Progetto Ingegneria Della Conoscenza 2022-2023
Gianluga Laera 718556

Introduzione

Nell'era dell'informazione, la sicurezza dei sistemi informatici è diventata una priorità fondamentale. L'integrità, la riservatezza e la disponibilità delle informazioni sono aspetti cruciali per garantire un funzionamento ottimale di ogni organizzazione. Tuttavia, con l'avvento di tecnologie sempre più sofisticate, anche le minacce informatiche si sono evolute, rendendo i metodi tradizionali di protezione spesso inadeguati o inefficaci contro attacchi avanzati. In questo contesto, l'Intrusion Detection System (IDS) emerge come una soluzione di rilievo, essendo in grado di rilevare, registrare e potenzialmente prevenire tentativi di intrusione.

L'Intrusion Detection può essere concepita come un problema di classificazione: dati una serie di eventi o transazioni, l'obiettivo è identificare se ciascun evento è legittimo o malevolo. Le tecniche tradizionali si basano su firme o pattern noti di attacchi, ma sono spesso inefficaci contro nuovi o sconosciuti tipi di intrusione. Ecco dove l'ingegneria della conoscenza, in particolare le tecniche di machine learning, possono giocare un ruolo di primo piano.

Il machine learning, attraverso l'analisi di grandi volumi di dati e l'apprendimento automatico da essi, può rilevare anomalie o schemi sospetti che potrebbero non essere immediatamente evidenti con metodi convenzionali. L'addestramento di modelli predittivi su vasti dataset può aiutare a identificare nuovi tipi di attacchi, migliorando la reattività e l'efficacia dei sistemi IDS.

Il presente progetto universitario intende esplorare, sviluppare e valutare l'applicazione di diverse tecniche di machine learning al problema dell'intrusion detection. Attraverso lo studio dei dati a disposizione e analisi comparative ci proponiamo di indagare la potenzialità e le sfide di questa sinergia tra ingegneria della conoscenza e sicurezza informatica.

L'obiettivo finale del progetto è dimostrare le competenze apprese durante il corso di Ingegneria della Conoscenza, applicate ad un problema reale.

Argomenti trattati

- Apprendimento Supervisionato
- Apprendimento Probabilistico
- Sistemi basati su Conoscenza
    - Ontologie e Condivisione della Conoscenza

Strumenti utilizzati

L'intero progetto è stato sviluppato in python in particolare utilizzando jupyter notebook

IDS

La pervasività della connettività Internet, sia nelle sfere personali che professionali, ha ampliato l'area vulnerabile agli attacchi informatici. La sfida è difendersi sia dalle minacce note che da quelle sconosciute.

Uno dei principali metodi per salvaguardare le risorse accessibili tramite rete è attraverso l'uso di Sistemi di Rilevazione delle Intrusioni (IDS). Tuttavia, gli IDS tradizionali presentano limiti poiché sono principalmente basati su firme e possono rilevare solo minacce note. Vi è un potenziale nell'utilizzare l'apprendimento automatico (Machine Learning, ML), una sottocategoria dell'intelligenza artificiale, per potenziare questi sistemi, poiché permette ai computer di apprendere e rilevare nuove minacce senza essere esplicitamente programmati.

La complessità è aggravata con l'avvento di tecnologie come Internet delle Cose (IoT), intelligenza artificiale (AI) e computazione quantistica. Questo aumenta il livello di minaccia e, pertanto, la necessità di un cambiamento nell'approccio alla sicurezza della rete. L'anlisi dei Big Data (BDA) svolge un ruolo significativo nel monitoraggio e nella previsione di potenziali minacce, aiutando così le aziende nella presa di decisioni.

Un IDS è un software progettato per rilevare attività insolite o potenzialmente dannose. I servizi web, facilmente accessibili al pubblico, sono particolarmente vulnerabili. Gli attacchi web sono ora più focalizzati sull'exploit delle vulnerabilità delle applicazioni piuttosto che sulla violazione dei sistemi.

Lobiettivo di questo progetto è di analizzare un dataset contenete esempi di traffico di rete al fine di addestrare e valutare modelli di classificazione che siano in grado di predire se una certà attività è malevola o meno.

Dataset

Il dataset che si è scelto di utilizzare è NSL-KDD, questo potrebbe non rappresentare a pieno le reti attuali. Tuttavia, si ritiene comunque sia una scelta valida, anche perchè largamente utilizzato in letteratura.

Questo dataset è un aggiornamento di un altro dataset chiamato KDD'99, infatti ne risolve alcune criticità:

- Non contiene record ridondanti, evitando quindi che i classificatori subiscano un bias a causa di record frequenti.
- I set di test proposti non hanno duplicati, assicurando che la performance dei metodi di apprendimento non sia influenzata da record ricorrenti.
- La selezione dei record da ogni gruppo di difficoltà è inversamente proporzionale alla loro presenza nel dataset KDD originale, permettendo una variazione maggiore nei tassi di classificazione tra diversi metodi di apprendimento.
- Le dimensioni dei set di addestramento e di test sono ragionevoli, permettendo test completi senza selezioni casuali, garantendo risultati consistenti e comparabili tra diverse ricerche.

Le tipologie di attacchi etichettate all'interno del dataset ricadono in una di queste 4 categorie:

- Denial of Service Attack (DoS): si verifica quando un aggressore sovraccarica o riempie eccessivamente una risorsa di calcolo o memoria, impedendo così alle richieste legittime di essere eseguite o negando agli utenti autorizzati l'accesso a un sistema.
- User to Root Attack (U2R): è una tipologia di attacco dove l'intruso inizia con l'accesso a un account utente normale del sistema (ottenuto magari intercettando password o attraverso tecniche di ingegneria sociale) e riesce a sfruttare una vulnerabilità per ottenere accesso come superutente o "root".
- Remote to Local Attack (R2L): avviene quando un aggressore, pur avendo la capacità di inviare pacchetti a una macchina attraverso una rete ma senza possedere un account su tale macchina, sfrutta una vulnerabilità per ottenere accesso locale come utente di quella macchina.
- Probing Attack: è un tentativo di raccogliere dati su una rete di computer con l'obiettivo di aggirare le sue misure di sicurezza.

Bisogna sottolineare come gli esempi dell'insieme di test non provengono dalla stessa distribuzione di probabilità degli esempi dell'insieme di addestramento. Questa scelta è stata presa a seguito dell'assunzione che la maggior parte delle nuove tipologie di attacchi sono grossomodo varianti di attacchi già conosciuti, dunque riconoscere la "firma" degli attacchi noti è sufficiente per individuarne di nuovi.

Le feature disponibili nel dataset sono divisibili in quattro categorie distinte:

- Feature di Base: tutti i dettagli estraibili da una connessione di tipo TCP/IP
- Feature di Traffico: comprende le feature calcolate rispetto ad un certo intervallo di tempo, sono ulteriormente distinte a seconda che le connessioni abbiano stesso host o stesso servizio:

    - "stesso host": esaminano solo le connessioni negli ultimi 2 secondi che hanno lo stesso host di destinazione della connessione corrente, e calcolano statistiche relative al comportamento del protocollo, al servizio, ecc.
    - "stesso servizio": esaminano solo le connessioni negli ultimi 2 secondi che hanno lo stesso servizio della connessione corrente.
    - PROBLEMA: esistono diversi attacchi di esplorazione lenti che esaminano gli host (o le porte) utilizzando un intervallo temporale molto più lungo di 2 secondi, ad esempio, uno ogni minuto. Quindi non sono individuabili entro una finestra temporale di 2 secondi.
    - SOLUZIONE: le feature "stesso host" e "stesso servizio" vengono ricalcolate, ma basandosi sulla finestra di connessione di 100 connessioni piuttosto che su una finestra temporale di 2 secondi.
- Feature di Contenuto: sono feature che individuano comportamenti sospetti all'interno di porzioni di dati. Gli attacchi R2L e U2R non hanno dei pattern di intrusione molto frequenti, perché questi sono incorporati nelle porzioni di dati dei pacchetti e coinvolgono normalmente solo una singola connessione, mentre gli attacchi DoS e Probing coinvolgono molte connessioni verso pochi host in uno stretto intervallo di tempo. Dunque per individuare questi attacchi meno evidenti si sfruttano queste feature come, ad esempio, il numero di tentativi di accesso falliti.

Segue la tabella contenente le feature ed una loro breve descrizione:

Feature	Dominio	Descrizione	
duration	real	Durata della connessione	
protocol_type	{tcp,udp, icmp}	Tipo di protocollo utilizzato	
service	*	Servizio utilizzato dalla rete di destinazione	
flag	{ OTH, REJ, RSTO, RSTOS0, RSTR, S0, S1,	Stato della connessione (Errore o Normale)	

Feature	Dominio	Descrizione	
	S2, S3, SF, SH }		
src_bytes	real	Numero di byte di dati trasferiti dalla sorgente alla destinazione	
dst_bytes	real	Numero di byte di dati trasferiti dalla destinazione alla sorgente	
land	{0, 1}	Se porta e indirizzo IP di sorgente e destinazione sono uguali, verrà impostato su 1, altrimenti 0	
wrong_fragment	real	Numero totale di frammenti errati in una connessione	
urgent	real	Numero di pacchetti urgenti (pacchetti con bit urgente attivato)	
hot	real	Numero di indicatori "hot", ovvero ingressi	

Feature	Dominio	Descrizione	
		in una directory di sistema	
num_failed_logins	real	Numero di tentativi di accesso falliti	
logged_in	{0, 1}	Mostra lo stato di accesso (1 - accesso riuscito, 0 - altrimenti)	
num_compromised	real	Numero di condizioni compromesse	
root_shell	real	Mostra lo stato della shell di root (1 se ottenuta la shell di root, altrimenti 0)	
su_attempted	real	Impostato su 1 se viene utilizzato il comando "su_root", altrimenti impostato su 0	
num_root	real	Numero di operazioni effettuate come root	
num_file_creations	real	Numero di operazioni di creazione di file	

Feature	Dominio	Descrizione	
num_shells	real	Numero di prompt shell in una connessione	
num_access_files	real	Numero di operazioni sui file di controllo degli accessi	
num_outbound_cmds	real	Numero di comandi in uscita in una sessione ftp	
is_host_login	{0, 1}	Se l'accesso avviene come root o admin, questo viene impostato su 1, altrimenti 0	
is_guest_login	{0, 1}	Impostato su 1 se l'accesso avviene come ospite, altrimenti 0	
count	real	Numero di connessioni allo stesso host di destinazione	
srv_count	real	Numero di connessioni allo stesso servizio (numero di porta)	

Feature	Dominio	Descrizione	
serror_rate	real	Percentuale di connessioni con flag attivato (#4) s0, s1, s2 o s3, tra le connessioni aggregate nel conteggio (#23)	
srv_serror_rate	real	Percentuale di connessioni con flag attivato (#4) s0, s1, s2 o s3, tra le connessioni aggregate nel conteggio srv (#24)	
rerror_rate	real	Percentuale di connessioni con flag attivato (#4) REJ, tra le connessioni aggregate nel conteggio (#23)	
srv_rerror_rate	real	Percentuale di connessioni con flag attivato (#4) REJ, tra le connessioni aggregate nel conteggio srv (#24)	
same_srv_rate	real	Percentuale di connessioni	

Feature	Dominio	Descrizione	
		agli stessi servizi, tra le connessioni aggregate nel conteggio (#23)	
diff_srv_rate	real	Percentuale di connessioni a servizi diversi, tra le connessioni aggregate nel conteggio (#23)	
srv_diff_host_rate	real	Percentuale di connessioni a macchine di destinazione diverse tra le connessioni aggregate nel conteggio srv (#24)	
dst_host_count	real	Numero di connessioni con lo stesso indirizzo IP dell'host di destinazione	
dst_host_srv_count	real	Numero di connessioni con lo stesso numero di porta	
dst_host_same_srv_rate	real	Percentuale di connessioni allo stesso	

Feature	Dominio	Descrizione	
		servizio tra le connessioni aggregate nel conteggio dell'host di destinazione (#32)	
dst_host_diff_srv_rate	real	Percentuale di connessioni a un servizio diverso tra le connessioni aggregate nel conteggio dell'host di destinazione (#32)	
dst_host_same_src_port_rate	real	Percentuale di connessioni alla stessa porta di origine tra le connessioni aggregate nel conteggio srv dell'host di destinazione (#33)	
dst_host_srv_diff_host_rate	real	Percentuale di connessioni a macchine di destinazione diverse tra le connessioni aggregate nel	

Feature	Dominio	Descrizione	
		conteggio dthtt(#33)	
dst_host_serror_rate	real	Percentuale di connessioni con flag attivato (#4) s0, s1, s2 o s3, tra le connessioni aggregate nel conteggio dell'host di destinazione (#32)	
dst_host_srv_serror_rate	real	Percentuale di connessioni con flag attivato (#4) s0, s1, s2 o s3, tra le connessioni aggregate nel conteggio srv dell'host di destinazione (#33)	
dst_host_rerror_rate	real	Percentuale di connessioni con flag attivato (#4) REJ, tra le connessioni aggregate nel conteggio dell'host di destinazione (#32)	

Feature	Dominio	Descrizione	
dst_host_srv_rerror_rate	real	Percentuale di connessioni con flag attivato (#4) REJ, tra le connessioni aggregate nel conteggio srv dell'host di destinazione (#32)	

Per ultima abbiamo la feature Attack nel cui dominio appaiono tutte le tipologie di attacchi modellati dal dataset, tutti questi attacchi verrano poi mappati, durante la fase di preprocessing, nelle 4 classi di attacchi illustrate in precedenza, come mostrato nella tabella seguente:

Attack Class	Attack Type
DoS	Back, Land, Neptune, Pod, Smurf, Teardrop, Apache2, Udpstorm, Processtable, Worm
Probe	Satan, Ipsweep, Nmap, Portsweep, Mscan, Saint
R2L	Guess_Password, Ftp_write, Imap, Phf, Multihop, Warezmaster, Warezclient, Spy, Xlock, Xsnoop, Snmpguess, Snmpgetattack, Http_tunnel, Sendmail, Named
U2R	Buffer_overflow, Loadmodule, Rootkit, Perl, Sqlattack, Xterm, Ps

Il dataset presenta inoltre un ulteriore colonna che rappresenta una valutazione di difficoltà dell'esempio (da 0 a 21), poiché attualmente non è di utilità al progetto si è scelto di ometterla.

EDA

Prima di poter costruire efficaci sistemi basati sulla conoscenza, è essenziale comprendere in profondità i dati a disposizione. Qui entra in gioco l'Analisi Esplorativa dei Dati (EDA, dall'inglese Exploratory Data Analysis).

L'EDA rappresenta una tappa fondamentale nella pre-elaborazione e analisi dei dati. Essa serve per comprendere a fondo la struttura dei dati al fine di identificare pattern e anomalie e individuare e gestire valori mancanti o irregolari. Questo processo può anche svelare importanti correlazioni e tendenze nei dati, fornendo indicazioni preziose per la modellazione.

In un'era dominata dai dati, la capacità di esplorare e comprendere questi dati diventa sempre più vitale, rendendo l'EDA una competenza inestimabile nell'ingegneria della conoscenza.

> Di seguito vengono riportate solo le rilevazioni effettuate durante la suddetta analisi, per vedere il codice fare riferimento al notebook

Il dataset è stato a monte già suddiviso dagli sviluppatori in set di addestramento e set di test, di seguito parallelamente esploreremo entrambi i dataset.

Il training set è composto da 125973 esempi per 43 colonne, non sono presenti tuple ripetute e non vi è assenza di valori. Delle 43 feature solo 4 sono di tipo categorico mentre le restanti sono di tipo continuo. Similmente il test set presenta 22544 per 43 colonne e non presenta esempi ripetuti o valori mancanti.

> Di seguito si analizzano in dettagli le peculiarità del dataset in parallelo tra training set e test set:
>
> - qualora non fosse specificato si starà parlando del training set
> - quando non viene riportato il test set è perchè non esistono differenze sostanziali con il training set e per questo motivo viene omesso

La classe Target

La feature obiettivo scelta per il task di classificazione sarà la feature `attack_class`. Questa feature è stata aggiunta in fase di analisi e presenta i valori assunti dalla feature `attack` mappati nelle 4 categorie precedentemente discusse (Normal, DoS, Probe, R2L, U2R).

Questa scelta di raggruppamento è utile a generare un task di classificazione più generale e quindi a migliorare l'efficienza dei modelli che si andranno a costruire.

La prima scelta progettuale che si è dovuta prendere è stata: scegliere se trasformare o meno il task di classificazione in uno di tipo binario. Infatti si potrebbe ridurre la nostra feature target ad una di tipo binario semplicemente facendo solo distinzione tra connessioni normali e connessioni anomale

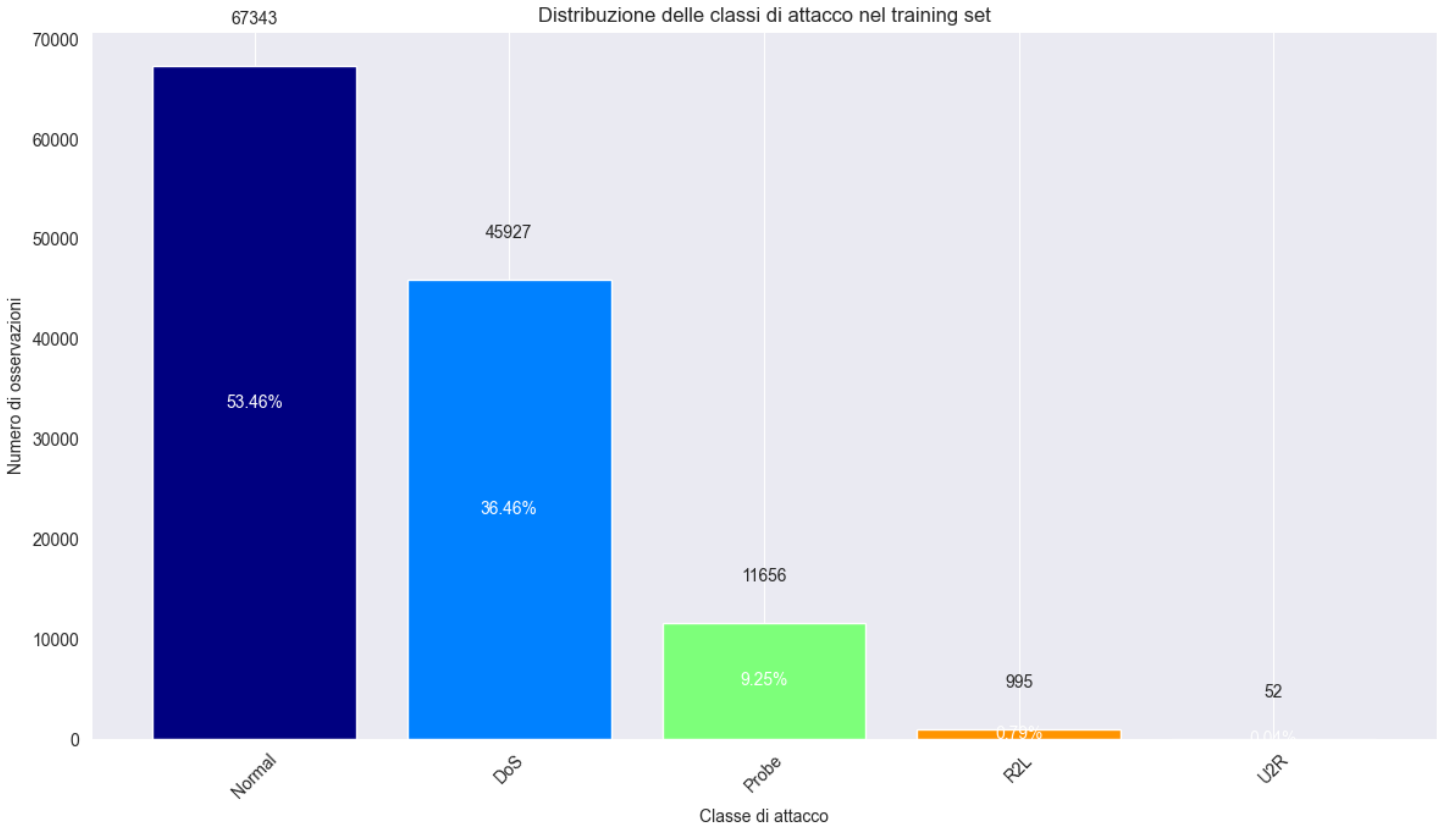
La scelta di utilizzare un classificatore binario porterebbe dei notevoli vantaggi:

- **Semplicità:** Un classificatore che distingue tra traffico normale e anomalo è generalmente più semplice da realizzare.
- **Velocità:** più veloce da addestrare e richiederebbe meno risorse in fase di esecuzione.
- **Generalizzazione:** Può essere più robusto contro le varianti non note di attacchi, dato che non è focalizzato sulla distinzione fine tra le tipologie.

La scelta dunque andrebbe operata in virtù di quello che poi rappresenterà il progetto finale. Se il sistema è destinato a fornire rapidamente e semplicemente una distinzione tra attività potenzialmente dannose e attività legittime, un classificatore binario potrebbe essere sufficiente. Tuttavia, se l'IDS deve essere implementato in un ambiente altamente sensibile dove tipi specifici di attacchi richiedono risposte immediate e mirate, allora un classificatore multi-classe è più indicato.

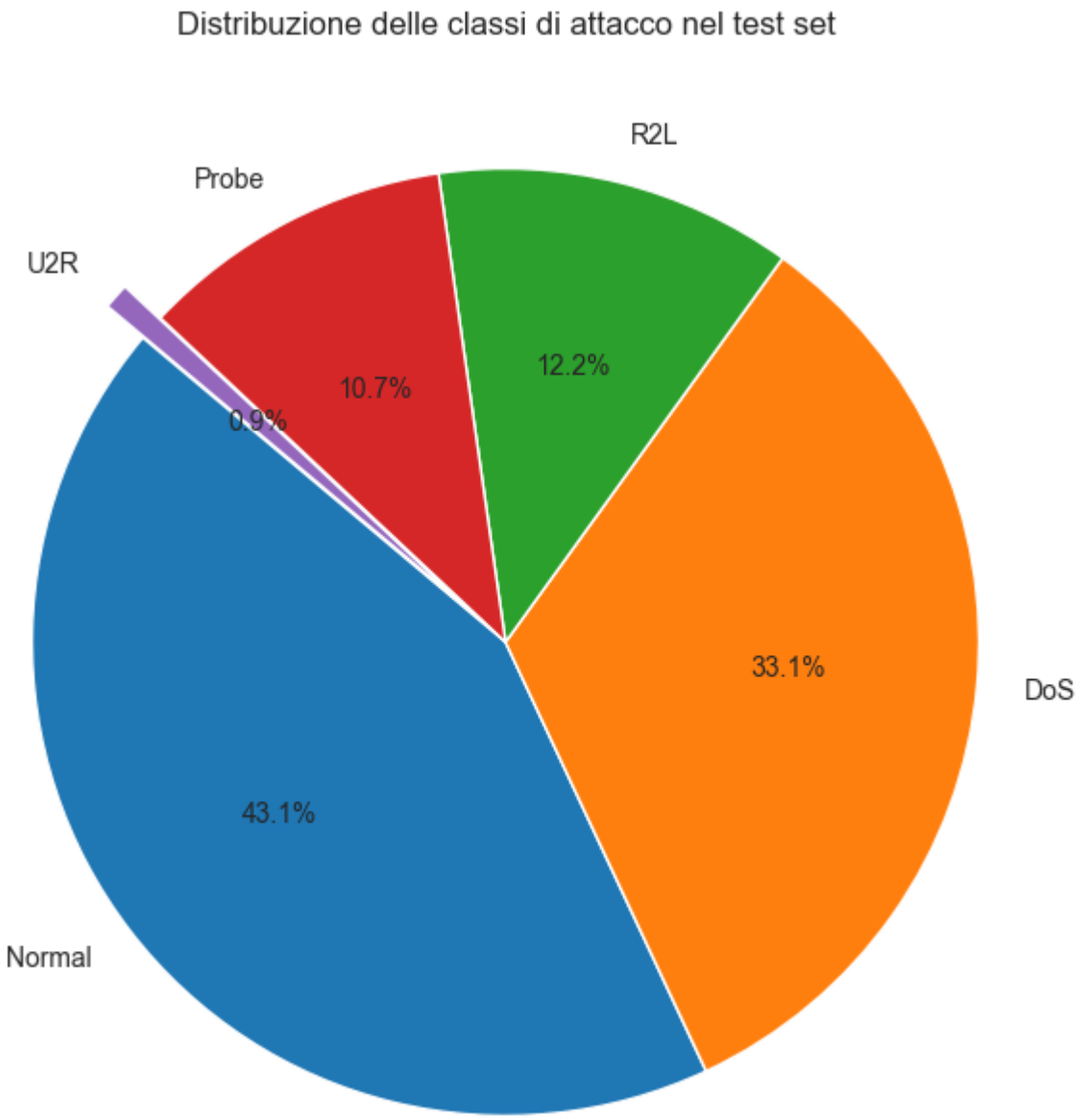
Questo progetto, come illustrato nell'introduzione, assume il carattere di un'esperienza didattica e non è orientato verso una distribuzione pratica. La decisione sulla direzione da prendere è stata intrapresa considerando la sfida che si presenta a chi redige il presente documento. Pertanto, in virtù del fatto che la letteratura offre una vasta gamma di esempi di classificatori binari in questo dominio, si è optato per esplorare l'approccio più complesso e meno trattato della classificazione multi-classe.

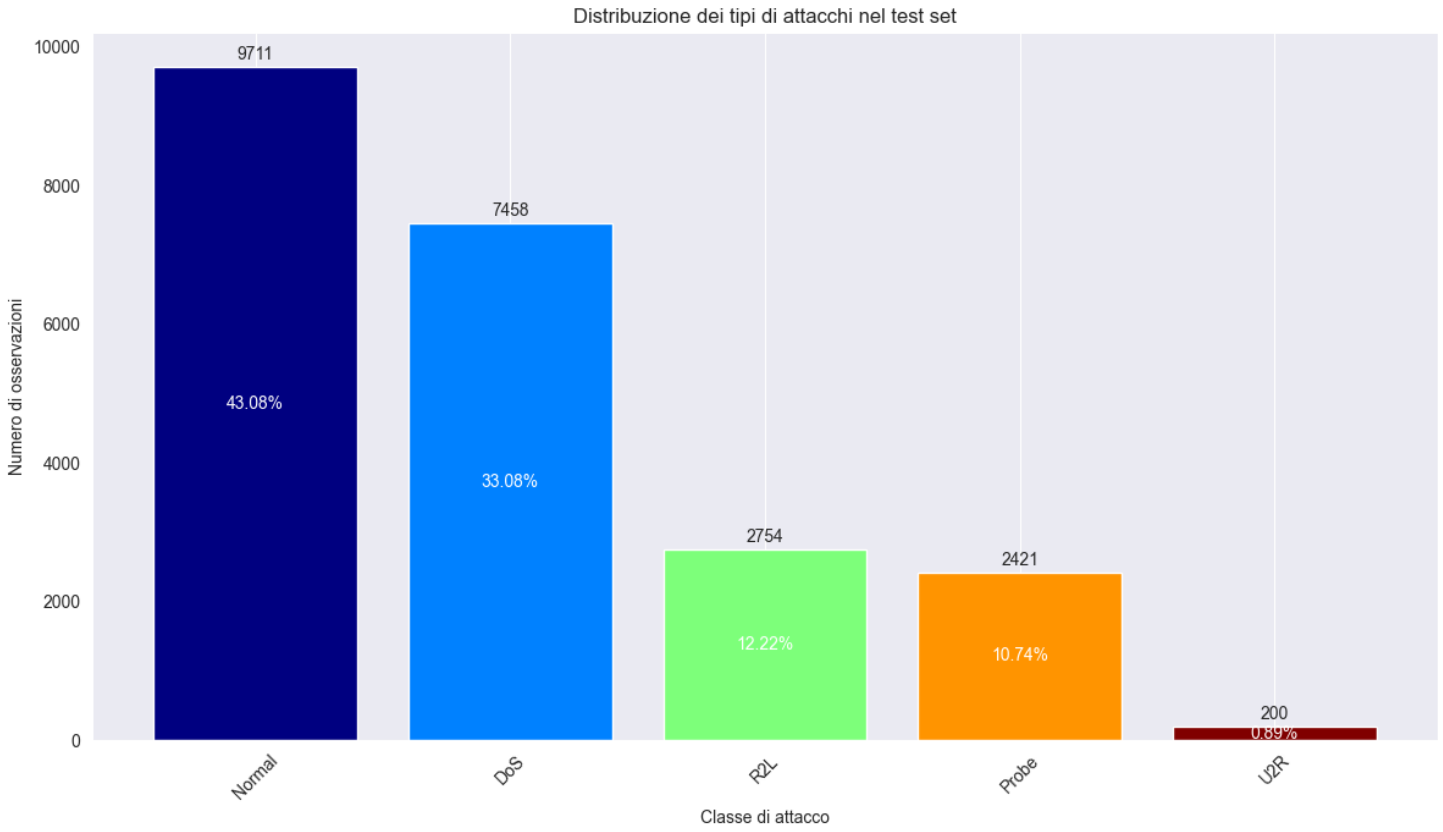
Distribuzione delle classi di attacco



Dall'istogramma fornito, possiamo trarre alcune osservazioni interessanti riguardo alla distribuzione dei tipi di attacco nel dataset NSL-KDD, in particolare:

- La maggior parte delle osservazioni rientra categoria Normal, indicando che la maggior parte del traffico di rete nel dataset è benigno.
- La significativa presenza di attacchi DoS (seconda categoria più presente) nel dataset suggerisce che questi attacchi potrebbero essere comuni o che il dataset potrebbe essere stato particolarmente focalizzato sulla cattura di tali attacchi.
- Gli attacchi di tipo "Probe" sono tentativi di scansionare una rete per identificare possibili vulnerabilità senza necessariamente sfruttarle. Questa categoria rappresenta una frazione minore del dataset rispetto ai "Normal" e ai "DoS", ma la sua presenza indica l'importanza di rilevare e prevenire tali tentativi di scansione.
- Le categorie "R2L" (Remote to Local) e "U2R" (User to Root) hanno una presenza molto bassa nel dataset, questo perchè sono meno facilmente rilevabili rispetto ad altri tipi di attacchi. Infatti questa è proprio la tipologia di attacchi preferita dagli hacker





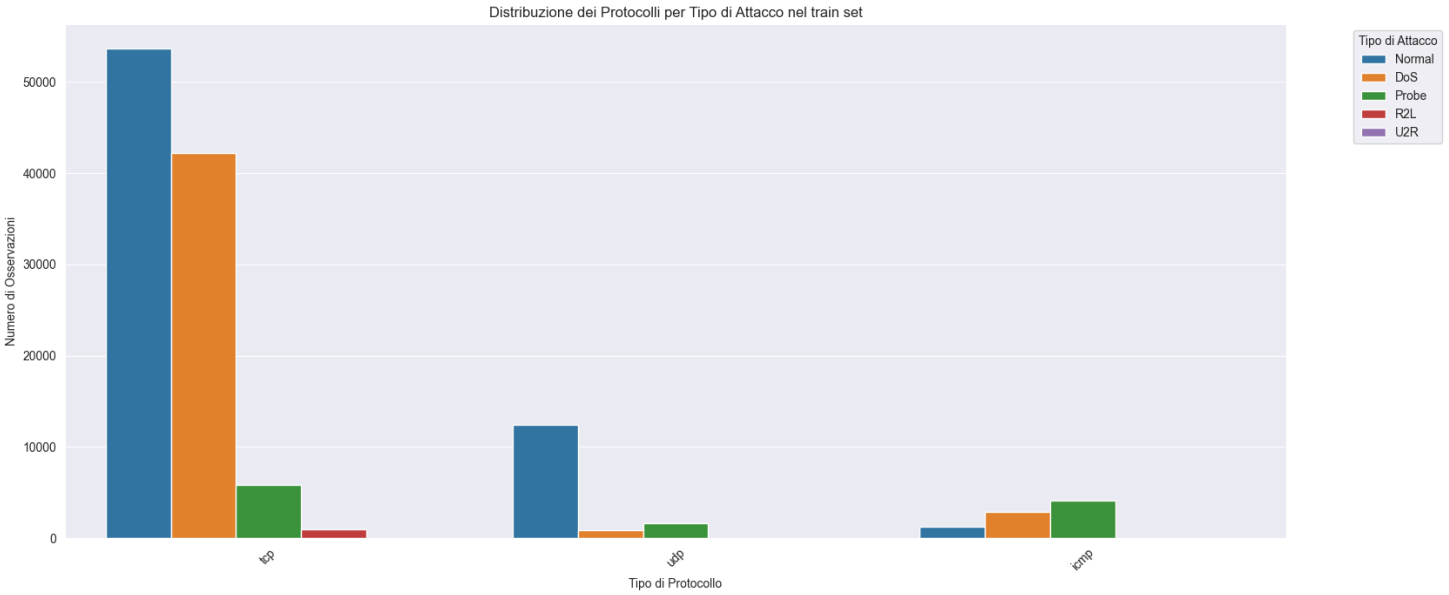
Per quanto riguarda il test set valgono grossomodo le riflessioni effettuate per l'insieme di addestramento, chiaramente in quest'ultimo aumentano gli esempi di comportamenti anomali poiché è sul riconoscere questi che il classificatore deve lavorare.

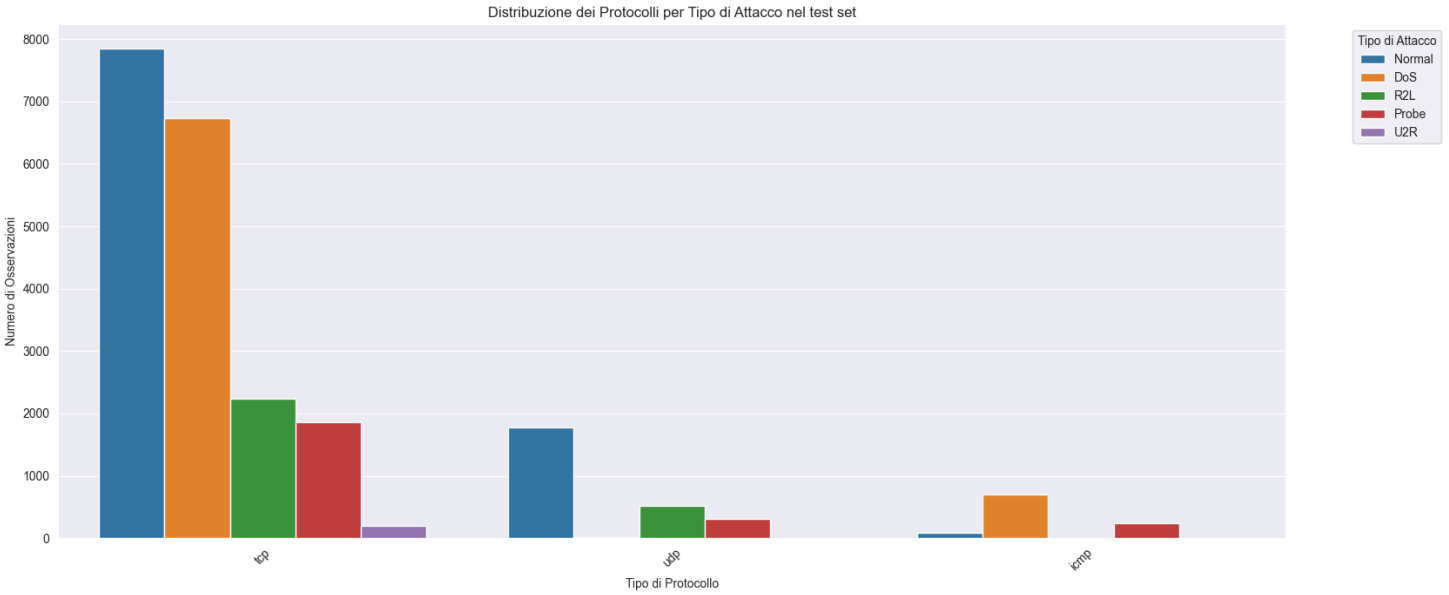
Date le distribuzioni, è evidente che il dataset non è bilanciato, questo è uno dei primi problemi che ha dato la scelta del classificatore multiclasse, infatti riunendo tutte le connessioni anomale, si sarebbe parzialmente risolto lo sbilanciamento.

Feature categoriche

Si prosegue l'analisi analizzando le feature categoriche per cercare di individuare quali feature potrebbero essere più importanti in fase di addestramento dei modelli.

Tipo di protocollo





Dagl'istogrammi presentati, emerge chiaramente come la gran parte degli attacchi provenga dal protocollo TCP. Ci sono diverse ragioni che spiegano questa predominanza. Prima di tutto, la pervasività del TCP: essendo alla base di numerosi servizi essenziali come l'HTTP e le e-mail, rappresenta un bersaglio di grande interesse per gli aggressori, permettendo loro di massimizzare l'effetto dell'attacco. La natura orientata alla connessione del TCP offre agli aggressori l'opportunità di instaurare una comunicazione costante con il sistema bersaglio. In aggiunta, la sua complessità strutturale, che comprende la gestione delle connessioni, il controllo della congestione e il riconoscimento dei pacchetti, può introdurre punti deboli sfruttabili. Infine, la facilità con cui si possono realizzare attacchi come l'iniezione di pacchetti o le intercettazioni man-in-the-middle tramite TCP rende questo protocollo particolarmente esposto a minacce.

Risulta anche interessante vedere come le connessioni di tipo icmp (utilizzate per il ping o il tracerout) sono sfruttate soprattutto per compiere attacchi.

Sulla base dei dati visti e delle considerazioni fatte, sembra che questa feature possa giocare un ruolo significativo nella classificazione degli attacchi, specialmente per quelli con meno esempi a disposizione.

Tipi di attacco

La caratteristica `attack` assume molti valori, per questo motivo al posto di rappresentarla tramite istogramma si è pensato fosse meglio vederla sotto forma di tabella:

	Attacco	**Numero di Esempi**	**Percentuale**
1	normal	67343	53.458
2	neptune	41214	32.717
3	satan	3633	2.884
4	ipsweep	3599	2.857
5	portsweep	2931	2.327
6	smurf	2646	2.1
7	nmap	1493	1.185
8	back	956	0.759
9	teardrop	892	0.708
10	warezclient	890	0.707
11	pod	201	0.16
12	guess_passwd	53	0.042
13	buffer_overflow	30	0.024
14	warezmaster	20	0.016
15	land	18	0.014
16	imap	11	0.009
17	rootkit	10	0.008

	Attacco	Numero di Esempi	Percentuale
18	loadmodule	9	0.007
19	ftp_write	8	0.006
20	multihop	7	0.006
21	phf	4	0.003
22	perl	3	0.002
23	spy	2	0.002

Nella tabella sopra abbiamo 23 tipi differenti di attacchi e le loro distribuzioni. Abbiamo un'ulteriore conferma del fatto che il dataset non è distribuito uniformemente. Esistono, infatti, numerosi attacchi che compaiono solo in un numero molto limitato di esempi mentre, attacchi come "normal" e "Neptune" compaiono da soli nell'85% degli esempi. Ci sono 16 attacchi, su un totale di 23, in cui i punti dati rappresentano meno dell'1%.

Risulta interessante, invece, guardare gli attacchi presenti nel test set:

	Attacco	Numero di Esempi	Percentuale
1	normal	9711	43.076
2	neptune	4657	20.657
3	guess_passwd	1231	5.46
4	mscan	996	4.418
5	warezmaster	944	4.187
6	apache2	737	3.269
7	satan	735	3.26
8	processtable	685	3.039
9	smurf	665	2.95
10	back	359	1.592
11	snmpguess	331	1.468
12	saint	319	1.415

	Attacco	Numero di Esempi	Percentuale
13	mailbomb	293	1.3
14	snmpgetattack	178	0.79
15	portsweep	157	0.696
16	ipsweep	141	0.625
17	httptunnel	133	0.59
18	nmap	73	0.324
19	pod	41	0.182
20	buffer_overflow	20	0.089
21	multihop	18	0.08
22	named	17	0.075
23	ps	15	0.067
24	sendmail	14	0.062
25	rootkit	13	0.058
26	xterm	13	0.058
27	teardrop	12	0.053
28	xlock	9	0.04
29	land	7	0.031
30	xsnoop	4	0.018
31	ftp_write	3	0.013
32	worm	2	0.009
33	loadmodule	2	0.009
34	perl	2	0.009
35	sqlattack	2	0.009
36	udpstorm	2	0.009

	Attacco	Numero di Esempi	Percentuale
37	phf	2	0.009
38	imap	1	0.004

Come possiamo vedere dalla tabella, abbiamo 38 tipi di attacchi, ben 15 in più del training set. Per il resto valgono le stesse considerazioni fatte per il set di addestramento, normal e neptune restano i valori con più esempi. Sebbene, in questo caso, gli esempi siano un po' meglio distribuiti tra i diversi attacchi, resta comunque un evidente sbilanciamento.

Per quanto riguarda la presenza di più tipi di attacchi nel test set, questa è una pratica comune, infatti ciò può riflettere la volatilità e l'evoluzione delle minacce informatiche, oltre a simulare condizioni realistiche in cui un sistema deve rilevare attacchi sconosciuti. Dunque si tratta di una scelta deliberata per valutare l'abilità del modello di generalizzare e per assicurare che sia robusto di fronte a minacce non ancora note durante l'addestramento.

Possiamo quindi osservare quali sono gli attacchi solo presenti nel test set:

Attacchi solo nel Test Set
sqlattack
ps
mailbomb
xlock
mscan
worm
snmpgetattack
sendmail
udpstorm
apache2
xterm
snmpguess
processtable

Attacchi solo nel Test Set
saint
httptunnel
xsnoop
named

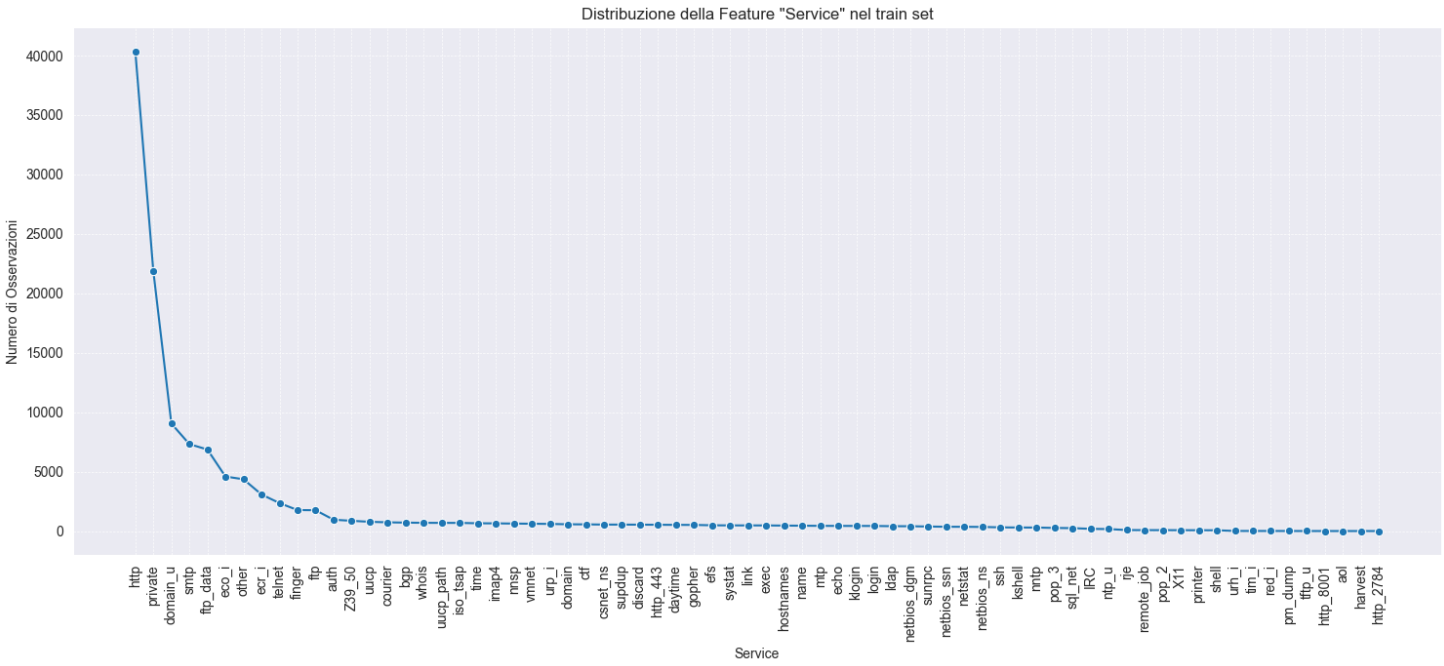
Come possiamo vedere ci sono 17 nuovi tipi di attacco, ciò però va contro quello che avevamo evidenziato prima, ovvero che la differenza tra gli attacchi nel training set e quelli nel test set fosse di 15, dunque risulta probabile che anche il training set abbia tipi di attacco che non sono presenti nel test set, infatti:

Attacchi solo nel Training Set
warezclient
spy

La presenza di tipi di attacco nel training set che non compaiono nel test set potrebbe sembrare controintuitiva, dato che, come si è detto, ci si aspetta il contrario. Le ragioni possono essere disparate, escludendo eventuali errori degli sviluppatori, potrebbe essere dovuto alle tecniche di campionamento e selezione dei dati, oppure alle strategie utilizzate proprio per la divisione del dataset. Ad ogni modo questa curiosità non influenza più di tanto il nostro percorso.

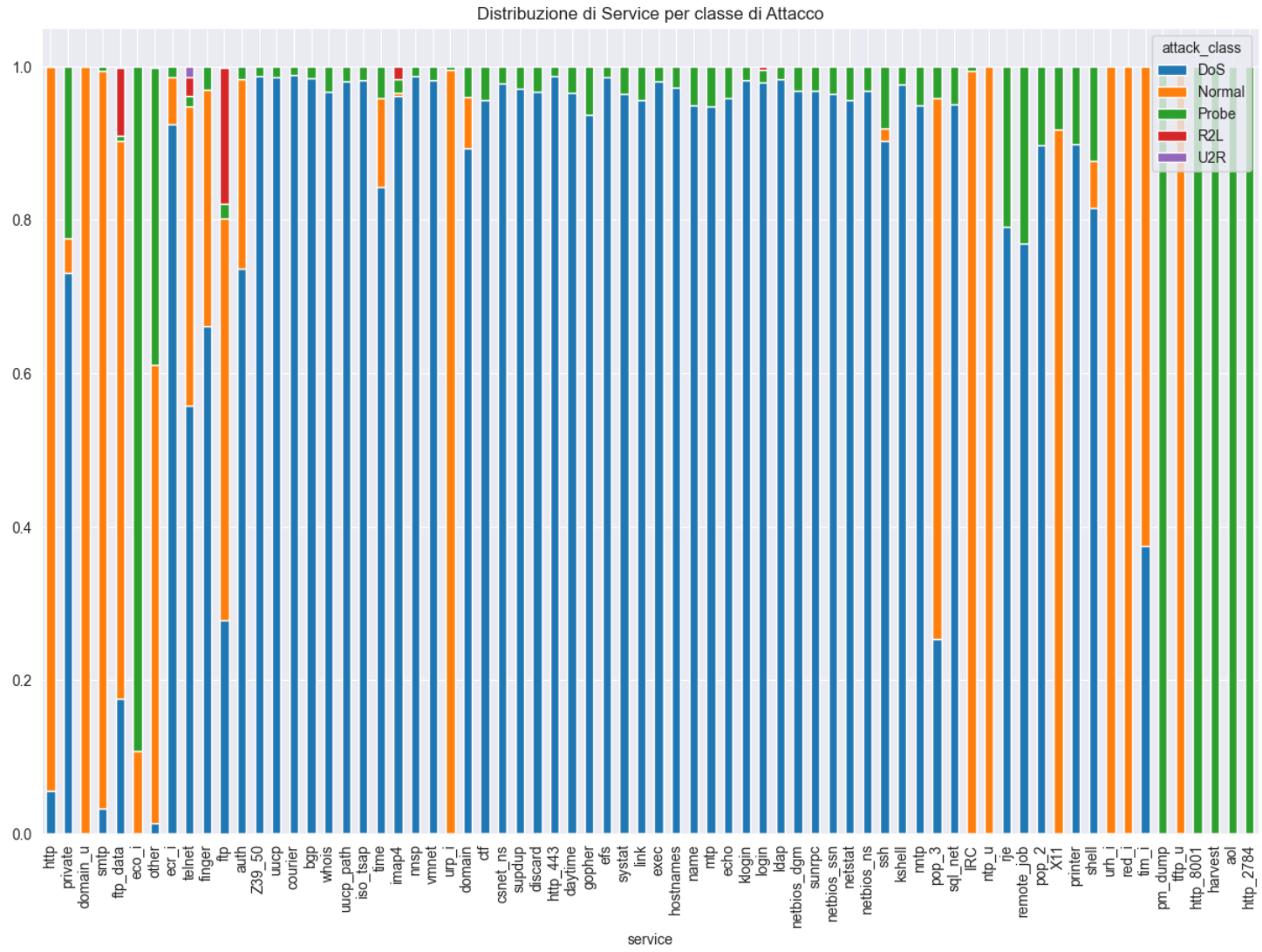
Poiché si è scelto di raggruppare i tipi specifici di attacco nelle quattro categorie descritte in precedenza, si è deciso di rimuovere la feature qui osservata. In questo modo possiamo prevenire una forma di ridondanza ed evitare un potenziale Information Leakage, che potrebbe fornire un bias in fase di addestramento di un modello. Infatti conoscere il tipo di attacco è una discriminante per identificare la classe di appartenenza, ma in condizioni reali non si ha disposizione l'etichetta del tipo d'attacco.

Service



Come ci si poteva aspettare solo alcuni servizi sono effettivamente utilizzati, guardando la distribuzione probabilmente solo i primi valori saranno utili alla classificazione, questo, se dimostrato, garantirebbe la possibilità di ridurre il dominio di questa feature a possibilmente i primi 10 valori per frequenza.

Il fatto che la maggior parte delle osservazioni sia addensata sul lato sinistro può però portare a delle problematiche in termini di classificazione, infatti il modello potrebbe essere esposto in modo sproporzionato solo verso alcuni valori. Tuttavia questa distribuzione potrebbe anche indicare come la maggior parte delle osservazioni ha un comportamento "tipico", mentre solo alcune osservazioni mostrano un comportamento "atipico". Conoscendo il dominio, possiamo affermare con certezza che questa distribuzione rappresenta la realtà, banalmente vedere il servizio HTTP come il più frequente è un qualcosa di aspettato.

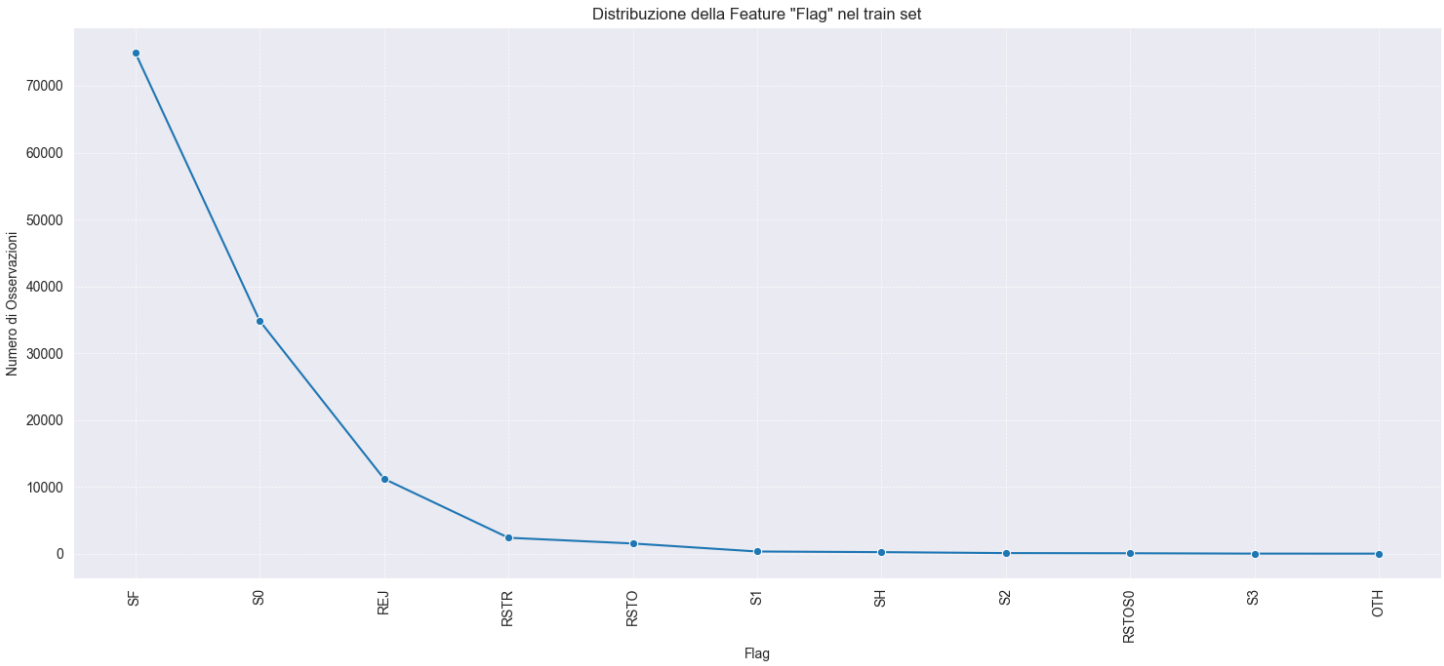


Attraverso il grafico a barre impilate qui sopra possiamo visualizzare la distribuzione congiunta di `service` rispetto alla feature target `attack_class` , questa ci aiuta a comprendere alcune cose riguardanti questa feature:

- diversi servizi presentano prevalentemente un colore , questo indica che quel particolare servizio è fortemente associato ad un certo tipo di attacco
    - le connessioni che utilizzano il servizio `private` sono per lo più utilizzate per atttacchi Sostanzialmente possiamo riuscire ad identificare più facilmente quali servizi sono particolarmente vulnerabili o bersagliati da specifici tipi di attacco, fornendo una visione chiara di dove potrebbero essere necessarie ulteriori misure di sicurezza o indagini.

Tutto questo suggerisce come anche questa feature potrebbe essere importante ai fini della classificazione.

Flag



Esattamente come per il caso precedente, anche qui vediamo un addensamento delle osservazioni su pochi valori a sinistra. Anche in questo caso il tutto è verosimile, infatti il flag SF sta per "Successful Finish" e indica che la connessione è stata stabilita con successo e terminata normalmente, dunque è ovvio che sia il più frequente. Allo stesso modo il fatto che S0 sia il secondo più frequente rispecchia la topologia dei dati, infatti questo flag indica "connessione tentata ma non stabilita" ed è una condizione tipica che si verifica durante un attacco di tipo DoS, quando si tenta di esaurire le risorse disponibili di un server inviando tante richieste di connessione (SYN) senza però mai stabilire una connessione (ACK). Allo stesso modo anche REJ e RSTR corrispondono per lo più a connessioni malevole.

Feature Continue

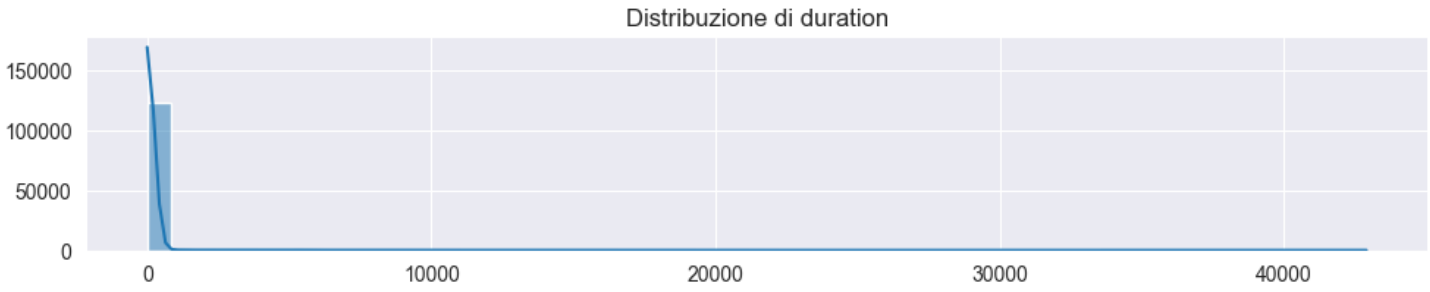
Per prima cosa tra le feature continue si sono distinte le feature binarie:

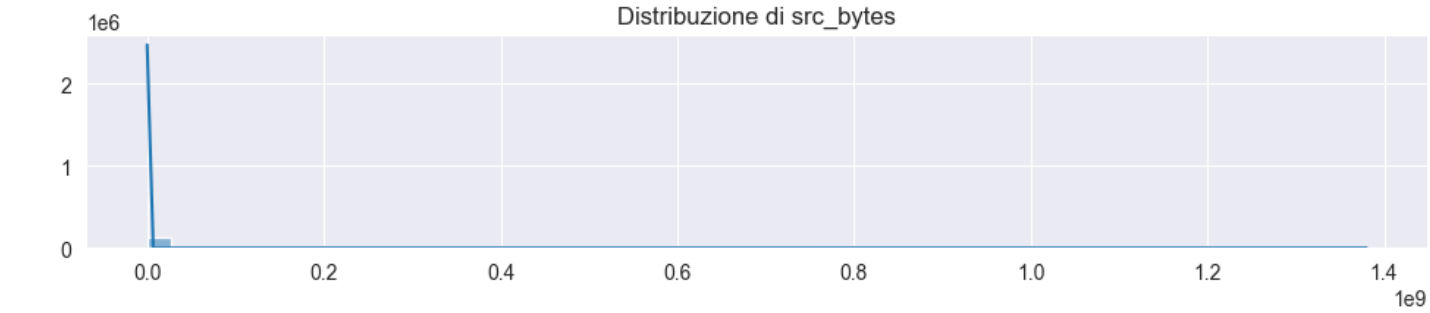
> Feature binarie: `['land', 'logged_in', 'root_shell', 'is_host_login', 'is_guest_login']`

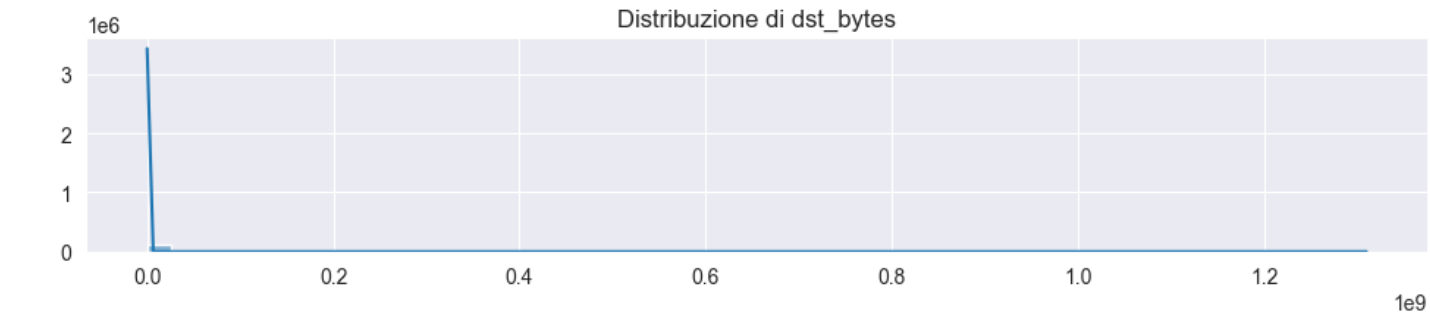
Queste sono considerabili come variabili discrete che possono assumere il valore di 0 o 1.

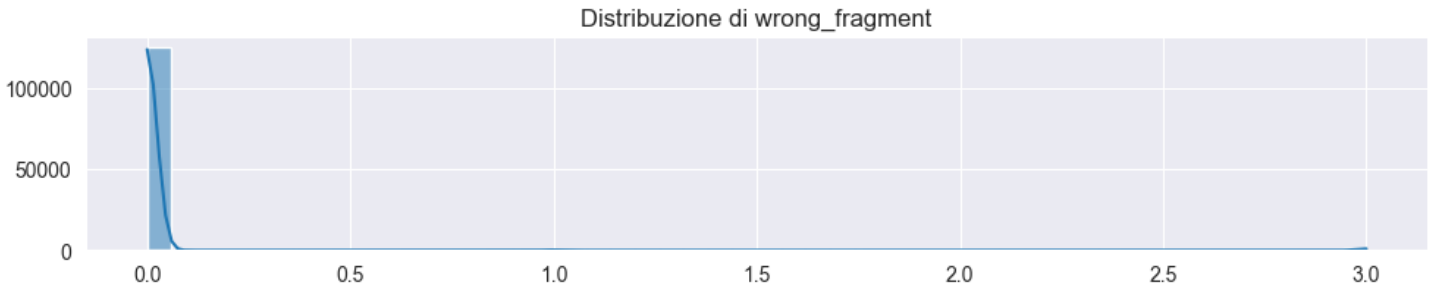
Si è notato che la feature `num_outbound_cmds` è composta da soli 0, quindi si è deciso di rimuoverla.

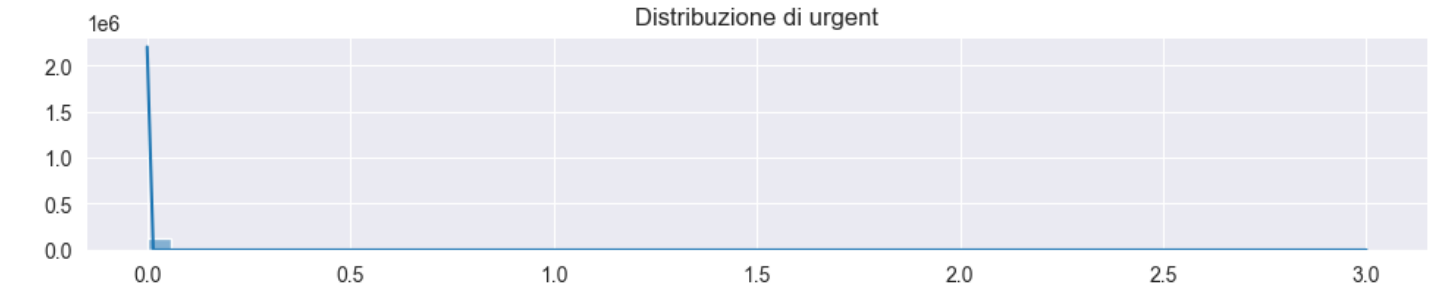
Visto l'elevato numero di caratteristiche continue si inizia analizzando gli istogrami di un sottogruppo di queste, per avere un'idea della loro distribuzione

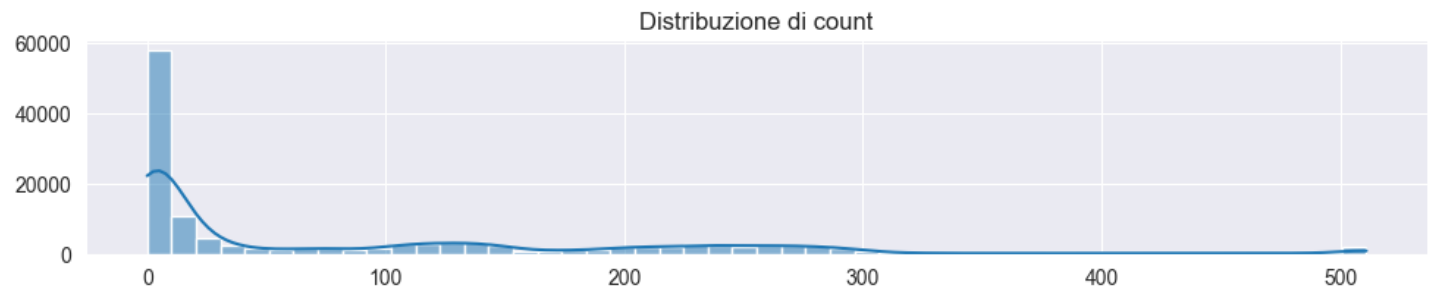

Distribuzione di duration
150000
100000
50000
0
0
10000
20000
30000
40000

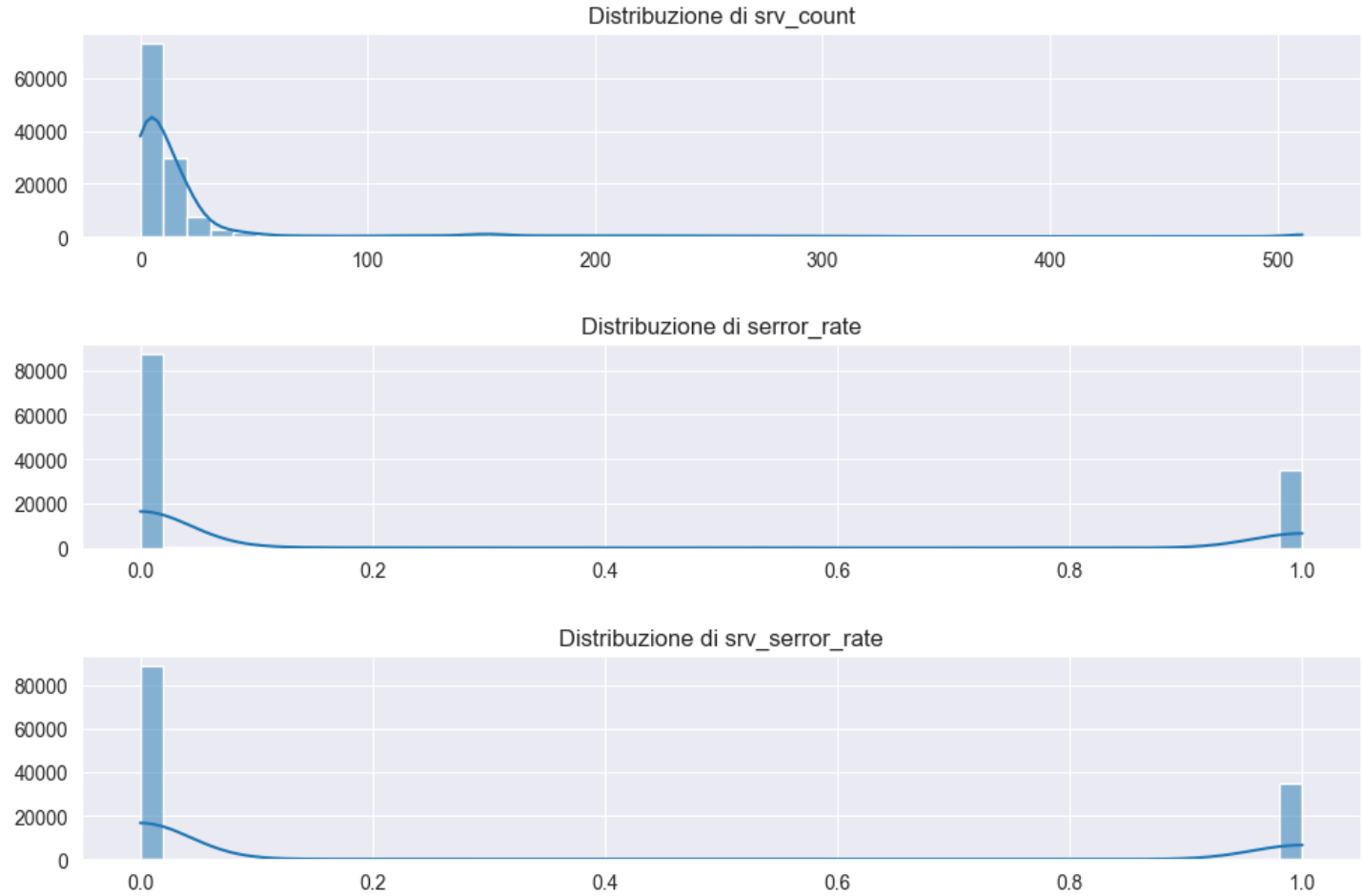

Distribuzione di src_bytes
1e6
2
1
0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1e9


Distribuzione di dst_bytes
1e6
3
2
1
0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1e9


Distribuzione di wrong_fragment
100000
50000
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0


Distribuzione di urgent
1e6
2.0
1.5
1.0
0.5
0.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0


Distribuzione di count
60000
40000
20000
0
0
100
200
300
400
500



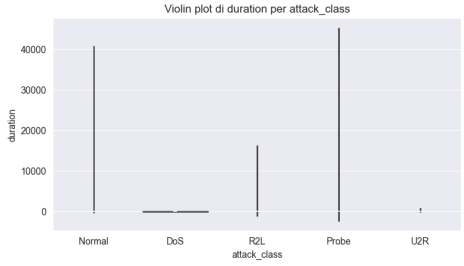
Dagli istogrammi vediamo:

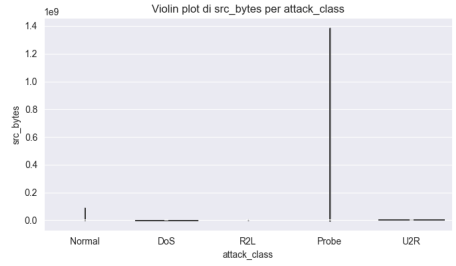
- Molte delle caratteristiche come 'duration', 'src_bytes', 'dst_bytes' e 'wrong_fragment' sono fortemente inclinate verso valori bassi, indicando che la maggior parte delle connessioni hanno brevi durate e coinvolgono piccole quantità di dati.
- Caratteristiche binarie come 'urgent' sono fortemente sbilanciate con quasi tutti i valori a zero, indicando che queste sono occorrenze rare nel traffico di rete.
- Le feature 'count' e 'srv_count', che sono relative a connessioni allo stesso host e servizio, mostrano anche una distribuzione inclinata, con un picco nei valori più bassi.
- Le caratteristiche del tasso di errore ('serror_rate' e 'srv_serror_rate') mostrano distribuzioni bimodali, suggerendo che per molte connessioni, o non c'è errore o c'è un alto tasso di errore, che potrebbe essere indicativo di tipi specifici di attacchi.

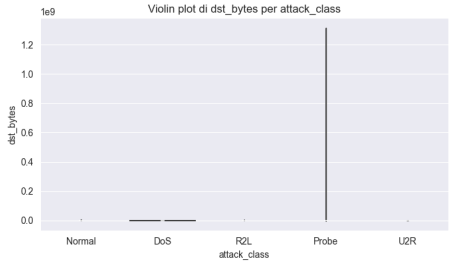
Ora andiamo a visualizzare i Violin Plot delle rimanenti variabili continue. La scelta di visualizzazione è ricaduta sui violin plot perché sono particolarmente utili per visualizzare la distribuzione di una variabile numerica lungo diversi livelli di una variabile categorica:

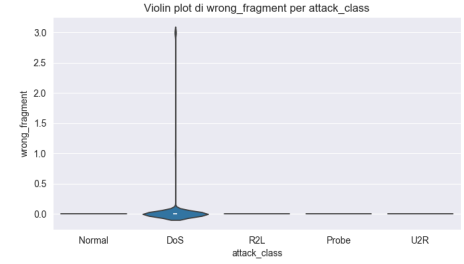
- La parte centrale di un violin plot è simile a un box plot e mostra la mediana (una linea orizzontale) e il primo e terzo quartile (i confini della parte spessa del "violino").

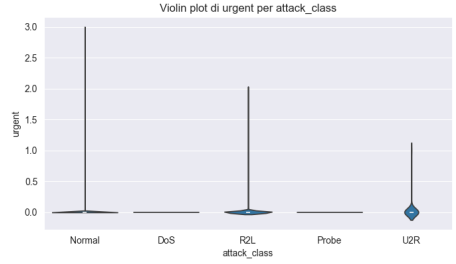
- La forma esterna del violin plot rappresenta una stima della densità kernel della distribuzione dei dati, simmetrica rispetto alla sua asse verticale. Ciò significa che quanto più larga è una sezione del "violino", più dati ci sono in quell'intervallo di valori.
  I violin plot forniscono una visualizzazione della densità dei dati, permettendo di vedere dove si concentrano i valori.

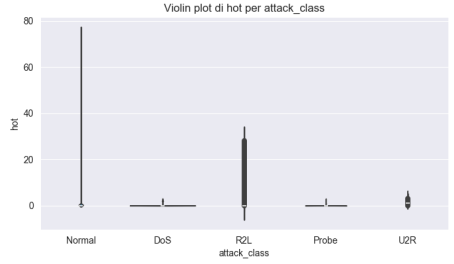

Violin plot di duration per attack_class
duration
attack_class
Normal
DoS
R2L
Probe
U2R

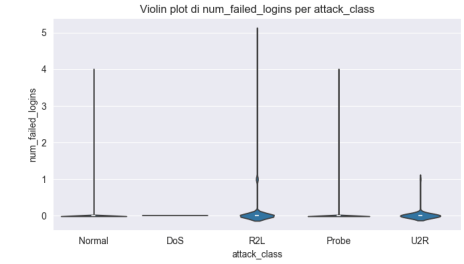

Violin plot di src_bytes per attack_class
1e9
src_bytes
attack_class
Normal
DoS
R2L
Probe
U2R

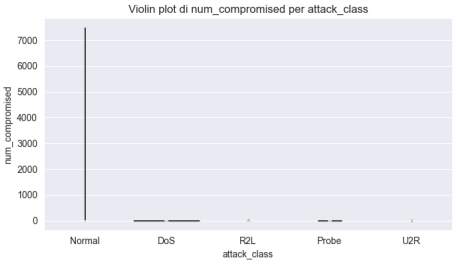

Violin plot di dst_bytes per attack_class
1e9
dst_bytes
attack_class
Normal
DoS
R2L
Probe
U2R

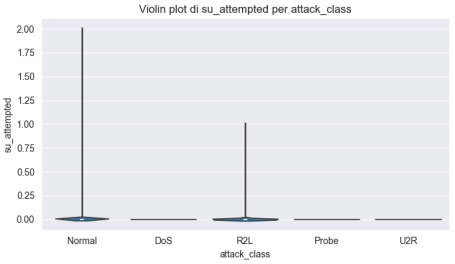

Violin plot di wrong_fragment per attack_class
wrong_fragment
attack_class
Normal
DoS
R2L
Probe
U2R

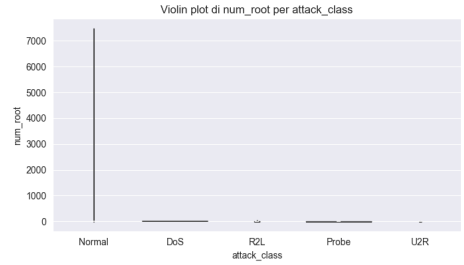

Violin plot di urgent per attack_class
urgent
attack_class
Normal
DoS
R2L
Probe
U2R

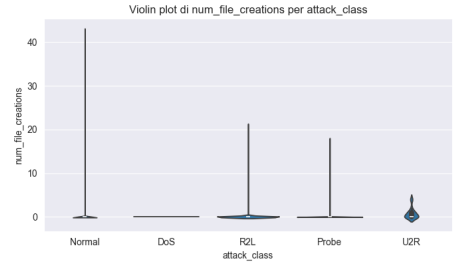

Violin plot di hot per attack_class
hot
attack_class
Normal
DoS
R2L
Probe
U2R

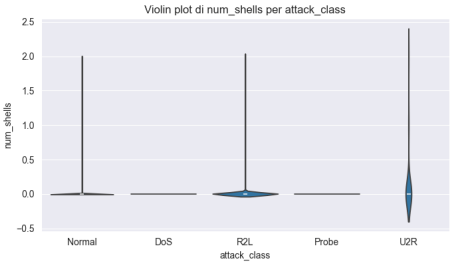

Violin plot di num_failed_logins per attack_class
num_failed_logins
attack_class
Normal
DoS
R2L
Probe
U2R

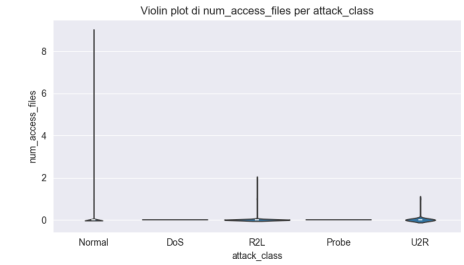

Violin plot di num_compromised per attack_class
num_compromised
attack_class
Normal
DoS
R2L
Probe
U2R

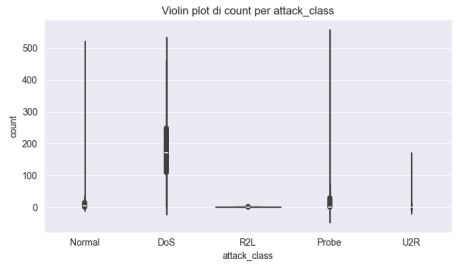

Violin plot di su_attempted per attack_class
su_attempted
attack_class
Normal
DoS
R2L
Probe
U2R

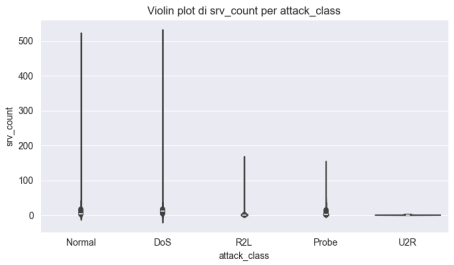

Violin plot di num_root per attack_class
num_root
attack_class
Normal
DoS
R2L
Probe
U2R

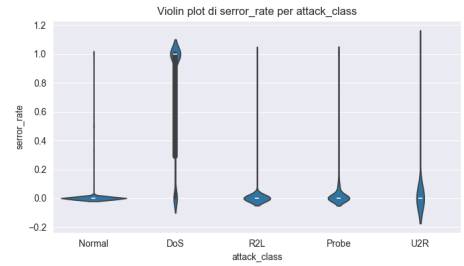

Violin plot di num_file_creations per attack_class
num_file_creations
attack_class
Normal
DoS
R2L
Probe
U2R

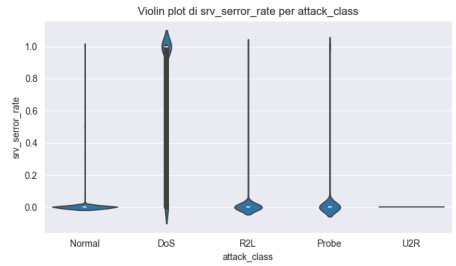

Violin plot di num_shells per attack_class
num_shells
attack_class
Normal
DoS
R2L
Probe
U2R

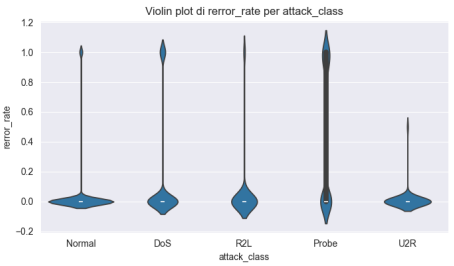

Violin plot di num_access_files per attack_class
num_access_files
attack_class
Normal
DoS
R2L
Probe
U2R

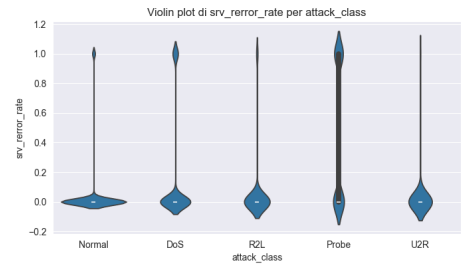

Violin plot di count per attack_class
count
attack_class
Normal
DoS
R2L
Probe
U2R

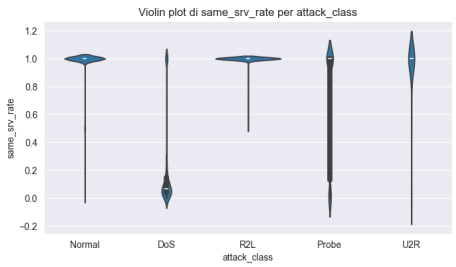

Violin plot di srv_count per attack_class
srv_count
attack_class
Normal
DoS
R2L
Probe
U2R

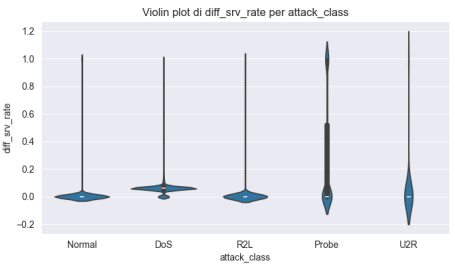

Violin plot di serror_rate per attack_class
serror_rate
attack_class
Normal
DoS
R2L
Probe
U2R

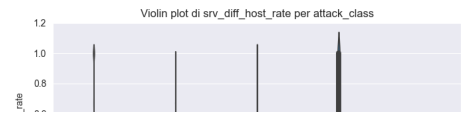

Violin plot di srv_serror_rate per attack_class
srv_serror_rate
attack_class
Normal
DoS
R2L
Probe
U2R

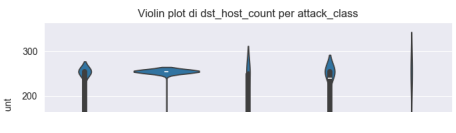

Violin plot di rerror_rate per attack_class
rerror_rate
attack_class
Normal
DoS
R2L
Probe
U2R

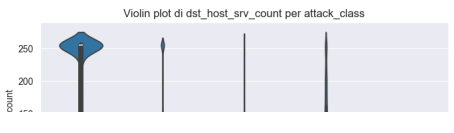

Violin plot di srv_rerror_rate per attack_class
srv_rerror_rate
attack_class
Normal
DoS
R2L
Probe
U2R

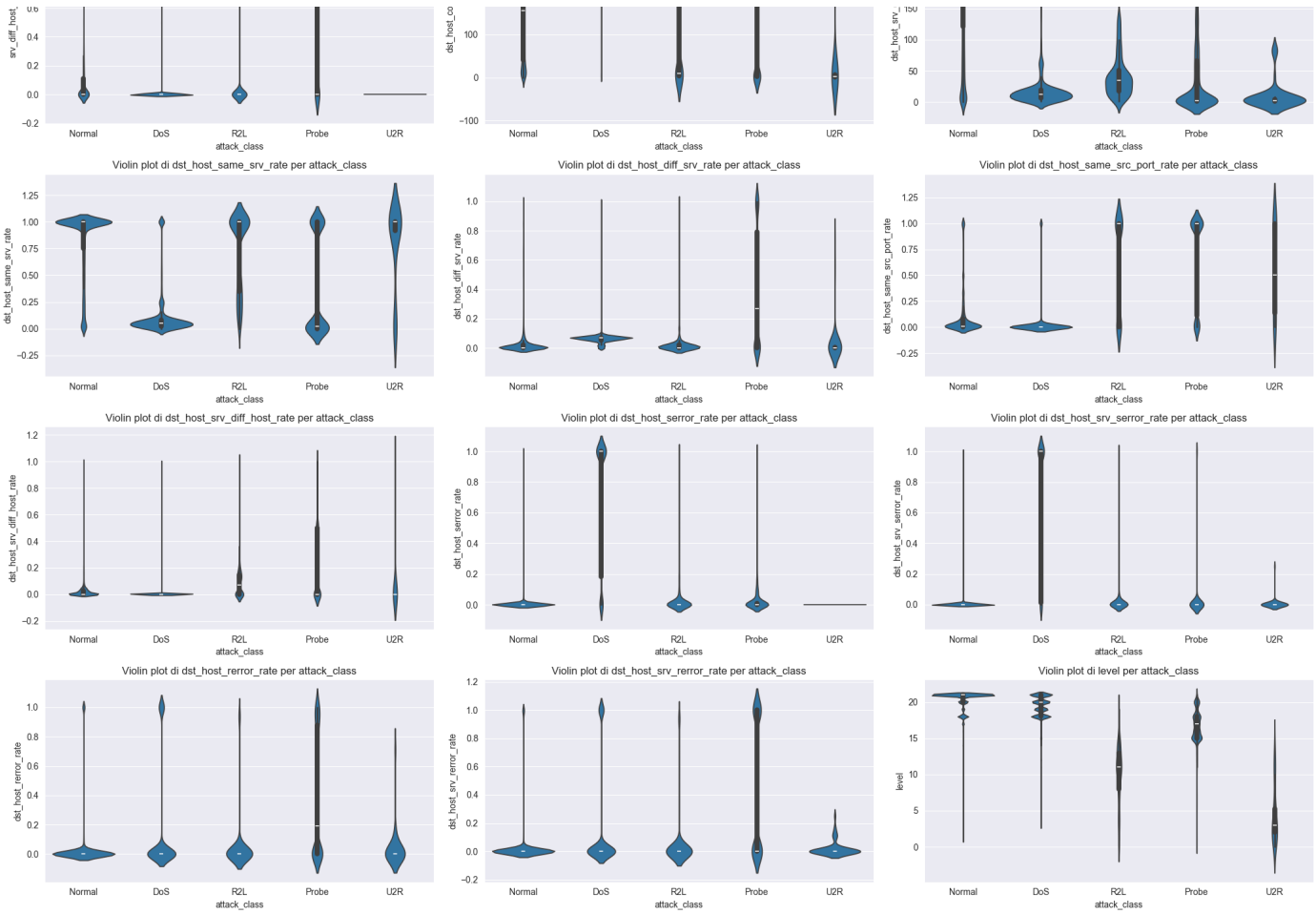

Violin plot di same_srv_rate per attack_class
same_srv_rate
attack_class
Normal
DoS
R2L
Probe
U2R


Violin plot di diff_srv_rate per attack_class
diff_srv_rate
attack_class
Normal
DoS
R2L
Probe
U2R


Violin plot di srv_diff_host_rate per attack_class


Violin plot di dst_host_count per attack_class


Violin plot di dst_host_srv_count per attack_class



Guardando più nel dettaglio alcune feature:

1. **Duration**: La maggior parte degli attacchi ha una durata molto breve, ad eccezione degli attacchi R2L. Questo indica che gli attacchi R2L potrebbero richiedere più tempo per essere effettuati. La durata potrebbe essere una feature importante da considerare, in particolare per rilevare attacchi R2L.
2. **Src_bytes e Dst_bytes**: La maggior parte degli attacchi, eccetto U2R, ha un numero relativamente basso di bytes inviati e ricevuti. Gli attacchi U2R, invece, presentano un picco di bytes inviati, suggerendo che potrebbero esserci tentativi di sovraccarico o altri tipi di attacchi che richiedono un elevato volume di traffico.
3. **Num_failed_logins**: Gli attacchi R2L mostrano un aumento in questa feature, il che ha senso dato che gli attacchi R2L spesso implicano tentativi di accesso non autorizzati.
4. **Num_compromised e Root_shell**: Anche in questo caso, la maggior parte delle classi ha valori molto bassi, ma U2R e DoS presentano picchi significativi, suggerendo che questi attacchi potrebbero compromettere le macchine o tentare di ottenere l'accesso come root.
5. **Su_attempted, Num_root, Num_file_creations, Num_shells**: Queste feature mostrano variazioni significative tra le diverse classi di attacchi, indicando che potrebbero essere utili nella classificazione.

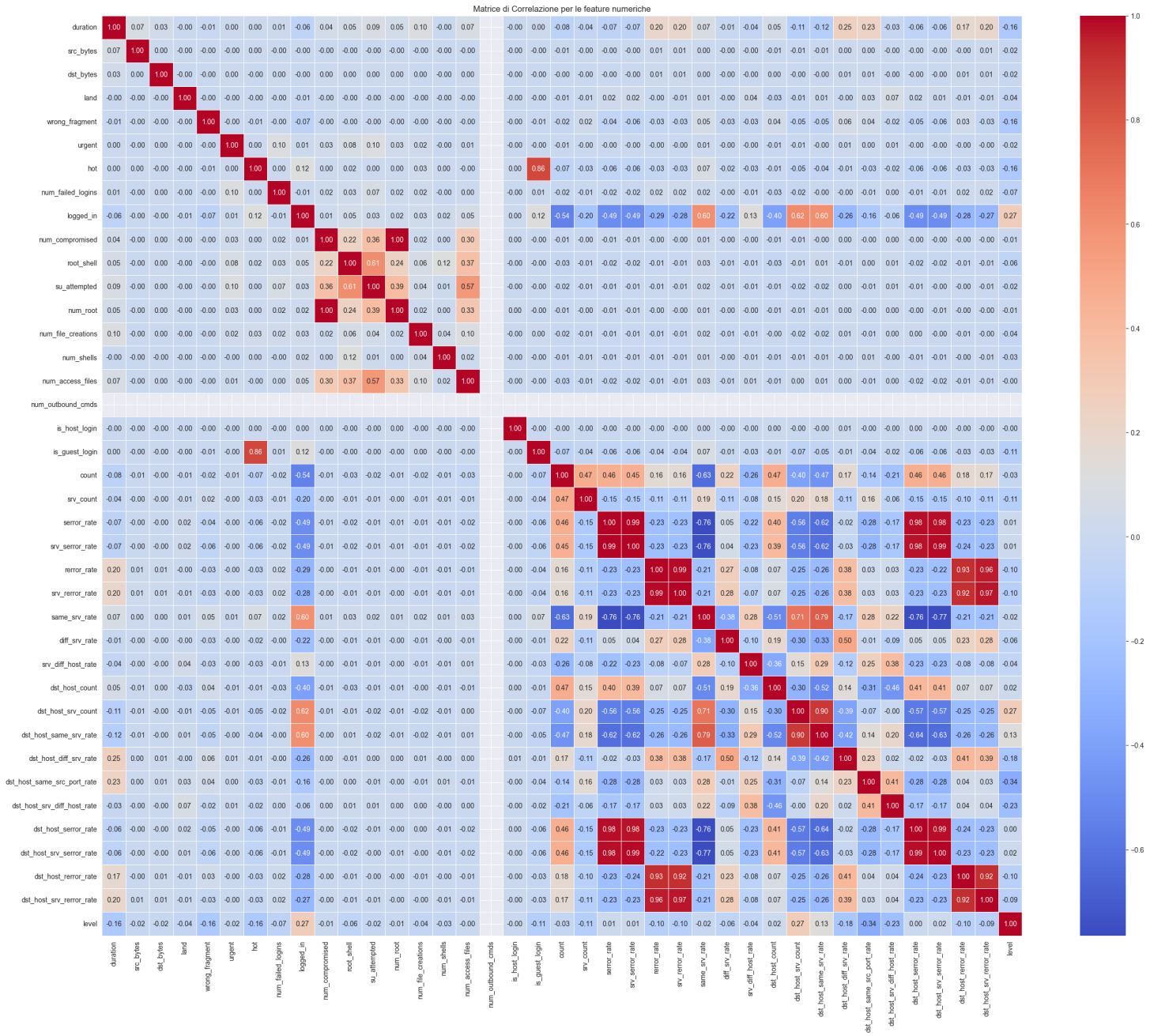
6. **Num_access_files**: R2L e U2R mostrano un aumento, il che è atteso poiché questi attacchi spesso coinvolgono l'accesso ai file.

Sicuramente alcune delle feature che riguardano i tassi di connessione (`dst_host_same_srv_rate` ...) sono altamente variegate tra le classi, quindi sottolineano una certa separabilità tra i dati e per questo motivo potrebbero essere importanti per identificare particolari tipi di attacchi.

Le informazioni ottenute dalla visione dei grafici forniscono una panoramica iniziale delle distribuzioni e delle relazioni tra le diverse feature e le classi di attacco. Tuttavia, questi grafici da soli non sono sufficienti per una selezione informata e dettagliata delle feature. Ci sono molte sfaccettature e interazioni tra le feature che potrebbero non essere immediatamente evidenti solo guardando le visualizzazioni.

Correlazione tra feature continue

Si passa ora a esaminare le correlazioni tra le caratteristiche numeriche. Questo ci aiuterà a capire quali caratteristiche sono altamente correlate e quindi individuare eventuali feature ridondanti che si potrebbe considerare di rimuovere in seguito.



- Le celle in rosso intenso indicano una forte correlazione positiva tra le variabili. Ad esempio, le variabili `num_root` e `num_compromised` mostrano una correlazione molto alta, suggerendo che quando il valore di una aumenta, l'altro tende ad aumentare in modo simile. Questo può indicare che quando un sistema è compromesso, ci sono spesso privilegi di root acquisiti.
- Le celle in blu intenso rappresentano una forte correlazione negativa. Ad esempio, le variabili `serror_rate` e `same_srv_rate` mostrano una correlazione negativa molto alta, il che indica che sono variabili che si comportano in modo opposto tra loro.
- La maggior parte delle celle ha colori vicini al bianco o ha valori molto bassi, indicando una correlazione debole o inesistente tra le variabili. Ciò suggerisce che molte delle feature nel dataset sono indipendenti l'una dall'altra, il che è positivo dal punto di vista della ridondanza delle informazioni.

- In contesti come la regressione, la presenza di variabili altamente correlate (multicollinearità) può essere problematica. Poichè intendiamo costruire dei modelli predittivi dovremo prendere in considerazione la rimozione o la combinazione di variabili altamente correlate.

Guardare semplicemente i valori di correlazione non è sufficiente, ciò che è davvero interessante è comprendere i dati e scoprire la naturea delle relazioni tra le variabili. Ad esempio, perché `srv_count` e `serror_rate` sono correlati? C'è una ragione logica dietro a ciò, o è semplicemente una coincidenza nei dati?

Dunque andiamo ad esaminare più dettagliatamente gli aspetti chiave:

- `src_bytes` e `dst_bytes` : Queste rappresentano rispettivamente la quantità di dati inviati dalla sorgente al destinatario e viceversa. Sebbene non sembrino avere una forte correlazione tra di loro, è essenziale monitorarle poiché anomalie o picchi improvvisi in questi valori possono indicare attività sospette, come tentativi di DDoS o esfiltrazione di dati.
- `num_failed_logins` : Un numero elevato di tentativi di accesso falliti in un breve periodo può indicare tentativi di attacco di forza bruta. La sua correlazione con altre variabili può aiutare a identificare se ci sono schemi specifici associati a questi tentativi. Ad esempio, se viene rilevato un aumento significativo in `num_failed_logins` e contemporaneamente si osserva un aumento in `num_compromised` , la combinazione di queste due condizioni è altamente indicativa di un'attività malevola.
- `srv_count` , `srv_serror_rate` , e `serror_rate` : Queste variabili sono correlate tra di loro e riguardano il numero di connessioni al servizio e la frequenza di errori. Un alto `serror_rate` associato a un alto srv_count può indicare che un servizio specifico è sotto attacco o ha problemi di configurazione.
- `hot` rappresenta il numero di "hot indicators", che sono attività sospette. Questa variabile, se correlata con altre, come `num_compromised` o `num_root` , può offrire intuizioni sulla natura e sull'efficacia degli attacchi.
- `dst_host_srv_count` e `dst_host_same_srv_rate` : Queste variabili forniscono informazioni sulle connessioni esterne a un host. Una forte correlazione tra queste due variabili può indicare che quando ci sono molte connessioni esterne a un host, tendono a essere al medesimo servizio.

Preprocessing

In questa sezione ci occupiamo di preparare il dataset per l'addestramento dei modelli.

Utilizzeremo gli algoritmi definiti all'interno della libreria python scikit-learn.

Codifica feature Categoriche

Per prima cosa si trasformano le feature categoriche in numeriche, questo è un passo essenziale del pre-processing poiché i modelli di ML lavorano con dati numerici. Considerando i dati a nostra disposizione, dove le categorie non hanno necessariamente un ordine, la scelta è caduta sul one-hot encoding.

- One-Hot Encoding: Crea una nuova colonna per ogni valore unico nella colonna categorica. Ad esempio, presa la colonna "protocol_type" con valori "TCP", "UDP", e "ICMP", vengono create tre nuove colonne, una per "TCP", una per "UDP", e una per "ICMP". Quindi nella colonna comparirà un 1 o uno 0 aseconda che la particolare transazione utilizzi o meno il protocollo TCP.

La controindicaizone del One-Hot Encoding è che aumenta notevolmente la dimensionalità dei dati, questo a seconda del modello che si intende utilizzare potrebbe essere un problema. Gli alberi di decisione o altri modelli basati so boosting tendenzailmente ben gestiscono l'elevata dimensionalità, mentre per modelli come le Support Vector Machines (SVM) o la regressione logistica, la dimensionalità può essere un problema, quindi andranno trovate delle soluzioni.

Altro problema derivante dall'utilizzo di questa codifica è il fatto che, come abbiamo visto nella sezione precedente, ci sono dei valori delle feature categoriche che sono presenti nel test set e non sono presenti nel train set, e viceversa. Sebbene in precedenza sia stato evidenziato solo per `attack`, questa problematica si presenta anche nelle altre feature. Quindi utilizzando il One-hot avremmo un numero di colonne diverso tra insieme di addestramento e di test e questo potrebbe creare dei problemi con i modelli.

Si è comunque mantenuta la scelta del One-hot encoding, preferendo la maggiore dimensionalità ad altri metodi di encoding come il Label Encodig che invece fornisce una sorta di relazione d'ordine intrinseca che tuttavia nel nostro caso specifico non serve. Dunque le colonne nel test set non compatibili con quelle del training set saranno ignorate.

Per la feature target, invece, si è optato per la Label Encoding, poiché non è necessario preoccuparsi di introduzioni di bias come una relazione d'ordine:

- `Normal = 1, DoS = 0, Probe = 2, R2L = 3, U2R = 4`.

Al termine delle codifiche abbiamo ottenuto un totale 118 colonne, decisamente un numero elevato.

> Per garantire la capacità del modello di generalizzare, i passaggi di preelaborazione come la codifica, la normalizzazione e lo scaling devono essere applicati distintamente ai dataset di addestramento e di test. Quando la preelaborazione è condotta prima della divisione, può verificarsi un fenomeno noto come 'data leakage' (perdita di dati). Ciò si riferisce all'infusione

> involontaria di informazioni dal dataset di test nel processo di addestramento, risultando in stime di performance eccessivamente ottimistiche e in definitiva fuorvianti. Mantenendo separati i passaggi di preelaborazione, salvaguardiamo il modello contro tali pregiudizi e manteniamo l'integrità del nostro processo di validazione.
>
> Da un punto di vista statistico, i parametri di normalizzazione (come la media e la deviazione standard) dovrebbero essere derivati esclusivamente dai dati di addestramento per mantenere i confini statistici e la rilevanza. Allo stesso modo, i metodi di codifica delle categorie devono essere adattati solo sui dati di addestramento per gestire correttamente eventuali discrepanze nei livelli categorici tra i dataset.
> Come si può vedere da questo esempio di codice, addestriamo il modello che fa la codifica OneHot sui dati di training e poi utilizziamo lo stesso modello addestrato per trasformare anche i dati di test:

```
# Addestra l'encoder sui dati di addestramento
encoder.fit(df_train[categorical_features])

# Trasforma le feature categoriche negli insiemi di train e di test
train_encoded = encoder.transform(df_train[categorical_features])
test_encoded = encoder.transform(df_test[categorical_features])
```

Normalizzazione valori continui

@TODO

Come tecnica di normalizzazione per i valori continui si è scelto lo Standard Scaler. Questo perchè le tecniche di normalizzazione che non si basano sui valori minimi e massimi del dataset, sono meno sensibili agli outlier.

Standard Scaler

Lo Standard Scaler standardizza le caratteristiche di un dataset scalando ciascuna caratteristica in modo che abbia media $\mu = 0$ e varianza $\sigma^2 = 1$.

Questo lo si ottiene sottraendo la media di ciascuna caratteristica e dividendo per la deviazione standard.

La standardizzazione è un passo cruciale in molte pipeline di machine learning perché molti algoritmi, come quelli basati su misure di distanza o gradiente, performano meglio quando i dati sono su una

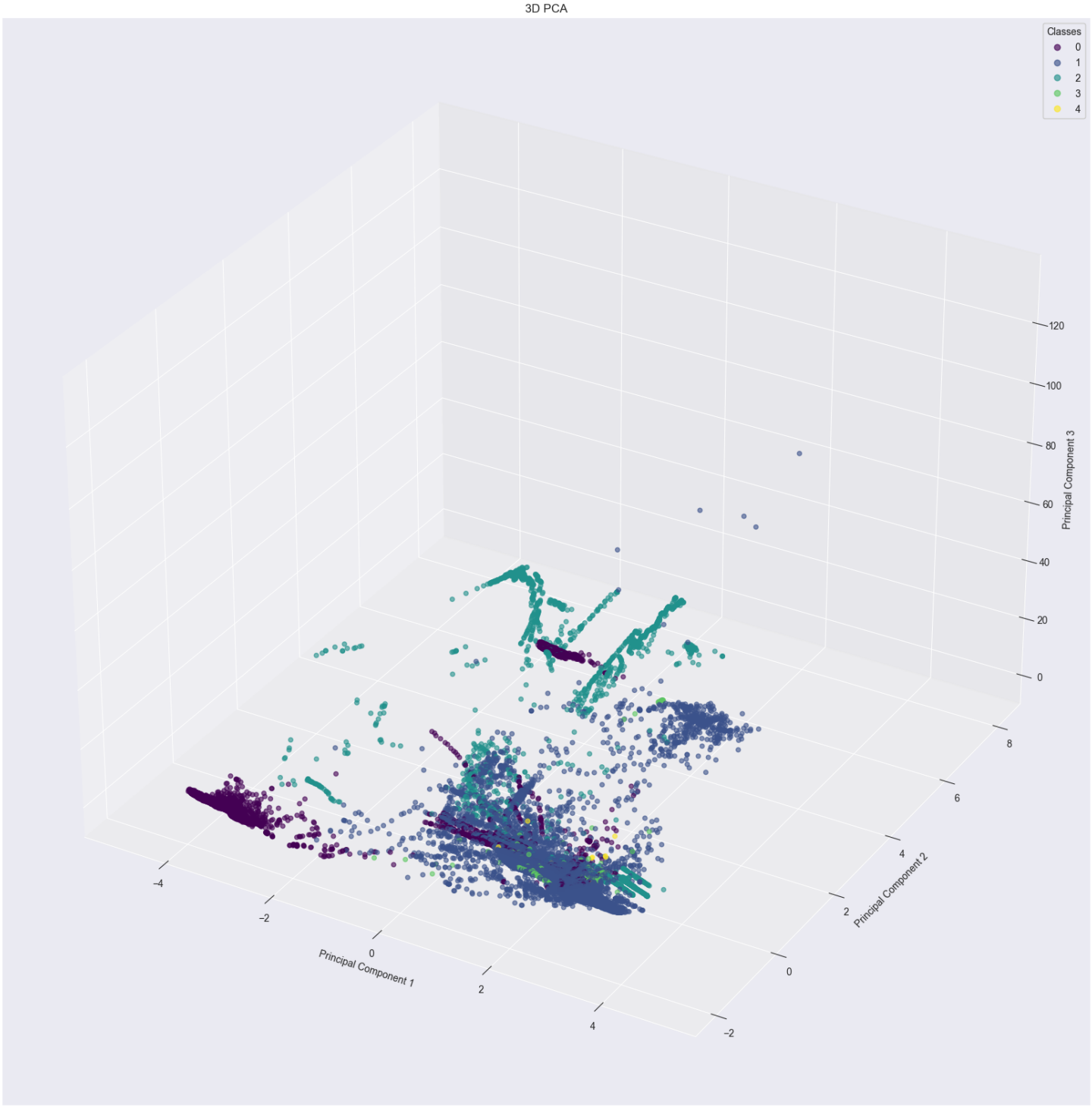
scala comune. Utilizzare lo Standard Scaler aiuta a garantire che le caratteristiche contribuiscano in modo equo al modello, evitando che caratteristiche con varianze più grandi forniscano bias ingiustificati.

EDA parte 2

Dopo aver ampliamente preprocessato i dati possiamo possiamo effettuare delle utleriori analisi utilizzando la visualizzazione di grafici

PCA

L'Analisi delle Componenti Principali (PCA) può essere utilizzata per ridurre la dimensionalità del set di dati a tre dimensioni, che possono poi essere visualizzate. Ciò può fornire una comprensione della struttura sottostante dei dati e di quanto bene le diverse categorie di attacchi siano separate in uno spazio a dimensione ridotta.



- Distribuzione degli esempi: Tutti gli esempi sembrano ben distribuiti suelle due assi orizzontali (PC1 e PC2), mentre non lo sono particolarmente sull'asse PC3. Qesto indica che gli esempi hanno una certa predisiìposizione verso un sottoinsieme delle feature. Inoltre sembra che il dataset naturalmente si divida in diversi sottogruppi
- Densità: abbiamo diverse zone ad alta intensità di esempi, questo è indice di probabili pattern comuni. Tuttavia le classi target si sovrappongono in questi raggruppamenti, dunque sarà difficile per gli algoritmi di classificazione distinguere le diverse classi
- Formazioni di Cluster: sono distinguibili diversi cluster che ben raggruppano le classi con lo stesso colore. Tuttavia abbiamo raggruppamenti diversi per le medesime classi che però sono

molto corposi, questo potrebbe evidenziare che esistono delle sottocategorizzazioni forti per una stessa classe di attacco. In seguito troveremo formalmente questi cluster attraverso l'apprendimento non supervisionato
- Outlier: Abbiamo diversi punti sparsi nel grafico, questi sono Outlier.
- Distanza tra le classi: Sulla base di quanto vediamo possiamo aspettarci un migliore separabilità nella classificazione tra Normal e DoS, una separabilità leggermente minore per Probe e probabilmente sarà quasi impossibile distinguere le ultime due classi

La scarsa separabilità sicuramente dovuta nella maggior parte allo sbilanciamento del dataset, infatti gli esempi per la classe 4 (U2R) sono talmente pochi da non risultare neanche distinguibili nel grafico.

Outliers

Basandoci sui grafici a violino visti in precedenza, era già possibile individuare punti che hanno comportamenti assimilabili agli outlier. A livello visivo questi spesso sono osservati come punti isolati dalla maggior parte dei dati, tipicamente visibili nei grafici come punti singoli o linee che si estendono oltre la sezione più ampia del 'violino'.

Nel nostro caso, diversi grafici mostrano una linea verticale molto lunga che si estende dal corpo principale del violino, il che suggerisce una serie di punti più alti o più bassi della maggioranza. Questo è particolarmente evidente in caratteristiche come `src_bytes`, `dst_bytes`, `duration` e altri.

A seguito della PCA abbiamo avuto una dimostrazione ancora più evidente della loro presenza.

È importante notare che nel nostro contesto, esempi che appaiono come outlier potrebbero essere invece elementi fondamentali dei dati. Un valore del tutto anomalo potrebbe essere cruciale per il riconoscimento di un certo attacco. Una connessione con una durata particolarmente lunga, o un trasferimento di dati molto elevato potrebbero essere indice di un comportamento anomalo e potenzialmente pericoloso.

Pertanto, alla luce di quanto detto si è scelto di mantenere questi valori poichè non vi sono informazioni sufficienti per poter procedere alla rimozione di quest'ultimi.

Bilanciamento

Come abbiamo visto nelle sezioni precedenti, la distribuzione delle classi di attacco è chiaramente sbilanciata. I dati della categoria Normal rappresentano il 53,46% degli esempi, mentre la categoria U2R conta solo lo 0,04%.

Sicuramente in un contesto reale le connessioni Normal sono molto maggiori di quelle anomale, tuttavia addestrando il nostro sistema con un numero così elevato di connessioni Normal, senza un corrispettivo numero elevato rappresentante delle altre connessioni, potrebbe andare in Overfitting verso la prima classe. Dunque immaginando il nostro modello in un ambiente di rete reale, riconoscerebbe precisamente le connessioni normali, ignorando le connessioni anomale che potrebbero essere identificative di un attacco, quindi avrebbe scarse prestazioni.

Attualmente, per affrontare il problema dello sbilanciamento dei dati, si utilizzano metodi di campionamento come il sovracampionamento (oversampling) e il sottocampionamento (undersampling). I metodi di sovracampionamento ripetono o sintetizzano nuovi campioni per le classi insufficienti, mentre il sottocampionamento ridurrà i campioni delle classi più grandi.

Per iniziare si potrebbero usare metodi Ensamble come il Bagging o il Boosting:

- Utilizzare tecniche di Bagging come le Random Forest, può aiutare in quanto vengono creati diversi sottoinsiemi del dataset originale e quindi viene addestato un modello su ciascuno per poi combinare gli output
- Metodi invece come l'AdaBoost si focalizzano di più sugli esempi mal classificati dal modello precedente, aggiungendo un peso maggiore alle classi minoritarie

Dunque utilizzando metodi di Ensamble e valutando le performance con le metriche come F1-score, Precision, Recall, and ROC AUC, vediamo se riusciamo ad ottenere risultati soddisfacenti.

	precision	**recall**	**f1-score**	**support**
0	0.96	0.82	0.89	7458
1	0.66	0.97	0.79	9711
2	0.83	0.60	0.69	2421
3	1.00	0.05	0.09	2754
4	0.00	0.00	0.00	200
accuracy			0.76	22544
macro avg	0.69	0.49	0.49	22544
weighted avg	0.82	0.76	0.72	22544

Purtroppo i risultati non sono soddisfacenti:

- Come aspettato, la classificazione funziona bene per Normal e DoS, peggiora leggermente per Probe, ma crolla del tutto sulle ultime due.
    - R2L: il fatto di avere una precisione massima, ed un recall bassissimo, indica il fatto che il classificatore è sicurissimo nella classifcazione di questa classe, ma attribuisce quasi sempre la classe sbagliata
    - U2R: il classificatore non è stato in grado in nessun caso di classificare correttamente gli esempi
- Per quanto l'accuratezza a 76% possa sembrare positiva, l'f1 mediato a 49% indica, come evidente, una significativa disparità nelle performance a seconda delle classi

> AdaBoost ha ottenuto risultati anche peggiori

Questo è sicuramente dovuto allo sbilanciamento delle classi, occorre quindi trovare un modo per bilanciarle

SMOOTEENN

Si è scelto di utilizzare un approccio ibrido, si utilizzerà prima il metodo di sovracampionamento SMOTE per amplificare i campioni con categorie di campioni minori, a seguire si sfrutterà il metodo di sottocampionamento ENN per rimuovere il rumore, formando così il metodo SMOTE-ENN (Synthetic Minority Over-sampling Technique + Edited Nearest Neighbors)

> Il processo dietro a SMOTEENN può essere così esposto:
> SMOTE: aumenta la dimensione del campione della classe minoritaria nel dataset, producendo campioni sintetici. Questo viene fatto trovando i "vicini" più prossimi e creando nuovi esempi che sono una combinazione dei tratti di questi vicini.
> ENN: Dopo aver applicato SMOTE, alcuni dei nuovi esempi creati si sovrappongano con quelli della classe maggioritaria o siano rumore. ENN aiuta a rimuovere tali esempi, eliminando ogni esempio della classe maggioritaria che ha una diversità di classe tra i suoi k vicini più prossimi. In altre parole, se un esempio della classe maggioritaria è circondato da esempi di una classe diversa, viene rimosso.

Purtroppo anche in questo caso, sebbene i risultati siano migliorati leggermente, sulle due classi minoritarie, continuano a sussistere problemi di classificazione.

A questo punto si è comunque scelto di proseguire con il dataset bilanciato e valutare altri modelli di classificazione.

Feature Reduction

Per cercare di ridurre la dimensionalità, quindi il numero di feature, sono stati presi diversi approcci:

Feature Reduction: Random Forest

Il Random Forest è un metodo di addestramento di Ensamble che è meno propenso all'overfitting. Tale modello è in grado di fornire una stima affidabile dell'importanza delle feature grazie al suo meccanismo che media tra un insieme di alberi di decisione.

Per prima cosa si è addestrato il modello sul nostro dataset preprocessato e bilanciato. Da questo modello si è estratta una gerarchia di feature in ordine di importanza, a seguito di diversi test si è visto che le prestazioni rimanevano sostanzialmente equivalenti, con la solità difficoltà nella classificazione delle ultime due classi.

Dunque si è provata un'altra tecnica.

Feature Reduction: Variance Treshold

La riduzione delle caratteristiche utilizzando una soglia di varianza, prevede l'eliminazione delle feature la cui varianza non soddisfa una certa soglia. L'idea è che le caratteristiche con bassa varianza sono meno probabili di essere informative o discriminanti nella previsione della variabile target. Abbiamo ottenuto un numero di 53 feature, tuttavia anche in questo caso non si sono riscontrati miglioramenti sensibili.

In conclusione si è scelto di mantenere la totalità delle feature poiché non si è ottenuto nessun miglioramento sia in termini di efficienza dei modelli sia in termini di velocità di computazione.

Inoltre a questo punto, ridurre la dimensionalità potrebbe rimuovere feature importanti alla lcassificazione delle classi meno rappresentate, portando a peggiorare ulteriormente la situazione.

Apprendimento Supervisionato

Gli obiettivi che cercheremo di raggiungere in questa fase sono di individuare dei modelli che in modo efficiente classifichino le tipologie di attacchi è che quantomeno riescano a distinguere una connessione normale da una pericolosa. Tra i vincoli che ci poniamo ci sono la volontà di ottenere un

modello facilmente intrepretabile che impieghi un tempo consono per la classificazione, visto che in un contesto di distribuzione questi modelli dovrebbero essere relativamente veloci.

Per ciascuno dei modelli addestrati di seguito si è utilizzato:

- k-fold cross validation con k = 5
- grid-search per la ricerca degli iperparametri migliori utilizzando l'f1 come metodo di confronto

> Vista la grossa dimensione del dataset, per motivi computazionali l'addestramento è stato effettuato su un dataset contenente il 20% degli esempi del dataset completo. Questo dataset è stato fornito dagli stessi sviluppatori e presenta, in proporzione, la stessa distribuzione di esempi per classe di attacco.
>
> Anche la scelta degli iperparametri è stata effettuata considerando il costo computazionale, per questo motivo si sono scelti quelli ritenuti più importanti

Decision Tree

Gli iperparametri considerati sono:

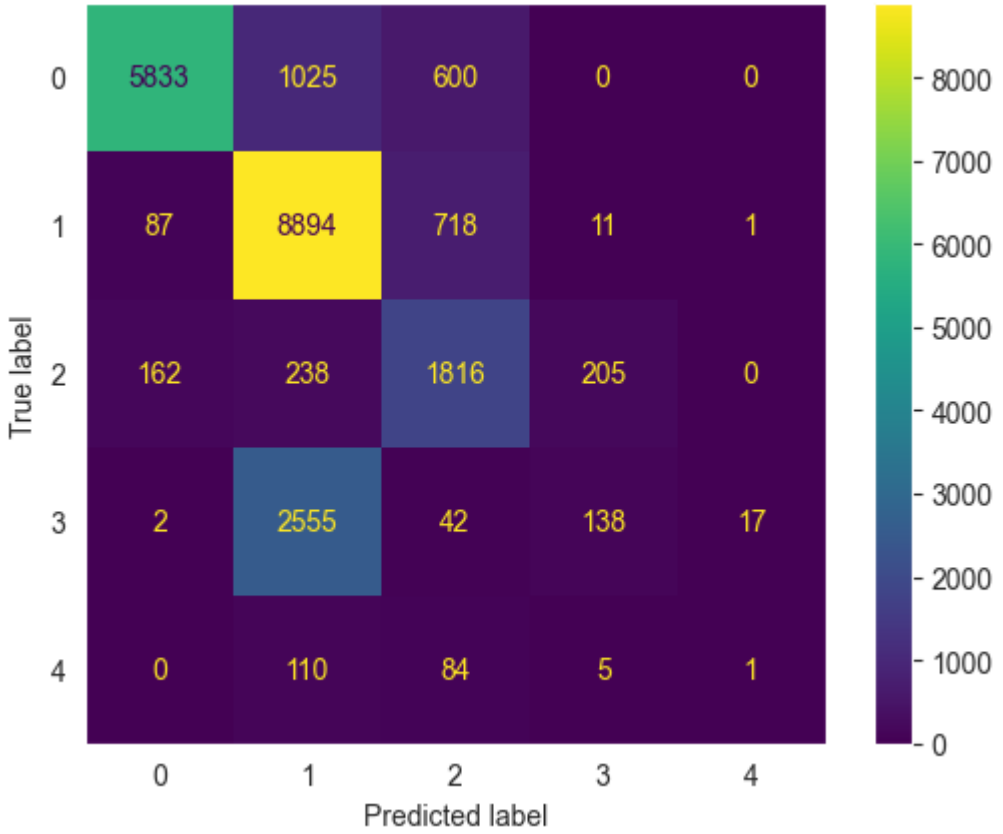
- criterion: misura la qualità della divisione all'interno dell'albero
  - gini: valutà l'impurità di un nodo rispetto alle classi. Ad ogni suddivisione dei nodi si cerca di minimizzare questa variabile al fine di ottenere partizioni omogenee e prestazioni migliori
  - entropy: misura l'entropia dei nodi. Maggiore è l'etrnopia migliore è la suddivisione
- max_depth: è la profondità massima dell'albero, può essere:
  - None: profondità illimitata. I nodi vengono espansi fino a quando tutte le foglie sono pure o contengono meno campioni di quanti indicati da min_samples_split
  - Un qualunque numero, qui si sono scelti 5 10 e 50
- min_samples_split: Il numero minimo di campioni per poter splittare un nodo (2, 10)
- min_samples_leaf: Il numero minimo di campioni possibili su un nodo foglia (1, 4)

Gli iperparametri scelti:

```
{'criterion': 'gini', 'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2}
```

Class	Precision	Recall	F1 Score	Support
0	0.96	0.78	0.86	7458.0
1	0.69	0.92	0.79	9711.0
2	0.56	0.75	0.64	2421.0

Class	Precision	Recall	F1 Score	Support
3	0.38	0.05	0.09	2754.0
4	0.05	0.01	0.01	200.0
macro avg	0.53	0.50	0.48	22544.0
weighted avg	0.72	0.74	0.70	22544.0
Overall			**Accuracy**	0.74



KNN

Gli iperparametri considerati sono:

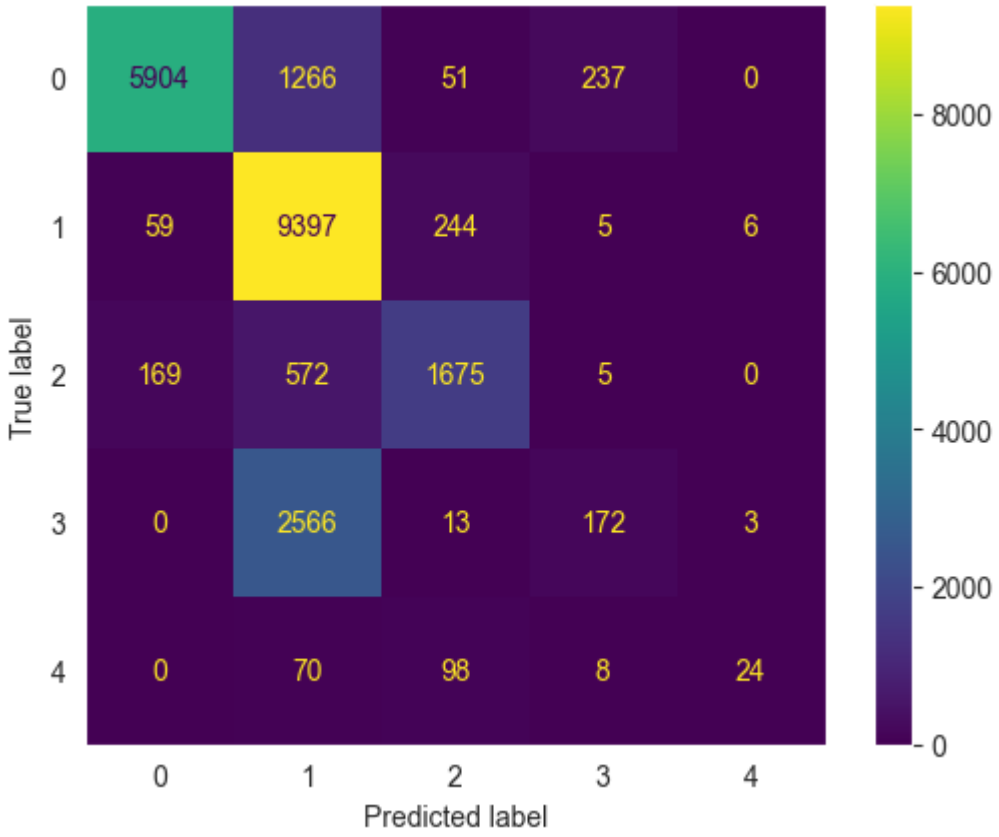
- `n_neighbors` : il numero di vicini considerati durante la predizione [3, 10, 15]
- `weights` : i pesi assegnati ai vicini, possono essere:
  - `uniform` : ogni vicino ha lo stesso peso
  - `distance` : i vicini più prossimi hanno peso maggiore
- `p` : Il tipo di calcolo utilizzato per calcolare la distanza
  - 1: distanza di Manhattan

- 2: distanza Euclidea

Gli iperparametri scelti:

```
{'n_neighbors': 3, 'p': 1, 'weights': 'distance'}
```

Class	Precision	Recall	F1 Score	Support
0	0.96	0.79	0.87	7458.0
1	0.68	0.97	0.80	9711.0
2	0.80	0.69	0.74	2421.0
3	0.40	0.06	0.11	2754.0
4	0.73	0.12	0.21	200.0
macro avg	0.72	0.53	0.54	22544.0
weighted avg	0.75	0.76	0.73	22544.0
Overall			**Accuracy**	0.76



Naive Bayes

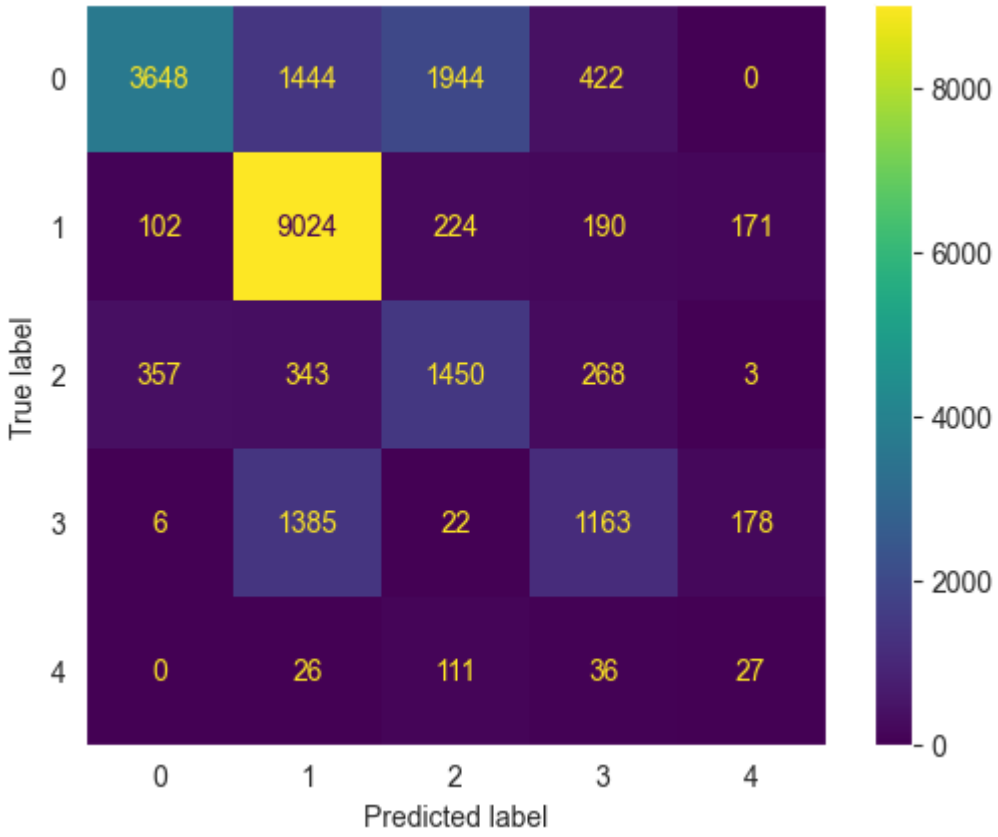
Il modello Naive Bayes non è motlo sensibile agli iperparametri tuttavia si è scomunque scelto di utilizzarli:

- `var_smoothing` : [1e-9, 1e-8, 1e-7, 1e-6, 1e-5]

Gli iperparametri scelti:

```
{'var_smoothing': 1e-07}
```

Class	Precision	Recall	F1 Score	Support
0	0.89	0.49	0.63	7458.0
1	0.74	0.93	0.82	9711.0
2	0.39	0.60	0.47	2421.0
3	0.56	0.42	0.48	2754.0
4	0.07	0.14	0.09	200.0
macro avg	0.53	0.51	0.50	22544.0
weighted avg	0.72	0.68	0.67	22544.0
Overall			**Accuracy**	0.68



Logistic Regression

Gli iperparametri considerati sono:

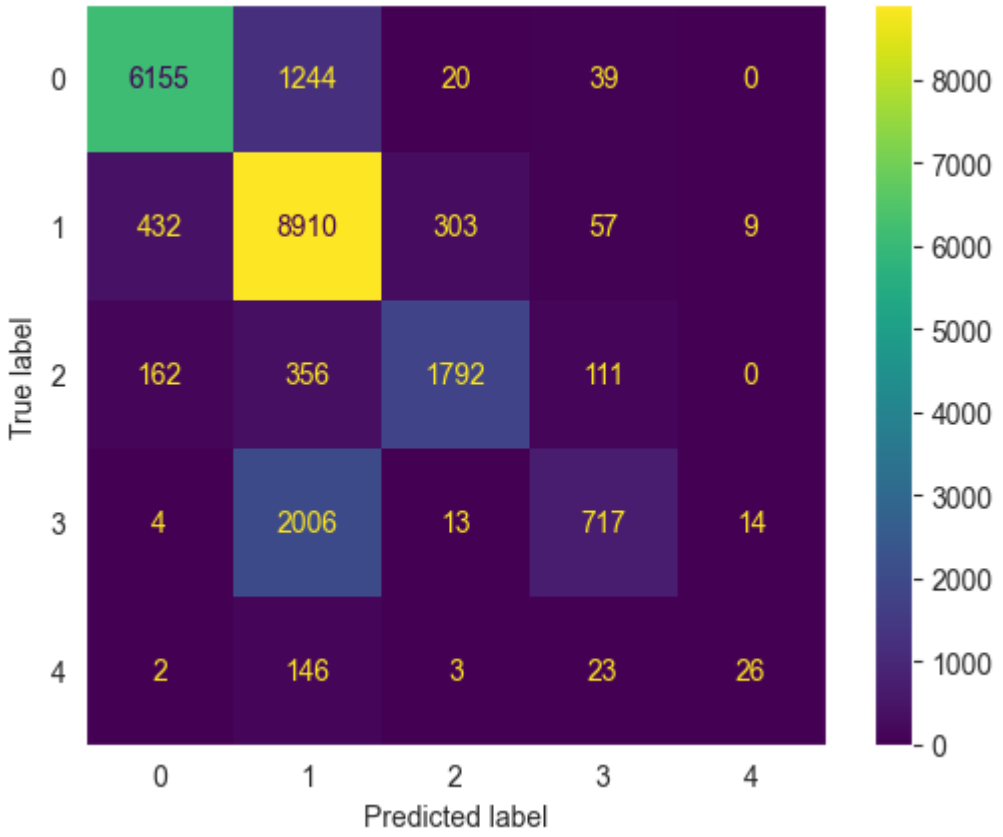
- `C` : parametro di regolarizzazione, più sono bassi i valori più la regolarizzazione aumenta[0.1, 1.0, 10]
- `solver` : Algoritmo utilizzato nel problema di ottimizzazione ['newton-cg', 'lbfgs', 'liblinear']
- `max_iter` : Numero massimo di iterazioni prima di convergere [100, 200]

Gli iperparametri scelti:

```
{'C': 10, 'max_iter': 100, 'solver': 'newton-cg'}
```

Class	Precision	Recall	F1 Score	Support
0	0.91	0.83	0.87	7458.0
1	0.70	0.92	0.80	9711.0
2	0.84	0.74	0.79	2421.0
3	0.76	0.26	0.39	2754.0
4	0.53	0.13	0.21	200.0

Class	Precision	Recall	F1 Score	Support
macro avg	0.75	0.57	0.61	22544.0
weighted avg	0.79	0.78	0.76	22544.0
Overall			**Accuracy**	0.78



SVM

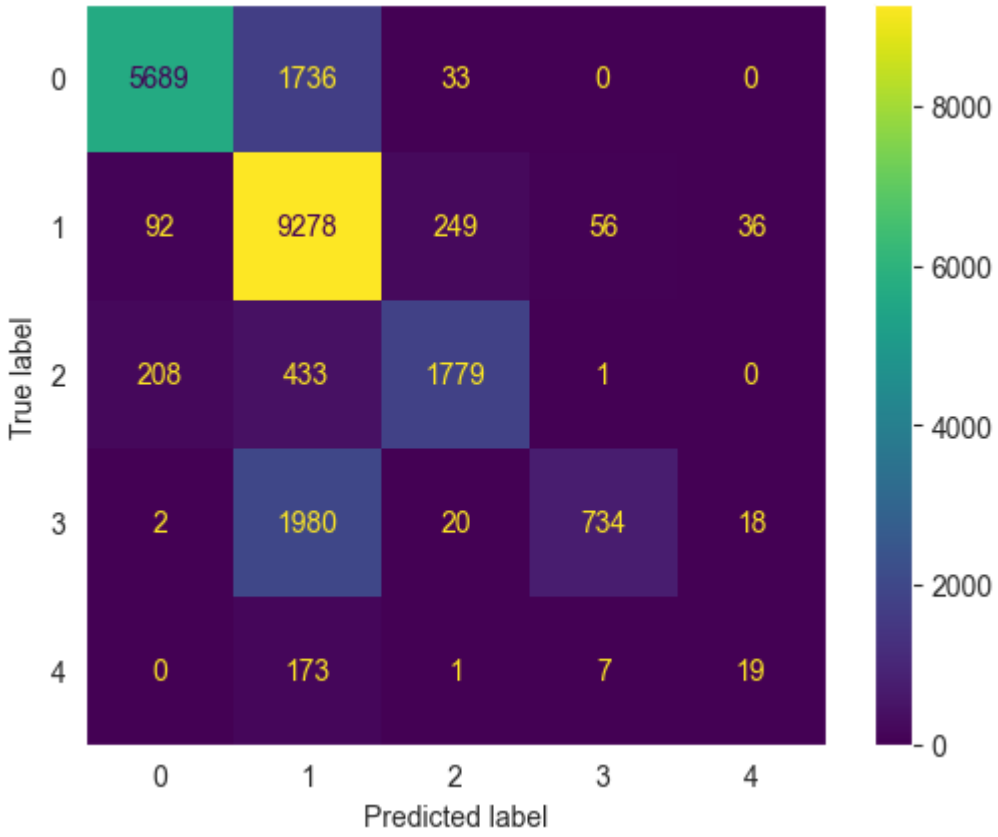
Gli iperparametri considerati sono:

- `C` : parametro di regolarizzazione, inversamente proporzionale al valore (0.1, 1.0)
- `kernel` : tipo di kernel utilizzato dall'algoritmo
  - `linear` : trasformazione lineare
  - `poly` : trasformazione polinomiale
  - `rbf` : funzione radiale
- `gamma` : coefficiente per il kernel
  - `scale` : scala fissa per gamma
  - `auto` : gamma vincolata alla dimensione dei dati di ingresso

Gli iperparametri scelti:

`C : 0.5, kernel : rbf, gamma : scale`

Class	Precision	Recall	F1 Score	Support
0	0.95	0.76	0.85	7458.0
1	0.68	0.96	0.80	9711.0
2	0.85	0.73	0.79	2421.0
3	0.92	0.27	0.41	2754.0
4	0.26	0.10	0.14	200.0
macro avg	0.73	0.56	0.60	22544.0
weighted avg	0.81	0.78	0.76	22544.0
Overall			**Accuracy**	0.78



Random Forest

Gli iperparametri considerati sono:

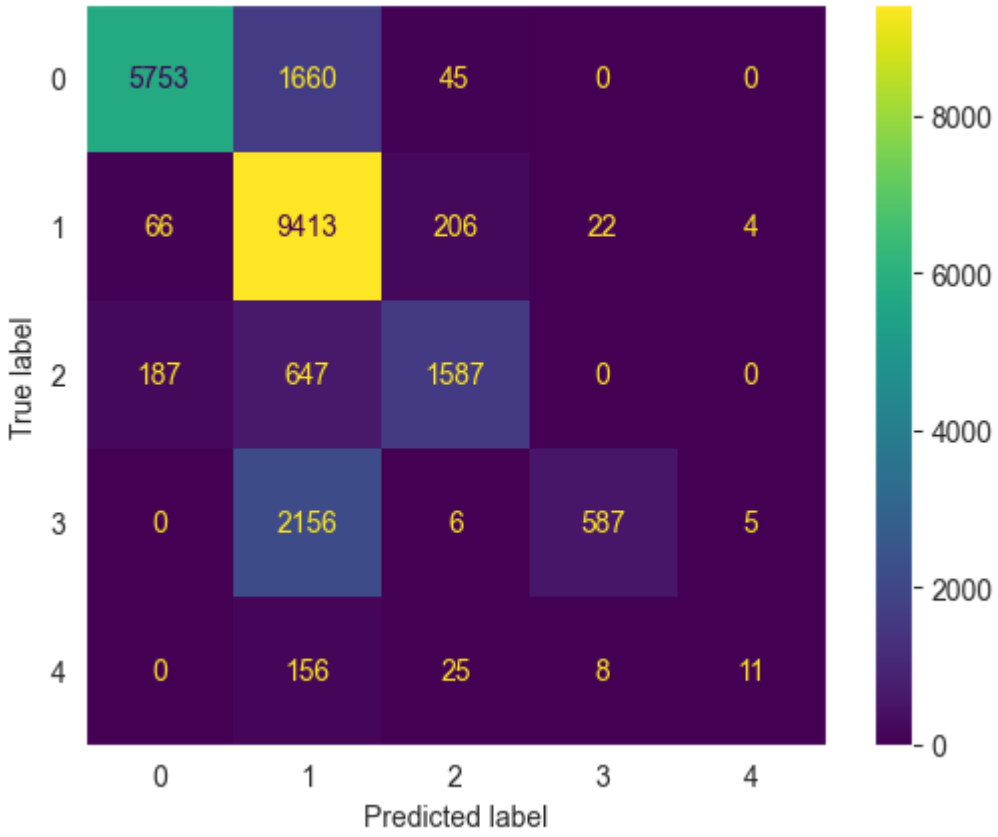
> per dettagli sugli iperparametri fare riferimento alla sezione su Decision Tree

- n_estimators: (10, 100, 200)
- criterion: (gini, entropy)
- max_depth: (10, 50, None)
- min_samples_split: (2, 10, 15)

Gli iperparametri scelti:

```
n_estimators : 200, max_depth : 10, criterion : 'gini', min_samples_split : 10
```

Class	Precision	Recall	F1 Score	Support
0	0.96	0.77	0.85	7458.0
1	0.67	0.97	0.79	9711.0
2	0.85	0.66	0.74	2421.0
3	0.95	0.21	0.35	2754.0
4	0.55	0.06	0.10	200.0
macro avg	0.80	0.53	0.57	22544.0
weighted avg	0.82	0.77	0.75	22544.0
Overall			**Accuracy**	0.77



AdaBoost

Gli iperparametri considerati sono:

`n_estimators` : numero di learner deboli che vanno addestrati in modo iterativo [50, 100, 200]

`learning_rate` : peso applicato al classificatore ad ogni iterazione di boosting [0.01, 0.1, 1]

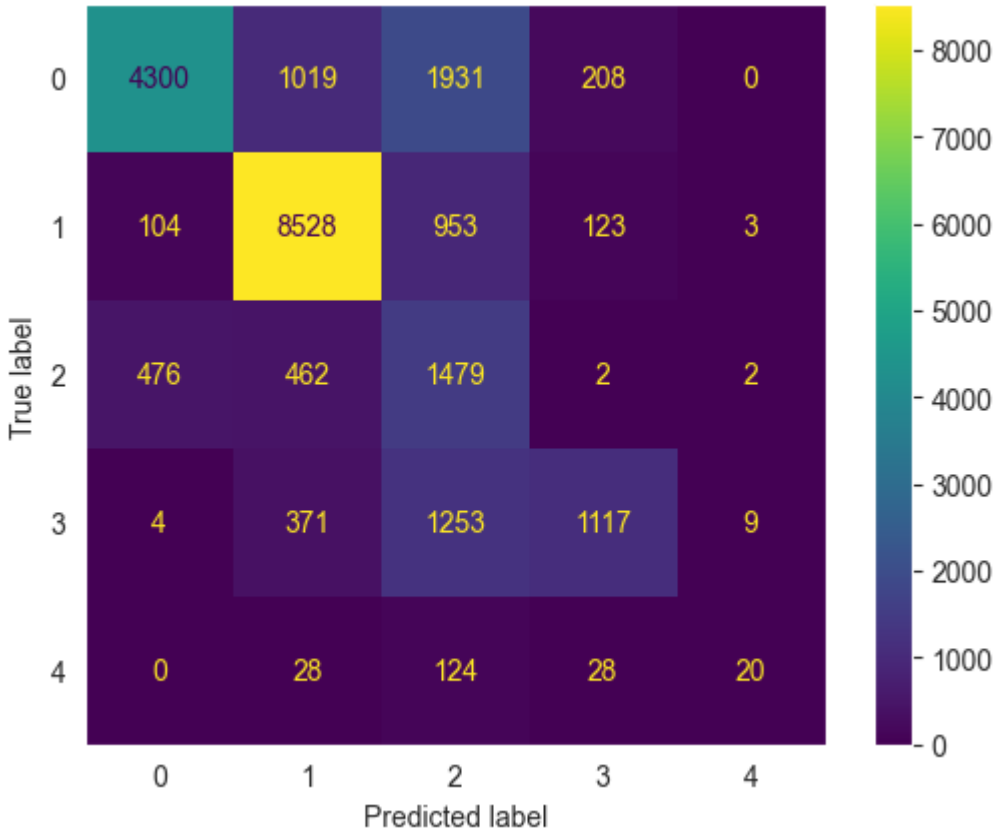
`estimator` : il learner di base utilizzato ['DecisionTreeClassifier', 'GaussianNB']

Gli iperparametri scelti:

`random_state : 42, n_estimators : 200, learning_rate : 1, estimator : GaussianNB`

Class	Precision	Recall	F1 Score	Support
0	0.88	0.58	0.70	7458.0
1	0.82	0.88	0.85	9711.0
2	0.26	0.61	0.36	2421.0
3	0.76	0.41	0.53	2754.0
4	0.59	0.10	0.17	200.0
macro avg	0.66	0.51	0.52	22544.0

Class	Precision	Recall	F1 Score	Support
weighted avg	0.77	0.69	0.70	22544.0
Overall			**Accuracy**	0.69



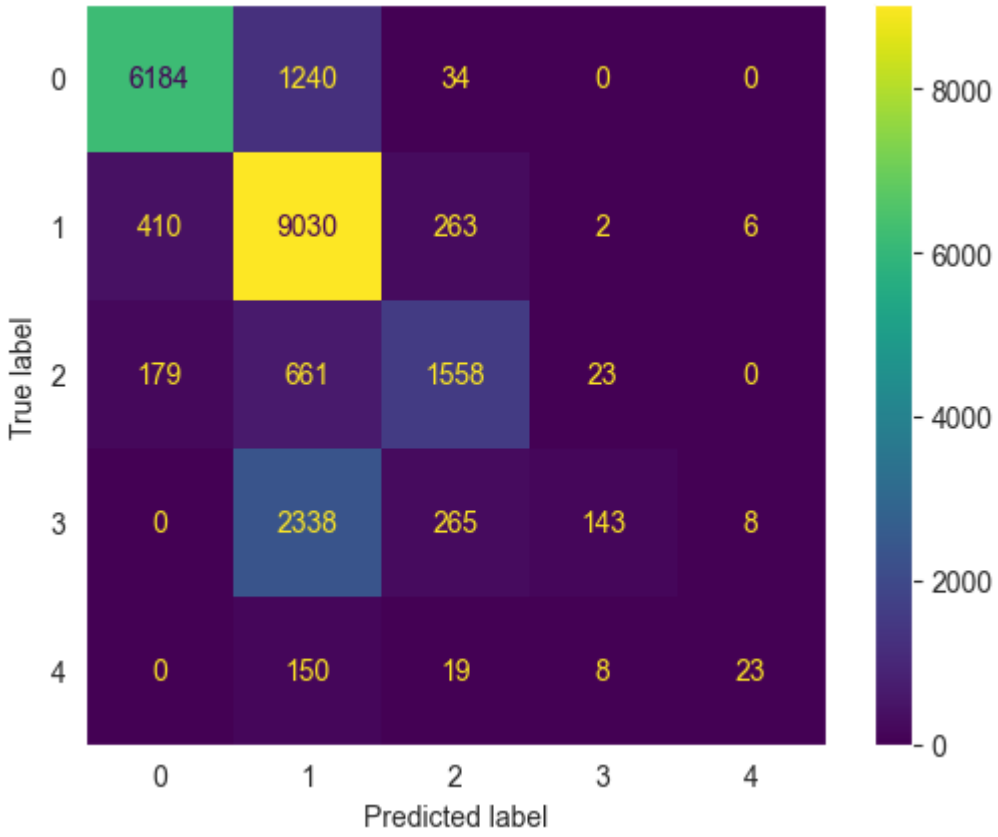
Rete Neurale

Si è sviluppata anche una rete neurale del sequente tipo:

- livelli:
  i. Livello di Input: definito implicitamente, prende in input tutte le feature del dataset
  ii. Primo livello nascosto: livello denso (completamente connesso) composto da 128 neuroni, utilizza la funzione di attivazione ReLU
  iii. Secondo livello nascosto: livello denso composto da 64 neuroni, anche questo utilizza la funzione di attivazione ReLU
  iv. Livello di Output: ha un numero di neuroni equivalente al numero di classi target. Utilizza la funzione di attivazione Softmax, una funzione standard per i task multiclasse, questa restituisce una distribuzione di probabilità tra le classi.
- Compilazione: il modello è compilato con la seguente configurazione

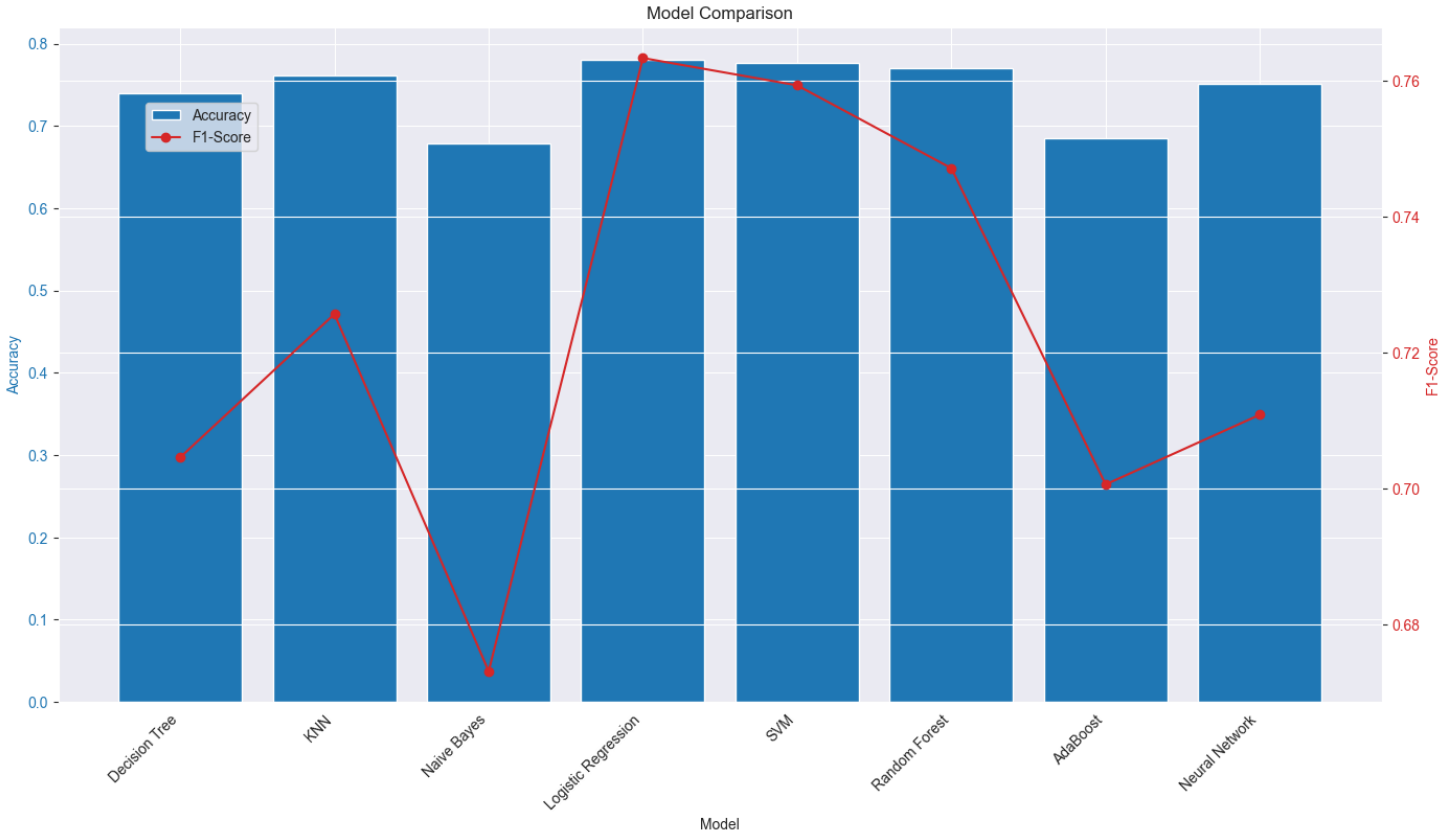
    - Ottimizzatore: Adam, algoritmo di ottimizzazione che aggiorna i pesi della rete in modo iterativo in base ai dati di addestramento
    - funzione di perdita (loss function): Corssentropy categorica, una funzione di predita usata per le classificazioni multi classe, si aspetta le classi target codificate con la One-hot
    - Metrica: Accuratezza
- Addestramento
    - epoche: 100, ovvvero l'intero dataset passa attraverso la rete 100 volte
    - batch_size: 32, ovvero ad ogni passo si utilizzano 32 campioni per addestrare la rete. Ad ogni aggiornamento l'ottimizzatore aggiorna i pesi e i bias dopo ogni lotto (batch)

Class	Precision	Recall	F1 Score	Support
0	0.91	0.83	0.87	7458.0
1	0.67	0.93	0.78	9711.0
2	0.73	0.64	0.68	2421.0
3	0.81	0.05	0.10	2754.0
4	0.62	0.12	0.19	200.0
macro avg	0.75	0.51	0.52	22544.0
weighted avg	0.77	0.75	0.71	22544.0
Overall			**Accuracy**	0.75



Valutazione

Model	Accuracy	F1-Score
Decision Tree	0.74	0.70
KNN	0.76	0.73
Naive Bayes	0.68	0.67
Logistic Regression	0.78	0.76
SVM	0.78	0.76
Random Forest	0.77	0.75
AdaBoost	0.69	0.70
Neural Network	0.75	0.71



In termini di accuratezza e f1 i modelli migliori sono Logistic Regression e SVM. Forse è preferibile però il primo perchè performa leggermente meglio sulle classi minoritarie oltre ad essere più facilmente interpretabile rispetto ad una SVM.

Tuttavia i risultati ottenuti non sono sufficienti, un IDS mira ad indentificare accuratamente e poi a classificare delle attività nella rete potenzialmente dannose. Il basso recall delle classi 3 e 4, è indice del fatto che il modello non riconosce un numero elevato di casi positivi (attacchi) e questo è un problema grave per un IDS, perché non può permettersi di non riconoscere un numero significativo di attacchi. Allo stesso modo la bassa precisione per la classe 4 ci dice che le predizioni sono corrette poco più della metà delle volte, quindi ci possono essere parecchi falsi positivi. Ancora, il basso f1 per le classi 3 e 4 indicano scarse performance sulle stesse, questo significa che il sistema non è affidabile nel distinguere un comportamento normale da uno anomalo per queste categorie. L'elevata differenza tra il macro average F1 score la media pesata di F1, il primo considera tutte le classi in maniera equa, invece il secondo considera il supporto di tutte le classi. L'elevata differenza suggerisce come il modello sia eccessivamente influenzato dalle classi più rappresentate, a tal punto da coprire le scarse performance sulle classi meno frequenti.

Sebbene si sia cercato di gestire lo sbilanciamento del dataset, questo comunque è riuscito ad influenzare negativamente il sistema. La mancata individuazione di feature che fossero discriminanti per le classi meno rappresentate ha ulteriormente ridotto le prestazioni.

Rappresentazione della conoscenza: Ontologia

Le ontologie in informatica sono sistemi strutturati di termini e concetti utilizzati per descrivere un particolare dominio di conoscenza, in modo che sia comprensibile sia agli umani che alle macchine.

Nel contesto degli IDS (Intrusion Detection Systems), le ontologie giocano un ruolo cruciale, esse possono infatti migliorare le capacità di rilevazione, grazie alla correlazione degli eventi.

- Flessibilità: possono essere aggiornate e modificate facilmente, consentendo agli IDS di adattarsi rapidamente all'evoluzione delle minacce e delle tecniche di attacco.
- Interoperabilità: Utilizzando ontologie standardizzate, diversi sistemi di rilevazione intrusioni possono condividere e comprendere le informazioni gli uni degli altri, migliorando la collaborazione e l'efficacia complessiva nella difesa da attacchi informatici.
- Riduzione dei Falsi Positivi e Negativi: Grazie alla precisa definizione dei comportamenti e delle attività di rete, le ontologie possono aiutare a ridurre il numero di falsi positivi (allarmi non necessari) e falsi negativi (mancata rilevazione di un attacco reale) degli IDS.

Visti i risultati ottenuti addestrando i diversi modelli sul dataset, si è pensato di offrire una diversa rappresentazione del dominio in modo da poter inferire nuova conoscenza soprattuto per qeulle classi di attacchi poco rappresentate nel dataset.

Dunque si è partiti dal dataset precedente per costruire una Ontologia

Classi

Come classe principale si è scelto `NetworkTraffic` , si è ritenuto come questo termini incapsuli in modo ottimale l'essenza del dataset, che fondamentalmente rappresenta un insieme di "Network activities" che possono distinguersi in traffico normale o diverse tipologie di attacchi

La scelta di modellare alcune entità come classi, deriva dal fatto che, esse esistono indipendentemente l'una dall'altra. Sebbene il Protocollo internet faccia parte del NetworkTraffic questo ha una sua valenza anche se preso singolarmente, quindi giustifica la sua rappresentazione come classe. Invece, ad esempio, Flag è stato rappresentato come proprietà poichè identifica un certo stato durante un'attività di rete, quindi non esiste al di fuori del traffico.

Di seguito si presentano le diverse classi e sottoclassi con le proprietà ad esse associate

NetworkTraffic:

Object Properties:

- hasProtocolType : individua il protocollo utilizzato dall'attività di rete
- hasServiceType: individua il serviizo utilizzato
- hasAttack: se è un'attività d'attacco ne individua la classe

Data properties:

- count
- dst_bytes
- duration
- flag
- hot
- is_guest_login
- is_host_login
- land
- logged_in
- num_access_files
- num_compromised
- num_failed_logins
- num_file_creations
- num_outbound_cmds
- num_root
- num_shells
- root_shell
- src_bytes
- srv_count
- su_attempted
- urgent
- wrong_fragment

> Fare riferimento alla tabbella che descrive le diverse feature per informazioni riguardo il significato di queste proprietà

Attack

Classe che modella le tipologie di attacchi, suddivisa in sottoclassi:

- Sottoclassi:

    - DoS
    - Probe
    - R2L
    - U2R

Ciascuna di queste sottoclassi ha a sua volta sottoclassi riguardanti l'attaco specifico.

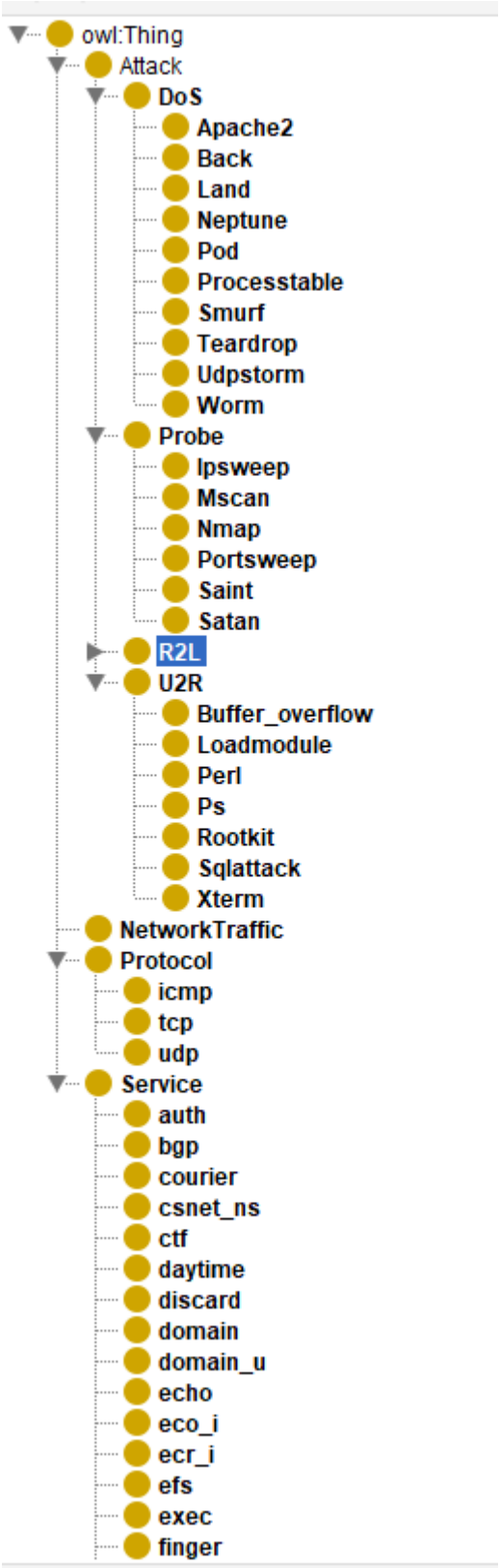
Protocol

Modella i diversi protocolli:

- icmp
- tcp
- udp

Service

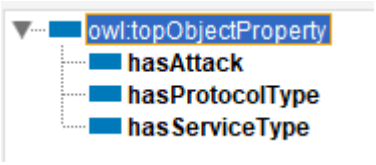
Modella i diversi servizi, tutti rappresentati come sottoclassi di Service

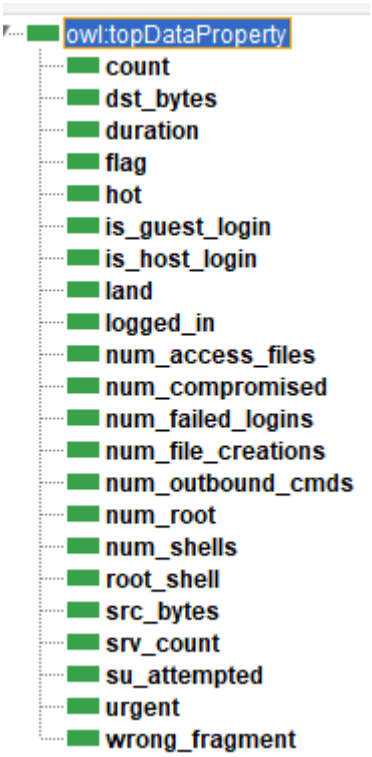


Proprietà

Definite le classi dobbiamo costruire la struttura interna delle entità modellate. Le proprietà si distinguono in:

- Object Property: modellano le relazioni che intercorrono tra più entità (classi)
  - hasProtocolType : individua il protocollo utilizzato dall'attività di rete
    - Dominio : Il dominio di una proprietà specifica la classe di entità a cui può essere applicata. In questo caso quindi è NetworkTraffic
    - Range : Il range di una proprietà definisce la classe di entità a cui essa fa riferimento. In questo caso quindi è Protocol
  - hasServiceType: individua il serviizo utilizzato
    - Dominio: NetworkTraffic
    - Range: Service
  - hasAttack: se è un'attività d'attacco ne individua la classe
    - Dominio: NetworkTraffic
    - Range: Attack
- Data property: relaziona un'entità con un valore di tipo primitivo
  - Domino: in questo caso hanno tutte come dominio NetworkTraffic
  - Range:
    - Binario {0,1}: land, logged_in, is_host_login, is_guest_login
    - Reale: count, dst_bytes, duration, flag, hot, num_access_files, num_compromised, num_failed_logins, num_file_creations, num_outbound_cmds, num_root, num_shells, root_shell, src_bytes, srv_count, su_attempted, urgent, wrong_fragment





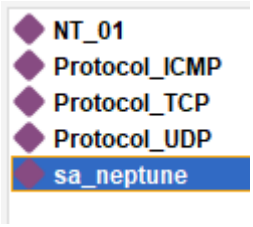
Individui

Gli individui rappresentano le istanze delle classi, dunque nel nostro caso si stratta degli esempi contenuti nel dataset.

Per come si è scelto di progettare l'ontologia le sottoclassi di Protocol e Service, vengono rappresentate come individui generici (e.g. Protcol_TCP). Questi individui sono necessari per poi poter stabilire le relazioni con gli individui di NetworkTraffic, dunque il generico individuo avrà la relazione ad esempio hasProtoclType che lo legherà all'individuo Protocol_UDP.

Lo stesso discorso vale per per il tipo di attacco, l'individuo di NetworkTraffic sarà in relazione con l'individuo che identifica un certo attacco (e.g. Neptune). Inoltre pocihé si è utilizzato il meccanismo delle classi e sottoclassi, non è stato neceesario introdurre una relazione che legasse l'attacco specifico alla classe di appartenenza (e.g. DoS), poiché questa informazione è insita nel concetto di sottoclasse.

La scelta di modellare queste entità come classi è stata presa anche in virtù della modularità, infatti, con questa progettazione in futuro si potrebbero aggiungere nuovi individui che modellano ad esempio un evento specifico di una connessione TCP, aggiungendo quindi ulteriore semantica all'individuo NetworkTraffic



Per quanto riguarda le feature del dataset non modellate nell'ontologia, esse non sono state inserite per non aumentare la complessità dell'ontologia. Infatti, rendere l'ontologia estremamente dettagliata potrebbe renderla più difficilmente gestibile. Una elevata complessità comporta minori prestazioni nella gestione delle query. Quindi si è scelto di includere solo le feature ritenute più importanti e rappresentative. Inoltre le caratterstiche non introdotte sono facilmente calcolabili combinando le feature invece presenti nell'ontologia.

Integrazione con Python

> Fare riferimento al file ontology.py

Per rendere più efficiente l'integrazione di elementi nell'ontologia, soprattutto per quanto riguarda l'inserimento degli individui, si è scelto di utilizzare la libreria `owlready2` per poter manipolare l'ontologia direttamente dal codice.

Dunque attraverso il codice è stato possibile inserire in modo automatizzato tutti gli esempi del dataset.

Inoltre è stato possibile formulare e quindi effettuare delle query per interrogare l'ontologia.

BIBLIOGRAFIA

M. Tavallaee, E. Bagheri, W. Lu and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, Ottawa, ON, Canada, 2009, pp. 1-6, doi: 10.1109/CISDA.2009.5356528.

Sarika Choudhary, Nishtha Kesswani,
Analysis of KDD-Cup'99, NSL-KDD and UNSW-NB15 Datasets using Deep Learning in IoT,
Procedia Computer Science,
Volume 167,
2020,
Pages 1561-1573,
ISSN 1877-0509,
https://doi.org/10.1016/j.procs.2020.03.367.

Gustavo E. A. P. A. Batista, Ronaldo C. Prati, and Maria Carolina Monard. 2004. A study of the behavior of several methods for balancing machine learning training data. SIGKDD Explor. Newsl. 6, 1 (June 2004), 20–29. https://doi.org/10.1145/1007730.1007735