Alberi di decisione
nei sistemi di rilevazione delle intrusioni

Scaramuzzino Giovanna

1.Introduzione
L'elaborato assegnato richiedeva di riprodurre i risultati delle tabelle 2 e 3, riguardanti esclusivamente Decision tree, contenuti nell'Articolo "Amor et al. 2004".

1.1.Descrizione Dataset

Il dataset sul quale sono stati calcolati questi risultati è KDD CUP 1999. Di esso, è stato usato solo il 10% corrispondente a 494019 connessioni training e 311029 connessioni di test. Ogni connessione è descritta da 41 caratteristiche discrete e continue e contrassegnata per essere normale o un attacco, con esattamente un tipo specifico di attacco per linea. Gli attacchi son raggruppati in 4 classi: DOS, U2R, R2L, PROBING.

È importante notare che i dati del test non provengono dalla stessa distribuzione di probabilità dei dati di allenamento e includono tipi di attacco specifici non inclusi nei dati di addestramento. Ciò rende il compito più realistico.

2.Implementazione
L'elaborato è stato svolto in Python, versione 3.6.3. Le dipendenze istallate per eseguire il codice implementato sono:

- ScyPy, richiesto per l'istallazione di scikit-learn;
- NumPy, richiesto per l'istallazione di scikit-learn;
- Pandas, modulo per l'analisi dei dati;
- Scikit-learn, contenente gli algoritmi necessari per il classificatore richiesto.

Tutte queste dipendenze sono state istallate tramite il comando da terminale *pip install* seguito il nome della dipendenza desiderata.
Il codice è in formato .ipynb, eseguibile tramite la web app Jupyter Notebook, che consente di condividere documenti che contengono live code.

2.1.Dataset
E' stata usata la funzione **read_csv()** della libreria *pandas* per poter leggere il file CSV sottoforma di Dataframe. Per far ciò è necessario passare a tale funzione il path del file. Inoltre è stato attribuito ad ogni colonna il nome delle rispettive caratteristiche di connessione, passando anch'esso come parametro alla funzione.

2.2.Manipolazione dati
Sono stati creati dei dizionari in modo tale da poter associare ad ogni parola contenuta nel dataset un numero intero. Questo è stato fatto per rendere compatibili i dati in input al classificatore.

L'ultima colonna del Dataframe, sia per quanto riguarda i dati di test che di train, contenuti già in file importati separatamente, comprende la *label* che indentifica se si tratta di una connessione normale o di uno dei 4 tipi di attacchi. Tale colonna è stata opportunatamente separata dai restanti dati.

2.4.Api utilizzate

L'implementazione si basa sul classificatore Decision Tree reperibile in *scikit-learn.*
E' stata usata la funzione **accuracy_score()** di *scikit-learn* per studiare l'accuratezza del train set del test set. Per accuratezza si intende il rapporto tra gli esempi correttamente classificati e gli esempi totali.
Per riprodurre la matrice di confusione è stata invece usata la funzione **cunfusion_matrix()** di *scikit-learn.* La matrice generata è costituita da una riga e una colonna per ogni classe dove ogni colonna della matrice rappresenta i valori predetti, mentre ogni riga rappresenta i valori reali. Dunque, l'elemento sulla riga *i* e sulla colonna *j* è il numero di casi in cui il classificatore ha classificato la classe "vera" *i* come classe *j.*

2.5.Riproduzione dati tabelle

I dati delle tabelle da riprodurre si basano sulla divisione in 5 classi.
Per la raccolta dei dati sono state usate 2 strategie: prima e dopo la classificazione:

- attacchi **prima** della classificazione: sono ottenibili modificando sia i dati di train che di test, raggruppando le connessioni, tramite l'uso dei dizionari, nelle varie classi di appartenenza: DOS, R2L, U2R, Probing e Normal.

- Attacchi **dopo** la classificazione: in questo caso i dati di train non vengono modificati e ogni connessione classificata come 1 dei 38 attacchi è affetta da attacchi della classe cui appartiene dopo la classificazione. Ciò è possibile ottenerlo modificando i dati predetti e di test, assegnando a ciascun attacco la propria classe di appartenenza.

3.Riproduzione dei Risultati

Per eseguire lo script bisogna aver istallato la Jupyter Notebook.
Da linea di comando, aprire la web app tramite il seguente comando:

jupyter notebook

All'esecuzione di tale istruzione, si aprirà il browser in una schermata dove sarà possibile ricercare la cartella del proprio computer nella quale è stato salvato il file. Fare doppio click sul file 'Progetto IA' per aprirlo.
Il file contiene già un run eseguito per aver subito a disposizione i risultati senza dover necessariamente scaricare i dati ed attendere i tempi di esecuzione.
Nel caso si volessero eseguire i run è necessario scaricare dal link http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html:
- dati di train: kddcup.data_10_percent.gz
- dati di test: corrected.gz

Estrarre i file utilizzando WinRAR e sostituire con il proprio percorso quello da me inserito nel codice (in **read_csv()**).

4.Risultati e conclusioni

4.1.Accuratezza dati di train e test

Per quanto riguarda l'accuratezza dei dati di train, è stato ottenuto lo stesso risultato sia nel caso prima che dopo la classificazione.

Nel caso del test invece è stata ottenuta un'accuratezza maggiore pima della classificazione; dunque alcune connessioni corrette prima della classificazione vengono classificate come errate dopo la classificazione.

```
Accuracy of Decision Tree classifier on train set:100.00%

Accuracy of Decision Tree classifier on test set after classification:90.39%

Accuracy of Decision Tree classifier on test set before classification: 92.38%
```

4.2.Confusion Matrix

La matrice di confusione relativa a 5 classi, mostra che le connessioni Normal, Probing e DOS classificano bene, a differenza di R2L e U2R .
Ciò è dovuto al fatto che nei dati di train vi sono pochi attacchi di tipo R2L e U2R, facendo sì che queste classi siano soggette ad un apprendimento debole e dunque ad una errata classificazione delle connessioni di test che le appartengono veramente.

Per PCC(*Percent of Correct Classification)* si intende la valutazione dell'efficienza della classificazione, che è stata calcolata come: somma del numero di attacchi classificati correttamente(si trovano sulla diagonale) diviso il numero totale di attacchi delle 5 classi.

Riporto qui sotto la matrice di confusione ottenuta (tabella 1) e la matrice di confusione del documento (tabella 2), dove i valori tra parentesi sono relativi agli attacchi dopo la classificazione.

→	Normal	DOS	R2L	U2R	Probing
Normal (60593)	**98.15% (99.37%)**	1.37% (0.15%)	0.02% (0.01%)	0.01% (0.05%)	0.45% (0.42%)
DOS (229853)	2.41% (5.25%)	**97.52% (94.55%)**	0.00% (0.09%)	0.00% (0.00%)	0.06% (0.12%)
R2L (16189)	73.76% (88.60%)	0.04% (6.14%)	**2.50% (2.31%)**	20.80% (2.91%)	2.99% (0.04%)
U2R (228)	30.26% (82.46%)	0.00% (0.44%)	2.19% (10.53%)	**9.21% (3.95%)**	58.33% (2.63%)
Probing (4166)	16.78% (19.13%)	3.70% (3.86%)	0.58% (0.02%)	0.26% (0.00%)	**78.68% (76.98%)**
PCC	92.38% (90.39%)				

Tabella 1: Matrice di confusione relativa a 5 classi

→	Normal	DOS	R2L	U2R	Probing
Normal (60593)	**99.50%** (**99.43%**)	0.13% (0.14%)	0.01% (0.02%)	0.01% (0.02%)	0.36% (0.39%)
DOS (229853)	2.76% (2.94%)	**97.24%** (**96.57%**)	0.00% (0.10%)	0.00% (0.00%)	0.00% (0.39%)
R2L (16189)	96.55% (75.77%)	0.02% (2.79%)	**0.52%** (**0.45%**)	0.15% (4.27%)	2.76% (16.71%)
U2R (228)	79.82% (23.25%)	2.63% (0.00%)	1.75% (5.26%)	**7.89%** (**13.60%**)	7.89% (57.89%)
Probing (4166)	19.54% (15.22%)	5.16% (6.67%)	0.34% (0.19%)	0.00% (0.00%)	**74.96%** (**77.92%**)
PCC	92.06% (92.80%)				

Tabella 2: Matrice di confusione relativa a 5 classi con valori ottenuti nel documento

4.3Conclusioni

I risultati ottenuti non sono esattamente quelli delle tabelle del documento. Per quanto riguarda l'accuratezza sui dati di train la variazione a confronto con le tabelle del documento non è superiore allo 0,01%, mentre per i dati di test non è superiore all' 1%. Anche per le matrici di confusione, come è possibile vedere confrontando le tabelle, i valori ottenuti nella tabella 1 si discostano da quelli riportati nella tabella 2.

Tale differenza potrebbe essere dovuta al non aver effettuato la potatura dell'albero di decisione, cosa che possiamo notare in quanto non abbiamo alcun errore sul training set. É stato deciso però di non effettuare tale potatura dell'albero poiché non erano riportate sul documento le specifiche utilizzate per permettermi di ottenere esattamente gli stessi risultati.

Inoltre, l'albero di decisione è stato costruito tramite l'algoritmo di *scikit-learn* che implementa una versione ottimizzata dell'algoritmo CART(Classification And Regression Tree) ed è stata usata l'entropia come criterio di split; anche ciò potrebbe influire sui diversi valori ottenuti, dato che il documento non forniva nessuna informazione a riguardo.

5.Riferimenti

[1] Stuart J.Russell, P.Norving, Arti_cial Intelligence: A Modern Approach, 3rd Edition, Pearson Education,2010

[2] http://scikit-learn.org/stable/documentation.html, documentatione scikit-learn

[3] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, dataset KDD CUP 1999

[4] https://www.semanticscholar.org/paper/Naive-Bayes-vs-decision-trees-in-intrusion-systems-Amor-Benferhat/16a778c5d83cce2f4c4af46efafb927e7d0d8e60, articolo di riferimento