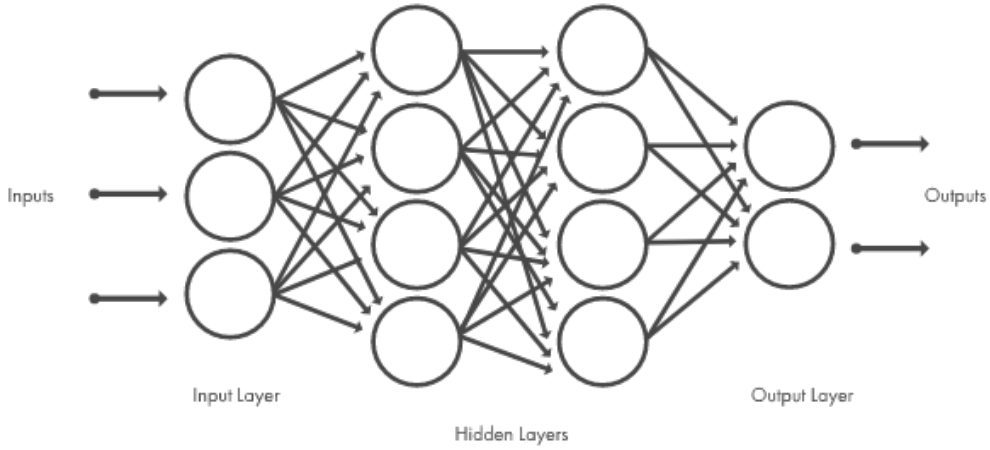

Figura 4: Schema di una generica rete neurale feed-forward tratto da [28].

Una ANN riceve i segnali in ingresso provenienti dal mondo esterno attraverso l'input layer e restituisce i risultati dell'elaborazione per mezzo di uno o più output layer, passando per un numero di layer nascosti. Questi strati possono essere parzialmente o completamente collegati.

Le reti neurali tradizionali contengono fino a 2-3 strati nascosti, mentre si parla di reti profonde o di *deep learning* quando si considerano reti aventi un gran numero di strati nascosti. Il termine "*deep*" fa per l'appunto riferimento al numero di livelli nascosti nelle reti neurali ed alcune delle più complesse reti profonde possono arrivare ad avere fino a 150 strati nascosti. I modelli di deep learning generalmente sono addestrati in maniera supervisionata su dataset di grandi dimensioni sfruttando hardware aventi un elevato grado di parallelismo, come GPU e Tensor Core, e pertanto capaci di ridurre notevolmente i tempi di elaborazione dei problemi di classificazione (da giorni a ore).

La tecnica per l'apprendimento supervisionato delle ANN più frequentemente impiegata è l'algoritmo di *backpropagation* (BP), che valuta il gradiente dell'errore della rete rispetto ai pesi e li aggiorna in modo tale da minimizzare l'errore di predizione relativo all'insieme di addestramento. Ad ogni esempio di addestramento fornito in ingresso alla rete i parametri del modello vengono aggiornati per convergere gradualmente verso il minimo. A seguito dell'addestramento la rete è in grado di riconoscere la relazione incognita che lega le variabili di ingresso a quelle di uscita e di effettuare predizioni, consentendo di risolvere problemi di

classificazione o di regressione. Tuttavia le ANN soffrono di minimi locali, pertanto l'apprendimento può richiedere molto tempo. Uno dei principali vantaggi delle ANN è che, con uno o più livelli nascosti, possono generare modelli non lineari in grado di catturare relazioni complesse tra gli attributi di input e le etichette di classificazione. Le ANN rappresentano quindi un potente strumento in molte attività di classificazione, inclusi gli IDS.

Un esempio di applicazione delle ANN nel rilevamento delle intrusioni è quello di Lippmann e Cunningham [1], i quali hanno proposto un sistema che sfrutta una selezione di keyword e le ANN per il rilevamento di anomalie nelle sessioni telnet. Tale sistema utilizza una rete neurale per effettuare la stima della probabilità a posteriori di un attacco e un'altra rete neurale per classificarlo. Esso ha ottenuto un tasso di rilevamento dell'80% con 1 falso allarme al giorno, con un miglioramento significativo rispetto agli approcci precedenti. Bivens et al. [1] descrivono invece un IDS che combina un metodo di clustering e le ANN. L'IDS da essi proposto utilizza una mappa auto-organizzante (SOM) per apprendere pattern di traffico normale, attraverso il processo di clustering, che vengono poi utilizzati per l'addestramento di un Multilayer Perceptron, che in potrà quindi essere utilizzato per effettuare predizioni. Il sistema può essere riavviato per apprendere nuovi modelli di traffico e addestrare un nuovo MLP. Lo studio ha utilizzato i dati TCP/IP della challenge DARPA 1999 ed è stato in grado di prevedere con successo il 100% del comportamento normale. Tuttavia non è stato in grado di riconoscere alcune tipologie di attacchi e ha prodotto per alcuni di essi un tasso di falsi allarmi pari al 76%. Taher et al. [7] hanno anch'essi impiegato una ANN per sviluppare un modello di rilevamento delle intrusioni basato su firme. I test sono stati effettuati sul dataset NSL-KDD ed in questi il modello basato su ANN ha ottenuto prestazioni superiori nella classificazione rispetto ad un modello analogo basato su SVM, producendo un tasso di rilevamento del 94,02%.

Le ANN vengono spesso utilizzate come elemento costitutivo di sistemi più complessi, come ad esempio proposto in [55].

2.2.2 Alberi di decisione

Un albero di decisione o Decision Tree (DT) è una struttura ad albero nella quale le foglie rappresentano classificazioni, i nodi rappresentano attributi, ed in base ai valori che questi

assumono permette di giungere ad una decisione o ad una classificazione. I successori di ciascun nodo sono tanti quanti sono i valori dell'attributo e per ciascuno di essi vi è un sottoalbero da considerare, fino ad arrivare alle foglie che rappresentano le etichette per ogni classe o la decisione finale come mostrato nella Figura 5.

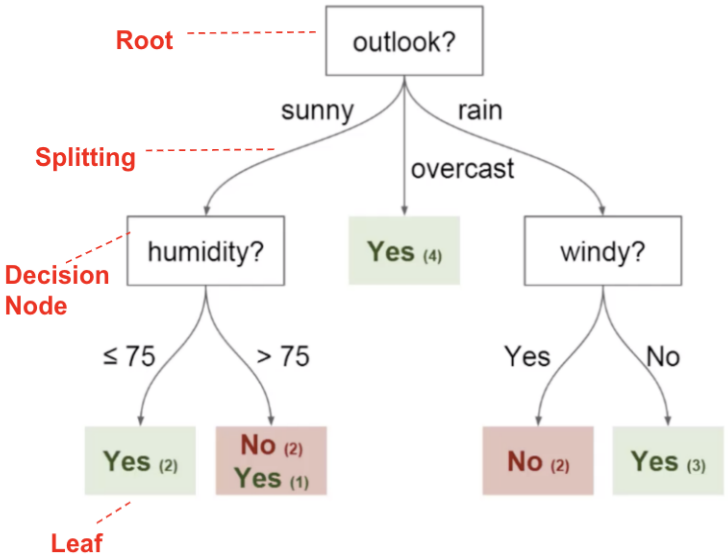

Figura 5: Schema di un generico albero di decisione tratto da [24].

Uno dei metodi più noti per costruire in maniera automatica alberi di decisione è l'algoritmo C4.5 [46]. Questo costruisce alberi di decisione a partire da un insieme di addestramento utilizzando il concetto di entropia dell'informazione.

L'algoritmo C4.5 seleziona in ogni nodo dell'albero l'attributo che suddivide più efficacemente l'insieme di esempi in sottoinsiemi. Il criterio di suddivisione impiegato è il guadagno di informazione o *information gain* (IG) normalizzato (*differenza di entropia*). L'idea alla base è quindi di selezionare ogni volta l'attributo migliore, ovvero quello che separa i dati nel miglior modo possibile nelle diverse classi esistenti e che permette di ottenere il maggior guadagno di informazione, di utilizzare tale attributo come radice dell'albero ottenuto dalla suddivisione del nodo corrente nei diversi risultati dell'attributo, e di applicare ricorsivamente tale processo a tutti i sottoalberi via via generati, fino al raggiungimento di un criterio di arresto. Alcuni dei criteri di arresto sono il raggiungimento

dell'altezza massima dell'albero, o se il miglior criterio di suddivisione non va oltre una certa soglia prestabilita in termini di guadagno di informazione.

Per ogni insieme di addestramento vi è un albero di decisione consistente, in grado di spiegare tutti gli esempi, ma l'obiettivo è trovare l'albero di decisione più semplice e compatto, in modo che questo non si sovradatti ai dati (*overfitting*) e che, se questi sono rappresentativi, abbia una buona capacità di generalizzare a partire dagli esempi. Scegliendo opportunamente gli attributi in modo da considerare prima quelli che permettono di classificare meglio i dati si ottiene un albero di decisione più semplice, compatto e meno profondo. Gli alberi più grandi hanno infatti un maggior grado di precisione nella classificazione ma presentano una scarsa capacità di generalizzazione.

È possibile ottenere alberi più piccoli attraverso una potatura (*pruning)* a partire da quelli più grandi. C4.5 utilizza la tecnica detta *error-based pruning*, con la quale un sottoalbero viene sostituito con una foglia se la percentuale stimata di errore risulta minore. Gli algoritmi come C4.5 sono più semplici e facili da implementare rispetto ad algoritmi più complessi come gli SVM. Uno dei principali vantaggi degli alberi di decisione è che sono di facile comprensione per gli esseri umani, in quanto la struttura ad albero permette di ottenere una rapida panoramica dei dati e di comprendere il perché di una determinata decisione (algoritmo di AI *explainable*).

Bilge et al. [47] [1] hanno introdotto un metodo di analisi passiva su larga scala di dati DNS, i quali sono stati raccolti ed utilizzati per generare un dataset di addestramento, ed un sistema di rilevamento, EXPOSURE, per rilevare in modo efficace ed efficiente i nomi di dominio coinvolti in attività malevole. Nel metodo proposto è stato utilizzato un classificatore di tipo DT, generato dal programma J48, che costituisce un'implementazione dell'algoritmo C4.5 in grado di generare alberi di decisione con o senza *pruning*. Per valutare l'accuratezza di tale sistema la classificazione dell'insieme di addestramento è stata effettuata tramite una validazione incrociata ed una suddivisione percentuale, nella quale il 66,66% dell'insieme di addestramento è stato utilizzato per l'addestramento e il restante 33,33% per verificare la correttezza. I risultati hanno restituito un tasso di rilevamento pari al 98.5% ed un tasso di falsi positivi dello 0.9%.

2.2.3 Random Forest

Le foreste casuali o Random Forest (RF) rappresentano una soluzione efficace al problema dell'*overfitting* sull'insieme di addestramento che si ha con gli alberi di decisione. Le RF sono una delle tecniche più utilizzate grazie alla loro accuratezza, semplicità e flessibilità, e possono essere impiegate per scopi di classificazione e regressione. La loro natura non lineare le rende altamente adattabili a una vasta gamma di dati e situazioni. Il nome deriva dalle "Random Decision Forests" proposte nel 1995 da Tin Kam Ho dei Bell Labs, che sviluppò un algoritmo per effettuare predizioni a partire da dati casuali. Queste furono estese nel 2006 da Leo Breiman e Adele Cutler fino a diventare ciò che sono oggi.

Una Random Forest è un metodo di apprendimento automatico supervisionato che combina gli alberi di decisione e l'apprendimento d'insieme. L'apprendimento d'insieme o *Ensemble Learning* si basa sull'utilizzo combinato di più algoritmi di apprendimento automatico per ottenere prestazioni predittive migliori rispetto ad un qualsiasi algoritmo applicato singolarmente.

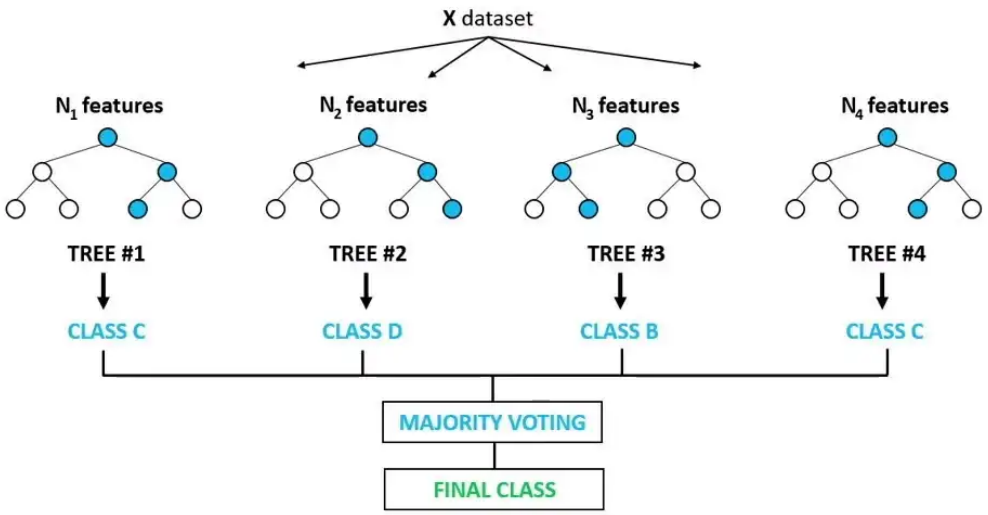

Figura 6: Schema del funzionamento di una generica Random Forest tratto da [25].

Uno dei principali metodi ensemble è il *bagging (o bootstrap aggregations)* e consiste nell'addestrare un classificatore su diversi sottoinsiemi dello stesso dataset (esempi *bootstrap*). Una RF è composta da molti alberi di decisione aggregati tramite bagging che

utilizzano come input attributi (*features*) selezionati casualmente, con l'obiettivo di ridurre la varianza. Il bagging permette ai singoli alberi di decisione di effettuare un campionamento casuale con sostituzione sull'insieme di addestramento, in modo tale da avere risultati differenti nei singoli alberi. Ogni albero quindi, invece che prendere in considerazione tutti i dati disponibili, ne include solo alcuni. I singoli alberi sono pertanto addestrati su insiemi di dati diversi ed utilizzano attributi diversi per prendere le decisioni. I dati di questi alberi vengono poi messi insieme per ottenere una predizione finale, più accurata, la quale può essere determinata tramite una votazione a maggioranza *(majority voting*) o ponderata (si veda la Figura 6).

Tra i vantaggi delle RF vi sono un ridotto numero di parametri del modello, la resistenza all'*overfitting*, un'alta efficienza con tutti i tipi di dati e tempi di addestramento brevi. Un altro importante vantaggio delle RF è che la varianza del modello diminuisce all'aumentare del numero di alberi nella foresta, mentre la distorsione rimane la stessa. Le RF presentano anche alcuni svantaggi, come la perdita di prestazioni dovuta alla presenza di variabili correlate e la dipendenza dal generatore casuale dell'implementazione. Un ulteriore svantaggio è dato dalla scarsa interpretabilità del modello, in quanto esse, a differenza dei DT, non sono spiegabili ma rappresentano dei modelli "black box", per cui è difficile capire come o perché arrivano a una certa decisione.

Nell'ambito del rilevamento di intrusioni l'applicazione delle Random Forest è stata descritta da Zhang et al. [1]. Nel loro lavoro essi hanno proposto l'utilizzo di due RF, una per l'*anomaly dectection*, l'altra per la *misuse detection*, e hanno descritto come implementare un sistema di rilevamento di anomalie attraverso l'utilizzo di una misura di prossimità all'interno della RF, in particolare la somma dei quadrati delle prossimità tra gli alberi e gli attributi. Tale sistema ottiene un tasso di rilevamento del 95% con un tasso di falsi allarmi pari all'1%. Un altro esempio è rappresentato dal lavoro di Bilge et al. [1], e dal sistema DISCLOSURE, il quale effettua il rilevamento di botnet Command and Control (C&C) utilizzando le RF su dati NetFlow. I risultati dei test del sistema su dati del mondo reale hanno restituito un tasso di rilevamento pari al 65% ed un tasso di falsi positivi dell'1%. Un altro studio in merito si deve a Jabbar et al. [2], i quali hanno proposto un classificatore Ensemble che combina RF e l'Average One-Dependence Estimator (AODE) al fine di aumentare la precisione nel rilevamento e di ridurre i falsi allarmi.

2.2.4 K-Nearest Neighbors

L'algoritmo K-Nearest Neighbors (K-NN) è uno dei più semplici algoritmi supervisionati di apprendimento automatico che può essere utilizzato per risolvere sia problemi di classificazione che di regressione. K-NN è basato sull'assunzione che dati simili siano distribuiti nelle immediate vicinanze, e cattura l'idea di somiglianza attraverso il calcolo della distanza. Esso costituisce un esempio di algoritmo di apprendimento basato sulle istanze o pigro (*lazy*), poiché non apprende immediatamente dall'insieme di addestramento, ma memorizza semplicemente tutti i dati disponibili (l'intero dataset) e al momento della classificazione esegue un'azione su di esso, classificando una nuova istanza, in base alla somiglianza, in una categoria molto simile. Tali metodi sono in contrasto con metodi di apprendimento quali gli alberi di decisione, che tentano invece di strutturare i dati prima di effettuare una classificazione.

In K-NN ciascuna istanza è rappresentata come un punto nello spazio n-dimensionale degli attributi. La Figura 7 mostra come avviene la classificazione di un punto in tale spazio: l'algoritmo procede calcolando le distanze tra un'istanza di test e tutti gli esempi nell'insieme di addestramento, selezionando poi i k esempi più vicini ad essa (*nearest neighbors*), ed assegnando un'etichetta di classe scelta in base ad una votazione di maggioranza (*majority voting*) che determina la più frequente, nel caso di una classificazione, o calcolando la media delle etichette, in caso di regressione.

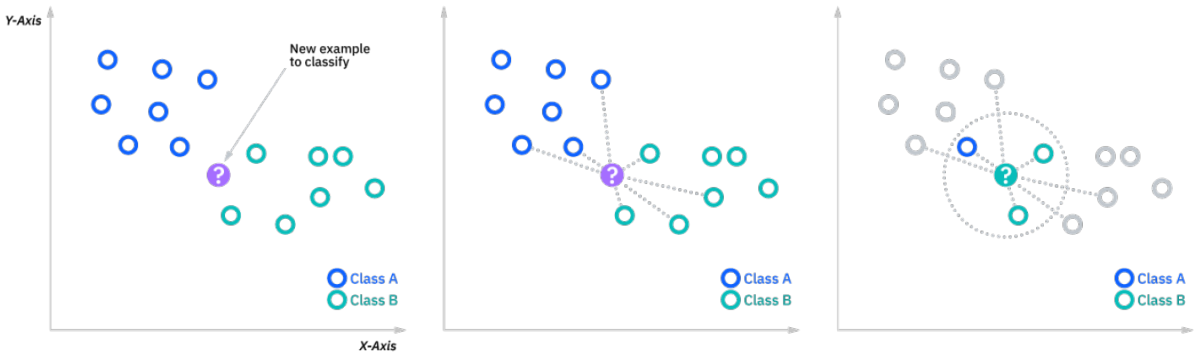

Figura 7: Schema del funzionamento dell'algoritmo K-NN tratto da [40].

La scelta del valore di k è molto importante, in quanto un valore troppo piccolo può portare all'*overfitting*, mentre un valore troppo grande può portare a classificazioni non corrette a causa dell'inclusione tra i vicini dell'istanza di test di punti distanti da essa.

È possibile migliorare la classificazione pesando il contributo di ciascun esempio secondo la sua distanza ed effettuando quindi una votazione di maggioranza pesata.

Per calcolare le distanze è comunemente utilizzata la distanza euclidea,

$$dist\_euclidean\ (A,B)\ =\ \sqrt{\sum_{i=1}^{r}\quad (x_i\ -\ y_i)^2\qquad},$$

sebbene sia possibile scegliere altre distanze in base all'applicazione. Oltre a quella euclidea, tra le più comuni vi sono:

$$dist\_Minkowsky\ (A,B)\ =\ (\sum_{i=1}^{m}\quad |x_i\ -\ y_i|^r\ )^{1/r},$$

$$dist\_correlation\ (A,B)\ =\ \frac{\sum_{i=1}^{m}\quad (x_i-\mu_i)\quad (y_i-\mu_i)}{\sqrt{\sum_{i=1}^{m}\quad (x_i-\mu_i)^2\ \sum_{i=1}^{m}\quad (y_i-\mu_i)^2}},$$

$$dist\_Chi\_square\ (A,B)\ =\ \sum_{i=1}^{m}\quad \frac{1}{sum_i}\left(\frac{x_i}{size_Q}\ -\frac{y_i}{size_I}\right)^2.$$

Tra i vantaggi dell'algoritmo vi è il fatto che con esso non è necessario costruire un modello, impostare diversi parametri (*algoritmo non parametrico*) o formulare ipotesi aggiuntive sui dati; K-NN è semplice da implementare e comprendere, ed è versatile in quanto può essere utilizzato per la classificazione e la regressione. K-NN presenta però alcuni svantaggi tra i quali che ha sempre bisogno di determinare il valore di k, il che potrebbe essere complesso in alcuni casi, e risulta suscettibile al rumore poiché la classificazione è effettuata localmente. Inoltre presenta un elevato costo computazionale a causa del calcolo delle distanze tra l'istanza di test e tutti i punti dell'insieme di addestramento, in tal caso si parla

infatti in letteratura del fenomeno "*curse of dimensionality*". K-NN può tuttavia essere opportunamente applicato come punto di riferimento per tutti gli altri classificatori poiché fornisce buone prestazioni di classificazione nella maggior parte degli IDS.

In particolare Wenchao Li et al. [48] hanno proposto un sistema di rilevamento delle intrusioni basato sull'algoritmo K-NN in grado di rilevare attacchi di tipo flooding in una rete di sensori wireless. L'algoritmo di rilevamento richiede due parametri: il valore di K, ed un valore di cutoff. Il primo corrisponde al numero dei vicini da includere nell'algoritmo K-NN mentre il secondo rappresenta un valore di soglia o threshold utilizzato per stabilire l'eventuale presenza di un'anomalia in un dato nodo. L'algoritmo di rilevamento K-NN identifica i nodi anomali confrontando il valore di una funzione K-distance (che effettua la somma delle distanze euclidee dei nodi più adiacenti divisa per K) ed il valore di cutoff di ciascun nodo. Il vettore di features che descrive un nodo è la frequenza dei messaggi RREQ: nell'attacco flooding i nodi anomali inviano messaggi RREQ più frequentemente rispetto ai nodi normali, pertanto questi possono essere identificati confrontando la frequenza di invio dei messaggi RREQ di ogni nodo della rete. Il sistema ottiene un tasso di rilevamento pari a circa il 98,5% ed un tasso di falsi allarmi del 4,63% con un cutoff pari a 10, mentre con un cutoff superiore a 20 presenta un tasso medio di rilevamento del 99,0% ed il tasso medio di falsi allarmi è dell'1,5%.

2.2.5 Algoritmi genetici

Gli algoritmi genetici (GA) sono tecniche euristiche di ottimizzazione combinatoria basate sul paradigma dell'evoluzionismo darwiniano, ovvero sul fatto che una data popolazione si evolve in modo tale da adattarsi ad un ambiente. Gli individui di questa popolazione sopravvivono meglio se si sono adattati all'ambiente e quindi trasmettono i loro caratteri genetici ai loro discendenti, in modo tale che nelle generazioni successive la popolazione risulti sempre più adattata all'ambiente. All'interno di un GA ogni individuo corrisponde ad una soluzione del problema dato, la quale è rappresentata con una stringa di bit (geni) detta cromosoma. Vi è poi una funzione di valutazione, o funzione di fitness, che associa ad ogni stringa un valore di fitness (o indice di adattamento) e che agisce in maniera analoga all'ambiente per le popolazioni, premiandole o meno in base alle loro caratteristiche.

Un GA, a partire da una popolazione generata casualmente, produce una nuova generazione attraverso operazioni di selezione, incrocio e mutazione. Nel processo di selezione sono selezionati per la riproduzione gli individui migliori, ovvero le stringhe con i più alti valori di fitness, successivamente viene applicata l'operazione di incrocio, che prevede lo scambio di geni (bit) tra una coppia di stringhe, in modo da generare nuove stringhe aventi la prima parte appartenente ad una stringa genitore e la seconda parte all'altra stringa. L'operazione successiva è la mutazione, ovvero il cambiamento spontaneo di parte del codice genetico: un bit all'interno di una stringa viene alterato in maniera casuale. Alla fine del processo gli individui con il più elevato valore di fitness sono copiati nella generazione successiva. Questo fa sì che statisticamente le soluzioni successive avranno un valore di fitness più alto e saranno in grado quindi di fornire prestazioni migliori. Un'applicazione dei GA nel contesto degli IDS è stata proposta da Khan [1]: a partire da due sottoinsiemi di 10000 istanze del dataset KDD 1999, utilizzate per le fasi di addestramento e di test, viene effettuata una Principal Component Analysis (PCA) al fine di ridurre il numero di attributi, e sono applicati gli algoritmi genetici per evolvere un set di regole per il rilevamento di intrusioni. Dalla popolazione finale ottenuta viene prelevata una regola per classificare le istanze come normali o attacchi. Il sistema restituisce un accuracy del 93,45% ed un tasso di falsi positivi del 10,8% per la classe normale, mentre il 94,19% e il 2,75% per la classe di attacchi.

2.2.6 Naïve Bayes

I classificatori Naïve Bayes (NB) applicano un modello che esprime in forma probabilistica la relazione tra gli attributi e le etichette di classe tramite l'impiego del *teorema di Bayes*. Questo esprime la relazione tra le probabilità condizionate di due eventi nel seguente modo:

$$P(A \mid B) = \frac{P(B \mid A)\,P(A)}{P(B)},$$

in cui *P(A)* e *P(B)* sono le probabilità degli eventi A e B, *P(A | B)* è la probabilità condizionata dell'evento A dato l'evento B e *P(B | A)* è la probabilità condizionata dell'evento B dato l'evento A.

Non esiste un singolo algoritmo per l'addestramento di tali classificatori, ma una famiglia di algoritmi basati sulla comune ipotesi "ingenua" (*naïve*) che gli attributi di un'istanza di input

del problema da classificare siano indipendenti tra loro, mentre nella pratica ciò è raramente vero. Un classificatore NB è costruito a partire da un modello il quale, data un'istanza X in ingresso, corrispondente ad un vettore di n *features* (attributi) x = $(x_1, x_2, \ldots, x_n)$, calcola la probabilità condizionata della classe C data l'istanza X applicando il teorema di Bayes come segue:

$$P(C \mid x_1, \ldots, x_n) = \frac{P(x_1,\ldots,x_n \mid C)\, P(C)}{P(x_1,\ldots,x_n)};$$

questo viene effettuato per ciascuna delle classi e successivamente viene scelta la classe con la probabilità più alta.

In tale calcolo:

- è possibile trascurare $P(x_1, \ldots, x_n)$ poiché esso è costante per tutte le classi;
- $P(C)$ rappresenta la probabilità priori della classe C pertanto è calcolabile attraverso il rapporto Nc/N tra il numero di record dell'insieme di addestramento aventi quell'etichetta di classe e il numero totale dei record;
- È possibile semplificare il calcolo di $P(x_1, \ldots, x_n \mid C)$ attraverso l'ipotesi *naïve* dell'indipendenza degli attributi $x_1, \ldots, x_n$.

Si ottiene quindi:

$$P(x_1,\ldots,x_n \mid C) = P(x_1 \mid C) \cdot P(x_2 \mid C) \cdot \ldots \cdot (x_n \mid C),$$

ed il modello del classificatore pertanto assegna ad un'etichetta di classe y^* una classe C_k tramite la seguente equazione:

$$y^* = argmax_{k \in \{1,\ldots,K\}} P(C_k) \prod_{i=1}^{n} P(x_i \mid C_k),$$

dove n è il numero di attributi, x_i è l'i-esimo attributo, C_k è la k-esima classe, $P(C_k)$ è la probabilità a priori di C_k, e $P(x_i \mid C_k)$ è la probabilità condizionata dell'attributo x_i data la classe C_k.

Un classificatore NB può essere quindi facilmente addestrato utilizzando una struttura di apprendimento supervisionata. In molte applicazioni pratiche, per effettuare la stima dei parametri i modelli NB utilizzano il metodo della massima verosimiglianza (*maximum likelihood*).

Tra i vantaggi dei classificatori NB essi sono altamente scalabili, richiedono per l'apprendimento un numero di parametri lineare nel numero di attributi, il loro addestramento può essere completato in tempo lineare, piuttosto che attraverso una costosa approssimazione iterativa, utilizzata per molti altri tipi di classificatori. In generale offrono buone prestazioni in molte complesse situazioni del mondo reale e sono tra i classificatori che vengono maggiormente utilizzati grazie alla loro facilità d'uso ed efficienza computazionale e confrontati con gli algoritmi più sofisticati come RF, ANN e SVM, sebbene restituiscano in genere risultati peggiori rispetto ad essi. Uno dei maggiori svantaggi è che in presenza di attributi significativamente correlati l'efficacia del modello può ridursi drasticamente, come dimostrato nel lavoro di Koc et al. [2]. In tale studio è stato inoltre evidenziato come, nel contesto degli IDS, un metodo NB applicato a dataset di grandi dimensioni ottenga un'accuracy ridotta. Essi hanno quindi proposto il più sofisticato metodo Hidden Naïve Bayes (HNB) per effettuare il rilevamento in presenza di dataset di tali dimensioni, ed aventi un alto grado di correlazione tra gli attributi. Un altro esempio è rappresentato dal lavoro di Amor et al. [1], nel quale una rete bayesiana, nella forma semplice nodo radice-foglie equivalente ad un metodo NB, è stata utilizzata per classificare diverse tipologie di attacchi sul dataset KDD 1999. Il metodo proposto ha ottenuto un'accuracy del 98% e dell'89% per le classi normali e anomale rispettivamente.

2.2.7 Support Vector Machines

L'algoritmo delle Support Vector Machine (SVM) fa parte dei metodi supervisionati di apprendimento automatico e può essere impiegato per effettuare tecniche di regressione e di classificazione. Esso mira a suddividere i dati determinando un iperpiano di separazione nello spazio delle features tra due classi in modo tale da massimizzare la distanza tra l'iperpiano e i punti più vicini di ciascuna classe, che prendono il nome di *support vectors*.

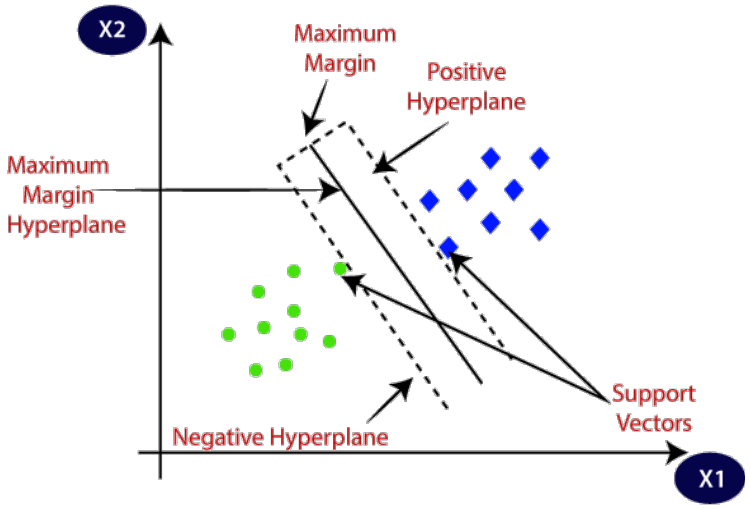

Figura 8: Grafico che mostra il funzionamento di una generica SVM tratto da [42].

Dal punto di vista formale, una SVM crea un iperpiano di separazione in uno spazio n-dimensionale. La Figura 8 mostra come avviene la suddivisione dei dati in tale spazio.

Il primo passo dell'algoritmo è cercare di separare linearmente i dati nelle etichette di classe corrispondenti e viene utilizzato il margine attorno a una funzione di separazione come parametro aggiuntivo per valutare la qualità della separazione. Se i dati non sono separabili linearmente, ovvero se l'iperpiano di separazione non è lineare come spesso accade per i dati del mondo reale, le SVM utilizzano un kernel per mappare i dati di addestramento in uno spazio di dimensioni maggiori nel quale essi possono essere separati linearmente. A tale scopo è possibile ottenere diversi tipi di iperpiani di separazione applicando differenti kernel, in particolare di tipo lineare, polinomiale, con una funzione di base radiale (RBF) o tangente iperbolica. I classificatori basati sulle SVM sono classificatori binari e la classificazione multiclasse viene effettuata attraverso l'utilizzo di una SVM per ogni coppia di classi. Le SVM presentano diversi vantaggi, tra i quali il fatto che esse offrono una buona capacità di generalizzazione, sono particolarmente utili quando il numero di features è grande, il numero di punti dati è piccolo e l'algoritmo risulta tra i più veloci sotto queste condizioni, inoltre, come detto, esse sono in grado di risolvere problemi anche non lineari. Tra i principali svantaggi vi è che la creazione del modello può non avvenire in tempi brevi e che, sebbene in certi casi restituiscano un grado di accuratezza paragonabile a quello degli alberi di decisione, al contrario di questi ultimi le SVM non sono spiegabili. I metodi basati su SVM

sono tra i più comunemente utilizzati in letteratura per il rilevamento di anomalie. Hu et al. [1] hanno impiegato per la classificazione delle anomalie sul dataset DARPA 1998 una Robust Support Vector Machine (RSVM), ovvero una variante delle SVM in cui l'iperpiano di separazione è mediato per essere più uniforme ed il parametro di regolarizzazione è determinato in modo automatico. I risultati dello studio hanno mostrato buone prestazioni di classificazione in presenza di rumore, simulato con un'errata etichettatura dell'insieme di addestramento, e ha raggiunto un'accuracy del 75% senza falsi allarmi ed un'accuracy del 100% con un FAR del 3%. Wagner et al. [1] hanno utilizzato un classificatore SVM One-class su dati Netflow del mondo reale e su dati di attacchi simulati tramite il Flame tool, introducendo un nuovo tipo di kernel che è utilizzato come misura di similarità tra i record sequenziali Netflow. Il sistema ottiene tassi di rilevamento tra l'89% ed il 94% con un tasso di falsi positivi tra lo 0% ed il 3%. Un altro studio, condotto da Shon e Moon [1], combina gli algoritmi Self-Organizing-Map (SOM), GA, e SVM allo scopo di rilevare nuovi attacchi nel traffico di rete. In tale approccio una SOM viene utilizzata per realizzare un profiling dei pacchetti, la selezione delle features è effettuata tramite GA, mentre la classificazione avviene attraverso SVM. In particolare sono state utilizzate due tipologie di SVM: la prima, la Enhanced SVM, è derivata da una SVM one-class per la classificazione di dati non etichettati, mentre la seconda, la supervised soft-margin SVM, garantisce elevate prestazioni nel rilevamento in modo supervisionato ma non è in grado di rilevare nuovi attacchi. Nel presente lavoro è stato utilizzato il dataset DARPA 1999, selezionando un sottoinsieme molto ridotto di attacchi (circa l'1%) rispetto al traffico normale.

2.3 Metodi di apprendimento non supervisionato

Gran parte degli AIDS non supervisionati utilizzano tecniche di clustering per identificare le istanze potenzialmente dannose in un determinato dataset.

L'obiettivo del clustering è partizionare un dataset contenente dati non etichettati in un insieme finito e discreto di strutture dati nascoste o classi dette *cluster*, aventi un elevato grado di somiglianza interna e contenenti pattern simili, senza alcuna conoscenza pregressa ma esclusivamente in base alle relazioni tra di loro.

È possibile distinguere diversi approcci nel clustering. Questi sono principalmente:

- *Connectivity-based:* i cluster sono generati connettendo gli oggetti in base alla loro distanza. Tale approccio si basa sull'idea che gli oggetti siano più correlati agli oggetti vicini che a quelli più lontani. Questi algoritmi non forniscono un unico partizionamento del dataset in input, ma forniscono un'ampia gerarchia di cluster che si fondono tra loro a determinate distanze;
- *Centroid-based:* consiste nel generare i cluster a partire dal calcolo dei centroidi effettuato sul dataset in ingresso;
- *Distribution-based:* è il modello di clustering più strettamente correlato alla statistica. I cluster sono definiti come oggetti appartenenti alla stessa distribuzione con un'alta probabilità;
- *Density-based:* in questo approccio i cluster rappresentano aree di maggiore densità rispetto al resto del dataset. Vengono considerati come punti di rumore e di confine gli oggetti in aree sparse, necessari per separare i cluster.

Tra le tecniche di tipo non supervisionato più utilizzate vi sono quelle basate su:

- *Clustering gerarchico;*
- *K-means;*
- *Expectation-Maximization;*
- *Self-Organizing Maps;*
- *DBSCAN.*

Nel seguito vengono descritti ed analizzati i principali metodi per ciascun approccio di clustering elencato.

2.3.1 Clustering gerarchico

Il clustering gerarchico è una tecnica che mira a creare una gerarchia di cluster. Gli approcci per il clustering gerarchico sono normalmente classificati in due categorie:

- *Agglomerativi:* algoritmi bottom-up di clustering in cui i cluster hanno sottocluster, che a loro volta hanno sottocluster e coppie di cluster vengono combinate man mano che si sale nella gerarchia;

- *Divisivi:* algoritmi di clustering gerarchico in cui iterativamente il cluster con il diametro maggiore nello spazio delle features viene selezionato e diviso in sottocluster di diametro inferiore.

Per stabilire quali cluster devono essere combinati (approccio agglomerativo) o quale cluster deve essere suddiviso (approccio divisivo) è necessario definire una misura di dissimilarità tra cluster. Questa può essere definita attraverso l'utilizzo di diverse metriche che quantificano la distanza tra coppie di elementi e di un criterio di collegamento che specifica la dissimilarità di due cluster come funzione della distanza a coppie tra elementi nei due insiemi. La scelta di una metrica appropriata incide sulla forma dei cluster, poiché alcuni elementi possono essere più vicini utilizzando una distanza e più lontani utilizzandone un'altra. In genere si utilizzano come metriche:

- *distanza euclidea;*
- *distanza di Manhattan;*
- *distanza di Mahalanobis;*
- *distanza di Hamming.*

Il criterio di collegamento (*linkage criterion*) specifica la distanza tra i cluster come funzione delle distanze tra gli elementi degli insiemi. Poiché un cluster è costituito da più elementi, ci sono diversi modi per calcolare la distanza da utilizzare. Le scelte più diffuse per tale criterio sono:

- *collegamento singolo:* viene scelto il minimo delle distanze;
- *collegamento completo:* viene scelto il massimo delle distanze;
- *collegamento medio*: viene scelta la media aritmetica o ponderata delle distanze.

I metodi di clustering gerarchico non sono molto robusti nei confronti dei valori anomali, che si presenteranno come cluster aggiuntivi o causeranno l'unione di altri cluster (*fenomeno del concatenamento*).

Un esempio di applicazione di clustering gerarchico nel rilevamento delle intrusioni di rete è rappresentato dal lavoro di Horng et al. [49]. In tale studio, effettuato sul dataset KDD 1999, viene proposto un IDS che combina un algoritmo di clustering gerarchico BIRCH (Balanced Iterative Reduction and Clustering using Hierarchies), comunemente utilizzato per eseguire il clustering su dataset di grandi dimensioni, ed un classificatore di tipo SVM.

Le prestazioni del sistema risultano strettamente legate all'algoritmo di clustering, e alla sua capacità di fornire al classificatore punti di alta qualità, ossia di fare in modo che i punti astratti nel dataset di dimensioni ridotte ottenuto rappresentino tutti i data points nel dataset originale. L'algoritmo BIRCH seleziona come features a partire dal dataset KDD il valore delle entry iniziali, la loro somma lineare (LS) e quadratica (SS) e costruisce dei Clustering Feature (CF) tree a partire da esse in base a 2 parametri, un fattore di branching e un valore di threshold. Effettuando la scansione di tutte le entry dei nodi foglia nel CF iniziale viene costruito un CF tree più piccolo, rimuovendo i valori anomali e raggruppando i sottocluster nei cluster più grandi. Nel metodo da essi proposto vengono costruiti quattro alberi CF per le classi di attacco DoS, U2R, R2L e Probe ed un albero CF per i pacchetti normali e viene effettuata una feature selection per ciascun tipo di attacco. Successivamente vengono addestrati quattro classificatori SVM separatamente, corrispondenti ai quattro tipi di attacchi, in base al centroide di tutte le entry nei nodi foglia degli alberi CF, i quali sono poi combinati per creare un sistema di rilevamento delle intrusioni. Rispetto ad altri IDS utilizzati sul medesimo dataset, nel complesso il sistema ha ottenuto prestazioni migliori nella precisione del rilevamento ed in particolare nella detection di attacchi di tipo DoS e Probe. I risultati dei test del sistema hanno raggiunto un'accuracy del 95,72% con un tasso di falsi positivi dello 0,73%.

2.3.2 K-means

L'algoritmo K-means è tra gli algoritmi di apprendimento non supervisionati più semplici. È una tecnica di clustering basata sulla distanza e non necessita di calcolare le distanze tra tutte le combinazioni di record in un dataset. Esso applica come criterio di somiglianza la metrica euclidea e separa n istanze in ingresso in un numero di cluster k, specificato in anticipo dall'utente, in modo tale che ogni istanza venga assegnata al cluster che ha la media (centroide) più vicina.

Dato quindi un insieme di istanze $x = (x_1, x_2, \ldots, x_n)$, nel quale ogni istanza è un vettore reale d-dimensionale, K-means partiziona le n istanze $\texttt{in k (} \leq \texttt{ n) insiemi S} = \{S_1, S_2, \ldots, S_k\}$ per minimizzare la varianza intra-cluster.

Formalmente, K-means è definito come:

$$argmin_S \sum_{i=1}^{k} \sum_{x \in S_i} \left||x - \mu_i|\right|^2 = argmin_S \sum_{i=1}^{k} |S_i| \, Var \, S_i \ .$$

Dato un insieme iniziale di k centroidi selezionati casualmente, l'algoritmo procede iterativamente alternando 2 step:

- *Step di assegnazione:* assegna ciascuna istanza al cluster il cui centroide ha la distanza euclidea minima;
- *Step di aggiornamento:* ricalcola i centroidi per le istanze assegnate a ciascun cluster.

L'algoritmo converge quando le assegnazioni non cambiano più. K-means rappresenta quindi un problema di ottimizzazione: esso determina i centroidi e assegna gli elementi al cluster il cui centroide è più vicino, ossia la cui distanza al quadrato dal cluster è minima. Tale problema è noto per essere *NP-hard*, e quindi l'approccio comune è cercare solo soluzioni approssimate. Un metodo approssimato particolarmente noto è l'*algoritmo di Lloyd*, spesso indicato semplicemente come "algoritmo K-means", tuttavia esso trova solo un ottimo locale e viene comunemente eseguito più volte con diverse inizializzazioni casuali. Variazioni dell'algoritmo K-means spesso includono ottimizzazioni come la scelta della migliore di più esecuzioni, la scelta dei centri iniziali in modo meno casuale (*K-means++*), ma anche la restrizione dei centroidi ai membri del dataset (*K-medoid*), o il consentire un'assegnazione di cluster fuzzy (*Fuzzy c-means*). Tra i principali svantaggi, esso richiede che il numero di cluster k sia specificato in anticipo dall'utente.

Nello studio di Peng et al. [50] viene proposto un metodo di clustering per gli IDS che utilizza una versione migliorata dell'algoritmo K-means insieme ad un'analisi dei componenti principali (PCA) sul dataset KDDCUP99. In tale studio vengono testati una porzione pari a circa al 10% del dataset e l'intero dataset. Allo scopo di migliorare l'efficienza del clustering il dataset iniziale viene normalizzato e viene eseguita una PCA, la quale consente inoltre di ridurre le dimensioni dei dati. Il metodo proposto quindi utilizza una versione dell'algoritmo K-means, detta Mini Batch K-means, per effettuare per il clustering dei dati inizializzando i centroidi dei cluster tramite l'opzione "K-means++", in modo da evitare che l'algoritmo ricada in ottimi locali. A differenza dell'algoritmo K-means, questo non utilizza ogni volta tutti i record nel dataset, ma ne seleziona un sottoinsieme in modo casuale ed in tal modo riduce notevolmente i tempi di elaborazione. I risultati sperimentali e l'analisi della complessità temporale comparati con le tecniche tradizionali mostrano che il

metodo è efficace ed efficiente, e che il clustering proposto può essere applicato per il rilevamento di intrusioni in presenza di big data.

2.3.3 Expectation-Maximization

Un metodo di clustering basato su un modello statistico si basa sull'assunzione che i dati sono generati da una composizione di distribuzioni di probabilità. In genere il clustering viene effettuato utilizzando un *mixture density model* di k distribuzioni di probabilità, nel quale ognuna delle distribuzioni rappresenta un cluster.

L'algoritmo Expectation-Maximization (EM) è un noto metodo statistico iterativo per determinare la massima verosimiglianza (*maximum likelihood*) locale o le stime massime a posteriori (MAP) dei parametri. Molto simile a k-means, è possibile pensare ad EM come ad un'estensione di quest'ultimo nei seguenti due modi: assegna ciascun elemento ad un cluster secondo un peso, calcolato attraverso una o più distribuzioni di probabilità, che rappresenta la probabilità di appartenenza, e massimizza l'intera probabilità dei dati, dati i cluster finali. EM effettua una stima iniziale (*guess*) dei parametri del modello attraverso la selezione randomica di k elementi per rappresentare le medie o centroidi dei cluster, procede riassegnando iterativamente gli elementi in base al modello, ed utilizza quest'ultimi per aggiornare le stime dei parametri.

Come suggerito dal nome, l'algoritmo consiste nell'alternare iterativamente le seguenti due fasi:

- *Expectation (E):* calcola la probabilità di appartenenza di ciascun elemento al cluster i-esimo, per ciascuno dei cluster, a partire dalla distribuzione di probabilità corrente dei parametri del modello;
- *Maximization (M):* questo passo rappresenta la massimizzazione della probabilità delle distribuzioni rispetto ai dati: i parametri del modello sono ricalcolati utilizzando le stime di probabilità precedenti.

L'algoritmo procede fino a quando i parametri del modello non convergono o si raggiunge un numero massimo di iterazioni. La convergenza avviene in genere in tempi rapidi, tuttavia non è garantito il raggiungimento del massimo globale. Tra i vantaggi dell'algoritmo EM

esso è semplice e facile da implementare, ha una complessità computazionale lineare rispetto al numero di caratteristiche dei dati di input ed al numero di elementi.

Un esempio di applicazione negli IDS è dato dal lavoro di Ghimire et al. [51], i quali propongono un nuovo modello ibrido che utilizza l'algoritmo EM, tramite un modello di miscela gaussiana (GMM), ed un classificatore Naïve Bayes sul dataset NSL-KDD. In un GMM, è possibile definire la forma del cluster attraverso due parametri, la media e la deviazione standard, utilizzando i quali i cluster possono assumere qualsiasi tipo di forma ellittica. Il sistema effettua preliminarmente una fase di preprocessing dei dati, la quale include la codifica dei dati e la feature selection. Viene poi applicato il clustering tramite l'algoritmo EM, il quale raggruppa i dati nel numero appropriato di cluster. Successivamente i dati raggruppati con valori anomali vengono quindi classificati utilizzando il classificatore Naïve Bayes e sono inviati ad un modulo di Decision Making. Dal confronto dei risultati con altri modelli emerge che l'algoritmo EM (GMM) con Naïve Bayes produce un tasso di rilevamento notevolmente migliore per gli attacchi a bassa frequenza, come R2L e U2R. Inoltre, le prestazioni per Probe sono migliori rispetto ad altri modelli, ma le prestazioni della classe DoS sono superiori in altri sistemi di rilevamento delle intrusioni.

I. Mukhopadhyay et al. [52] hanno proposto un algoritmo di rilevamento chiamato EMID in grado di identificare tentativi di autenticazione fraudolenti utilizzando tecniche statistiche per la stima dei parametri e il clustering. L'algoritmo EM è qui utilizzato per stimare i parametri corrispondenti agli utenti fraudolenti e legittimi durante la fase di addestramento, ed un classificatore bayesiano è utilizzato in seguito per classificare i parametri stimati, al fine di stabilire se un individuo che tenta di accedere al sistema sia effettivamente un utente autentico o un utente fraudolento. I risultati della simulazione dell'implementazione di EMID in MATLAB hanno dimostrato la sua efficacia nel rilevamento di intrusioni corrispondenti a tentativi di autenticazione fraudolenti.

2.3.4 Self-Organizing Maps

Una mappa auto-organizzante o *Self Organizing Map* (SOM) è un algoritmo di apprendimento automatico non supervisionato basato sulle reti neurali artificiali utilizzato per la riduzione della dimensionalità dei dati in ingresso (*input space*) generando una loro rappresentazione in uno spazio di dimensione inferiore (*map space*), in genere

bidimensionale. Il modello fu descritto per primo da Teuvo Kohonen, per cui esse sono anche note come *mappe di Kohonen*.

L'algoritmo SOM si basa sull'idea di realizzare una rappresentazione topologica dei dati, in modo tale che dati simili si trovino vicini all'interno del *map space* e che le proprietà dei dati siano mantenute, con un certo grado di approssimazione, anche dopo la riduzione della dimensionalità.

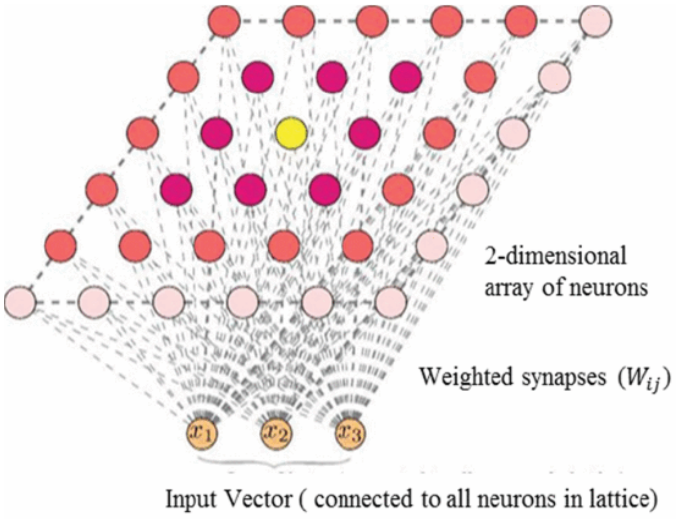

Figura 9: Struttura di una generica SOM tratto da [4].

Un metodo SOM utilizza una rete neurale nella quale ogni ingresso è connesso a tutti i neuroni di uscita. Ad ogni nodo (o neurone) del *map space* è associato un vettore dei pesi della stessa dimensione dei vettori d'ingresso (si veda la Figura 9), che rappresenta la posizione del nodo nell'*input space*. Il numero di nodi e la loro disposizione sono specificati in anticipo dall'utente come parametri, ed i vettori dei pesi sono inizializzati con valori casuali.

L'idea alla base consiste nel cercare di "spostare" i vettori dei pesi associati ai nodi verso i dati di input, riducendo una metrica di distanza come quella euclidea in maniera tale da non alterare la topologia indotta dal *map space.*

Durante la fase di addestramento viene effettuato il *mapping* dei vettori di input in una griglia bidimensionale di nodi, viene quindi costruita la mappa e la rete neurale si organizza attraverso un processo competitivo: per ogni esempio di addestramento presentato in

ingresso al modello viene calcolata la distanza euclidea tra quest'ultimo e tutti i vettori dei pesi. In seguito viene determinato il nodo il cui vettore peso associato è il più vicino al vettore dell'*input space* in base alla distanza euclidea, il quale è detto *Best Matching Unit* (*BMU*). I pesi del BMU e dei neuroni vicini ad esso all'interno della griglia SOM vengono aggiornati in modo da essere avvicinati al vettore d'ingresso. La formula utilizzata per l'aggiornamento dei pesi è la *legge di apprendimento Kohonen,* la quale esprime la seguente relazione:

$$W_v(t\ +\ 1)\ =\ W_v(t)\ +\ \alpha\ h_{vm}(\ X\ -\ W_v(t)),$$

nella quale $W_v(t + 1)$ rappresenta il nuovo vettore dei pesi del nodo v, $W_v(t)$ il precedente vettore dei pesi del nodo v, α è un coefficiente di apprendimento monotono decrescente, h_{vm} è una funzione che definisce l'insieme dei vicini del nodo v ed X è il vettore di ingresso.

La funzione di vicinato (*neighbourhood)* h_{vm} dipende dalla distanza nel reticolo fra il BMU e il neurone v, in genere corrisponde ad una gaussiana, e decresce nel tempo. All'inizio quando l'insieme dei vicini è ampio, vi è un'auto-organizzazione che avviene su scala globale (*ordering phase*), successivamente quando rimangono solo pochi neuroni i pesi convergono in una stima locale (*tuning phase*). Attraverso il processo di addestramento quindi nodi vicini tenderanno a specializzarsi nel riconoscere ingressi simili. Come accade con qualsiasi tecnica che utilizza le reti neurali i dati in ingresso vengono presentati molte volte al modello, in modo tale che in seguito all'addestramento la mappa può essere utilizzata per classificare nuovi vettori d'ingresso.

Wang et al. [53] hanno proposto un IDS Anomaly-based nel quale tre SOM sono state addestrate a partire dai dati di addestramento campionati e preelaborati sui livelli di sistema, di processo e di rete di tre host in una LAN. Il dataset di addestramento comprende dati rilevanti che simulano operazioni normali quali l'utilizzo di applicazioni, la navigazione web, fpt e telnet etc. Nel metodo proposto una funzione, che impiega la distanza euclidea come metrica per valutare la distanza tra una nuova istanza ed un dato cluster, è utilizzata per determinare se essa è anomala o no. Sebbene le ipotesi sperimentali siano più semplici di quelle reali, l'analisi dei risultati del monitoraggio sui tre livelli degli strumenti di hacking NMAP, HYDRA dimostra l'efficacia del metodo di rilevamento proposto.

2.3.5 DBSCAN

L'algoritmo DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*) è il metodo di clustering basato sulla densità più diffuso e tra i più citati in letteratura. Simile al clustering basato sul collegamento, DBSCAN è una tecnica che si basa sulla connessione dei punti che soddisfano un criterio di densità, definito come un numero minimo di punti che ricadono all'interno di un cerchio di un raggio dato. Un cluster è costituito da elementi connessi in base alla densità ed in base ad essa può assumere diverse forme. DBSCAN è pertanto in grado di determinare cluster di forma arbitraria ed in presenza di valori rumorosi (valori anomali), a differenza di molti altri metodi di clustering. L'idea principale alla base di DBSCAN quindi è che un punto appartiene a un cluster se è vicino a molti punti di quel cluster.

I due parametri chiave dell'algoritmo DBSCAN sono:

- *eps*: rappresenta la distanza che determina la vicinanza. Due punti sono considerati vicini se la distanza tra loro è minore o uguale a *eps*.
- *minPts*: costituisce il numero minimo di punti per definire un cluster.

Sulla base di questi due parametri, i punti vengono classificati come:

- *core point*: un punto è un *core point* se ci sono almeno *minPts* punti (incluso il punto stesso) nell'area circostante di raggio *eps;*
- *border point*: un punto è un *border point* se è raggiungibile da un *core point*, cioè si trova nell'area circostante al *core point*, e se il numero di punti presenti all'interno di essa è inferiore a *minPts*;
- *outlier*: un punto è un *outlier* se non è un *core point* e non è raggiungibile da alcun *core point*.

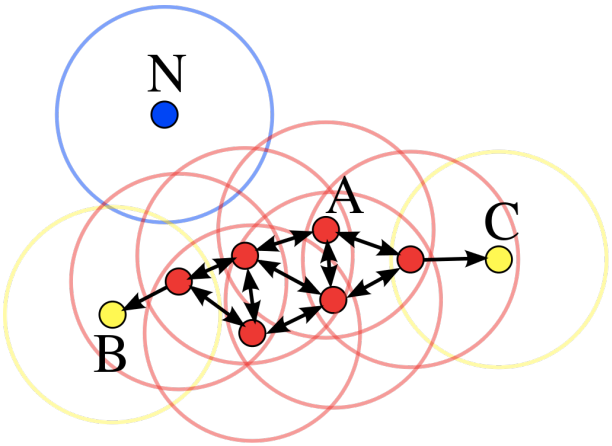

Figura 10: Schema di esempio che mostra come DBSCAN classifica i diversi punti tratto da [41].

La Figura 10 mostra un esempio per ciascuno di questi punti. In questo caso, il valore di *minPts* è 4. I punti rossi sono *core point* perché ci sono almeno 4 punti nell'area circostante con raggio *eps*. Quest'area è mostrata con i cerchi nella figura. I punti gialli sono *border points* perché si trovano nell'area circostante un *core point* e hanno meno di 4 punti all'interno del loro vicinato. I punti B e C hanno due punti (compreso il punto stesso) all'interno del loro vicinato (cioè l'area circostante con un raggio di *eps*). Infine N rappresenta un *outlier* poiché esso non è un *core point* e non può essere raggiunto da un *core point*.

I parametri *minPts* ed *eps* sono forniti in ingresso all'algoritmo come parametri. L'algoritmo DBSCAN inizia quindi selezionando casualmente un punto di partenza del dataset e determinando il suo insieme dei vicini (*eps*-vicinato) di raggio *eps*. Per valutare la distanza tra i punti viene utilizzata la distanza euclidea come nell'algoritmo K-means.

Se l'*eps*-vicinato contiene almeno *minPts* punti, il punto viene contrassegnato come *core point* e viene creato un nuovo cluster, mentre se ciò non avviene il punto viene etichettato come rumore e successivamente potrebbe entrare a far parte di un cluster. Se un punto è associato ad un cluster tutti i punti all'interno del suo *eps*-vicinato sono aggiunti al cluster. Se questi nuovi punti sono anche *core points*, vengono aggiunti al cluster anche i punti che si trovano nei loro *eps*-vicinati. Questo processo continua fino a quando il cluster viene completato. Lo step successivo consiste nel selezionare casualmente un altro punto tra quelli non ancora visitati ed applicare la stessa procedura. L'algoritmo termina quando tutti i punti

sono stati visitati. Attraverso tali step, DBSCAN è in grado di trovare regioni ad alta densità e separarle da regioni a bassa densità.

DBSCAN rispetto ad altri algoritmi di clustering non richiede la specifica a priori di un numero fisso di cluster, identifica inoltre i valori anomali come rumore, ed è in grado di determinare cluster di dimensioni e forma arbitraria, in particolare quelli non linearmente separabili non rilevabili con altri algoritmi di clustering (per esempio K-means, EM). Lo svantaggio principale di DBSCAN è che non può raggruppare efficacemente insiemi di dati che presentano grandi differenze in termini di densità, poiché in tal caso la combinazione *minPts-eps* non può essere scelta in modo appropriato per tutti i cluster. Questo inconveniente si verifica in particolare con dataset di dimensioni molto elevate. DBSCAN visita infatti ogni punto del dataset anche più volte nel caso di punti candidati a cluster differenti. Esso esegue poi il calcolo dei vicini per ogni punto e, se viene utilizzata una struttura indicizzata che permette un'esecuzione in O(log n), si ottiene un tempo globale di esecuzione pari a O(n log n). Senza l'uso di strutture indicizzate o in presenza di dati degenerati (ad esempio se tutti i punti si trovano entro una distanza inferiore ad *eps*), la complessità del tempo di esecuzione nel caso peggiore è pari a $O(n^2)$.

Per evitare il ricalcolo delle distanze è possibile calcolare la matrice delle distanze e mantenere in memoria solo la sua triangolare superiore (di dimensioni $(n^2 - n)/2$). Ciò richiede quindi $O(n^2)$, mentre un'implementazione di DBSCAN non basata su matrice richiede solo O(n). Un modo per ridurre la complessità consiste nel precalcolare la matrice delle distanze e di passarla poi come parametro all'algoritmo. DBSCAN restituisce essenzialmente gli stessi risultati (è deterministico per i punti centrali e di rumore, ma non per i punti di confine) ad ogni esecuzione, pertanto non è necessario eseguirlo più volte. Una generalizzazione di DBSCAN è rappresentata dall'algoritmo OPTICS *(Ordering Points To Identify the Clustering Structure*), avente una complessità di $O(n^2)$, che elimina la necessità di scegliere un valore appropriato per il parametro *eps* e produce un risultato gerarchico correlato a quello del clustering di collegamento.

Blowers e Williams [1] hanno utilizzato DBSCAN per effettuare il clustering dei pacchetti di rete normali e anomali presenti nel dataset KDD. Le features sono selezionate attraverso un'analisi di correlazione sul dataset. Il FAR effettivo del sistema non viene riportato, ma quest'ultimo è controllato attraverso il valore di soglia del metodo di clustering. Tale

approccio raggiunge un tasso di rilevamento del 98%, che risulta molto elevato se comparato ad altri IDS presenti in letteratura che sfruttano il clustering per effettuare il rilevamento di anomalie.

Casas et al. [3] hanno proposto un algoritmo di Sub-Space Clustering and Evidence Accumulation (SSC-EA) che adotta come metodo di clustering DBSCAN per il rilevamento di intrusioni nelle reti. L'algoritmo divide lo spazio delle features in più sub-spaces e applica su ognuno di essi il metodo DBSCAN, quindi procede combinando i risultati per realizzare una nuova misura di similarità, in grado di evidenziare le anomalie (outliers) e i cluster di piccole dimensioni rilevate simultaneamente in differenti sub-spaces. L'algoritmo SSC-EA è stato integrato in un IDS completo, nel quale i pacchetti di rete sono dapprima convertiti in flussi ed in seguito aggregati a differenti livelli prima di essere elaborati da un modulo di change detection. Se una modifica è rilevata l'algoritmo effettua il clustering dei dati ed utilizza un valore di soglia predefinito per determinare se un dato cluster è malevolo o no. I risultati hanno mostrato che il sistema proposto è efficace nel rilevare gli attacchi e che l'utilizzo del sub-clustering si presta alla parallelizzazione del sistema e ad un rilevamento in tempo reale. Nello studio di Bohara et al. [3] gli algoritmi DBSCAN e K-means sono utilizzati per realizzare un IDS ibrido che combina le tecniche di rilevamento host-based e network-based utilizzando log di sistema e di rete. Le prestazioni dei due algoritmi sono state valutate e comparate su dati Firewall. Nei test DBSCAN ha ottenuto prestazioni nettamente superiori rispetto a K-means ed è stato adottato come metodo di clustering per il sistema proposto.

2.4 Riduzione della dimensionalità dei dati

Uno degli aspetti più importanti nelle tecniche di apprendimento automatico applicate a dataset di grandi dimensioni, come nel caso del rilevamento delle intrusioni, consiste nella riduzione del numero di features del dataset di partenza, in modo da considerare le più significative e tralasciare invece quelle più irrilevanti o ridondanti, la cui presenza si traduce spesso in una scarsa capacità di generalizzazione del modello di ML e porta all'overfitting [3]. Questo può essere fatto attraverso i seguenti due approcci:

- *Feature Selection:* le tecniche di Feature Selection (FS) consistono nel selezionare un sottoinsieme delle feature disponibili in modo che queste risultino le più rilevanti e meno ridondanti. Vi sono due principali categorie di metodi: i filters ed i wrappers [3]. Un metodo di FS di tipo filter non prende in considerazione la classificazione ma assegna un punteggio alle varie features utilizzando la teoria dell'Informazione e metodi statistici, come il guadagno di informazione (IG), il coefficiente di correlazione e l'entropia dell'informazione. Un metodo di FS wrapper valuta l'insieme delle features attraverso un modello di predizione basato sull'algoritmo di rilevamento utilizzato. Il processo ha due componenti principali: una strategia di ricerca e un criterio di valutazione. La strategia di ricerca è responsabile della selezione delle features da considerare come parte del sottoinsieme ottimale. Il criterio di valutazione assegna un punteggio a ciascuna feature. Se questo supera una certa soglia la feature viene considerata rilevante e inclusa nel sottoinsieme. In molti casi la selezione delle features risulta più importante della scelta dell'algoritmo di detection.
- *Feature Extraction:* le tecniche di Feature Extraction (FE) hanno come scopo la creazione (estrazione) di nuove features di qualità superiore a partire da quelle disponibili. Tra le più comuni tecniche vi è la Principal Component Analysis (PCA).

2.4.1 Selezione delle features basata sulla correlazione

La selezione delle features basata sulla correlazione (CBF) consiste nel rimuovere da un dataset le features che presentano un alto grado di correlazione tra di loro, al fine di evitare la sovrapposizione di informazioni e migliorare la performance del modello. Questo approccio può essere applicato attraverso diverse tecniche, tra le quali l'analisi del coefficiente di correlazione di Pearson [44]. Tale indice esprime la presenza di una relazione lineare tra due variabili statistiche ed è definito come il rapporto tra la loro covarianza ed il prodotto delle deviazioni standard. Formalmente:

$$\rho_{XY} = \frac{\sigma_{XY}}{\sigma_X \sigma_Y}.$$

Esso può assumere valori compresi nell'intervallo [-1, 1]. Un valore dell'indice di correlazione pari agli estremi di tale intervallo corrisponde a punti che si trovano esattamente su una linea retta. In generale si ha che:

- se $\rho_{XY} > 0$ le variabili risultano direttamente correlate;
- se $\rho_{XY} < 0$ le variabili risultano inversamente correlate;
- se $\rho_{XY} = 0$ le variabili sono incorrelate.

In alcuni casi tuttavia la rimozione delle features altamente correlate potrebbe comportare l'eliminazione di informazioni importanti per il modello, inoltre, la rimozione di tali features non sempre garantisce che il modello sia meno suscettibile all'overfitting.

2.4.2 Analisi delle componenti principali

L'analisi dei componenti principali (PCA) è una tecnica statistica utilizzata per la riduzione del numero di features in un dataset che esegue una trasformazione ortogonale su un insieme di features che presentano un certo livello di correlazione allo scopo di ottenere un nuovo insieme di features non correlate. Rappresenta quindi una trasformazione geometrica che allinea i dati rispetto agli autovettori, detti assi o componenti principali: nel nuovo sistema di riferimento le features risultano incorrelate, e le loro varianze sono uguali agli autovalori. Le componenti principali sono ordinate in base alla loro importanza, ovvero alla quantità di varianza spiegata nei dati.

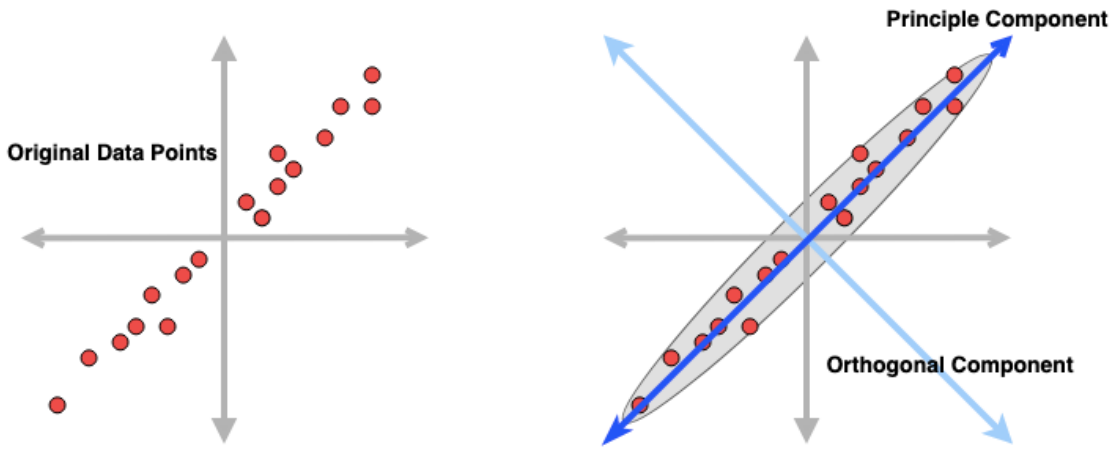

Figura 11: Grafici che mostrano come la PCA allinea i dati tratti da [45].

La PCA viene utilizzata nel machine learning per differenti scopi, tra i quali:

- *Riduzione della dimensionalità:* la PCA è una delle tecniche più popolari applicate per ridurre il numero di feature in un dataset di input, il che può migliorare le prestazioni dei modelli di apprendimento automatico e aumentare la velocità di elaborazione;
- *Visualizzazione dei dati*: mediante la PCA è possibile visualizzare i dati in uno spazio di dimensione inferiore, il che può essere utile per comprendere la struttura dei dati, la loro distribuzione ed identificare quindi eventuali pattern nascosti.

Il processo di PCA prevede di calcolare la matrice di covarianza dei dati, di calcolare i vettori e autovalori della matrice di covarianza e di selezionare quindi le componenti principali più significative, ovvero quelle con gli autovalori maggiori. Una volta determinate le componenti principali, i dati possono essere trasformati nello spazio delle componenti principali e utilizzati per l'addestramento di un modello di apprendimento automatico.

2.5 Confronto tra metodi supervisionati e non supervisionati

In generale i metodi supervisionati non sono in grado di rilevare gli attacchi sconosciuti, ma offrono buone prestazioni con quelli noti con un basso tasso di falsi positivi. Al contrario, i metodi non supervisionati sono in grado di rilevare attacchi sconosciuti, ma presentano un elevato tasso di falsi positivi. Con l'evoluzione della varietà di attacchi, sono stati proposti nuovi metodi per combinare questi due approcci in un sistema ibrido, per ottenere sia un elevato grado di rilevamento che un basso tasso di falsi positivi e, in generale, risultati migliori. La ricerca ha infatti dimostrato che in tale ambito la combinazione di più algoritmi restituisce prestazioni migliori rispetto ai singoli algoritmi.

2.6 Metodi ibridi

Un sistema di rilevamento delle intrusioni ibrido è costituito da due componenti: il primo elabora i dati non classificati, il secondo riceve in ingresso i dati elaborati e li analizza per rilevare le attività di intrusione.

Gli IDS ibridi possono essere classificati in:

- *ibridi in cascata:* combinano gli algoritmi tradizionali in serie o in parallelo. Sono i più utilizzati in letteratura.
- *ibridi basati su integrazione:* mirano ad ottimizzare gli algoritmi classici, sono più efficienti e producono risultati migliori rispetto alle altre tecniche ibride.

Un esempio di sistema ibrido è dato dal lavoro di Farid et al. [2], nel quale viene proposto un AIDS ibrido basato su alberi di decisione e che utilizza un classificatore Naive Bayes. Il sistema è stato in grado di generare un tasso di rilevamento del 99,63% sul dataset KDD 1999. Song et al. [3] hanno presentato un IDS ibrido che combina il clustering ed un classificatore SVM one-class al fine di rilevare gli attacchi attraverso la modellazione del comportamento normale. Durante l'addestramento i dati vengono prima filtrati per isolare il traffico normale, successivamente vengono raggruppati in cluster e per ciascuno di essi viene creata una SVM one-class. Nella fase di test i dati sono raggruppati ed i cluster formati vengono confrontati con i modelli SVM one-class precedentemente creati. Se non vi è alcuna corrispondenza tra un cluster ed uno dei modelli SVM, questo viene considerato un attacco. Il metodo proposto però ha due svantaggi: richiede di effettuare regolarmente l'addestramento, operazione onerosa dal punto di vista computazionale, e può generare un alto tasso di falsi positivi poiché il comportamento normale di una rete cambia costantemente con l'aggiunta di nuovi hardware ed applicazioni. Lo studio di Elbasiony et al. [3] si è concentrato nello sviluppo di un IDS ibrido che utilizza le Random Forest e l'algoritmo weighted K-means per combinare i vantaggi degli IDS di tipo anomaly-based e misuse-based. Il sistema proposto ha due modalità: online e offline. Nella prima il traffico è confrontato con le signatures di tipo misuse utilizzando l'algoritmo RF e, se non esiste alcuna corrispondenza, esso viene inoltrato al modulo offline, nel quale il rilevamento delle anomalie sarà effettuato attraverso K-means, ed a partire dai risultati di quest'ultimo vengono generate nuove signatures che possono poi essere utilizzate dal modulo misuse-based. Un ulteriore esempio di IDS ibrido è dato dal lavoro di Chandrasekhar e Raghuveer [3], i quali hanno proposto un sistema di rilevamento delle intrusioni che utilizza K-means,

SVM e reti neurali fuzzy. K-means viene utilizzato per generare i cluster di addestramento a partire dal dataset iniziale e per ogni sottoinsieme di addestramento viene addestrato un diverso modello neuro-fuzzy, in grado produrre un vettore per la classificazione tramite un SVM. Il metodo proposto ha raggiunto tassi particolarmente elevati con le classi di attacco U2R e R2L.

3. Valutazione sperimentale delle tecniche di machine learning e analisi delle problematiche

In questo capitolo vengono presentati i risultati della valutazione sperimentale dei metodi analizzati e discussi nel presente elaborato. I test sono stati effettuati in primo luogo mediante i classificatori discussi nel paragrafo 2.2, e successivamente attraverso alcuni degli algoritmi di clustering presentati nel paragrafo 2.3. Questo capitolo descrive gli strumenti di sviluppo ed i dataset utilizzati, la metodologia di svolgimento dei test effettuati, presenta ed analizza i risultati ottenuti dalla valutazione sperimentale.

3.1 Tool di sviluppo

Per svolgere il presente lavoro di tesi si è scelto di utilizzare il linguaggio Python, attraverso la libreria Scikit-learn e la piattaforma Google Colab. Scikit-learn è una libreria open-source per il linguaggio Python che integra un'ampia gamma di algoritmi di machine learning. In particolare include algoritmi di classificazione (alberi di decisione, Naive Bayes, K-nearest neighbors, …), di regressione logistica, di clustering (K-means, DBSCAN, …), ed è progettato per operare con le librerie NumPy e SciPy. La piattaforma Google Colaboratory (Colab), basata su un progetto Open Source chiamato Jupyter, consente di scrivere ed eseguire codice Python attraverso un browser utilizzando un account Google. I test sono stati effettuati su una macchina messa a disposizione tramite tale piattaforma avente le seguenti specifiche: Intel Xeon CPU @2.20 GHz, 13 GB RAM, Tesla K80 accelerator, 12 GB GDDR5 VRAM.

3.2 Dataset CICIDS2017

Molti degli studi effettuati in letteratura utilizzano dataset in larga parte obsoleti e inaffidabili, in quanto soffrono della mancanza di diversità e volumi di traffico, ed in generale non coprono la varietà di attacchi odierni. Il dataset CIC-IDS-2017 [17] è stato realizzato dal CIC (Canadian Institute of Cybersecurity) con l'obiettivo di offrire un'alternativa a tali dataset e di riprodurre in maniera fedele il traffico di rete reale attraverso dati di rete in formato .pcap (Packet Capture Data). Esso comprende i risultati dell'analisi del traffico di rete effettuata utilizzando CICFlowMeter, un generatore e analizzatore di flussi di traffico di rete bidirezionale (dalla sorgente alla destinazione e viceversa), con flussi etichettati basati su timestamp, IP di origine e destinazione, porte di origine e destinazione, protocolli e attacco. È stato inoltre utilizzato il sistema B-Profile per profilare il comportamento astratto delle interazioni umane di 25 utenti in base ai protocolli HTTP, HTTPS, FTP, SSH ed e-mail, e generare in tal modo traffico di rete benigno.

Nome dei file	**Classi**
Monday-Hours.pcap_ISCX.csv	Benign (Normale traffico)
Tuesday-Hours.pcap_ISCX.csv	Benign, FTP-Patator, SSH Patator
Wednesday-.pcap_ISCX.csv	Benign, DoS GoldenEye, DoSHulk, DoS lowhttptest, DoS slowloris, Heartbleed
Thursday-WebAttacks.pcap_ISCX.csv	Benign, Brute Force, SQL Injection, XSS
Thursday-Infiltration.pcap_ISCX.csv	Benign, Infiltration
Friday-.pcap_ISCX.csv	Benign, Bot
Friday-PortScan.pcap_ISCX.csv	Benign, PortScan
Friday- DDoS.pcap_ISCX.csv	Benign, DDoS

Tabella 1: Struttura del dataset CICIDS2017 tratto da [4].

Come mostrato nella Tabella 1, il dataset è costituito da 8 file in formato .csv, contenenti dati acquisiti tra le 9 a.m. e le 5 p.m. nei giorni compresi tra lunedì 3 luglio 2017 e venerdì 7 luglio 2017. Ognuno dei file contiene al suo interno record corrispondenti a traffico benigno e record di determinate tipologie di attacchi, diverse per ciascun file. All'interno del presente dataset troviamo quindi attacchi quali:

- *Brute force:* rappresenta una delle tipologie di attacco più diffuse, e può essere utilizzata non solo per violare le password, ma anche per individuare pagine e contenuti nascosti in un'applicazione web;
- *PortScan:* un attaccante invia richieste di connessione ad un host target in un intervallo di indirizzi di porte, con l'obiettivo di individuare le porte sulle quali è attivo un servizio. L'aggressore può pertanto determinare quali servizi sono presenti sull'host;
- *DoS Hulk:* il nome dell'attacco deriva dall'omonimo tool hulk, progettato per generare volumi di traffico univoco e offuscato su un server HTTP target, aggirando i meccanismi di caching allo scopo di esaurire il pool di risorse del server;
- *DoS Goldeneye:* attacco DoS realizzato mediante il tool GoldenEye, il quale è utilizzato per saturare le risorse di un server target inviandogli un gran numero di richieste HTTP persistenti (con l'header *keep alive*), in modo tale che questo non sarà in grado accettare le richieste degli utenti legittimi;
- *DoS slow-HTTP e slowloris:* Slowloris è un tool che consente ad un attaccante di abbattere un server Web target impiegando una larghezza di banda minima. Slowloris cerca di aprire un gran numero di connessioni con il server Web target e di mantenerle aperte il più a lungo possibile inviando richieste HTTP parziali. Periodicamente, invierà successivi header HTTP, aggiungendo, ma senza mai completare, la richiesta. I server manterranno queste connessioni aperte, riempiendo il pool massimo di connessioni simultanee, e negando pertanto ulteriori tentativi di connessione da parte degli utenti legittimi;
- *Heartbleed:* tale attacco deriva da un bug nella libreria di crittografia OpenSSL (corretto nel 2014), che è un'implementazione ampiamente utilizzata del protocollo Transport Layer Security (TLS). Normalmente viene sfruttato inviando ad un host

target (in genere un server) una richiesta heartbeat malformata con un piccolo payload e con il campo length impostato ad un valore superiore alla norma, in modo tale da suscitare la risposta della vittima contenente dati presenti nella memoria di quest'ultima;

- *Web Attacks:* in questa categoria rientrano gli attacchi Web Attacks – SQL Injection, Cross-Site Scripting (XSS) e Brute Force HTTP. Nel primo l'attaccante crea una stringa malevola di comandi SQL e la inserisce (*injection*) tra i parametri di una query SQL, quindi la utilizza per forzare il database ed ottenere informazioni sugli utenti. Un attacco XSS si verifica quando un utente malintenzionato utilizza un'applicazione Web per inviare ad un host target codice dannoso, generalmente sotto forma di segmenti di codice javascript, ma può anche includere HTML, Flash o qualsiasi altro tipo di codice che il browser è in grado eseguire. L'attaccante avrà in tal modo accesso a dati privati della vittima, come cookie o altre informazioni sulla sessione, e potrà effettuare ulteriori azioni come il reindirizzamento della vittima a contenuti Web sotto il suo controllo. La terza tipologia di attacco consiste nel provare l'elenco di tutte le possibili password in un form di login di una pagina web allo scopo di trovare quella corretta;
- *Infiltration:* sfrutta in genere una vulnerabilità di un software per eseguire una backdoor sull'host target e condurre diversi attacchi sulla rete della vittima come IP sweep e PortScan;
- *Botnet:* in questo attacco una botnet, ovvero una serie di dispositivi infettati da malware, è utilizzata dall'attaccante per eseguire varie tipologie di attacchi: può essere utilizzata per inviare spam, rubare dati, infiltrarsi in una rete o in un sistema e per sferrare attacchi DoS distribuiti;
- *DDoS:* attacco DoS distribuito, ovvero realizzato in modo inondare la larghezza di banda o le risorse di un host target, rendendo in tal modo indisponibile il servizio che esso fornisce. Un tale attacco è spesso il risultato di più sistemi compromessi (ad esempio, una botnet) che inondano la vittima generando un enorme traffico di rete.

Nella Figura 12 è mostrato come sono distribuite le diverse classi di attacchi all'interno del dataset: come si può notare il numero di campioni benigni risulta preponderante rispetto alle altre label, per cui ne risulta un dataset fortemente sbilanciato.

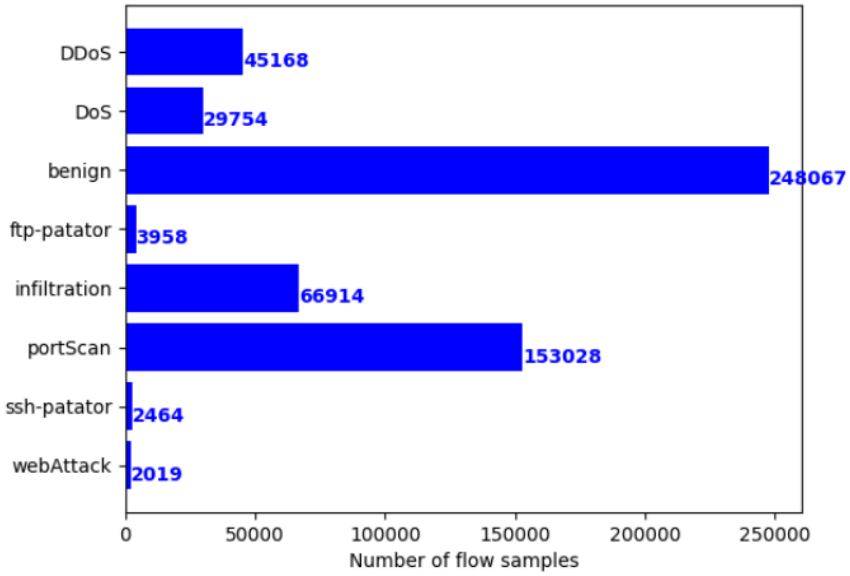

Figura 12: Distribuzione delle classi del CICIDS2017 tratto da [56].

Gli autori del lavoro *"Benchmarking of Machine Learning for Anomaly Based Intrusion Detection Systems in the CICIDS2017 Dataset"* [4], hanno effettuato una valutazione sperimentale di alcuni metodi di machine learning applicati al dataset in esame in questo lavoro di tesi.

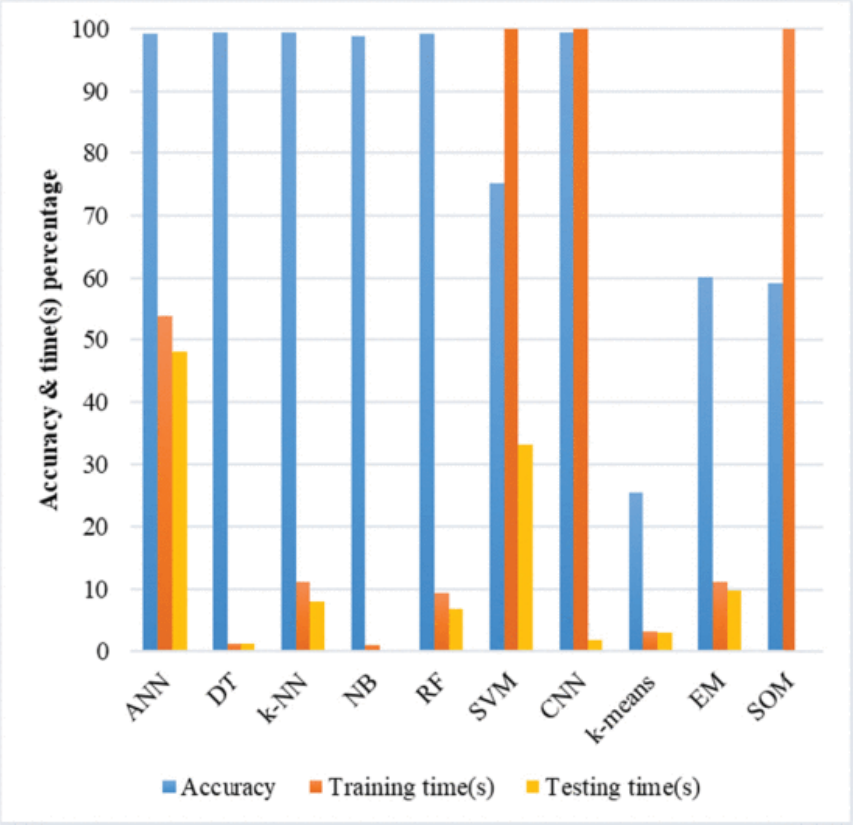

Figura 13: Grafico dei risultati del benchmark tratto da [4].

In tale paper è effettuato un benchmark dei principali metodi di Machine Learning supervisionati e non supervisionati sul dataset CICIDS2017, in particolare sul frammento contenente le etichette (Benign, Brute Force, SQL Injection, XSS), ed in esso viene evidenziata l'assenza di un singolo algoritmo di ML che sia in grado di rilevare tutti i tipi di attacchi. Come mostrato nella Figura 13, i modelli di AIDS basati su K-NN, DT e NB ottengono prestazioni molto buone in termini di accuracy, mentre i modelli che utilizzano gli algoritmi SOM ed EM ottengono scarse prestazioni a causa dei loro elevati tassi di falsi positivi e di falsi negativi. In generale gli algoritmi di apprendimento supervisionato superano quelli non supervisionati, e tra questi gli algoritmi K-NN, DT e NB hanno una maggiore capacità di rilevare i *Web Attacks* rispetto agli altri. DT e K-NN emergono come i migliori algoritmi di apprendimento supervisionato quando sia il tempo di addestramento che quello di test vengono considerati e ignorati mentre EM risulta il miglior algoritmo di apprendimento non supervisionato.

3.3 Metodologia di valutazione

Per effettuare una valutazione delle tecniche di machine learning descritte in questo lavoro di tesi è stata seguita la procedura descritta di seguito. I risultati ottenuti sono stati poi confrontati con i risultati sperimentali descritti in [4].

Ciascuna prova è stata effettuata attraverso i seguenti step:

1. *Preprocessing:* preelaborazione di ciascun file .csv ad eccezione del primo, contenente solo record di traffico benigno (`Monday-Hours.pcap_ISCX.csv`);
2. *Riduzione della dimensionalità:* applicazione di una tecnica di Feature Selection o Feature Extraction (PCA) in modo da ridurre il numero delle features del dataset;
3. *Label replacement:* sostituzione dei valori di tipo categorico delle etichette nel dataset con valori numerici;
4. *Dataset split:* suddivisione del dataset in un insieme di addestramento ed un insieme di test;
5. *Addestramento:* applicazione di uno dei metodi supervisionati o non supervisionati sul training set;
6. *Test:* generazione delle label predette per il test set;
7. *Calcolo delle metriche di valutazione:* calcolo della matrice di confusione e delle metriche di valutazione *Accuracy, Precision, Recall, F1-Score* per ciascuna classe di attacco.

3.4 Preprocessing dei dati

Il preprocessing dei dati consiste nell'effettuare su ciascun frammento di dataset contenuto nel rispettivo file .csv le seguenti operazioni:

- *Cleaning:* prevede la rimozione dei record contenenti valori mancanti, dei valori duplicati e dei valori di tipo non numerico (*NaN*) o infinito;
- *Scaling:* in tale step è effettuata la normalizzazione dei valori del dataset attraverso il metodo StandardScaler() della libreria Scikit-learn, nell'intervallo [0, 1]. La standardizzazione Z di un campione x è calcolata come:

$$z = \frac{x - \mu}{\sigma},$$

dove μ e σ rappresentano la media e la deviazione standard dei campioni di addestramento rispettivamente.

3.5 Riduzione della dimensionalità

In tali prove sono state utilizzate due tecniche per ridurre il numero di features di partenza. In primo luogo si è adottata una tecnica di Feature Selection basata sulla rimozione delle features aventi un elevato grado di correlazione (> 0.8) in base al coefficiente di Pearson, ottenendo una riduzione delle features dal valore iniziale di 78 a sole 40. Alternativamente a questa è stata inoltre applicata su ciascun frammento di dataset considerato l'implementazione della PCA fornita dalla libreria Scikit-learn, scegliendo un numero di componenti principali in base al grafico della varianza cumulativa, ovvero la varianza totale spiegata dalle prime *n* componenti principali, nel punto in cui questa raggiunge un livello di saturazione, non aumentando in maniera significativa in presenza di ulteriori componenti. In questo caso la scelta del numero di componenti principali da utilizzare è stata effettuata partendo dalle prime 30, sufficienti per quasi tutti dei frammenti del dataset CICIDS2017 a descrivere più del 95% della varianza cumulativa, ed incrementando tale numero fino ad arrivare a descriverne il 99,2% circa.

3.6 Split del dataset

Il dataset è stato suddiviso mediante una *Stratified K-fold cross-validation* in due parti: il 66,66% è stato utilizzato come insieme di addestramento, mentre il restante 33,33% è stato utilizzato per la fase di test. Tale scelta è stata effettuata poiché in una tipica validazione incrociata tramite *K-Fold* il dataset viene suddiviso in k sezioni casuali (fold), delle quali k-1 sono utilizzate per l'addestramento del modello e la restante è utilizzata per la validazione. Il processo viene ripetuto per k volte, cambiando ad ogni esecuzione la parte adibita alla validazione. Tramite una *Stratified K-Fold cross validation*, invece, il dataset viene diviso

in modo da mantenere la stessa proporzione di campioni per ogni classe in ogni fold. Così facendo viene garantito che ogni sezione contiene un insieme rappresentativo di tutte le classi del dataset, che tiene conto della distribuzione ed evita problemi di sbilanciamento delle classi. In particolare quindi, in presenza di dataset sbilanciati, come il CICIDS2017, costituisce un metodo più preciso ed affidabile per valutare le prestazioni di un modello.

3.7 Misure di prestazioni

Questa sezione presenta le metriche di valutazione adottate nelle prove:

- *Confusion Matrix:* utilizzata principalmente nell'apprendimento supervisionato per valutare l'accuratezza della predizione di un classificatore. Nella Figura 14 è mostrato un esempio di confusion matrix: ogni riga della tabella corrisponde a un risultato predetto dal classificatore, mentre ogni colonna corrisponde a un risultato effettivo;

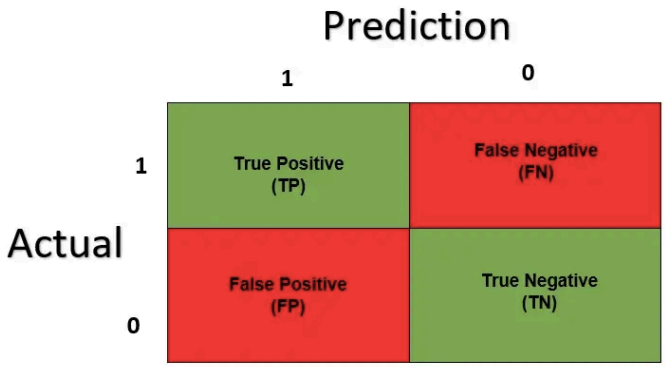

Figura 14: struttura di una generica matrice di confusione tratta da [4].

- *Accuracy:* è definita come il rapporto tra le istanze correttamente classificate dei veri positivi e dei veri negativi ed il numero totale delle istanze;

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN};$$

- *Precision:* consiste nel rapporto tra le istanze positive correttamente classificate su tutte le istanze classificate come positive;

$$Precision = \frac{TP}{TP + FP};$$

- *Recall o Sensitivity o True Positive Rate (TPR):* rappresenta il rapporto tra il numero delle istanze positive identificate correttamente sul numero di istanze appartenenti alla classe positiva;

$$Recall = \frac{TP}{TP + FN};$$

- F1-score: è definita come la media armonica della *precision* e della *recall*;

$$F1\_score = 2\,\frac{Precision * Recall}{Precision + Recall};$$

Ulteriori metriche comunemente utilizzate sono:

- *Specificity o True Negative Rate (TNR):* rappresenta il rapporto tra le istanze correttamente classificate come negative sul numero di istanze appartenenti alla classe positiva;

$$Specificity = \frac{TN}{TN + FP};$$

- *FAR o False Positive Rate (FPR) o Fall-out:* è definito come il rapporto tra le istanze classificate erroneamente come positive sul numero di istanze che appartengono alla classe negativa;

$$FAR = \frac{FP}{TN + FP} = 1 - Specificity;$$

- *Receiver Operating Characteristic (ROC):* curva che descrive il TPR in funzione del FAR per diverse impostazioni dei parametri (si veda la Figura 15). Ad un elevato valore di FAR sull'asse x corrisponde un'elevata Sensitivity sull'asse y.

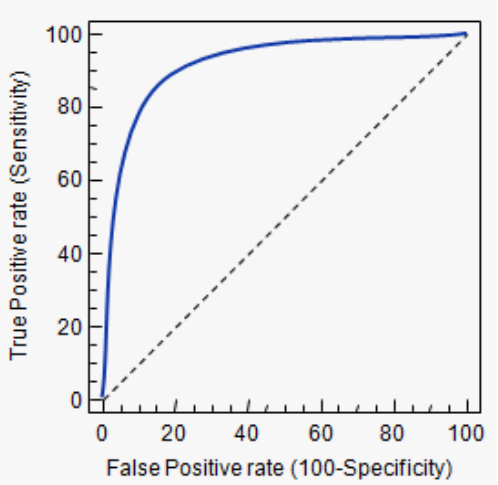

Figura 15: Grafico di esempio di una generica curva ROC tratto da [38].

3.8 Risultati sperimentali dei metodi supervisionati

La valutazione sperimentale dei metodi supervisionati analizzati nel paragrafo 2.2, sul dataset CICIDS2017 ha restituito i seguenti risultati:

Classificatore	**Label di classe**	**Con Feature selection correlation-based**				**Con Feature extraction mediante PCA**			
		Accuracy	**Precision**	**Recall**	**F1_score**	**Accuracy**	**Precision**	**Recall**	**F1_score**
A-NN	Benign	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	FTP-Patator	0.97	0.97	1.00	0.98	0.96	0.97	1.00	0.98
	SSH-Patator	0.95	0.97	0.97	0.97	0.97	0.99	0.98	0.98
	Valori globali	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
DT	Benign	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	FTP-Patator	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

	SSH-Patator	1.00	1.00	1.00	1.00	0.98	0.99	0.99	0.99
	Valori globali	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
RF	Benign	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	FTP-Patator	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	SSH-Patator	1.00	1.00	1.00	1.00	0.99	0.99	0.99	0.99
	Valori globali	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
K-NN	Benign	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	FTP-Patator	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	SSH-Patator	0.98	0.99	0.99	0.99	0.98	0.99	0.99	0.99
	Valori globali	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
GA	Benign	0.97	0.97	1.00	0.98	0.97	0.97	1.00	0.98
	FTP-Patator	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	SSH-Patator	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Valori globali	**0.97**	**0.94**	**0.97**	**0.95**	**0.97**	**0.94**	**0.97**	**0.95**
NB	Benign	0.99	1.00	0.99	1.00	0.59	1.00	0.59	0.74
	FTP-Patator	0.96	0.97	0.99	0.98	0.04	0.04	0.98	0.09
	SSH-Patator	0.68	0.68	0.99	0.81	0.33	0.33	0.99	0.49
	Valori globali	**0.99**	**1.00**	**0.99**	**0.99**	**0.60**	**0.97**	**0.60**	**0.72**
SVM	Benign	0.99	0.99	0.99	0.99	0.99	0.99	1.00	1.00
	FTP-Patator	0.82	0.82	1.00	0.90	0.91	0.91	1.00	0.95

	SSH-Patator	0.41	0.70	0.49	0.58	0.49	0.98	0.50	0.66
	Valori globali	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**

Tabella 2: Valutazione delle performance dei metodi supervisionati sul file "Tuesday-WorkingHours.pcap_ISCX.csv".

Classificatore	**Label di classe**	**Con Feature selection correlation-based**				**Con Feature extraction mediante PCA**			
		Accuracy	**Precision**	**Recall**	**F1_score**	**Accuracy**	**Precision**	**Recall**	**F1_score**
A-NN	Benign	0.96	0.97	0.99	0.98	0.95	0.97	0.99	0.98
	DoS GoldenEye	0.36	0.55	0.50	0.53	0.00	0.00	0.00	0.00
	DoSHulk	0.98	0.99	1.00	0.99	0.97	0.97	1.00	0.99
	DoS slowhttptest	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	DoS slowloris	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Heartbleed	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Valori globali	**0.97**	**0.95**	**0.97**	**0.96**	**0.97**	**0.95**	**0.97**	**0.96**
DT	Benign	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	DoS GoldenEye	0.99	0.99	1.00	1.00	0.97	0.99	0.99	0.99
	DoSHulk	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	DoS slowhttptest	0.98	0.99	0.99	0.99	0.98	0.99	0.99	0.99
	DoS slowloris	0.98	0.99	0.99	0.99	0.98	0.99	0.99	0.99

	Heartbleed	0.75	0.75	1.00	0.86	0.67	1.00	0.67	0.80
	Valori globali	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
RF	Benign	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	DoS GoldenEye	0.99	1.00	1.00	1.00	0.99	0.99	0.99	0.99
	DoSHulk	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	DoS slowhttptest	0.99	1.00	0.99	0.99	0.99	1.00	0.99	1.00
	DoS slowloris	0.99	0.99	0.99	0.99	0.99	0.99	1.00	1.00
	Heartbleed	0.67	1.00	0.67	0.80	0.33	1.00	0.33	0.50
	Valori globali	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
K-NN	Benign	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	DoS GoldenEye	0.98	0.99	1.00	0.99	0.98	0.99	0.99	0.99
	DoSHulk	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	DoS slowhttptest	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99
	DoS slowloris	0.98	0.99	0.99	0.99	0.99	0.99	1.00	0.99
	Heartbleed	0.67	1.00	0.67	0.80	0.67	1.00	0.67	0.80
	Valori globali	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
GA	Benign	0.92	0.94	0.98	0.96	0.92	0.94	0.97	0.96
	DoS GoldenEye	0.00	0.00	0.00	0.00	0.00	0.16	0.00	0.01
	DoSHulk	0.89	0.95	0.94	0.94	0.90	0.94	0.96	0.95

	DoS slowhttptest	0.48	0.83	0.53	0.65	0.00	0.00	0.00	0.00
	DoS slowloris	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Heartbleed	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Valori globali	**0.94**	**0.92**	**0.94**	**0.93**	**0.94**	**0.91**	**0.94**	**0.92**
NB	Benign	0.90	1.00	0.90	0.95	0.46	0.91	0.48	0.63
	DoS GoldenEye	0.13	0.13	0.99	0.23	0.29	0.33	0.70	0.45
	DoSHulk	0.68	0.93	0.71	0.81	0.45	0.49	0.86	0.62
	DoS slowhttptest	0.05	0.06	0.17	0.09	0.22	0.24	0.83	0.37
	DoS slowloris	0.20	0.21	0.93	0.34	0.19	0.22	0.52	0.31
	Heartbleed	0.67	1.00	0.67	0.80	0.33	1.00	0.33	0.50
	Valori globali	**0.83**	**0.95**	**0.83**	**0.88**	**0.61**	**0.75**	**0.61**	**0.62**
SVM	Benign	0.97	0.99	0.98	0.99	0.97	0.99	0.98	0.98
	DoS GoldenEye	0.83	0.93	0.88	0.91	0.79	0.94	0.83	0.88
	DoSHulk	0.96	0.97	0.99	0.98	0.96	0.97	0.99	0.98
	DoS slowhttptest	0.71	0.90	0.76	0.83	0.69	0.88	0.76	0.82
	DoS slowloris	0.74	0.92	0.79	0.85	0.75	0.96	0.78	0.86
	Heartbleed	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Valori globali	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**

Tabella 3: Valutazione delle performance dei metodi supervisionati sul file "Wednesday-workingHours.pcap_ISCX.csv".

Classificatore	Label di classe	Con Feature selection correlation-based				Con Feature extraction mediante PCA			
		Accuracy	**Precision**	**Recall**	**F1_score**	**Accuracy**	**Precision**	**Recall**	**F1_score**
A-NN	Benign	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	Brute Force	0.59	0.60	0.99	0.74	0.60	0.60	1.00	0.75
	XSS	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Sql Injection	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Valori globali	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**
DT	Benign	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	Brute Force	0.55	0.72	0.70	0.71	0.58	0.74	0.74	0.74
	XSS	0.26	0.39	0.43	0.41	0.27	0.40	0.45	0.43
	Sql Injection	0.58	0.58	1.00	0.74	0.21	0.30	0.43	0.35
	Valori globali	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**1.00**	**0.99**	**1.00**
RF	Benign	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	Brute Force	0.67	0.69	0.95	0.80	0.61	0.74	0.78	0.76
	XSS	0.05	0.55	0.06	0.10	0.25	0.43	0.38	0.40
	Sql Injection	0.43	1.00	0.43	0.60	0.14	1.00	0.14	0.25
	Valori globali	**1.00**	**1.00**	**1.00**	**0.99**	**1.00**	**1.00**	**1.00**	**1.00**
K-NN	Benign	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

	Brute Force	0.62	0.73	0.81	0.76	0.63	0.74	0.81	0.77
	XSS	0.22	0.43	0.32	0.37	0.23	0.43	0.34	0.38
	Sql Injection	0.11	0.33	0.14	0.20	0.12	0.50	0.14	0.22
	Valori globali	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
GA	Benign	0.99	0.99	1.00	0.99	0.99	0.99	1.00	0.99
	Brute Force	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	XSS	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Sql Injection	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Valori globali	**0.99**	**0.97**	**0.99**	**0.98**	**0.99**	**0.97**	**0.99**	**0.98**
NB	Benign	0.99	1.00	0.99	0.99	0.94	1.00	0.94	0.97
	Brute Force	0.07	0.15	0.12	0.13	0.00	0.00	0.00	0.00
	XSS	0.25	0.25	0.95	0.39	0.22	0.23	0.93	0.37
	Sql Injection	0.09	0.09	1.00	0.17	0.00	0.00	1.00	0.01
	Valori globali	**0.98**	**0.99**	**0.98**	**0.98**	**0.94**	**0.99**	**0.94**	**0.96**
SVM	Benign	0.99	1.00	1.00	1.00	0.99	0.99	1.00	0.99
	Brute Force	0.43	0.49	0.79	0.61	0.35	0.56	0.48	0.52
	XSS	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Sql Injection	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Valori globali	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**

Tabella 4: Valutazione delle performance dei metodi supervisionati sul file "Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX.csv".

Classificatore	Label di classe	Con Feature selection correlation-based				Con Feature extraction mediante PCA			
		Accuracy	**Precision**	**Recall**	**F1_score**	**Accuracy**	**Precision**	**Recall**	**F1_score**
A-NN	Benign	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	Infiltration	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Valori globali	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
DT	Benign	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	Infiltration	0.50	0.60	0.75	0.67	0.28	0.45	0.42	0.43
	Valori globali	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
RF	Benign	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	Infiltration	0.58	1.00	0.58	0.74	0.25	1.00	0.25	0.40
	Valori globali	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
K-NN	Benign	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	Infiltration	0.29	0.67	0.33	0.44	0.21	0.60	0.25	0.35
	Valori globali	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
GA	Benign	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	Infiltration	0.00	0.00	0.00	0.00	0.07	0.33	0.08	0.13
	Valori globali	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
NB	Benign	0.97	1.00	0.97	0.98	0.97	1.00	0.97	0.99
	Infiltration	0.00	0.00	1.00	0.01	0.00	0.00	0.75	0.01

	Valori globali	**0.97**	**1.00**	**0.97**	**0.98**	**0.97**	**1.00**	**0.97**	**0.99**
SVM	Benign	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	Infiltration	0.07	0.33	0.08	0.13	0.00	0.00	0.00	0.00
	Valori globali	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**

Tabella 5: Valutazione delle performance dei metodi supervisionati sul file "Thursday-WorkingHours-Afternoon-Infiltration.pcap_ISCX.csv".

Classificatore	**Label di classe**	**Con Feature selection correlation-based**				**Con Feature extraction mediante PCA**			
		Accuracy	**Precision**	**Recall**	**F1_score**	**Accuracy**	**Precision**	**Recall**	**F1_score**
A-NN	Benign	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	Bot	0.62	1.00	0.62	0.77	0.61	0.97	0.62	0.76
	Valori globali	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
DT	Benign	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	Bot	0.98	0.99	0.99	0.99	0.81	0.89	0.90	0.90
	Valori globali	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
RF	Benign	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	Bot	0.97	0.99	0.97	0.98	0.83	0.95	0.87	0.91
	Valori globali	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
K-NN	Benign	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	Bot	0.70	0.81	0.84	0.82	0.77	0.87	0.87	0.87

	Valori globali	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
GA	Benign	0.99	0.99	1.00	0.99	0.99	0.99	1.00	0.99
	Bot	0.01	0.43	0.01	0.02	0.02	1.00	0.02	0.04
	Valori globali	**0.99**	**0.98**	**0.99**	**0.98**	**0.99**	**0.99**	**0.99**	**0.99**
NB	Benign	0.83	1.00	0.83	0.90	0.97	1.00	0.98	0.99
	Bot	0.06	0.06	1.00	0.11	0.18	0.21	0.57	0.30
	Valori globali	**0.83**	**0.99**	**0.83**	**0.90**	**0.97**	**0.99**	**0.97**	**0.98**
SVM	Benign	0.99	0.99	1.00	1.00	0.99	1.00	1.00	1.00
	Bot	0.29	0.62	0.35	0.44	0.48	0.70	0.60	0.65
	Valori globali	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**

Tabella 6: Valutazione delle performance dei metodi supervisionati sul file "Friday-WorkingHours-Morning.pcap_ISCX.csv".

Classificatore	**Label di classe**	**Con Feature selection correlation-based**				**Con Feature extraction mediante PCA**			
		Accuracy	**Precision**	**Recall**	**F1_score**	**Accuracy**	**Precision**	**Recall**	**F1_score**
A-NN	Benign	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	PortScan	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	Valori globali	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
DT	Benign	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	PortScan	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

	Valori globali	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
RF	Benign	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	PortScan	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	Valori globali	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
K-NN	Benign	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	PortScan	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	Valori globali	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
GA	Benign	0.99	0.99	1.00	1.00	0.99	1.00	0.99	0.99
	PortScan	0.99	1.00	0.99	1.00	0.99	0.99	1.00	0.99
	Valori globali	**1.00**	**1.00**	**1.00**	**1.00**	**0.99**	**0.99**	**0.99**	**0.99**
NB	Benign	0.98	0.99	0.99	0.99	0.97	0.98	0.99	0.99
	PortScan	0.98	0.99	0.99	0.99	0.98	0.99	0.98	0.99
	Valori globali	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**
SVM	Benign	0.99	1.00	0.99	1.00	1.00	1.00	1.00	1.00
	PortScan	0.99	0.99	1.00	1.00	1.00	1.00	1.00	1.00
	Valori globali	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**

Tabella 7: Valutazione delle performance dei metodi supervisionati sul file "Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX.csv".

Classificatore	Label di classe	Con Feature selection correlation-based				Con Feature extraction mediante PCA			
		Accuracy	Precision	Recall	F1_score	Accuracy	Precision	Recall	F1_score
A-NN	Benign	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	DDoS	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	Valori globali	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
DT	Benign	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	DDoS	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	Valori globali	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
RF	Benign	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	DDoS	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	Valori globali	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
K-NN	Benign	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	DDoS	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	Valori globali	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
GA	Benign	0.93	1.00	0.94	0.97	0.94	1.00	0.94	0.97
	DDoS	0.95	0.95	1.00	0.98	0.96	0.96	1.00	0.98
	Valori globali	**0.97**	**0.97**	**0.97**	**0.97**	**0.98**	**0.98**	**0.98**	**0.98**
NB	Benign	0.97	1.00	0.97	0.99	0.76	0.97	0.78	0.86
	DDoS	0.98	0.98	1.00	0.99	0.84	0.85	0.98	0.91
	Valori globali	**0.99**	**0.99**	**0.99**	**0.99**	**0.89**	**0.90**	**0.89**	**0.89**
SVM	Benign	0.98	1.00	0.98	0.99	0.99	1.00	0.99	0.99

	DDoS	0.98	0.98	1.00	0.99	0.99	0.99	1.00	1.00
	Valori globali	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**

Tabella 8: Valutazione delle performance dei metodi supervisionati sul file "Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv".

I risultati sono coerenti con quanto riportato in [4] e confermano le ottime capacità di rilevamento dei metodi supervisionati considerati con entrambe le tecniche per la selezione delle features applicate. Nello specifico i metodi testati raggiungono valori di accuracy globali che oscillano tra 0.97 e 1.00 nella maggior parte dei casi per quasi tutti i classificatori.

3.9 Risultati sperimentali dei metodi non supervisionati

In questa sezione sono presentati i risultati della valutazione sperimentale di alcuni dei metodi di tipo non supervisionato (clustering) presentati nel paragrafo 2.3 effettuata sul dataset CICIDS2017. Sono stati ottenuti i seguenti risultati:

Metodo di Clustering	**Label di classe**	**Con Feature selection correlation-based**				**Con Feature extraction mediante PCA**			
		Accuracy	**Precision**	**Recall**	**F1_score**	**Accuracy**	**Precision**	**Recall**	**F1_score**
K-means	Benign	0.29	0.95	0.29	0.45	0.77	0.96	0.79	0.87
	FTP-Patator	0.03	0.03	0.49	0.06	0.00	0.00	0.00	0.00
	SSH-Patator	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Valori globali	**0.29**	**0.92**	**0.29**	**0.43**	**0.77**	**0.93**	**0.77**	**0.84**

EM	Benign	0.28	0.95	0.28	0.44	0.23	0.94	0.24	0.38
	FTP-Patator	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	SSH-Patator	0.02	0.02	0.50	0.04	0.01	0.01	0.51	0.02
	Valori globali	**0.28**	**0.92**	**0.28**	**0.42**	**0.24**	**0.91**	**0.24**	**0.37**
SOM	Benign	0.12	1.00	0.12	0.21	0.24	1.00	0.06	0.12
	FTP-Patator	0.00	0.05	0.49	0.10	0.00	0.03	0.50	0.05
	SSH-Patator	0.01	0.00	0.00	0.00	0.02	0.04	0.49	0.07
	Valori globali	**0.12**	**0.97**	**0.12**	**0.21**	**0.08**	**0.97**	**0.08**	**0.12**

Tabella 9: Valutazione delle performance degli algoritmi K-means, EM e SOM sul file "Tuesday-WorkingHours.pcap_ISCX.csv".

Metodo di Clustering	**Label di classe**	**Con Feature selection correlation-based**				**Con Feature extraction mediante PCA**			
		Accuracy	**Precision**	**Recall**	**F1_score**	**Accuracy**	**Precision**	**Recall**	**F1_score**
K-means	Benign	0.24	0.64	0.27	0.38	0.24	0.63	0.28	0.39
	DoS GoldenEye	0.00	0.00	0.00	0.00	0.01	0.01	0.12	0.02
	DoSHulk	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	DoS slowhttptest	0.04	0.11	0.06	0.08	0.00	0.00	0.12	0.01
	DoS slowloris	0.00	0.00	0.03	0.00	0.02	0.02	0.20	0.04
	Heartbleed	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Valori globali	**0.19**	**0.44**	**0.19**	**0.27**	**0.19**	**0.40**	**0.19**	**0.25**

EM	Benign	0.36	0.67	0.44	0.53	0.43	0.96	0.44	0.60
	DoS GoldenEye	0.00	0.00	0.00	0.00	0.03	0.03	0.42	0.05
	DoSHulk	0.18	0.58	0.20	0.30	0.00	0.00	0.00	0.00
	DoS slowhttptest	0.02	0.02	0.59	0.05	0.01	0.01	0.48	0.03
	DoS slowloris	0.03	0.03	0.14	0.05	0.00	0.00	0.00	0.00
	Heartbleed	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Valori globali	**0.35**	**0.62**	**0.35**	**0.44**	**0.29**	**0.61**	**0.29**	**0.38**
SOM	Benign	0.16	0.87	0.17	0.28	0.03	0.10	0.04	0.05
	DoS GoldenEye	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	DoSHulk	0.37	0.87	0.39	0.54	0.13	0.18	0.33	0.23
	DoS slowhttptest	0.03	0.03	0.73	0.06	0.03	0.05	0.05	0.05
	DoS slowloris	0.00	0.00	0.12	0.00	0.02	0.02	0.32	0.03
	Heartbleed	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Valori globali	**0.24**	**0.85**	**0.24**	**0.36**	**0.13**	**0.12**	**0.13**	**0.11**

Tabella 10: Valutazione delle performance degli algoritmi K-means, EM e SOM sul file "Wednesday-workingHours.pcap_ISCX.csv".

Metodo di Clustering	**Label di classe**	**Con Feature selection correlation-based**	**Con Feature extraction mediante PCA**

		Accuracy	Precision	Recall	F1_score	Accuracy	Precision	Recall	F1_score
K-means	Benign	0.27	1.00	0.27	0.43	0.79	0.98	0.79	0.88
	Brute Force	0.03	0.03	0.90	0.06	0.00	0.00	0.05	0.00
	XSS	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Sql Injection	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Valori globali	**0.28**	**0.98**	**0.28**	**0.42**	**0.79**	**0.97**	**0.79**	**0.87**
EM	Benign	0.46	1.00	0.46	0.63	0.47	0.98	0.47	0.64
	Brute Force	0.00	0.00	0.09	0.01	0.00	0.00	0.10	0.01
	XSS	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Sql Injection	0.00	0.00	0.57	0.00	0.00	0.00	0.43	0.00
	Valori globali	**0.45**	**0.99**	**0.45**	**0.62**	**0.47**	**0.97**	**0.47**	**0.63**
SOM	Benign	0.24	0.95	0.24	0.38	0.47	1.00	0.47	0.64
	Brute Force	0.00	0.00	0.09	0.01	0.00	0.00	0.00	0.00
	XSS	0.00	0.00	0.00	0.00	0.01	0.01	0.95	0.02
	Sql Injection	0.00	0.00	0.00	0.00	0.00	0.00	0.57	0.00
	Valori globali	**0.24**	**0.94**	**0.24**	**0.38**	**0.46**	**0.99**	**0.46**	**0.63**

Tabella 11: Valutazione delle performance degli algoritmi K-means, EM e SOM sul file "Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX.csv".

Metodo di Clustering	Label di classe	Con Feature selection correlation-based				Con Feature extraction mediante PCA			
		Accuracy	**Precision**	**Recall**	**F1_score**	**Accuracy**	**Precision**	**Recall**	**F1_score**
K-means	Benign	0.82	1.00	0.82	0.90	0.87	1.00	0.87	0.93
	Infiltration	0.00	0.00	0.83	0.00	0.00	0.00	1.00	0.00
	Valori globali	**0.82**	**1.00**	**0.82**	**0.90**	**0.87**	**1.00**	**0.87**	**0.93**
EM	Benign	0.22	1.00	0.22	0.36	0.77	1.00	0.77	0.87
	Infiltration	0.00	0.00	0.17	0.00	0.00	0.00	1.00	0.00
	Valori globali	**0.22**	**1.00**	**0.22**	**0.36**	**0.77**	**1.00**	**0.77**	**0.87**
SOM	Benign	0.35	1.00	0.35	0.51	0.18	1.00	0.18	0.30
	Infiltration	0.00	0.00	0.33	0.00	0.00	0.00	0.00	0.00
	Valori globali	**0.35**	**1.00**	**0.35**	**0.51**	**0.18**	**1.00**	**0.18**	**0.30**

Tabella 12: Valutazione delle performance degli algoritmi K-means, EM e SOM sul file "Thursday-WorkingHours-Afternoon-Infiltration.pcap_ISCX.csv".

Metodo di Clustering	Label di classe	Con Feature selection correlation-based				Con Feature extraction mediante PCA			
		Accuracy	**Precision**	**Recall**	**F1_score**	**Accuracy**	**Precision**	**Recall**	**F1_score**
K-means	Benign	0.38	0.41	0.86	0.55	0.16	0.99	0.16	0.28
	Bot	0.00	0.01	0.00	0.00	0.60	0.60	1.00	0.75
	Valori globali	**0.38**	**0.19**	**0.38**	**0.25**	**0.63**	**0.77**	**0.63**	**0.54**
EM	Benign	0.32	0.37	0.72	0.49	0.36	0.98	0.36	0.53
	Bot	0.00	0.01	0.00	0.00	0.66	0.66	0.99	0.79
	Valori globali	**0.32**	**0.17**	**0.32**	**0.22**	**0.71**	**0.80**	**0.71**	**0.68**

SOM	Benign	0.25	0.04	0.03	0.03	0.10	0.99	0.21	0.35
	Bot	0.00	0.53	0.43	0.48	0.00	0.00	0.00	0.00
	Valori globali	**0.25**	**0.31**	**0.25**	**0.28**	**0.10**	**0.44**	**0.10**	**016**

Tabella 13: Valutazione delle performance degli algoritmi K-means, EM e SOM sul file "Friday-WorkingHours-Morning.pcap_ISCX.csv".

Metodo di Clustering	**Label di classe**	**Con Feature selection correlation-based**				**Con Feature extraction mediante PCA**			
		Accuracy	**Precision**	**Recall**	**F1_score**	**Accuracy**	**Precision**	**Recall**	**F1_score**
K-means	Benign	0.24	0.98	0.24	0.39	0.80	0.99	0.80	0.89
	PortScan	0.01	0.01	0.59	0.02	0.00	0.00	0.03	0.00
	Valori globali	**0.25**	**0.97**	**0.25**	**0.39**	**0.80**	**0.98**	**0.80**	**0.88**
EM	Benign	0.72	0.99	0.72	0.84	0.73	1.00	0.73	0.84
	PortScan	0.01	0.01	0.35	0.02	0.02	0.02	0.65	0.05
	Valori globali	**0.72**	**0.98**	**0.72**	**0.83**	**0.73**	**0.99**	**0.73**	**0.84**
SOM	Benign	0.24	0.97	0.24	0.39	0.26	0.99	0.27	0.42
	PortScan	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Valori globali	**0.24**	**0.96**	**0.24**	**0.38**	**0.26**	**0.98**	**0.26**	**0.41**

Tabella 14: Valutazione delle performance degli algoritmi K-means, EM e SOM sul file "Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX.csv".

Metodo di Clustering	Label di classe	Con Feature selection correlation-based				Con Feature extraction mediante PCA			
		Accuracy	**Precision**	**Recall**	**F1_score**	**Accuracy**	**Precision**	**Recall**	**F1_score**
K-means	Benign	0.41	0.44	0.86	0.58	0.42	0.45	0.89	0.60
	DDoS	0.15	0.61	0.17	0.26	0.15	0.66	0.17	0.27
	Valori globali	**0.47**	**0.53**	**0.47**	**0.40**	**0.48**	**0.57**	**0.48**	**0.41**
EM	Benign	0.46	0.71	0.56	0.63	0.26	0.32	0.60	0.41
	DDoS	0.62	0.71	0.83	0.77	0.00	0.01	0.00	0.00
	Valori globali	**0.71**	**0.71**	**0.71**	**0.71**	**0.26**	**0.14**	**0.26**	**0.18**
SOM	Benign	0.35	0.40	0.72	0.52	0.12	0.21	0.22	0.22
	DDoS	0.15	0.46	0.18	0.26	0.23	0.38	0.36	0.37
	Valori globali	**0.41**	**0.43**	**0.41**	**0.37**	**0.30**	**0.31**	**0.30**	**0.30**

Tabella 15: Valutazione delle performance degli algoritmi K-means, EM e SOM sul file "Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv".

Con una Feature Selection di tipo correlation-based l'algoritmo EM risulta il migliore nella maggior parte dei casi, in base ai valori globali di accuracy, recall e di F1-score; l'algoritmo K-means restituisce in genere valori simili e, nel caso della Feature Extraction effettuata tramite PCA, superiori a quelli di EM, mentre l'algoritmo SOM si conferma il peggiore nel rilevamento a causa degli elevati elevati tassi di falsi positivi e di falsi negativi generati. I migliori risultati ottenuti con gli algoritmi EM e K-means tuttavia non raggiungono le prestazioni degli algoritmi supervisionati applicati ai medesimi frammenti di dataset. Da questo studio preliminare emerge quindi che l'utilizzo dei metodi di clustering non supervisionati non risulta conveniente rispetto a quelli supervisionati, poiché sia in letteratura che nei test effettuati questi hanno restituito risultati sensibilmente peggiori.

4. Soluzione proposta e valutazione sperimentale

*** OMISSISS ***

5. Conclusioni

In questo lavoro di tesi è stata effettuata un'analisi dei principali metodi di machine learning supervisionati e non supervisionati applicati nel rilevamento delle intrusioni in letteratura. Le prove sperimentali preliminari effettuate sul dataset CICIDS2017 hanno evidenziato come non risulti conveniente utilizzare i metodi non supervisionati rispetto a quelli supervisionati date le loro inferiori prestazioni nel rilevamento. Tuttavia i metodi non supervisionati risultano ancora una delle tecniche più efficaci per il rilevamento di anomalie legate ad attacchi sconosciuti. Questo lavoro di tesi si è quindi focalizzato sulla possibilità di realizzare un sistema ibrido basato su metodi non supervisionati, ma che utilizzi in aggiunta anche i metodi supervisionati per ottenere un miglioramento delle prestazioni. In particolare, nella soluzione descritta nel capitolo 4 è stata proposta una combinazione tra l'algoritmo K-means ed un classificatore Random Forest. I risultati ottenuti applicando due diverse tecniche per la selezione delle features, ossia correlation-based Feature Selection e PCA, hanno dimostrato come il metodo proposto si sia rivelato valido per il rilevamento delle intrusioni sul dataset testato, restituendo prestazioni significativamente migliori rispetto ai metodi non supervisionati comunemente utilizzati in tale ambito. Sebbene i risultati siano promettenti, questi non sono paragonabili a quelli restituiti dai metodi supervisionati applicati al medesimo frammento di dataset e vi sono ampi margini di miglioramento; in particolare l'applicazione di un partizionamento di tipo dinamico in base alla dimensione di ciascun cluster potrebbe portare a risultati migliori.

Bibliografia

[1] Anna L. Buczak, Erhan Guven, "*A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection",* IEEE Communications Surveys & Tutorials, 2016

[2] Ansam Khraisat, Iqbal Gondal, Peter Vamplew, Joarder Kamruzzaman, "*Survey of intrusion detection systems: techniques, datasets and challenges",* Springer, 2019

[3] Antonia Nisioti, Alexios Mylonas, Paul D. Yoo, Vasilios Katos, "*From Intrusion Detection to Attacker Attribution: A Comprehensive Survey of Unsupervised Methods"*, IEEE Communications Surveys & Tutorials, 2018

[4] Ziadoon Kamil Maseer, Robiah Yusof, Nazrulazhar Bahaman, Salama A. Mostafa, Cik Feresa Mohd Foozy, "*Benchmarking of Machine Learning for Anomaly Based Intrusion Detection Systems in the CICIDS2017 Dataset"*, IEEE Access, 2021

[5] Mohammad Najafimehr, Sajjad Zarifzadeh, Seyedakbar Mostafavi, "*A hybrid machine learning approach for detecting unprecedented DDoS attacks",* Springer, 2022

[6] H. Haripriya, Prathibhamo, Yashwant RPai, M. Sai Sandeep, Arya M Sankar, Srinivas Nag Veerla, Prema Nedungadi Amrita, *"Multi Label Prediction Using Association Rule Generation and Simple k-Means",* IEEE, 2016

[7] Elijah M. Maseno, Zenghui Wang, Hongyan Xing, *"A Systematic Review on Hybrid Intrusion Detection System"*, Hindawi Publishing Corporation, 2022

[8] Scikit Learn (disponibile alla URL: https://scikit-learn.org/)

[9] Matplotlib (disponibile alla URL: https://matplotlib.org)

[10] Cluster Analysis (disponibile alla URL: https://en.wikipedia.org/wiki/Cluster_analysis)

[11] Iman Sharafaldin, Arash Habibi Lashkari, Ali A. Ghorbani, *"A Detailed Analysis of the CICIDS2017 Data Set"*, Springer, 2019

[12] Miel Verkerken, Laurens D'hooge, Tim Wauters, Bruno Volckaert, Filip De Turck, *"Unsupervised Machine Learning Techniques for Network Intrusion Detection on Modern Data"*, IEEE, 2020

[13] Stuart Russell, Peter Norvig, *"Artificial Intelligence A Modern Approach"*, Pearson, 2010

[14] Pandas (disponibile alla URL: https://pandas.pydata.org/)

[15] Ensemble Learning (disponibile alla URL: https://en.wikipedia.org/wiki/Ensemble_learning)

[16] DBSCAN Python Example: The Optimal Value For Epsilon (disponibile alla URL: https://towardsdatascience.com/machine-learning-clustering-dbscan-determine-the-optimal-value-for-epsilon-eps-python-example-3100091cfbc)

[17] Intrusion Detection Evaluation Dataset (CIC-IDS2017) (disponibile alla URL: https://www.unb.ca/cic/datasets/ids-2017.html)

[18] Guo Pu, Lijuan Wang , Jun Shen, and Fang Dong, *"A Hybrid Unsupervised Clustering-Based Anomaly Detection Method",* Tsinghua Science and Technology, 2021

[19] Self-organizing map (disponibile alla URL: https://en.wikipedia.org/wiki/Self-organizing_map)

[20] Evaluation of Machine Learning Algorithms for Intrusion Detection System (disponibile alla URL: https://medium.com/cuelogic-technologies/evaluation-of-machine-learning-algorithms-for-intrusion-detection-system-6854645f9211)

[21] Classificatore Random Forest (disponibile alla URL: https://en.wikipedia.org/wiki/Random_forest)

[22] Classificatore Naive Bayes (disponibile alla URL: https://en.wikipedia.org/wiki/Naive_Bayes_classifier)

[23] Algoritmo Expectation-Maximization (disponibile alla URL: https://en.wikipedia.org/wiki/Expectation%E2%80%93maximization_algorithm)

[24] Immagine di un generico Decision Tree (disponibile alla URL: https://www.vebuso.com/2020/01/decision-tree-intuition-from-concept-to-application/)

[25] Immagine di una generica Random Forest (disponibile alla URL: https://medium.com/analytics-vidhya/random-forest-classifier-and-its-hyperparameters-8467bec755f6)

[26] Immagine di una struttura di rete con NIDS e HIDS (disponibile alla URL: https://it.frwiki.wiki/wiki/Syst%C3%A8me_de_d%C3%A9tection_d%27intrusion)

[27] Immagine del funzionamento di un percettrone (disponibile alla URL: https://www.javatpoint.com/single-layer-perceptron-in-tensorflow)

[28] Immagine di una generica rete neurale (disponibile alla URL: https://www.mathworks.com/discovery/deep-learning.html#:~:text=Deep%20learning%20is%20a%20machine,a%20pedestrian%20from%20a%20lamppost.)

[29] Iman Sharafaldin, Arash Habibi Lashkari and Ali A. Ghorbani, *"Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization",* International Conference on Information Systems Security and Privacy (ICISSP), 2018

[30] Algoritmo K-means (available at https://en.wikipedia.org/wiki/K-means_clustering)

[31] Attacco XSS (available at URL: https://owasp.org/www-community/attacks/xss/)

[32] Slowloris tool (disponibile alla URL: https://en.wikipedia.org/wiki/Slowloris_(computer_security))

[33] GoldenEye tool (disponibile alla URL: https://manpages.ubuntu.com/manpages/focal/man1/goldeneye.1.html)

[34] Dos Hulk tool (disponibile alla URL: http://www.effecthacking.com/2017/11/hulk-web-server-dos-tool.html)

[35] Emmanouil Vasilomanolakis, Shankar Karuppayah, Max Mühlhäuser, Mathias Fischer, *"Taxonomy and survey of collaborative intrusion detection"*, ACM Computing Surveys, Volume 47, Issue 4, July 2015

[36] R. Bace and P. Mell, *"NIST Special Publication on Intrusion Detection Systems"*, 2001

[37] Karen Scarfone Peter Mell, *"Guide to Intrusion Detection and Prevention Systems (IDPS)"*, Recommendations of the National Institute of Standards and Technology, NIST Special Publication 800-94, February 2007

[38] Immagine di una generica curva ROC (disponibile alla URL: https://www.medcalc.org/manual/roc-curves.php)

[39] Immagine delle equazioni delle distanze in K-NN (disponibile alla URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4978658/#:~:text=K%2Dnearest%20neighbor%20(k%2D,decide%20the%20final%20classification%20output.)

[40] Immagine del funzionamento di K-NN (disponibile alla URL: https://www.ibm.com/it-it/topics/knn)

[41] Immagine dei punti classificati in DBSCAN (disponibile alla URL: https://it.wikipedia.org/wiki/Dbscan)

[42] Immagine del funzionamento di una SVM (disponibile alla URL: https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm)

[43] Immagine di una generica matrice di confusione (disponibile alla URL: https://towardsdatascience.com/confusion-matrix-un-confused-1ba98dee0d7f)

[44] Indice di correlazione di Pearson (disponibile alla URL: https://it.wikipedia.org/wiki/Indice_di_correlazione_di_Pearson)

[45] Immagine PCA: Principal Component Analysis (disponibile alla URL: https://www.baeldung.com/cs/principal-component-analysis)

[46] R. Quinlan, C4.5: Programs for Machine Learning. San Mateo, CA, USA: Morgan Kaufmann, 1993

[47] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi, "*EXPOSURE: Finding malicious domains using passive DNS analysis*", presented at the 18th Annu. Netw. Distrib. Syst. Secur. Conf., 2011

[48] Wenchao Li, Ping Yi , Yue Wu, Li Pan, and Jianhua Li, *"A New Intrusion Detection System Based on KNN Classification Algorithm in Wireless Sensor Network"*, Hindawi Publishing Corporation, 2014

[49] Shi-Jinn Horng, Ming-Yang Su, Yuan-Hsin Chen, Tzong-Wann Kao, Rong-Jian Chen, Jui-Lin Lai, Citra Dwi Perkasa, *"A novel intrusion detection system based on hierarchical clustering and support vector machines"*, Expert Systems with Applications, 2011

[50] Kai Peng, Victor C. M. Leung, Qingjia Huang, *"Clustering Approach Based on Mini Batch Kmeans for Intrusion Detection System Over Big Data"*, IEEE Access, 2018

[51] Loka Raj Ghimire, Roshan Chitrakar, *"Integration of Expectation Maximization using Gaussian Mixture Models and Naïve Bayes for Intrusion Detection"*, Journal of Computer Science Research, Bilingual Publishing, 2021

[52] I.Mukhopadhyay, M.Chakraborty, *"EMID: A Novel Expectation Maximization based Intrusion Detection Algorithm"*, IEMCON, 2011

[53] Chun-dong Wang, He-feng Yu, Huai-bin Wang & Kai Liu, *"SOM-Based Anomaly Intrusion Detection System"*, International Conference on Embedded and Ubiquitous Computing EUC 2007: Embedded and Ubiquitous Computing pp 356–366

[54] Google Colab (disponibile alla URL: https://colab.research.google.com/)

[55] V. Agate, F.M. D'Anna, A. De Paola, P. Ferraro, G. Lo Re and M. Morana, "*A Behavior-Based Intrusion Detection System Using Ensemble Learning Techniques*", 6th Italian Conference on Cybersecurity, ITASEC 2022, Rome (Italy), June 2022

[56] Onur Barut, Yan Luo, Tong Zhang, Weigang Li, Peilong Li, *"NetML: A Challenge for Network Traffic Analytics"*, NetAI Workshop at SIGCOMM, 2020

[57] A. De Paola, S. Gaglio, G. Lo Re, M. Ortolani, "An ambient intelligence architecture for extracting knowledge from distributed sensors", ICIS '09:

Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and HumanNovember 2009

[58] De Paola A., Favaloro S., Gaglio S., Lo Re G., Morana M., Malware detection through low-level features and stacked denoising autoencoders, (2018) CEUR Workshop Proceedings, 2058

[59] De Paola A., Gaglio S., Lo Re G., Morana M. A hybrid system for malware detection on big data, (2018) INFOCOM 2018 - IEEE Conference on Computer Communications Workshops, pp. 45 – 50, DOI:10.1109/INFCOMW.2018.8406963

[60] Agate V., Curaba M., Ferraro P., Lo Re G., Morana M., Secure e-voting in smart communities, (2020) CEUR Workshop Proceedings, 2597, pp. 1 – 11

[61] Concone F., De Paola A., Lo Re G., Morana M., Twitter analysis for real-Time malware discovery, (2017) 2017 AEIT International Annual Conference: Infrastructures for Energy and ICT: Opportunities for Fostering Innovation, AEIT 2017, 2017-January, pp. 1 – 6, DOI: 10.23919/AEIT.2017.824055

[62] Bordonaro A., De Paola A., Lo Re G., Morana M., Smart Auctions for Autonomic Ambient Intelligence Systems, (2020) Proceedings - 2020 IEEE International Conference on Smart Computing, SMARTCOMP 2020, art. no. 9239687, pp. 180 – 187, DOI: 10.1109/SMARTCOMP50058.2020.00043