Relazione per Elaborato di Intelligenza Artificiale

Simone Casini

Algoritmo Alberi di Decisione

L'algoritmo si trova all'interno di decision_tree_alg.py e si basa sulla funzione dt_learn() .
La struttura del dataset e dell'albero sono riprese dall' aima python repository su github e sono rappresentate rispettivamente nelle classi Dataset, Tree e Leaf.
L' algoritmo nel passo base valuta inizialmente se tutti gli esempi hanno la stessa classificazione e in tal caso restituisce il valore comune a tutti.
Successivamente valuta se il numero di esempi è pari a zero, in tal caso restituisce la foglia corrispondente al valore del target più presente negli esempi del nodo padre, calcolato con majority_value().
Se invece non sono rimasti più attributi allora restituisce la foglia con il valore di target più presente negli esempi, utilizzando sempre majority_value().
Se non si verifica nessuna delle condizioni precedenti, allora è necessario selezionare un attributo su cui fare lo split dell'albero.
L'attributo migliore è selezionato attraverso il calcolo dell'information gain, calcolato utilizzando l'entropia come misura di impurità. L'attributo migliore è quello che ci permette di ottenere il maggior information gain, ossia di ridurre maggiormente l'entropia, cioè l'impurità.
Una volta selezionato l'attributo migliore, indicato con A, si crea un albero avente come root test A.
Per ogni valore di A si crea un subtree formato da esempi con lo stesso valore di A; la creazione del subtree corrisponde ad una chiamata ricorsiva dell'algoritmo.
Al termine della creazione del subtree questo viene aggiunto come nuovo ramo all'albero creato al passo prima.
Alla fine si restituisce l'albero creato.

Rimozione Casuale

Per rimuovere casualmente i valori dal dataset si utilizza la funzione removeRandomValues() in dataset.py, che prende come argomento la probabilità con cui si vuole rimuovere.
Per ogni esempio si considera ogni attributo (escluso il target) e se la funzione probability() restituisce True, allora si rimuove il valore di quell'attributo dall'esempio.
I valori sono rimossi ogni volta dal training all'interno del 10-fold-cross-validation.

Gestione valori mancanti

Come descritto nella sezione 3.7.4 in Mitchell(1997), se in un esempio utilizzato per il training mancano alcuni valori per degli attributi ad un nodo n, si può assegnare un valore di default.
Questo valore di default corrisponde al valore più comune presente negli esempi al nodo n per quell'attributo.
Nell'algoritmo questo è implementato all'interno della funzione select_best_attribute() usando manage_missing_values(): per ogni attributo si gestiscono i valori mancanti scegliendo il valore più comune con most_common_value() e dopodiché si esegue il calcolo dell'information gain rispetto a quell'attributo.

10-fold-cross-validation

È utilizzata per calcolare l'accuratezza, consiste nel dividere il dataset in 10 parti uguali e utilizzare nove parti come training set e la restante come validation.
Questo è ripetuto per 10 volte, in modo che ogni gruppo o fold sia utilizzato come validation almeno una volta, dopodiché si fa la media dei risultati.
Nel test questo viene ripetuto 10 volte e viene fatta la media.

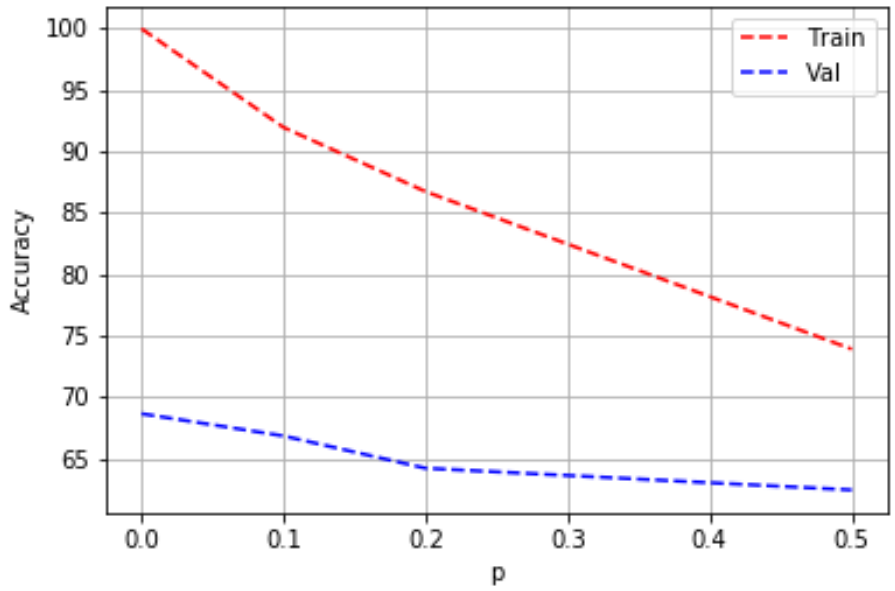
Riproduzione Risultati

Per riprodurre i risultati si devono utilizzare tutti i file .py e scaricare i file txt che corrispondono ai dataset.
Nei vari main_nome_dataset.py sono presenti i nomi degli attributi dei dataset e i loro valori che attraverso la funzione test_plot() sono usati per creare i dataset.
Per ognuno dei dataset è presente un main_nome_dataset.py su cui facendo il run si può riprodurre il risultato.

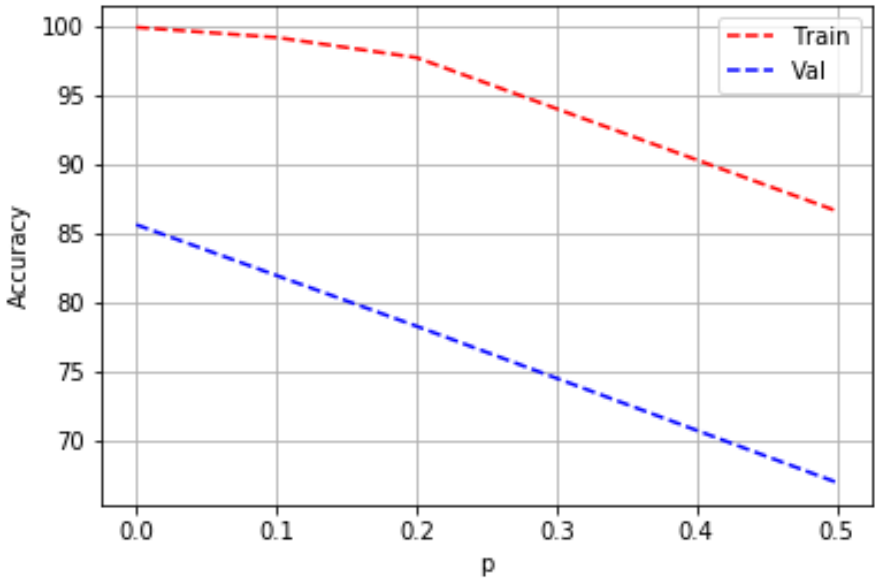
Dataset

I dataset sono presi dall' UCI Machine Learning Repository.
Balance-scale è un dataset contenente 625 esempi. Tic-tac-toe endgame è un dataset contenente 958 esempi. Car-evaluation è un dataset contenente 1728 esempi.

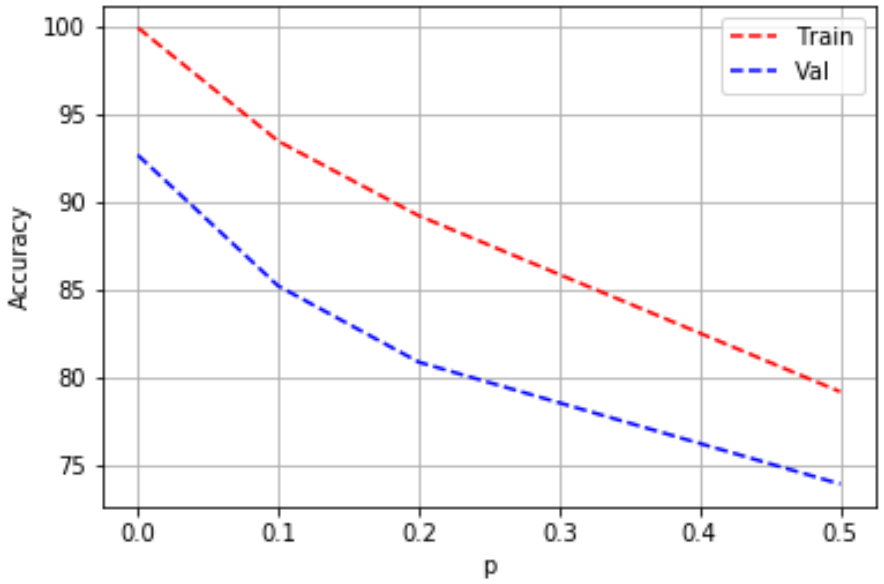
Balance-scale



Tic-Tac-Toe Endgame



Car-Evaluation



Conclusioni

Si può notare come nel dataset di dimensione maggiore la differenza di accuratezza tra training e validation resti constante all'aumentare dei valori mancanti; invece nel caso di dataset di dimensione inferiori, come per Balance-Scale, questa differenza va a diminuire.