UNIVERSITÀ DEGLI STUDI DI BARI ALDO MORO | DIPARTIMENTO DI INFORMATICA

# Cybersecurity_Attacks

ICON project

Realizzato da:

Martina Capone, Donato Boccuzzi

# Indice

## Introduzione

In un momento storico in cui le minacce informatiche sono in costante aumento e sempre più sofisticate, il monitoraggio della rete risulta di estrema importanza al fine di proteggere informazioni di carattere aziendale e privato. Infatti, le reti rivestono un ruolo centrale nelle organizzazioni e garantirne la sicurezza non è solo una questione tecnica, ma una priorità strategica.

Il software sviluppato è un sistema di analisi dei dati di sicurezza informatica che trae ispirazione da un Security Information and Event Management. Un SIEM è un software che aggrega, normalizza e analizza i log e gli eventi di sicurezza provenienti da diverse fonti per rilevare, monitorare e classificare le minacce di sicurezza [1] .

In particolare, il nostro software utilizza delle metriche standard, ragionamento logico e algoritmi di machine learning con l'obbiettivo di monitorare il traffico della rete, verificando se un evento è sicuro o rischioso, identificare e classificare i protocolli utilizzati e valutare il rischio associato ai vari tipi di attacchi.

### Requisiti funzionali

Per poter eseguire il software è necessario che siano installati Python e SWI-Prolog e che siano disponibili le seguenti librerie:

- Pandas: importazione e gestione dei dataset;
- Scikit_learn: utilizzata per l'apprendimento automatico e analisi di dati;
- Pyswip: permette l'utilizzo di Prolog all'interno di un software Python;
- Numpy: utilizzata per l'elaborazione numerica e operazioni su array;
- Tensorflow: framework utilizzato per l'apprendimento delle reti neurali;
- Keras: API utilizzato per la costruzione e l'addestramento di modelli di deep learning;
- Kneed: utilizzata per calcolare il punto elbow nel k-Means;
- Seaborn: utilizzata per visualizzare la matrice di confusione con intensità di colori differenti a seconda dei valori dei dati;
- Matplotlib: utilizzata per la visualizzazione di grafici.

### Installazione e avvio

Non sono richieste procedure complesse per l'installazione e l'esecuzione del software. Qualora le librerie sopramenzionate siano già disponibili, aprire il progetto con l'IDE e avviare il software dal file main.py.

## Dataset

Cybersecurity_Attacks.csv è un dataset messo a disposizione da Incribo.com che offre varie opportunità di utilizzo per l'analisi e la ricerca nel campo della sicurezza informatica. Cybersecurity_Attacks.csv è un insieme di dati ben strutturato che registra eventi di rete dal 1° gennaio 2020 all'11 ottobre 2023, per un totale di 40.000 voci. I record del dataset sono composti da 25 parametri diversi:

- *Timestamp*: momento in cui si è verificata l'attività di rete.
- *Source IP Address*: indirizzo IP del mittente o dell'iniziatore del traffico di rete.
- *Destination IP Address*: indirizzo IP del destinatario o del bersaglio del traffico di rete.
- *Source Port*: numero di porta utilizzato dall'indirizzo IP sorgente.
- *Destination Port*: numero di porta utilizzato dall'indirizzo IP di destinazione.
- *Protocol*: protocollo di comunicazione utilizzato.
- *Packet Length*: dimensione del pacchetto in byte.
- *Packet Type*: tipologia del pacchetto.
- *Traffic Type*: tipo di traffico.
- *Payload Data*: dati trasmessi nel pacchetto.
- *Malware Indicators*: indicatori di attività potenzialmente dannosa o presenza di malware.
- *Anomaly Scores*: punteggio che indica deviazioni dal comportamento atteso.
- *Alerts/Warnings*: notifiche o avvisi generati da sistemi di sicurezza o strumenti di monitoraggio.
- *Attack Type*: tipo di attacco rilevato o sospettato.
- *Attack Signature*: schemi o firme specifiche associate ad attacchi noti.
- *Action Taken*: azioni eseguite in risposta a minacce o anomalie rilevate.
- *Severity Level*: livello di gravità associato a un avviso o evento.
- *User Information*: Informazioni sull'utente coinvolto nell'attività di rete.
- *Device Information*: Informazioni sul dispositivo e sul browser coinvolto nell'attività di rete.
- *Network Segment*: segmento o subnet della rete in cui si è verificata l'attività.
- *Geo-location Data*: informazioni sulla posizione geografica associata agli indirizzi IP.

- *Proxy Information*: informazioni sui server proxy coinvolti nella comunicazione di rete.
- *Firewall Logs*: log generati dai dispositivi firewall che indicano il traffico consentito o bloccato.
- *Log Source*: la fonte o l'origine della voce di log.
- *IDS/IPS Alerts*: avvisi generati dai sistemi di rilevamento delle intrusioni (IDS) o di prevenzione delle intrusioni (IPS) che indicano attività sospette o dannose.

**Preprocessing**

Cybersecurity_Attacks.csv è costituito da dati grezzi a causa della natura eterogenea delle informazioni raccolte, provenienti da vari dispositivi di rete. Dunque, il preprocessing è essenziale per sfruttare il dataset e ottenere un sistema performante. Il software utilizza la seguente pipeline per gestire il preprocessing del dataset.

- Gestione di 'Device Information':
  La feature 'Device Information' contiene uno user agent per ogni evento di rete. Uno user agent è una stringa che un'applicazione client invia a un server web come parte di una richiesta. Questa stringa identifica l'applicazione client, il sistema operativo, la versione del software e altre informazioni pertinenti sul dispositivo e sul software utilizzati per accedere al web [2]. 'Device Information' contiene dunque informazioni estremamente rilevanti per il nostro scopo. Al fine di estrarre il sistema operativo e il browser utilizzato è stata prevista un'analisi della stringa che utilizza delle espressioni regolari per cercare i valori corrispondenti e memorizzare le informazioni relative in due nuove features: 'OS' e 'Browser';

- Creazioni di variabili dummy:
  Vengono manipolate le feature 'Packet Type', 'Protocol', 'Action Taken', 'Traffic Type', 'Log Source', 'OS', 'Browser' e 'Attack Type' al fine di trasformare le variabili categoriali in una forma numerica adatta all'analisi dei dati;

- Normalizzazione:
  Vengono normalizzate le feature 'Source Port', 'Destination Port', 'Packet Length', 'Anomaly Scores' e 'BaseScore' attraverso la tecnica del Min-Max scaling al fine di stabilizzare i dati numerici in un intervallo compreso tra 0 e 1. La normalizzazione assicura che tutte le features contribuiscano in modo equo durante il processo di addestramento del modello.

- Eliminazione dei valori nulli e gestione delle informazioni:
  Vengono manipolate le features 'Alerts/Warnings', 'Malware Indicators', 'Firewall Logs', 'IDS/IPS Alerts', 'Proxy Information' per sostituire l'eventuale e unico valore presente con 1. Qualora il valore non sia presente, viene impostato a 0;

- Eliminazione di features non necessarie:
  Ai fini dell'applicazione d'interesse non sono rilevanti svariate features come 'Payload Data' in quanto è costituito da testi in latino. Questi testi sono genericamente utilizzati come filler e non forniscono alcuna informazione utile o specifica che possa essere utilizzata nelle nostre analisi. Inoltre, 'User Information' e 'Geo-location Data' sono state rimosse in quanto informazioni troppo specifiche. 'Network Segment' e 'Attack Signature' sono state escluse dal dataset poiché rivelano due tipi di pattern senza specificarne la generazione. Non vengono sfruttati 'Source IP Address' e 'Destination IP Address' in quanto, dopo un'attenta analisi, si è constato che sono tutti indirizzi IP diversificati. Inoltre, la feature 'Severity Level' è stata rimossa perché non utile ai nostri fini dato che non è ben specificato il modo in cui è stato calcolato il grado di severità;

- Introduzione della feature 'BaseScore':
  Per ottenere dei modelli performanti viene calcolata una metrica, attraverso la programmazione logica, che valuta la gravità dell'attacco. Per maggiori informazioni si rimanda alla parte successiva.

Nel sistema, il preprocessing viene utilizzato dai modelli di apprendimento (supervisionato e non) e per formare la base di conoscenza (senza sfruttare, però, lo step di normalizzazione e di creazione di variabili dummy).

## Ragionamento logico

Il modulo logico del software utilizza Prolog, un linguaggio di programmazione logica dichiarativa. È basato sulla logica del primo ordine e si distingue per il suo paradigma di programmazione dichiarativo, dove il programmatore specifica le regole e i fatti del problema, mentre il sistema Prolog si occupa di risolvere le interrogazioni in base a queste regole [3]. Nel software, il ragionamento logico viene utilizzato per calcolare il *BaseScore* e per verificare se un evento di rete è sicuro oppure no. Prima di introdurre le specifiche della Knowledge Base è necessario porre enfasi sulle metriche utilizzate.

### Metriche di vulnerabilità

Il Common Vulnerability Scoring System v2.0 è un metodo messo a disposizione dal National Institute of Standards and Technology e utilizzato per fornire una misura qualitativa della gravità di un attacco informatico. In particolare, la metrica *BaseScore* dà come risultato un punteggio numerico compreso tra 0 e 10 che rappresenta la gravità dell'attacco [4]. Il NIST fornisce diverse categorie di gravità in base al punteggio del *BaseScore*:

| | |
|---|---|
| 0.0 | Nessun rischio |
| 0.1 – 3.9 | Gravità bassa |
| 4.0 – 6.9 | Gravità media |
| 7.0 – 8.9 | Gravità alta |
| 9.0 – 10 | Gravità critica |

Il *BaseScore* è il risultato di una serie di operazioni matematiche applicate ai dati di input. Di seguito sono riportate le specifiche:

$$BaseScore = (0.6 * Impact + 0.4 * Expolitability - 1.5) * f(Impact)$$

$$dove$$

- $Impact = 10.41 * (1 - (1 - ConfImpact) * (1 - IntegImpact) * (1 - AvailImpact))$
- $Exploitability = 20 * AccessComplexity * Authentication * AccessVector$
- $f(Impact) = 0\ se\ Impact = 0;\ 1.176\ altrimenti$

La prima parte del calcolo riguarda l'*Impact* che misura l'impatto complessivo sulla riservatezza, integrità e disponibilità delle informazioni:

$$ConfImpact = \begin{cases} 0, & Se\ l'impatto\ sulla\ confidenlità\ è\ nullo \\ 0.275, & Se\ l'impatto\ sulla\ confidenlità\ è\ parziale \\ 0.66, & Se\ l'impatto\ sulla\ confidenlità\ è\ completo \end{cases}$$

$$IntegImpact = \begin{cases} 0, & Se\ l'impatto\ sull'integrità\ è\ nullo \\ 0.275, & Se\ l'impatto\ sull'integrità\ è\ parziale \\ 0.66, & Se\ l'impatto\ sull'integrità\ è\ completo \end{cases}$$

$$AvailImpact = \begin{cases} 0, & Se\ l'impatto\ sulla\ disponibilità\ è\ nullo \\ 0.275, & Se\ l'impatto\ sulla\ disponibilità\ è\ parziale \\ 0.66, & Se\ l'impatto\ sulla\ disponibilità\ è\ completo \end{cases}$$

La seconda parte del calcolo riguarda l'*Exploitability* che misura la facilità con cui viene eseguito l'attacco. Exploitability utilizza informazioni come la complessità di accesso, i privilegi richiesti e il vettore di accesso al sistema:

$$AccessVector = \begin{cases} 0.395, & Se\ l'accesso\ al\ sistema\ avviene\ tramite\ l'accesso\ locale \\ 0.646, & Se\ l'accesso\ al\ sistema\ avviene\ tramite\ la\ rete\ locale \\ 1, & Se\ l'accesso\ al\ sistema\ avviene\ tramite\ la\ rete \end{cases}$$

$$AccessComplexity = \begin{cases} 0.35, & Se\ la\ complessità\ di\ accesso\ alle\ informazioni\ è\ alta \\ 0.61, & Se\ la\ complessità\ di\ accesso\ alle\ informazioni\ è\ media \\ 0.71, & Se\ la\ complessità\ di\ accesso\ alle\ informazioni\ è\ bassa \end{cases}$$

$$Authentication = \begin{cases} 0.704, & Se\ il\ sistema\ non\ richiede\ autenticazioni\ per\ ottenere\ privilegi \\ 0.56, & Se\ il\ sistema\ richiede\ l'autenticazione\ per\ ottenere\ privilegi \\ 0.45, & Se\ Se\ il\ sistema\ richiede\ più\ autenticazioni\ per\ ottenere\ privilegi \end{cases}$$

**Knowledge Base**

Una Knowledge Base (KB) è un insieme di conoscenze organizzate in modo da poter essere utilizzate da un sistema per formulare interrogazioni. Il ragionamento avviene usando la logica matematica per dimostrare la corrispondenza tra i fatti e le regole definite. Nello specifico, i fatti sono verità immutabili, mentre una regola è una dichiarazione che afferma che qualcosa è vero in base ad altre cose che sono vere.

**Fatti**

Il file *aggregations.pl* contiene i fatti per poter calcolare il *BaseScore*. Per poter generare la base di conoscenza, vengono aggregati diversi elementi facenti parte dello stesso evento di rete di Cybersecurity_Attacks.csv al fine di poter valutare le informazioni sotto i diversi punti di vista della metrica CVSSv2.0. Ad ogni aggregazione viene associato uno score, utile per poter calcolare il *BaseScore* (il quale ci darà informazioni affidabili circa la gravità dell'attacco). Nonostante una feature possa apparire più volte nelle diverse aggregazioni, contribuirà in maniera diversa al calcolo dello score a seconda della rilevanza rispetto al fatto stesso. Lo score verrà standardizzato secondo i parametri del CVSSv2.0 prima di dover calcolare il *BaseScore*.

Nella knowledge base sono stati definiti quattro tipi di fatti principali:

1. access_complexity

   Contiene le informazioni relative alla complessità di accesso alle informazioni, lo score viene calcolato a seconda delle tipologie di protocollo, attacco e pacchetto, se nella rete c'è un firewall e se ci sono allerte da parte di sistemi IDS/IPS e sistemi di rilevazione malware.
2. authentication

   Contiene le informazioni relative all'autenticazione nel sistema per ottenere determinati privilegi. Al calcolo dello score contribuiscono, dunque, la presenza di un proxy di rete, la tipologia di attacco e di traffico e il sistema operativo del sistema stesso.
3. confidential_impact

   Contiene le informazioni riguardo la confidenzialità dei dati. Lo score viene calcolato in base al tipo di pacchetto e al tipo di traffico. È determinante la lunghezza del pacchetto in quanto pacchetti di dati più lunghi rappresentano un rischio di confidenzialità maggiore.
4. integrity_impact

   Contiene le informazioni circa l'integrità dei dati e contribuiscono al calcolo dello score la tipologia di pacchetto e soprattutto il protocollo utilizzato a livello di trasporto. Infatti, un pacchetto trasportato tramite UDP sarà molto meno sicuro rispetto ad un protocollo TCP.

Sono stati definiti altri fatti di supporto a quelli principali, utili al calcolo del BaseScore. In particolare, vengono memorizzati tutti i valori standard per *AccessVector, ConfImpact, IntegImpact, AvailImpact, Autenthication* e *AccessComplexity* con le relative etichette messe a disposizione dal CVSSv2.0. Le informazioni riguardanti gli standard CVSSv2.0 vengono memorizzate con la rappresentazione in triple, ossia individuo-proprietà-valore in maniera tale da rappresentare in maniera più omogenea una conoscenza già ben strutturata.

È necessario specificare che, non avendo parametri per poter valutare la disponibilità delle informazioni, si è dato per scontato che *AvailImpact* sia nullo, ossia la disponibilità delle informazioni non è intaccata per tutti gli eventi di rete. Inoltre, dopo una valutazione di Cybersecurity_Attacks.csv, si è dato per scontato che l'accesso al sistema viene eseguito tramite la rete per tutti i network event registrati. Dunque, *AccessVector* sarà sempre pari ad 1.

Il file *evaluations.pl* contiene i fatti per determinare se un evento di rete è sicuro oppure no. In particolare, contiene un solo tipo di fatto, network_event, in cui è memorizzata l'azione compiuta rispetto all'attacco e due indicatori di sicurezza: il *BaseScore* e l'Anomaly Score. Le regole valuteranno queste informazioni per determinare se un evento di rete risulta rischioso.

**Regole**

Le query su una base di conoscenza sono utilizzate per interrogare la KB e ottenere informazioni specifiche. In Prolog, le query sono espresse usando la sintassi della programmazione logica e possono essere impiegate per effettuare ricerche basate sui fatti e sulle regole presenti nella KB.

Per quanto riguarda il calcolo dell'indice di gravità, vengono utilizzate diverse regole per determinare il *BaseScore*:

1. Estrazione dei valori standard CVSSv2.0

```
ac_score(Protocol, Attack_Type, Packet_Type, Firewall, IDS_Alerts, Malware, AC) :-
    access_complexity(P, A, Pt, F, AL, M, VALUE),
    P=Protocol, A=Attack_Type, Pt=Packet_Type, F=Firewall, AL=IDS_Alerts, M=Malware,
    (
        VALUE < 0.47 -> LABEL = low;
        VALUE < 0.67 -> LABEL = medium;
        LABEL = high
    ),
    prop(LABEL, accessComplexity, AC).
```

La regola estrae il valore standard `AC` (*AccessComplexity*) utile per il calcolo del BaseScore. Per poterlo estrapolare vengono effettuate le seguenti operazioni:

a. Il predicato `access_complexity(P, A, Pt, F, AL, M, VALUE)` recupera lo score corrispondente in base agli atomi e lo assegna a `VALUE`;
b. Viene compiuta un'analisi sull'intervallo di `VALUE` e assegnato a `LABEL` la specifica etichetta;
c. Il predicato `prop(LABEL, accessComplexity, AC)` recupera il valore standard in base all'etichetta e lo assegna ad `AC` (*AccessComplexity*).

Allo stesso modo funzionano le regole `au_score`, `conf_score` e `integ_score` e che estraggono rispettivamente `AU` (*Authentication*), `C` (*ConfImpact*), `I` (*IntegImpact*).

2. Calcolo di Impact

```
impact(PT, TT, PL, P, IMPACT) :-
    conf_score(PT, TT, PL, C),
    integ_score(PT, P, I),
    prop(_, availImpact, A),
    IMPACT is 10.41 * (1 - (1 - C) * (1 - I) * (1 - A)).
```

La regola calcola il valore di *Impact* attraverso le seguenti operazioni:

a. La query `conf_score(PT, TT, PL, C)` estrare il valore `C` (*ConfImpact*);
b. La query `integ_score(PT, P, I)` estrae il valore `I` (*IntegImpact*);
c. Il predicato `prop(_, availImpact, A)` recupera il valore standard della proprietà availImpact e lo assegna ad `A` (*AvailImpact*);
d. Il predicato `IMPACT is` **[...]** calcola il valore *Impact* e lo assegna al risultato `IMPACT`.

3. Calcolo di Exploitability

```
exploitability(P, AT, PT, F, A, M, PR, TT, OS, EXPLOIT) :-
    ac_score(P, AT, PT, F, A, M, AC),
    au_score(PR, AT, TT, OS, AU),
    prop(_, accessVector, AV),
    EXPLOIT is 20 * AC * AU * AV.
```

La regola calcola il valore di Exploit (*Exploitability*) in modo estremamente simile alla regola impact:

a. La query `ac_score(P, AT, PT, F, A, M, AC)` estrare il valore `AC` (*AccessComplexity*);
b. La query `au_score(PR, AT, TT, OS, AU)` estrae il valore `AU` (*Authentication*);
c. Il predicato `prop(_, accessVector, AV)` recupera il valore standard della proprietà accessVector e lo assegna ad `AV` (*AccessVector*);
d. Il predicato `EXPLOIT is [...]` calcola il valore *Exploitability* e lo assegna al risultato `EXPLOIT`.

4. Calcolo del BaseScore

```
BaseScore(Packet_Type, Traffic_Type, Packet_Length, Protocol, Attack_Type,
          Firewall, IDS_Alerts, Malware, Proxy, Os, BASESCORE) :-

    impact(PT, TT, PL, P, IMPACT),
    PT=Packet_Type, TT=Traffic_Type, PL=Packet_Length, P=Protocol,

    exploitability(P, AT, PT, F, A, M, PR, TT, OS, EXPLOIT),
    P=Protocol, AT=Attack_Type, PT=Packet_Type, F=Firewall, A=IDS_Alerts,
    M=Malware, PR=Proxy, TT=Traffic_Type, OS=Os,

    (
        IMPACT = 0 -> F_IMPACT = 0;
        F_IMPACT = 1.176
    ),

    BASESCORE is (0.6 * IMPACT + 0.4 * EXPLOIT - 1.5) * F_IMPACT.
```

La regola calcola il BaseScore attraverso le seguenti operazioni:

a. La query `impact(PT, TT, PL, P, IMPACT)` calcola il valore *Impact* e lo assegna ad IMPACT;
b. La query `exploitability(P, AT, PT, F, A, M, PR, TT, OS, EXPLOIT)` calcola il valore di *Exploitability* e lo assegna a `EXPLOIT`;
c. Viene assegnato il valore di `F_IMPACT` mediante un controllo su `IMPACT`;
d. Il predicato `BASESCORE is [...]` calcola il *BaseScore* e lo assegna a `BASESCORE`.

La regola `BaseScore` viene utilizzata nel preprocessing del dataset per introdurre la nuova feature 'BaseScore' di estrema utilità per ottenere dei modelli performanti.

È possibile utilizzare la regola attraverso query con nuovi network event per calcolare il BaseScore, e quindi dedurre la gravità dell'attacco.

Inoltre, è possibile richiamare separatamente la regola `impact` e la regola `exploitability` per scoprire rispettivamente la gravità dell'impatto e la facilità con cui viene eseguito l'attacco.

Per verificare se un evento di rete è sicuro oppure no vengono utilizzate delle regole che valutano tre categorie principali: Action Taken, ossia l'azione intrapresa in caso di attacco, BaseScore, cioè l'indice di gravità, e Anomaly Score, ossia un punteggio che indica deviazioni dal comportamento atteso.

1. Determinazione dei parametri sicuri

```
is_safe(Basescore, AnomalyScore) :-
    network_event('Ignored', Basescore, AnomalyScore),
    Basescore < 3.9,
    AnomalyScore < 50.

is_safe(Basescore, AnomalyScore) :-
    network_event('Logged', Basescore, AnomalyScore),
    Basescore < 6.9,
    AnomalyScore < 60.

is_safe(Basescore, AnomalyScore) :-
    network_event('Blocked', Basescore, AnomalyScore),
    Basescore < 8.9,
    AnomalyScore < 80.
```

Le regole verificano se i parametri `Basescore` e `AnomalyScore`, sono considerati sicuri oppure no in base alle seguenti operazioni:

a. Il predicato `network_event('Ignored', Basescore, AnomalyScore)` recupera gli eventi di rete etichettati come 'Ignored' in base a `Basescore` e `AnomalyScore` contenuti nei fatti. Allo stesso modo funziona per 'Logged' e 'Blocked';
b. Viene compiuto un controllo su `Basescore`;
c. Viene compiuto un controllo su `AnomalyScore`.

Se le condizioni sono soddisfatte, allora i parametri sono considerati sicuri (True), altrimenti no (False).

La regola `is_safe` classifica un evento di rete come rischioso se i parametri di sicurezza Basescore e AnomalyScore superano determinate soglie, anche se l'azione intrapresa è 'Blocked'. Questa scelta riflette l'importanza di considerare tali eventi come potenziali indicatori di vulnerabilità nel sistema, suggerendo che la struttura potrebbe non essere sufficientemente robusta.

2. Determinazione dell'evento sicuro

```
safe_event(Packet_Type, Traffic_Type, Packet_Length, Protocol, Attack_Type,
           Firewall, IDS_Alerts, Malware, Proxy, Os, AnomalyScore) :-

           basescore(Packet_Type, Traffic_Type, Packet_Length, Protocol, Attack_Type,
           Firewall, IDS_Alerts, Malware, Proxy, Os, BASESCORE),
           is_safe(BASESCORE, AnomalyScore).
```

La regola, in base all'evento di rete, stabilisce se l'evento stesso è sicuro oppure no:

a. Determina il BaseScore attraverso la query `basescore`
b. La query `is_safe(Basescore, AnomalyScore)` determina se i parametri sono sicuri oppure no.

Se le condizioni sono soddisfatte, allora l'evento è considerato Safe Event, altrimenti Risky Event.

In generale, la query `safe_event` fornisce un metodo più intuitivo e naturale per interrogare la base di conoscenza verificando se l'evento di rete è sicuro. Calcola automaticamente e verifica i punteggi di sicurezza, semplificando così l'interazione con il sistema.

Segue un esempio di possibili risultati di interrogazione, svolta in ambiente Python e sfruttando Pyswip per interrogare la KB:

```
----------------------------------------------------------------------------------------------------
network_event = ('Data', 'HTTP', 'Long', 'UDP', 'Intrusion', 'Log Data', 'None', 'IoC Detected',
'Proxy', 'Windows', 77.0)

BASESCORE:  7.511301336000001 , IMPACT:  7.843935000000001 , EXPLOITABILITY:  7.952
--- Risky Event ---
----------------------------------------------------------------------------------------------------
network_event = ('Control', 'DNS', 'Short', 'ICMP', 'DDoS', 'Log Data', 'Alert Data', 'None',
'Proxy', 'iPhone OS', 34.0)

BASESCORE:  2.8384524000000004 , IMPACT:  2.86275 , EXPLOITABILITY:  5.49
+++ Safe Event +++
```

## Apprendimento non supervisionato

L'apprendimento non supervisionato riguarda l'uso di algoritmi per trovare pattern e strutture nei dati senza etichette predefinite. È utilizzato per esplorare i dati, ridurre la dimensionalità e creare gruppi omogenei di dati simili.

In questo progetto, l'algoritmo k-Means è stato impiegato per creare due cluster: uno basato su caratteristiche di rete, e l'altro su indicatori di attacco. Questa clusterizzazione ha aiutato a migliorare le prestazioni dei modelli di classificazione, riducendo l'overfitting e facilitando l'analisi dei dati.

Il primo cluster, **Network Features Cluster**, utilizza caratteristiche di rete come porta di origine, porta di destinazione e lunghezza del pacchetto. Questa clusterizzazione ha permesso di identificare pattern comportamentali all'interno dei dati, migliorando la capacità dei modelli di catturare relazioni complesse.

Il secondo cluster, **UserAgent Cluster**, è stato creato in due modi differenti in base al modello di classificazione utilizzato. La forma base prevede esclusivamente le informazioni riguardanti il sistema operativo e il browser; invece, la forma estesa comprende caratteristiche specifiche relative ai tipi di attacco, come tipo di attacco, indicatori di malware, punteggi di anomalia e avvisi di IDS/IPS.

Questa suddivisione presenta due principali vantaggi:

1. Miglioramento della capacità predittiva: con un cluster dedicato agli indicatori di attacco, i modelli di machine learning possono meglio riconoscere e prevedere comportamenti malevoli, migliorando l'accuratezza e l'affidabilità delle previsioni.
2. Riduzione dell'overfitting: creare cluster specifici aiuta a ridurre l'overfitting, poiché i modelli possono generalizzare meglio le informazioni utili dai dati di addestramento senza concentrarsi su rumori o variazioni non significative.

Il metodo k-Means è stato scelto per la clusterizzazione dei dati per la sua efficacia nel raggruppare osservazioni simili in cluster distinti.

La scelta del numero di cluster è del tutto automatizzata. Infatti, per determinare il numero ottimale di cluster da utilizzare, è stata adottata la tecnica della “curva di gomito”. La curva del gomito mostra la variazione dell'inerzia al variare del numero di cluster, dove l'inerzia rappresenta la somma delle distanze quadrate tra

ogni punto dati e il centro del cluster assegnato. L'algoritmo esegue il k-Means iterando da 1 a 12 cluster, calcola l'inerzia, plottando la curva e identificando il "gomito" dove la diminuzione dell'inerzia diventa meno significativa.

I risultati vengono visualizzati tramite un grafico per rendere chiare le decisioni prese riguardo al numero di cluster. Il grafico mostra la varianza intra-cluster al variare di k, evidenziando il punto ottimale identificato dalla KneeLocator.

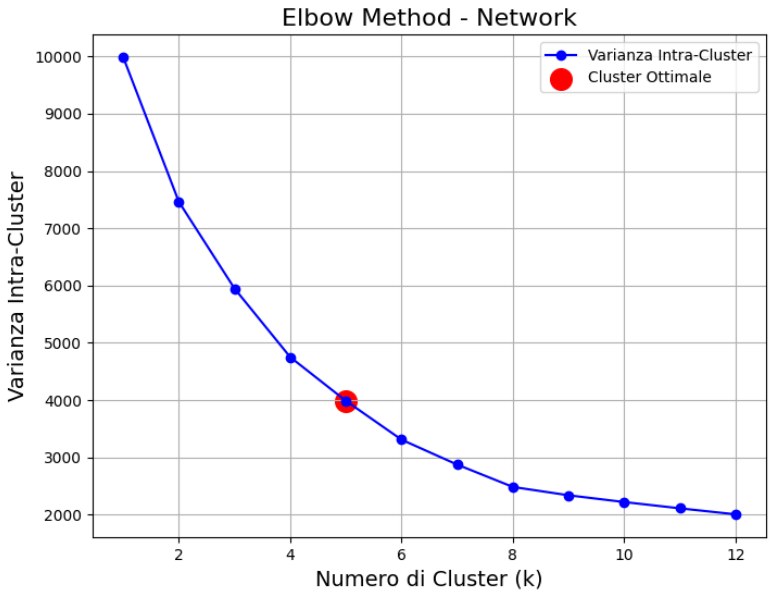


Figura 1: Curva del gomito per le caratteristiche di rete

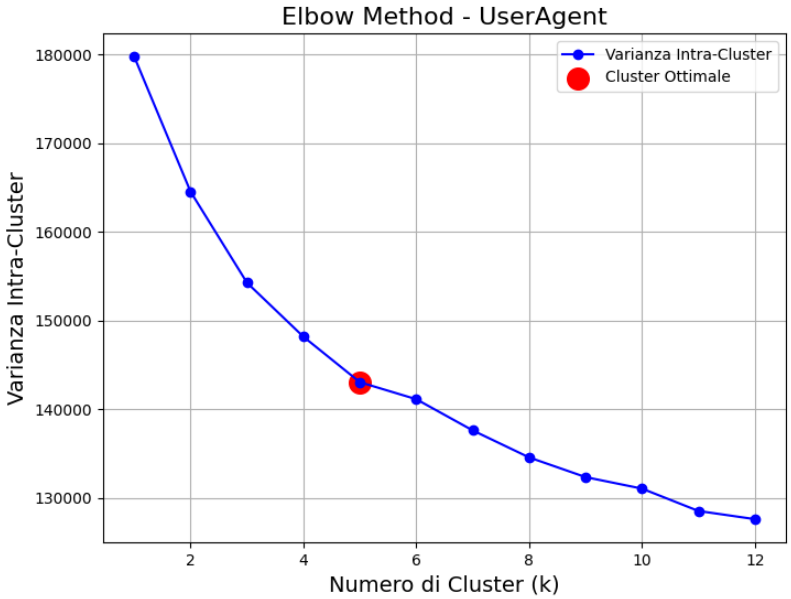


Figura 2: Curva del gomito per le caratteristiche dell'attacco

## Apprendimento supervisionato

L'apprendimento supervisionato è una tecnica di machine learning in cui un modello viene addestrato su un set di dati etichettati. Questo tipo di apprendimento è composto principalmente da due compiti: la regressione e la classificazione. La regressione è utilizzata per prevedere valori continui, mentre la classificazione viene utilizzata per prevedere categorie discrete [5].

### Classificazione

La classificazione è il processo di identificazione della categoria a cui appartiene una nuova osservazione, basandosi su un insieme di dati di addestramento che contengono osservazioni già etichettate.

In questo progetto, sono stati confrontati quattro modelli di classificazione per valutare quale sia il più efficace per il problema in esame.

I modelli considerati sono:

- Decision Tree,
- Random Forest,
- K-Nearest Neighbors (KNN),
- Multi-Layer Perceptron (MLP).

Si è deciso di utilizzare la feature *Protocol* per la classificazione. La scelta è stata motivata dall'importanza del protocollo di rete nell'analisi del traffico e nella rilevazione di possibili attacchi informatici.

L'etichetta *Protocol* è stata preprocessata utilizzando prima LabelEncoder e in seguito One-Hot-Encoding.

La Label-Encoding è una tipologia di codifica delle etichette che permette di convertire ciascuna categoria della colonna in un numero univoco. Tuttavia, l'uso dei numeri introduce una relazione implicita tra le categorie. Ad esempio, non esiste alcuna relazione gerarchica tra i vari tipi di protocolli, ma usando i numeri, si potrebbe pensare che il protocollo “ICMP”, ad esempio codificato come 0, abbia una rilevanza minore rispetto al protocollo “UDP”, codificato come 2. Questo potrebbe portare l'algoritmo a fraintendere che i dati hanno un qualche tipo di gerarchia, come 0 < 1 < 2 e potrebbe dare 2 volte più peso a "UDP" nel calcolo rispetto al protocollo “ICMP”.

Questo problema di ordinamento viene risolto mediante un altro approccio alternativo chiamato One-Hot Encoding. In questa rappresentazione, ogni categoria è trasformata in un array in cui l'indice corrispondente alla classe è impostato a 1, mentre tutti gli altri sono impostati a 0. Questo passaggio è essenziale perché la maggior parte degli algoritmi di machine learning richiede input numerici.

Sono state sperimentate diverse configurazioni di input per valutare l'effetto di queste trasformazioni sulle prestazioni dei modelli. La prima clusterizzazione, che incorporava caratteristiche di rete, è stata integrata in tutti i classificatori. Questa scelta è stata motivata dalla necessità di aggregare dati con simili caratteristiche di rete, migliorando così la capacità del modello di identificare pattern ricorrenti nel traffico di rete.

La seconda clusterizzazione è stata applicata solo nei modelli Decision tree, Random Forest e KNN per ottimizzare le prestazioni dei modelli e mitigare il rischio di overfitting. Le feature originali utilizzate per questa clusterizzazione comprendono 'OS_Linux', 'OS_Mac OS', 'OS_Windows', 'OS_iPad OS', 'OS_iPhone OS', 'Browser_Firefox', 'Browser_MSIE', 'Browser_Opera' e 'Browser_Safari'.

Per i modelli Decision Tree, KNN e Random Forest è stata utilizzata la K-fold Validation con 10 fold, per migliorare l'affidabilità delle valutazioni dei modelli. Questo approccio suddivide il dataset in dieci parti, utilizzando ciascuna parte una volta come set di test e le altre nove come set di addestramento.

Inoltre è utilizzata GridSearchCV, una classe di scikit-learn che implementa una ricerca esaustiva sugli iperparametri di un modello. Funziona combinando una griglia di valori di iperparametri specificati dall'utente e testandoli tutti usando la tecnica di cross-validation. È utilizzato per identificare la combinazione ottimale di iperparametri che massimizza le prestazioni del modello su un set di dati.

**Decision Tree**

Il Decision Tree è un modello basato su alberi decisionali e utilizzato per la classificazione. Gli iperparametri analizzati sono:

- max_depth: indica la massima profondità dell'albero.
- criterion: indica la funzione utilizzata per misurare la qualità di una divisione.

- min_samples_split: corrisponde al numero minimo di campioni richiesti per dividere un nodo interno.
- min_samples_leaf: corrisponde al numero minimo di campioni richiesti per essere in un nodo foglia.

I parametri ottimali trovati per il Decision Tree Classifier sono i seguenti:

- Criterion: "entropy"
- Max Depth: 30
- Min Samples Leaf: 1
- Min Samples Split: 10

Questi parametri sono stati determinati utilizzando GridSearchCV con una cross-validation su 10 fold per massimizzare l'accuratezza del modello.

La matrice di confusione rappresenta le prestazioni del modello Decision Tree nella classificazione delle tre classi di traffico di rete: TCP, UDP e ICMP. Le righe della matrice indicano le classi reali, mentre le colonne indicano le classi previste.

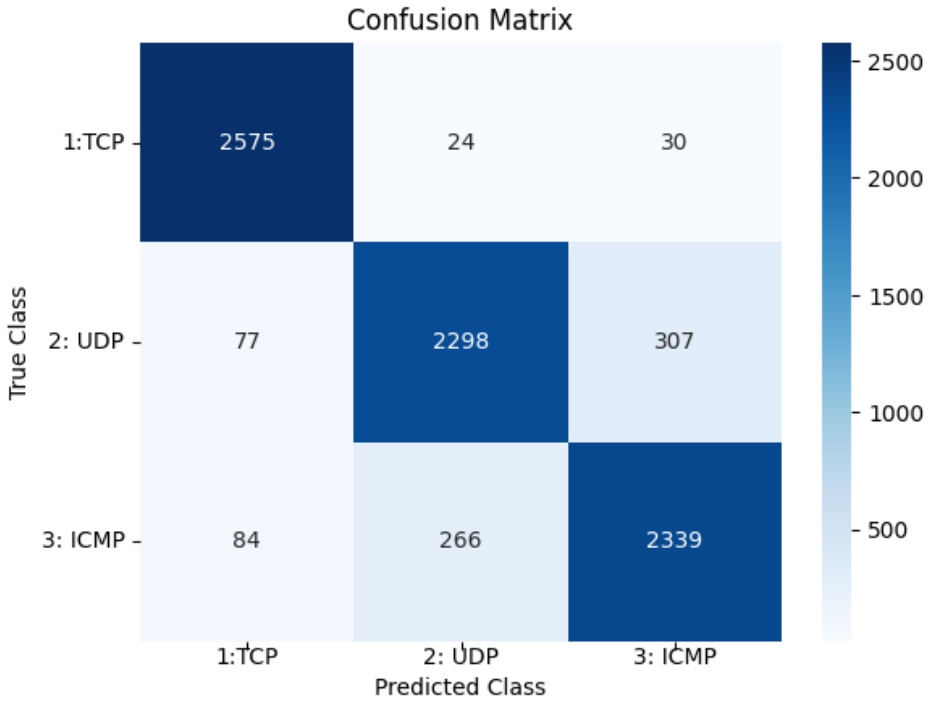


Figura 1: Matrice di confusione per il classificatore Decision Tree

In particolare, a classe TCP è stata classificata correttamente nella maggior parte dei casi, con un totale di 2575 istanze correttamente identificate. La classe UDP mostra una maggiore difficoltà nel modello, con 307 istanze erroneamente classificate come ICMP. La classe ICMP ha una buona accuratezza complessiva, con 2339 istanze correttamente identificate, sebbene ci siano ancora alcune confusioni con la classe UDP.

I risultati ottenuti utilizzando il Decision Tree Classifier con i migliori iperparametri trovati sono:

- Accuracy: 0.899625
- Precision (micro): 0.9132089836315188
- Recall (micro): 0.899625
- F1_micro score: 0.9063660978527801
- F1_macro score: 0.9066785906938485

**Random Forest**

Il Random Forest Classifier è un ensemble di alberi decisionali che migliora la robustezza e la generalizzazione rispetto ai singoli alberi. Gli iperparametri sperimentati sono:

- n_estimators: indica il numero di alberi utilizzati.
- criterion: indica la funzione per misurare la qualità di una divisione.
- max_depth: corrisponde alla massima profondità di ogni albero nell'ensemble.
- min_samples_split: indica il numero minimo di campioni richiesti per dividere un nodo interno.
- min_samples_leaf: indica il numero minimo di campioni richiesti per essere in un nodo foglia.

I migliori iperparametri trovati per il Random Forest Classifier sono:

- Criterion: entropy
- Max Depth: 20
- Min Samples Leaf: 1
- Min Samples Split: 5
- Number of Estimators: 100

La seguente matrice di confusione della Random Forest fornisce una panoramica chiara sulle prestazioni del modello nel classificare le tre classi:

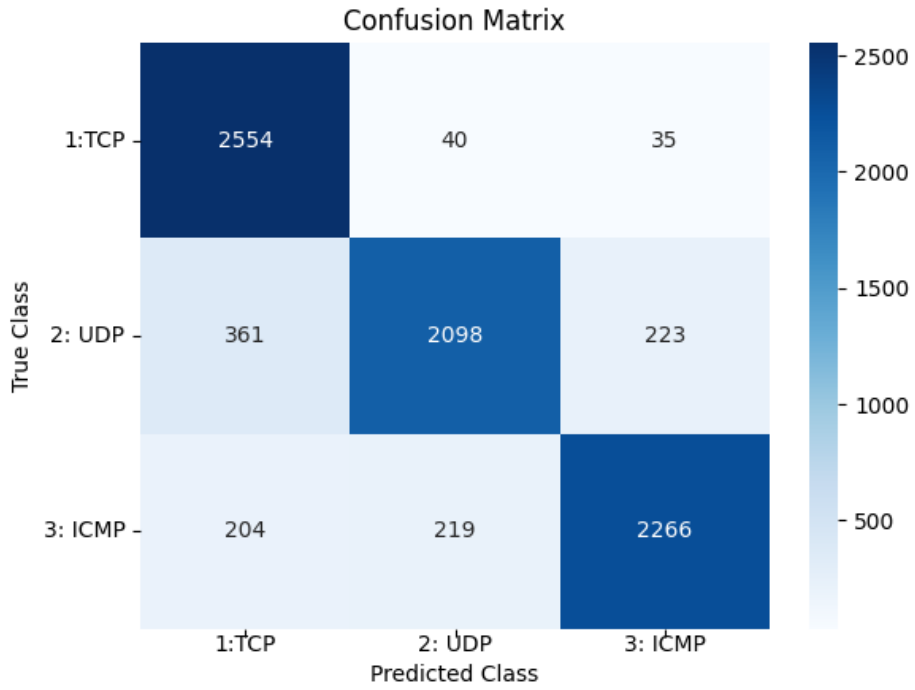


Figura 2: Matrice di confusione per il classificatore Random Forest

Il Random Forest presentano i seguenti risultati sul test set:

- Accuracy: 0.843625
- Precision (micro): 0.9182312925170067
- Recall (micro): 0.843625
- F1_micro score: 0.8793485342019544
- F1_macro score: 0.8792436647429612

**KNN**

Il K-Nearest Neighbors è un algoritmo di classificazione basato su istanze che predice il valore di una nuova osservazione sulla base della sua vicinanza con i punti di dati nel set di addestramento.

Il KNN, con lo stesso preprocessing effettuato per gli altri modelli di classificazioni presenta una accuratezza del 35%, questo significa che il modello non riesce a classificare correttamente la maggior parte degli esempi. Pertanto si è sperimentata

l’estensione della clusterizzazione UserAgent Cluster aggiungendo le seguenti features: 'Attack Type_Intrusion', 'Attack Type_Malware', 'Malware Indicators', 'Anomaly Scores', 'Alerts/Warnings', 'IDS/IPS Alerts', 'Proxy Information', 'Firewall Logs', 'Packet Type_Data', 'Action Taken_Ignored', 'Action Taken_Logged', 'Traffic Type_FTP', 'Traffic Type_HTTP', 'Log Source_Server'.

Questa estensione ha potato a un notevole miglioramento delle metriche:

- Accuracy,0.6795
- Precision (micro),0.71847739888977
- Recall (micro),0.6795
- F1_micro score,0.6984453295644353
- F1_macro score,0.6981840504113898

Il fatto che il KNN possa funzionare meglio con un numero limitato di features può dipendere da diversi fattori:

1. Maledizione della dimensionalità: Il KNN calcola le distanze tra i punti nel dataset per determinare i vicini più prossimi. Con un elevato numero di dimensioni, le distanze tra i punti diventano meno significative. Questo perché, con più dimensioni, la distanza euclidea tende a diventare più omogenea, cioè i punti possono essere equidistanti l'uno dall'altro. Di conseguenza, la capacità del KNN di identificare correttamente i vicini più prossimi può diminuire, influenzando negativamente le prestazioni del modello.
2. Overfitting: Con un numero elevato di features, il modello KNN potrebbe memorizzare i dati anziché generalizzare. Pertanto trovare un numero ridotto di features rilevanti può aiutare a migliorare la capacità del modello di generalizzare su nuovi dati.

La matrice di confusione per il modello KNN mostra buone prestazioni complessive, con la maggior parte delle istanze correttamente classificate, ma c'è spazio per migliorare la distinzione tra le classi TCP e UDP.

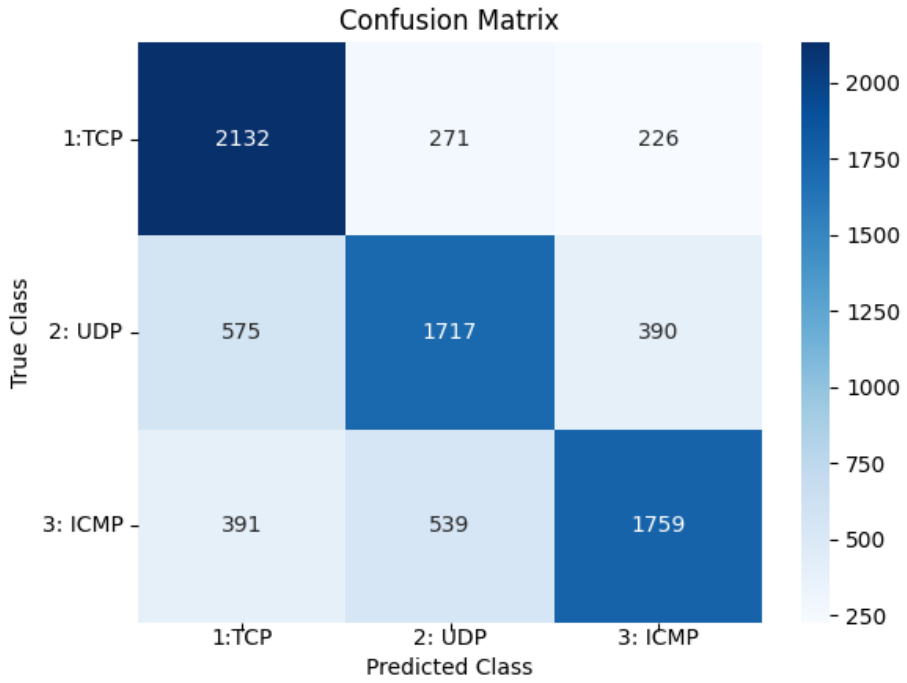


Figura 3: Matrice di confusione per il classificatore KNN

**MLP**

Il modello MLP è una tipologia di rete neurale artificiale che è particolarmente adatta per compiti di classificazione.

Le features in input del modello utilizzano esclusivamente la clusterizzazione basata su Network Features Cluster, poiché questo approccio evita l'overfitting e garantisce una buona accuracy. L'uso della clusterizzazione basata su UserAgent Cluster, invece, comporta una perdita di accuracy. Le etichette sono preprocessate allo stesso modo degli altri modelli.

Il dataset viene suddiviso in :

- 70% di dati per l'addestramento,
- 20% di dati per il test,
- 10% di dati per la validazione. Questo è utilizzato per monitorare le prestazioni del modello durante l'addestramento e prevenire l'overfitting.

Il modello è costruito come una sequenza di strati ed è caratterizzato da:

| Nr | Tipo di layer | Dimensione input | Dimensione output | Funzione di attivazione | Dropout |
|---|---|---|---|---|---|
| 1 | Dense | 25 | 128 | ReLU | |
| 2 | Dropout | | | | 50% |
| 3 | Dense | 128 | 64 | ReLU | |
| 4 | Dropout | | | | 50% |
| 5 | Dense | 64 | 32 | ReLU | |
| 6 | Dense | 32 | 3 | Softmax | |

- I Dense Layers, costituiscono le componenti principali della rete. Sono configurati con dimensioni decrescenti per ridurre gradualmente le caratteristiche e trovare le combinazioni più rilevanti.
- La funzione di attivazione ReLU (Rectified Linear Unit) è usata per introdurre non-linearità, consentendo al modello di apprendere pattern complessi.
- I dropout con una probabilità del 50% aiutano a prevenire l'overfitting, spegnendo casualmente alcune unità durante l'addestramento.
- Lo strato finale ha tre neuroni, uno per ogni classe, con una funzione di attivazione softmax, che produce una distribuzione di probabilità per la classificazione multiclasse.

Per la compilazione del modello vengono utilizzati:

- l'ottimizzatore Adam che combina i vantaggi di due metodi di ottimizzazione: AdaGrad e RMSProp. È scelto per la sua efficienza e capacità di adattarsi dinamicamente al tasso di apprendimento. L'ottimizzatore Adam è settato con un tasso di apprendimento di 0.0001, scelto per garantire che il modello impari gradualmente e con stabilità.
- la funzione di perdita categorical_crossentropy, adatta per la classificazione multiclasse.
- l'accuracy è usata come metrica di valutazione.

Inoltre, sono utilizzate 2 callbacks che permettono di intervenire durante l'addestramento di un modello per monitorare e modificare il comportamento del processo di addestramento. Le callbacks vengono eseguite automaticamente al termine di ogni epoca. È stato inserito un numero massimo di epoche pari a 5000, impostato per garantire che il modello abbia sufficiente tempo per convergere, ma l'early stopping generalmente interromperà l'addestramento prima. L'Early Stopping è un callback che interrompe l'addestramento se l'accuratezza di validazione non migliora per 15 epoche consecutive, prevenendo l'overfitting e risparmiando tempo di calcolo.

La seconda callback utilizzata è ModelCheckpoint che salva i pesi del modello ogni volta che si ottiene un miglioramento nell'accuratezza di validazione. Questo garantisce che si conservino solo i migliori pesi, prevenendo la perdita di informazioni cruciali in caso di interruzione dell'addestramento.

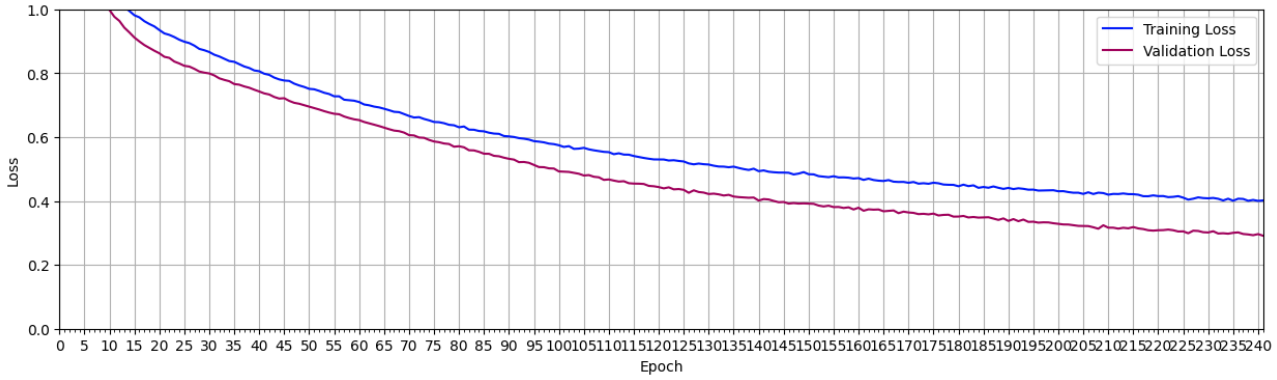


Figura 4: Curva di apprendimento del MLP rispetto la loss

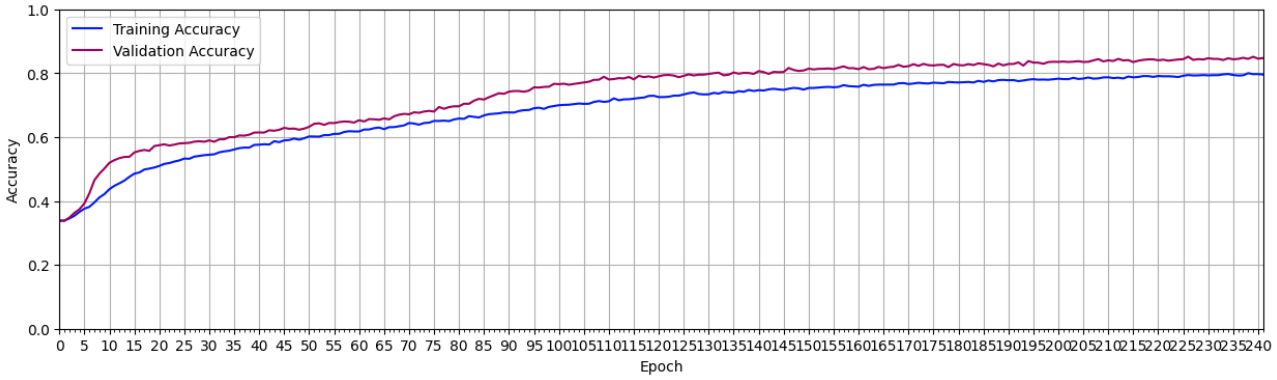


Figura 5: Curva di apprendimento del MLP rispetto l'accuracy

**Regressione**

La regressione è una tecnica statistica utilizzata per studiare la relazione tra una variabile dipendente (o target) e una o più variabili indipendenti (o features). Il suo obiettivo principale è di modellare e comprendere come le variazioni nelle variabili indipendenti influenzino la variabile dipendente, permettendo di fare previsioni o inferenze basate sui dati disponibili [6].

Nel progetto, il *BaseScore* rappresenta un indicatore chiave originariamente calcolato mediante regole definite in Prolog. Abbiamo scelto di utilizzare Prolog per la sua capacità di gestire relazioni logiche e categoriche. Tuttavia, Prolog ha una limitazione: se manca un valore, non può dedurre un risultato. Per superare questo ostacolo e aumentare la robustezza del sistema, è stato implementato un secondo approccio al calcolo, basato sulla regressione. Questo metodo è in grado di gestire i valori mancanti e può rivelare relazioni lineari tra le variabili. Quindi, è stato addestrato un modello di regressione per predire il *BaseScore* utilizzando un insieme specifico di features. Un approccio combinato ci permette di sfruttare i punti di forza di entrambi i metodi e colmare una possibile mancanza di informazioni, seppur approssimando il valore, mediante la regressione.

Le features selezionate per la regressione sono:

- 'Attack Type_Intrusion', 'Attack Type_Malware', 'Malware Indicators', 'Anomaly Scores', 'Alerts/Warnings', 'IDS/IPS Alerts', 'Proxy Information', 'Firewall Logs', 'Packet Type_Data', 'Action Taken_Ignored', 'Action Taken_Logged', 'Traffic Type_FTP', 'Traffic Type_HTTP', 'Log Source_Server', 'OS_Linux', 'OS_Mac OS', 'OS_Windows', 'OS_iPad OS', 'OS_iPhone OS', 'Browser_Firefox', 'Browser_MSIE', 'Browser_Opera', 'Browser_Safari'.
- Il raggruppamento " Network Features Cluster ", ottenuto attraverso clustering k-Means sulle features: 'Source Port', 'Destination Port', 'Packet Length'.

Il modello è basato su una rete neurale MLP, scelta per la sua capacità di gestire dati complessi e ottimizzare la predizione di *BaseScore*:

- È strutturato con strati densamente connessi che riducono gradualmente le caratteristiche per identificare le combinazioni più rilevanti.

- Comincia con un layer Dense di 100 unità attivate dalla funzione ReLU, seguito da dropout al 20% per prevenire l'overfitting.
- Questo pattern continua con strati successivi di dimensioni decrescenti (75 e 50 unità), ognuno con dropout per migliorare la generalizzazione.
- Lo strato finale consiste in un singolo neurone per la regressione del valore "BaseScore".

| Nr | Tipo di layer | Dimensione input | Dimensione output | Funzione di attivazione | Dropout |
|---|---|---|---|---|---|
| 1 | Dense | 26 | 100 | ReLU | |
| 2 | Dropout | | | | 20% |
| 3 | Dense | 100 | 75 | ReLU | |
| 4 | Dropout | | | | 20% |
| 5 | Dense | 75 | 50 | ReLU | |
| 6 | Dropout | | | | 20% |
| 7 | Dense | 50 | 1 | | |

Il modello è compilato con l'ottimizzatore Adam, che adatta dinamicamente il tasso di apprendimento durante l'addestramento per garantire una convergenza stabile e graduale (tasso di apprendimento = 0.0001). La funzione di perdita utilizzata è mean_squared_error, adatta per la regressione.

Durante l'addestramento, sono implementati due callbacks:

- **Early Stopping**: Interrompe l'addestramento se l'accuratezza di validazione non migliora per 15 epoche consecutive, prevenendo l'overfitting e risparmiando tempo di calcolo.
- **ModelCheckpoint**: Salva i pesi del modello ogni volta che si ottiene un miglioramento nell'accuratezza di validazione, garantendo di conservare solo i migliori pesi e prevenendo la perdita di informazioni critiche in caso di interruzione dell'addestramento.

**Valutazione del Modello**

Dopo l'addestramento, il modello viene valutato utilizzando un set di test separato. Le metriche di valutazione sono:

- Il MSE (Mean Squared Error) rappresenta la media dei quadrati degli errori tra le predizioni del modello e i valori osservati nel set di test. Un valore più basso indica una maggiore precisione delle previsioni.
- Il MAE (Mean Absolute Error) è la media degli errori assoluti tra le predizioni del modello e i valori osservati nel set di test. È una misura della deviazione media delle previsioni dal valore reale.
- Il RMSE (Root Mean Squared Error) è la radice quadrata del MSE e fornisce una stima della deviazione standard degli errori delle previsioni. È particolarmente utile perché penalizza maggiormente gli errori più grandi.
- MSLE (Mean Squared Log Error) misura il logaritmo dei quadrati degli errori tra il logaritmo naturale delle previsioni e il logaritmo naturale dei valori reali nel set di test. È utile quando le previsioni tendono ad essere sottostimate rispetto ai valori reali.

I risultati della valutazione del modello di regressione sono i seguenti:

| **MSE** | **MAE** | **RMSE** | **MSLE** |
|---|---|---|---|
| 0.0023 | 0.0298 | 0.0480 | 0.0010 |

Questi risultati indicano che il modello di regressione MLP ha prestazioni generalmente buone nel predire il "BaseScore" basandosi sulle features selezionate. L'MSE e l'RMSE sono relativamente bassi, suggerendo che il modello è efficace nel ridurre gli errori nelle sue predizioni. Il MAE mostra una deviazione media delle predizioni di circa 0.0298, indicando una buona precisione. Infine, l'MSLE suggerisce che il modello gestisce bene la scala logaritmica delle differenze tra previsioni e valori reali.

Questi risultati confermano l'efficacia del modello MLP nella sua applicazione per la regressione del "BaseScore" nel contesto specifico del progetto.

Di seguito è riportato il grafico delle curve di training loss e test loss per visualizzare come il modello ha imparato nel corso delle epoche di addestramento.

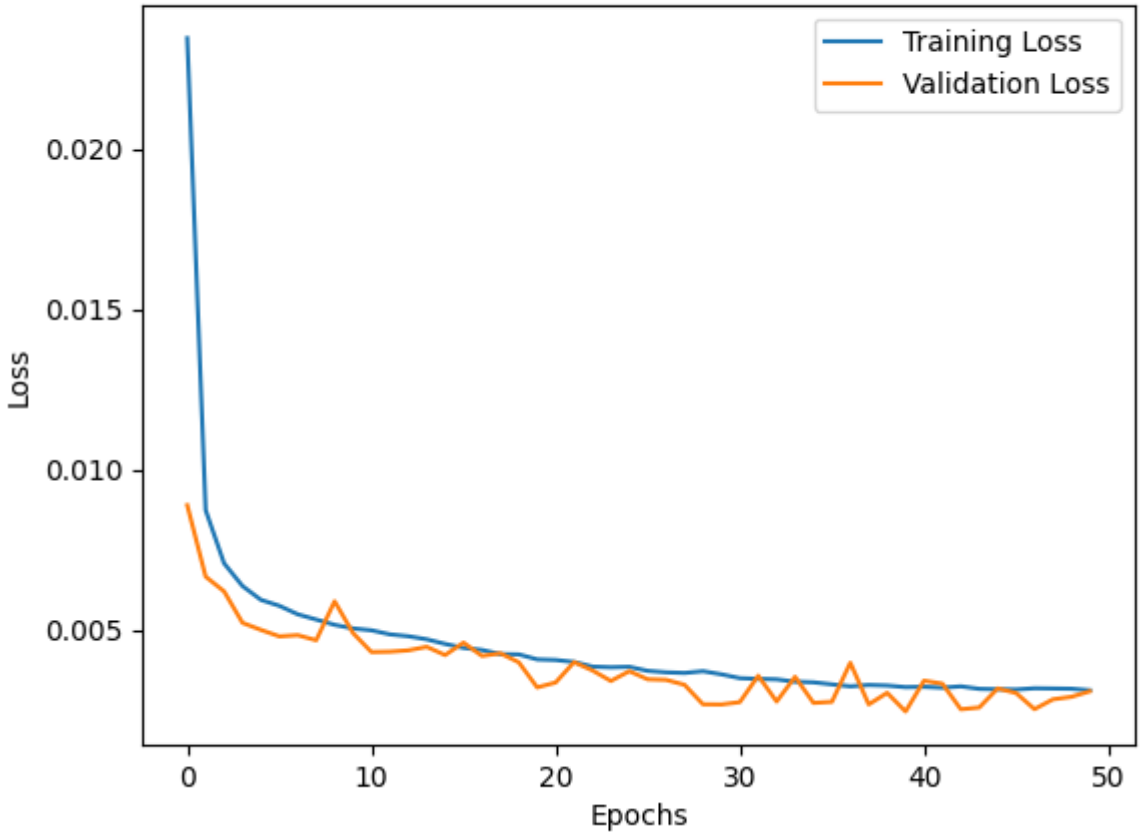


Figura 1: Curve di training loss e test loss

## Sviluppi futuri

Per quanto concerne gli sviluppi futuri, si considerano diverse direzioni per estendere e migliorare il progetto. Una di queste potrebbe essere l'integrazione di un sistema di acquisizione dati in tempo reale. Questo permetterebbe di monitorare e analizzare il traffico di rete in tempo reale, migliorando la capacità di rilevare e rispondere agli attacchi.

Inoltre, si potrebbero utilizzare tecniche di analisi predittiva per identificare pattern di attacco prima che si verifichino, permettendo una prevenzione proattiva.

Si potrebbe anche implementare un sistema di risposta automatica agli attacchi che non solo invia notifiche, ma prende anche azioni immediate per mitigare l'attacco, come bloccare IP sospetti o limitare il traffico.

Infine, si potrebbero considerare altri modelli di machine learning e tecniche di analisi dei dati per migliorare la capacità di classificare i tipi di protocollo e fare la regressione sul "BaseScore".

## Riferimenti Bibliografici


[1] IBM, “What is Security Information and Event Management (SIEM)?”, https://www.ibm.com/topics/siem.

[2] J. Roßmann, T. Gummer, and L. Kaczmirek, “Working with user agent strings” *J Stat Softw*, vol. 92, 2020, doi: 10.18637/jss.v092.c01.

[3] D. Poole and A. Mackworth, *Artificial Intelligence: Foundations of Computational Agents*, 3rd ed. Cambridge, UK: Cambridge University Press, [Ch.15].

[4] NIST, “Common Vulnerability Scoring System Calculator version 2.0”, https://nvd.nist.gov/vuln-metrics/cvss/v2-calculator.

[5] IBM, “What is supervised learning?”, https://www.ibm.com/topics/supervised-learning.

[6] GeeksforGeeks, “Regression in machine learning”, https://www.geeksforgeeks.org/regression-in-machine-learning.