Relazione Progetto di Machine Learning: Realizzazione di un IDS

Jhoannis Caccamo

2024-2025

1 Introduzione

Un Intrusion Detection System (IDS) è un dispositivo o software progettato per monitorare reti e sistemi informatici al fine di identificare attività dannose e violazioni della sicurezza. Nel panorama digitale contemporaneo, le minacce informatiche rappresentano una realtà sempre più pressante e sofisticata: con l'evoluzione tecnologica, infatti, gli aggressori sviluppano tecniche di attacco sempre più elaborate e difficili da individuare.

In questo scenario emergono gli IDS basati su machine learning, che sfruttano le capacità di apprendimento automatico per offrire un livello di sorveglianza più efficace e adattivo rispetto ai sistemi tradizionali. Questi sistemi sono in grado di analizzare pattern complessi e di adattarsi a nuove tipologie di minacce, migliorando costantemente le proprie prestazioni.

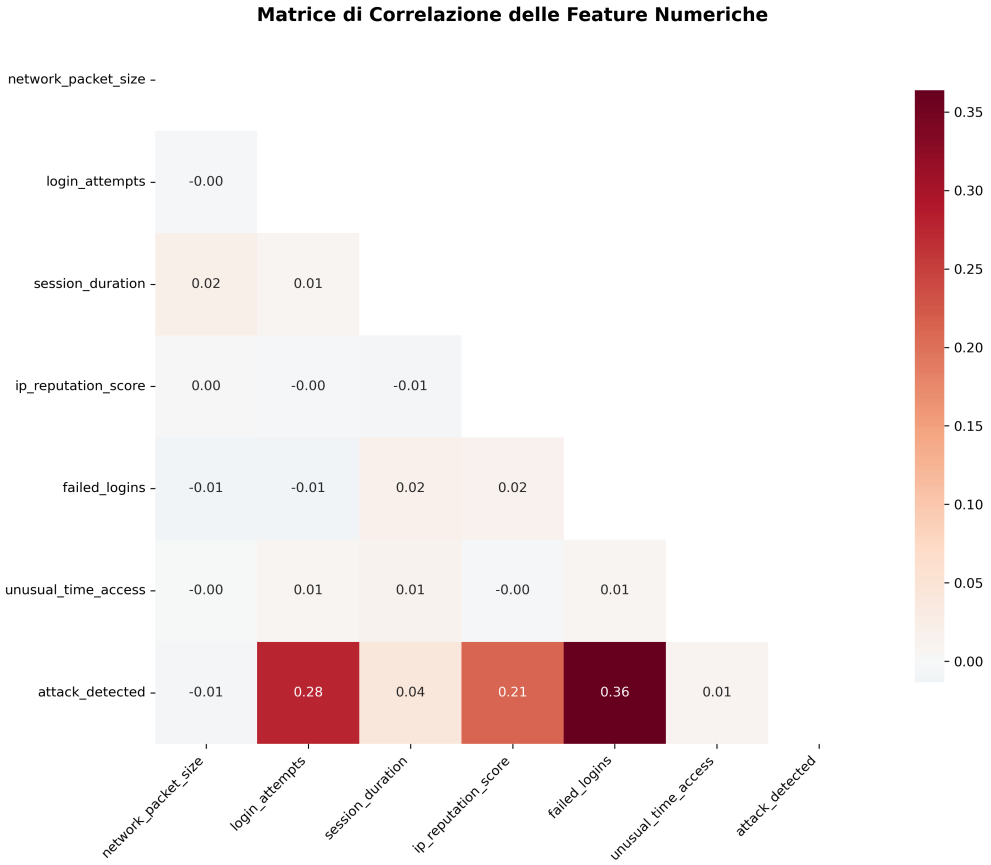
Il problema centrale che questa tecnologia si propone di risolvere consiste nell'analizzare un insieme di caratteristiche comportamentali per determinare se un utente o un processo presenta un comportamento malevolo, distinguendo così le attività legittime da quelle potenzialmente pericolose per la sicurezza del sistema.

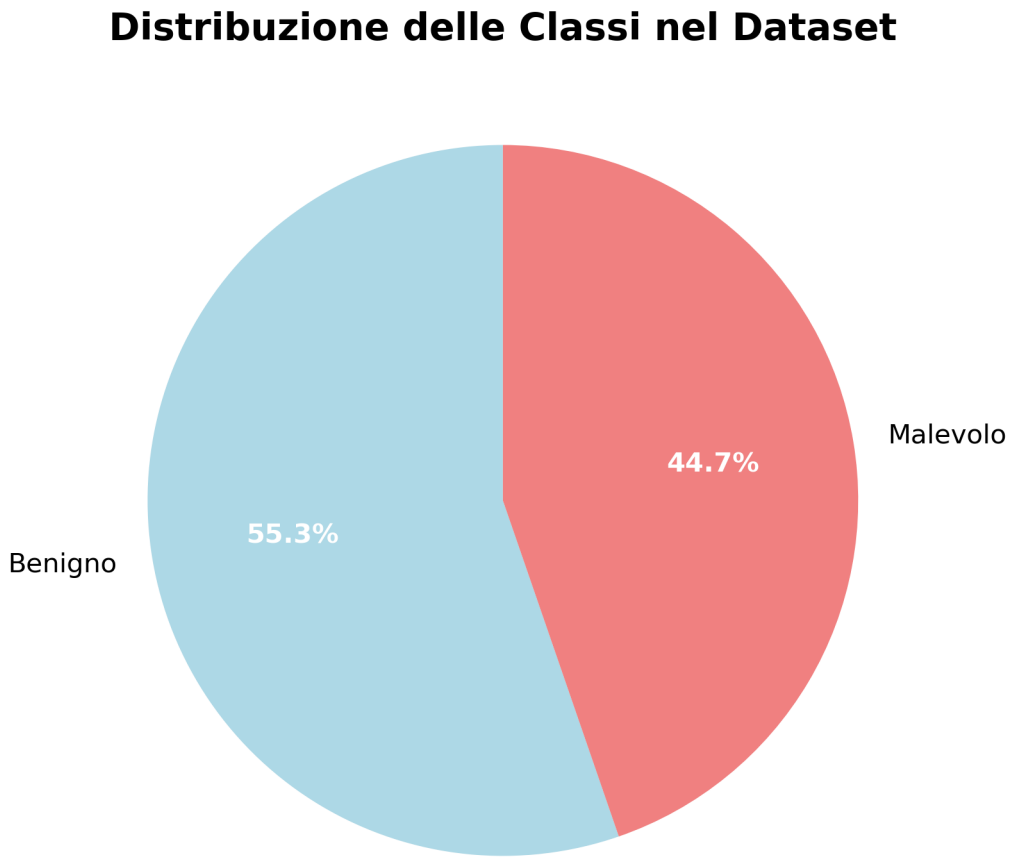
2 Dataset

Il dataset utilizzato in questo progetto, presente nel file cybersecurity_intrusion_data.csv, è stato scaricato da Kaggle, già etichettato. Il dataset presenta le seguenti colonne:

- **session_id**: Identificatore univoco per ciascuna sessione di rete, utilizzato per distinguere e tracciare le singole connessioni o interazioni utente.
- **network_packet_size**: Dimensione dei pacchetti di rete trasmessi durante la sessione, espressa in byte. Valori anomali possono indicare tentativi di attacco come buffer overflow o data exfiltration.

- **protocol_type**: Tipologia di protocollo di comunicazione utilizzato (es. TCP, UDP, HTTP, HTTPS). Diversi protocolli possono essere associati a differenti pattern di comportamento malevolo.
- **login_attempts**: Numero totale di tentativi di accesso effettuati durante la sessione. Un numero elevato può indicare attacchi di tipo brute force o dictionary attack.
- **session_duration**: Durata complessiva della sessione espressa in secondi o minuti. Sessioni eccessivamente lunghe o brevi possono essere indicatori di attività sospette.
- **encryption_used**: Variabile binaria che indica se durante la sessione è stata utilizzata crittografia (1) o meno (0). L'assenza di crittografia può aumentare la vulnerabilità.
- **ip_reputation_score**: Punteggio di reputazione dell'indirizzo IP sorgente, basato su database di threat intelligence. Valori bassi indicano IP associati ad attività malevole note.
- **failed_logins**: Numero specifico di tentativi di login falliti durante la sessione. Correlato con login_attempts ma focalizzato esclusivamente sui fallimenti autenticativi.
- **browser_type**: Tipologia di browser o user agent utilizzato per la connessione. Alcuni malware utilizzano user agent specifici o inusuali.
- **unusual_time_access**: Variabile binaria che indica se l'accesso è avvenuto in orari considerati inusuali (1) o normali (0), basandosi su pattern comportamentali storici.
- **attack_detected**: Variabile target del dataset, indica se è stato rilevato un attacco (1) o se l'attività è considerata benigna (0). Questa è la variabile dipendente che il modello deve predire.


Matrice di Correlazione delle Feature Numeriche
network_packet_size
login_attempts
session_duration
ip_reputation_score
failed_logins
unusual_time_access
attack_detected
-0.00
0.02 0.01
0.00 -0.00 -0.01
-0.01 -0.01 0.02 0.02
-0.00 0.01 0.01 -0.00 0.01
-0.01 0.28 0.04 0.21 0.36 0.01
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00



3 Metodi

Nel contesto dell'intrusion detection, alcune feature comunemente presenti nei dataset di rete, come il tipo di protocollo utilizzato o il tipo di browser, risultano essere irrilevanti o addirittura controproducenti per l'efficacia del sistema di rilevamento. Di seguito vengono analizzate le motivazioni di tale irrilevanza.

3.1 Tipo di Protocollo

I protocolli di comunicazione (HTTP, HTTPS, FTP, SSH, etc.) sono utilizzati indistintamente sia dal traffico normale che da quello malevolo. Un attaccante può sfruttare qualsiasi protocollo disponibile per condurre i propri attacchi, rendendo la scelta del protocollo più dipendente dal servizio target che dall'intento malevolo dell'azione.

3.1.1 Bassa Correlazione con Minacce

Il protocollo utilizzato presenta una correlazione molto debole con la natura malevola del traffico:

- HTTP può trasportare sia richieste legittime che attacchi web (SQL injection, XSS)
- SSH può essere utilizzato per amministrazione normale o per attacchi brute force
- Il protocollo stesso non fornisce indicazioni significative su comportamenti anomali

3.2 Tipo di Browser

3.2.1 Diversità Normale

Gli utenti legittimi utilizzano una varietà di browser web (Chrome, Firefox, Safari, Edge) e la distribuzione di tali browser varia geograficamente e temporalmente. Non esiste una correlazione significativa tra l'utilizzo di un browser specifico e comportamenti malevoli.

3.2.2 Facilità di Manipolazione

Le informazioni relative al browser sono facilmente falsificabili:

- Gli User-Agent possono essere modificati arbitrariamente
- Gli attaccanti spesso mascherano la propria identità utilizzando User-Agent di browser comuni
- I bot malevoli frequentemente imitano browser legittimi per evitare la detection

3.3 Problematiche per l'Intrusion Detection

L'inclusione di feature irrilevanti introduce rumore nel processo di apprendimento del modello. Mentre una feature come "Browser = Chrome 91.0" fornisce informazioni poco significative, una feature come "Frequenza richieste = 1000/sec" rappresenta un segnale molto più rilevante per la detection di anomalie.

3.3.1 Overfitting

L'utilizzo di feature irrilevanti può portare il modello a:

- Associare erroneamente browser o protocolli specifici a minacce
- Ridurre la capacità di generalizzazione su nuovi dati
- Creare bias verso configurazioni specifiche dell'ambiente di training

3.3.2 Concept Drift

Le feature relative a browser e protocolli sono soggette a rapidi cambiamenti:

- I browser si aggiornano continuamente con nuove versioni
- Nuove versioni potrebbero essere classificate erroneamente come anomale
- Il modello diventa rapidamente obsoleto richiedendo frequenti riaggiornamenti

3.4 Modello proposto

La scelta di utilizzare un Multi Layer Perceptron è motivata da diverse considerazioni:

- **Capacità di approssimazione universale**: Gli MLP possono approssimare qualsiasi funzione continua, rendendoli adatti per catturare pattern complessi nei dati di intrusion detection
- **Gestione della non-linearità**: I dati di traffico di rete presentano relazioni non-lineari complesse che gli MLP possono modellare efficacemente
- **Robustezza al rumore**: Gli MLP mostrano buona tolleranza al rumore presente nei dati di rete reali
- **Efficienza computazionale**: Rispetto ad architetture più complesse, gli MLP offrono un buon compromesso tra prestazioni e costi computazionali

4 Metodi

Questa sezione presenta l'approccio metodologico adottato per la risoluzione del problema di intrusion detection. La soluzione proposta si basa sull'utilizzo di un Multi Layer Perceptron (MLP), implementato attraverso una classe personalizzata `CyberSecurityMLP` che integra preprocessing, training, valutazione e interpretabilità del modello.

4.1 Panoramica dell'Approccio

L'architettura complessiva del sistema proposto è illustrata nella Figura 1. Il sistema è strutturato in modo modulare, con componenti distinti per il preprocessing dei dati, la classificazione tramite MLP e la valutazione delle performance.

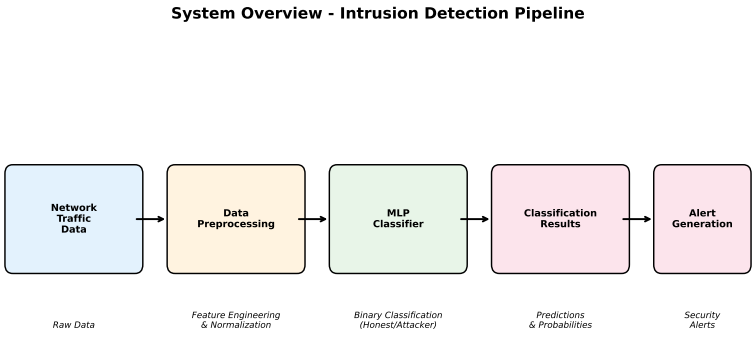


Figura 1: Panoramica generale del sistema di intrusion detection proposto

Il flusso di elaborazione prevede l'acquisizione di dati di traffico di rete, la loro normalizzazione e preprocessing, la classificazione binaria (Honest/Attacker) tramite MLP e la valutazione delle performance attraverso multiple metriche.

4.2 Architettura del Multi Layer Perceptron

4.2.1 Motivazione della Scelta

La scelta di utilizzare un Multi Layer Perceptron è motivata dalle seguenti considerazioni specifiche per il dominio della cybersecurity:

- **Capacità di modellazione non-lineare**: I pattern di attacco presentano relazioni complesse e non-lineari tra le feature che gli MLP possono catturare efficacemente
- **Robustezza al rumore**: I dati di rete reali contengono spesso rumore e anomalie che gli MLP gestiscono bene grazie alla loro natura distribuita
- **Interpretabilità**: Attraverso tecniche di permutation importance, è possibile analizzare il contributo delle singole feature
- **Efficienza computazionale**: Bilanciamento ottimale tra prestazioni e costi computazionali per deployment in tempo reale

4.2.2 Configurazione Architetturale

L'architettura del MLP implementato è illustrata nella Figura 2 e presenta la seguente configurazione:

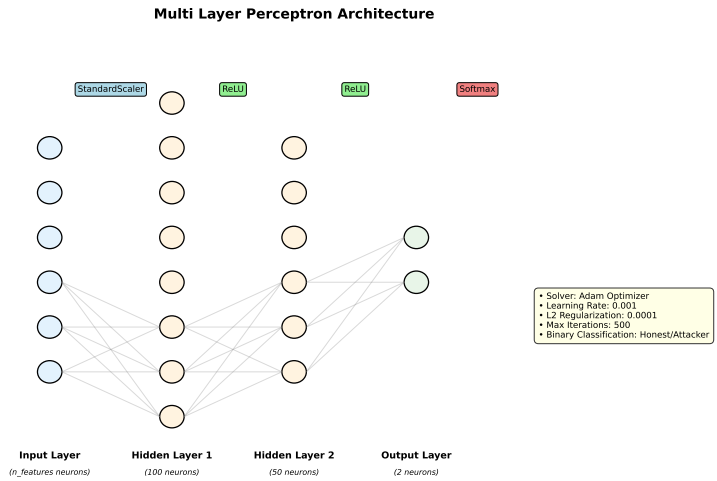


Figura 2: Architettura dettagliata del Multi Layer Perceptron

- **Layer di input**: Dimensione variabile in base al numero di feature ($n_{features}$)
- **Hidden layer 1**: 100 neuroni con attivazione ReLU
- **Hidden layer 2**: 50 neuroni con attivazione ReLU
- **Layer di output**: 2 neuroni per classificazione binaria con attivazione softmax

Questi sono solo parametri di default usati come base di partenza per la costruzione del nostro MLP.

4.3 Pipeline di Preprocessing

4.3.1 Standardizzazione delle Feature

Il preprocessing implementa una pipeline robusta illustrata nella Figura 3:

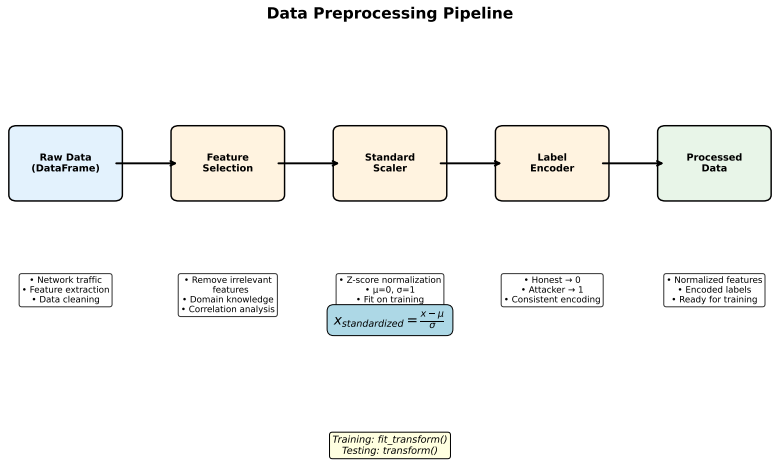


Figura 3: Pipeline di preprocessing dei dati

La standardizzazione delle feature viene eseguita utilizzando `StandardScaler`:

$$x_{standardized} = \frac{x - \mu}{\sigma} \tag{1}$$

dove μ è la media e σ la deviazione standard calcolate sul training set. Questa normalizzazione è cruciale per: garantire convergenza stabile dell'algoritmo di ottimizzazione e prevenire che feature con scale maggiori dominino l'apprendimento.

4.3.2 Encoding delle Etichette

Le etichette categoriche (Honest/Attacker) vengono codificate numericamente tramite `LabelEncoder`, garantendo consistenza tra training e inference e facilitando il calcolo delle metriche di performance.

4.4 Processo di Training e Ottimizzazione

4.4.1 Strategia di Validazione

Il training implementa una strategia di validazione robusta:

1. **Split stratificato**: Divisione training/validation (80/20) mantenendo distribuzione delle classi
2. **Monitoraggio in tempo reale**: Valutazione delle performance su validation set durante training

3. **Early stopping implicito**: Limitazione del numero di iterazioni per prevenire overfitting

La funzione di loss utilizzata è la categorical cross-entropy:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K} y_{ij}\log(\hat{y}_{ij}) \tag{2}$$

4.5 Sistema di Valutazione Completo

4.5.1 Metriche di Performance

Il sistema implementa un framework di valutazione comprensivo che include:

- **Accuracy**: Frazione di predizioni corrette
- **AUC Score**: Area sotto la curva ROC per valutare capacità discriminativa
- **Classification Report**: Precision, recall, F1-score per classe
- **Confusion Matrix**: Analisi dettagliata degli errori di classificazione

4.5.2 Visualizzazioni Diagnostiche

Il sistema genera automaticamente visualizzazioni per l'analisi delle performance:

- **Curva ROC**: Valutazione del trade-off sensitivity/specificity
- **Confusion Matrix heatmap**: Identificazione pattern di errore
- **Training loss curve**: Monitoraggio convergenza del modello

4.6 Pipeline di Inferenza

La Figura 4 illustra la pipeline completa per la classificazione in tempo reale:

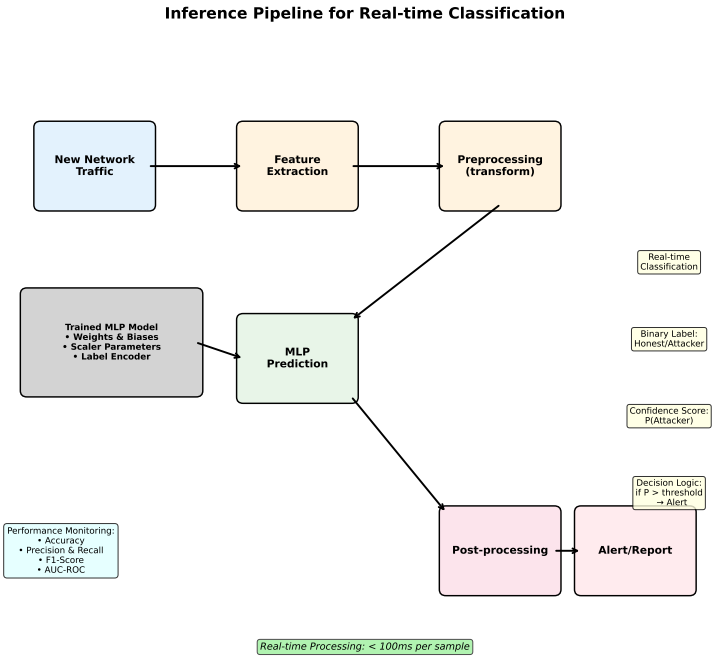


Figura 4: Pipeline di inferenza per classificazione in tempo reale

Il processo di inferenza include:

1. **Acquisizione dati**: Ricezione di nuovi campioni di traffico
2. **Preprocessing**: Applicazione delle stesse trasformazioni del training
3. **Predizione**: Classificazione tramite modello addestrato
4. **Post-processing**: Decodifica delle predizioni e calcolo delle probabilità
5. **Output**: Etichetta binaria e confidence score

5 Valutazione

Per valutare le performance del modello proposto, sono state utilizzate quattro metriche di valutazione standard nel campo del machine learning e della classificazione. Queste metriche forniscono una visione completa delle capacità predittive del sistema, analizzando diversi aspetti delle prestazioni classificatorie.

5.1 Metriche Utilizzate

5.1.1 Accuracy

L'accuracy rappresenta la frazione di predizioni corrette sul totale delle predizioni effettuate:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

dove:

- TP = True Positives (veri positivi)
- TN = True Negatives (veri negativi)
- FP = False Positives (falsi positivi)
- FN = False Negatives (falsi negativi)

Questa metrica fornisce una misura generale delle prestazioni del modello, indicando la percentuale di classificazioni corrette.

5.1.2 Precision

La precision misura la capacità del modello di identificare correttamente i casi positivi, calcolata come:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4}$$

Questa metrica è particolarmente importante quando il costo dei falsi positivi è elevato, in quanto indica la proporzione di predizioni positive che sono effettivamente corrette.

5.1.3 Recall

Il recall quantifica la capacità del modello di identificare tutti i casi positivi presenti nel dataset:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{5}$$

Questa metrica è fondamentale quando è critico non perdere casi positivi, misurando la proporzione di casi positivi reali che sono stati correttamente identificati.

5.1.4 F1-Score

L'F1-score rappresenta la media armonica tra precision e recall, fornendo una singola metrica che bilancia entrambi gli aspetti:

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6}$$

Questa metrica è particolarmente utile quando si desidera un equilibrio tra precision e recall, specialmente in presenza di dataset sbilanciati.

6 Esperimenti

Questa sezione presenta gli esperimenti condotti per ottimizzare e valutare le performance del Multi Layer Perceptron proposto per il task di intrusion detection. Gli esperimenti sono stati strutturati in modo sistematico per esplorare lo spazio degli iperparametri e identificare la configurazione ottimale del modello.

6.0.1 Strategia di Validazione

Per garantire robustezza e affidabilità dei risultati, è stata adottata una strategia di validazione a tre livelli:

1. **Train/Validation/Test Split**: 60%/20%/20% con stratificazione per mantenere la distribuzione delle classi
2. **Cross-validation**: 5-fold stratificata per l'ottimizzazione degli iperparametri
3. **Holdout finale**: Test set completamente isolato per valutazione finale

6.1 Metodologia di Tuning degli Iperparametri

6.1.1 Approccio Sistematico al Tuning

È stata implementata una classe `MLPTuner` per automatizzare il processo di ottimizzazione degli iperparametri. L'approccio segue una metodologia strutturata che compie una esplorazione sistematica dello spazio dei parametri tramite Grid Search.

6.1.2 Spazio degli Iperparametri Esplorato

La Tabella 1 presenta lo spazio completo degli iperparametri investigato:

Tabella 1: Spazio degli iperparametri esplorato durante il tuning

Parametro	**Valori Testati**	**Motivazione**
`hidden_layer_sizes`	(50,), (100,), (150,), (50,25), (100,50), (150,75)	Bilanciamento complessità vs overfitting
`activation`	relu, tanh, logistic	Diverse caratteristiche di non-linearità
`solver`	adam, sgd, lbfgs	Algoritmi di ottimizzazione per diverse dimensioni
`alpha`	0.0001, 0.001, 0.01	Controllo regolarizzazione L2
`learning_rate_init`	0.001, 0.01	Velocità di convergenza del training

6.2 Risultati del Tuning degli Iperparametri

6.2.1 Processo di Ottimizzazione

Il processo di Grid Search ha esplorato 108 combinazioni di iperparametri utilizzando 5-fold cross-validation. Dopo l'esplorazione sistematica, la configurazione ottimale identificata è:

Tabella 2: Configurazione ottimale degli iperparametri

Parametro	**Valore Ottimale**
`hidden_layer_sizes`	(50, 25)
`activation`	logistic
`solver`	adam
`alpha`	0.001
`learning_rate_init`	0.001
`max_iter`	1000

6.2.2 Analisi della Convergenza

La Figura 5 mostra l'evoluzione della loss function durante il training del modello ottimale:

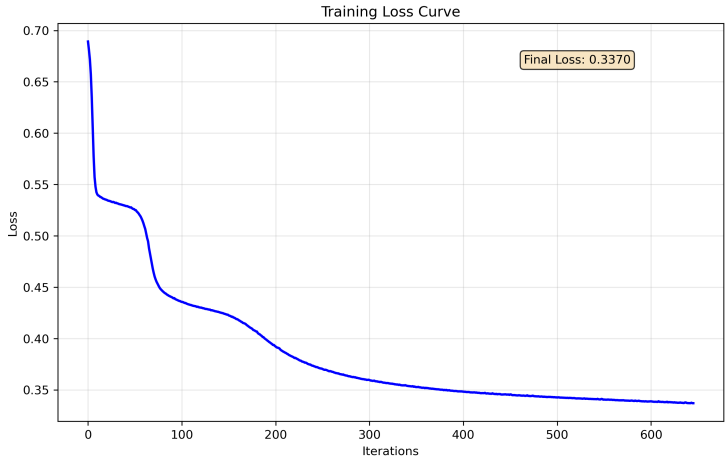


Figura 5: Curva di convergenza del modello ottimale durante il training

6.3 Valutazione delle Performance

6.3.1 Risultati su Validation Set

Durante la fase di tuning, il modello ottimale ha mostrato le seguenti performance sul validation set:

Tabella 3: Performance del modello ottimale su validation set

Metrica	**Honest**	**Attacker**
Precision	0.82	0.99
Recall	0.99	0.74
F1-Score	0.90	0.84
Accuracy		0.8781
AUC-ROC		0.0.8722

6.3.2 Risultati Finali su Test Set

Le performance finali del modello ottimale sul test set completamente isolato sono presentate nella Tabella 4:

Tabella 4: Performance finali del modello su test set

Metrica	**Honest**	**Attacker**
Precision	0.80	0.99
Recall	0.99	0.71
F1-Score	0.89	0.83
Accuracy		0.8643
AUC-ROC		0.8605

6.3.3 Analisi della Confusion Matrix

La Figura 7 presenta la confusion matrix dettagliata per il test set:

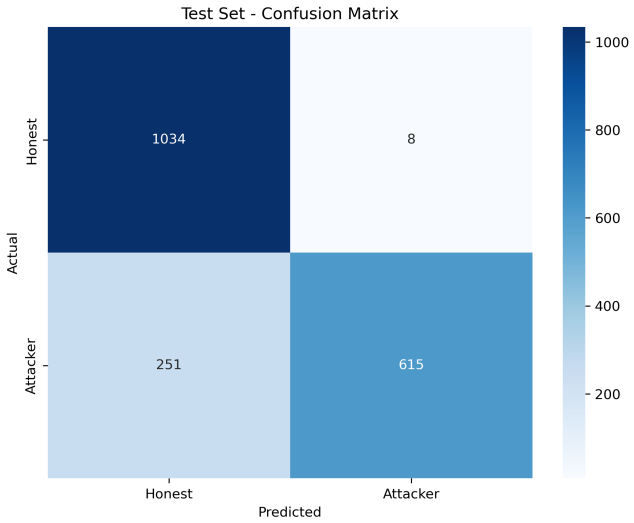


Figura 6: Confusion matrix del modello finale su test set

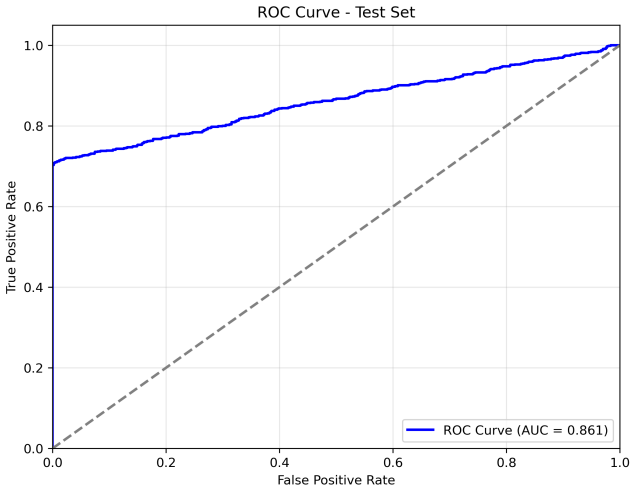


Figura 7: Curva ROC del modello finale sul test set

L'analisi rivela:

- **True Negatives**: 1,034

- **False Positives**: 251
- **False Negatives**: 8
- **True Positives**: 615

6.4 Discussione dei Risultati

I risultati ottenuti dimostrano che il modello è riuscito ad apprendere bene dai dati di training generalizzando sui dati di validation e test. Possiamo notare un numero particolarmente basso di falsi negativi, ciò significa che il sistema non genererà di frequente falsi allarmi bloccando utenti in realtà legittimi. Lo stesso non si può dire per i falsi positivi, una frazione considerevole (circa il 29%) di attacchi malevoli non è stata riconosciuta come tale dal sistema.

7 Demo

Per la realizzazione della demo sono stati creati un semplice sito web (hostato in locale) con autenticazione e un server python. Il sito raccoglie i dati sugli utenti che effettuano il login mentre il server sancisce se l'utente è malevolo oppure no. Il video in allegato mostra un esempio di detection. Come prima cosa viene registrato un nuovo utente e questi fa il login inserendo le credenziali richieste. Il sistema non riconosce alcuna minaccia e l'utente viene segnalato come legittimo. In seguito c'è un tentativo di brute forcing della password dell'utente admin, il sistema interpeta l'atto come un attacco e segnala la minaccia.

8 Codice

Il codice consegnato è organizzato nei seguenti file:

- **Progetto Machine Learning.ipynb**: contiene la definizione del modello, gli esperimenti fatti per determinare i parametri e le valutazioni effettuate.
- **cybersecurity_intrusion_data.csv**: contiene il dataset.
- **cybersecurity_mlp_model.jolib**: contiene il modello finale addestrato che sarà caricato dal server python.
- **index.php**: contiene il sito web della demo.
- **server.py**: contiene il server pyhton.

9 Conclusione

Il presente progetto ha affrontato il problema dell'intrusion detection attraverso l'applicazione di tecniche di machine learning su un dataset di attività di rete. L'obiettivo principale era quello di sviluppare un sistema in grado di distinguere automaticamente tra traffico benigno e attività malevole basandosi su caratteristiche comportamentali degli utenti e dei processi.Durante lo sviluppo di questo progetto sono emerse diverse considerazioni importanti da cui apprendere:

Importanza della Qualità dei Dati: La fase di analisi esplorativa si è rivelata cruciale per comprendere le caratteristiche del dataset e identificare potenziali problemi. La presenza di outliers e la distribuzione delle feature hanno influenzato significativamente le performance dei modelli, sottolineando l'importanza di un preprocessing accurato.

Selezione delle Feature: Non tutte le caratteristiche disponibili si sono dimostrate ugualmente utili per la classificazione. Si è reso necessario identificare le variabili più discriminanti, migliorando sia l'efficienza computazionale che le performance predittive.

9.1 Miglioramenti futuri

Per quanto riguarda i miglioramenti futuri si potrebbe pensare di raccogliere più feature. L'introduzione di feature derivate, come rapporti tra variabili esistenti o aggregazioni temporali, potrebbe catturare pattern più complessi nel comportamento degli utenti. L'inclusione di informazioni temporali e l'analisi di sequenze di eventi potrebbero rivelare pattern di attacco più sofisticati che non emergono dall'analisi di singole sessioni isolate.