

**Vulnerability\_Attack**

Documentazione sul caso di studio

Ingegneria della Conoscenza AA 2023-2024

Realizzato da:

Martina Capone, 758280, [m.capone16@studenti.uniba.it](mailto:m.capone16@studenti.uniba.it)

Donato Boccuzzi, 758273, [d.boccuzzi8@studenti.uniba.it](mailto:d.boccuzzi8@studenti.uniba.it)

Repository:

<Link da mettere>

**Indice**

[Introduzione 3](#_Toc169638536)

[Requisiti funzionali 3](#_Toc169638537)

[Installazione e avvio 3](#_Toc169638538)

[Dataset 4](#_Toc169638539)

[Preprocessing 5](#_Toc169638540)

[Ragionamento logico 7](#_Toc169638541)

[Metriche di vulnerabilità 7](#_Toc169638542)

[Knowledge Base 8](#_Toc169638543)

[Fatti 9](#_Toc169638544)

[Regole 10](#_Toc169638545)

[Apprendimento supervisionato 15](#_Toc169638546)

[Classificazione 15](#_Toc169638547)

[Decision Tree 16](#_Toc169638548)

[Random Forest 18](#_Toc169638549)

[KNN 20](#_Toc169638550)

[MLP 22](#_Toc169638551)

[Regressione 25](#_Toc169638552)

[Sviluppi futuri 28](#_Toc169638553)

[Riferimenti Bibliografici 29](#_Toc169638554)

# **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 calcolate con il ragionamento logico e algoritmi di machine learning con l’obbiettivo di monitorare il traffico della rete, identificare e classificare eventuali incidenti di sicurezza e valutare il rischio associato ai vari tipi di attacchi.

## **Requisiti funzionali**

Il software è realizzato in Python e per poterlo eseguire è necessario che siano installate 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 kMeans;
* Imblearn: **MARTI NON SO CHE FA** (controlla anche tensorflow);
* Seaborn: **MARTI NON SO CHE FA**;
* 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 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 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 è costituito da dati grezzi a causa della natura eterogenea delle informazioni raccolte, provenienti da vari dispositivi di monitoraggio delle reti. 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 ' Timestamp ':

'Timestamp' è codificato in un'unica stringa che comprende data e ora degli eventi di rete. Al fine di poter utilizzare l’informazione, il preprocessor separa 'Timestamp' in 'Day', 'Month', 'Year', 'Minute', 'Hour'. Questa scelta facilita la gestione temporale dei dati, consentendo un'analisi più accurata delle variazioni nel tempo e semplificando l'identificazione di tendenze.

* 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 'Hour', 'Minute', 'Year', 'Month', 'Day', '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 non contiene realmente informazioni, oscurate da Incribo.com per tutelare la privacy degli utenti. 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.

* 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.

# **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]. 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:

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

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:

## **Knowledge Base**

Una Knowledge Base (KB) è un insieme di conoscenze organizzate in modo da poter essere utilizzate da un programma o da un sistema per rispondere a domande e 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**

Per poter generare la base di conoscenza, vengono aggregati diversi elementi facenti parte dello stesso evento di rete di Cybersecurity\_Attacks al fine di poter valutare le informazioni sotto i diversi punti di vista della metrica CVSSv2.0. Ad ogni fatto viene associato uno score, registrato nella Knowledge Base, 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.

1. 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.

1. 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.

1. 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 CVSSv.20 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, 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.

### **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. Nello specifico, 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:

* 1. Il predicato **access\_complexity(P, A, Pt, F, AL, M, VALUE)** recupera lo score corrispondente in base agli atomi e lo assegna a VALUE;
  2. Viene compiuta un’analisi sull’intervallo di VALUE e assegnato a LABEL la specifica etichetta;
  3. Il predicato **prop(LABEL, accessComplexity, AC)** recupera il valore standard in base all’etichetta e lo assegna ad AC.

Allo stesso modo funzionano le regole **au\_score**, **conf\_score** e **integ\_score**che estraggono rispettivamente AU (*Authentication*), C (*ConfImpact*), I (*IntegImpact*).

1. 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:

1. La query **conf\_score(PT, TT, PL, C)** estrare il valore C (*ConfImpact*);
2. La query **integ\_score(PT, P, I)**estrae il valore I (*IntegImpact*);
3. Il predicato **prop(\_, availImpact, A),** recupera il valore standard della proprietà availImpact e lo assegna ad A (*AvailImpact*);
4. Il predicato **IMPACT is […]** calcola il valore *Impact* e lo assegna al risultato IMPACT.
5. 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:

1. La query **ac\_score(P, AT, PT, F, A, M, AC)** estrare il valore AC (*AccessComplexity*);
2. La query **au\_score(PR, AT, TT, OS, AU)**estrae il valore AU (*Authentication*);
3. Il predicato **prop(\_, accessVector, AV)** recupera il valore standard della proprietà accessVector e lo assegna ad AV (*AccessVector*);
4. Il predicato **EXPLOIT is […]** calcola il valore *Exploitability* e lo assegna al risultato EXPLOIT.
5. 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:

1. La query **impact(PT, TT, PL, P, IMPACT)** calcola il valore *Impact* e lo assegna ad IMPACT;
2. La query **exploitability(P, AT, PT, F, A, M, PR, TT, OS, EXPLOIT)** calcola il valore di *Exploitability* e lo assegna a EXPLOIT;
3. Viene assegnato il valore di **F\_IMPACT** mediante un controllo su IMPACT;
4. 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 non ancora registrati in Cybersecurity\_Attacks 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.

**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 temporali e 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 **Temporal Features Cluster** utilizza caratteristiche temporali e di rete come giorno, mese, anno, ora, porta di origine, porta di destinazione e lunghezza del pacchetto. Questa clusterizzazione ha permesso di identificare pattern temporali e comportamentali all'interno dei dati, migliorando la capacità dei modelli di catturare relazioni complesse.

Il secondo cluster **Attack Profile Cluster** è stato creato utilizzando caratteristiche specifiche relative ai tipi di attacco, come tipo di attacco, indicatori di malware, punteggi di anomalia e avvisi di IDS/IPS. La segmentazione basata su queste caratteristiche ha consentito di separare e identificare differenti tipologie di attività malevole.

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.

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 10 cluster, calcola l'inerzia, plottando la curva e identificando il "gomito" dove la diminuzione dell'inerzia diventa meno significativa.

Immagine che contiene testo, linea, Diagramma, diagramma

Descrizione generata automaticamenteI 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.

Immagine che contiene testo, linea, Diagramma, diagramma

Descrizione generata automaticamenteFigura 1: Curva del gomito per le feature temporali

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.

## **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) e 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 il LabelEncoder per trasformare le etichette categoriali in valori numerici. 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 temporali e di rete, è stata integrata in tutti i classificatori. Questa scelta è stata motivata dalla necessità di aggregare dati con simili caratteristiche temporali e di rete, migliorando così la capacità del modello di identificare pattern ricorrenti nel traffico di rete.

La seconda clusterizzazione è stata applicata solo in alcune istanze per ottimizzare le prestazioni dei modelli e mitigare il rischio di overfitting. Le feature originali utilizzate per questa clusterizzazione comprendevano dettagli specifici su attacchi informatici e indicatori di malware, quali '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'.

L’accuracy dei quattro classificatori prima della clusterizzazione risulta:

|  |  |  |  |
| --- | --- | --- | --- |
| Decision tree | Random Forest | KNN | MLP |
| 0.89 | 0.84 | 0.38 | 0.85 |

I risultati dell’acuracy dopo la clusterizzazione delle features sono:

|  |  |  |  |
| --- | --- | --- | --- |
| Decision tree | Random Forest | KNN | MLP |
| 0.74 | 0.74 | 0.70 | 0.64 |

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.

### **Decision Tree**

Il Decision Tree è un modello basato su alberi decisionali e utilizzato per la classificazione. È stato configurato con l’iperparametro *random\_state* posto uguale a 42, che premette di determinare la riproducibilità dei risultati durante l'addestramento del modello.

Con l’inserimento della seconda clusterizzazione delle feature, il modello mostra un calo di prestazioni in termini di accuratezza. Tuttavia, la clusterizzazione ha aiutato a mitigare l'overfitting, con un bilanciamento più equilibrato tra l'accuratezza di training e quella di validazione.Immagine che contiene testo, diagramma, linea, Diagramma

Descrizione generata automaticamente

Figura 1: Curva di apprendimento del Decision Tree prima della clusterizzazione

Immagine che contiene testo, diagramma, linea, Diagramma

Descrizione generata automaticamente

Figura 2: Curva di apprendimento del Decision Tree dopo la clusterizzazione

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.

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamenteIn particolare, a classe TCP è stata classificata correttamente nella maggior parte dei casi, con un totale di 2333 istanze correttamente identificate. La classe UDP mostra una maggiore difficoltà nel modello, con 844 istanze erroneamente classificate come TCP. La classe ICMP ha una buona accuratezza complessiva, con 1956 istanze correttamente identificate, sebbene ci siano ancora alcune confusioni con le altre classi.

### **Random Forest**

Il Random Forest Classifier è un ensemble di alberi decisionali che migliora la robustezza e la generalizzazione rispetto ai singoli alberi. È stato configurato con i seguenti iperparametri:

* *n\_estimators* = 100: Specifica il numero di alberi utilizzati per migliorare la precisione predittiva.
* *random\_state* = 42

Anche per il modello Random Forest, i risultati sono simili a quelli ottenuti con il Decision Tree. La clusterizzazione delle feature non ha portato a un aumento dell'accuratezza complessiva del modello. Tuttavia, è stata osservata una riduzione dell'overfitting, con una convergenza più stretta tra l'accuratezza di training e quella di test.Immagine che contiene testo, diagramma, linea, Diagramma

Descrizione generata automaticamente

Figura 3: Curva di apprendimento del Random Forest prima della clusterizzazione

Immagine che contiene testo, diagramma, linea, Diagramma

Descrizione generata automaticamente

Figura 4: Curva di apprendimento del Random Forest dopo la clusterizzazione.

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamenteLa seguente matrice di confusione della Random Forest fornisce una panoramica chiara sulle prestazioni del modello nel classificare le tre classi di traffico di rete.

Il modello Random Forest ha mostrato un'ottima capacità di classificare correttamente il traffico TCP, con solo poche istanze erroneamente classificate come UDP o ICMP. Per il traffico UDP, il modello ha una significativa confusione con il traffico TCP, indicando che alcune caratteristiche del traffico UDP potrebbero essere simili a quelle del traffico TCP. Il modello ha anche classificato correttamente la maggior parte del traffico ICMP, sebbene ci sia una certa confusione con il traffico UDP.

### **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.

Diversamente dai modelli precedenti, il modello KNN ha mostrato un miglioramento significativo dell'accuratezza dopo la clusterizzazione delle feature. Questo suggerisce che la clusterizzazione ha aiutato a creare gruppi di feature più rilevanti per il modello KNN, migliorando la sua capacità di classificare correttamente i dati.

Immagine che contiene testo, diagramma, linea, Diagramma

Descrizione generata automaticamente

Figura 5: Curva di apprendimento del KNN prima della clusterizzazione

Immagine che contiene testo, diagramma, mappa, Diagramma

Descrizione generata automaticamente

Figura 6: Curva di apprendimento del KNN dopo la clusterizzazione.

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.

Immagine che contiene testo, schermata, numero, Carattere

Descrizione generata automaticamente

### **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 solamente la clusterizzazione basata sui fati tempolai e di rete perché addestrando il modello senza la clusterizzazione basata sulle tipologie di attacco non presenta overfittin è possiede una buona accuracy. Invece applicando la seconda clusterizzazione si nota solo una perdita dell’accuracy. Le etichette sono preprocessate come per gli 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: Immagine che contiene testo, schermata, numero, Carattere

Descrizione generata automaticamente

* 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.

Immagine che contiene linea, Diagramma, testo, diagramma

Descrizione generata automaticamenteLa 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 7: Curva di apprendimento del MLP rispetto la loss

Immagine che contiene linea, Diagramma, diagramma, testo

Descrizione generata automaticamente

Figura 8: 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.

Nel nostro progetto, il "basescore" rappresenta un indicatore chiave originariamente calcolato mediante regole definite in Prolog. Tuttavia, abbiamo implementato un secondo approccio basato sulla regressione per addestrare un modello capace di predire il "basescore" utilizzando un insieme specifico di features.

Le features selezionate per la regressione includono sia indicatori di attività di rete che raggruppamenti temporali:

* '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'.
* Un raggruppamento chiamato "Temporal Features Cluster", ottenuto attraverso clustering k-means sulle features temporali: 'Day', 'Month', 'Year', 'Minute', 'Hour', '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 del "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".

Immagine che contiene testo, schermata, numero, Carattere

Descrizione generata automaticamente

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.

|  |  |  |  |
| --- | --- | --- | --- |
| **MSE** | **MAE** | **RMSE** | **MSLE** |
| 0.0084 | 0.0706 | 0.0918 | 0.0040 |

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

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 0.07, 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 di training loss e test loss per visualizzare come il modello ha imparato nel corso delle epoche di addestramento.

Immagine che contiene testo, diagramma, schermata, Diagramma

Descrizione generata automaticamente

# **Sviluppi futuri**

Fatevi il vostro progetto e non cacate con gli sviluppi futuri.

# **Riferimenti Bibliografici**

[1] IBM, “What is Security Information and Event Management (SIEM)?,” IBM.

[2] J. Roßmann, T. Gummer, and L. Kaczmirek, “Working with user agent strings in stata: The parseuas command,” *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”, NIST.