Intrusion Detection: Perceptron e Random Forest a confronto

NICCOLÒ BOANINI

Aprile 2024

INDICE

ELENCO DELLE TABELLE

SOMMARIO

In questa relazione si descrive l'utilizzo di implementazioni disponibili di Perceptron e Random Forest di scikit-learn [1] al problema dell'Intrusion Detection, riproducendo i risultati delle tabelle 3 e 4 presenti in [2], utilizzando rispettivamente Perceptron e Random Forest al posto di Naive Bayes e Decision Tree. I dati provengono dalla pagina della KDD Cup 1999 [3] e sono state tratte ulteriori informazioni sul dataset da [4].

1 INTRODUZIONE

Un software di rilevazione delle intrusioni di rete (*intrution detection software*) protegge una rete informatica da utenti non autorizzati. Il compito del rilevatore in termini di apprendimento è quello di costruire un modello predittivo, ovvero un classificatore, in grado di distinguere le connessioni malevoli (*attacks*) da quelle "buone" (*normal*). Ogni record del dataset [3] corrisponde a una connessione, circa 100 bytes di informazione suddivisi tra 41 *features* e una *label*.

- Le *features* includono informazioni di base della connessione TCP (indirizzo IP, porta, stato) e caratteristiche del traffico sulla rete. In generale, possono essere valori numerici (*continuous*) oppure stringhe (*symbolic*).

- La *label* è univoca per ciascun record e sono presenti in tutto 39 classi possibili (*normal*+38 tipologie di attacchi, che ricadono in 4 categorie principali: DOS, R2L, U2R, Probing).

  Per le nostre analisi, considereremo proprio il caso con 5 categorie di attacchi (*normal*+le suddette quattro) effettuando un *gathering before classification*, cioè una modifica del dataset che consiste nel raggruppamento degli attacchi appartenenti alla medesima categoria. Per un completo dizionario delle tipologie d'attacco si consulti [4].

Gestiremo il 10% dell'intero dataset KDD'99, corrispondente a circa mezzo milione di connessioni di addestramento e oltre trecento mila connessioni di test e utilizzeremo Random Forest e Perceptron come classificatori, sfruttando le relative implementazioni disponibili in `scikit-learn`.

2 IMPLEMENTAZIONE

Di seguito si riporta la procedura implementativa adottata (replicabile) per arrivare all'ottenimento dei risultati.

CARICAMENTO DEL DATASET. Sono stati caricati (in locale) i datasets di train e di test disponibili sulla pagina della KDD Cup 1999 [3], in particolare le versioni ridotte (*10 percent*), come eseguito in [2]. Successivamente si sono assegnati i nomi delle colonne (features) per facilitare future indicizzazioni.

GATHERING. Prima di procedere con le classificazioni, si è effettuato il *gathering*, modificando quindi il dataset con la raccolta degli attacchi in cinque categorie.

PREPROCESSING. Per preparare i dati al processo di classificazione, è stato applicato il *preprocessing* utilizzando il `MinMaxScaler` e il `LabelEncoder`. Il primo è stato impiegato per normalizzare le feature, riducendo l'impatto delle differenze di scala tra di esse. Il secondo è stato cruciale per codificare le etichette dei dati, trasformandole in valori numerici e rendendo così possibile l'addestramento dei modelli di machine learning utilizzati.

ADDESTRAMENTO DEL MODELLO. Il modello è stato addestrato sul training set utilizzando Random Forest e Perceptron. In quest'ultimo caso, sono state utilizzate due versioni differenti: quella con il classificatore OneVsRest e quella con il OneVsOneClassifier.

VALUTAZIONI DELLE PERFORMANCE. Per replicare la Tabella 2 di [2], si è valutata la performance (score in percentuale) del modello addestrato sul Training Set e sul Testing Set.

PREDIZIONI. Si sono quindi effettuate predizioni sui dati di test utilizzando il modello addestrato e per riprodurre la Tabella 3 di [2] sono state stampate le matrici di confusione e il PCC (Percent of Correct Classification).

3 RISULTATI OTTENUTI

Nelle tabelle 1, 2 e 3 sono riportati i risultati ottenuti utilizzando le implementazioni di Random Forest e Perceptron, quest'ultimo in entrambe le versioni OneVsOne e OneVsRest[1].

1 Il modello Perceptron (One vs One) crea un classificatore binario per ogni coppia di classi, mentre il modello Perceptron (One vs Rest) crea un classificatore per ogni classe rispetto al resto.

Tabella 1: PCC (Percent of Correct Classification)

TRAINING SET	TESTING SET
Random Forest	
99.9984%	92.4081%
Perceptron (One vs One)	
99.6652%	91.9831%
Perceptron (One vs Rest)	
99.3909%	92.0438%

Tabella 2: Matrice di confusione di Random Forest

	Normal	DOS	R2L	U2R	Probing
Normal (60593)	**99.55% (60318)**	0.09% (57)	0.00% (1)	0.00% (2)	0.35% (215)
DOS (229853)	2.77% (6358)	**97.23% (223489)**	0.00% (0)	0.00% (0)	0.00% (6)
R2L (16189)	96.92% (15691)	0.00% (0)	**3.05% (494)**	0.02% (4)	0.00% (0)
U2R (228)	92.11% (210)	0.00% (0)	0.88% (2)	**1.75% (4)**	5.26% (12)
Probing (4166)	20.04% (835)	5.28% (220)	0.00% (0)	0.00% (0)	**74.68% (3111)**

Tabella 3: Matrici di confusione di Perceptron

Perceptron One-vs-One					
	Normal	DOS	R2L	U2R	Probing
Normal (60593)	**99.70% (60413)**	0.15% (89)	0.04% (22)	0.00% (0)	0.11% (69)
DOS (229853)	2.93% (6732)	**97.05% (223068)**	0.00% (0)	0.00% (0)	0.02% (53)
R2L (16189)	99.89% (16171)	0.02% (3)	**0.09% (15)**	0.00% (0)	0.00% (0)
U2R (228)	97.37% (222)	2.19% (5)	0.44% (1)	**0.00% (0)**	0.00% (0)
Probing (4166)	34.18% (1424)	3.46% (144)	0.00% (0)	0.00% (0)	**62.36% (2598)**

Perceptron (One vs Rest)					
	Normal	DOS	R2L	U2R	Probing
Normal (60593)	**99.67% (60396)**	0.18% (110)	0.01% (8)	0.00% (0)	0.13% (79)
DOS (229853)	2.89% (6647)	**97.10% (223176)**	0.00% (0)	0.00% (0)	0.01% (30)
R2L (16189)	99.92% (16176)	0.02% (4)	**0.05% (8)**	0.00% (0)	0.01% (1)
U2R (228)	95.61% (218)	3.51% (8)	0.88% (2)	**0.00% (0)**	0.00% (0)
Probing (4166)	30.46% (1269)	4.66% (194)	0.00% (0)	0.00% (0)	**64.88% (2703)**

4 ANALISI DEI RISULTATI OTTENUTI

Tutti i classificatori utilizzati hanno una buona capacità di adattamento ai dati, con accuratezze di addestramento superiori al 99%. Nei test set, il Random Forest mostra l'accuratezza più elevata (92.4081%), seguito dal Perceptron One vs Rest (92.0438%) e infine dal Perceptron One vs One (91.9831%). Sebbene ci sia una leggera differenza nell'accuratezza tra i tre classificatori, tutte e tre le varianti hanno prestazioni comparabili.

SENSIBILITÀ DELLE CLASSI Perceptron e Random Forest sembrano avere una buona capacità di distinguere le connessioni normali dagli attacchi, come indicato dall'alta precisione nella classe "Normal". Tuttavia, tutti i modelli mostrano una maggiore difficoltà nel distinguere tra i diversi tipi di attacchi, come evidenziato dalle percentuali di predizioni errate soprattutto delle classi R2L, U2R e Probing. Questo è dovuto al fatto che le occorrenze di questi attacchi nel training set sono molto poche in percentuale (inferiori allo 0.80% in tutti e tre i casi). I dati sono cioè sbilanciati. In dettaglio, Perceptron sembra avere maggiori difficoltà nel catturare in modo efficace le relazioni nei dati in queste categorie minoritarie, arrivando a classificare erroneamente tutti gli attacchi di tipo U2R che sono quelli più rari. Un modello più complesso come Random Forest riesce seppur di poco a migliorare la situazione e riuscendo a classificare correttamente un numero maggiore di attacchi in tutte e cinque le categorie.

NOTA SUI MODELLI DI PERCEPTRON UTILIZZATI Sebbene il modello Perceptron (One vs One) sia più complesso in termini di numero di classificatori binari creati rispetto al modello Perceptron (One vs Rest), la maggiore complessità ha portato a una migliore capacità di discriminazione tra le classi, specialmente per le categorie minoritarie come U2R e R2L.

5 CONCLUSIONI

- **Performance Generale**: Tutti e tre i modelli hanno dimostrato di essere in grado di classificare le connessioni di rete con un'accuratezza complessivamente elevata anche sui test set. Tuttavia, ciascun modello presenta vantaggi e limitazioni specifiche a seconda del contesto.
- **Random Forest**: Si è distinto per la sua elevata accuratezza globale e la capacità di gestire in modo efficace classi minoritarie come U2R e R2L.
- **Perceptron One vs One e One vs Rest**: Entrambi hanno presentato risultati simili in termini di accuratezza complessiva. Tuttavia, il Perceptron One vs One potrebbe essere più adatto per affrontare specificamente le classi minoritarie perché mostra maggiore sensibilità considerando coppie di classi.

REFERENCES

[1] F. Pedregosa and Varoquaux et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[2] Amor et al. Naive bayes vs decision trees in intrusion detection systems. *Proceedings of the ACM Symposium on Applied Computing*, 2004.

[3] Irvine University of California. Kdd cup dataset. 1999.

[4] Mehdi Hosseinzadeh Aghdam and Peyman Kabiri. Feature selection for intrusion detection system using ant colony optimization. *International Journal of Network Security*, 2016.