Naive Bayes for Intrusion Detection

Niki Di Costanzo

4 settembre 2020

1 Introduzione

In questo esercizio è stato utilizzato l'algoritmo Naive Bayes di *scikit-learn* per il problema del rilevamento delle intrusioni, che riguarda il processo di monitoraggio e analisi degli eventi che si verificano in un sistema informatico al fine di rilevare segni di problemi di sicurezza.

2 Dataset KDD Cup 1999

Nel dataset KDD Cup 1999 sono presenti connessioni TCP / IP, descritte da 41 funzioni discrete e continue ed etichettate come normali o come attacchi. Gli attacchi rientrano in quattro categorie principali:

- Denial of service (DOS)
- User to root attacks (U2R)
- Remote to user attacks (R2L)
- Probing (Probe)

Nell'esperimento, riportato nell'articolo, è stato usato solo il 10% dell'intero dataset che corrisponde a 494019 connessioni di training e 311029 connessioni di test. I dataset sono stati scaricati dalla pagina della KDD Cup 1999; per training set è stato usato *kddcup.data_10_percent.gz*, per il test set *corrected.gz*.
Il test set non ha la stessa distribuzione di probabilità del training set, infatti include tipi di attacco non presenti in quest'ultimo. Il training set contiene 24 tipi di attacchi, invece il test set 38; ciò rende gli attacchi più realistici.

3 Procedimento

In questa relazione si considera il caso delle cinque classi, in cui si raggruppano i vari attacchi nelle 4 categorie e si è interessati a sapere a quale categoria appartiene una determinata connessione.
Ci sono due strategie per raccogliere i risultati prima o dopo la classificazione:

- **Raccolta prima della classificazione:** l'idea è di modificare leggermente il dataset raggruppando gli attacchi appartenenti alla stessa categoria di attacco.
- **Raccolta dopo la classificazione:** il training set rimane invariato. Tuttavia, ogni connessione, classificata in una dei 38 attacchi, è assegnato a una delle quattro categorie a cui appartiene.

Nel training set il numero di istanze U2R è molto piccolo (0,01%, ovvero 225). Per garantire il loro minimo apprendimento sono stati duplicati fino ad avere il 0,22% (1035) di istanze di training. Ciò ha avuto un'influenza molto piccola sulle istanze U2R e nessuna sulle altre istanze.

3.1 Preprocessing

Per estrarre i dati dal dataset è stata usata la funzione *read_ csv()* della libreria pandas.
Nel dataset sono presenti dati categorici, discreti e continui:

- I dati categorici sono stati convertiti in numeri utilizzando *LabelEncoder* di scikit-learn.
- Gli attributi continui sono stati discretizzati utilizzando *KbinDiscretize*. Come codifica è stata usata *onehot-dense* e come bin è stato scelto un valore che fornisse prestazioni migliori.

3.2 Classificazione

Il classificatore Naive Bayes si basa sull'applicazione del teorema di Bayes, con l'assunzione *naive* dell'indipendenza condizionale tra ogni coppia di caratteristiche dato il valore della variabile di classe.
Questa analisi si basa sul classificatore bayesiano, *GaussianNB*, che è stato usato per generare un modello. Per prima è stato creato il classificatore Naive Bayes, che successivamente è stato addestrato usando il training. Infine, il classificatore "istruito" è stato usato per l'analisi del test set.

4 Risultati

La valutazione della classificazione si basa sulla *percentuale di corretta classificazione*(**PCC**) delle istanze appartenenti al training e test set.
La funzione *accuracy_ score* è stata usata per calcolare la PCC.

$$PCC = \frac{TP+TN}{TP+TN+FP+FN}$$

La tabella 1 mostra le PCC relative alle sperimentazioni fatte prima e dopo la classificazione.

Training set	Test set
94.27% (97.74%)	87.91% (79.25%)

Tabella 1: PCC

Le tabelle 2 e 3 presentano le **matrici di confusione**, che restituiscono una rappresentazione dell'accuratezza di classificazione statistica. Ogni colonna della matrice rappresenta i valori predetti mentre ogni riga rappresenta i valori reali. L'elemento sulla riga i e sulla colonna j rappresenta il numero di casi in cui il classificatore ha classificato la classe "vera" i come classe j. Attraverso questa matrice è osservabile se vi è "confusione" nella classificazione di diverse classi. La tabella 2 mostra la matrice di confusione relative alle sperimentazioni fatte prima, la tabella 3 dopo la classificazione.
Per calcolare le matrici è stata usata la funzione *confusion_ matrix* della libreria*sklearn.metrics*.
La tabella mostra che le connessioni Norma e DOS sono classificate bene con le due tecniche. Questo non è il caso delle connessioni R2L e U2R che sono sempre classificate in modo errato. Ciò è dovuto al fatto che le percentuali nel training set degli attacchi di U2R e R2L sono molto basse(0.22% per U2R e 0.23% per R2L).

	Normal	**DoS**	**R2L**	**U2R**	**Probe**
Normal	**91%**	1.6%	1.6%	5.3%	0.26%
DoS	5.6%	**93%**	0.26%	0.11%	0.96%
R2L	82%	0.006%	**6.3%**	12%	0.012%
U2R	4.4%	0%	66%	**16%**	13%
Probe	14%	2.1%	3.6%	7.4%	**73%**

Tabella 2: Matrice di confusione prima della classificazione

	Normal	**DoS**	**R2L**	**U2R**	**Probe**
Normal	**98%**	0.12%	0.43%	0.45%	1.3%
DoS	20%	**80%**	0%	0.005%	0.24%
R2L	91%	0%	**6.1%**	1.9%	1.4%
U2R	75%	0%	4.4%	**3.5%**	18%
Probe	24%	2.4%	0%	3.6%	**70%**

Tabella 3: Matrice di confusione dopo la classificazione

5 Conclusioni

I risultati presenti nell'articolo sono stati ottenuti utilizzando la stima della densità kernel che ha portato ad un aumento significativo della PCC.
Avendo usando la libreria scikit-learn non è stato possibile usare Naive Bayes insieme alla densità del kernel poiché non è disponibile; per questo è stata usata una distribuzione gaussiana. Per ottenere risultati migliori sono stati discretizzati gli attributi continui; i risultati sono inferiori rispetto a quelli presenti nell'articolo ma comunque soddisfacenti.