UNIVERSITA' DEGLI STUDI DI
NAPOLI FEDERICO II

Scuola Politecnica e delle Scienze di Base
Corso di Laurea in Ingegneria Informatica

Elaborato in **Network Security**

Intrusion Detection System using Machine Learning

Anno Accademico 23/24

Studente:
Francesco Pio Manna
matr. M63001485

Indice

Introduzione

Il crescente utilizzo e la crescente dipendenza dalle infrastrutture informatiche ha portato a un aumento significativo delle minacce alla sicurezza, rendendo la protezione dei sistemi informatici e informativi una priorità assoluta. In questo contesto, gli Intrusion Detection Systems (IDS) giocano un ruolo cruciale, poiché consentono di monitorare e analizzare il traffico di rete al fine di identificare comportamenti anomali o malevoli che potrebbero compromettere la sicurezza di tali sistemi. Tuttavia, con l'aumento della complessità e della numerosità degli attacchi verso le piattaforme informatiche, i tradizionali IDS basati su regole stanno diventando sempre meno efficaci, richiedendo soluzioni più avanzate e adattive.

Negli ultimi anni, lo sviluppo del machine learning ha aperto nuove opportunità nel campo della sicurezza informatica. Grazie alla capacità di apprendere da grandi quantità di dati, identificare pattern complessi e adattarsi a nuove minacce, il machine learning è diventato uno strumento potente per migliorare l'efficacia degli IDS. L'applicazione di tecniche di machine learning agli IDS consente non solo di migliorare la capacità di rilevamento delle intrusioni, ma anche di ridurre i falsi positivi e di adattarsi dinamicamente a nuove forme di attacco.

Lo scopo di questo progetto è quello di analizzare un dataset preesistente relativo a traffico di rete, comprendere quali sono le features che possono aiutare a discriminare traffico malevolo da traffico normale ed utilizzare algoritmi di machine learning per etichettare il traffico, individuando gli attacchi.

Capitolo 1

Intrusion Detection Systems

In questo capitolo è riportata una breve descrizione del concetto di intrusione di sicurezza, di IDS e una loro tassonomia di classificazione.

1.1 Intrusioni

Le **intrusioni**, in ambito di sicurezza sono definite dal RFC come: "un evento di sicurezza, o una combinazione di multipli eventi di sicurezza, che costituiscono un incidente di sicurezza nel quale un intruder ottiene, o cerca di ottenere, accesso al sistema o alle risorse di sistema senza avere l'autorizzazione". Più semplicemente: "un tipo di minaccia dove un entità non autorizzata ottiene accesso a dati sensibili attraverso l'elusione delle protezioni del sistema di sicurezza".
Questa definizione, per quanto scontata, contiene un aspetto abbastanza particolare, ovvero il fatto che si intenda l'intrusione come tentativo, questo perché il solo tentativo può essere un attacco. Segnare ogni tentativo, però, potrebbe generare moltissimi falsi positivi, e quindi portare alla perdita di fiducia nel sistema di controllo.
Gli IDS sono importanti per i costi di gestione delle intrusioni, e quindi effettuare azioni tempestive, le quali permettono di salvare delle aziende, ma anche perché quando i sistemi di prevenzione delle intrusioni (IPS) falliscono, allora la seconda linea di difesa sono i sistemi di rilevamento delle intrusioni (IDS).
I problemi reali nel rilevare un'intrusione risiedono nel fatto che ogni giorno nascono 0days ed exploit nuovi, ma se ben configurati possono rilevare intrusioni anche contro 0days. Di solito quando si scoprono intrusioni si parla di incidente di sicurezza.

1.1.1 Tipologie di intruder

Una delle più importanti minacce alla sicurezza è costituita dagli intruder (l'altra è rappresentata dai malware), spesso indicato come hacker o cracker, con l'obiettivo di accedere ad un sistema per cui non si ha l'autorizzazione, acquisire i privilegi e effettuare azioni non autorizzate. La maggior parte degli intruder sono persone estranee al sistema

(outsiders/masquerade), ma c'è anche una buona percentuale interna all'azienda stessa (insiders/misfeasor). Di quest'ultima categoria si deve fare particolarmente attenzione perché possono conoscere "il sistema aziendale". Esistono diverse tipologie di Intruder:

- **Cyber criminali**: Sono individui o membri di un gruppo criminale organizzato con l'obiettivo di ottenere una ricompensa finanziaria.
- **Hackttivisti**: Sono individui, di solito operanti come insider, o membri di un gruppo più ampio di attaccanti esterni, motivati da cause sociali o politiche. Lo scopo dei loro attacchi è quello di promuovere e pubblicizzare la loro causa.
- **Organizzazioni sponsorizzate dallo Stato**: Sono gruppi di hacker sponsorizzati dai governi per condurre attività di spionaggio o sabotaggio.
- **Altri**: Sono hacker con motivazioni diverse da quelle sopra elencate. Inoltre, in questa categoria ci sono gli hacker "etici" che dopo aver scovato delle falle in un sistema, informano l'organizzazione "bucata".

Gli Intruder possono essere classificati anche in base alle proprie skill:

- **Apprendisti (script-kiddies)**: Hacker con competenze tecniche minime che utilizzano principalmente toolkit di attacco esistenti (i più facili da cui difendersi).
- **Journeyman**: Hacker con competenze tecniche sufficienti per modificare ed estendere toolkit di attacco per utilizzare vulnerabilità appena scoperte o acquistate (più difficile difendersi da loro rispetto al precedente).
- **Master**: Hacker con competenze tecniche di alto livello in grado di scoprire nuove categorie di vulnerabilità o scrivere nuovi potenti toolkit di attacco (più difficile difendersi da loro rispetto al precedente).

Gli intrusi utilizzano una metodologia di attacco comune:

1. **Identificazione e raccolta di informazioni del target**: l'attaccante identifica e caratterizza i sistemi bersaglio utilizzando informazioni pubblicamente disponibili.
2. **Accesso iniziale**: l'accesso iniziale al sistema target in genere sfruttando una vulnerabilità.
3. **Escalation di privilegi**: azioni malevoli intraprese sul sistema per aumentare i privilegi dell'attaccante.
4. **Sfruttamento del sistema**: azioni dell'attaccante per accedere o modificare le informazioni o le risorse del sistema, o per navigare verso un altro sistema di destinazione.

5. **Mantenimento dell'accesso**: installazione di backdoor o altri software dannosi per permettere all'attaccante di accedere quando vuole al sistema

6. **Copertura delle tracce**: consiste nel rimuovere le prove dell'attacco.

1.2 IDS

Un sistema di **IDS** è un sistema che cerca di individuare le intrusioni. Dall'RFC: "un processo o un sottosistema, implementato in software o hardware, che automatizza le attività di monitoraggio degli eventi che si verificano in una rete di computer e l'analisi degli stessi alla ricerca di segnali di problemi di sicurezza. Un IDS è un sistema di allarme di sicurezza per rilevare ingressi non autorizzati."
Gli elementi principali di un IDS sono la dashboard, la parte che effettua l'analisi per capire se c'è stata un'intrusione e una sonda che raccoglie le informazioni.
Un IDS si basa su tre componenti logici:

- **Sensori**: Sono responsabili della raccolta dei dati grezzi che un IDS andrà ad utilizzare per rilevare eventuali attività non autorizzate. Tipicamente possono esserci molteplici data source relative a quelle parti del sistema che potrebbero contenere prove di un'intrusione (ad esempio pacchetti di rete, file di log e system call traces). I sensori raccolgono e inoltrano queste informazioni all'analizzatore.

- **Analizzatori**: Prendono in ingresso l'output dei sensori e di altri analizzatori per determinare se si è verificata un'intrusione. Nel caso in cui venga rilevata un'intrusione, allora si produce in output un avviso/report che contiene le prove dell'intrusione e l'azione da intraprendere. Gli input del sensore possono anche essere memorizzati per analisi e revisioni future in un componente di archiviazione o database.

- **Interfaccia utente/Componente Manager**: Permette all'operatore di visualizzare gli output del sistema IDS (può anche notificarlo) e controllarne le caratteristiche.

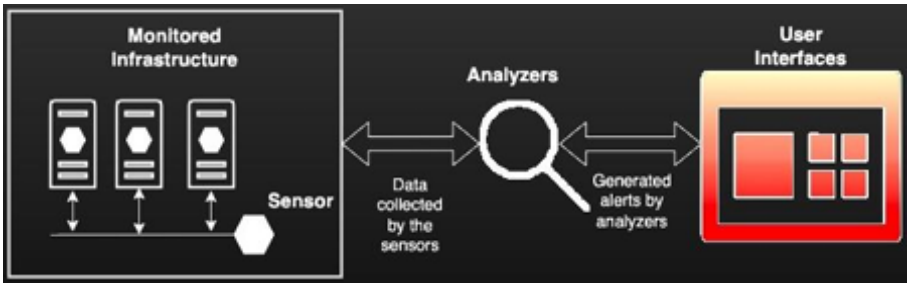


Figura 1.1: Componenti di un IDS

1.2.1 Tipologie

Per quanto riguarda la classificazione in base ai sensori, gli IDS possono essere divisi in tre famiglie:

- **Host-based IDS (HIDS)**: Monitorano le caratteristiche di un singolo host al fine di rilevare attività sospette. Il vantaggio principale di un HIDS è che può rilevare sia intrusioni esterne che interne, cosa impossibile con Network-Based IDS (NIDS) o firewall.
- **Network-based IDS (NIDS)**: Monitorano il traffico in un punto specifico di una rete o nel punto di interconnessione di più reti per rilevare attività sospette. Pone attenzione sulle attività dei protocolli di rete, trasporto e applicativo. I NIDS sono tipicamente inclusi nell'infrastruttura di sicurezza perimetrale o nei firewall di un'organizzazione e si concentrano sui tentativi di intrusione esterni (a differenza di HIDS).
- **IDS distribuiti o ibridi**: Combina le informazioni provenienti da una serie di sensori, spesso sia host che di rete, in un analizzatore centrale in grado di identificare e rispondere meglio alle attività di intrusione. Tipicamente un DIDS utilizza un IDS centrale che combina le informazioni monitorate sugli host dagli HIDS e le informazioni monitorare sul traffico di rete dai NIDS. In altri termini è come se gli HIDS e i NIDS fossero "sensori".

Per quanto riguarda la classificazione in base agli analyzer abbiamo:

- **Rule Based**: sono basati sull'uso di regole e determinano se ci sono degli eventi che non rispettano le regole scritte. Il grosso limite è che se non c'è scritta una regola, allora non vediamo l'intrusione e benché abbiano questo grosso limite vengono molto usati nella pratica.
- **Anomaly Based**: dovrebbero essere quelli che risolvono molti problemi in ambito di cybersecurity perché funzionano in base a come dovrebbe agire normalmente un utente, infine segnalano un comportamento anomalo che si discosta dall'uso tipico di un utente

1.2.2 Caratteristiche e criticità degli IDS

Le strutture di autenticazione, le strutture di controllo degli accessi e i firewall svolgono tutti un ruolo nel contrastare le intrusioni. Gli IDS seguono la strada del rilevamento delle intrusioni. Il motivo dell'utilizzo sempre più massiccio degli IDS in contesti aziendali e non è motivato da una serie di considerazioni, tra cui:

- Se un'intrusione viene rilevata con sufficiente rapidità, l'intruso può essere identificato ed espulso dal sistema prima che vengano causati danni o compromessi dati. Anche se il rilevamento non è sufficientemente tempestivo da impedire l'intrusione, quanto prima viene rilevata l'intrusione, tanto minore sarà l'entità del danno e più rapido sarà il recupero;
- Un IDS efficace può fungere da deterrente, agendo così per prevenire le intrusioni;
- Il rilevamento delle intrusioni consente di raccogliere informazioni sulle tecniche di intrusione che possono essere utilizzate per rafforzare le misure di prevenzione delle intrusioni.

Il rilevamento delle intrusioni si basa sul presupposto che il comportamento dell'intruso differisca da quello di un utente legittimo in modo che possono essere quantificati. Naturalmente, non ci si può aspettare una distinzione netta ed esatta tra l'attacco di un intruso e il normale utilizzo delle risorse da parte di un utente autorizzato. Dobbiamo piuttosto aspettarci che ci sia una certa sovrapposizione.

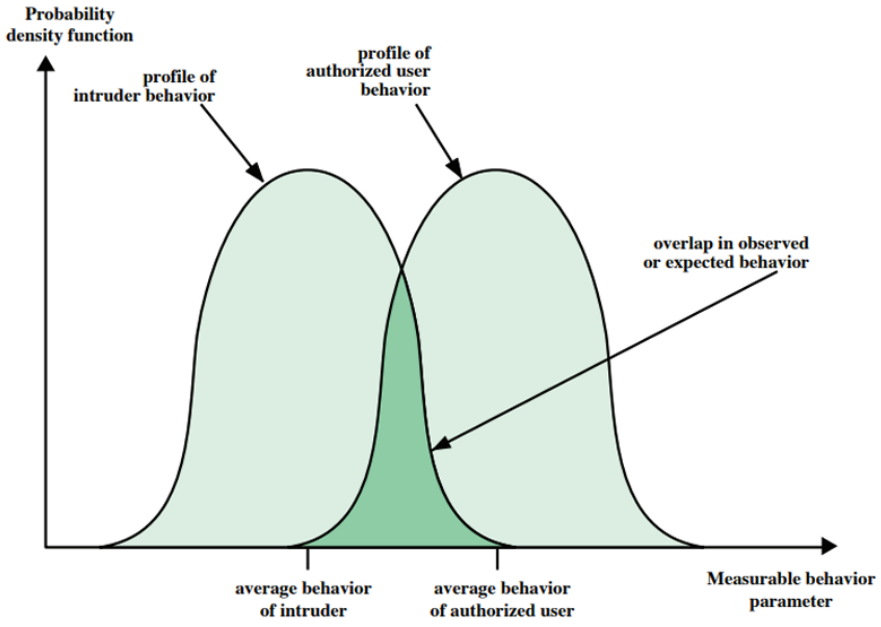


Figura 1.2: Profili di comportamento

Sebbene il comportamento tipico di un intruso differisca dal comportamento tipico di un utente autorizzato, esiste una sovrapposizione di questi comportamenti. Pertanto, un'interpretazione poco rigorosa del comportamento degli intrusi, che catturerà un maggior numero di intrusi, porterà anche a un certo numero di falsi positivi, o falsi allarmi, in cui gli utenti autorizzati vengono identificati come intrusi. D'altro canto, il tentativo di limitare i falsi positivi attraverso un'interpretazione restrittiva del comportamento degli intrusi porterà a un aumento dei falsi negativi, ovvero degli intrusi non identificati come

tali. Idealmente si desidera che un IDS abbia un alto tasso di rilevamento (rapporto tra gli attacchi rilevati e quelli totali), riducendo al minimo il tasso di falsi allarmi (rapporto tra gli attacchi classificati in modo errato e l'utilizzo totale normale).

Base-Rate Fallacy

Per essere utile, un IDS dovrebbe rilevare una percentuale sostanziale di intrusioni mantenendo il tasso di falsi allarmi a un livello accettabile. Se viene rilevata solo una percentuale modesta di intrusioni effettive, il sistema fornisce un falso senso di sicurezza. D'altra parte, se il sistema attiva frequentemente un allarme quando non c'è alcuna intrusione (un falso allarme), i gestori del sistema inizieranno a ignorare gli allarmi, oppure si perderà molto tempo ad analizzare i falsi allarmi. Purtroppo, a causa della natura delle probabilità in gioco, è molto difficile soddisfare lo standard di un alto tasso di rilevamenti con un basso tasso di falsi allarmi. In generale, se il numero effettivo di intrusioni è basso rispetto al numero di utilizzi legittimi di un sistema, il tasso di falsi allarmi sarà elevato. Questo è un esempio di un fenomeno noto come fallacia del tasso di base (**Base-Rate Fallacy**).

Capitolo 2

Dataset

Il dataset NSL-KDD rappresenta un'evoluzione del KDD cup99 dataset. Nel corso degli anni questo dataset è diventato un ottimo benchmark per comparare le prestazioni di vari IDS e per comprendere il modo in cui un IDS possa distinguere il traffico malevolo da quello normale. Infatti, il dataset rappresenta un aspetto cruciale per la realizzazione di un buon IDS: deve avere delle dimensioni e delle caratteristiche che siano quanto più possibile simili al reale traffico dati presente in rete.

2.1 Descrizione del dataset

La raccolta dei dati per il dataset originale KDD Cup 1999 è stata effettuata utilizzando un TCP dump della rete militare DARPA (Defense Advanced Research Projects Agency). Il TCP dump ha catturato il traffico di rete simulato all'interno di un ambiente di test progettato per rappresentare una rete aziendale reale, includendo sia traffico legittimo che traffico generato da attacchi informatici. I dati sono costituiti da 5 milioni di record, ognuno dei quali approssimativamente di 100 bytes.
Il dataset fornito è suddiviso in 4 parti: KDDTest+, KDDTest-21, KDDTrain+, KDDTrain+_20Percent. Il KDDTest-21 è un sottoinsieme del dataset di test, ma privato delle entry più difficili da classificare (Score di 21).
Il dataset è composto da 43 features, di cui 41 si riferiscono al traffico di input mentre le ultime 2 sono l'etichetta (attacco o normale) e lo score (che può essere visto come il grado di difficoltà nella classificazione di quella determinata entry).

2.1.1 Tipologie di attacchi

Il dataset, oltre a stabilire se una determinata entry è relativa a traffico malevolo oppure no, specifica anche la tipologia di attacco a cui si riferisce. In particolare, il dataset è costituito da 4 differenti classi di attacco:

- **Denial of Service**: è un attacco mirato a rendere un servizio, un sistema o una rete inaccessibile agli utenti legittimi. Questo viene ottenuto inondando il bersaglio

con una quantità enorme di traffico o richieste, sovraccaricando le sue risorse e impedendogli di funzionare correttamente. Di conseguenza, il sistema può rallentare drasticamente o smettere completamente di rispondere alle richieste legittime.

- **Probe**: il probing è un attacco in cui l'attaccante mira a ottenere accesso a un computer e ai suoi file attraverso un punto debole o vulnerabilità del sistema.
- **U2R (User to Root)**: conosciuto anche come privilege escalation, è una tipologia di attacco in cui l'attaccante cerca di ottenere dei permessi aggiuntivi (da utente privilegiato) sulla macchina target per compiere azioni malevoli.
- **R2L (Remote to Local)**: è un tipo di attacco in cui l'utente ottiene accesso da remoto alla macchina locale della vittima.

Di seguito è riportata una descrizione delle tipologie di attacco nel dataset:

Classes:	DoS	Probe	U2R	R2L
Sub-Classes:	• apache2 • back • land • neptune • mailbomb • pod • processtable • smurf • teardrop • udpstorm • worm	• ipsweep • mscan • nmap • portsweep • saint • satan	• buffer_overflow • loadmodule • perl • ps • rootkit • sqlattack • xterm	• ftp_write • guess_passwd • httptunnel • imap • multihop • named • phf • sendmail • Snmpgetattack • spy • snmpguess • warezclient • warezmaster • xlock • xsnoop
Total:	11	6	7	15

Figura 2.1: Tipologie di attacco dataset

E' importante notare che alcune tipologie di attacco sono presenti solo nel test set, per simulare i zero-day attack.

2.1.2 Features list

Per quanto riguarda le feature fornite dal dataset, è possibile suddividerle in 4 categorie principali:

- **Basic features**: sono ottenute direttamente dall'header del pacchetto, senza esaminarne il contenuto (duration, protocol_type, service, ecc..);
- **Content features**: sono determinate analizzando il contenuto del pacchetto TCP;

- **Time features**: queste features descrivono informazioni sul traffico considerando finestre temporali della durata di un certo numero di secondi, sotto forma di conteggi o rates;
- **Traffic features**: sono features che descrivono informazioni relative ad un certo gruppo di connessioni (non raggruppate per tempo). Queste features sono introdotte per individuare attacchi che richiedono una finestra più ampia.

La lista completa delle features può essere visualizzata a questo link e nelle figure 2.2 e 2.3.

1	Duration	Length of time duration of the connection
2	Protocol Type	Protocol used in the connection
3	Service	Destination network service used
4	Flag	Status of the connection – Normal or Error
5	Src Bytes	Number of data bytes transferred from source to destination in single connection
6	Dst Bytes	Number of data bytes transferred from destination to source in single connection
7	Land	If source and destination IP addresses and port numbers are equal then, this variable takes value 1 else 0
8	Wrong Fragment	Total number of wrong fragments in this connection
9	Urgent	Number of urgent packets in this connection. Urgent packets are packets with the urgent bit activated
10	Hot	Number of “hot” indicators in the content such as: entering a system directory, creating programs and executing programs
11	Num Failed Logins	Count of failed login attempts
12	Logged In	Login Status : 1 if successfully logged in; 0 otherwise
13	Num Compromised	Number of "compromised” conditions
14	Root Shell	1 if root shell is obtained; 0 otherwise
15	Su Attempted	1 if "su root" command attempted or used; 0 otherwise

(a)

16	Num Root	Number of "root" accesses or number of operations performed as a root in the connection
17	Num File Creations	Number of file creation operations in the connection
18	Num Shells	Number of shell prompts
19	Num Access Files	Number of operations on access control files
20	Num Outbound Cmds	Number of outbound commands in an ftp session
21	Is Hot Logins	1 if the login belongs to the "hot" list i.e., root or admin; else 0
22	Is Guest Login	1 if the login is a "guest" login; 0 otherwise
23	Count	Number of connections to the same destination host as the current connection in the past two seconds
24	Srv Count	Number of connections to the same service (port number) as the current connection in the past two seconds
25	Serror Rate	The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in count (23)
26	Srv Serror Rate	The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in srv_count (24)
27	Rerror Rate	The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in count (23)
28	Srv Rerror Rate	The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in srv_count (24)
29	Same Srv Rate	The percentage of connections that were to the same service, among the connections aggregated in count (23)

(b)

Figura 2.2: Lista features

30	Diff Srv Rate	The percentage of connections that were to different services, among the connections aggregated in count (23)
31	Srv Diff Host Rate	The percentage of connections that were to different destination machines among the connections aggregated in srv_count (24)
32	Dst Host Count	Number of connections having the same destination host IP address
33	Dst Host Srv Count	Number of connections having the same port number
34	Dst Host Same Srv Rate	The percentage of connections that were to different services, among the connections aggregated in dst_host_count (32)
35	Dst Host Diff Srv Rate	The percentage of connections that were to different services, among the connections aggregated in dst_host_count (32)
36	Dst Host Same Src Port Rate	The percentage of connections that were to the same source port, among the connections aggregated in dst_host_srv_count (33)
37	Dst Host Srv Diff Host Rate	The percentage of connections that were to different destination machines, among the connections aggregated in dst_host_srv_count (33)
38	Dst Host Serror Rate	The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in dst_host_count (32)
39	Dst Host Srv Serror Rate	The percent of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in dst_host_srv_count (33)

(a)

40	Dst Host Rerror Rate	The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in dst_host_count (32)
41	Dst Host Srv Rerror Rate	The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in dst_host_srv_count (33)
42	Class	Classification of the traffic input
43	Difficulty Level	Difficulty level

(b)

Figura 2.3: Lista features

Merita un'analisi separata la feature *flag*. Quest'ultima rappresenta lo stato della connessione (determinato dal valore del flag). In figura 2.4 è riportata una breve descrizione per ogni possibile valore:

Flag	Value	Flag	Description
SF	Normal establishment and termination. Note that this is the same symbol as for state S1. You can tell the two apart because for S1 there will not be any byte counts in the summary, while for SF there will be	RSTO	Connection reset by the originator
REJ	Connection attempt rejected	RSTR	Connection reset by the responder
S0	Connection attempt seen, no reply	OTH	No SYN seen, just midstream traffic (a “partial connection” that was not later closed)
S1	Connection established, not terminated	RSTOS0	Originator sent a SYN followed by a RST, we never saw a SYN-ACK from the responder
S2	Connection established and close attempt by originator seen (but no reply from responder)	SH	Originator sent a SYN followed by a FIN, we never saw a SYN ACK from the responder (hence the connection was “half” open)
S3	Connection established and close attempt by responder seen (but no reply from originator)	SHR	Responder sent a SYN ACK followed by a FIN, we never saw a SYN from the originator. (Not in NSL-KDD but still a flag)

Figura 2.4: Analisi valori feature flag

Capitolo 3

Implementazione

In questo capitolo viene presentata la soluzione adottata. Il progetto comprende un'analisi preliminare del dataset, una successiva analisi esplorativa, lo studio delle features utili alla classificazione, un preprocessing dei dati per prepararli al processo di classificazione e una valutazione dei risultati ottenuti.

3.1 Import librerie

In questa fase si procede con l'import delle librerie utili nella realizzazione del progetto e a una lettura del dataset, fornito in formato txt.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os
import itertools

import warnings

from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.metrics import recall_score, precision_score, f1_score
from sklearn.metrics import roc_auc_score

from sklearn.model_selection import train_test_split
from sklearn.feature_selection import mutual_info_classif
from sklearn.feature_selection import SelectKBest
```

Listing 3.1: Import librerie

3.2 Lettura del dataset

Segue una lettura del dataset, che viene fornito senza i nomi delle features sulle colonne.

```
directory = r"C:\NS_Project\NSL-KDD_Dataset"

train_filename = r"KDDTrain+.txt"
test_filename = r"KDDTest+.txt"

train_path = os.path.join(directory, train_filename)
test_path = os.path.join(directory, test_filename)

df = pd.read_csv(train_path)
df_test = pd.read_csv(test_path)

columns = (['duration'
,'protocol_type'
,'service'
...
,'attack'
,'level'])

df.columns = columns
df_test.columns = columns

df_0 = df.copy()
df_test_0 = df_test.copy()
```

Listing 3.2: Lettura del dataset

3.2.1 Statistiche numeriche features

Si procede al calcolo di diverse statistiche numeriche sulle features, tra cui media, quartili e deviazione standard. Le features con deviazione standard bassa non verranno considerate ai fini della classificazione in quanto non portatrici di informazioni.

	count	mean	std	min	25%	50%	75%	max
duration	125972.00	287.15	2604.53	0.00	0.00	0.00	0.00	42908.00
src_bytes	125972.00	45567.10	5870354.48	0.00	0.00	44.00	276.00	1379963888.00
dst_bytes	125972.00	19779.27	4021285.11	0.00	0.00	0.00	516.00	1309937401.00
land	125972.00	0.00	0.01	0.00	0.00	0.00	0.00	1.00
wrong_fragment	125972.00	0.02	0.25	0.00	0.00	0.00	0.00	3.00
urgent	125972.00	0.00	0.01	0.00	0.00	0.00	0.00	3.00
hot	125972.00	0.20	2.15	0.00	0.00	0.00	0.00	77.00
num_failed_logins	125972.00	0.00	0.05	0.00	0.00	0.00	0.00	5.00
logged_in	125972.00	0.40	0.49	0.00	0.00	0.00	1.00	1.00
num_compromised	125972.00	0.28	23.94	0.00	0.00	0.00	0.00	7479.00
root_shell	125972.00	0.00	0.04	0.00	0.00	0.00	0.00	1.00
su_attempted	125972.00	0.00	0.05	0.00	0.00	0.00	0.00	2.00
num_root	125972.00	0.30	24.40	0.00	0.00	0.00	0.00	7468.00
num_file_creations	125972.00	0.01	0.48	0.00	0.00	0.00	0.00	43.00
num_shells	125972.00	0.00	0.02	0.00	0.00	0.00	0.00	2.00
num_access_files	125972.00	0.00	0.10	0.00	0.00	0.00	0.00	9.00
num_outbound_cmds	125972.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
is_host_login	125972.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00
is_guest_login	125972.00	0.01	0.10	0.00	0.00	0.00	0.00	1.00
count	125972.00	84.11	114.51	0.00	2.00	14.00	143.00	511.00
srv_count	125972.00	27.74	72.64	0.00	2.00	8.00	18.00	511.00
serror_rate	125972.00	0.28	0.45	0.00	0.00	0.00	1.00	1.00

Figura 3.1: Statistiche numeriche features

3.2.2 Matrice di correlazione

Viene calcolata la matrice di correlazione per le features numeriche. Per migliorare le prestazioni del classificatore e per ridurre la dimensionalità del dataset, vengono individuate le coppie di features altamente correlate, in quanto portano la stessa informazione e sono dunque ridondanti. Di seguito, per motivi di spazio, viene riportata solo un estratto della matrice.

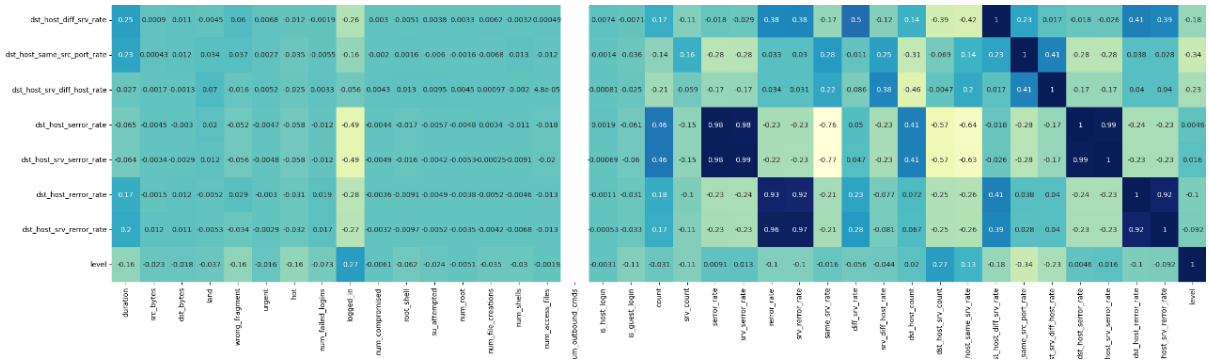


Figura 3.2: Estratto matrice di correlazione

Si individuano le seguenti coppie con correlazione $-0.95 \leq corr \leq 0.95$.

	Feature 1	Feature 2	Correlation
0	num_compromised	num_root	1.00
1	num_root	num_compromised	1.00
2	serror_rate	srv_serror_rate	0.99
3	serror_rate	dst_host_serror_rate	0.98
4	serror_rate	dst_host_srv_serror_rate	0.98
5	srv_serror_rate	serror_rate	0.99
6	srv_serror_rate	dst_host_serror_rate	0.98
7	srv_serror_rate	dst_host_srv_serror_rate	0.99
8	rerror_rate	srv_rerror_rate	0.99
9	rerror_rate	dst_host_srv_rerror_rate	0.96
10	srv_rerror_rate	rerror_rate	0.99
11	srv_rerror_rate	dst_host_srv_rerror_rate	0.97
12	dst_host_serror_rate	serror_rate	0.98
13	dst_host_serror_rate	srv_serror_rate	0.98
14	dst_host_serror_rate	dst_host_srv_serror_rate	0.99
15	dst_host_srv_serror_rate	serror_rate	0.98
16	dst_host_srv_serror_rate	srv_serror_rate	0.99
17	dst_host_srv_serror_rate	dst_host_serror_rate	0.99
18	dst_host_srv_rerror_rate	rerror_rate	0.96
19	dst_host_srv_rerror_rate	srv_rerror_rate	0.97

Figura 3.3: Features altamente correlate

Interessante notare come le connessioni definite "compromesse" a livello di sicurezza siano anche quelle associate a un accesso root. La feature *num_ compromised* indica quante variabili, come file o processi, sono state compromesse durante una connessione specifica. Una variabile è considerata compromessa quando è stata oggetto di accesso non autorizzato, alterazioni, o è stata usata per scopi malevoli.
Seguono correlazioni legate a informazioni specifiche delle varie connessioni. Di fondamentale importanza sono le features *count*, che indica il numero di tentativi di connessioni verso lo stesso host di destinazione della connessione corrente, negli ultimi 2 secondi e *srv_ count*, che invece indica il numero di connessioni verso lo stesso servizio (port number) della connessione corrente, negli ultimi 2 secondi. Dal numero di connessioni individuate da queste due features, si basano le altre features. Ad esempio, se consideriamo le coppie di feature altamente correlate *serror rate* e *srv serror rate*, la prima indica la percentuale di connessioni con flag di errore, tra le connessioni individuate dalla feature *count*, mentre la seconda indica la stessa quantità, ma riferendosi alle connessioni individuate dalla feature *srv_ count*. In sostanza quindi c'è una correlazione tra il numero di connessioni verso lo stesso host e quello verso lo stesso servizio. Dalle prime analisi, si potrebbe pensare che questo sia dovuto alla presenza piuttosto numerosa di attacchi DoS, che tipicamente vengono realizzati verso lo stesso host target, sfruttando uno specifico servizio (port number).

3.3 Data Cleaning

In questa fase si procede all'individuazione di eventuali valori nulli e duplicati che in questo caso sono assenti. Poi si individuano i valori unici per le feature categoriche.

```
Column: service
------------------------------
Unique Values (70): ['other' 'private' 'http' 'remote_job' 'ftp_data' 'name' 'netbios_ns'
 'eco_i' 'mtp' 'telnet' 'finger' 'domain_u' 'supdup' 'uucp_path' 'Z39_50'
 'smtp' 'csnet_ns' 'uucp' 'netbios_dgm' 'urp_i' 'auth' 'domain' 'ftp'
 'bgp' 'ldap' 'ecr_i' 'gopher' 'vmnet' 'systat' 'http_443' 'efs' 'whois'
 'imap4' 'iso_tsap' 'echo' 'klogin' 'link' 'sunrpc' 'login' 'kshell'
 'sql_net' 'time' 'hostnames' 'exec' 'ntp_u' 'discard' 'nntp' 'courier'
 'ctf' 'ssh' 'daytime' 'shell' 'netstat' 'pop_3' 'nnsp' 'IRC' 'pop_2'
 'printer' 'tim_i' 'pm_dump' 'red_i' 'netbios_ssn' 'rje' 'X11' 'urh_i'
 'http_8001' 'aol' 'http_2784' 'tftp_u' 'harvest']

Value Counts:
service
http         40338
private      21853
domain_u      9043
smtp          7313
ftp_data      6859
             ...
tftp_u           3
http_8001        2
aol              2
harvest          2
http_2784        1
Name: count, Length: 70, dtype: int64
====================================
```

Figura 3.4: Valori unici feature *Service*

Segue inoltre un'analisi degli outlier mediante box plot. Tale analisi ha evidenziato l'importanza degli outlier ai fini della discriminazione da parte del classificatore, per cui si è ritenuto necessario mantenere le feature che presentavano un numero di outlier significativo, senza rimuovere questi ultimi.

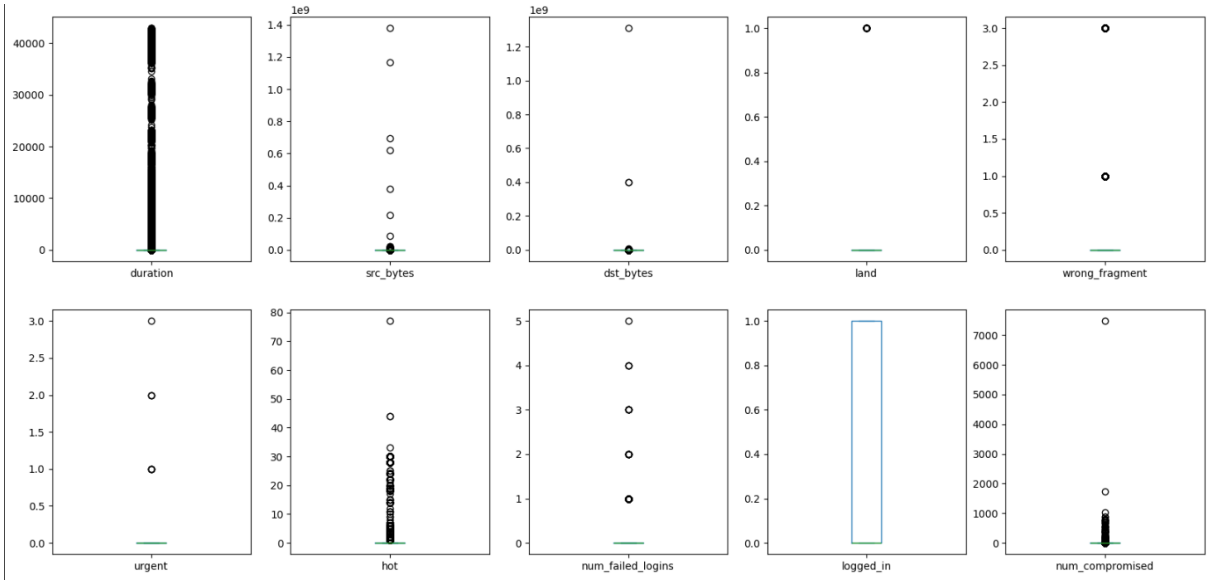


Figura 3.5: Box plots

3.4 Analisi esplorativa

In questa fase viene eseguita un'analisi approfondita delle caratteristiche del dataset, cercando di individuare features portatrici di informazione.

3.4.1 Tipo protocollo

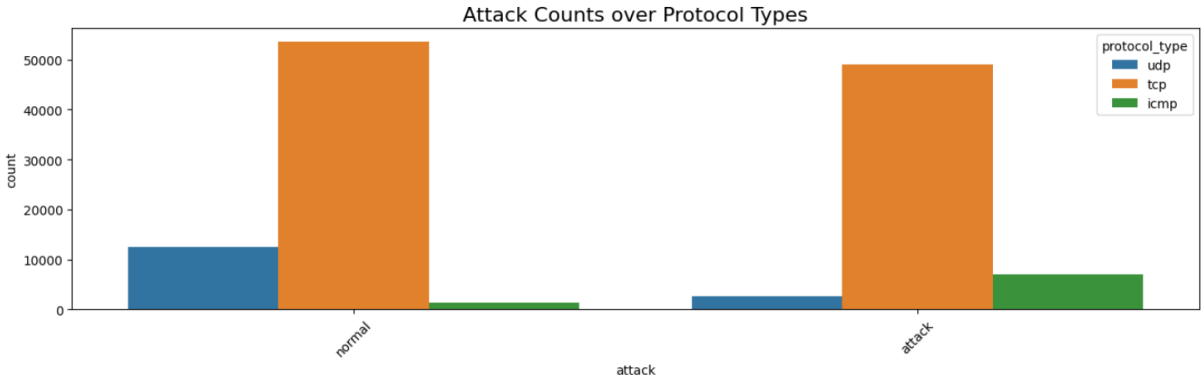
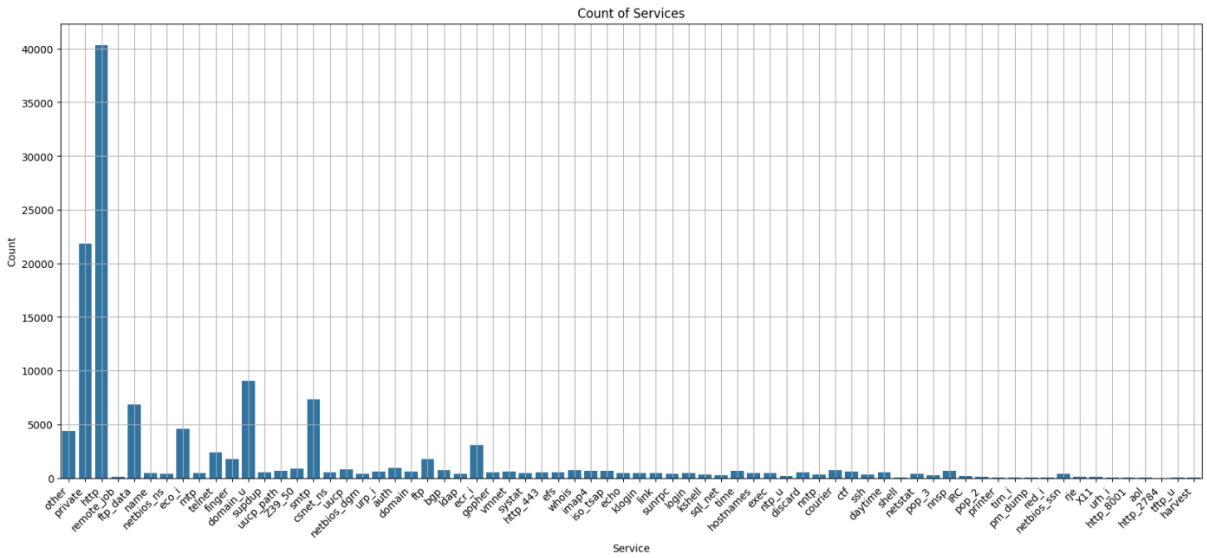


Figura 3.6: Distribuzione degli attacchi per protocollo

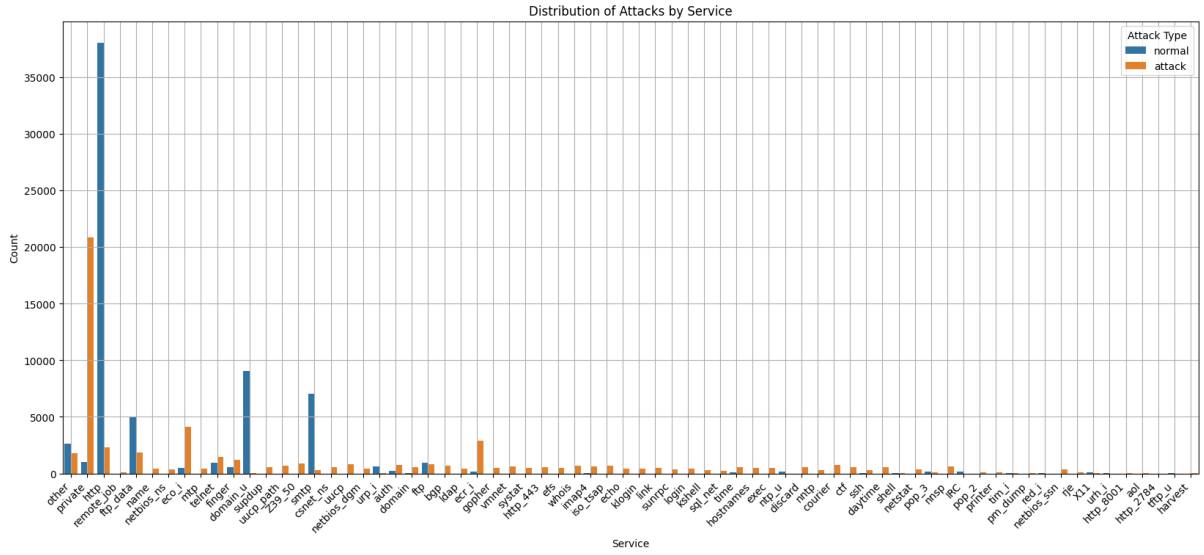
Si osserva come il protocollo TCP sia quello maggiormente coinvolto, essendo le sue entries più numerose. Il protocollo ICMP contribuisce poco al dataset, ma quasi tutte le sue entries sono relative ad attacchi.

3.4.2 Service

Servizi usati in generali



Servizi coinvolti in attacchi

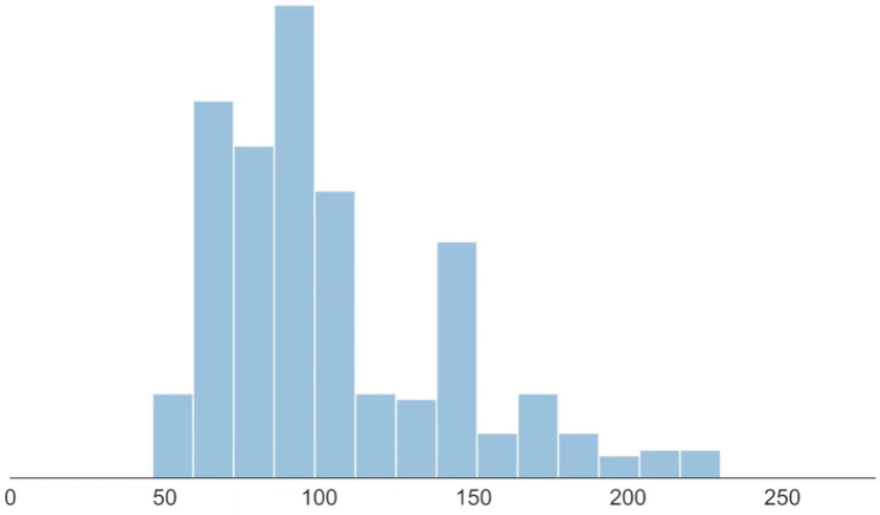


Osserviamo come la maggior parte degli attacchi abbiano come servizio *private*. Quando la feature *service* assume il valore *private*, significa che la connessione è diretta a un servizio personalizzato o non pubblico, che potrebbe essere più vulnerabile a intrusioni a causa della sua natura meno conosciuta o documentata. Questa informazione può essere dunque utile in fase di classificazione.

3.4.3 KDE Plot Duration by Flag

Kernel Density Estimation

La KDE è una tecnica che permette di ricavare la PDF da un insieme di dati di cui disponiamo. Generalmente, quando disponiamo di un insieme di dati, è utile andare a stimare la distribuzione dalla quale i dati provengono. Una tecnica semplice è quella di ricavare l'istogramma.



Si può anche ricavare una versione continua "smoothed" di questo istogramma, attraverso la KDE. La KDE è una tecnica non parametrica, in quanto non fa alcuna assunzione sulla distribuzione sottostante. Costruire la KDE è piuttosto semplice poiché, a partire da un insieme finito di dati è possibile aggiungere un kernel (in questo caso una gaussiana) centrata su un punto, e fare questo per tutti i data-point.

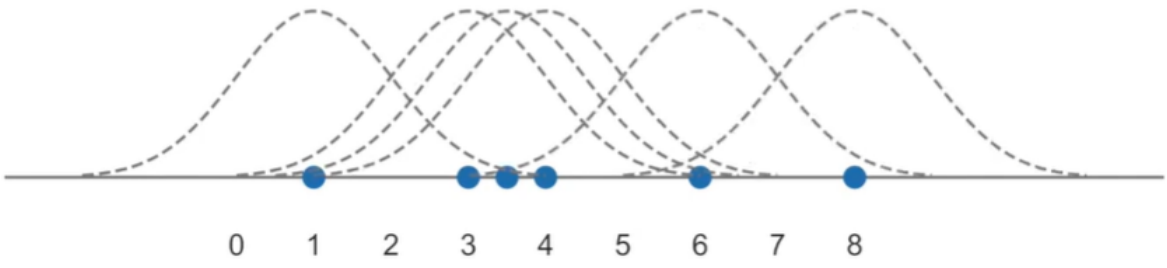
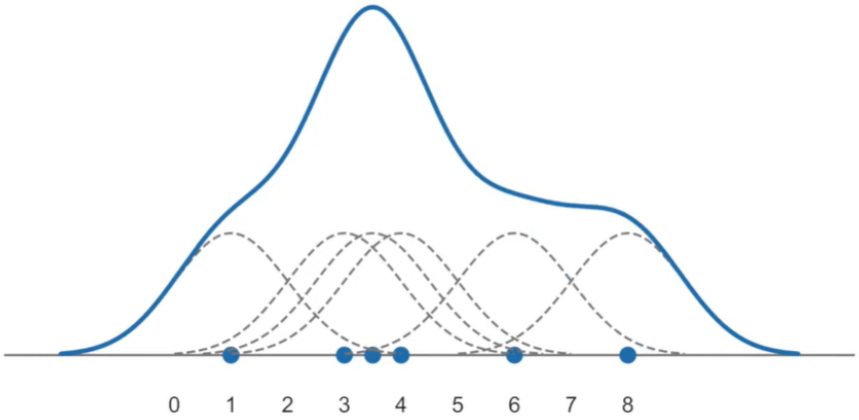


Figura 3.7: Kernel gaussiano

Negli intervalli di sovrapposizione tra kernel relativi a punti diversi semplicemente i kernel vengono sommati. I risultato finale sarà quindi una stima della PDF dei dati.



In particolare, la KDE viene utilizzata per avere maggiori informazioni sulla distribuzione del campo *duration*, che indica la durata della connessione, rispetto al valore dei flag.

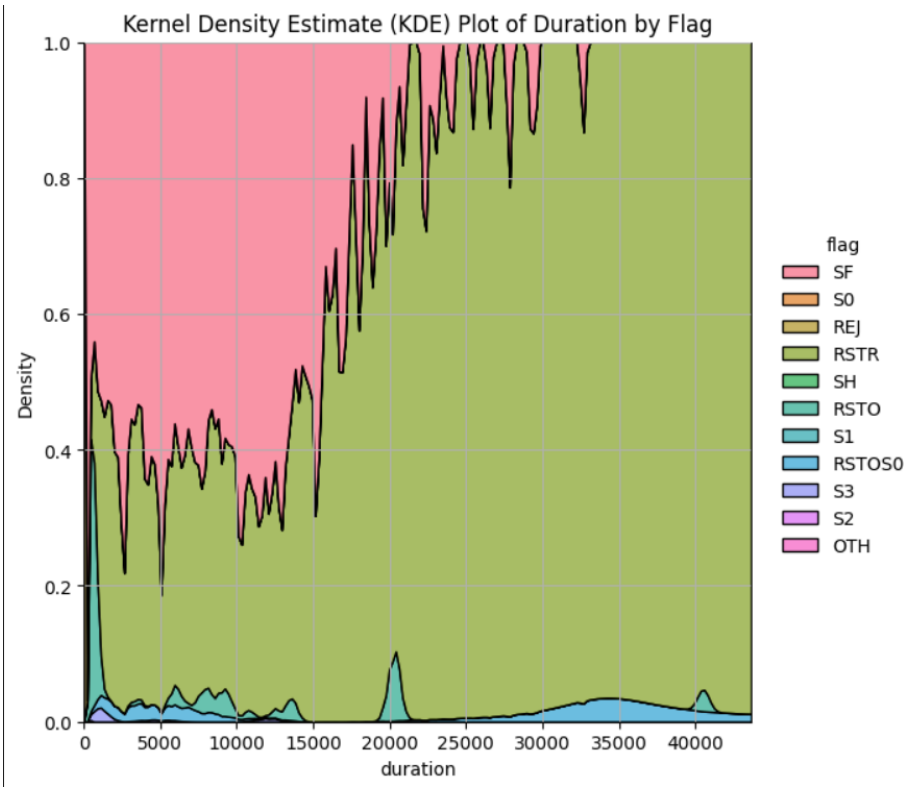


Figura 3.8: KDE Plot Duration by Flag

Dalla KDE si può notare il colore rosso dello sfondo, legato al fatto che sono presenti numerose connessioni con il flag SF, associato a connessioni che sono state aperte e chiuse con successo. Inoltre si può notare un picco in corrispondenza di valori di *duration* elevati per quanto riguarda i flag RSTO e RSTR. Questi flag indicano che la connessione è stata chiusa dal mittente o dal destinatario, in seguito magari ad un timeout.

3.4.4 Analisi tipo attacco per protocollo

protocol_type	icmp	tcp	udp
attack_type			
back	0	956	0
buffer_overflow	0	30	0
ftp_write	0	8	0
guess_passwd	0	53	0
imap	0	11	0
ipsweep	3117	482	0
land	0	18	0
loadmodule	0	9	0
multihop	0	7	0
neptune	0	41214	0
nmap	981	265	247
normal	1309	53599	12434
perl	0	3	0
phf	0	4	0
pod	201	0	0
portsweep	5	2926	0
rootkit	0	7	3
satan	32	2184	1417
smurf	2646	0	0
spy	0	2	0
teardrop	0	0	892
warezclient	0	890	0
warezmaster	0	20	0

Dalla tabella si può notare come ci siano protocolli che sono più spesso target di attacco. Inoltre, alcuni attacchi come satan, nmap e ipsweep sono attacchi cross-protocol.
Notiamo che l'attacco coinvolto nel maggior numero di entry è *Neptune*. Di seguito sono riportate delle brevi descrizioni dei tipi di attacco maggiormente coinvolti.

Neptune

Neptune è un tipo di attacco DoS che sfrutta delle vulnerabilità nella gestione delle connessioni TCP. In un attacco Neptune, l'attaccante invia una serie di pacchetti SYN alla vittima, che è la prima parte di una connessione TCP. Tuttavia, l'attaccante non completa il processo di hanshake TCP, cioè non invia la risposta finale ACK necessaria per stabilire una connessione. Questo lascia il sistema di destinazione con una serie di connessioni incomplete in uno stato semi-aperto.
Ogni connessione aperta consuma risorse di sistema, come la memoria, e se il sistema riceve abbastanza di questi pacchetti SYN senza mai ricevere il pacchetto ACK finale, le risorse del sistema possono essere esaurite. Questo porta il sistema a non essere in grado di gestire nuove connessioni legittime, causando una forma di DoS, dove gli utenti legittimi non riescono ad accedere ai servizi offerti dal sistema attaccato. Dunque, Neptune è un tipo specifico di attacco SYN flood.

Satan

Satan è un attacco di tipo Probe che sfrutta uno strumento chiamato SATAN (Security Administrator Tool for Analyzing Networks), che è progettato per rilevare vulnerabilità di sicurezza in una rete.
SATAN è uno strumento che esegue una scansione di rete per identificare macchine attive, porte aperte, servizi in esecuzione e potenziali vulnerabilità note (come configurazioni errate, versioni obsolete di software, o porte non sicure). Lo strumento automatizza il processo di scansione e fornisce un rapporto dettagliato delle vulnerabilità scoperte, classificandole in base al livello di rischio.
L'obiettivo principale dell'attacco *satan* è raccogliere informazioni dettagliate su una rete target.

Smurf

Smurf è un attacco DoS che sfrutta le funzionalità di broadcast delle reti IP. L'attacco è tipicamente suddiviso in 3 fasi:

1. **Invio di pacchetti**: l'attaccante invia pacchetti ICMP di tipo echo request a un indirizzo IP broadcast;

2. **Amplificazione**: i dispositivi della rete rispondono a questi pacchetti di broadcast con pacchetti di echo reply, che vengono poi inviati all'indirizzo IP di origine specificato dall'attaccante, che è l'indirizzo IP della vittima (magari sottoposto a spoofing);

3. **Saturazione della banda**: questo provoca una grande quantità di traffico di risposta al target, saturandone la banda e i servizi, rendendo il sistema o la rete inutilizzabile.

3.4.5 Analisi flag

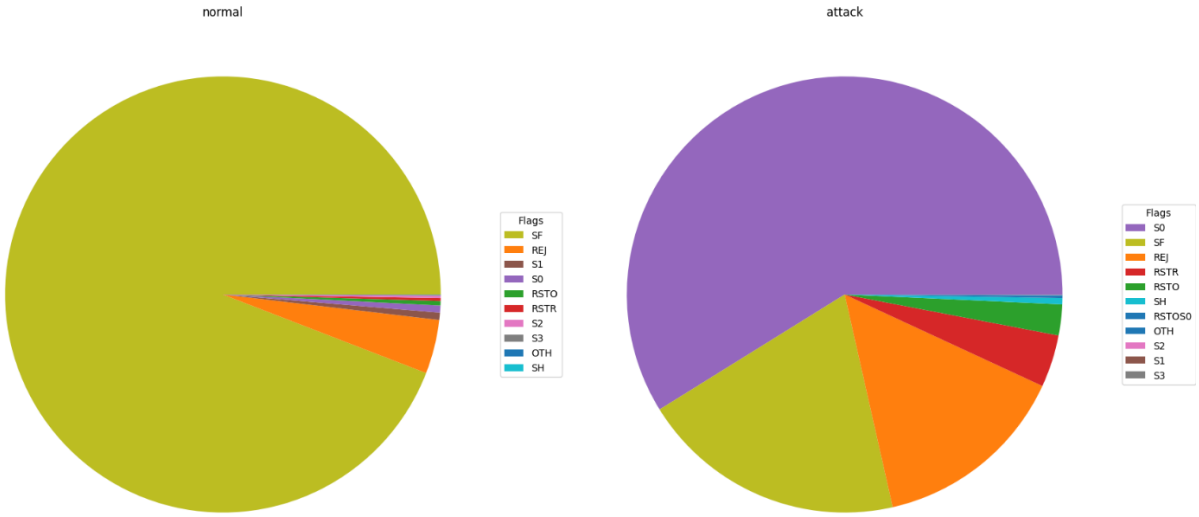


Figura 3.9: Distribuzione flag

Come era lecito aspettarsi, il traffico normale è caratterizzato per la maggior parte dal flag SF attivo, il quale indica che la connessione è stata stabilita ed è terminata con successo. Gli attacchi invece sono caratterizzati per la maggior parte dal flag S0 attivo, il quale indica che c'è stato un tentativo di connessione, ma senza risposta. Tuttavia, anche una parte significativa degli attacchi è caratterizzata dal flag SF attivo. Nonostante ciò, la features flag sembra avere abbastanza potere discriminativo.
A seguito di questa analisi esplorativa, le features protocol, flag e service sembrano essere utili per un eventuale problema di classificazione. Tale supposizione tuttavia dovrà essere confermata dalla tecniche di feature selection che verranno utilizzate successivamente.

3.5 Pre-processing

In questa fase, i dati vengono preparati per la classificazione finale.

3.5.1 Encoding

Le feature categoriche *protocol_ type*, *service*, *flag* e *attack* vengono convertite in feature numeriche.

```
cat_features = df.select_dtypes(include='object').columns
cat_features = cat_features.drop('attack_type')

for feature in cat_features:

    unique_cats = pd.concat([df[feature], df_test[feature]]).unique
        ()

    cat_mapping = {category: code for code, category in enumerate(
        unique_cats)}

    mapping_df = pd.DataFrame(list(cat_mapping.items()), columns=[
        feature, 'Code'])

    df[feature] = df[feature].map(cat_mapping)
    df_test[feature] = df_test[feature].map(cat_mapping)
```

3.5.2 Split train e validation

Il dataset viene suddiviso in train e validation set con percentuali 80-20.

```
df_train, df_val, y_train, y_val = train_test_split(df, y, test_size
    = 0.2, random_state = 43)
```

3.5.3 Feature Engineering e Selection

In questa fase si utilizza il criterio della mutua informazione per scegliere le features da utilizzare come input per il modello di machine learning.

Mutua Informazione

La **mutua informazione** è un valore non-negativo che misura la mutua dipendenza tra due variabili aleatorie. In particolare, è una misura statistica che quantifica la dipendenza tra due variabili. Nel contesto della selezione delle features in un problema di machine learning, la mutua informazione viene utilizzata per valutare quanto ciascuna feature sia informativa rispetto alla variabile di output (ad esempio, la classe da predire).

Se la mutua informazione tra una feature e l'output è alta, significa che la feature fornisce molte informazioni utili per prevedere l'output e, quindi, è una buona candidata

per essere inclusa nel modello. Al contrario, se la mutua informazione è bassa, la feature non aggiunge molto valore predittivo e potrebbe essere scartata.

Questa grandezza è una buona alternativa al coefficiente di correlazione di Pearson, poiché è in grado di misurare non solo relazioni lineari. Inoltre, è utilizzabile sia per variabili continue che per variabili discrete.

3.5.4 Features scelte

In figura 3.10 sono riportate le features ordinate secondo il criterio della mutua informazione. Essendo il dataset piuttosto grande, si è deciso di scegliere 15 features sulle 41 inizialmente disponibili.

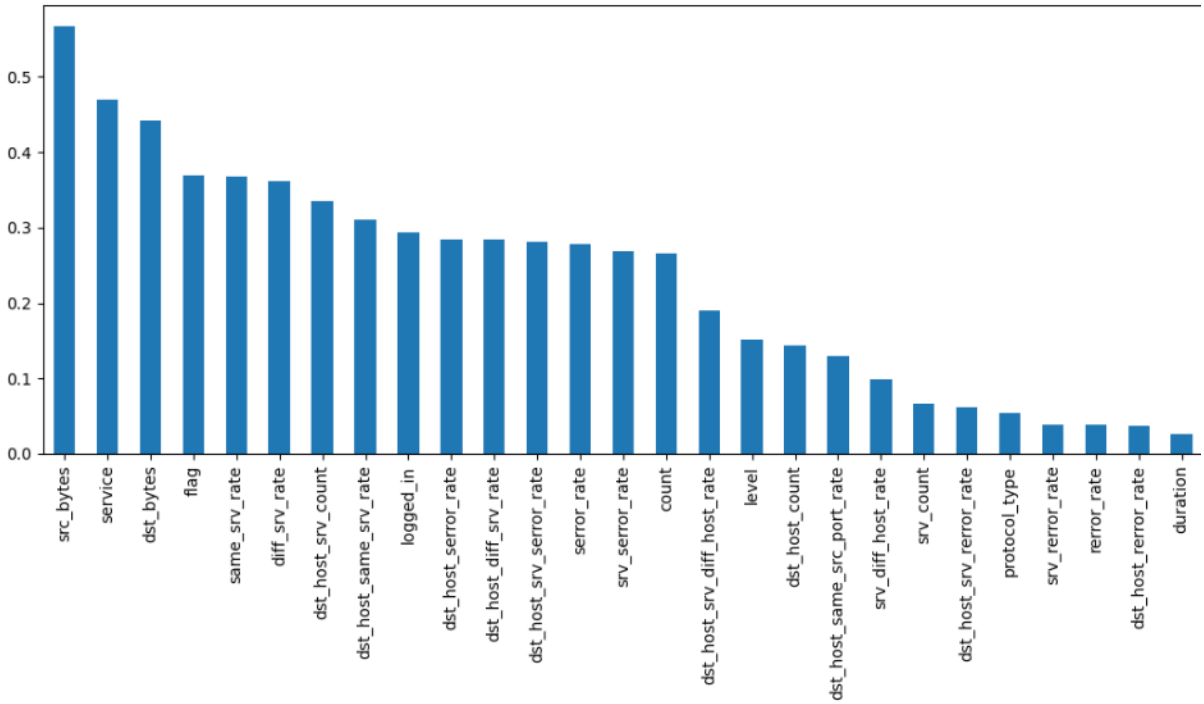


Figura 3.10: Features ordinate per valore di mutua informazione

Successivamente, si è proceduto al calcolo della correlazione tra le features scelte, in modo da capire se fosse possibile scegliere un sottoinsieme ancora più piccolo, mantenendo però lo stesso contenuto informativo.

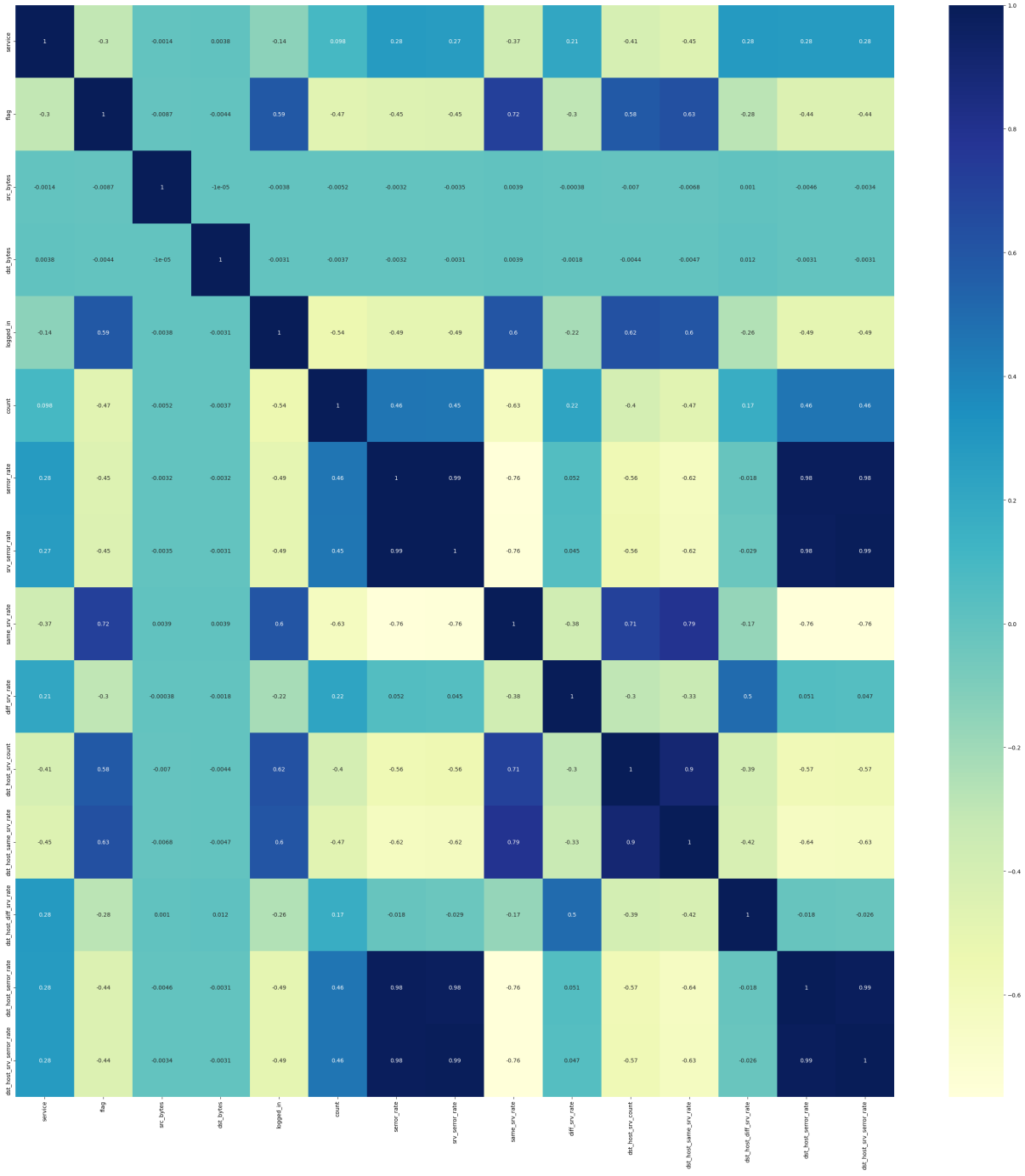


Figura 3.11: Matrice di correlazione tra le 15 features scelte

In seguito all'analisi della matrice, si sono individuate alcune coppie di features altamente correlate, per cui da 15 features sono state individuate le 12 features finali da mandare in input al modello. Le **features scelte** sono le seguenti: *service, flag, src_bytes, dst_bytes, logged_in, count, serror_rate, same_srv_rate, diff_srv_rate, dst_host_srv_count, dst_host_same_srv_rate* e *dst_host_diff_srv_rate*.

3.5.5 Normalizzazione

La fase di normalizzazione non è necessaria in caso di utilizzo di un modello di machine learning basato su alberi di decisione, in quanto:

- **Struttura dell'albero**: un albero di decisione suddivide i dati basandosi su soglie nei valori delle features, senza fare affidamento su metriche di distanza. Le decisioni vengono prese confrontando i valori di una singola features con una soglia, indipendentemente dall'unità di misura o dalla scala di quella feature:
- **Scelte di suddivisione**: gli algoritmi per costruire un albero di decisione scelgo i punti di split massimizzando criteri come l'entropia o l'information gain. Questi criteri non sono influenzati dalla scala delle features poiché considerando solo l'ordine dei valori e non le loro dimensioni relative;
- **Impatto della scala**: se una feature ha valori molto grandi, questo non le conferisce un'importanza maggiore nell'albero rispetto a una feature con valori più piccoli. Gli alberi di decisione non assegnano pesi alle feature in base alla loro scala, ma si basano esclusivamente sulla capacità di una features di separare i dati in base alla variabile target.

Tuttavia, è stato anche utilizzando il percettrone multilivello, che invece richiede una fase di normalizzazione, per cui è stata implementata. In particolare, è stata introdotta una normalizzazione **Z-Score**. Questo tipo di normalizzazione sottrae la media da ogni feature e divide per la deviazione in standard in modo da ottenere feature a media nulla e varianza (o deviazione standard) unitaria, rispettando la seguente formula:

$$X_{\text{norm}} = \frac{X - \mu}{\sigma} \tag{3.1}$$

Infine, è riportato il codice per l'implementazione.

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

df_norm = scaler.fit_transform(df)
df_test_norm = scaler.transform(df_test)
df_val_norm = scaler.transform(df_val)
```

Listing 3.3: Normalizzazione Z-Score

3.6 Modello

Una volta conclusa la fase di pre-processing del dataset, in cui si sono effettuate una serie di operazioni utili a ottenere un input ottimizzato per il problema di machine learning, si è proceduto con la scelta del modello.

3.6.1 XGBoost

XGBoost, acronimo di **eXtreme Gradient Boosting** è un algoritmo di machine learning basato su tecniche di boosting, una tecnica che combina più modelli deboli (ad esempio, alberi di decisione) per creare un modello forte. L'idea chiave del boosting è costruire modelli in modo sequenziale, dove ogni nuovo modello cerca di correggere gli errori del modello precedente. Questo processo avviene iterativamente, aggiungendo nuovi modelli che correggono gli errori residui del modello complessivo.

Nel caso del **Gradient Boosting**, ogni nuovo modello è addestrato a minimizzare la funzione di perdita (che misura l'errore tra le previsioni del modello e i dati reali) usando il gradiente di quella funzione. In questo contesto il gradiente rappresenta la direzione in cui bisogna muoversi per ridurre l'errore.

Funzionamento del modello

XGBoost introduce diverse innovazioni rispetto al tradizionale Gradient Boosting:

- **Regolarizzazione**: XGBoost introduce termini di regolarizzazione L1 ed L2 per controllare la complessità del modello. Questo aiuta a prevenire l'overfitting, rendendo il modello più generalizzabile;
- **Tree Pruning**: mentre altri algoritmi fanno crescere gli alberi di decisione fino ad un certo livello e poi li potano, XGBoost utilizza una tecnica chiamata **max depth** combinata con **pruning by minimum loss**. Questo significa che durante la crescita dell'albero, i rami che non contribuiscono sufficientemente alla riduzione dell'errore vengono tagliati immediatamente;
- **Gestione dei missing values**: XGBoost gestisce efficacemente i valori mancanti assegnando automaticamente le istanze mancanti al lato del nodo che minimizza la perdita durante la fase di addestramento.

Tuning degli iperparametri

I principali iperparametri da settare sono:

- **n_estimators**: numero di alberi da costruire;
- **max_depth**: profondità massima degli alberi;

- **learning_rate**;
- **subsample**: frazione dei campioni utilizzata per addestrare l'albero;
- **colsample_bytree**: frazione di feature usate per costruire ciascun albero.

```
param_grid = {
    "n_estimators": [50,64,100,128],
    "max_depth": [2, 3, 4,5,6],
    "learning_rate": [0.01,0,0.03, 0.05, 0.1],
    "subsample": [0.5, 0.8],
    "colsample_bytree": [0.5, 0.8]
}
```

Si è deciso di procedere con una ricerca a griglia, dove vengono provate tutte le possibili di combinazioni tra i valori degli iperparametri da settare. Nel caso specifico, con 4 valori per *n_estimators*, 5 valori per *max_depth*, 5 valori per *learning_rate*, 2 valori per *subsample* e 2 valori per *colsample_bytree*, si avranno 4x5x5x2x2 = 400 coombinazioni diverse.

```
from sklearn.model_selection import GridSearchCV

XGB_model = XGBClassifier(random_state=42)

XGB_grid_model = GridSearchCV(XGB_model,
                          param_grid,
                          scoring="f1",
                          n_jobs=-1,
                          return_train_score=True).fit(X_train,
                             y_train)
```

I risultati migliori sono stati ottenuti con i valori degli iperparametri riportati in figura 3.12.

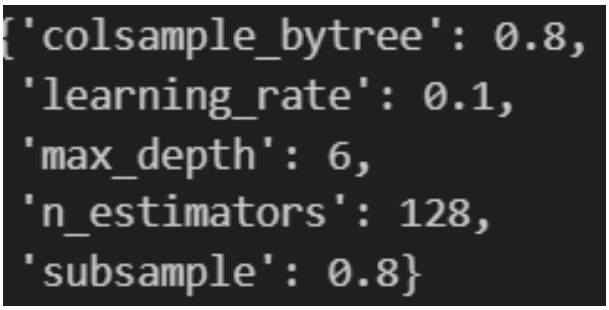
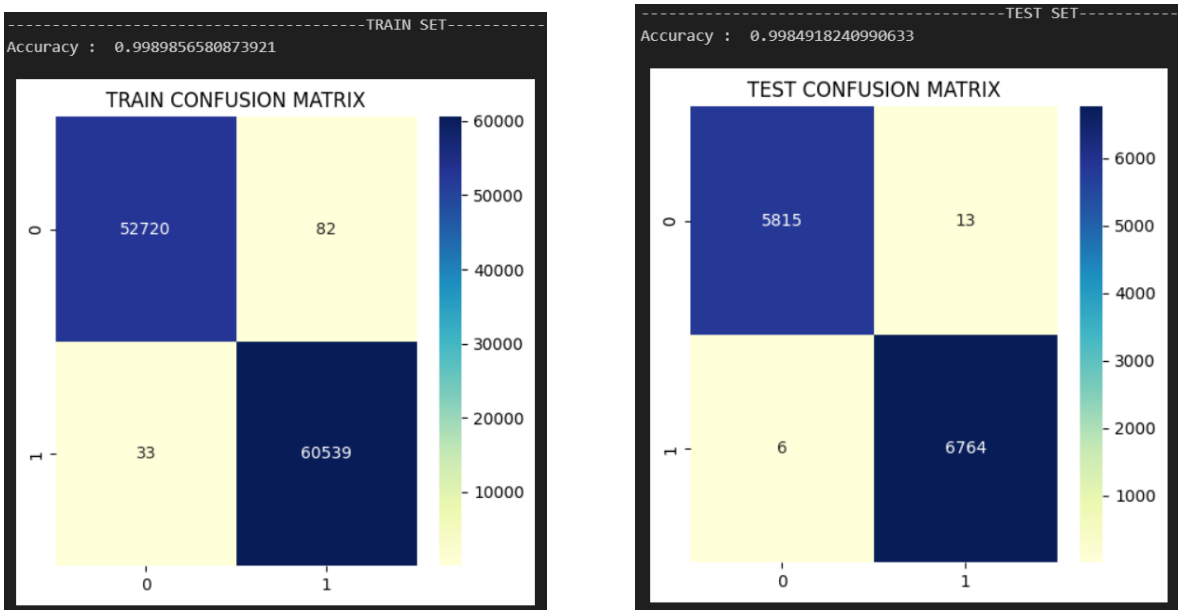


Figura 3.12: Hyperparameters best values

Evaluation

Di seguito sono riportati i risultati ottenuti con i valori degli iperparametri appena presentati.



(a) Risultati Train Set

(b) Risultati Validation Set

Figura 3.13: Risultati XGBoost con iperparametri ottimi

Si osserva come, sia per il train che per il validation, **otteniamo un numero di falsi negativi esiguo**, e in generale otteniamo un numero di falsi positivi maggiore. Questa è una proprietà fondamentale per un IDS, in quanto è preferibile avere più falsi positivi che falsi negativi. Infatti, i falsi negativi sono molto più pericolosi poiché un'intrusione reale può passare inosservata, permettendo agli attaccanti di compromettere la sicurezza del sistema, rubare dati o causare danni senza essere rilevati. Inoltre, è preferibile tollerare un certo numero di falsi positivi per garantire che tutte le minacce potenziali siano rilevate.

Feature importance

Un importante beneficio che si ha nell'utilizzo del gradient boosting è che, dopo che gli alberi sono stai costruiti, è abbastanza immediato recuperare l'importanza relativa di ciascuna feature.

Generalmente, l'importanza fornisce un punteggio che indica quanto utile è stata una determinata feature nella costruzione di un certo albero di decisione. Più una feature viene utilizzata per la costruzione degli alberi, più alta sarà la sua importanza relativa. L'importanza è calcolata per un singolo albero di decisione attraverso la stima del tasso di miglioramento che ha portato quell'attributo per un certo punto di split, pesato per il numero di istanze che arrivano a quel determinato nodo. La metrica di performance

può essere la purezza. Successivamente, i vari punteggi vengono pesati attraverso tutti gli alberi di decisione costruiti dal modello.

Di seguito, in figura 3.14, è riportato il valore di importanza per ogni feature.

	XGB_importance
same_srv_rate	0.29
dst_bytes	0.18
src_bytes	0.14
flag	0.11
diff_srv_rate	0.07
logged_in	0.04
count	0.04
service	0.04
dst_host_srv_count	0.04
dst_host_same_srv_rate	0.02
serror_rate	0.02
dst_host_diff_srv_rate	0.02

Figura 3.14: Features importance

Si nota come ai fini della classificazione è importante la quantità di byte scambiata all'interno di una connessione (*src_bytes* e *dst_bytes*), il *flag* (il quale da informazioni specifiche sull'esito e sullo stato di una connessione) e il valore *same_srv_rate*, che indica la percentuale di connessioni verso lo stesso servizio, tra le connessioni individuate dalla feature *count*, che a sua volta indica il numero di tentativi di connessioni verso lo stesso host di destinazione della connessione corrente, negli ultimi 2 secondi. Il protocollo non ha rivestito un ruolo importante nell'analisi principalmente perché quasi tutto il traffico catturato era TCP, quindi avrebbe portato un contributo discriminativo piuttosto esiguo.

Test

Successivamente, il modello è stato valutato sul dataset di test, ottenendo i seguenti risultati.

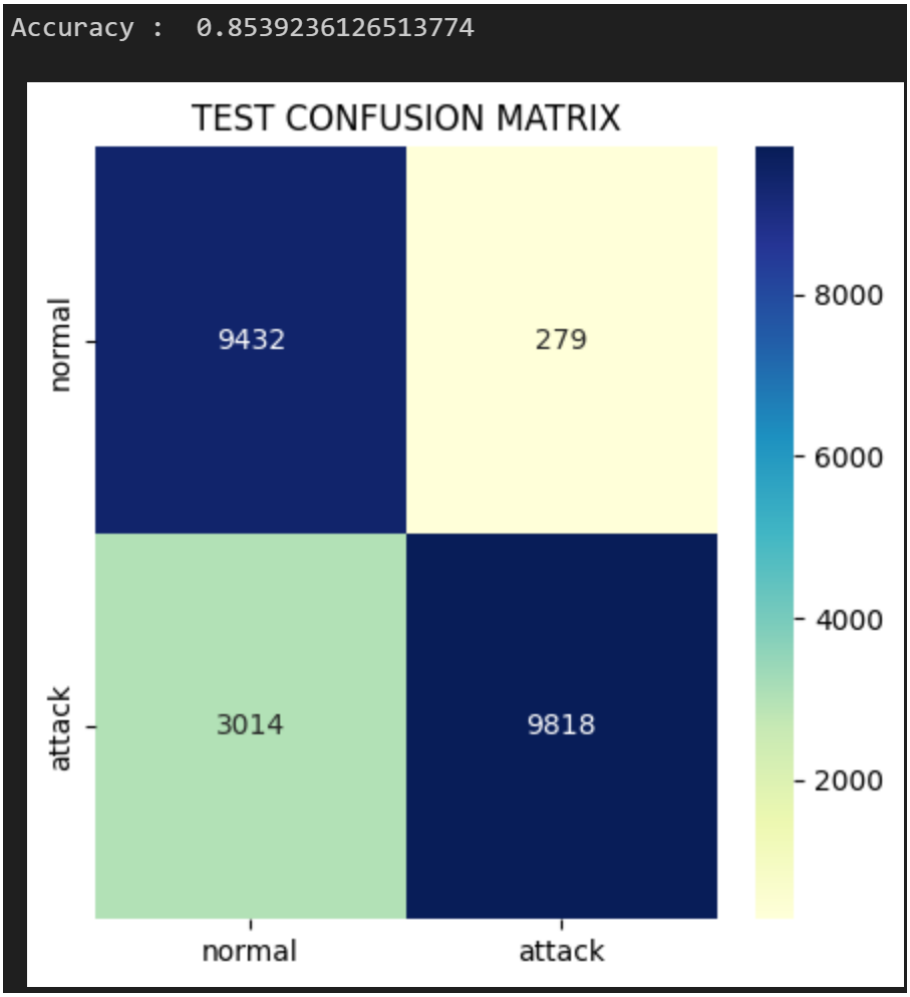


Figura 3.15: Risultati XGBoost sul test set

Poiché i risultati ottenuti non sono stati ritenuti soddisfacenti, a causa del numero elevato di falsi positivi rilevati, si è proceduto all'utilizzo di un altro modello, in particolare del percettrone multilivello.

3.6.2 MLP

Un **percettrone multilivello** (MLP) è un tipo di rete neurale artificiale utilizzato comunemente in task di classificazione. Un MLP è costituito da almeno tre strati di neuroni:

1. **Input layer**: riceve i dati in input. Ogni neurone in questo strato rappresenta una feature del dataset in input;

2. **Hidden layers**: uno o più strati intermedi tra l'input e l'output. Ogni strato nascosto è costituito da neuroni che elaborano l'input ricevuto dallo strato precedente e applicano una funzione di attivazione per introdurre una non-linearità;

3. **Output layer**: fornisce l'output della rete. Il numero di neuroni in questo strato dipende dal tipo di problema/classificazione.

Ognu neurone è connesso a ciascun neurone del successivo strato tramite un **peso** che viene aggiornato durante l'addestramento. Le **funzioni di attivazione** non lineari (come le ReLU, Sigmoide, Tanh) sono applicate nei neuroni per introdurre non linearità, permettendo alla rete di apprendere relazioni complesse nei dati.
Il MLP viene addestrato utilizzando l'**algoritmo di discesa del gradiente**, che calcola l'errore dell'output rispetto al valore target e aggiorna i pesi dei neuroni per minimizzare questa differenza.
Viene introdotta anche la **loss function**, che ha lo scopo di misurare la qualità delle previsioni del modello durante l'addestramento e di guidare l'aggiornamento dei pesi. Per il calcolo del gradiente della loss function rispetto ai pesi viene utilizzato l'algoritmo di **backpropagation**.

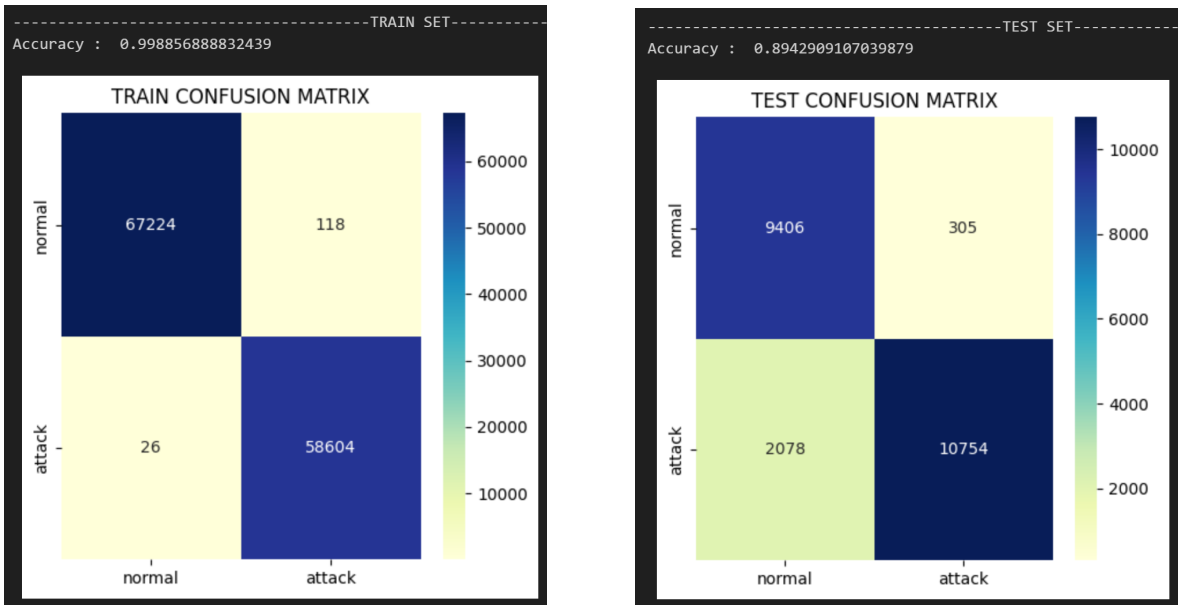
Tuning degli iperparametri

Nel progetto è stato utilizzato un MLP con i parametri riportati di seguito.

```
mlp = MLPClassifier(hidden_layer_sizes=(10, 10),
                    activation='relu',
                    solver='adam',
                    max_iter=300,
                    random_state=1)
```

Test

Sono stati ottenuti i seguenti risultati sul train e sul test set.



(a) Risultati Train Set

(b) Risultati Test Set

Figura 3.16: Risultati MLP

I risultati, per quanto decisamente migliori rispetto al modello precedente, presentano tuttavia un numero di falsi negativi ancora elevato.

3.7 Valutazione risultati e criticità

In questa sezione, si procede ad un'analisi dettagliata dei falsi negativi individuati precedentemente. In particolare, di seguito è riportata la distribuzione delle tipologie di attacco maggiormente mancate.

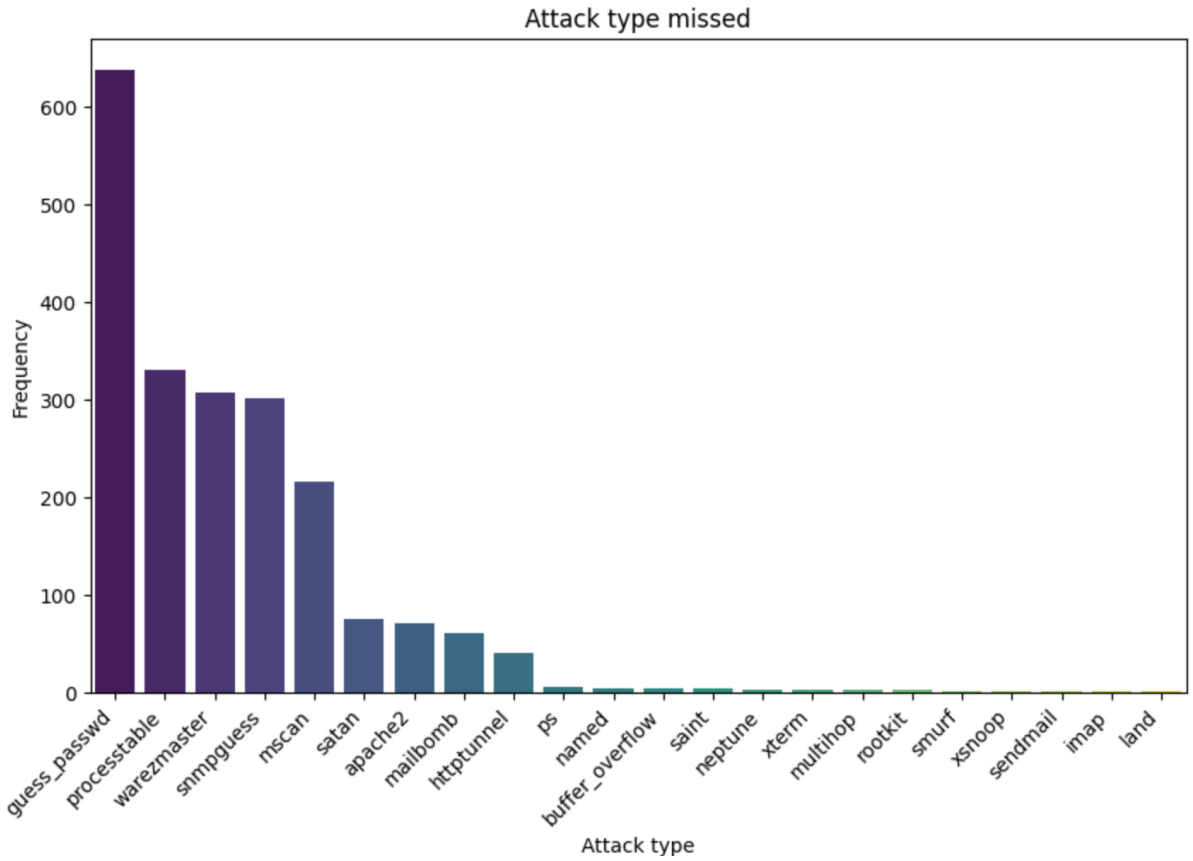


Figura 3.17: Distribuzione attacchi non rilevati

Osserviamo che l'attacco più difficile da rilevare è il *guess_passwd*, un tipo di attacco R2L, in cui un attaccante mira a scoprire la password della vittima con un attacco brute force. Inoltre, si riporta anche la distribuzione dei tipi di attacchi nel dataset originale.

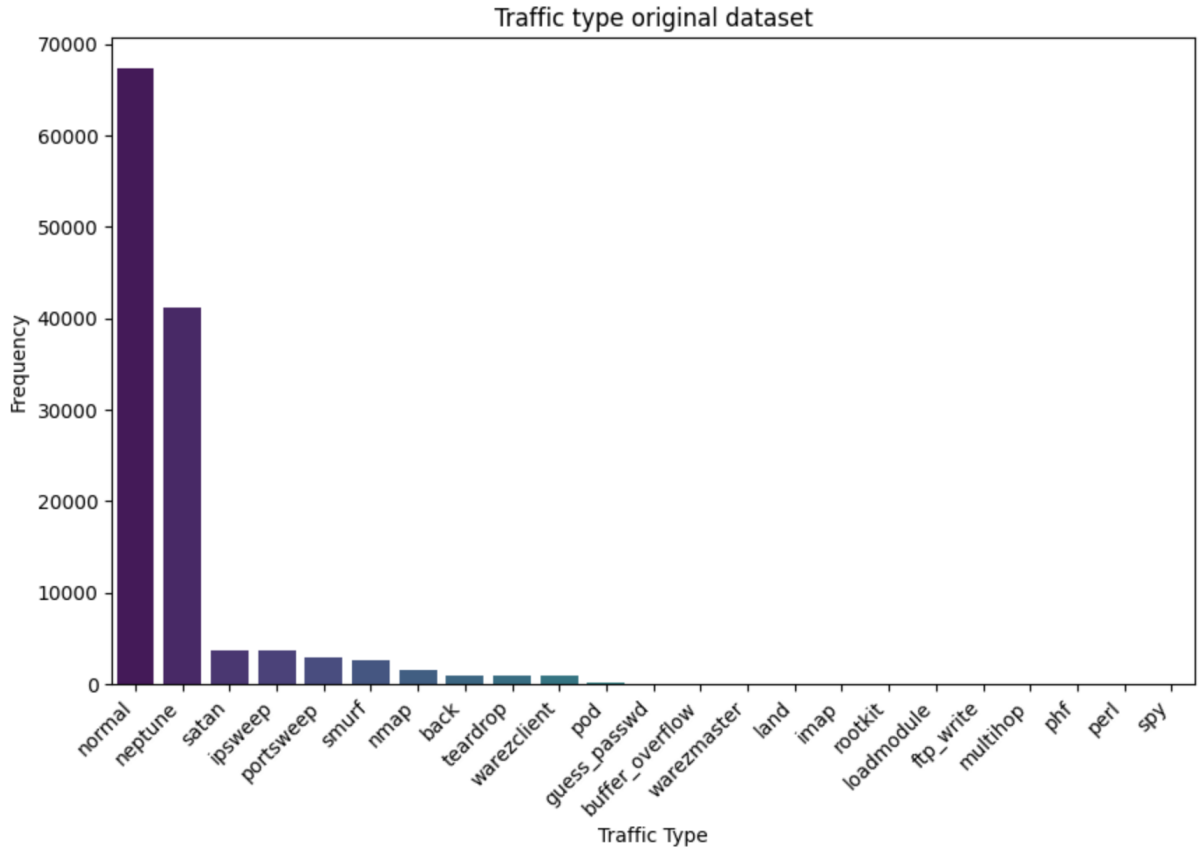


Figura 3.18: Traffic type original dataset

Come si può osservare, nel dataset di train sono presenti pochissime istanze relative a *guess_passwd*, che ne rendono difficile la classificazione. Normalmente, un buon IDS dovrebbe riuscire a individuare anche zero-day attack.

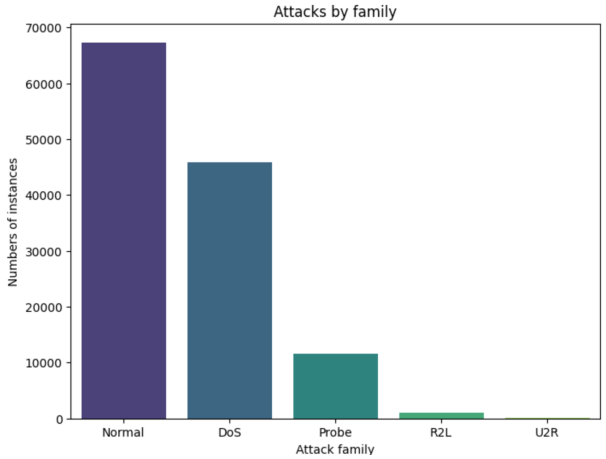


Figura 3.19: Attacks by family

Tuttavia, dalla figura 3.19, si può osservare come il dataset di train sia piuttosto sbilanciato in termini di famiglie di attacco, avendo pochissime istanze per attacchi R2L

e U2R, limitando di fatto il comportamento del classificatore. Si riporta per completezza anche la distribuzione delle famiglie di attacco nel test set. Il test set appartiene ad una distribuzione di probabilità diversa per simulare al meglio uno scenario reale di rete, in quando un IDS anomaly-based, a differenza di quelli rule-based, dovrebbe essere in grado di rilevare anche attacchi non conosciuti. Tuttavia, essendo il dataset sbilanciato nelle famiglie di attacco, gli attacchi Dos e Probe vengono individuati più facilmente rispetto a quelli R2L e U2R proprio per la presenza più numerosa di campioni ad essi associati. Infatti, il classificatore si comporta abbastanza bene per gli attacchi che sono presenti sia nel dataset di train che di test, mentre su alcune tipologie di attacchi non presenti nel train (zero-day) presenta delle difficoltà. In particolare, la lista degli attacchi presenti nel test set ma non nel train set è la seguente: *apache2, httptunnel, mailbomb, mscan, named, processtable, ps, saint, sendmail, snmpgetattack, snmpguess, sqlattack, udpstorm, worm, xlock, xsnoop e xterm.*