Progetto Intelligenza Artificiale "Predizione con Albero Decisionale"

Lorenzo Italiano - 6134469

20/03/2019

1 Descrizione

Il codice utilizzato permette di fare uso del *DecisionTree* di scikit-learn per determinare se un nuovo impiegato avrá uno stipendio maggiore rispetto alla mediana degli stipendi dei vari impiegati nel dataset.
Il dataset è reperibile su Stack Overflow Developer Survey, 2017.
È necessario scaricare il dataset per il corretto funzionamento del codice.

Sono state usate le seguenti librerie:

- **numpy**, **pandas** per operare sul dataset
- **matplotlib** per disegnare i vari grafici
- **sklearn** per definire il decision tree, per creare la roc curve e per utilizzare il 10-fold cross validation
- **pydotplus**, **graphivz**, **collections** per disegnare il grafo dell'albero

Nella prima parte del codice viene riadattato il dataset per poter essere letto da DecisionTree.

In seguito è stato definito il target ed è stato diviso il dataset in train-set e test-set per determinare la precisione della previsione dell'albero.

È stato inoltre verificato quale fosse il parametro *max_depth* adatto a seconda del train-set, seguendo le indicazioni dell'articolo:
"InDepth: Parameter tuning for Decision Tree" di *Mohtadi Ben Fraj*,
illustrandone il grafico rispetto alla sua variazione.

Infine, dopo aver creato l'albero decisionale ed aver determinato la precisione della previsione mediante il 10-fold cross validation (≈ 0.83), è stato disegnato il grafo dell'albero, seguendo il codice ottenuto dall'articolo:
"Creating and Visualizing Decision Trees with Python" di *Russel*.

1.1 Setup Dataset

Per poter definire il decision tree è risultato essere necessario rimodellare il dataset.

Dopo aver eliminato le righe con valore *NaN* nella colonna Salario, averla estratta dal dataset e aver eliminato le colonne *'ExpectedSalary'* e *'NonDeveloperType'* (poiché una irrilevante e l'altra contenente solo valori *NaN*), sono stati sostituiti i valori *NaN* delle altre colonne: quelle contenenti stringhe con una stringa 'NaN' e quelle contenenti numeri con la mediana degli altri numeri della colonna.

Dopodiché tutte le colonne sono state divise in un numero di colonne pari al numero di risposte date, tutte con valori booleani (**1** se l'impiegato aveva dato quella risposta e **0** altrimenti), inoltre, poiché il dataset contiene risposte multiple, ogni risposta è stata divisa grazie al metodo *str.get_dummies()* della libreria **pandas**.

2 Ricerca del Miglior 'max_depth'

Seguendo l'articolo, citato nella descrizione, di *Mohtadi Ben Fraj* è stato possibile creare un grafico della precisione dell'albero, nella metrica **AUC** (=*Area Under Curve*), al variare del parametro *max_depth*, il quale indica l'altezza massima dell'albero.

Come mostra la **Figure 1**, aumentando l'altezza dell'albero, si ha un incremento sia nel train-set che nel test-set (in quanto riesce a catturare più informazioni), al quale segue un calo nel test-set a causa del fenomeno dell'*overfitting*, che si osserva quando si ha un numero eccessivo di parametri rispetto al numero di osservazioni.

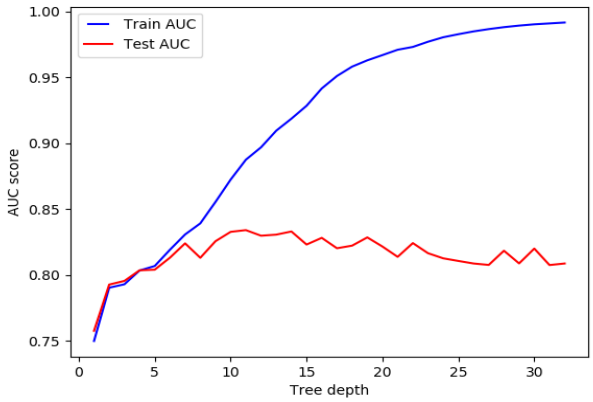

Figure 1: score al variare del max_depth

Dopodiché è trovata la profonditá ideale ad avere una precisione migliore a seconda della divisione effettuata del dataset in train-set e test-set.

La precisione viene calcolata mediante la funzione *score()* del DecisionTreeClassifier preso dalla libreria **sklearn**, la metrica AUC di **matplotlib** ed infine con il metodo del *10-fold cross validation* fornito sempre da **sklearn**.

Utilizzando questa serie di azioni la *precisione* della predizione con l'albero decisionale risulta essere di circa l'**83%**.

3 Grafo del Decision Tree

Nell'ultima parte del codice vi è una parte commentata, in quanto serve a disegnare e a memorizzare il grafo dell'albero decisionale ottenuto.

Il metodo utilizzato per disegnare è quello dell'articolo, citato nella descrizione, di *Russel*, che consiste di creare un *dot_data* dell'albero per trasformarlo poi in un grafo. In seguito vengono colorati i nodi in maniera alternata ed infine viene memorizzato il grafo.

Per poter disegnare un grafo utilizzando le librerie citate nella descrizione (**pydotplus**, **graphivz**, **collections**) non è sufficiente importarle. Infatti **graphivz** deve essere innanzitutto scaricato direttamente dal sito ufficiale e in seguito va creato un *path* tramite il pannello di controllo del computer per poter essere utilizzato correttamente.

Per creare un *path* basta seguire le seguenti indicazioni:
$PannellodiControllo \rightarrow SistemaeSicurezza \rightarrow Sistema \rightarrow ImpostazioniAvanzate \rightarrow Variabilidiambiente \rightarrow Path \rightarrow Nuovo$,
in cui va specificato il percorso alla cartella *Graphviz* che, se scaricata nella cartella predefinita, risulta come:

C:\Program Files (x86)\Graphviz2.38\bin

Nonostante sembra non essere utilizzato nel codice è importante includere anche *graphviz* e seguire le suddette indicazioni per non incappare nell'errore:

GraphViz\'s executables not found.

A causa delle dimensioni del grafo, per poterlo visualizzare è necessario cliccare qui (*non disponibile per dispositivi mobile*).
È inoltre possibile trovare nella repository il file DOT del grafo, consultabile tramite blocco note o in ambiente di sviluppo python.