Analisi di Decision Tree e Random Forest

Intelligenza Artificiale

Federico Magnolfi

Febbraio 2018

Abstract

Utilizzando implementazioni esistenti di Decision Tree e Random Forest, si analizzano i dati di Stack Overflow Developer Survey, 2017: si vuole calcolare l'accuratezza ottenuta tramite una 10-Fold Cross Validation nel predire se il salario di un partecipante è sopra o sotto la mediana.

Descrizione del problema

Il dataset in questione contiene i dati ottenuti dal sito Stack Overflow nel 2017 tramite un sondaggio posto ai propri utenti. Ad ogni utente sono state fatte diverse domande, riguardanti principalmente i loro interessi, abitudini, lavoro, istruzione, esperienze nell'ambito informatico. Una delle domande poste riguarda il salario percepito dall'utente.

Si vogliono testare due diversi algoritmi di apprendimento supervisionato per predire se il salario di un utente è sopra o sotto la mediana, conoscendo le varie risposte che sono state date durante il sondaggio. I due algoritmi in questione sono Decision Tree e Random Forest, per ognuno dei quali si misura l'accuratezza ottenuta tramite una 10-Fold Cross Validation.

Descrizione dell'esperimento

L'intero dataset è contenuto in un file csv; si osserva che ogni utente non risponde alla maggior parte delle domande, ovvero il dataset ha molti dati mancanti. Si è quindi reso necessario eliminare ogni riga in cui non è specificato il salario.

Inoltre, per cercare di ridurre il numero di dati mancanti e migliorare quindi l'accuratezza delle previsioni, si è scelto di eliminare quelle colonne per cui i dati mancanti superano una certa percentuale. Sebbene può accadere che una delle colonne eliminate abbia valori ai quali corrisponde sempre lo stesso output desiderato, si ritiene che la presenza di troppi valori nulli possa condizionare l'interpretazione dell'attributo da parte degli algoritmo.

Si prevede lo svolgimento di più test, per fare la media e ottenere dei risultati più precisi. All'inizio di ogni test viene mischiato casualmente il dataset, in modo da evitare che i risultati siano influenzati da un particolare ordinamento dei dati. Si esegue quindi la 10-Fold Cross Validation prima con un Decision Tree e poi con una Random Forest, misurando in entrambi i casi sia l'accuratezza media sul Train Set che quella sul Test Set.

Un'alta accuratezza sul Train Set significa che l'algoritmo è in grado di riuscire a classificare almeno gli esempi che sono stati usati visti durante il processo di apprendimento, ovvero si ha un basso underfitting. Un'accuratezza sul Test Set che si avvicina molto a quella ottenuta sul Train Set significa che l'algoritmo è in grado di classificare bene anche esempi che non ha visto durante il processo di apprendimento, ovvero di ha un basso overfitting.

Si riportano infine i risultati ottenuti e si comparano mediante un istogramma.

Aspettative

In un Decision Tree, si cerca di regolare parametri quali altezza massima e minimo numero di esempi per foglia, per trovare un buon compromesso per ridurre sia l'errore di bias che l'errore di varianza. Il primo è un errore sistematico che porta ad avere underfitting, si diminuisce aumentando l'altezza massima dell'albero e diminuendo il numero minimo di campioni per foglia. Il secondo è un errore dovuto al troppo adattamento ai dati del train set, per ridurlo si adottano le misure opposte rispetto alle precedenti.

Per Random Forest valgono gli stessi principi, con la differenza che in genere si riesce a trovare un compromesso migliore: questo perché la foresta è composta da un insieme di alberi e non da uno solo, quindi si riesce a diminuire l'overfitting, senza aumentare considerevolmente l'errore di bias.

Ci si aspetta quindi che i due algoritmi si comporteranno in modo simile, tra i due molto probabilmente Random Forest risulterà migliore proprio per la sua capacità di ottenere un migliore compromesso tra errore di bias e quello di varianza.

Implementazione

Si implementa quanto descritto mediante il linguaggio di programmazione Python, versione 3. Gli algoritmi in analisi fanno parte della libreria scikit-learn, le classi usate per i classificatori sono DecisionTreeClassifier e RandomForestClassifier.

Il programma è composto da tre file:

- **prepare.py**: contiene una funzione che richiama con ordine funzioni contenute in *utilities.py*, per leggere il file csv, filtrare il dataset dai dati non significativi, e creare le strutture dati che contengono le informazioni del dataset. Queste strutture dati vengono poi salvate su dei file tramite la libreria pickle.
- **exp.py**: questo è il file che realizza l'esperimento vero e proprio, all'inizio di questo file sono definiti tutti i parametri che regolano il funzionamento, come i dettagli del file csv, il numero di test da effettuare o i parametri richiesti dagli algoritmi di apprendimento.

  Dopodiché inizia l'esperimento: vengono caricate innanzitutto le strutture dati dai file, che sono creati se non sono presenti. Esegue 10 test come descritto precedentemente, i risultati vengono mostrati sulla console e tramite un istogramma, usando la libreria matplotlib.
- **utilities.py**: contiene tutte le varie funzioni che vengono usate dagli altri file, in modo da separare, per quanto possibile, la logica di funzionamento del programma da alcuni dettagli implementativi. Le funzioni principali servono per leggere il file csv, filtrare le colonne con troppi valori nulli, trasformare le stringhe del dataset in numeri attraverso i LabelEncoder.

I parametri che regolano il funzionamento del programma sono di due tipologie: alcuni relativi alla lettura del dataset, altri relativi all'esecuzione dei test. Si riportano di seguito i parametri principali con relativa spiegazione, il valore usato nei test viene riportato tra parentesi.

Parametri relativi alla lettura del dataset:

- fileDataset ("dataset/survey_results_public.csv"): percorso del file csv contenente il dataset;
- separator (","): carattere separatore nel file csv;
- nameColSalary ("Salary"): nome della colonna nella quale è contenuto lo stipendio;
- nonAvailableValue ("NA"): stringa che nel file csv rappresenta i dati mancanti;
- acceptedNA (0.05): numero compreso tra 0 e 1 che esprime il numero di valori mancanti ammessi per colonna.

Parametri relativi all'apprendimento:

- K (10): parametro K relativo alla K-Fold cross validation;
- numTests (10): numero di test da eseguire;
- maxDepth (6): massima profondità di ogni albero, sia per Decision Tree che per Random Forest;
- minSamplesLeaf (55): numero di campioni per ogni foglia, sia per Decision Tree che per Random Forest;
- nEstimators (20): numero di alberi nella foresta.

I valori usati nell'esperimento sono stati individuati in modo non rigoroso, quindi molto probabilmente non saranno ottimi, ma sono tali da avere buoni valori di accuratezza ed un basso overfitting, oltre che un buon tempo di esecuzione dei test.

Utilizzo del programma

Per eseguire l'esperimento, i passi sono i seguenti:

1. Procurarsi il dataset, quello usato nell'esperimento è reperibile alla pagina https://www.kaggle.com/stackoverflow/so-survey-2017. In realtà è possibile utilizzare un qualsiasi dataset che sia in formato csv e preveda una colonna per lo stipendio. È necessario inserire, all'inizio del file *exp.py* il corretto percorso al file csv, assegnando l'opportuno parametro.
2. *Opzionale*: modificare a proprio piacimento i parametri di interesse, che sono tutti definiti verso l'inizio del file *exp.py*.
3. Eseguire il file *exp.py*: alla prima esecuzione vengono generati i file intermedi, dalla seconda in poi, e se esistono già, vengono soltanto letti. Se si vuole eseguire un nuovo esperimento, è possibile cambiare i parametri relativi all'apprendimento senza che il file csv venga letto per intero nuovamente. Se si volesse anche cambiare i parametri relativi alla lettura del dataset si può fare, ma è necessario cancellare almeno uno dei file intermedi.

Nota: i tempi di esecuzione variano sicuramente in base alla piattaforma di esecuzione, ma indicativamente sono di una decina di secondi per la preparazione dei file, e di qualche secondo per ogni test. L'avanzamento del programma viene mostrato sulla console.

Risultati sperimentali

Si riportano di seguito i risultati ottenuti nell'esperimento, le accuratezze sono arrotondate a tre cifre significative:

Tabella 1: Accuratezze medie ottenute, in percentuale

	Decision Tree	Random Forest
Train Set	80.7%	80.7%
Test Set	80.1%	80.4%

Si è quindi registrato un overfitting dello 0.6% per Decision Tree e dello 0.3% per Random Forest.

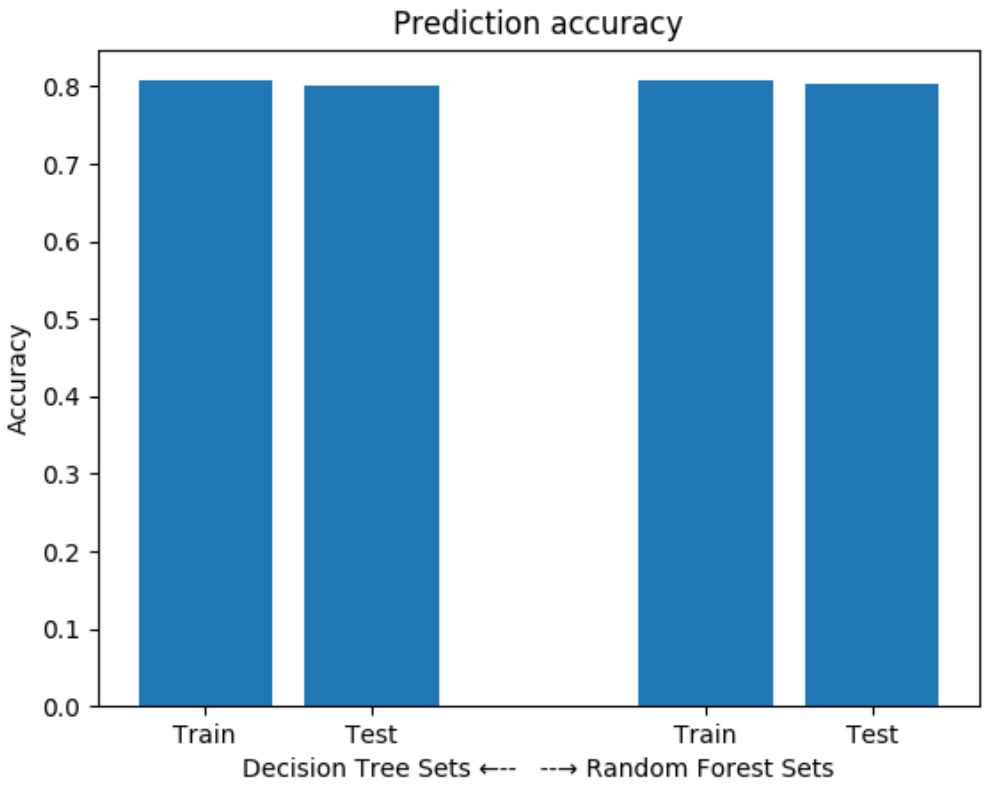

Figura 1: Grafico delle accuratezze, con numeri compresi tra 0 e 1

Conclusione

I due algoritmi si sono rilevati praticamente identici nelle accuratezze ottenute. Si nota comunque che Random Forest ha effettivamente avuto un overfitting leggermente minore rispetto a Decision Tree, ovvero una migliore accuratezza nel generalizzare, seppur di poco.