



UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea Magistrale in Informatica

TESI DI LAUREA

Utilizzo di tecniche di Machine Learning per l'Attack Detection in ambito IoT: uno studio empirico

RELATORE

Prof. Fabio Palomba

Dott. Giammaria Giordano

Università degli Studi di Salerno

CANDIDATO

Daniele Palmieri

Matricola: 0522500751

Anno Accademico 2022-2023

Questa tesi è stata realizzata nel

sesa^{lab}
SOFTWARE ENGINEERING
SALERNO

"Non c'è nulla di nobile nell'essere superiore a qualcun altro.

La vera nobiltà è essere superiore a chi eravamo ieri."

Ernest Hemingway

Abstract

La diffusione dei servizi e dei dispositivi nell' Internet of Things ha aperto nuovi studi nell'ambito della sicurezza. Il presente lavoro di ricerca ha lo scopo di andare a confrontare, sulla base degli studi esistenti in letteratura, diverse tecniche di Machine Learning impiegate sul task di attack detection nei dispositivi IoT, andando eventualmente a sottolineare ed evidenziare alcune criticità emerse dai lavori già svolti e proponendo delle possibili soluzioni ad esse. In modo preliminare verranno analizzate alcuni dei lavori svolti in letteratura, le criticità presenti e le soluzioni proposte, dopodichè verranno presentati i modelli adottati con conseguente fase di sperimentazione. I risultati dell'esperimento hanno dimostrato che l'IA può essere uno strumento utile per il task di attack detection, seppur non risulta essere ancora pienamente soddisfacente. Tuttavia i modelli non sono l'unica cosa da tenere in considerazione, questi infatti risultano avere delle performance molto simili tra loro, un buon lavoro di preprocessing dei dati e l'utilizzo di metriche di valutazione adeguate sono fondamentali per l'integrità dei risultati.

Indice

Elenco delle Figure	iv
1 Introduzione	1
1.1 Contesto applicativo	1
1.2 Motivazione e obiettivi	4
1.3 Metodo di ricerca	5
1.4 Risultati	5
1.5 Struttura della tesi	6
2 Background e stato dell'arte	7
2.1 Background teorico	7
2.1.1 Tasks di privacy affrontati con l'IA	8
2.1.2 Domini IoT in cui sono state applicate le tecniche di IA	11
2.1.3 Tipologie di algoritmi impiegati	11
2.1.4 Dataset utilizzati	12
2.1.5 Tecniche di validazione impiegate	14
2.1.6 Metriche di valutazione utilizzate	15
2.2 Background teorico sul task di attack detection	16
2.3 Algoritmi e dataset utilizzati per l'attack detection	19
2.3.1 Algoritmi	19

2.3.2	Dataset	23
2.4	Riflessioni sulle Limitazioni dello Stato dell'Arte	26
3	Metodologia proposta	29
3.1	Dataset impiegato e pre-processing	30
3.1.1	Analisi NLS-KDD	30
3.1.2	Pre-processing	30
3.2	Algoritmi di ML utilizzati	35
3.2.1	Logistic Regression	35
3.2.2	Artificial Neural Network	36
3.3	Tecniche di validazione utilizzate	38
3.4	Metriche di valutazione adottate	38
3.5	Configurazione degli iperparametri	39
3.5.1	Training - Validation split e Folds utilizzati	39
3.5.2	RFE, Oversamplig e StandardScaling	40
3.5.3	Iperparametri algoritmi di ML	40
4	Risultati	43
4.1	Risultati ottenuti senza oversampling e con drop degli attributi categorici	43
4.1.1	Support Vector Machine	44
4.1.2	Logistic Regression	49
4.1.3	KNN	53
4.1.4	Decision Tree	58
4.1.5	Random Forest	61
4.1.6	Neural Network	64
4.2	Risultati ottenuti applicando l'oversampling	69
4.2.1	Support Vector Machine	69
4.2.2	Logistic Regression	71
4.2.3	KNN	71
4.2.4	Decision Tree	72
4.2.5	Random Forest	72
4.2.6	Neural Network	73
4.3	Risultati ottenuti applicando One Hot Encoding	73

4.3.1	Support Vector Machine	73
4.3.2	Logistic Regression	74
4.3.3	KNN	74
4.3.4	Decision Tree	74
4.3.5	Random Forest	75
4.3.6	Neural Network	75
4.4	Risultati ottenuti su Binary Dataset	75
4.4.1	Support Vector Machine	76
4.4.2	Logistic regression	76
4.4.3	KNN	77
4.4.4	Decision Tree	78
4.4.5	Random Forest	78
4.4.6	Neural Network	79
5	Conclusioni	80
5.1	Interpretazione dei risultati	80
5.2	Risultati finali per le domande di ricerca	83
5.3	Limitazioni e sviluppi futuri	84
	Bibliografia	85

Elenco delle figure

2.1	Topics frequency, fonte [1]	10
2.2	(a) Training Accuracy (b) Test Accuracy, fonte [2]	17
2.3	fonte [3]	18
2.4	fonte [3]	18
2.5	Illustrazione SVM, fonte [4]	20
2.6	Illustrazione Decision Tree.	21
2.7	Illustrazione Random Forest, fonte [5]	22
2.8	Illustrazione K-NN, fonte [6]	23
3.1	Rappresentazione distribuzione Training set.	31
3.2	Rappresentazione distribuzione Test set.	31
3.3	Rappresentazione distribuzione Training set binario.	32
3.4	OneHotEncoding.	33
3.5	Training set dopo aver applicato SMOTE.	34
3.6	Logistic Regression	36
3.7	Immagine che mostra come i neuroni X_n generano un risultato Y_n utilizzando i pesi W_n	36
3.8	Deep Neural Network.	37
3.9	Matrice di confusione.	39
3.10	Struttura Neural Network implementata.	42

4.1	Distribuzione dataset senza oversampling.	44
4.2	SVM, test set prediction su Dos, original data (a), StandardScaling (b).	45
4.3	SVM, test set prediction su Probe, original data (a), StandardScaling (b).	46
4.4	SVM, test set prediction su R2L, original data (a), StandardScaling (b).	47
4.5	SVM, test set prediction su U2R, original data (a), StandardScaling (b).	48
4.6	SVM, test set prediction su Probe, feature selection.	49
4.7	Logistic Regression, test set prediction su Dos, original data (a), StandardScaling (b).	50
4.8	Logistic Regression, test set prediction su Probe, original data (a), StandardScaling (b).	51
4.9	Logistic Regression, test set prediction su R2L, original data (a), StandardScaling (b).	52
4.10	Logistic Regression, test set prediction su U2R, original data (a), StandardScaling (b).	53
4.11	KNN, test set prediction su Dos, original data (a), StandardScaling (b).	54
4.12	KNN, test set prediction su Probe, original data (a), StandardScaling (b).	55
4.13	KNN, test set prediction su R2L, original data (a), StandardScaling (b).	56
4.14	KNN, test set prediction su U2R, original data (a), StandardScaling (b).	57
4.15	KNN, test set prediction su Dos, RFE (a), Probe (b)	58
4.16	DT, test set prediction su Dos, original data (a), StandardScaling (b).	59
4.17	DT, test set prediction su Probe, original data (a), StandardScaling (b).	60
4.18	DT, test set prediction su R2L, original data (a), StandardScaling (b).	60
4.19	DT, test set prediction su U2R, original data (a), StandardScaling (b).	61
4.20	RF, test set prediction su Dos, original data (a), StandardScaling (b).	62
4.21	RF, test set prediction su Probe, original data (a), StandardScaling (b).	63
4.22	RF, test set prediction su R2L, original data (a), StandardScaling (b).	63
4.23	RF, test set prediction su U2R, original data (a), StandardScaling (b).	64
4.24	NN, test set prediction su Dos, original data (a), StandardScaling (b).	65
4.25	NN, test set prediction su Probe, original data (a), StandardScaling (b).	66
4.26	NN, test set prediction su R2L, original data (a), StandardScaling (b).	67
4.27	NN, test set prediction su U2R, original data (a), StandardScaling (b).	68
4.28	NN, test set prediction su Dos(a) e su Probe(b), feature selection.	69

4.29 SVM, binary test set.	76
4.30 LR, binary test set.	77
4.31 KNN, binary test set.	77
4.32 DT, binary test set.	78
4.33 RF, binary test set.	79
4.34 NN, binary test set.	79
5.1 U2R: k-fold cross validation(a), stratified cross validation.(b)	81

CAPITOLO 1

Introduzione

1.1 Contesto applicativo

Nell'ultima decade si è vista una crescita esponenziale dell'Internet of Things(IoT). Per IoT intendiamo una rete di oggetti (things) dotati di sensori, software e altre tecnologie integrate, con lo scopo di connettere e scambiare dati con altri dispositivi e sistemi in rete. Questi dispositivi vanno da comuni oggetti domestici a sofisticati strumenti industriali. Esempi di dispositivi IoT possono essere: il robot lava pavimenti, le telecamere IP, il frigo intelligente, i sistemi smart car o i popolarissimi dispositivi dotati di assistenti virtuali. Tutti questi strumenti migliorano la qualità della vita agli utenti fornendo una serie di comfort che si traducono spesso in minor lavoro da svolgere da parte dell'utente stesso. Un esempio è l'utilizzo del frigo intelligente che comunica all'utente i cibi che sono presenti nel frigo e quali tra essi sono in scadenza, suggerendo anche alcune ricette da poter preparare con gli ingredienti a disposizione. Ma qual è l'altro lato della medaglia?

Se da un lato è comodo controllare cosa c'è nel frigo mentre si è al supermercato, allo stesso tempo, questa connettività fa sì che i dispositivi connessi siano vulnerabili ad attacchi di cybercriminali, il che si traduce in un maggior numero di punti di accesso per le minacce digitali. Un attacco ad un dispositivo IoT potrebbe fornire ai

malintenzionati informazioni sensibili ottenute attraverso sensori come videocamera o microfono, oppure potrebbe causare dei disagi su più livelli.

Ad esempio nel 2016 la botnet Mirai ha infettato numerosi dispositivi IoT, principalmente router e telecamere IP. Successivamente i dispositivi infettati sono stati utilizzati per buttare giù con un attacco DDoS, Dyn, una società che fornisce un servizio di Dns, mettendo in ginocchio gran parte delle reti Internet in Europa e negli Stati Uniti, rendendo non disponibili per ore importanti siti web come Twitter, Netflix, Spotify e molti altri.

Un altro esempio si è verificato nel novembre 2016 in Finlandia: cybercriminali hanno interrotto il riscaldamento di due condomini nella città di Lappeenranta. L'attacco è stato portato a termine con un DDoS che, andando a colpire i regolatori smart del riscaldamento degli edifici, li obbligava a riavviarsi continuamente, impedendo di fatto di erogare il riscaldamento creando non pochi disagi ai residenti.

Uno scenario più allarmante è stato dato nel 2015. Due ricercatori, Charlie Miller e Chris Valasek, sono stati in grado di assumere il controllo totale di un SUV Jeep Cherokee sfruttando una vulnerabilità dell'aggiornamento del firmware. I ricercatori hanno dirottato il veicolo sulla rete cellulare Sprint, riuscendo in seguito a modificare il funzionamento di diversi sistemi, dall'aria condizionata ai tergicristalli. Il test ha dimostrato con successo, inoltre, la possibilità di manomettere il veicolo facendolo accelerare, rallentare e persino sterzare, potendo potenzialmente causare gravi incidenti.

In generale i dispositivi IoT possono essere oggetto di molteplici tipologie di attacco che possiamo andare a raggruppare in tre macro categorie:

- A livello del dispositivo:
 - I dispositivi possono essere il mezzo principale con cui vengono avviati gli attacchi. Le parti di un dispositivo da cui possono provenire le vulnerabilità sono la memoria, il firmware, l'interfaccia fisica, l'interfaccia web e i servizi di rete. Gli aggressori possono anche trarre vantaggio da impostazioni predefinite non sicure, componenti obsoleti e meccanismi di aggiornamento non sicuri.

- A livello di rete:
 - Gli attacchi possono avvenire direttamente sulla rete e sui canali di comunicazione che collegano tra loro i componenti IoT. I protocolli utilizzati nei sistemi IoT possono avere problemi di sicurezza che possono interessare l'intero sistema. I sistemi IoT sono inoltre suscettibili agli attacchi di rete noti come denial of service (DoS) e spoofing.
- A livello di applicazione:
 - Le vulnerabilità nel software correlato per i dispositivi IoT (es. un'app Android per controllare il dispositivo) possono portare a sistemi compromessi. Le applicazioni possono, ad esempio, essere sfruttate per rubare le credenziali degli utenti o per inviare aggiornamenti firmare al dispositivo che però nascondono codice malevolo.

Possiamo dedurre quindi che tutti i principali componenti dei sistemi IoT possono essere sfruttati da potenziali attaccanti. Questo rende fondamentale, da parte dei produttori, l'utilizzo di misure che possano andare a garantire all'utente finale privacy e sicurezza. Alcune misure di sicurezza tipicamente adottate sono autenticazione a due fattori, protocolli di encryption e protezioni hardware.

Tuttavia, allo stato attuale, tali misure di sicurezza non sono sempre adeguate e talvolta non vengono neppure adottate. Meneghello et Al. [7] ha definito l'IoT come "Internet of Threats", l'internet delle minacce, andando ad evidenziare l'urgente necessità di meccanismi automatizzati che possano supportare il rilevamento di problemi di privacy e sicurezza nei sistemi IoT.

La comunità scientifica ha iniziato a lavorare su questa necessità introducendo tecniche basate ad esempio su blockchain e schemi di data aggregation. Di recente il focus si è spostato sull'utilizzare tecniche di Intelligenza Artificiale (IA) per l'identificazione di potenziali attacchi nell'ambito dell'IoT.

Nella Systematic Literature Review svolta da Giordano et al. [1], che è alla base di questo lavoro di tesi, è emerso come il focus dei ricercatori in questo ambito è incentrato maggiormente su sei task:

- Network Analysis
- Attack Detection
- Framework Building
- User Authentication
- Malware Detection
- Privacy-Preserving Scheme

I task verranno discussi più nel dettaglio nel **Capitolo 2**.

1.2 Motivazione e obiettivi

Partendo dalla Systematic Literature Review, vedremo che molte delle ricerche condotte in letteratura hanno ottenuto risultati soddisfacenti. Tuttavia tali risultati potrebbero essere poco veritieri a causa di alcune criticità che non sono state adeguatamente affrontate. Dai risultati ottenuti dai ricercatori, infatti, sono emerse diverse limitazioni, come ad esempio l'utilizzo di metriche non sufficienti per una corretta valutazione dei modelli o l'impiego di dataset sbilanciati per il training dei modelli stessi. Tali limitazioni verranno discusse in modo approfondito nel **Capitolo 2**.

Il progetto svolto è incentrato in particolare sul task di attack detection che avviene sfruttando delle tecniche di Machine Learning (ML), una branca dell'IA che studia algoritmi e tecniche che consentono ad una macchina di imparare dai dati, identificare modelli autonomamente e prendere decisioni con un intervento umano ridotto al minimo.

L'obiettivo di tesi è quello di effettuare uno studio empirico, mettendo a confronto diverse tecniche di Machine Learning applicate sul task di attack detection. Partendo dalle limitazioni emerse dai lavori in letteratura, verranno adottate delle possibili soluzioni ad esse con lo scopo di andare a verificare se, mitigando tali limitazioni, l'approccio tramite IA può essere effettivamente una soluzione valida al problema di privacy preserving in scenari reali.

1.3 Metodo di ricerca

Come anticipato nella sezione precedente, in questo studio empirico si cerca di mitigare alcune delle limitazioni riscontrate su lavori già effettuati in letteratura. Nella Systematic Literature Review vengono analizzati e categorizzati molti lavori di ricerca con le relative criticità più frequenti. Al netto delle criticità individuate sono state valutate e implementate delle possibili soluzioni ad esse. Le mitigazioni sono state applicate sia in fase di pre-processing dei dati, andando ad agire dunque a livello del dataset, sia in fase di validazione dei modelli e valutazione dei risultati ottenuti. Mediante questi accorgimenti si è cercato, dunque, di andare ad effettuare uno studio empirico che potesse esprimere risultati influenzati il meno possibile da limitazioni a livello di pipelining. Tra gli accorgimenti adottati ci sono: l'utilizzo di tecniche di oversampling, l'utilizzo di diverse tecniche di validazione e l'utilizzo di metriche differenti per l'interpretazione dei risultati. Nel **Capitolo 3** viene illustrata nel dettaglio la metodologia proposta.

1.4 Risultati

I risultati ottenuti nella sperimentazione, consultabili nel **Capitolo 4**, verificano l'ipotesi secondo cui, in molti dei lavori presenti in letteratura, ci sono delle criticità importanti che influiscono in maniera non trascurabile sui risultati ottenuti; pertanto adottare degli accorgimenti per andare quantomeno a limitare tali criticità risulta essere di fondamentale importanza per la qualità dei risultati stessi. I modelli di ML già presenti e analizzati in letteratura, inoltre, risultano performare tutti molto bene in ambito sperimentale senza differenze di performance troppo marcate tra di loro, tuttavia tali performance potrebbero essere differenti in scenari d'uso reali. Durante lo studio infatti i modelli hanno performato in modo decisamente soddisfacente sul validation set e in fase di training, ma sono stati meno convincenti sul test set in cui sono stati somministrati ai modelli delle tipologie di dati che non hanno mai visto in fase di training.

1.5 Struttura della tesi

L'elaborato si compone di cinque capitoli totali, oltre questo primo capitolo introduttivo, gli altri capitoli sono:

- Capitolo 2: Background e Stato dell'arte.
 - In questo capitolo viene analizzato più in dettaglio lo stato dell'arte, il quadro generale di ciò che è già stato fatto ed è presente in letteratura traendo informazioni utili per l'impostazione della sperimentazione effettuata in questo studio.
- Capitolo 3: Metodologia proposta.
 - Nel capitolo 3 viene spiegato quello che è stato fatto in questo lavoro di tesi, quindi tutti gli accorgimenti e le mitigazioni adottate. Ad esempio: quali sono gli step di pre-processing effettuati sul dataset, la motivazione alla base della scelta del dataset stesso, come sono stati impostati gli iperparametri degli algoritmi e le varie tecniche implementate.
- Capitolo 4: Risultati.
 - Nel capitolo 4 vengono mostrati i risultati ottenuti durante lo studio effettuato. Tali risultati vengono illustrati andando a suddividere le varie fasi di sperimentazione in base alle differenti tecniche di ML adottate e alle diverse manipolazioni effettuate sul dataset; l'analisi dei risultati sarà supportata dalla rappresentazione delle matrici di confusione ottenute.
- Capitolo 5: Conclusioni.
 - Nel quinto e ultimo capitolo vi è un commento generale sullo studio condotto, verrà fatto un resoconto dei risultati e verranno tratte delle conclusioni sul lavoro svolto, sulle limitazioni riscontrate e sui possibili miglioramenti futuri.

Background e stato dell'arte

In questo capitolo viene effettuata una panoramica sul background e lo stato dell'arte relativi all'applicazione di algoritmi di machine learning nell'ambito della sicurezza su dispositivi IoT considerando alcuni dei lavori presenti in letteratura.

2.1 Background teorico

Con il crescente impiego dei dispositivi IoT, sono state effettuate numerose ricerche, da parte della comunità scientifica, riguardo l'utilizzo di tecnologie di Intelligenza Artificiale e Machine Learning per aumentare la sicurezza e la privacy nei suddetti dispositivi. Tali ricerche coprono gli ambiti più disparati, sfruttano tecnologie differenti e mirano a diverse necessità. Per andare a sintetizzare un background generale riguardo l'ambito di ricerca, andiamo ad analizzare il lavoro di Systematic Literature Review (che abbrevieremo in SLR) svolto da Giordano et al. [1], in cui, i ricercatori hanno effettuato una raccolta di 152 lavori presenti in letteratura sull'utilizzo di tecniche di Machine Learning per la privacy nei sistemi IoT, analizzandoli sotto diverse prospettive e andando a rispondere a diverse domande che analizzeremo nei punti successivi.

2.1.1 Tasks di privacy affrontati con l'IA

La prima domanda a cui si vuole rispondere nella SLR svolta da Giordano et al. riguarda i tasks di privacy preserving che sono stati affrontati utilizzando l'IA. Vogliamo infatti capire in che tipo di problemi, relativi alla privacy e alla sicurezza, l'impiego di tecniche di IA può venirci in aiuto. Dal lavoro svolto, sono stati evidenziati 6 task su cui i ricercatori hanno incentrato il loro focus:

- Network Analysis
- Attack Detection
- Framework Building
- User Authentication
- Malware Detection
- Privacy-Preserving Scheme

Network Analysis

È il primo task identificato da Giordano et Al. nella SLR, si tratta del task su cui i ricercatori hanno lavorato maggiormente, ricoprendo il 24.4% degli studi totali analizzati. Questo task consiste nel ricercare la presenza di traffico o attività malevole sulla rete, in modo che, attraverso modelli di IA, si vanno ad identificare eventuali informazioni sensibili trasmesse dai dispositivi IoT. Nella SLR viene citato il lavoro svolto da Fei et Al. [8] in cui viene collezionato il traffico della rete per allenare un algoritmo Random Forest in grado di riconoscere del traffico inusuale che potrebbe nascondere un attacco DDoS.

Attack Detection

Il task dell'attack detection, secondo la SLR, ricopre il 23.7% degli studi portati avanti dai ricercatori. Per attack detection intendiamo il task che mira a rilevare possibili azioni malevole, in particolare categorizzabili in due famiglie:

- **Intrusion Detection**
 - Si utilizza un componente software o hardware per rilevare possibili attacchi su un device IoT. Attraverso l'analisi del traffico di rete si va ad individuare attività sospette, come ransomware o phishing.
- **Anomaly Detection**
 - A differenza della intrusion detection, che analizza basandosi su attacchi conosciuti, in questo scenario ci si affida a modelli statistici per validare il traffico in entrata o in uscita.

Per questi tipi di task viene tipicamente utilizzata l'IA per analizzare il traffico alla ricerca di pattern che possono indicare un'intrusione nel sistema.

Framework Building

Consiste nella creazione di frameworks che possono essere poi utilizzati per sperimentare nuove tecniche al fine di addestrare modelli di machine learning mirati al contesto della privacy, ad esempio, Wang et al. [9] nel loro lavoro sono andati a definire un approccio di apprendimento federato che consente agli utenti di distribuire problemi di clustering in cloud.

User Authentication

Per user authentication ci si riferisce a strumenti di autenticazione che sfruttano sensori biometrici. Questa peculiarità è di particolare importanza nell'ambito della healthcare, in cui vengono utilizzati sensori biometrici per monitorare i pazienti e successivamente i parametri rilevati vengono utilizzati per generare degli ID che vengono poi usati per accedere ad aree o sistemi riservati.

Malware Detection

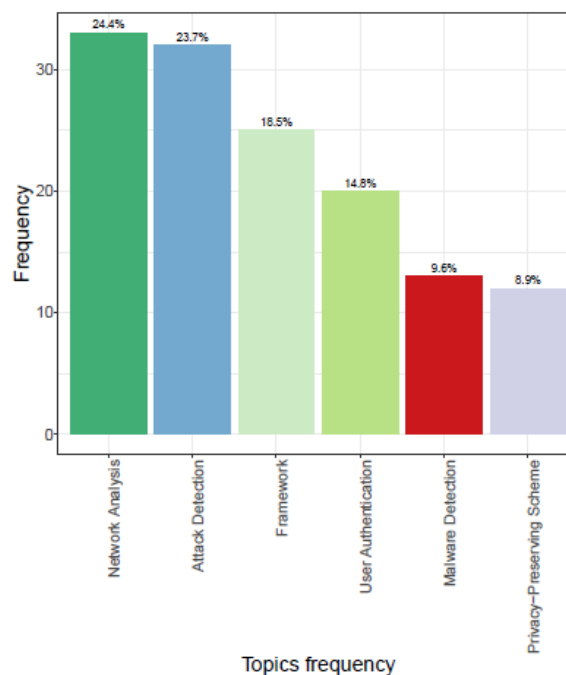
Questo task consiste nella creazione di agenti che analizzano i processi in esecuzione su un host al fine di identificare possibili malware. Il task più comune riguarda l'identificazione e la classificazione di vari tipi di malware; nello specifico abbiamo due trend di utilizzo:

- Il primo utilizza il mining di pattern per rilevare applicazioni malevole; ad esempio Darabian et Al. [10] hanno utilizzato mining di pattern sequenziali per rilevare le sequenze di codice operativo più frequenti di applicazioni IoT malevole. Tali sequenze sono poi utilizzate per distinguere applicazioni malevole da applicazioni benigne.
- Nel secondo trend vengono utilizzati approcci di machine learning supervisionato per classificare i malware, ad esempio Alam et Al. [11] hanno allenato un algoritmo Random Forest con dati di malware provenienti da applicazioni android in modo da identificare app malevole.

Privacy-Preserving Scheme

Fa riferimento alla definizione di nuovi protocolli per migliorare la privacy. Stando alla SLR, gli autori degli studi in questo task tendono ad utilizzare blockchain o meccanismi simili per mantenere la sicurezza dei dati. Nel lavoro svolto da Zhao et Al. [12] viene ideato un approccio di apprendimento federato basato su blockchain, in cui i dati raccolti da più sensori vengono archiviati all'interno di una blockchain in modo da preservarne la privacy prima di essere dati ai modelli di machine learning.

Figura 2.1: Topics frequency, fonte [1]



2.1.2 Domini IoT in cui sono state applicate le tecniche di IA

Abbiamo visto i tasks in cui le tecniche di IA sono impiegate. Ciò che viene naturale è chiedersi in quali domini quotidiani si possono andare a svolgere tali tasks. Giordano et Al. nella SLR, basandosi sui papers raccolti, sono andati ad individuare i principali domini di utilizzo per le tecniche di IA nell'ambito della privacy preserving per i sistemi IoT. Dai risultati ottenuti i ricercatori hanno deciso di creare un'etichetta "smart environment" per quei lavori di ricerca che erano applicabili a più tipi di domini che, in particolare, caratterizzano il 61.8% dei papers totali raccolti, i paper restanti ricadono per il 15.8% nel dominio della smart home, per il 13.2% nel dominio relativo all'healthcare, per il 5.3% nel dominio dell'industria e il restante divisi equamente tra smart city e electricity power. Dai risultati ottenuti si è riscontrata una certa versatilità dei domini applicativi, la predominanza dei lavori etichettati come smart environment è traducibile in un'alta flessibilità delle tecniche di IA ad essere applicabili in domini differenti.

2.1.3 Tipologie di algoritmi impiegati

Nella SLR è stata effettuata una classificazione delle tecniche di Machine Learning maggiormente utilizzate in letteratura. In particolar modo è stata riscontrata una forte preferenza nell'utilizzo di tecniche di Machine Learning di tipo supervisionato piuttosto che non supervisionato. Di seguito una breve spiegazione della differenza tra i due approcci:

- Machine Learning supervisionato:
 - si tratta di tecniche in cui il dataset su cui viene addestrato il modello contiene valori già etichettati per ogni input si conosce già il suo output.
- Machine Learning non supervisionato:
 - non si hanno a disposizione gli output relativi ai dati di input; il modello viene costruito solo con i dati di input e l'algoritmo scopre da solo le relazioni nascoste nei dati.

Il maggior utilizzo di tecniche di apprendimento supervisionato è dovuto alla natura dei task che tendono ad essere di classificazione o regressione. Nei rari casi in cui si è scelto di proseguire attraverso tecniche non supervisionate, queste venivano utilizzate per creare clustering e classificare traffico di rete o dispositivi, ma anche in combinazione con tecniche supervisionate. Gli algoritmi utilizzati dai ricercatori e che sono stati raccolti nella SLR sono:

- Machine Learning supervisionato:
 - Support Vector Machine.
 - Supervised Deep Learning.
 - Random Forest.
 - K-Nearest Neighbors.
 - Decision Tree.
 - Naive Bayes.
 - Logistic Regression.
- Machine Learning non supervisionato:
 - K-Means.

2.1.4 Dataset utilizzati

Dopo aver visto gli algoritmi più popolari in letteratura, andiamo a vedere quali sono i dataset che sono stati maggiormente impiegati dai ricercatori per l'addestramento dei vari modelli.

MNIST

Il dataset più impiegato, utilizzato nel 42% dei papers analizzati, risulta essere MNIST. Si tratta di una collezione di cifre scritte a mano creata appositamente per scopi di machine learning. Ad esempio Jiang et al. [13] hanno effettuato esperimenti con tecniche di offuscazione biometrica, applicate alle cifre del dataset, per valutare come le performance di un modello di deep neural network cambiano rispetto al caso in cui tale modello veniva addestrato con le cifre in chiaro.

CIFAR-10

Anche questo dataset è stato creato con lo scopo di essere utilizzato per testare tecniche di machine learning ed anche in questo caso si tratta di un dataset contenente immagini, circa 60000 categorizzate in 10 classi di circa 6000 immagini ciascuna. Questo dataset, come quello precedente, è utilizzato con lo scopo di capire se, in frammenti di immagini recuperati dai sensori, possono esserci problemi di privacy.

KDD Cup99

Trattasi di un dataset di segnali grezzi ottenuto in 9 settimane da TCP Dump. Nel dataset sono presenti 24 tipi di attacchi per il training e 14 tipi per il test. Questo tipo di dataset è utilizzato dai ricercatori per i task di attack detection in cui si vuole andare a rilevare un'intrusione.

DS2OS

Dataset contenente tracce di traffico dati ottenute in un ambiente IoT, utilizzato tipicamente per verificare algoritmi di anomaly detection. Ad esempio Hasan et al. [2] hanno sfruttato questo dataset in un task di anomaly detection per rilevare possibili attacchi su rete IoT.

Adult Dataset

Contiene informazioni sulle persone come, ad esempio, il reddito annuo. Questo dataset è in genere utilizzato dai ricercatori interessati a valutare tecniche per rilevare le perdite di dati personali.

Breast Cancer Wisconsin DataSet

Calcolato da immagini digitalizzate di FNA su massa mammaria. Il dataset contiene 569 istanze e 32 attributi (simmetria, punti concavi, area) ed è stato utilizzato per sperimentare tecniche di classificazione supervisionate che mirano a identificare potenziali perdite di dati personali.

CASIA-WebFace

Contiene 494414 immagini di volti di 10575 identità reali. Questo dataset è in genere utilizzato per la verifica del volto e le attività di identificazione del volto.

Wang et al [9] hanno effettuato esperimenti su tale dataset implementando una rete neurale profonda e un classificatore SVM per analizzare streaming video e andare ad identificare ed offuscare i volti per garantirne la privacy.

CTU-13

Dataset contenente traffico botnet acquisito sia durante traffico regolare che nel traffico in background. I ricercatori hanno utilizzato il set di dati per attività di rilevamento del malware. Bansal et al. [14] hanno impiegato più algoritmi di machine learning per il rilevamento di botnets.

Fashion MNIST

Questo dataset include le immagini degli articoli di ZALANDO e contiene circa 60.000 immagini di training e 10.000 esempi di test. È diviso in 10 classi e ogni categoria contiene circa 10.000 esempi. Gli studi principali, che hanno sfruttato questo dataset, riguardano la realizzazione di tecniche mirate a prevenire la fuoriuscita di informazioni private dovute all'identificazione delle persone dai loro abiti.

GeoLife

Contiene le traiettorie GPS raccolte in oltre 3 anni. Il dataset include informazioni sul timestamp e informazioni su latitudine, longitudine e altitudine. Tale dataset è stato utilizzato per allenare e testare tecniche per prevenire la localizzazione delle persone basandosi sulle loro coordinate.

2.1.5 Tecniche di validazione impiegate

Quando si implementa un algoritmo di machine learning è fondamentale valutare l'affidabilità per andare a capire se ci si trova in uno scenario dove i risultati che si ottengono sono buoni ad un primo occhio ma in realtà poco affidabili. Nel caso in cui un modello va in overfitting, ad esempio, si hanno buoni risultati generali dell'algoritmo sull'insieme dei dati di addestramento, tuttavia si ha che il modello non riesce a generalizzare su nuovi input o, viceversa, nel caso di underfitting si può andare a verificare che il modello non riesce ad apprendere dai dati di training poichè, probabilmente, non è impostato correttamente per riuscirci.

Per venire incontro a queste necessità di validazione dei modelli vengono utilizzate varie tecniche. Nella SLR le tecniche di validazione più frequenti rilevate sono:

- K-Fold Cross Validation:
 - Consiste nel dividere in modo casuale il dataset in k gruppi chiamati fold e ad ogni iterazione prendere un fold come testset e gli altri $k-1$ fold come training set e ripetere questo procedimento fin quando tutti i fold non sono stati utilizzati come test set. Una volta terminata la procedura, i risultati ottenuti vengono riassunti tramite indicatori statistici come, ad esempio, il numero medio di True Positive individuate durante le varie istanze di validazione.
- Random split:
 - Si va a dividere il dataset in modo casuale in percentuali di train e di test, solitamente 80% training e 20% test e si procede poi con il training del modello.
- Altro:
 - Nella SLR sono state individuate altre tecniche di validazione ma non quantitativamente rilevanti come, ad esempio, la Monte Carlo cross validation e la time-sensitive analysis.

2.1.6 Metriche di valutazione utilizzate

Una volta preparato e validato il modello è necessario valutarne le prestazioni. Per farlo si fa affidamento a diversi tipi di metriche, tra cui: accuracy, precision, recall, f1-score, ROC curve. Nel lavoro di SLR è stato riscontrato che il 23% degli studi hanno fatto affidamento esclusivamente sull'accuracy per valutare le performance dei modelli proposti, mentre in altri studi sono state utilizzate, in modo combinato, ulteriori metriche di valutazione tra quelle citate precedentemente.

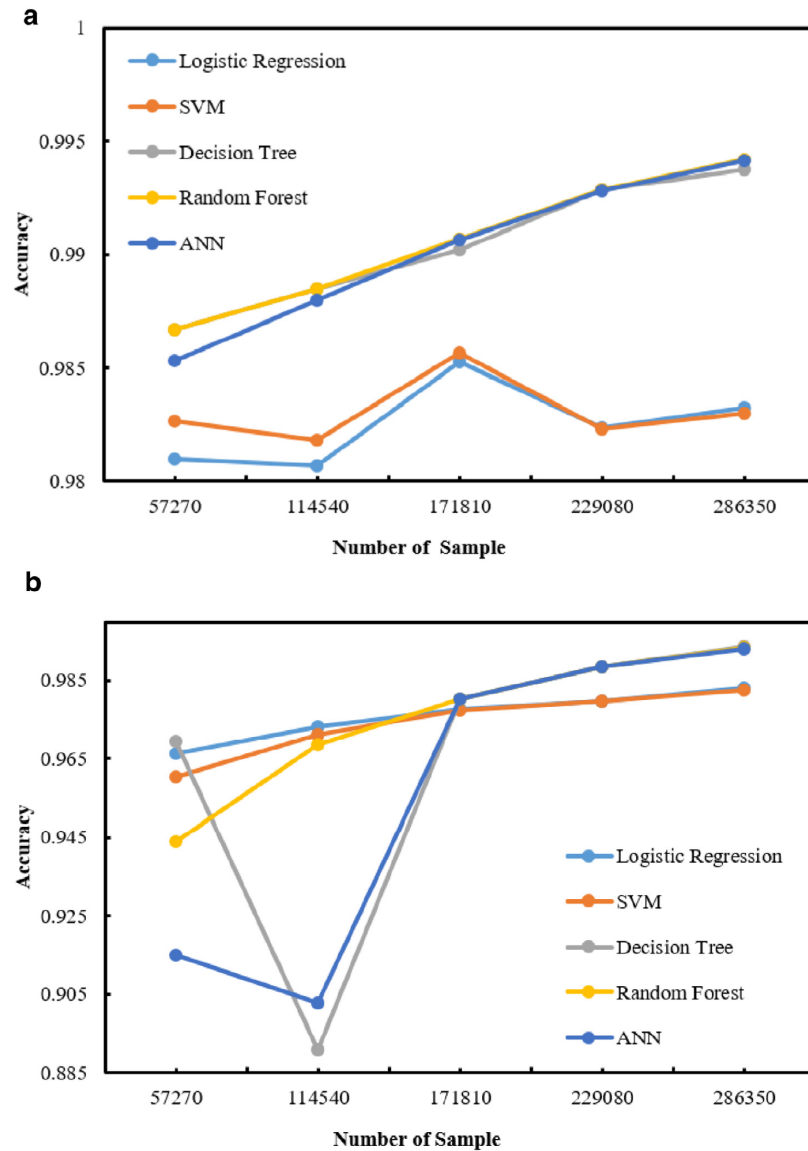
2.2 Background teorico sul task di attack detection

In questo specifico lavoro di tesi si è concentrato il focus sul task di attack detection. La scelta è stata ponderata basandosi sugli studi già effettuati in letteratura e su alcune criticità emerse che analizzeremo successivamente. Il task di attack detection, come abbiamo visto precedentemente, è il secondo task più esplorato dai ricercatori. Sono stati infatti effettuati svariati studi che, seppur simili nel contesto, differiscono tra di loro su aspetti quali: dataset utilizzati, tecniche utilizzate, approcci ecc...

Vediamo di seguito qualcuno dei lavori effettuati dai ricercatori in questo contesto ed analizzeremo, in un secondo momento, quelli che in generale sono stati gli aspetti più ricorrenti nei lavori raccolti nella SLR.

Nel lavoro svolto da Hasan et al. sono stati implementati e confrontati svariati modelli di Machine Learning supervisionato, in particolare: Regressione Logistica, SVM, Decision Tree, Random Forest, ANN. Lo studio è stato effettuato utilizzando il dataset DS2OS, visto precedentemente, per l'addestramento dei modelli. In Figura 3.8 sono riportate la training accuracy (a) e la test accuracy (b) ottenute da Hasan et al. con le varie tecniche implementate inoltre per la validazione dei modelli è stata utilizzata una 5-fold Cross Validation.

Nel lavoro svolto da Zhao et al. invece, è stata utilizzata prima la PCA, una tecnica di dimensionality reduction che va a diminuire il numero di features nei dati, dopodichè sono stati confrontati due algoritmi: regressione softmax e K-NN. Facendo esperimenti su un diverso numero di features, si è ottenuto 84.50% di accuracy per la regressione softmax e 85.24% di accuracy per K-NN. Tuttavia non sono state considerate altre metriche di valutazione. In questo caso il dataset utilizzato è KDD Cup 99, che analizzeremo più in dettaglio successivamente; le tecniche implementate, invece, sono ancora di apprendimento supervisionato.

Figura 2.2: (a) Training Accuracy (b) Test Accuracy, fonte [2]

Kohei Shiomoto [3] propone invece un approccio di apprendimento semi-supervisionato. Nel suo lavoro infatti viene considerata una GAN (generative adversarial network): si tratta di uno dei pochi lavori che non tratta apprendimento supervisionato e tecniche di shallow machine learning. Inoltre, viene utilizzato il dataset NLS-KDD ossia una versione migliorata di KDD Cup 99. Data la natura semi-supervisionata, Shiomoto, nel suo lavoro, va a sfruttare un numero ristretto di dati etichettati sfruttando poi un Adversarial autoencoder affiancato ad una GAN. Usando prettamente dati non etichettati, l'autoencoder è addestrato per estrarre 2 vettori di variabili latenti: uno per la classificazione e l'altro per la rappresentazione delle features del traffico di

rete, dopodiché la GAN è addestrata in modo da consentire a tali vettori di seguire rispettivamente una distribuzione categorica e una distribuzione Gaussiana. Successivamente, utilizzando i dati etichettati, l'autoencoder è addestrato a minimizzare il cross-entropy error. Infine il vettore delle variabili latenti per la classificazione è usato per classificare i nuovi dati in input in "traffico normale" o "attacco". In Figura 2.3 possiamo osservare i risultati ottenuti da Kohei utilizzando 1259 dati etichettati, mentre in Figura 2.4 i risultati ottenuti utilizzando solo 12 dati etichettati.

Figura 2.3: fonte [3]

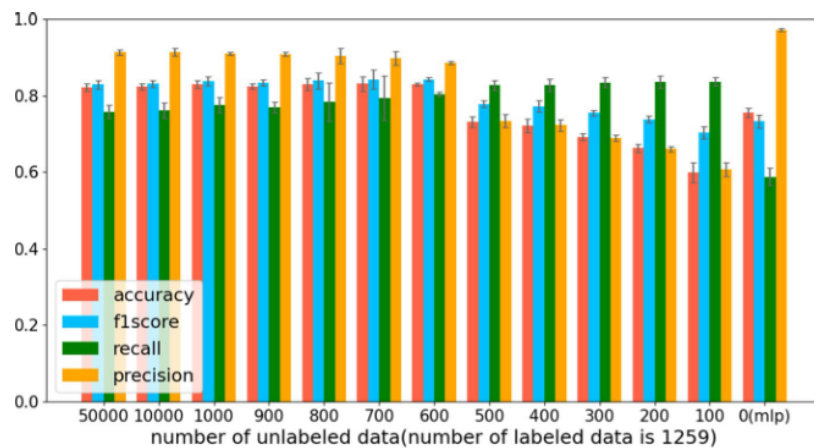
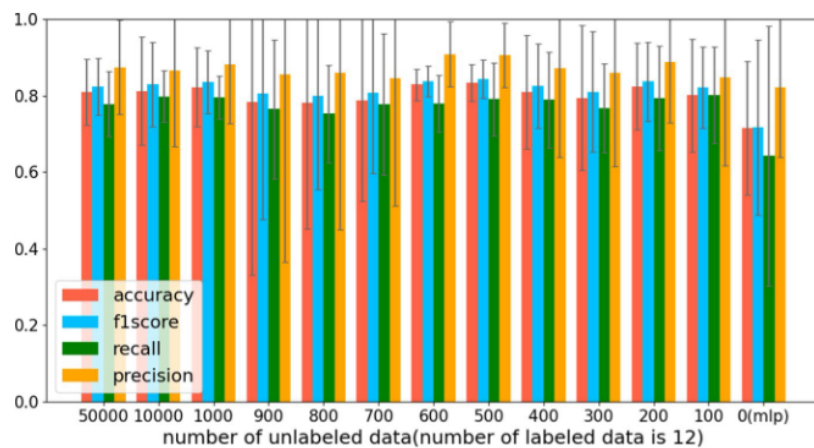


Figura 2.4: fonte [3]



Nel caso della ricerca svolta da Bansal et al. riscontriamo l'utilizzo di due dataset differenti: CTU-13 e UNB ISCX. Su questi due dataset vengono implementate e messe a confronto tre tecniche di Machine Learning differenti con lo scopo di andare a rilevare una BotNet.

Le tecniche proposte sono tre: due reti neurali, una Neural Network e una Recurrent Neural Network. In aggiunta alle reti neurali viene proposto il classificatore Gaussian Naive Bayes che va a classificare sfruttando dei clusters ottenuti mediante l'implementazione di SciKit-Learn del Gaussian Mixture Model. Dai risultati ottenuti si è riscontrato che il metodo di clustering non richiede un dataset di grandi dimensioni, tuttavia risulta avere delle performance non troppo esaltanti. La rete neurale, invece, risulta essere più performante, ma necessita di un dataset di dimensioni maggiori e di più tempo di addestramento; entrambe le tecniche richiedono una feature selection manuale. La rete neurale ricorrente, invece, richiede un dataset di dimensioni decisamente importanti, così come di tempi di training più onerosi, tuttavia, pur non necessitando di una feature selection manuale, risulta avere dei buoni risultati.

2.3 Algoritmi e dataset utilizzati per l'attack detection

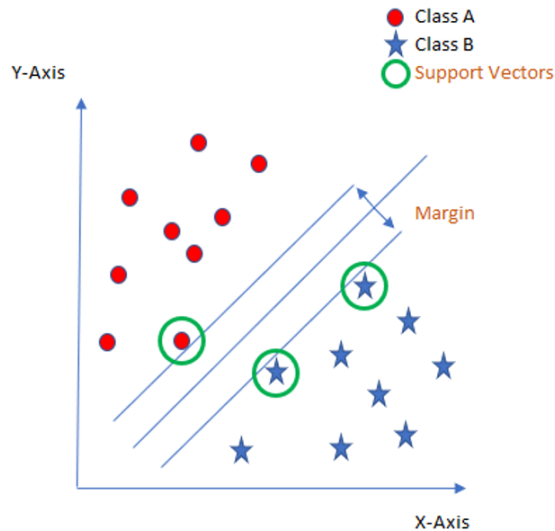
Abbiamo visto precedentemente come nella SLR siano stati individuati gli algoritmi di machine learning e i dataset più utilizzati in letteratura per il topic generico. Vedremo adesso quali sono gli algoritmi e i dataset più utilizzati in letteratura specificamente per il task di attack detection andando ad analizzarli più nello specifico.

2.3.1 Algoritmi

Le tecniche di Machine Learning supervisionato maggiormente utilizzate, per il task di attack detection, stando alla SLR, sono risultate essere: Support Vector Machine, Decision Tree, Random Forest e K-NN.

Support Vector Machine

Si tratta di un modello di Machine Learning potente e versatile. Lo scopo di SVM è quello di trovare la retta di separazione delle classi del dataset che massimizza il margine tra le classi stesse, dove con margine si intende la distanza minima dalla retta ai punti delle due classi. Maggiore è il margine, maggiore sarà la distanza tra le classi.

Figura 2.5: Illustrazione SVM, fonte [4]

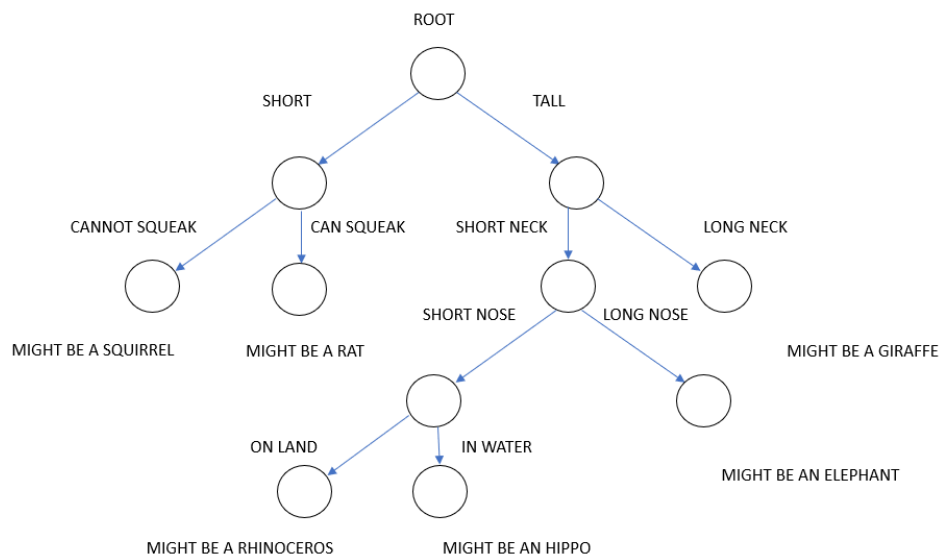
L'applicazione ideale di SVM è su dataset che sono linearmente separabili, tuttavia risulta possibile applicare SVM anche su dataset non linearmente separabili attraverso degli accorgimenti matematici sui dati.

La retta separatrice prende il nome di Decision Boundary. Se ne possono avere molteplici, lo scopo sta nel tracciare Decision Boundary che separano bene le istanze di classi diverse. Il modello viene influenzato nella classificazione dalle istanze presenti nei pressi del Decision Boundary che vanno a definire quelli che sono i support vector, ossia le istanze di classi differenti che hanno minore distanza tra di loro e che rappresentano dunque il caso più complesso. Ragionando in questi termini andiamo a creare dunque due semipiani, inoltre nel caso in cui imponiamo al modello che tutte le istanze di un semipiano devono appartenere esclusivamente ad una certa classe, parliamo di classificazione hard margin che, però, è attuabile solo su dati linearmente separabili. Introducendo una certa tolleranza alle violazioni dei margini, parliamo di classificazione soft margin.

Decision Tree

Sono modelli molto versatili che vanno ad effettuare il processo decisionale tramite un albero logico rovesciato dove ogni nodo rappresenta la condizione necessaria per compiere una decisione. Ogni nodo può essere o foglia o nodo interno: se foglia, indica il valore della classe assegnata all'istanza; se nodo interno, specifica il test effettuato su un attributo.

Figura 2.6: Illustrazione Decision Tree.



L'obiettivo è selezionare gli attributi più utili per classificare le istanze di training attraverso una strategia top down che consiste in una ricerca greedy degli attributi senza tornare a riconsiderare le precedenti scelte. Il modello continua lo splitting dei nodi fin quando non riesce ad ottenere che tutti gli elementi di un nodo appartengano alla medesima classe. Al fine di limitare l'overfitting viene effettuata una fase di pruning dove l'algoritmo va ad individuare gli attributi che non hanno contribuito allo splitting e va ad eliminare i rispettivi nodi. Al termine dell'algoritmo il percorso dalla radice alla foglia indica la classificazione di un'istanza.

Random Forest

È uno dei modelli di Machine Learning più diffusi. Fa parte della categoria dei metodi di apprendimento ensemble in cui si va ad utilizzare più modelli insieme per migliorare le prestazioni rispetto ad ognuno di loro preso singolarmente.

Nello specifico le Random Forest sono un ensemble di Decision Tree in cui ogni Decision Tree viene allenato su un sottoinsieme casuale di features e dati. Un ensemble che utilizza lo stesso tipo di modello viene chiamato bagging.

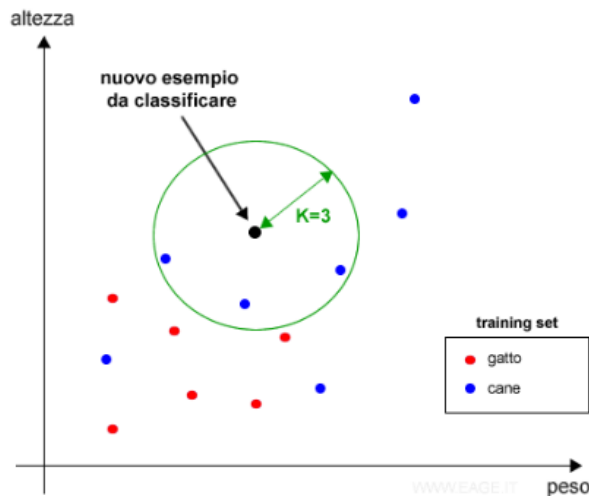
Figura 2.7: Illustrazione Random Forest, fonte [5]



Uno dei principali svantaggi degli alberi decisionali è che sono molto inclini ad overfitting cioè funzionano bene sui dati di addestramento, ma non sono così flessibili per fare previsioni su dati mai visti prima. I modelli Random Forest combinano la semplicità degli alberi decisionali con la flessibilità e la potenza di un modello ensemble. Nelle Random Forest ogni albero effettua una previsione: la previsione più votata sarà l'output della Random Forest.

K-NN

K-NN (K-Nearest Neighbors) è un algoritmo di riconoscimento dei pattern basato sulla vicinanza dei dati. La logica utilizzata da K-NN ci dice che: se un oggetto A ha caratteristiche simili a un oggetto B, probabilmente A appartiene alla stessa classe di B. Si tratta di un algoritmo di apprendimento supervisionato perché necessita di un gruppo di esempi di training già classificati. Quando l'algoritmo analizza i nuovi dati li classifica in base alla distanza rispetto agli esempi che ha a disposizione, valutando la distanza da essi.

Figura 2.8: Illustrazione K-NN, fonte [6]

Per il calcolo della distanza si possono adottare diversi criteri: la distanza euclidea o la distanza Manhattan per i dati numerici, distanza di Hamming per le stringhe e così via. Nell'algoritmo K-NN la configurazione del parametro K è fondamentale e sta ad indicare il numero di vicini che deve andare a considerare. Se K è troppo piccolo, il nuovo valore è assegnato in base ai pochi esempi più vicini. La classificazione è influenzata dal rumore nei dati. Se K è troppo grande, il nuovo valore viene classificato nella categoria più numerosa nella popolazione.

2.3.2 Dataset

Nell'ambito dell'attack detection, i dataset utilizzati, per la maggior parte delle ricerche, sono stati: KDD Cup 99 e DS2OS.

KDD Cup 99

Si tratta del dataset utilizzato per la terza International Knowledge Discovery and Data Mining Tools Competition, la cui sfida consisteva nel costruire un rilevatore di intrusioni di rete, un modello in grado di distinguere tra "bad connections", chiamate attacchi, e "normal connection". Tavallaee et al. [15], ci propongono un'analisi molto dettagliata su tutto il dataset KDD Cup 99. Dall'analisi effettuata riscontriamo alcuni dettagli chiave: il training set contiene circa 4,900,000 record di traffico dati, ciascuno

avente 41 features ed etichettato come “traffico normale” o “attacco”, dove nel caso di attacco viene indicato lo specifico tipo di attacco. In totale abbiamo 24 differenti tipi di attacco raggruppabili in 4 macro categorie:

- Denial of Service (DDoS):
 - è un attacco in cui l'attaccante va a sovraccaricare le risorse di un sistema negando agli utenti legittimi la possibilità di inviare ulteriori richieste.
- User to Root (U2R):
 - è una classe di exploit in cui l'attaccante inizia accedendo con un normale account utente sul sistema ed è in grado di sfruttare alcune vulnerabilità per ottenere l'accesso root al sistema.
- Remote to Local (R2L):
 - si verifica quando un utente malintenzionato, che ha la capacità di inviare pacchetti alla macchina su una rete, ma che non ha un account su quella macchina, sfrutta alcune vulnerabilità per ottenere l'accesso locale come utente di quella macchina.
- Probing attack:
 - è un tentativo di raccogliere informazioni su una rete di computer con lo scopo di aggirarne i controlli di sicurezza.

Una caratteristica importante di KDD Cup 99 risiede nel test set. Questo, infatti, contiene 14 tipi di attacco aggiuntivi ai 24 presenti nel training set per rendere il task di detection più realistico, basandosi sul concetto secondo cui nuovi attacchi sono solo la variante di attacchi già noti. Dunque quello che si conosce dal training set dovrebbe essere già sufficiente per la detection di nuovi attacchi. KDD Cup 99 ha tuttavia delle problematiche di una certa rilevanza:

- Distribuzione dei dati:
 - la distribuzione degli attacchi nel dataset risulta essere fortemente sbilanciata, rendendo la cross validation molto difficile da realizzare, andando a creare sottoinsiemi che contengono soltanto una specifica istanza di attacco. La presenza di un'elevata quantità di attacchi DDoS che risultano essere circa il 71% del dataset totale, va ad influenzare completamente le stime di valutazione sulla bontà di un eventuale modello.
- Ridondanza dei record:
 - uno dei problemi più gravi di KDD Cup 99 si ha nella forte ridondanza dei record che porta gli algoritmi di apprendimento ad avere un bias sui record più frequenti, causando la mancanza di un apprendimento adeguato su record meno frequenti, ma con maggior importanza sulla rete come gli attacchi U2R e R2L. L'esistenza di record ripetuti anche nel test set causa problemi nella valutazione, andando a premiare quei modelli che hanno un miglior detection rate sui record più frequenti.

DS2OS

Trattasi di un dataset creato appositamente per fornire tracce pubbliche di traffico dati generato da dispositivi IoT utilizzabili in ambito di ricerca. Il dataset contiene un totale di 357952 campioni con 10017 valori anomali (attacchi) e 347935 normali. Contiene 13 features e 7 diversi tipi di attacchi come Denial of Service, Malicious Operation, Malicious Control, Wrong Setup, Spying, Scan, e Probing. Per completezza descriviamo gli attacchi che non sono già stati analizzati nel punto precedente:

- Malicious Operation:
 - Questi attacchi sono generalmente causati da software dannoso. Il malware consiste in operazioni esca che interferiscono con le operazioni originali eseguite dal dispositivo. Questo genere di attacco può influire negativamente sulle prestazioni del dispositivo.

- Malicious Control:
 - Questo attacco fornisce l'accesso non autorizzato ai dispositivi di sistema o alle reti. Questo attacco viene lanciato con l'intento dannoso di accedere a informazioni riservate.
- Wrong Setup:
 - In questo attacco, un intruso può accedere alle informazioni riservate sui clienti o sull'azienda sfruttando l'errata configurazione del sistema.
- Spying:
 - Nello spying, un utente malintenzionato può ottenere l'accesso alle informazioni riservate attraverso un canale backdoor sfruttando le vulnerabilità del sistema.
- Scan:
 - L'attaccante esegue la scansione dei dispositivi di rete e raccoglie le informazioni degli indirizzi IP del client e degli indirizzi delle porte del server.

Anche DS2OS risulta essere sbilanciato.

2.4 Riflessioni sulle Limitazioni dello Stato dell'Arte

Da un occhio più analitico sugli studi effettuati sono emerse alcune criticità evidenziate anche nella SLR, sebbene in molti studi si vanno a completare mancanze evidenziate in altri, ci sono alcuni aspetti che sono risultati più frequenti:

- Dataset sbilanciati:
 - Entrambi i dataset visti precedentemente, nonostante siano i più utilizzati in letteratura per il task di attack detection, risultano essere fortemente sbilanciati ossia il numero di campioni per classe è decisamente differente tra le classi stesse. Un dataset sbilanciato può portare a problemi durante

la fase di addestramento e di sperimentazione, andando a generare dei risultati che possono essere poco attendibili.

- Focus su tecniche specifiche:
 - La quasi totalità dei lavori effettuati presenta l'utilizzo prettamente di tecniche di **shallow machine learning**, solo in pochi lavori viene considerata l'adozione di tecniche di deep learning. Per shallow machine learning intendiamo quella categoria di tecniche di apprendimento automatico che si basa sull'utilizzo di modelli di apprendimento che non hanno una struttura interna complessa. Questi modelli di apprendimento automatico più semplici includono algoritmi di regressione lineare, alberi decisionali, SVM (Support Vector Machines), Naive Bayes, Random Forest e altri ancora.
- Tecniche di Validazione:
 - L'utilizzo massiccio, nella quasi totalità dei lavori presenti in letteratura, della k-fold cross validation come tecnica di validazione dei modelli può risultare un punto critico. L'impiego della k-fold cross validation su dataset sbilanciati può portare a dei risultati che sono poco veritieri poiché, in ogni fold del dataset, la distribuzione delle classi potrebbe essere molto diversa dalla distribuzione reale del dataset completo. In particolare, la presenza di classi di minoranza nei fold di addestramento può portare ad un addestramento inadeguato del modello su queste classi, andando a generare delle metriche con valori molto alti ma poco affidabili.
- Metriche di Valutazione:
 - In molti lavori i ricercatori si sono basati esclusivamente sull'accuracy come metrica di valutazione dei risultati, tuttavia spesso la sola accuracy non è un buon metro di giudizio se non contestualizzata insieme ad altre metriche quali, precision, recall ed F-score.

- Pipelining:
 - Altra criticità risiede nella gestione della pipeline di ML, in vari studi infatti non viene detto in modo esplicito se e come è stata effettuata una fase di preprocessing e configuration del dataset, soprattutto nel caso di dataset sbilanciati.

CAPITOLO 3

Metodologia proposta

In questo terzo capitolo viene mostrato in che modo è stato affrontato l'obiettivo, quindi quali sono le research question che ci siamo posti, tutti gli accorgimenti e le mitigazioni adottate in merito alle criticità discusse e il setting degli algoritmi di ML implementati. Nello specifico vedremo: quali sono gli step di pre processing effettuati sul dataset, la motivazione alla base della scelta del dataset stesso, come sono stati impostati gli iper parametri degli algoritmi e le varie tecniche implementate.

L'obiettivo dello studio empirico è andare ad effettuare un confronto delle prestazioni ottenute tra differenti tecniche di Machine Learning impiegate sul task di attack detection, tenendo un occhio sulle criticità analizzate precedentemente e cercando, dove possibile, di eliminarle.

Al netto delle criticità emerse, definiamo alcune research question:

- RQ1. I dataset adottati in letteratura sono sbilanciati. È possibile adottare un dataset differente? È possibile adottare tecniche in pipelining per sopperire allo sbilanciamento dei dataset già visti?
- RQ2. Possiamo adottare anche tecniche che non siano di shallow machine learning?
- RQ3. Per la validazione dei modelli quali differenze ci sono utilizzando diverse tecniche di validazione?

RQ4. Oltre all'accuracy, quali insights è possibile ottenere utilizzando altre metriche di valutazione?

3.1 Dataset impiegato e pre-processing

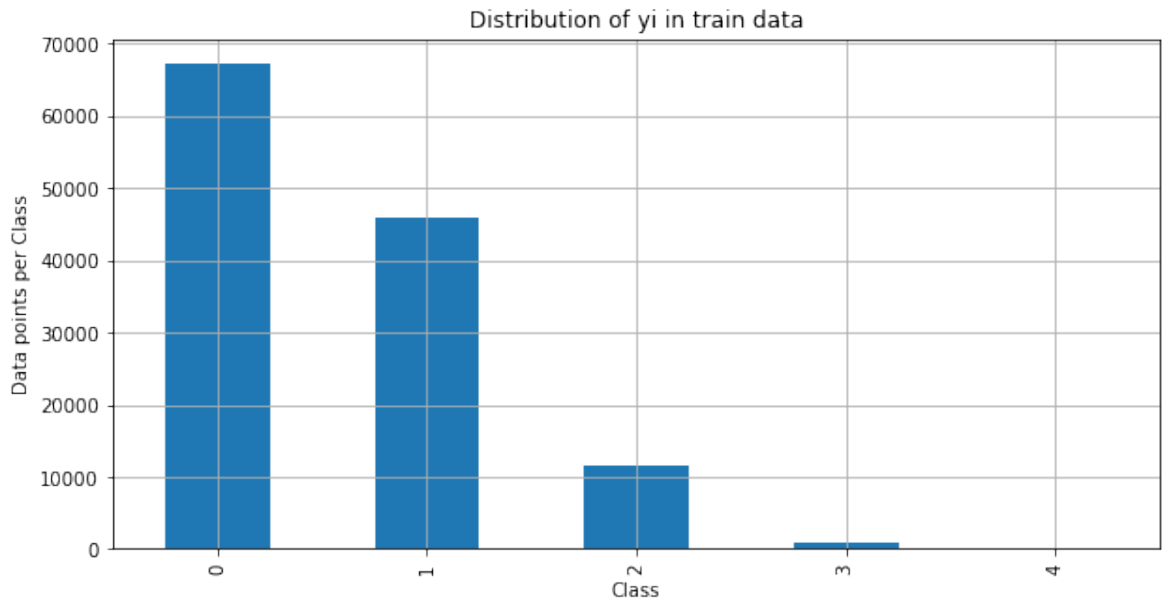
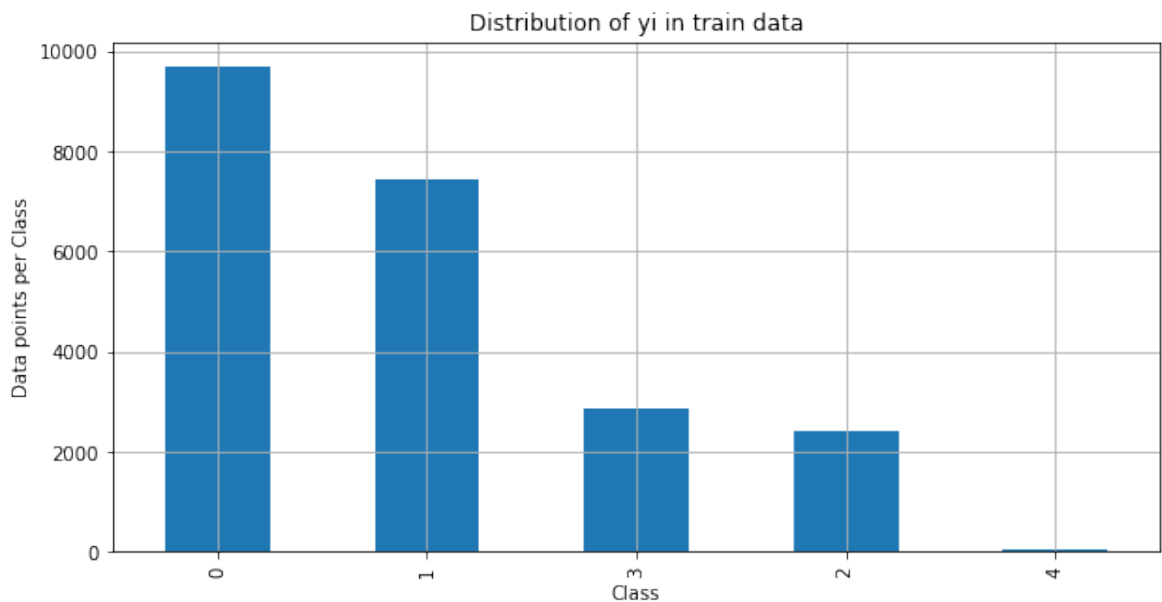
Per il task di attack detection abbiamo visto dalla SLR che i dataset KDD Cup 99 e DS2OS sono i dataset più utilizzati. Sfortunatamente in letteratura non sono disponibili molte alternative valide per l'addestramento dei modelli, tuttavia, i ricercatori del Canadian Institute for Cybersecurity hanno costruito il dataset NLS-KDD [16] esso è un sottoinsieme del dataset KDD Cup 99 creato con lo scopo di eliminarne le criticità. NLS-KDD, pur essendo a sua volta sbilanciato, risulta essere la versione migliore di KDD Cup 99. Infatti, uno dei problemi principali di KDD Cup 99, risiede nella forte ridondanza dei record, mentre con NLS-KDD è stata effettuata una pulizia sui records stessi andando a rimuovere tutte le ridondanze.

3.1.1 Analisi NLS-KDD

Il dataset NLS-KDD è diviso in training set e test set: nel training set sono presenti 125973 records, di cui 67343 sono di traffico normale; i restanti rappresentano 22 tipologie di attacco raggruppabili nelle medesime quattro macro categorie di attacco viste in KDD Cup 99: DDos, Probe, R2L e U2R. Nel test set sono invece presenti 22544 records di cui 9711 sono di traffico normale, i restanti rappresentano 37 tipologie di attacco raggruppabili nelle solite quattro macro categorie. Nel test set ci sono 15 tipologie di attacchi non presenti nel training set con lo scopo di fornire ai modelli maggiore eterogeneità di dati non visti nel training set.

3.1.2 Pre-processing

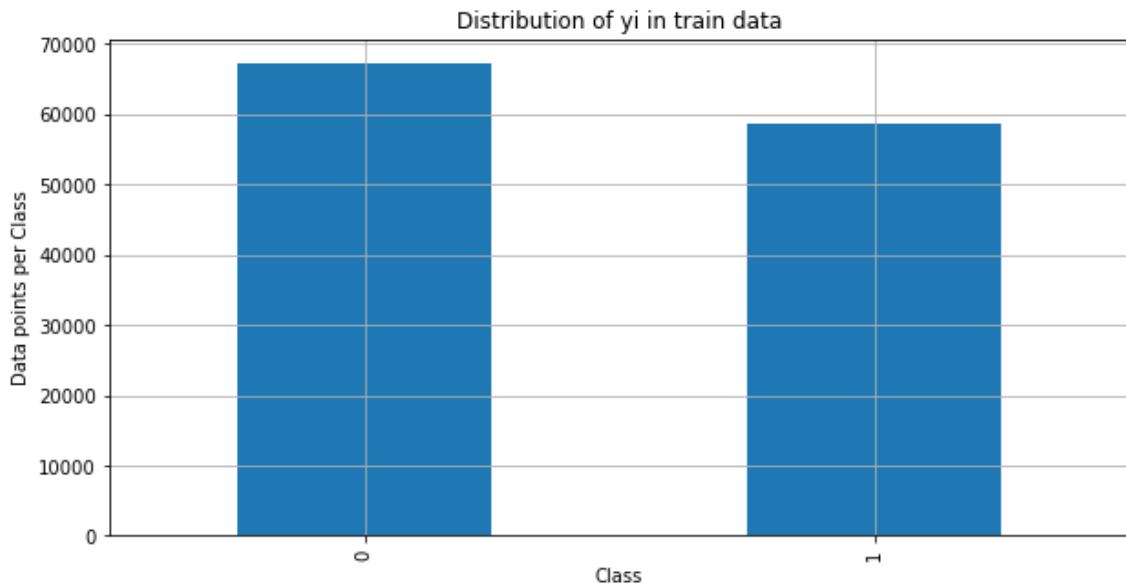
La prima operazione effettuata sul dataset è stata raggruppare tutte le diverse tipologie di attacco nelle quattro macro categorie, etichettando con 1 la categoria DDos, 2 la categoria Probe, 3 la categoria R2L, 4 la categoria U2R; i records che invece rappresentano traffico normale, sono stati etichettati con 0.

Figura 3.1: Rappresentazione distribuzione Training set.**Figura 3.2:** Rappresentazione distribuzione Test set.

Oltre a questo raggruppamento di tipo multiclasse, ai fini della sperimentazione, è stato effettuato anche un raggruppamento binario sia sul training set che sul test set, dove, per ogni classe di attacco, è stato creato un subdataset contenente tutti i records di quella classe di attacco più tutti i records del traffico normale; inoltre è stato creato anche un dataset binario in cui i records sono stati etichettati con 0 nel

caso di traffico normale, con 1 in caso di traffico contenente un attacco, senza fare distinzione sulla tipologia di attacco.

Figura 3.3: Rappresentazione distribuzione Training set binario.



Ogni elemento del dataset è composto da 42 features, di queste features tre sono di tipo categorico:

- protocol_type;
- service;
- flag;


Ci sono modelli che non possono lavorare con attributi di tipo categorico, ma necessitano che tutti i dati in input e in output siano di tipo numerico. Le soluzioni percorribili sono due:

- rimuovere questi attributi categorici dai dati;
- utilizzare tecniche di encoding per trasformare gli attributi categorici in attributi numerici;

Durante la fase di sperimentazione sono state eseguite entrambe le soluzioni effettuando dei test sia sul dataset **senza attributi categorici**, sia sul dataset con gli attributi categorici trasformati utilizzando il **OneHotEncoding**.

Il OneHotEncoding è una tecnica mediante la quale si introduce un attributo binario per ognuna delle variabili dell'attributo categorico.

Figura 3.4: OneHotEncoding.



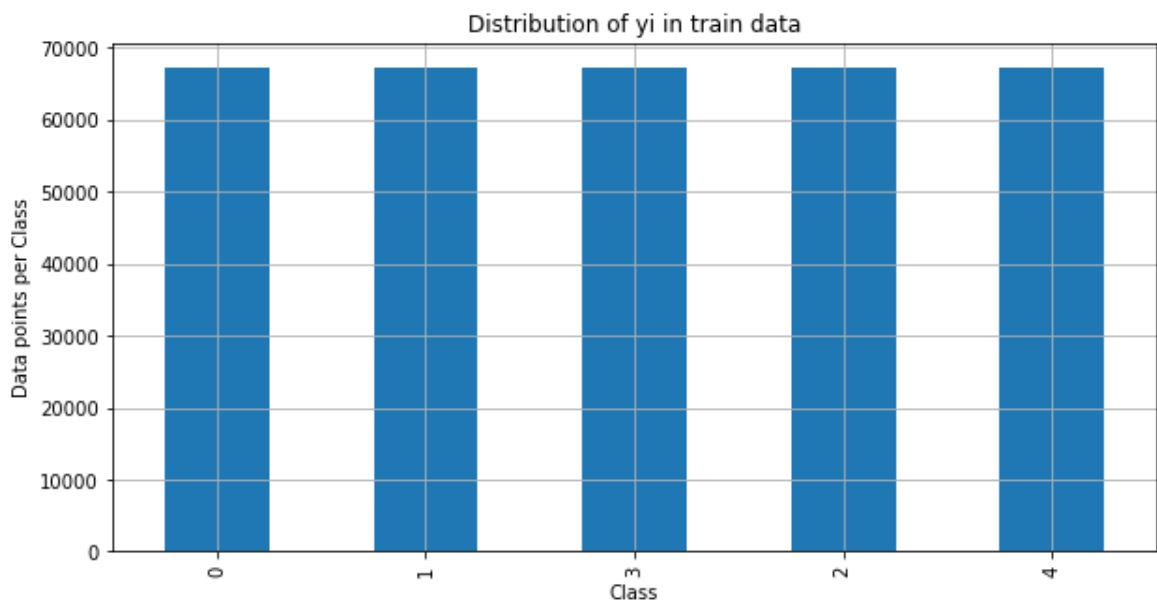
Colore		Rosso	Giallo	Verde
rosso		1	0	0
rosso		1	0	0
giallo		0	1	0
verde		0	0	1
giallo		0	1	0

Dopo aver applicato OneHotEncoding, il dataset finale è risultato essere composto da **123 features** (un numero di features elevato potrebbe generare problemi ai modelli nell'apprendimento, noto in letteratura come Curse of Dimensionality). È stata effettuata una feature selection per andare a scremare quello che è il numero totale di features in modo da poter effettuare dei test sia sul dataset originale che su un dataset contenente soltanto un sottoinsieme delle features originali. Per la **feature selection** è stato utilizzato l'algoritmo **RFE** (Recursive Feature Elimination) implementato attraverso la libreria scikit-learn. RFE per essere applicato ha bisogno che gli venga passato in input il numero di features che si vuole preservare ed un modello da addestrare; dopodichè lavora iterativamente per rimuovere le feature meno importanti dai dati addestrando il modello su un sottoinsieme di feature in ogni iterazione e valutando la loro importanza. Il processo inizia con un set completo di feature e un modello addestrato su di esse. Successivamente, vengono calcolate le importanze delle feature utilizzando un criterio specifico (ad esempio, l'accuratezza del modello o il coefficiente di correlazione) e viene rimossa la feature meno importante. Il modello viene quindi addestrato di nuovo sul set ridotto di feature e il processo di selezione delle feature viene ripetuto fino a quando non viene raggiunto un numero prefissato di feature. Nel nostro caso sono state selezionate 13 features mediante l'algoritmo Random Forest.

La fase successiva è stata quella di **oversampling**. Mediante l'oversampling si vanno a ribilanciare quelle classi del dataset che sono più sbilanciate rispetto alla classe che ha più samples, detta classe dominante. La tecnica di oversample utilizzata è stata SMOTE (Synthetic Minority Oversampling Technique) ed è una delle tecniche di oversampling più popolari: si vanno a prendere le osservazioni più vicine (distanza

euclidea) tra quelle della classe minoritaria, si fa la differenza tra i due vettori di features e si moltiplica questo valore per un numero casuale tra 0 e 1. In altre parole, si va ad applicare un perturbamento alla distanza tra due punti della classe minoritaria. Così facendo si creano osservazioni artificiali che accrescono il patrimonio di dati ma non ne modificano troppo il valore. SMOTE è stata applicata sia sul training set che sul test set. Nella sperimentazione effettuata sul dataset binario, non è stata applicata in quanto il dataset non è sbilanciato.

Figura 3.5: Training set dopo aver applicato SMOTE.



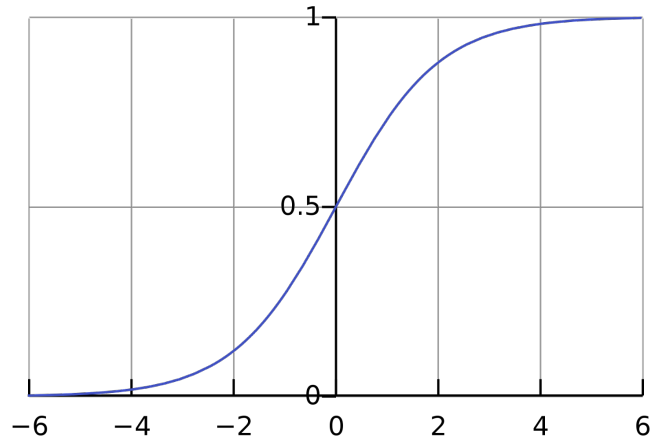
In fase di sperimentazione, sono stati effettuati dei test sia con il dataset sbilanciato sia con il dataset dopo aver applicato SMOTE. Dopo aver eseguito l'oversampling sui dati è stata effettuato un ulteriore step di pre-processing: la **standardizzazione**. Per standardizzazione si intende quel processo statistico di manipolazione dei dati che, nel caso di un dataset, modifica i valori di una o più features affinché abbiano le proprietà di una distribuzione gaussiana con media uguale a 0 e deviazione standard uguale a 1. La standardizzazione dei dati è stata effettuata utilizzando il metodo **StandardScaler** del pacchetto **preprocessing** appartenente alla libreria **scikit-learn**. Come per gli steps precedenti, anche in questo caso per la sperimentazione è stato considerato sia il dataset originale sia il dataset dopo l'applicazione di StandardScaler.

3.2 Algoritmi di ML utilizzati

Abbiamo visto nel capitolo sullo stato dell'arte come gli algoritmi di ML più utilizzati in letteratura per il task di attack detection sono stati: Support Vector Machine, Decision Tree, Random Forest e K-NN. Ai fini del nostro esperimento sono stati utilizzati i medesimi algoritmi, con l'aggiunta di **Logistic Regression** e, con lo scopo di testare tecniche differenti da quelle shallow machine learning, anche **Deep Neural Network**. Come per gli algoritmi già visti andiamo a spiegare brevemente cosa sono e come funzionano sia Logistic Regression che Deep Neural Network.

3.2.1 Logistic Regression

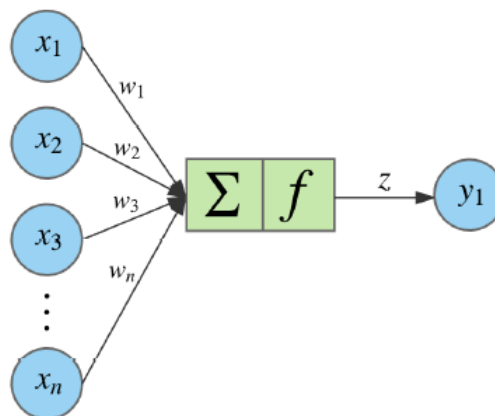
La Logistic Regression è un modello statistico utilizzato per la classificazione binaria. Questo modello è basato su una funzione logistica che può assumere valori tra 0 e 1. La funzione logistica è utilizzata per calcolare la probabilità che una determinata osservazione appartenga ad una delle due classi. Il processo di Logistic Regression comporta la selezione di un insieme di variabili indipendenti che influenzano la variabile dipendente binaria. Tali variabili possono essere continue, discrete o categoriche. Logistic Regression utilizza un algoritmo di ottimizzazione per trovare i coefficienti ottimali per ciascuna variabile indipendente, in modo da massimizzare la verosimiglianza dei dati. Una volta che sono stati trovati i coefficienti, la Logistic Regression può essere utilizzata per classificare le nuove osservazioni. La probabilità che un'osservazione appartenga ad una determinata classe viene calcolata utilizzando la funzione logistica; se la probabilità supera una certa soglia, l'osservazione viene assegnata alla classe corrispondente.

Figura 3.6: Logistic Regression

3.2.2 Artificial Neural Network

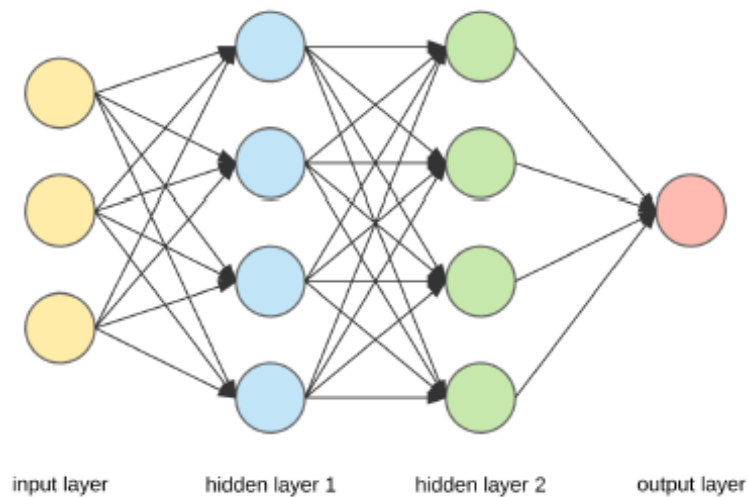
Le reti neurali artificiali (ANN) sono sistemi informatici ideati dall'osservazione del cervello umano. Il cervello, infatti, è formato da neuroni che si scambiano segnali per comunicare tra di loro. Allo stesso modo, una ANN si compone di nodi interconnessi. Ad ogni connessione è associato un valore numerico chiamato peso. Quando un nodo riceve l'input effettua una somma pesata, ovvero moltiplica ogni input per il peso della connessione su cui lo ha ricevuto, infine, somma tutti i risultati. Al risultato di questa somma applica una funzione non lineare e restituisce un output che rappresenta la predizione effettuata.

Figura 3.7: Immagine che mostra come i neuroni X_n generano un risultato Y_n utilizzando i pesi W_n .



I neuroni sono disposti su più strati connessi tra loro. Possiamo distinguere tre tipologie di strati: abbiamo uno strato di input e uno di output rispettivamente posti all'inizio e alla fine, poi ci sono quelli che sono definiti strati nascosti, ossia gli strati in cui viene effettuata la computazione descritta in precedenza. Se gli strati nascosti sono almeno due allora parliamo di **Deep Neural Network**. Una rete neurale artificiale viene addestrata usando un algoritmo chiamato backpropagation. Questo algoritmo cerca di minimizzare una funzione di perdita in modo iterativo andando a calcolare quali sono i valori ottimali dei pesi delle connessioni, ovvero quali valori producono una predizione migliore. Quindi la logica dell'algoritmo funziona andando in avanti fino alla fine e ottenere una predizione, a questo punto la rete calcola l'errore dell'output utilizzando una funzione di perdita per paragonare l'output giusto rispetto a quello ottenuto e restituisce la misura dell'errore. Data la misura di errore inizia a tornare indietro calcolando per ogni strato il contributo all'errore. Quello che fa l'algoritmo è far variare i pesi in modo da andare a minimizzare la funzione di perdita.

Figura 3.8: Deep Neural Network.



3.3 Tecniche di validazione utilizzate

Come già anticipato nelle criticità emerse, l'utilizzo della sola k-fold cross validation come tecnica di validazione potrebbe non essere appropriato. Per questo motivo, durante la sperimentazione effettuata, alla k-fold cross validation è stata affiancata la **stratified cross validation** che garantisce che la distribuzione delle classi nei fold di addestramento sia simile a quella del dataset completo. La stratified cross validation è stata affiancata alla k-fold cross validation sia sul dataset originale, sia sul dataset con oversampling.

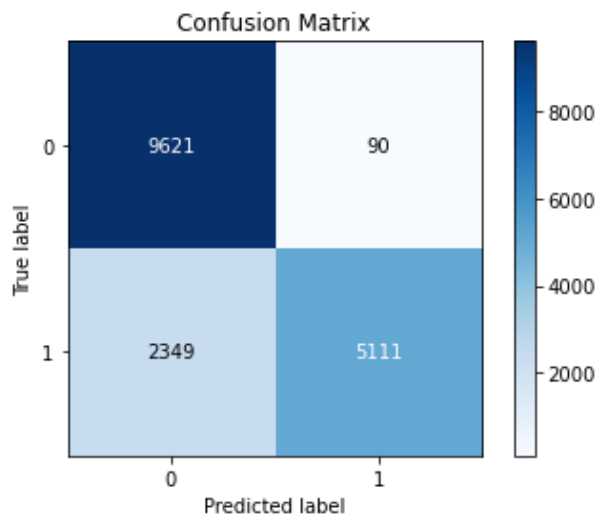
3.4 Metriche di valutazione adottate

La sola accuracy utilizzata nella maggior parte degli studi raccolti nella SLR potrebbe essere poco significativa; ad esempio nel caso di dataset sbilanciato e di k-fold cross validation come tecnica di validazione, questa metrica potrebbe essere non veritiera se non coadiuvata da altre metriche a supporto. Nella sperimentazione effettuata, oltre all'accuracy, sono state adottate altre tre metriche di valutazione:

- Precision:
 - è l'abilità di un classificatore di non etichettare un'istanza positiva che è in realtà negativa. Per ogni classe è definito come il rapporto tra True Positive e la somma di True Positive e False Positive.
- Recall:
 - è la capacità di un classificatore di trovare tutte le istanze positive. Per ogni classe è definito come il rapporto tra i True Positive e la somma dei True Positive e dei False Negative.
- F-measure:
 - è una media armonica ponderata delle metriche Precision e Recall in modo tale che il punteggio migliore sia 1 e il peggiore sia 0.

Per ogni risultato ottenuto è stata inoltre prodotta la matrice di confusione che aiuta a vedere il numero delle previsioni giuste e quelle sbagliate. La matrice di confusione è utile soprattutto per capire in che modo il modello sbaglia la classificazione.

Figura 3.9: Matrice di confusione.



Nella Figura 3.9 possiamo osservare un esempio di matrice di confusione estratto dalla sperimentazione: nella matrice ogni riga rappresenta le classi reali, ogni colonna rappresenta invece le classi predette dal modello. I valori nella tabella mostrano il numero di esempi per ogni combinazione di classe effettiva e classe predetta.

3.5 Configurazione degli iperparametri

In questa sezione andiamo a mostrare quali sono le configurazioni degli iperparametri adottate per le varie tecniche e per i vari algoritmi implementati.

3.5.1 Training - Validation split e Folds utilizzati

Per quanto riguarda le tecniche di validazione, il training set è stato splittato in training e validation set. È una tecnica molto nota in letteratura che serve a migliorare la capacità dei modelli nel generalizzare su nuovi dati; in particolare è stata effettuata una **suddivisione 80-20**, ossia l'80% del training set è stato utilizzato per il training del modello, mentre il restante 20% è stato destinato alla validazione. Da notare

che il validation set non è l'equivalente del test set, in quanto il primo è costruito artificialmente a partire dai dati di training; il secondo è costruito appositamente per contenere dati di test. Per quanto riguarda il numero di split impiegati è stato di **10 folds**, sia per la k-fold cv che per la stratified cv.

3.5.2 RFE, Oversamplig e StandardScaling

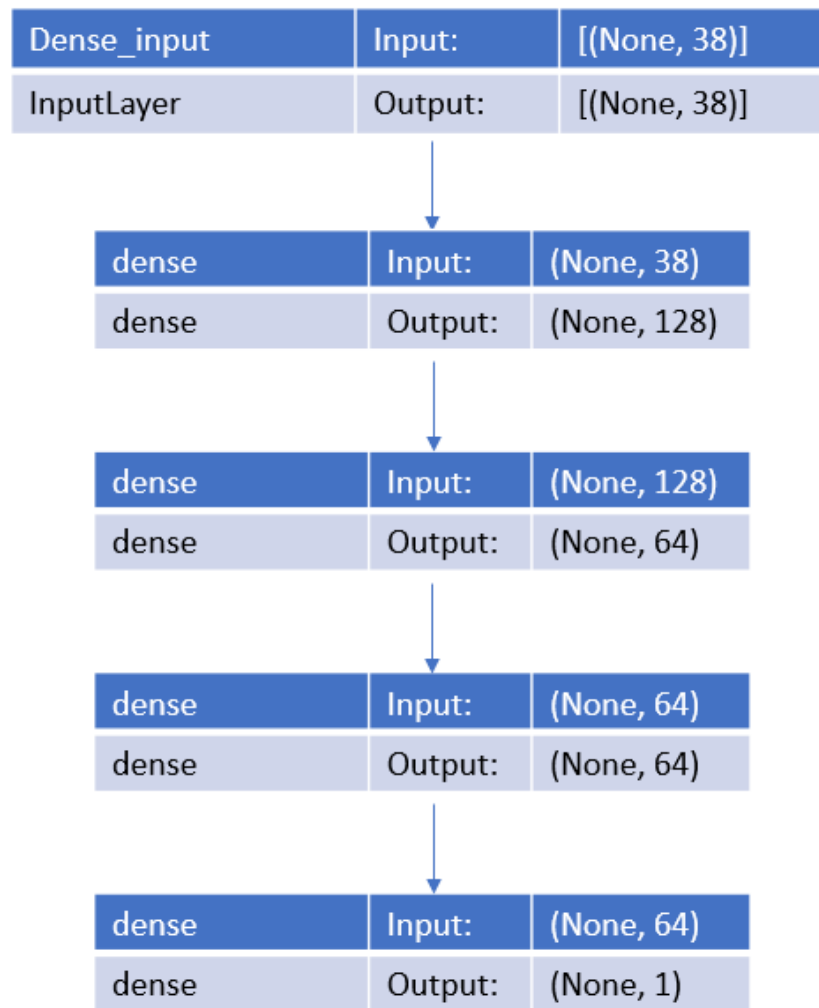
La selezione delle features come già anticipato è stata effettuata adottando l'algoritmo RFE mediante la libreria scikit-learn; come modello per la selezione è stato utilizzato RandomForest settato con: `n_estimators=64` e `n_jobs=2`. Il numero di features selezionate è stato di 13 ed è stato scelto questo numero di features poiché in linea con quello riscontrato in diversi paper sull'argomento. Per l'oversampling dei dati e la configurazione di SMOTE, impiegata attraverso la libreria **imbalanced learn**, non sono stati settati particolari parametri se non un intero come `random_state` per andare a controllare la randomizzazione dell'algoritmo, impostato a 42; gli altri parametri sono stati lasciati di default. Anche StandardScaler è stato implementato lasciando i parametri di default.

3.5.3 Iperparametri algoritmi di ML

Andremo a vedere di seguito, per ogni algoritmo di ML impiegato, come sono stati impostati, dove fatto, gli iperparametri del modello.

- Support Vector Machine:
 - SVM è stato implementato attraverso **scikit-learn.svm**, in particolare è stato utilizzato il modulo **LinearSVC**, gli iperparametri configurati sono stati: **random_state=1** per la riproducibilità; **tol=1e-5** per quanto riguarda lo stopping criteria; **dual=False** in quanto per il nostro problema `n_samples > n_features`, gli altri parametri sono stati lasciati di default.

- Logistic Regression:
 - implementato attraverso **scikit-learn.linear_model** con il modulo **LogisticRegression**. L'unico parametro settato è stato **max_iter=10000** per la convergenza dei solvers, gli altri parametri sono stati lasciati di default.
- KNN:
 - KNN è stato implementato mediante **scikit-learn.neighbors** con il modulo **KNeighborsClassifier**. Tutti i parametri in questo caso sono stati lasciati di default.
- DecisionTree:
 - L'implementazione di DT è stata realizzata mediante **scikit-learn.tree**, utilizzando il modulo **DecisionTreeClassifier**. Gli iperparametri assegnati sono stati: **criterion=gini** per l'impurità dei nodi; **max_depth=3** come profondità dell'albero; **min_samples_leaf=32** per il criterio di decisione sui nodi foglia. Gli altri parametri sono stati lasciati di default.
- RandomForest:
 - Per l'implementazione di RandomForest è stata usata la libreria **scikit-learn.ensemble**; in particolare il modulo **RandomForestClassifier**. Sono stati configurati due iperparametri: **n_estimators=64** e **n_jobs=2**. Il primo per impostare il numero di alberi nella random forest e il secondo per parallelizzare la computazione, gli altri sono stati lasciati come di default.
- Neural Network:
 - La struttura della Neural Network implementata è costituita da un layer di input, due hidden layers e un layer di output. È stata applicata la funzione ReLu per quanto riguarda l'input layer e gli hidden layers e la funzione Sigmoid per quanto riguarda il layer di output e, dunque, la classificazione finale. SoftMax e ReLu sono le funzioni più comunemente utilizzate e che garantiscono migliori risultati. Una rappresentazione della struttura della rete è visibile in Figura 3.10.

Figura 3.10: Struttura Neural Network implementata.

CAPITOLO 4

Risultati

In questo capitolo vengono mostrati i risultati più significativi ottenuti per ogni test effettuato sui vari modelli, andando a commentare eventuali differenze. In particolare la risposta alla RQ1 è presentata nella section 4.2 in cui vengono mostrati i risultati ottenuti dopo aver applicato l'oversampling. La RQ3 viene risposta nella section 4.1 e 4.2 in cui si vanno a considerare anche le performance delle tecniche di validazione dei modelli. Le RQ2 e RQ4 trovano risposta in ognuna delle sezioni di sperimentazione, in quanto in ogni fase sono state sperimentate tecniche di deep learning e in ogni fase sono state adottate più metriche per la valutazione dei modelli.

4.1 Risultati ottenuti senza oversampling e con drop degli attributi categorici

In prima battuta i modelli sono stati allenati sul dataset senza oversampling e facendo il drop degli attributi categorici. Il dataset risultante ha la struttura visibile in Figura 4.1, dove ricordiamo che nel training set il traffico di rete normale è rappresentato da 67343 samples, mentre nel test set da 9711 samples il totale delle features è 38 (escludendo le etichette). Sostanzialmente per ogni classe di attacco è stato costruito un sub dataset contenente il traffico normale e il traffico di quella determinata classe di attacco, in modo da vedere come performa il modello su ogni tipologia d'attacco.

Ogni test è stato effettuato sia sui dati originali che sui dati standardizzati con StandardScaler. Per ogni tecnica utilizzata discuteremo dei risultati ottenuti su training set, validation set e test set.

Figura 4.1: Distribuzione dataset senza oversampling.

```
Train:
Dimensions of DoS: (113270, 39)
Dimensions of Probe: (78999, 39)
Dimensions of R2L: (68338, 39)
Dimensions of U2R: (67395, 39)

Test:
Dimensions of DoS: (17171, 39)
Dimensions of Probe: (12132, 39)
Dimensions of R2L: (12596, 39)
Dimensions of U2R: (9778, 39)
```

Per ogni algoritmo implementato, andremo a vedere le performance ottenute prima sul set di features originali e, successivamente, sul set di features ottenute mediante feature selection.

4.1.1 Support Vector Machine

Risultati ottenuti utilizzando 38 features:

- Dos:
 - In linea generale su Dos entrambe le tecniche di validazione hanno dato buoni risultati, speculari sia su training set che su validation set, ottenendo 97% accuracy, 99% precision, 95% recall e 97% f-measure, sia con stratified cv che con k-fold cv. Andando ad applicare la standardizzazione si ottiene 98% su ogni metrica sia per training set che per validation set; andando, dunque, ad ottenere in media un lieve miglioramento. Per quanto riguarda le prestazioni sul test set, l'algoritmo si è comportato discretamente bene, ottenendo una accuracy del 83% una precision del 90% una recall del 68% e un f-measure del 77%. Applicando StandardScaler le prestazioni

migliorano sensibilmente, ottenendo 88% accuracy 90% precision 81% recall e 86% f-measure.

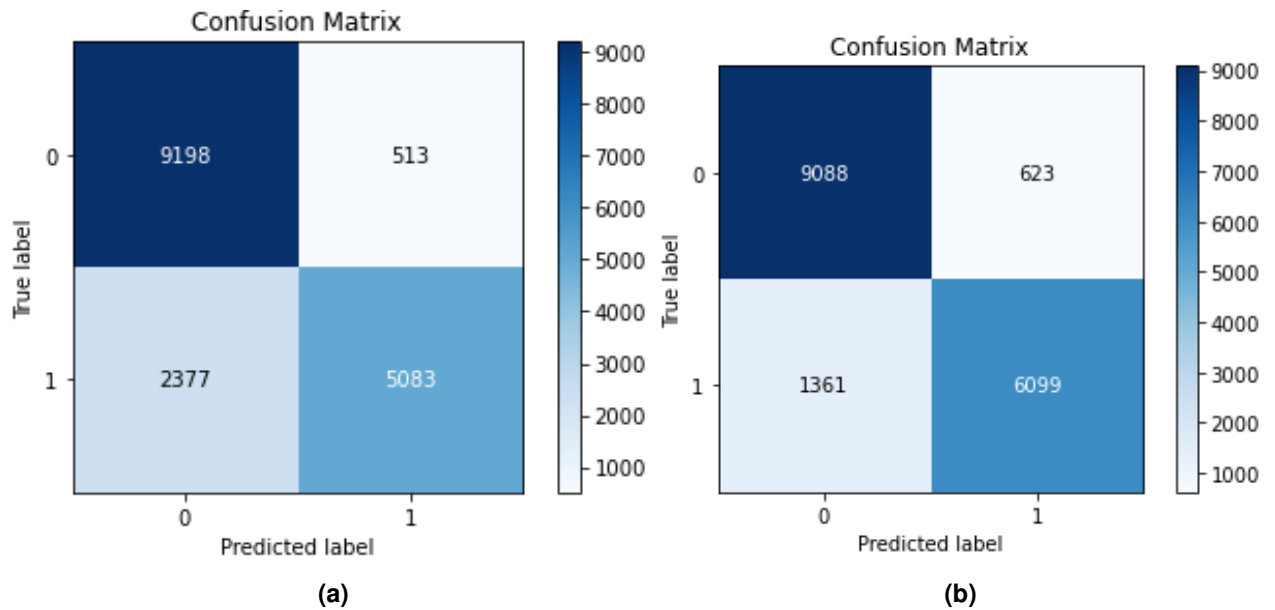


Figura 4.2: SVM, test set prediction su Dos, original data (a), StandardScaling (b).

- Probe:
 - Su Probe notiamo un decadimento di performance generale, k-fold cv sul training set ha una accuracy del 94%, una precision del 82%, una recall del 76% e un f-measure del 79%; medesime le performance sul validation set. Utilizzando stratified cv otteniamo invece sia sul training set che sul validation set una accuracy pari a 92%, una precision pari a 82% una recall di 65% e un f-measure di 69%. Andando ad utilizzare StandardScaler il quadro generale migliora, ottenendo una accuracy del 95%, precision 85%, recall 85% e f-measure 85% sia su training set che su validation set, per entrambe le tecniche. Sul test set si hanno performance discrete anche in questo caso, ottenendo 91% accuracy, 90% precision, 63% recall e 74% f-measure. Applicando StandardScaler ancora una volta le metriche migliorano, ottenendo accuracy 93%, precision 90%, recall 77% e f-measure 83%.

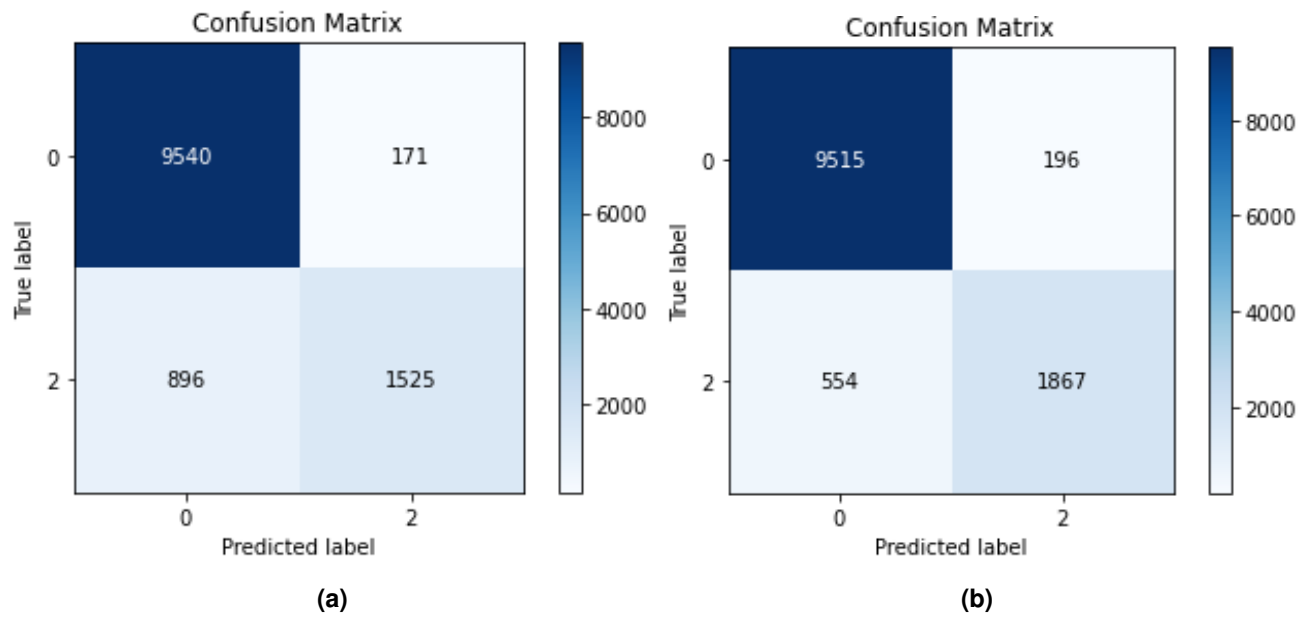


Figura 4.3: SVM, test set prediction su Probe, original data (a), StandardScaling (b).

- R2L:
 - In R2L, come è prevedibile a causa della scarsità di campioni, le performance sono poco convincenti. Senza StandardScaling, infatti, k-fold cv ha riscontrato una accuracy del 99%, tuttavia precision del 71%, recall 55% e f-measure 60%, performance non dissimili sia su training set che su validation set. Stratified cv non è andata molto meglio, mantenendo la stessa accuracy e precision di k-fold migliorando solo leggermente recall e f-measure con 57% e 63% rispettivamente. Applicando StandardScaler la situazione migliora ma non troppo: in entrambe le tecniche abbiamo accuracy invariata ed in entrambe le tecniche i risultati si equivalgono, ottenendo 75% precision, 67% recall e 71% f-measure. Nei test effettuati sul test set il calo di performance è stato drastico; a causa dell'ulteriore scarsità di campioni rispetto al training set si ottiene, infatti, una accuracy del 77%, una precision del 43% mentre recall e f-measure sono sotto il 10%. Chiaramente la quasi totale predominanza di campioni di traffico normale influenza totalmente il modello. Da come possiamo osservare nelle matrici di confusione in Figura 4.26 la percentuale di accuracy è

dettata unicamente dal campione di traffico normale e anche utilizzando StandardScaler la cosa resta invariata.

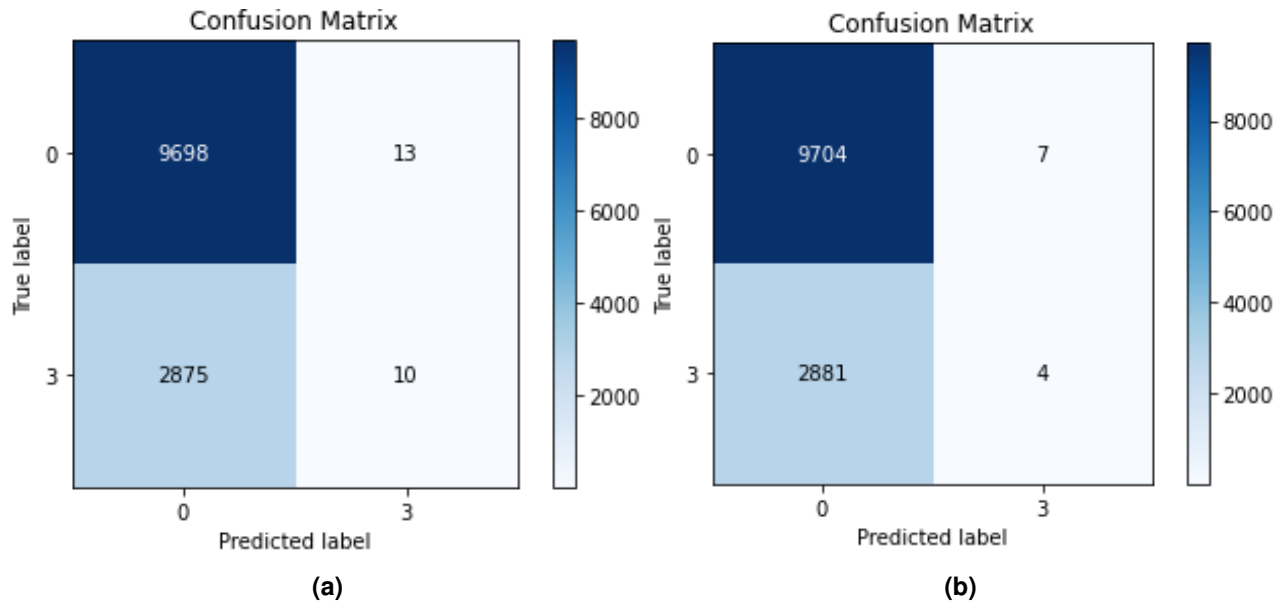


Figura 4.4: SVM, test set prediction su R2L, original data (a), StandardScaling (b).

- U2R:
 - L’influenza della classe di traffico normale, vista precedentemente in R2L, si presenta in modo ancora più marcato nei test effettuati su U2R. L’apparente accuracy del 99% ottenuta sia d k-fold cv che da stratified cv nasconde però un netto 0% sulle altre metriche, sia sul training set che sul validation set. Applicando StandardScaler i risultati migliorano nettamente, mantenendo in entrambi i casi il 99% di accuracy, aumentando però precision al 90%, recall al 56% e f-measure al 69% sul training set in entrambe le tecniche, mentre sul validation set vediamo k-fold cv performare leggermente meglio su precision, recall e f-measure rispettivamente pari a 81%, 53% e 62%, contro il 70%, 52% e 58% ottenuti con stratified cv, l’accuracy resta del 99%. Sul test set invece ci troviamo nella situazione analoga a R2L: abbiamo infatti un 99% di accuracy, 77% precision, 25% recall e 38% f-measure e anche usando standard scaling non ci sono miglioramenti.

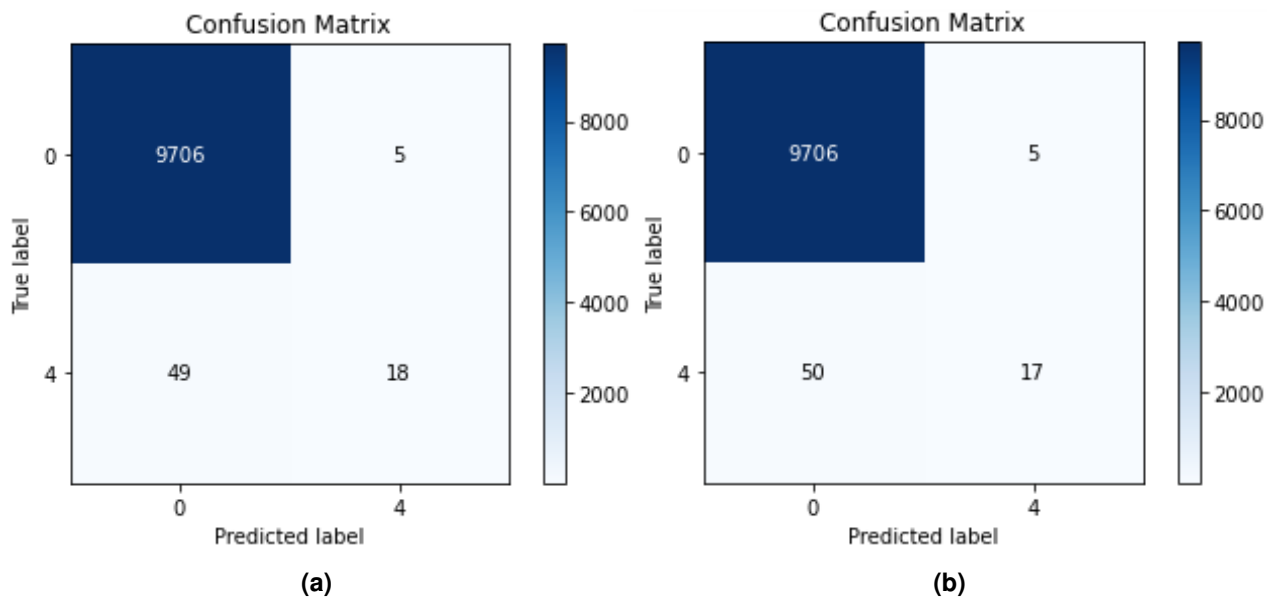
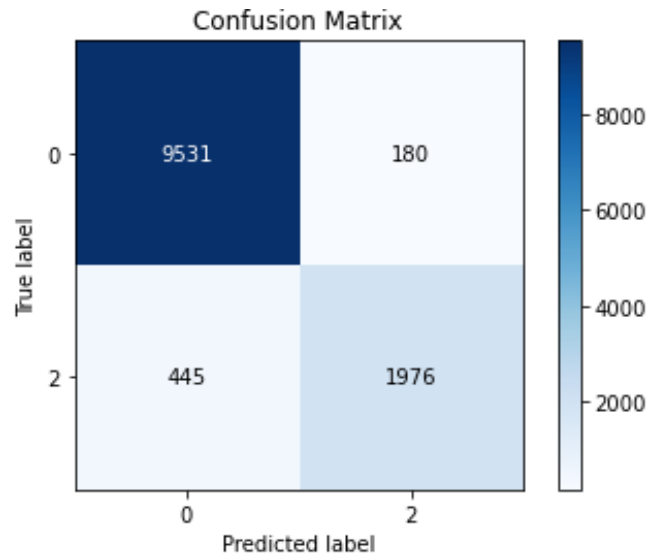


Figura 4.5: SVM, test set prediction su U2R, original data (a), StandardScaling (b).

Risultati ottenuti utilizzando 13 features:

Applicando RFE, su Dos si ottiene un leggero calo di performance, mentre su Probe si ha un sensibile miglioramento. Entrambe le tecniche sul training set hanno una accuracy del 95%, una precision del 83%, una recall del 84% e un f-measure del 83%, medesime performance sul validation set. Andando ad utilizzare StandardScaler le performance migliorano anche se in maniera poco rilevante. Andando a sperimentare sul test set le performance sono decisamente migliorate, ottenendo 94% accuracy, 91% precision, 81% recall e 86% f-measure; tuttavia applicando StandardScaler c'è un peggioramento. R2L e U2R, invece, mantengono le performance anomale dovute allo sbilanciamento.



(a)

Figura 4.6: SVM, test set prediction su Probe, feature selection.

4.1.2 Logistic Regression

Risultati ottenuti utilizzando 38 features:

- Dos:
 - Su Dos con Logistic regression entrambe le tecniche di validazione hanno mantenuto bene o male gli stessi risultati già visti precedentemente con SVM; solo la recall sembra essere calata nel caso della stratified cv e aumentata in k-fold cv, passando dal 95% visto con SVM, al 98% in k-fold cv e 92% in stratified cv sia su trainig set che su validation set. Con l'utilizzo di StandardScaling i risultati si omogeneano sul 98% in entrambe le tecniche su entrambi i set. Anche per quanto riguarda le prestazioni sul test set l'algoritmo si è discostato poco da SVM, ottenendo una accuracy del 82% una precision del 90% una recall del 65% e un f-measure del 76%, applicando StandardScaler le prestazioni migliorano sensibilmente e si ottengono risultati identici a quanto visto con SVM.

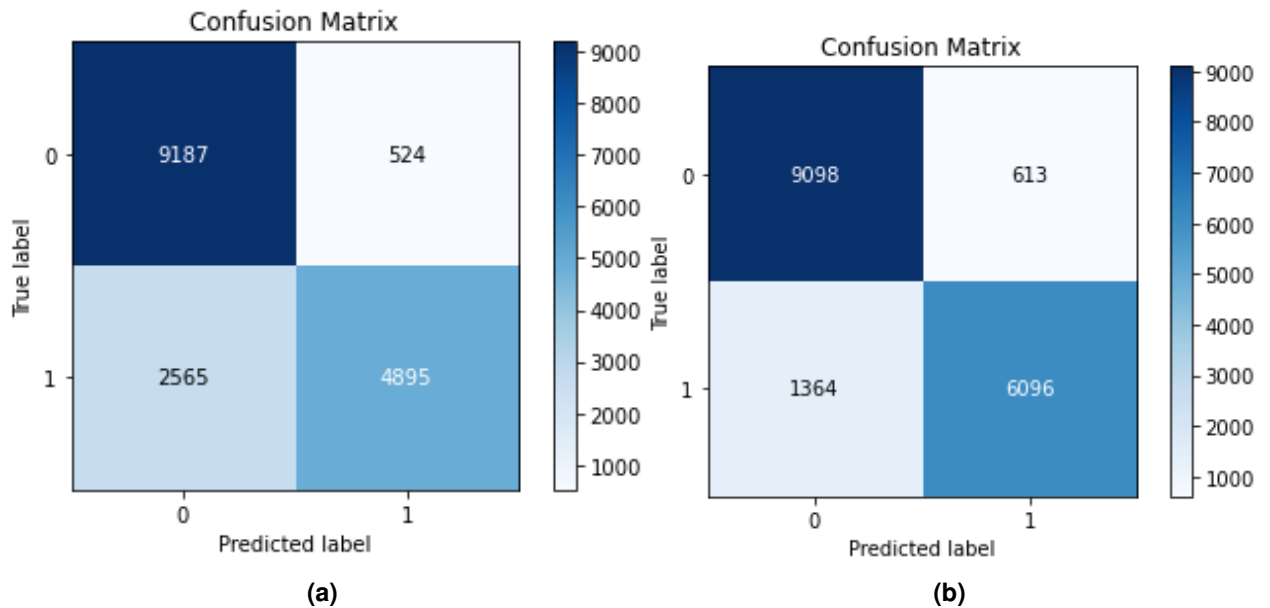


Figura 4.7: Logistic Regression, test set prediction su Dos, original data (a), StandardScaling (b).

- Probe:
 - In Probe notiamo le prime differenze marcate rispetto a SVM. Entrambe le tecniche infatti, pur mantenendo una accuracy e una precision accettabili, di 88% e 84%, notiamo un calo drastico di recall ed f-measure, ottenendo rispettivamente valori intorno al 24% e 36% sia su training set che su validation set. Applicando StandardScaling le performance migliorano decisamente, ottenendo per entrambe le tecniche 95% accuracy, 87% precision, 84% recall e 85% f-measure molto simili a quanto visto con SVM. Sul test set si hanno performance decisamente non soddisfacenti, ottenendo 79% accuracy 0% sulle altre metriche. Applicando StandardScaler le metriche migliorano e anche in questo caso otteniamo performance speculari a SVM.

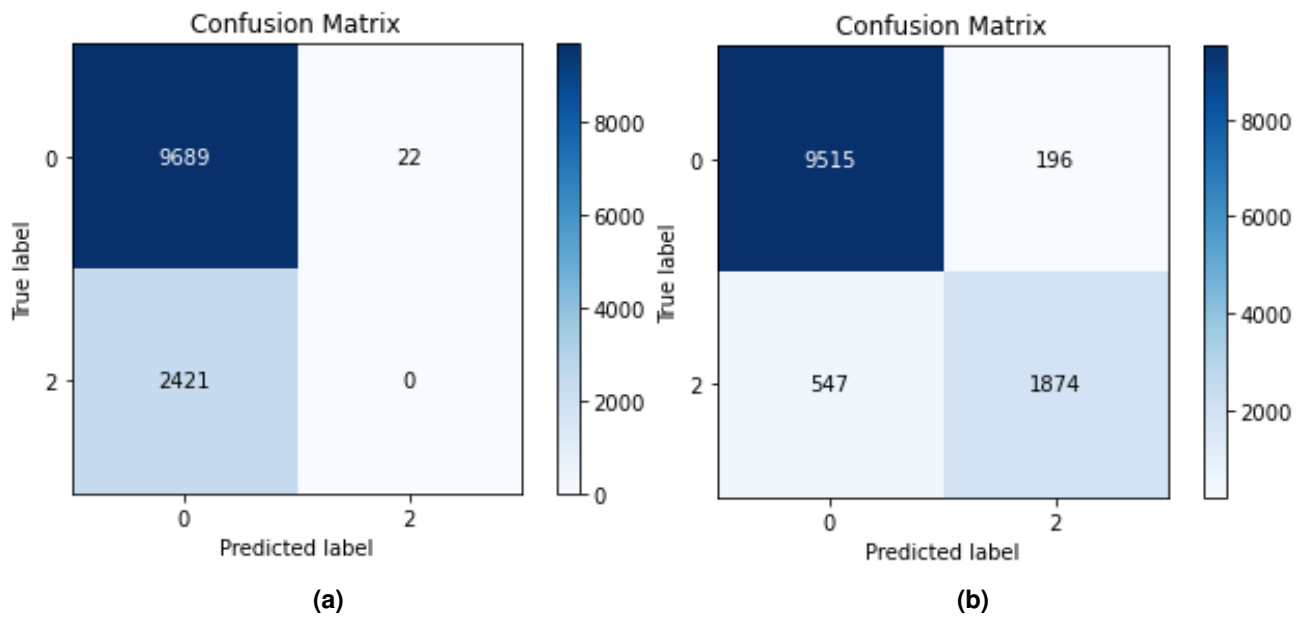


Figura 4.8: Logistic Regression, test set prediction su Probe, original data (a), StandardScaling (b).

- R2L:
 - La validation su R2L ha avuto un calo drastico, nonostante una accuracy del 98%, riscontriamo una precision del 58% e una recall ed f-measure inferiori all'1% per entrambe le tecniche di validazione sia su training set che su validation. Al solito con StandardScaler le prestazioni migliorano e mostrano anche una recall del 71% superiore a quella vista SVM, risultati in linea invece per le altre metriche. Sul test set le performance sono addirittura peggiori di SVM, e anche utilizzando StandardScaler non migliorano.

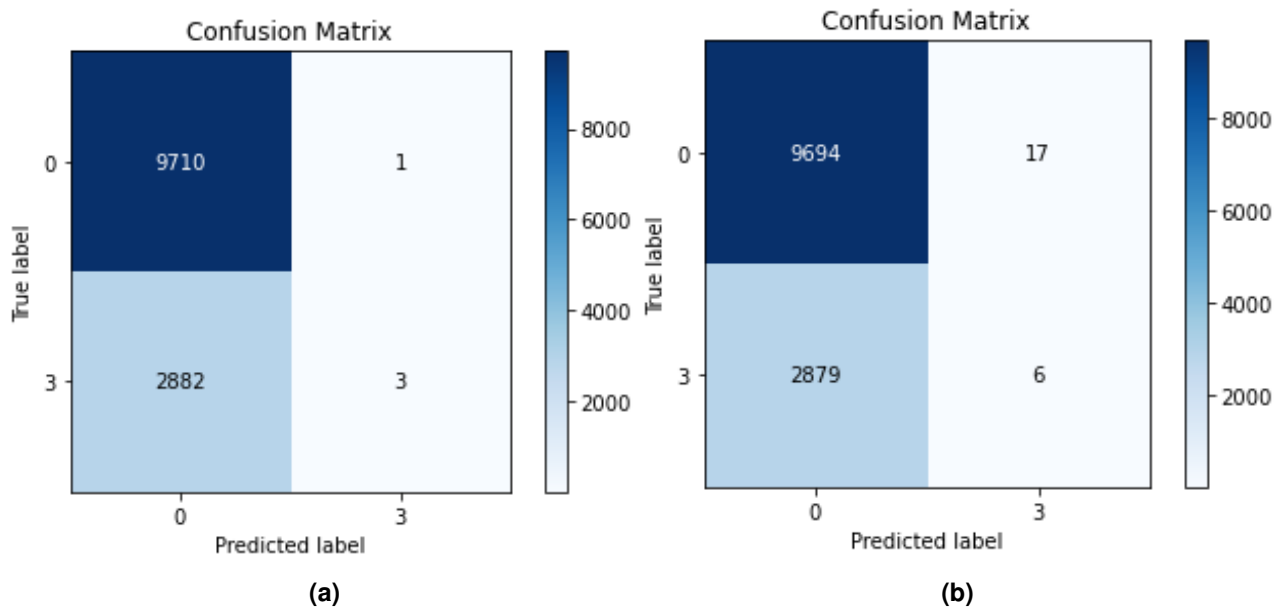


Figura 4.9: Logistic Regression, test set prediction su R2L, original data (a), StandardScaling (b).

- U2R:
 - Su U2R le performance delle tecniche di validazione risultano migliori rispetto a SVM: non abbiamo infatti metriche con 0%. Tuttavia, escludendo accuracy del 99%, anche qui lo sbilanciamento del dataset ha un impatto netto, difatti le altre metriche restano sotto il 30% sia sul training set che sul validation set. Applicando StandardScaler, le performance sono equivalenti ad SVM. Le performance sul test set sono identiche a quelle viste con SVM senza StandardScaling, mentre con StandardScaling risultano essere leggermente inferiori su recall e f-measure.

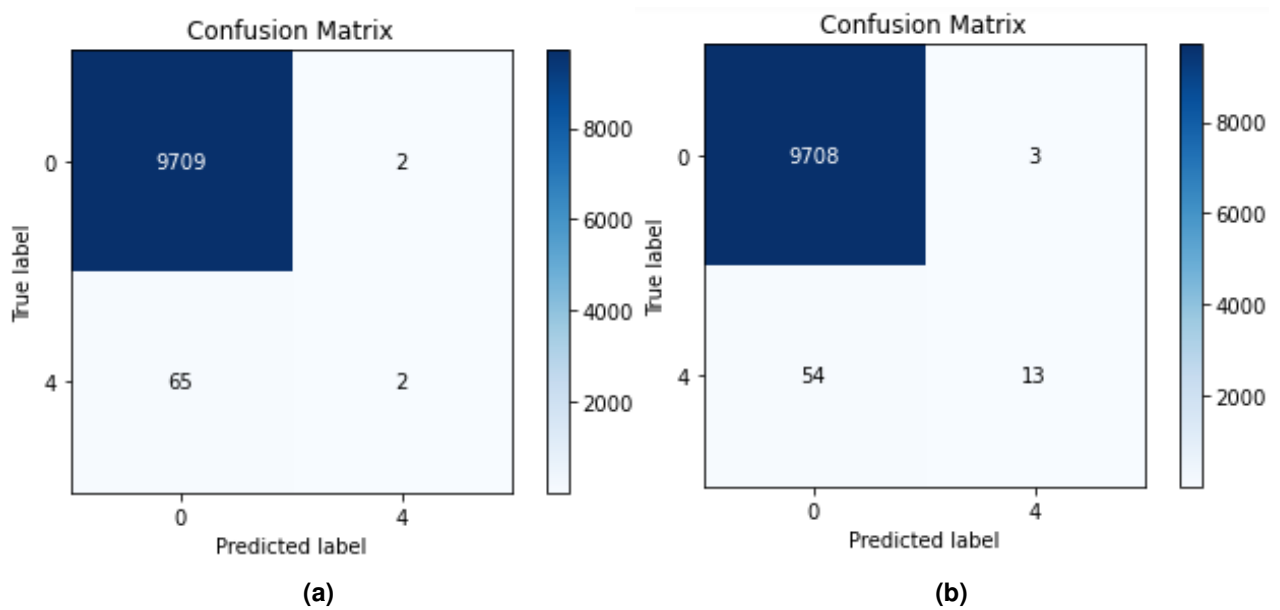


Figura 4.10: Logistic Regression, test set prediction su U2R, original data (a), StandardScaling (b).

Risultati ottenuti utilizzando 13 features:

RFE non porta nessun risultato differente rispetto a quanto visto in precedenza, se non per leggeri peggioramenti trascurabili su Dos.

4.1.3 KNN

Risultati ottenuti utilizzando 38 features:

- Dos:
 - Utilizzando KNN entrambe le tecniche di validazione su Dos ottengono un 99% in tutte le metriche sia su validation che su training set. Anche per quanto riguarda le prestazioni sul test set, KNN mostra risultati leggermente migliori delle tecniche viste in precedenza, ottenendo una accuracy del 88%, una precision del 96%, un recall del 75% e un f-measure del 84%. Applicando StandardScaler le prestazioni migliorano ulteriormente ottenendo 89% accuracy, 92% precision, 82% recall e 87% f-measure.

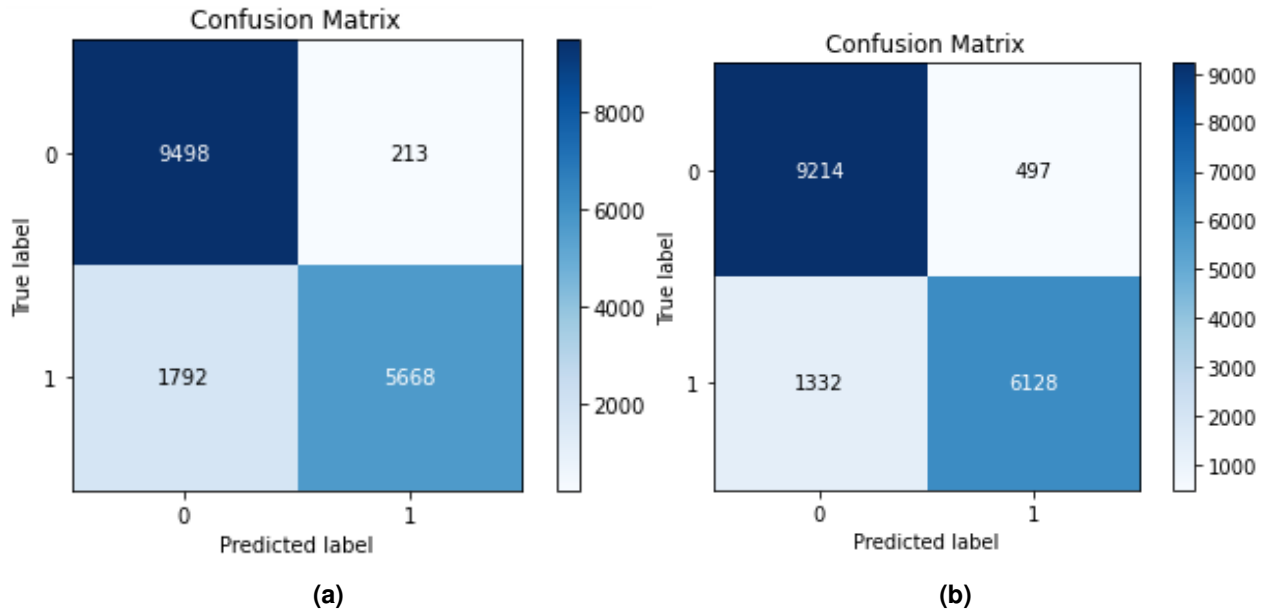


Figura 4.11: KNN, test set prediction su Dos, original data (a), StandardScaling (b).

- Probe:
 - Anche su probe l’algoritmo si mostra più performante rispetto a quelli visti prima, ottenendo risultati tra il 99% e il 98% su entrambe le tecniche sia su validation set che su training set. Tuttavia sul test set si hanno performance inferiori, ottenendo una accuracy pari a 91%, precision 87%, recall 64% e f-measure 74%, applicando lo scaling, si ha un peggioramento, seppur minimo, delle performance.

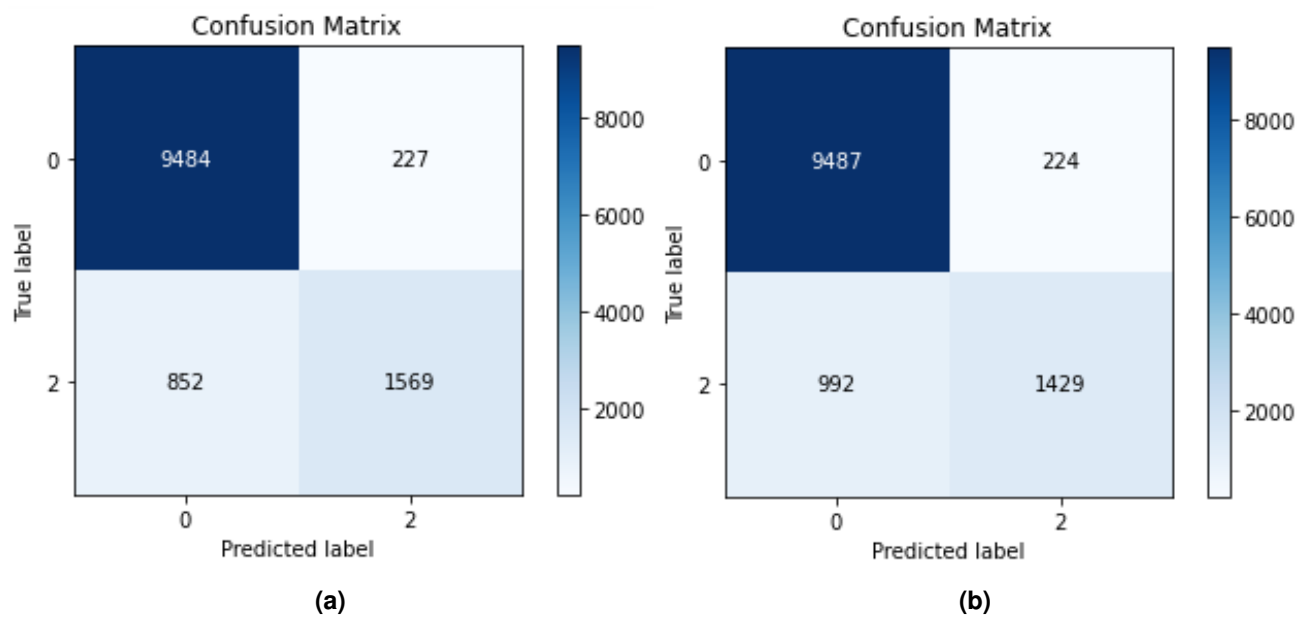


Figura 4.12: KNN, test set prediction su Probe, original data (a), StandardScaling (b).

- R2L:
 - Su R2L a differenza di SVM e LR con KNN non otteniamo un netto peggioramento delle performance: sia stratified cv che k-fold cv danno ottimi risultati su tutte le metriche oscillando su entrambi i dataset su percentuali tra il 94% e il 99% per tutte le metriche. Le metriche però non si riflettono sul test set che risulta presentare al solito un'accuracy del 77% e altre metriche inferiori al 1%.

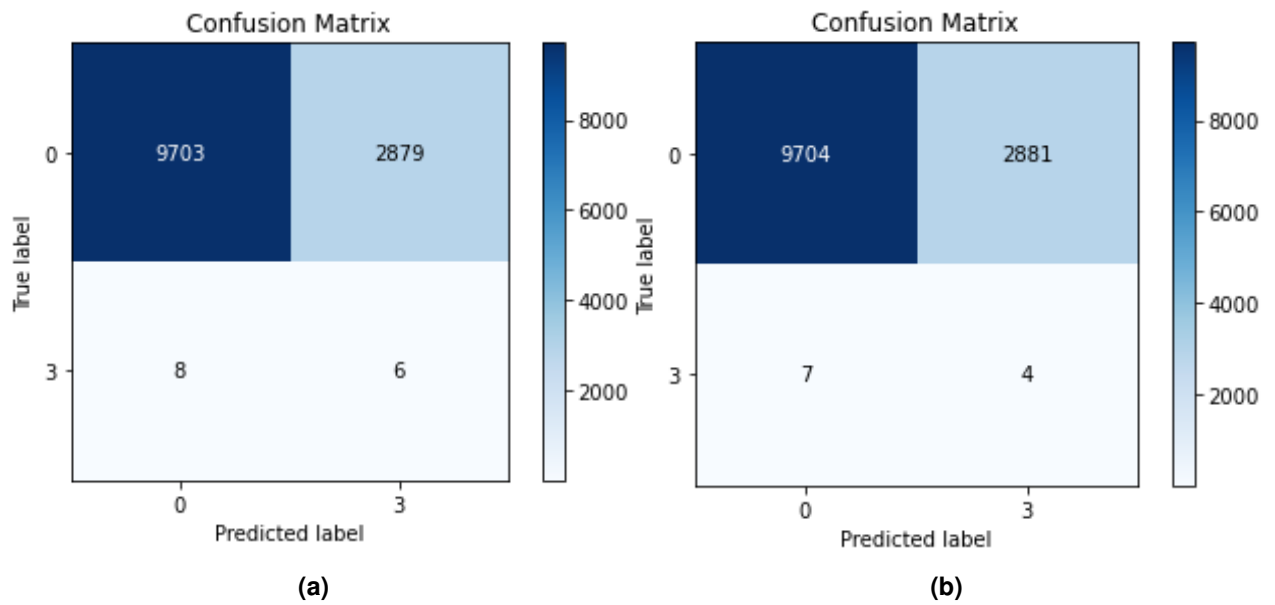


Figura 4.13: KNN, test set prediction su R2L, original data (a), StandardScaling (b).

- U2R:
 - Su U2R le performance delle tecniche di validazione iniziano ad essere più scarse, ottenendo il solito 99%, 82% precision ma 49% recall e 60% f-measure, tuttavia anche in questo caso sono superiori a SVM e LR. Sul test set, si ottengono 99% e 100% di accuracy e precision, ma valori di recall e f-measure sotto il 20% sia con StandardScaling che senza.

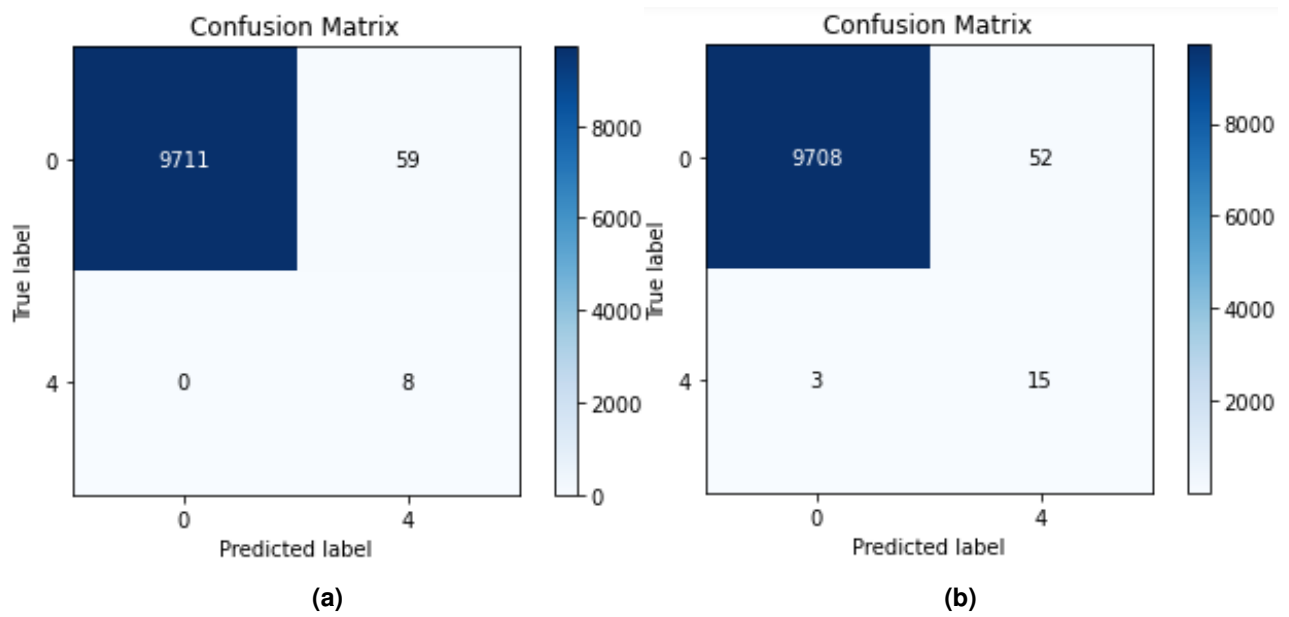


Figura 4.14: KNN, test set prediction su U2R, original data (a), StandardScaling (b).

Risultati ottenuti utilizzando 13 features:

Con l'utilizzo di RFE le performance risultano essere leggermente inferiori su Dos ottenendo una accuracy del 89% una precision del 96%, quindi superiore al caso con 38 features, ma una recall del 77% e un f-measure del 86%; mentre c'è un lieve miglioramento su Probe ottenendo una accuracy pari a 91%, precision 88%, recall 67% e f-measure 76%; R2L e U2R non mostrano differenze significative.

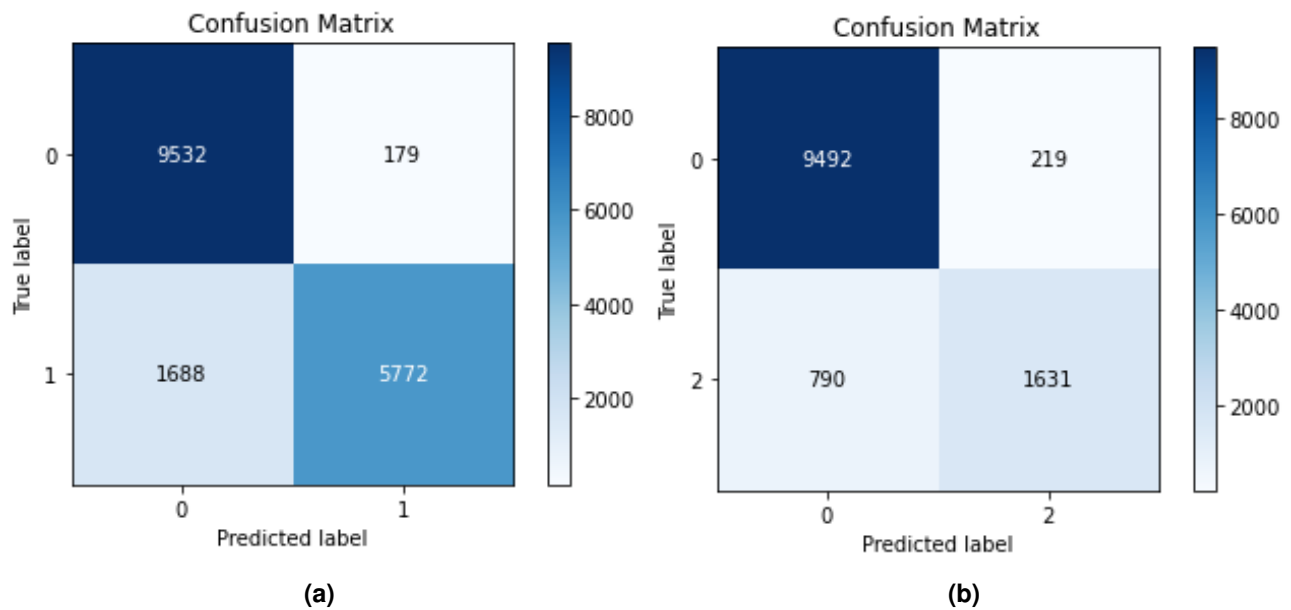


Figura 4.15: KNN, test set prediction su Dos, RFE (a), Probe (b)

4.1.4 Decision Tree

Risultati ottenuti utilizzando 38 features:

- Dos:
 - Rispetto alle performance viste precedentemente con KNN, le tecniche di validazione con DecisionTree hanno performato peggio, pur restando su ottimi valori. Si hanno, infatti, per entrambe le tecniche e sia su training set che validation set, una accuracy del 97%, precision del 99%, recall del 93% e f-measure del 96%. Sul test set vediamo dei numeri ben inferiori a quelli ottenuti durante la validazione, ma abbastanza in linea con le tecniche viste in precedenza; si ha infatti 83% di accuracy, 96% di precision, 65% recall e 77% f-measure. Applicando StandardScaler c'è un leggero peggioramento di tutte le metriche di circa due punti percentuale.

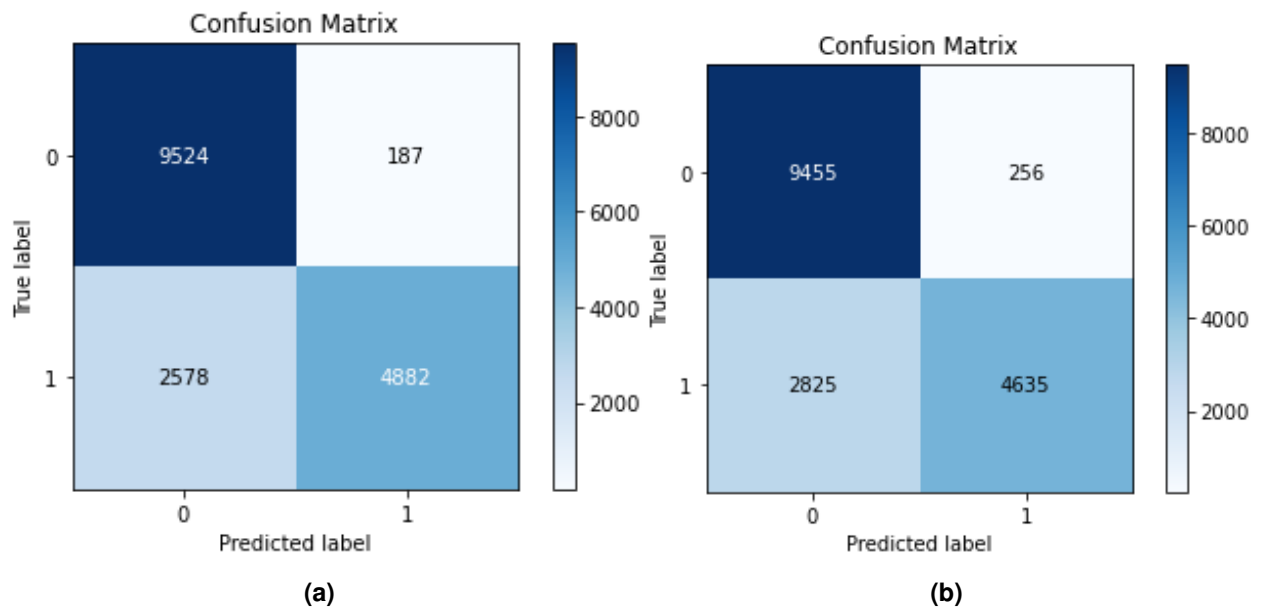


Figura 4.16: DT, test set prediction su Dos, original data (a), StandardScaling (b).

- Probe:
 - Anche su probe l’algoritmo si mostra meno performante rispetto a KNN ma più performante di SVM e LR, ottenendo risultati tra il 90% e il 97% su entrambe le tecniche sia su validation set che su training set. Tuttavia sul test set si hanno performance inferiori sia a KNN che alle altre tecniche viste, ottenendo una accuracy pari a 90%, precision 85%, recall 63% e f-measure 72%. Applicando lo scaling, si ha un peggioramento, anche importante delle performance, scendendo di cinque punti percentuali in accuracy e quattordici punti percentuali in precision.

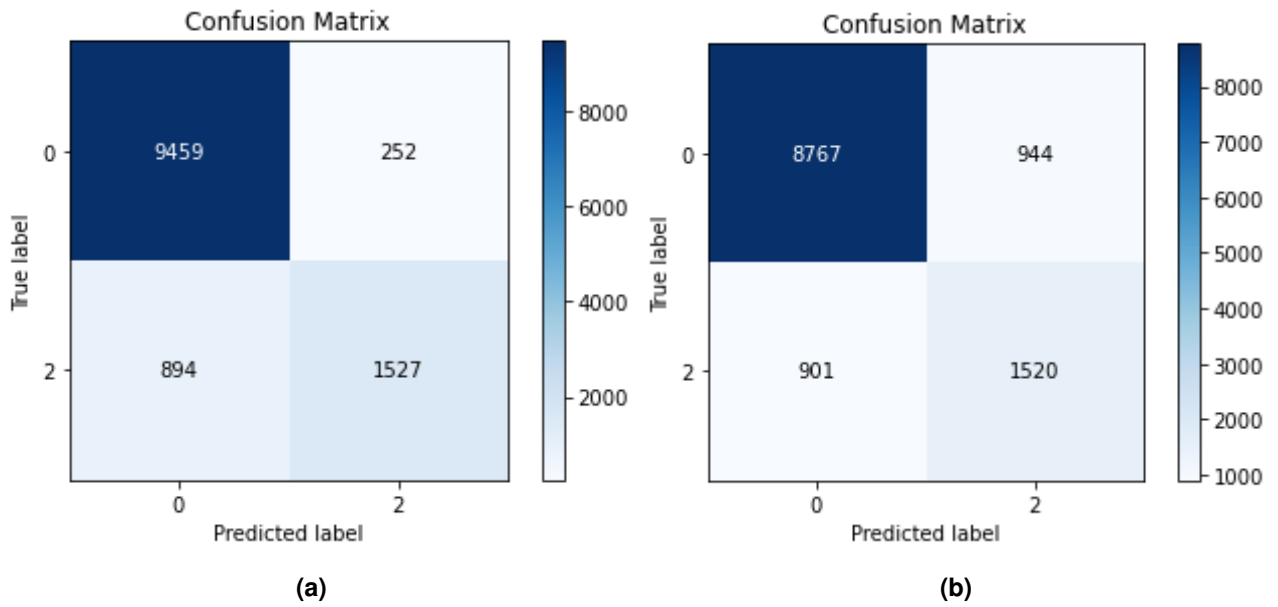


Figura 4.17: DT, test set prediction su Probe, original data (a), StandardScaling (b).

- R2L:

- R2L mostra la solita accuracy e precision sul 99% ma scarsissima recall ed f-measure, sia su training set che su validation set per entrambe le tecniche di validazione. La situazione non è diversa sul test set e lo StandardScaling ancora una volta non aiuta con questa tecnica.

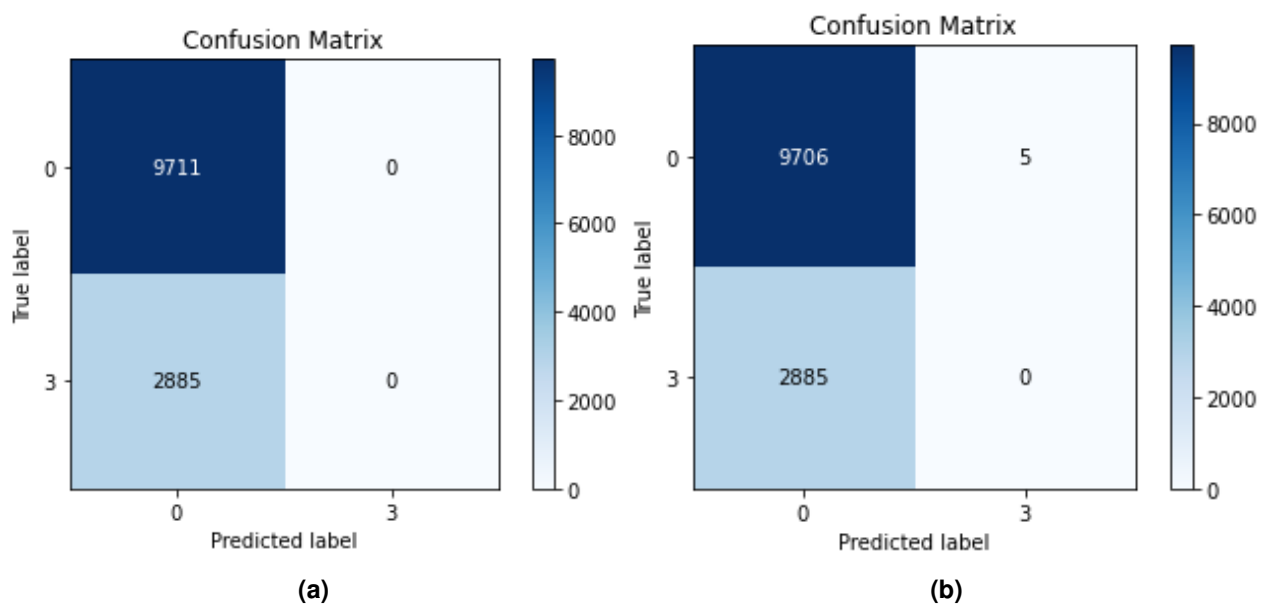


Figura 4.18: DT, test set prediction su R2L, original data (a), StandardScaling (b).

- U2R:
 - Situazione analoga a quanto visto con R2L anche su U2R a causa dello sbilanciamento, accuracy e precision discrete, ma con scarsa significatività a causa dei valori estremamente bassi delle altre metriche.

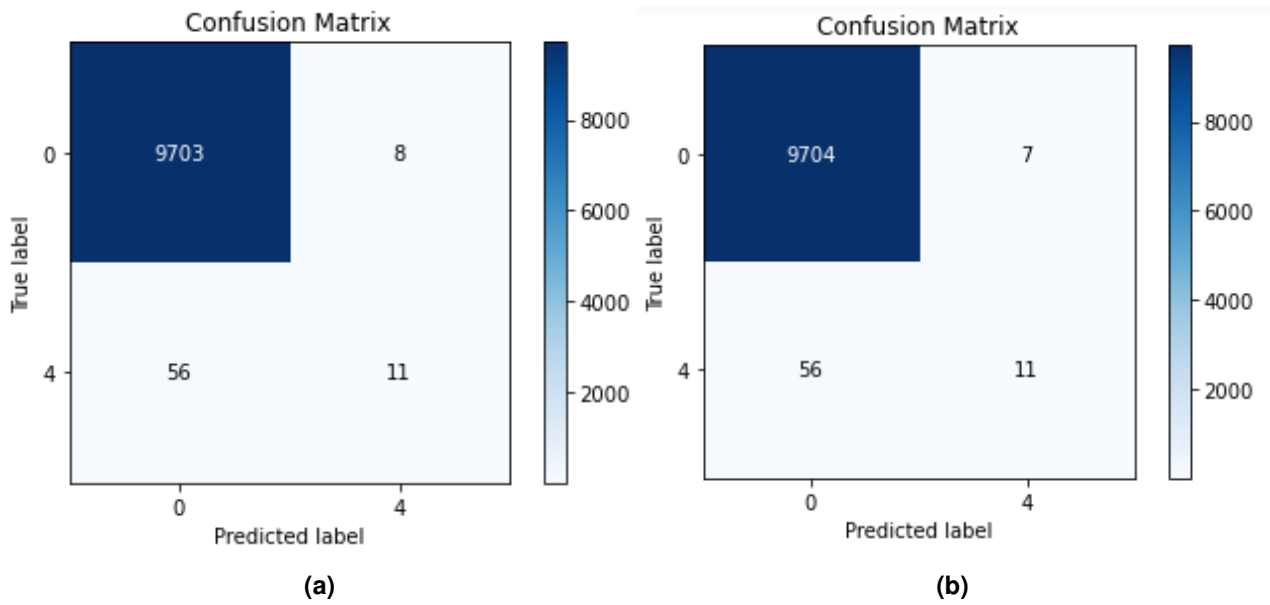


Figura 4.19: DT, test set prediction su U2R, original data (a), StandardScaling (b).

Risultati ottenuti utilizzando 13 features:

Con l'utilizzo di RFE le performance sono risultate del tutto identiche.

4.1.5 Random Forest

Risultati ottenuti utilizzando 38 features:

- Dos:
 - Mediante Random Forest torniamo ad avere risultati in linea con KNN, sul training set mediante le tecniche di validazione si ha un 99% su ogni metrica, analogamente anche sul validation set. Per quanto riguarda il test set si ha una accuracy del 90%, precision 98%, recall 78% e f-measure 87%. Lo scaling delle caratteristiche con StandardScaler riduce, seppur di poco, le performance.

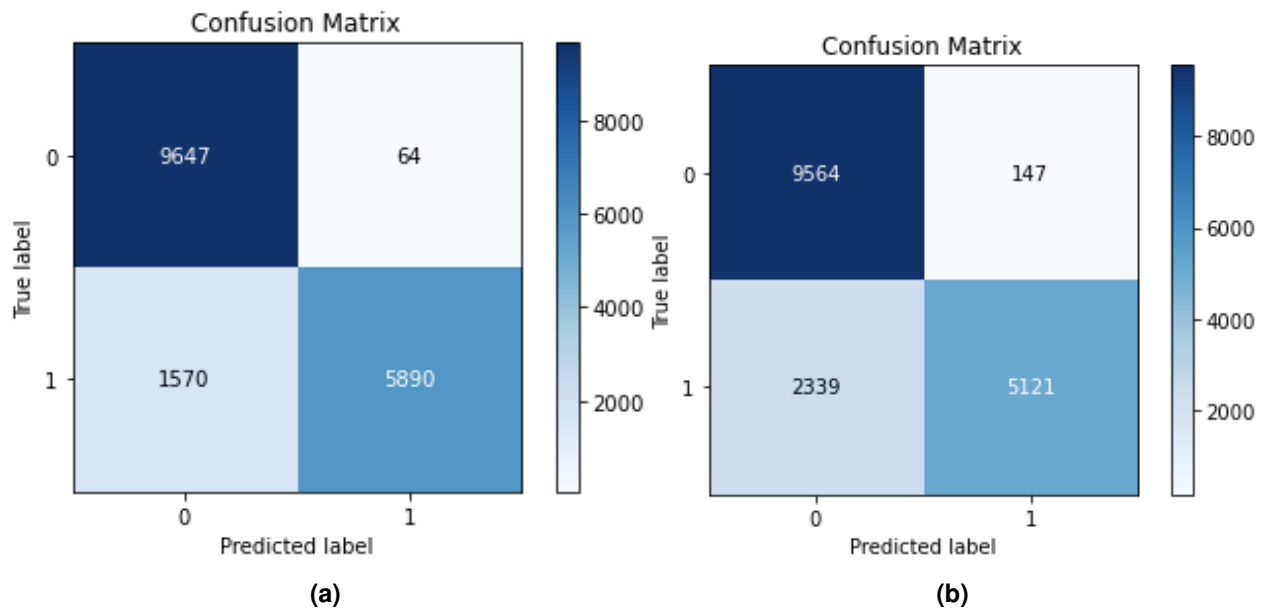


Figura 4.20: RF, test set prediction su Dos, original data (a), StandardScaling (b).

- Probe:
 - Anche su probe l’algoritmo si mostra in linea con KNN, ottenendo nuovamente metriche del 99% su entrambe le tecniche sia su validation set che su training set per tutte le metriche. Sul test set si hanno performance inferiori, ottenendo una accuracy pari a 90%, precision 87%, recall 59% e f-measure 71%, applicando lo scaling, si ha un peggioramento.

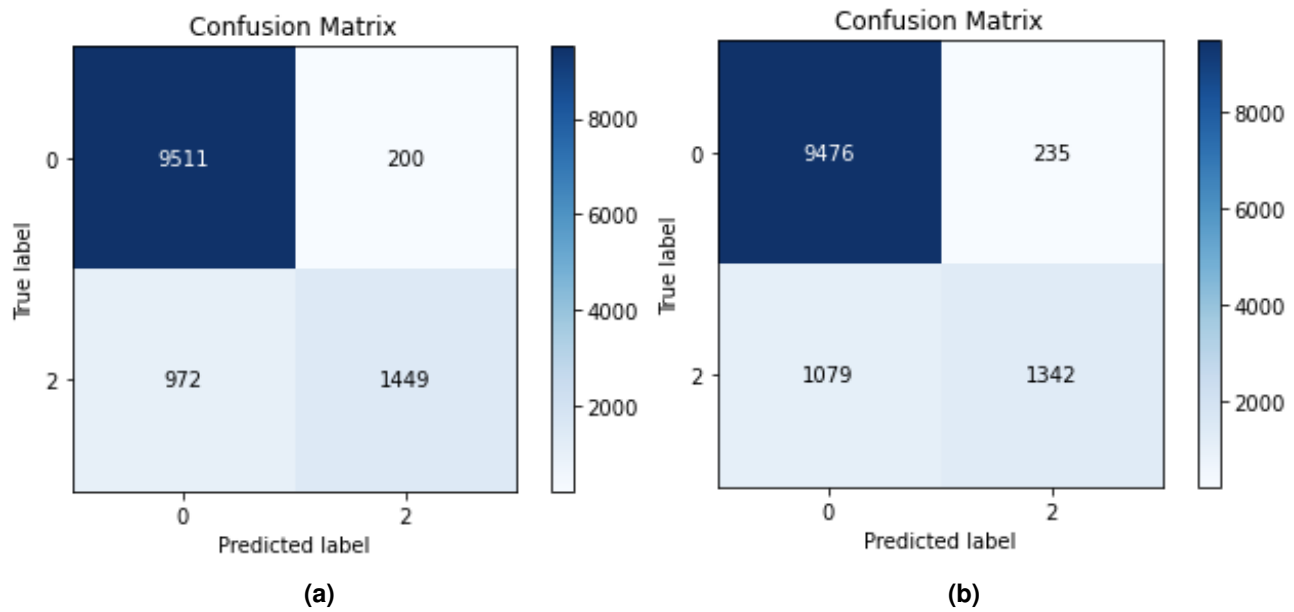


Figura 4.21: RF, test set prediction su Probe, original data (a), StandardScaling (b).

- R2L:

- Su R2L si ha accuracy e precision sul 99%, recall e f-measure sul 96%, sia su training set che su validation set per entrambe le tecniche di validazione. Al solito sul test set si ha recall e f-measure sotto il 10%.

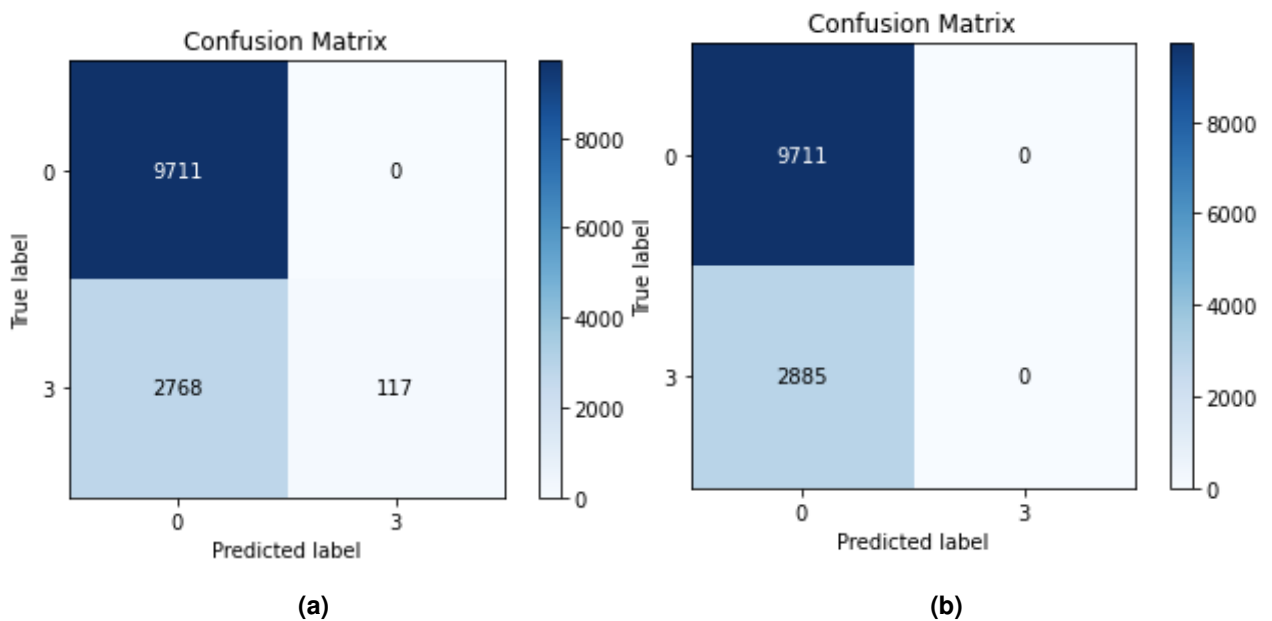


Figura 4.22: RF, test set prediction su R2L, original data (a), StandardScaling (b).

- U2R:
 - Situazione analoga a quanto visto con R2L anche su U2R.

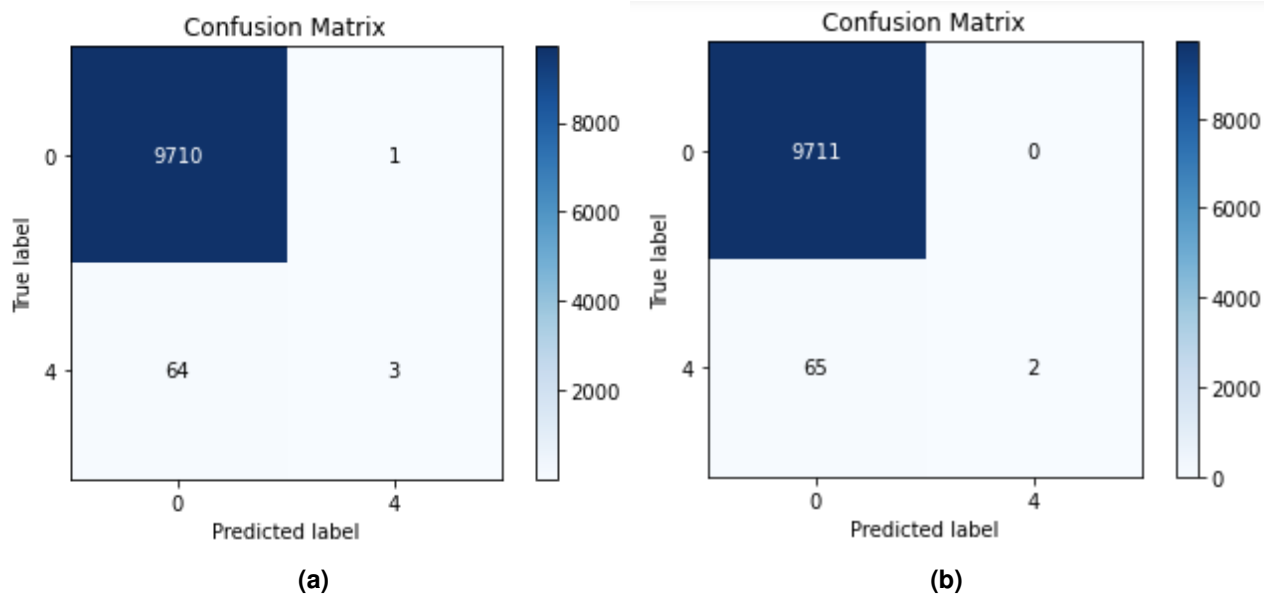


Figura 4.23: RF, test set prediction su U2R, original data (a), StandardScaling (b).

Risultati ottenuti utilizzando 13 features:

Come per DT anche RF non beneficia di feature selection, non ottenendo variazioni su nessun tipo.

4.1.6 Neural Network

Risultati ottenuti utilizzando 38 features:

- Dos:
 - Le performance ottenute mediante Neural Network sono migliori di quelle viste in precedenza: sul training set e validation set si ha 99% per ogni metrica per entrambe le tecniche. Sul test set le performance sono leggermente migliori rispetto a KNN; si ha infatti, accuracy pari a 89%, precision 97% recall 78% e f-measure 86%. Applicando lo StandardScaler si ha un leggero miglioramento ottenendo accuracy 91%, recall 86% e f-measure 89% a discapito di un abbassamento di precision che scende al 93%.

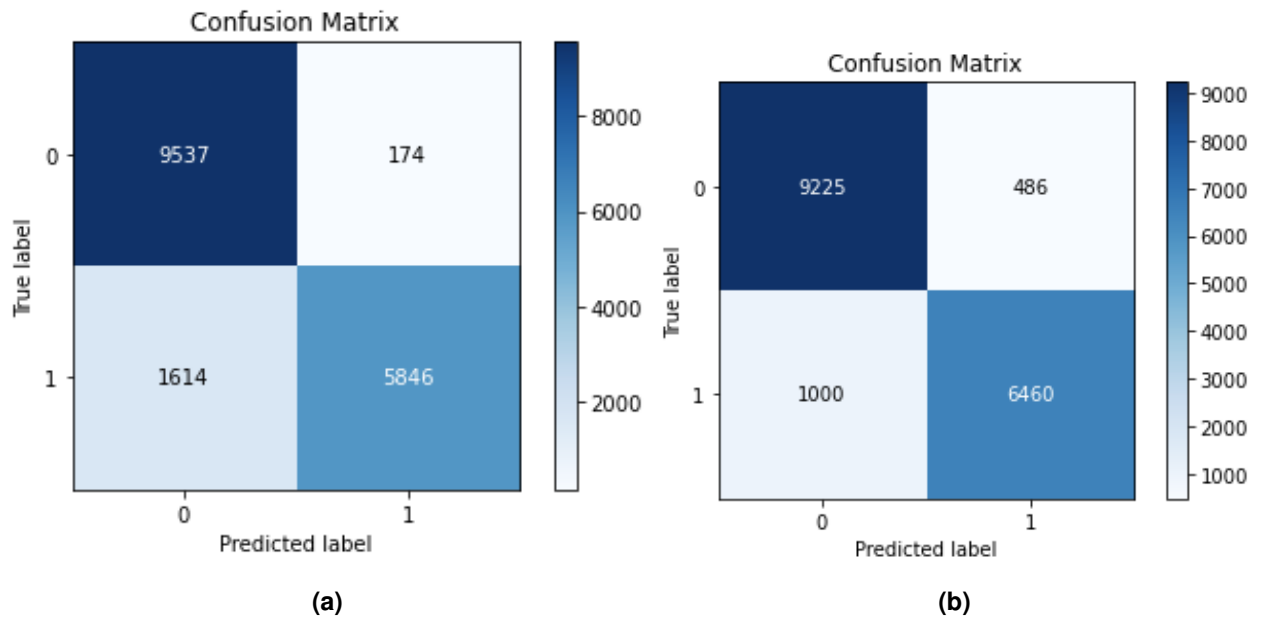


Figura 4.24: NN, test set prediction su Dos, original data (a), StandardScaling (b).

- Probe:
 - Su probe l’algoritmo si mostra bene o male in linea con KNN, ottenendo metriche tra il 97% e il 99% su entrambe le tecniche sia su validation set che su training set. Sul test set si ha una accuracy pari a 90%, precision 81%, recall 66% e f-measure 73%, applicando lo scaling, si ha un peggioramento di recall e f-measure e un lieve miglioramento di precision.

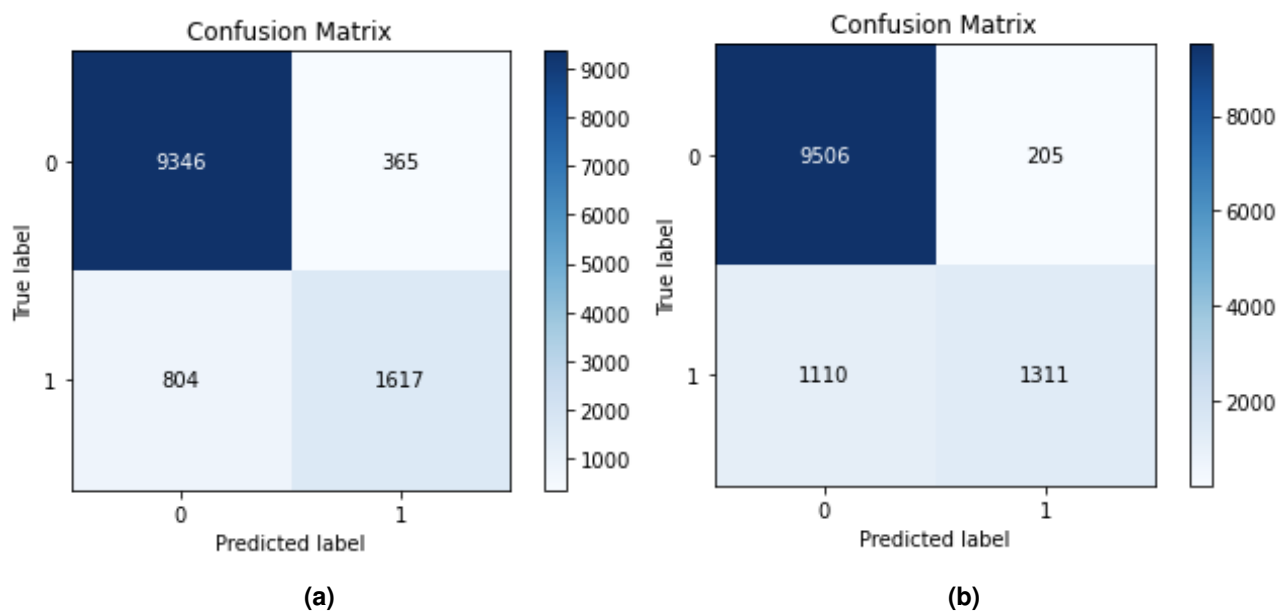


Figura 4.25: NN, test set prediction su Probe, original data (a), StandardScaling (b).

- R2L:
 - Su R2L si ha accuracy e precision sul 99%, mentre recall e f-measure sotto il 30%, sia su training set che su validation set per entrambe le tecniche di validazione. Sul test set invece si ha accuracy del 77%, mentre le altre metriche sono uguali a 0. Applicando lo scaling si ha un miglioramento nella precision che passa al 93% ma recall e f-measure restano sotto il 5%.

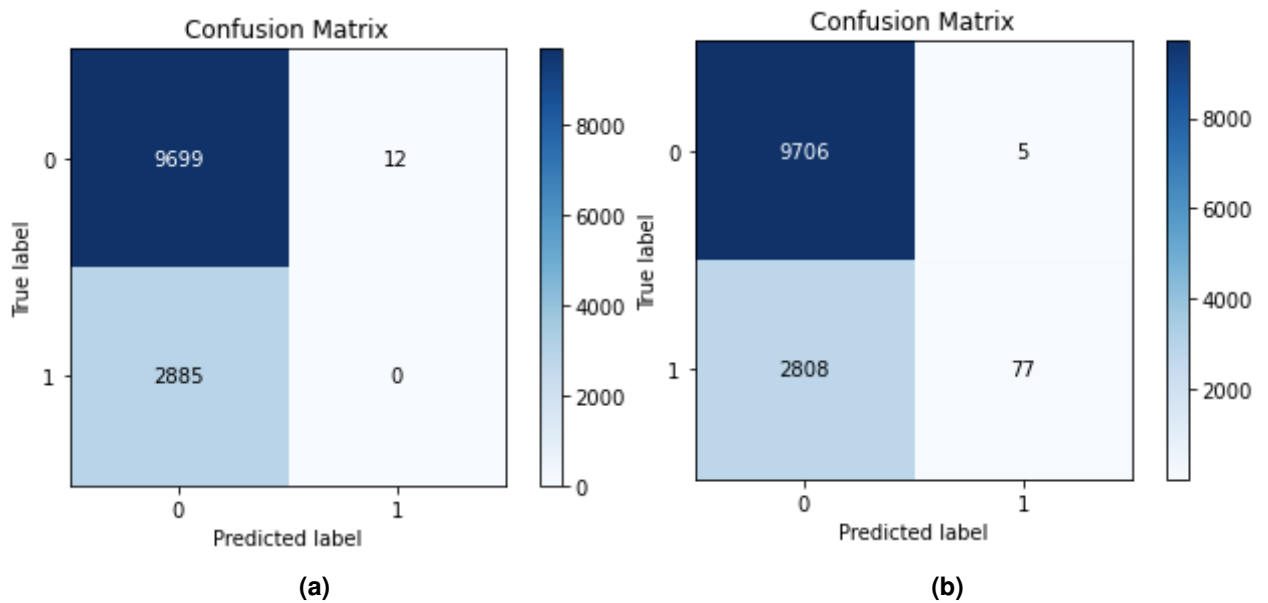


Figura 4.26: NN, test set prediction su R2L, original data (a), StandardScaling (b).

- U2R:
 - Per U2R si hanno su training set e validation set valori pari al 99% per l'accuracy e valori sotto l'1% per le altre metriche con k-fold cv. Applicando StandardScaler migliorano ottenendo sempre 99% accuracy, 94% precision, 55% recall e 68% f-measure. Con stratified cv si ha 99% accuracy, 37% precision, 25% recall e 26% f-measure. Applicando StandardScaler si ha 99% accuracy, 89% precision, 53% recall e 65% f-measure. Sul test set si ha 99% accuracy ma le altre metriche sono pari a 0. Applicando StandardScaler si ha 99% accuracy, 68% precision, 0.35% recall e 45% f-measure.

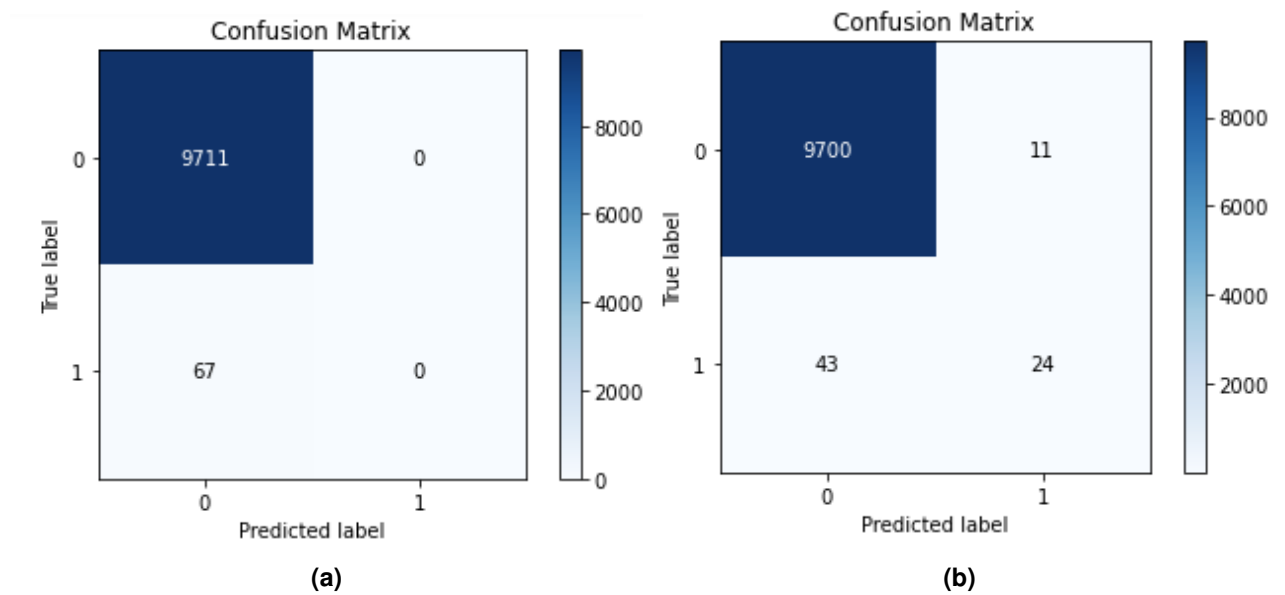


Figura 4.27: NN, test set prediction su U2R, original data (a), StandardScaling (b).

Risultati ottenuti utilizzando 13 features:

Lavorare con 13 features ha portato un leggero calo di performance sulla classe di attacco Dos: l'accuracy scende da 89 a 86%, la precision resta del 97%, la recall scende dal 78 al 70% e f-measure scende dal 86 al 81%. Su Probe mostrano risultati migliori rispetto alle 38 features: si ha, infatti, un leggero incremento di tutte le metriche sul test set, ottenendo: accuracy 91%, precision 87%, recall 67% e f-measure 76%. R2L e U2R continuano a mostrare risultati poco soddisfacenti.

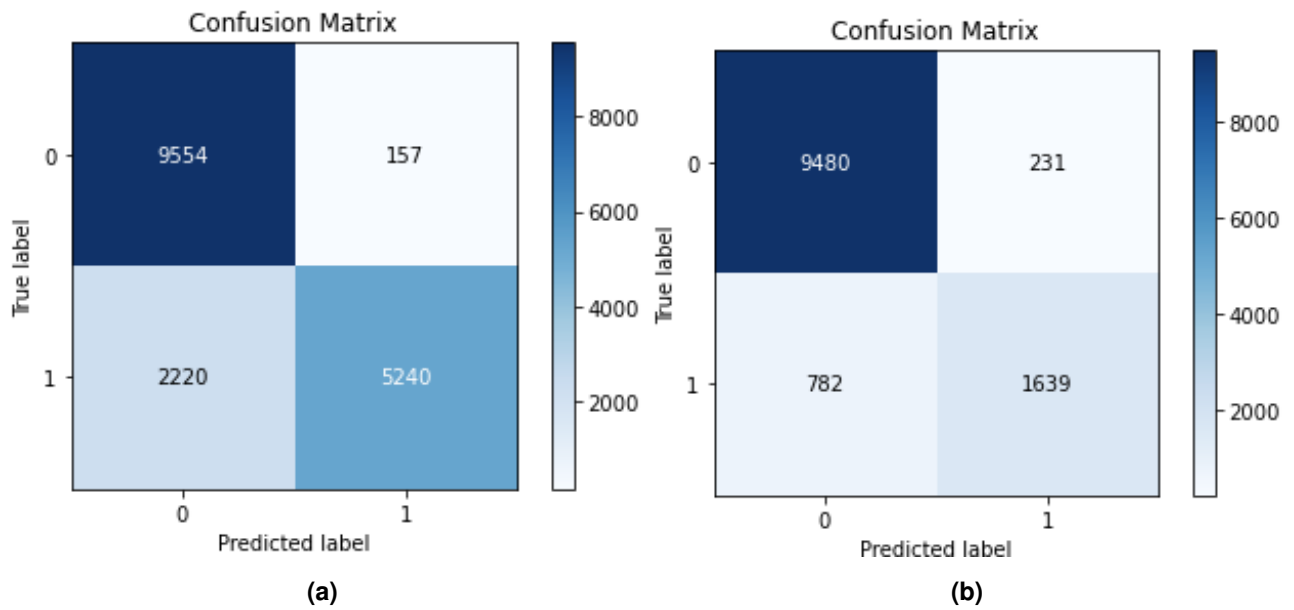


Figura 4.28: NN, test set prediction su Dos(a) e su Probe(b), feature selection.

4.2 Risultati ottenuti applicando l'oversampling

Dopo i primi test effettuati, è stato applicato l'oversampling al dataset in modo da andare ad effettuare un bilanciamento delle classi. Al fine di rendere più scorrevole la lettura dei risultati, andremo a discutere solo degli eventuali miglioramenti o peggioramenti di performance ottenuti, tralasciando le situazioni in cui le prestazioni ottenute sono rimaste invariate o sono variate in maniera poco significativa.

4.2.1 Support Vector Machine

- Probe:
 - Le tecniche di validazione su probe hanno beneficiato dell'oversampling, in particolare la stratified CV ha ottenuto un aumento di dieci punti percentuale su tutte le metriche che su cui si sono avute percentuali tra il 93% e il 96%. Inoltre l'oversampling, insieme allo StandardScaler e alla feature selection, sul test set ha incrementato precision, recall ed f-measure che sono passate rispettivamente da 90%, 77% e 83% a 96%, 87% e 91%, resta invariata l'accuracy.

- R2L:
 - Utilizzando l'oversampling le tecniche di validazione ottengono performance nettamente migliori. Ancora una volta stratified CV presenta delle metriche anche di venti e trenta punti percentuale superiori al caso senza oversampling, ottenendo valori tra il 96% e il 97%. Sul test set, coadiuvato da feature selection e scaling, le performance risultano migliori, anche se non ancora soddisfacenti. L'accuracy non è più anomala, assestandosi sul 69%; tuttavia recall ed f-measure, pur risultando ancora molto bassi, aumentano notevolmente rispettivamente al 43% e al 58%, la precision passa dal 43% al 89%.
- U2R:
 - Solito boost di performance su stratified CV, con valori di metriche compresi tra il 95% e il 99%. In questo caso anche sul test set si ha un ottimo beneficio dall'oversampling. L'accuracy apparentemente diminuisce passando dal 99% al 83%, ma risulta però essere supportata da buoni valori sulle altre metriche: la precision passa dal 77% al 97%, la recall dal 25% al 68% e f-measure da 38% a 80%. Tali risultati sono ottenuti senza feature selection e senza standard scaling, applicando queste due tecniche si ha uno scenario leggermente peggiore.

In tutte le tecniche successive, l'oversampling ha portato ad un netto miglioramento delle tecniche di validazione, sia su training set che sul validation set, con valori compresi in ogni caso tra il 92% e il 99%; pertanto, andremo ad evidenziare soltanto eventuali differenze nei risultati ottenuti sul test set.

4.2.2 Logistic Regression

- Probe:
 - Ancora una volta su probe si è beneficiato dell'oversampling, insieme allo StandardScaler e alla feature selection, ha incrementato precision, recall ed f-measure che sono passate rispettivamente da 90%, 77% e 83% a 96%, 90% e 93%, resta invariata l'accuracy.
- R2L:
 - Come per SVM le prestazioni migliorano molto ma non risultano essere soddisfacenti, l'accuracy si assesta sul 72%. Tuttavia recall ed f-measure non sono più inferiori al 10%, bensì risultano rispettivamente al 51% e al 65%, la precision arriva ad 88%.
- U2R:
 - Si beneficia molto dall'oversampling ancora una volta, l'accuracy diminuisce dal 99% al 83% ma è supportata da buoni valori sulle altre metriche, la precision passa dal 77% al 91%, la recall dal 25% al 72% e f-measure da 38% a 81%.

4.2.3 KNN

- Probe:
 - Come nei casi precedenti, probe beneficia dell'oversampling, l'accuracy peggiora, passando dal 91% al 85%. Tuttavia c'è un miglioramento sulle altre metriche, la precision passa da 88% a 96%, recall passa da 67% a 73% e f-measure da 76% a 83%.
- R2L:
 - Su R2L otteniamo i miglioramenti come già visto precedentemente, l'accuracy passa dal 77% al 64%, precision sale al 68% e recall ed f-measure rispettivamente 53% e 60%.

- U2R:
 - Ancora una volta un ottimo miglioramento, accuracy scende dal 99% al 76%, precision resta al 99%, ma recall ed f-measure salgono da valori inferiori al 20% rispettivamente al 53% e 69%.

4.2.4 Decision Tree

- Probe:
 - Buon margine di miglioramento: l'accuracy resta del 90%, la precision passa dal 85% al 94%, la recall dal 63% al 84% e f-measure dal 72% al 89%, utilizzando feature selection e StandardScaling si ottengono i medesimi risultati.
- R2L:
 - Come negli altri casi si hanno dei miglioramenti: accuracy e precision scendono a risultati più veritieri, rispettivamente 67% e 85% mentre recall ed f-measure risultano essere del 41% e 55%. Effettuare feature selection e StandardScaling porta comunque ad ottimi risultati, seppur leggermente inferiori a quelli visti poc'anzi.
- U2R:
 - Miglioramento netto: le performance risultano essere decisamente soddisfacenti, riuscendo ad ottenere 87% accuracy, 94% precision, 79% recall e 86% f-measure. Nuovamente, effettuare feature selection e scaling porta comunque ad ottimi risultati, ma leggermente inferiori.

4.2.5 Random Forest

- Random Forest mostra benefici dall'oversampling soltanto su probe, si passa infatti da una accuracy del 90% al 85%, tuttavia la precision passa dal 87% al 96%, la recall dal 59% al 74% e f-measure dal 71% al 83%.

4.2.6 Neural Network

- Probe:
 - L'oversampling porta ad un buon miglioramento, la precision passa da 87% al 96%, la recall da 67% al 84% e f-measure da 76% a 89%.
- R2L:
 - Le performance ottenute mediante oversampling, feature selection e standard scaling sono decisamente migliori, si ottiene 80% accuracy, 83% precision, 76% recall e 79% f-measure.
- U2R:
 - Si ha un buon miglioramento, eliminando valori sotto il 10%, si ottiene infatti 79% accuracy, 98% precision, 60% recall e 74% f-measure.

4.3 Risultati ottenuti applicando One Hot Encoding

In questa sezione andremo a vedere in quali casistiche di attacco o in quali algoritmi di ML sono stati ottenuti risultati migliori utilizzando tutte le features del dataset, dopo aver applicato OneHotEncoding per gli attributi categorici. In questa sezione tralascieremo i risultati ottenuti dalle tecniche di validazione in quanto quasi del tutto simili a quelli ottenuti senza OHE, andremo, dunque, a considerare solo le performance ottenute sul test set. Gli attacchi o gli algoritmi non presenti in questa sezione hanno ottenuto risultati peggiori o uguali a quelli analizzati fino ad ora nella sperimentazione.

4.3.1 Support Vector Machine

- Dos:
 - Si ha un leggero miglioramento di circa un punto percentuale in ogni metrica rispetto al caso con 38 features, si ottiene infatti 89% accuracy, 92% precision, 82% recall e 87% f-measure.

4.3.2 Logistic Regression

- Dos:
 - Si ha un miglioramento generale di performance rispetto alla casistica con 38 features, si ha 90% accuracy, 93% precision, 84% recall e 88% f-measure. I risultati sono ottenuti applicando anche StandardScaler.
- U2R:
 - Le performance risultano comunque presentare il solito problema di recall e f-measure poco soddisfacenti. Tuttavia, seppur presentando ancora delle percentuali basse, c'è un buon miglioramento: si ha, infatti, 99% accuracy, 69% precision, 40% recall e 50% f-measure. I risultati sono ottenuti applicando anche StandardScaler.

4.3.3 KNN

- Dos:
 - Otteniamo anche con questo algoritmo un leggero miglioramento di performance rispetto all'utilizzo delle sole 38 features per la casistica Dos, i risultati infatti mostrano 90% accuracy, 98% precision, 79% recall e 88% f-measure. I risultati sono ottenuti applicando anche StandardScaler.
- U2R:
 - Come per Logistic Regression anche in questa casistica non otteniamo performance soddisfacenti. Tuttavia, c'è un buon miglioramento: si ha, applicando StandardScaler, 99% accuracy, 82% precision, 35% recall e 50% f-measure.

4.3.4 Decision Tree

- Dos:
 - C'è un buon miglioramento, soprattutto su recall ed f-measure: si ha 86% accuracy, 96% precision, 71% recall e 82% f-measure.

- Probe:
 - Anche in questa casistica di attacco si hanno buoni miglioramenti su ogni performance in particolare ancora una volta su recall e f-measure: si ha 94% accuracy, 90% precision, 80% recall e 85% f-measure.

4.3.5 Random Forest

Con Random Forest l'utilizzo di tutte le features e OneHotEncoding non porta a nessun particolare beneficio in nessuna classe di attacco.

4.3.6 Neural Network

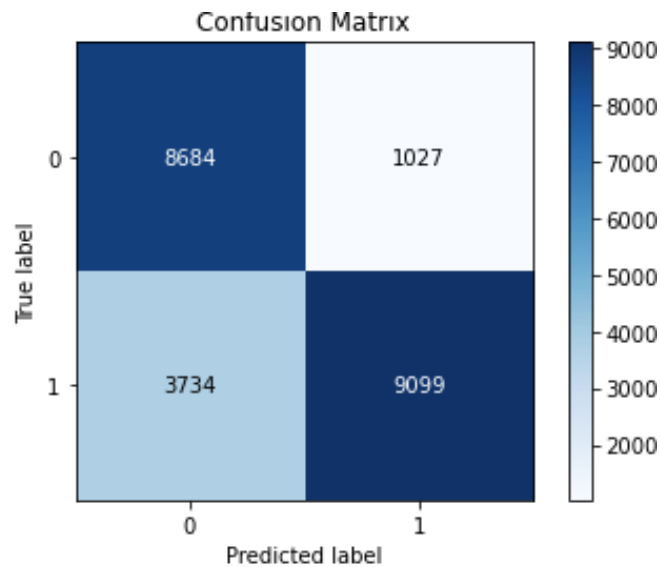
- Dos:
 - La rete neurale beneficia di un buon miglioramento utilizzando tutte le features abbinate allo StandardScaler: si passa dal 91 al 93% di accuracy, dal 93% al 98% di precision, dal 89 al 92% di f-measure, la recall resta invariata.
- Probe:
 - In questa tipologia di attacco l'utilizzo di tutte le features porta ad un leggero aumento di recall ed f-measure che passano dal 66 e 73% al 82 e 77%. Tuttavia la precision subisce un calo passando dal 81% al 73%.

4.4 Risultati ottenuti su Binary Dataset

In questa sezione andiamo ad illustrare i risultati che sono stati ottenuti sul dataset per ogni tecnica di ML applicata al test set tenendo conto soltanto di due classi: traffico normale e traffico durante un attacco, senza dunque entrare nella specifica tipologia di attacco.

4.4.1 Support Vector Machine

- Sul dataset binario SVM ottiene le performance migliori utilizzando tutte le features del dataset e lo StandardScaler, le metriche risultanti sono: 78% accuracy, 89% precision, 70% recall e 79% f-measure.

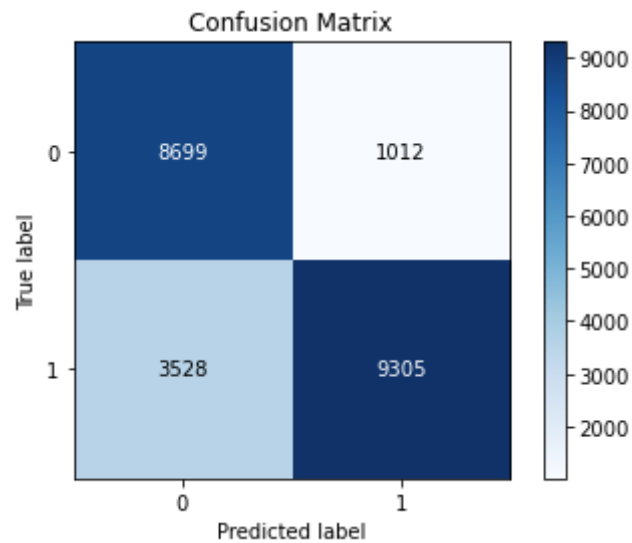


(a)

Figura 4.29: SVM, binary test set.

4.4.2 Logistic regression

- Anche per LR i risultati migliori si ottengono utilizzando tutte le features e lo StandardScaler, si ha 79% accuracy, 90% precision, 72% recall e 80% f-measure.

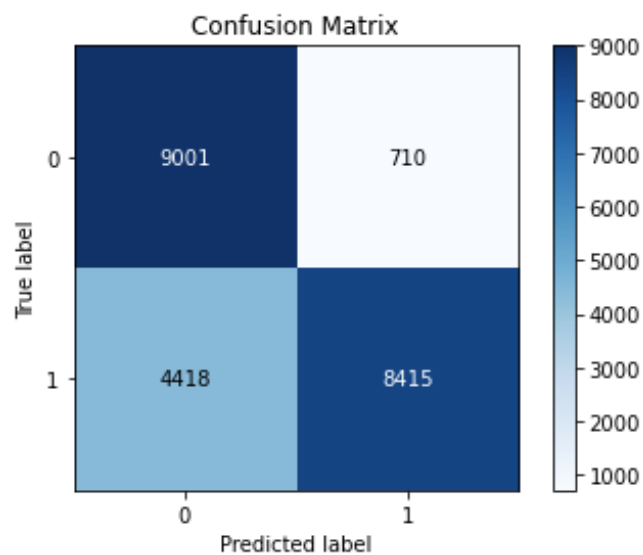


(a)

Figura 4.30: LR, binary test set.

4.4.3 KNN

- Ancora una volta per il dataset binario i migliori risultati si hanno sfruttando tutte le features e applicando lo StandardScaler, con KNN si ha 77% accuracy, 92% precision, 65% recall e 76% f-measure.



(a)

Figura 4.31: KNN, binary test set.

4.4.4 Decision Tree

- Anche nel caso di DT si hanno le performance più alte utilizzando tutte le features e StandardScaler, in particolare si ha 82% accuracy, 80% precision, 92% recall e 85% f-measure.

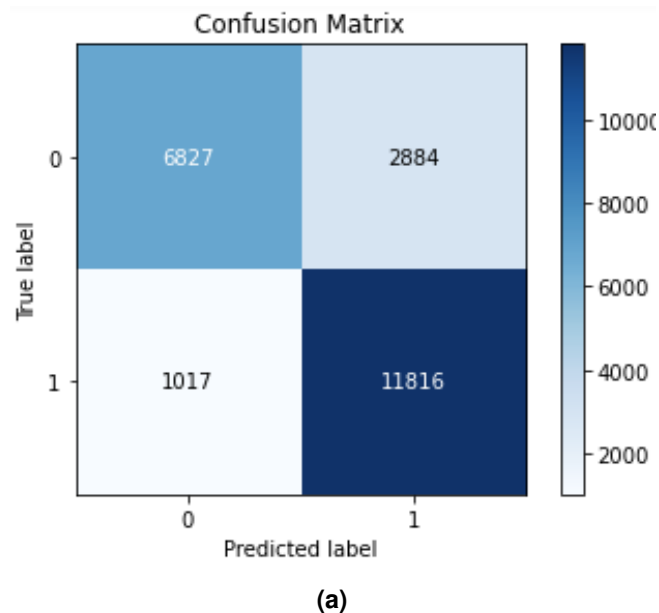
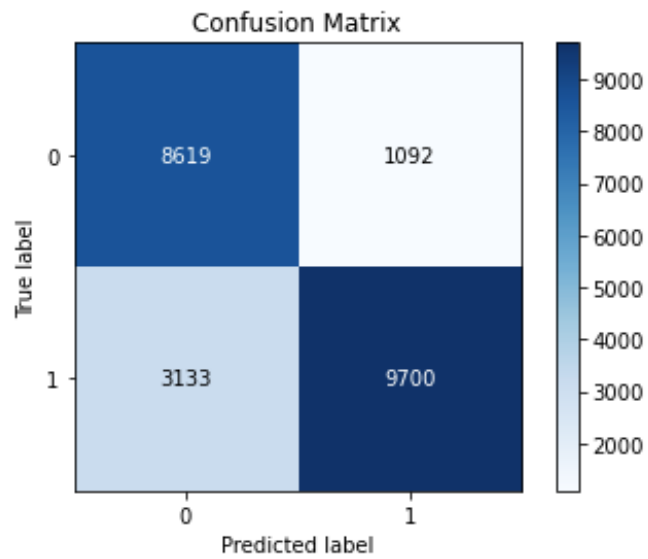


Figura 4.32: DT, binary test set.

4.4.5 Random Forest

- Random Forest performa meglio con la feature selection che, abbinata allo StandardScaler, fornisce performance molto simili a DT, ottenendo 81% accuracy, 89% precision, 75% recall e 82% f-measure.

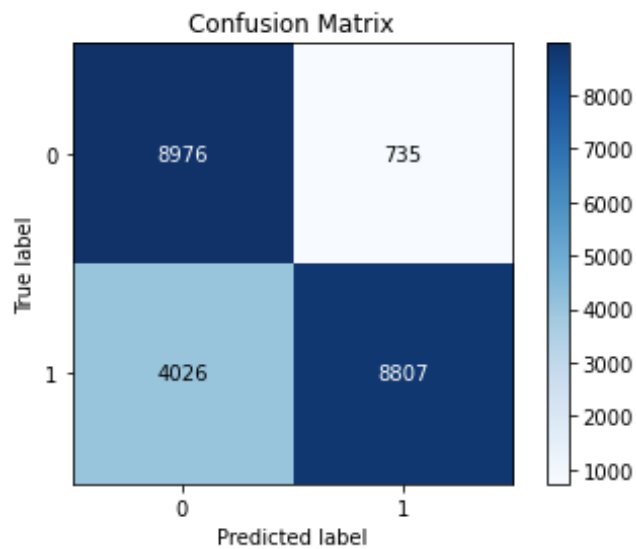


(a)

Figura 4.33: RF, binary test set.

4.4.6 Neural Network

- La rete neurale performa meglio utilizzando tutte le features e lo Standard-Scaler, si ha 78% accuracy, 92% precision, 68% recall e 78% f-measure.



(a)

Figura 4.34: NN, binary test set.

CAPITOLO 5

Conclusioni

Questo capitolo illustra quanto è emerso dalla sperimentazione, discutendo dei risultati più significativi. Sono, inoltre, riportate le limitazioni incontrate e alcuni possibili sviluppi futuri. Nel caso si volesse riprodurre la sperimentazione effettuata, il codice si può trovare al seguente link GitHub: <https://github.com/DanielePalmieri/Tesi-magistrale>

5.1 Interpretazione dei risultati

Nel lavoro di tesi condotto lo scopo era quello di andare a verificare il comportamento di tecniche di Machine Learning applicate al task di attack detection per la sicurezza nei sistemi IoT, tenendo conto di alcune criticità e alcune considerazioni emerse dalla SLR su delle sperimentazioni che sono state svolte in letteratura. Le variabili inserite nella sperimentazione sono state molteplici: è stato considerato un test set composto da tipologie di attacchi aggiuntive non contenute nel training set, è stato applicato uno scaling ai dati, è stata utilizzata una tecnica di oversampling, è stata effettuata una feature selection, sono state utilizzate due differenti tecniche di validazione e sono stati effettuati sia test droppando le features categoriche che test applicando OneHotEncoding a tali features; infine è stata condotta una valutazione sia sulle singole tipologie di attacco che su una distinzione binaria del dataset

etichettando i campioni in: "traffico regolare" e "attacco". Dai risultati ottenuti è ben visibile come l'utilizzo dell'oversampling porta a dei miglioramenti che, in alcuni casi, sono anche molto importanti; in particolare sulla classe di attacco U2R si ha un miglioramento netto delle performance, si passa infatti da percentuali per precision e recall inferiori al 10% a valori compresi tra il 70 e l'80%. Anche le due tecniche di validazione mostrano dei risultati differenti utilizzando l'oversampling: nonostante stratified cross validation sia una tecnica utilizzata nel caso di dataset sbilanciati, in questa particolare situazione lo sbilanciamento è troppo marcato. La tecnica, infatti, pur mantenendo le proporzioni dei fold bilanciate, risulta avere in questi ultimi troppi pochi esempi della classe di attacco; per cui la differenza con k-fold cross validation è irrilevante e l'accuracy ottenuta risulta essere non attendibile. Tuttavia, applicando l'oversampling, il numero fisso di campioni di classe d'attacco presente nei fold, garantito da stratified cross validation, ha permesso di ottenere risultati migliori rispetto a k-fold cross validation, Figura 5.1.

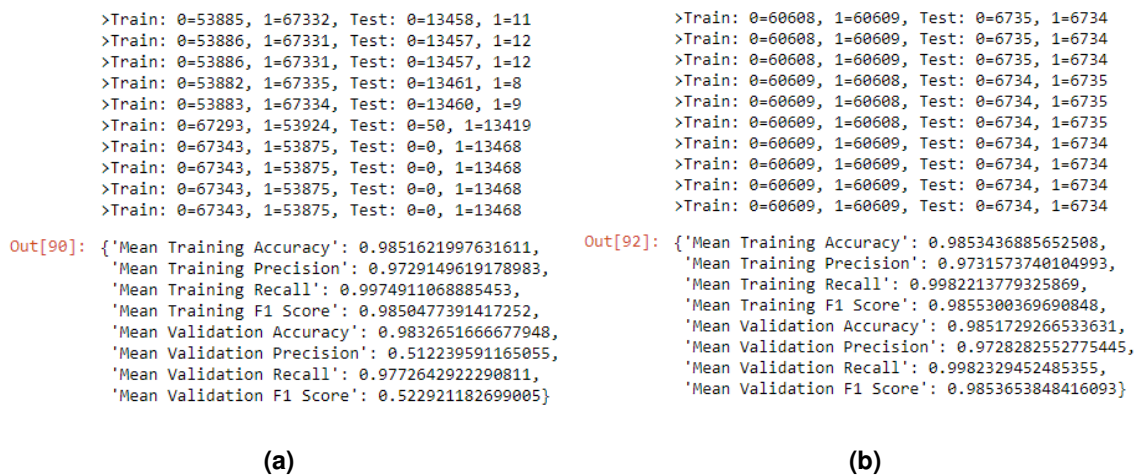


Figura 5.1: U2R: k-fold cross validation(a), stratified cross validation.(b)

Altri accorgimenti come lo standard scaling e la features selection portano a dei risultati che, per determinati algoritmi, sono migliori, mentre, in altri, portano ad un peggioramento delle performance. Ad esempio Random Forest e Decision Tree risultano performare peggio, seppur leggermente, applicando lo standard scaler; mentre sugli altri algoritmi si ha un miglioramento che, soprattutto sulle tecniche k-fold cross validation e stratified cross validation, risulta essere sensibilmente rilevante.

L'utilizzare tutte le features applicando OHE è sì è tradotto in un miglioramento delle performance soltanto per alcune tipologie di attacchi, in particolare Dos e non su tutte le tecniche di ML. Inoltre, in alcuni scenari, all'aumento di percentuale ottenuto in alcune metriche, corrispondeva un abbassamento di altre. In generale possiamo dire dunque che applicare OHE non risulta particolarmente rilevante. Dai risultati ottenuti è ben visibile come seppur ci siano delle differenze di performance tra le varie tecniche di ML impiegate, tali differenze non sono molto marcate; bensì impatta molto di più il preprocessing che viene applicato al dataset. La differenza più marcata è tra le performance ottenute sul validation set (ottenuto come porzione del training set) e il test set: sul validation set le metriche di valutazione raggiungono dei risultati decisamente soddisfacenti ed in linea con quelli già visti in alcuni dei papers presenti in letteratura, sul test set (contenente tipologie di attacchi differenti da quelle del training set), invece, le performance sono decisamente peggiori, seppur comunque accettabili. Tuttavia in letteratura l'utilizzo del test set costruito in questo modo non è molto presente, bensì si fa riferimento più alle performance delle tecniche di validazione applicate al validation set, che abbiamo visto essere migliori, ma meno rilevanti in uno scenario d'uso reale.

Alla luce di quanto visto, è chiaro che le tecniche di ML possono sicuramente essere utili per gestire il task di attack detection, nonostante le performance effettive non possono dirsi pienamente soddisfacenti. Tuttavia la fase di preprocessing dei dati è cruciale per ottenere dei risultati che siano quantomeno discreti ed è importante costruire una buona pipeline che includa quanti più accorgimenti possibili mirati a risolvere criticità nei dati. Oltre al preprocessing, tuttavia, è importante non sottovalutare l'utilizzo di più metriche di valutazione dei risultati. Durante la sperimentazione, infatti, è stato visto nero su bianco come il solo utilizzo dell'accuracy come metrica di valutazione non è affidabile, in più circostanze ci si è imbattuti in percentuali molto alte di accuracy che sono risultate, tuttavia, totalmente falsate dallo sbilanciamento del dataset e, adottando ulteriori metriche, è stato possibile vedere come in realtà alcune delle classi di attacco non venivano classificate per nulla in modo soddisfacente. Infine la differenza di performance tra il validation set e il test set ci suggerisce che i risultati sperimentali ottenuti non è detto che si riflettano allo stesso modo su casi d'uso reale, bensì il calo di performance potrebbe essere rilevante.

5.2 Risultati finali per le domande di ricerca

Dopo aver dato un'interpretazione generale ai risultati ottenuti, andiamo a rispondere schematicamente alle research question che ci siamo posti inizialmente.

RQ1. I dataset adottati in letteratura sono sbilanciati. È possibile adottare un dataset differente? è possibile adottare tecniche in pipelining per sopperire allo sbilanciamento dei dataset già visti?

- Il dataset maggiormente utilizzato in letteratura abbiamo visto essere KDD Cup 99. Sfortunatamente non sono disponibili tanti dataset alternativi e quelli che sono disponibili è raro che non siano sbilanciati. Tuttavia l'adozione di NLS-KDD è da preferire in quanto risulta essere la versione migliore di KDD Cup 99.

Durante la fase di sperimentazione abbiamo visto che si riesce a tamponare lo sbilanciamento mediante tecniche di pipelining quali l'oversampling. Adottare questa tecnica, infatti, ha portato ad un miglioramento netto delle performance in alcune delle classi più sbilanciate.

RQ2. Possiamo adottare anche tecniche che non siano di shallow machine learning?

- Nonostante la differenza non sia stata marcata, abbiamo visto empiricamente che un approccio di deep learning, attraverso una rete neurale, porta a dei risultati decisamente buoni, in alcuni casi anche migliori rispetto a tecniche di shallow machine learning.

RQ3. Per la validazione dei modelli, quali differenze ci sono utilizzando diverse tecniche di validazione?

- L'utilizzo della stratified cross validation è stato decisamente rilevante. Abbiamo visto, soprattutto nel caso dell'oversampling che abbiamo discusso nella sezione 5.1, che mantenere un bilanciamento delle classi nella creazione dei fold porta a dei risultati migliori rispetto alla classica k-fold cross validation.

RQ4. Oltre all'accuracy, quali insights è possibile ottenere utilizzando altre metriche di valutazione?

- Nello studio sono state utilizzate accuracy, precision, recall e f-measure per la valutazione dei risultati. Durante tutta la fase di sperimentazione è stato palese come la sola accuracy non sia una metrica affidabile. In molti casi, infatti, al presentarsi di un'accuracy che raggiungeva anche il 99% corrispondeva una precision o una recall inferiore al 10%; questo è un chiaro segno di come la percentuale di accuracy sia stata influenzata dallo sbilanciamento del dataset e quindi risulta essere irrilevante. In generale è risultata fondamentale l'adozione di più tecniche di valutazione per avere un quadro più parlante dei risultati.

5.3 Limitazioni e sviluppi futuri

Una delle limitazioni incontrate nel lavoro di tesi è stata sicuramente legata al dataset impiegato, NLS-KDD, infatti, per quanto migliore di KDD Cup 99, ha comunque le sue problematiche tra cui la più grande risiede nello sbilanciamento. Anche se questa caratteristica ci ha permesso di effettuare un confronto applicando l'oversampling, lavorare direttamente su un dataset non sbilanciato sarebbe stato sicuramente più significativo per la comparazione tra le tecniche di ML impiegate e, più in generale, per valutare le performance dell'IA nel task di attack detection. Le possibilità di sperimentazioni ulteriori sono molto ampie; tra i possibili sviluppi futuri, oltre valutare l'impiego di uno o più dataset totalmente differenti, con eventuali confronti anche tra i dataset stessi, si potrebbe sicuramente pensare di utilizzare altre tecniche di ML da aggiungere al confronto, o di incentrare lo studio esclusivamente su un approccio di deep learning, andando a sperimentare in modo più approfondito su reti neurali. Potrebbe essere utile, inoltre, approfondire ulteriormente tutta la fase di preprocessing del dataset, sia sperimentando ulteriori tecniche di scaling dei dati o di feature selection, sia introducendo nella pipeline approcci totalmente nuovi rispetto a quelli adottati in questo studio.

Bibliografia

- [1] F. F. Giammaria Giordano, Fabio Palomba, "On the use of artificial intelligence to deal with privacy in iot systems: A systematic literature review," *Journal of Systems and Software*. [Online]. Available: <https://www.sciencedirect.com/science/article/abs/pii/S0164121222001613> (Citato alle pagine iv, 3, 7 e 10)
- [2] M. I. I. Z. M. H. Mahmudul Hasan, Md. Milon Islam, "Attack and anomaly detection in iot sensors in iot sites using machine learning approaches," *Internet of Things*. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2542660519300241> (Citato alle pagine iv, 13 e 17)
- [3] K. Shiimoto, "Network intrusion detection system based on an adversarial auto-encoder with few labeled training samples," *Journal of Network and Systems Management*. [Online]. Available: https://www.researchgate.net/publication/364240631_Network_Intrusion_Detection_System_Based_on_an_Adversarial_Auto-Encoder_with_Few_Labeled_Training_Samples (Citato alle pagine iv, 17 e 18)
- [4] "Introduction to support vector machines." [Online]. Available: <https://michael-fuchs-python.netlify.app/2019/11/08/introduction-to-support-vector-machines/> (Citato alle pagine iv e 20)

-
- [5] "javatpoint." [Online]. Available: <https://static.javatpoint.com/tutorial/machine-learning/images/random-forest-algorithm2.png> (Citato alle pagine iv e 22)
- [6] "eage." [Online]. Available: <https://www.eage.it/machine-learning/k-nearest-neighbours> (Citato alle pagine iv e 23)
- [7] C. M. Z. D. P. M. Z. A. Meneghello, F., "Iot: Internet of threats? a survey of practical security vulnerabilities in real iot devices," *IEEE Internet of Things Journal* 6. [Online]. Available: <https://ieeexplore.ieee.org/document/8796409> (Citato a pagina 3)
- [8] Y. Q. C. M. W. X. F. J. Fei, J., "The abnormal detection for network traffic of power iot based on device portrait. scientific programming 2020." *Journal of Systems and Software*. [Online]. Available: <https://www.hindawi.com/journals/sp/2020/8872482/> (Citato a pagina 8)
- [9] L. A. S. B. S. Y. W. H. Wang, H., "Federated multi-view spectral clustering." *IEEE Access* 8. [Online]. Available: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9252122> (Citato alle pagine 9 e 14)
- [10] D. A. H. S. H. S. C. Darabian, H., "An opcode-based technique for polymorphic internet of things malware detection," *Concurrency and Computation: Practice and Experience*. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.5173> (Citato a pagina 10)
- [11] V. S. Alam, M.S., "Random forest classification for detecting android malware," *IEEE international conference on green computing and communications and IEEE Internet of Things and IEEE cyber, physical and social computing, IEEE*. [Online]. Available: <https://ieeexplore.ieee.org/document/6682136> (Citato a pagina 10)
- [12] T. Z. Shengchu Zhao, Wei Li and A. Y. Zomaya, "A dimension reduction model and classifier for anomaly-based intrusion detection in internet of things," *International Journal of Distributed Sensor Networks*. [Online]. Available: <https://ieeexplore.ieee.org/document/8328485> (Citato a pagina 10)

- [13] T. R. L. X. L. G. Jiang, L., "On lightweight privacy-preserving collaborative learning for internet-of-things objects," *Proceedings of the International Conference on Internet of Things Design and Implementation*. [Online]. Available: <https://dl.acm.org/doi/10.1145/3302505.3310070> (Citato a pagina 12)
- [14] M. S. Bansal, A., "A comparative analysis of machine learning techniques for botnet detection," *Proceedings of the 10th International Conference on Security of Information and Networks*. [Online]. Available: <https://dl.acm.org/doi/abs/10.1145/3136825.3136874> (Citato a pagina 14)
- [15] M. T. E. B. W. L. A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," *IEEE Symposium on Computational Intelligence for Security and Defense Applications*. [Online]. Available: <https://ieeexplore.ieee.org/document/5356528> (Citato a pagina 23)
- [16] "Nsl-kdd data set for network-based intrusion detection systems." [Online]. Available: <https://www.unb.ca/cic/datasets/nsf.html> (Citato a pagina 30)

Ringraziamenti

I ringraziamenti e le dimostrazioni d'affetto non sono mai stati il mio forte tuttavia, a conclusione del percorso, desidero menzionare tutte le persone che mi hanno accompagnato durante questi anni.

Ringrazio in primis il mio relatore, il professore Fabio Palomba che, in questi mesi di lavoro, mi ha sempre seguito dandomi suggerimenti preziosi durante la sperimentazione e nella stesura della tesi. Ringrazio in generale tutti i dottorandi del SeSa Lab per l'organizzazione strepitosa e il supporto costante durante tutte le fasi di laurea.

Ringrazio inoltre la mia famiglia per avermi sostenuto e avermi permesso di portare a termine gli studi universitari senza pressioni o preoccupazioni.

Ringrazio la mia fidanzata Sara sia per avermi aiutato a rivedere grammaticalmente tutto l'elaborato sia per essermi sempre vicina e spronarmi nei momenti più difficili.

Un grazie ad Assia che, nonostante il passare degli anni, mi è stata sempre accanto ed è una persona su cui posso sempre contare.

Ringrazio i "TSF": Silvio, Luciano, Antonio e Lorenzo. Senza di loro tutta l'esperienza universitaria sarebbe stata probabilmente più rapida ma sicuramente più pesante. La certezza di vedersi nella Landa degli Evocatori dopo le lezioni ha reso questi anni più spensierati.

Ringrazio i "soliti sospetti" per essere quegli amici di una vita che, nonostante le distanze e il sentirsi poco, all'occorrenza ci sono sempre. Un grazie di cuore quindi ad Alfonso, Ciccio, Geremia, Luca, Andrea Americo e Leonardo.

In questi ringraziamenti inserisco anche Valerio che, pur non essendo un sospetto a tutti gli effetti, è un caro amico di vecchia data che ho avuto il piacere di incontrare nuovamente durante l'Apple Academy e che continua ad accompagnarmi nei pomeriggi nella Landa degli evocatori.

Menzione d'onore va fatta sicuramente anche al gruppo "Mtg KVM" che sono riusciti a tenermi sempre distratto facendomi partecipare a tornei di Magic in giro per l'Italia il giorno prima degli esami, facendomi fare orari indicibili e causandomi fallimenti universitari che, tuttavia, sono stati di grande aiuto ad affrontare tutto il percorso in maniera più serena.

Questa tesi ha contribuito a piantare un albero in Colombia tramite il progetto Treedom.

<https://www.treedom.net/it/user/sesalab/event/sesa-random-forest>