Corso di Algoritmi e Strutture Dati 2018/2019

Laboratorio 10:

Realizzazione di un semplice software per gestire Alberi di decisione (Decision Trees)

Un albero di decisione è un **modello predittivo** utile per differenti scopi e spesso viene utilizzato come strumento per il supporto alle decisioni. Ad esempio un albero di decisione potrebbe essere utilizzato per stimare il fatturato di un punto vendita sulla base delle sue caratteristiche (dimensioni del punto vendita, città, numero di impiegati, ...) oppure per classificare alcuni movimenti bancari in ordinari o fraudolenti.

In un albero di decisione ogni nodo interno rappresenta una variabile, un arco verso un nodo figlio rappresenta un possibile valore (o un insieme di valori) per quella variabile e una foglia il valore predetto dall'albero di decisione a partire dai valori delle altre variabili, che nell'albero è rappresentato dal cammino (*path*) dal nodo radice (*root*) al nodo foglia.

Per semplicità, assumiamo che i valori che può assumere una **variabile** possono essere numerici oppure di semplice appartenenza ad una certa categoria. In particolare, possiamo dividere la tipologia delle variabili che possono essere utilizzate in un albero di decisione in due:

- Variabili quantitative. Ad esempio: peso, altezza, età, costo di un prodotto
- Variabili qualitative con valori non ordinabili (scala nominale). Sono anche chiamate variabili categoriche. Ad esempio il Gruppo sanguigno (che può assumere i seguenti valori: O, A, B, AB) o il tipo di malattia

Sulle variabili quantitative sono presenti le relazioni di uguaglianza (uguale e diverso) e le relazioni d'ordine (ovvero: minore, maggiore, minore e uguale, maggiore e uguale) mentre per le variabili qualitative con valori non ordinabili (scala nominale) è presente solo la relazione di uguaglianza.

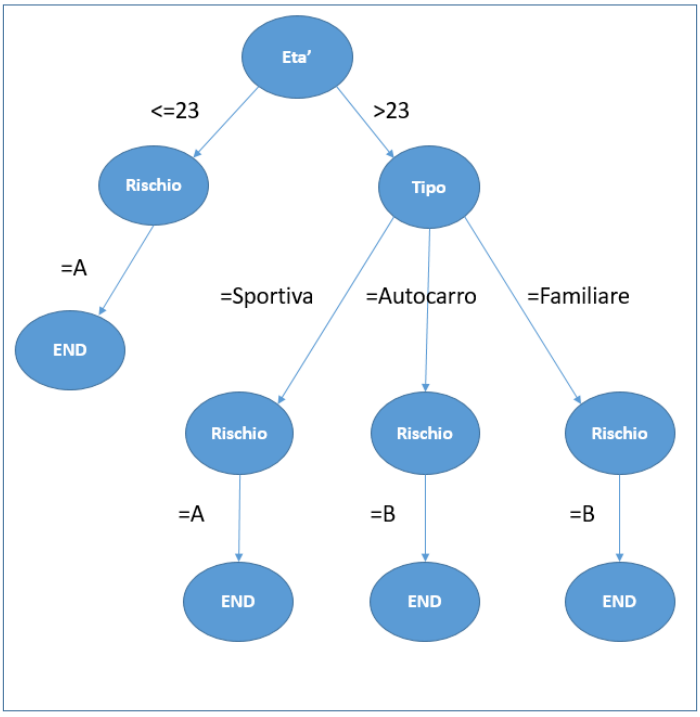
Normalmente un albero di decisione viene costruito utilizzando **tecniche di apprendimento** a partire dall'insieme dei dati iniziali, il quale solitamente viene suddiviso in due sottoinsiemi: il *training set* sulla base del quale si crea la struttura dell'albero e il *test set* che viene utilizzato per testare l'accuratezza del modello predittivo così creato. Tuttavia l'algoritmo di costruzione dell'albero di decisione esula dagli scopi del laboratorio 10.

Esempio di Albero di decisione

Supponiamo che i dati del training set provengano da una compagnia di assicurazioni, nella quale un esperto ha assegnato ad ogni utente un livello di rischio (A=alto, B=basso) basandosi su incidenti effettivamente avvenuti. Supponiamo che il training set sia il seguente (S sta per sportivo, FA sta per familiare, AU sta per autocarro):

	ETÁ	TIPO DI AUTOVEICOLO	RISCHIO
Utente 1	17	S	A
Utente 2	43	FA	B
Utente 3	68	FA	B
Utente 4	32	AU	B
Utente 5	23	FA	A
Utente 6	18	FA	A
Utente 7	20	FA	A
Utente 8	45	S	A
Utente 9	50	AU	B
Utente 10	64	AU	A
Utente 11	46	FA	B
Utente 12	40	FA	B

La compagnia di assicurazioni vorrebbe una procedura automatica che sia in grado di segnalare se un nuovo cliente puo' essere rischioso oppure no. L'esempio rappresenta proprio un tipico problema di classificazione e/o predizione che può essere risolto con la costruzione di un albero di decisione associato al training set di cui sopra. Un esempio di albero di decisione ricavato dalla tabella potrebbe essere il seguente:



A questo punto se si presentasse all'assicurazione un cliente di 49 anni con una macchina familiare, il manager dell'assicurazione potrebbe inferire che il cliente presenta un rischio basso di incidente e quindi potrebbe proporre una polizza a basso costo.

Implementazione

Lo scopo di questo laboratorio è quello di realizzare un programma C++ che sia in grado di:

(1) leggere da file un albero di decisione così come descritto sopra e modificarlo con le operazioni cancella nodo, aggiungi nodo, modifica nodo;

(2) visualizzare in modo testuale l'albero di decisione. La stampa dovrà contenere '**abbastanza informazione**' da poter permettere ad un utente del vostro software di ricostruire l'albero di decisione (ad esempio usando penna e carta);

(3) inferire e visualizzare le variabili dell'albero di decisione (nell'esempio in figura il programma dovrebbe visualizzare Età, Rischio e Tipo)
(4) effettuare una predizione a partire da un albero di decisione precedentemente inserito. In particolare il programma dovrebbe chiedere all'utente, uno alla volta, i valori da associare alla variabili durante il path che porta alla predizione. Ad esempio, dopo che l'utente ha inserito come Età il valore 45 il programma dovrebbe chiedere all'utente di inserire il Tipo di automobile. Se l'utente inserisce Sportiva, il programma dovrebbe stampare come predizione 'A' che corrsiponde ad alto rischio. Ricordiamo invece che non fa parte del laboratorio 10 la costruzione dell'albero di decisione a partire dai dati di training (essendo un algoritmo abbastanza complesso che vedrete negli anni a seguire). Potrebbero esistere casi in cui si ha più di una condizione vera sugli archi (ad esempio se abbiamo due condizioni a=5 e a>4 sugli due archi e il valore di 'a' è uguale a '5') oppure casi in cui nessuna condizione è vera (ad esempio quando abbiamo le condizioni a=5 e a>5 sugli unici due archi e il valore di 'a' è '4'). Nel primo caso il programma dovrà scegliere in modo casuale uno degli archi che hanno la condizione vera, nel secondo caso il programma dovrà stampare il testo: "la predizione non può avere luogo in quanto esiste un nodo per il quale non c'e' un arco percorribile".
(5) effettuare una predizione a partire da un albero di decisione precedentemente inserito e da un insieme di valori delle variabili. Il programma in questo caso dovrebbe chiedere un insieme di coppie (variabile, valore) all'utente e successivamente dopo avere acquisito tutti i valori dovrebbe provare ad istanziare le condizioni sugli archi al fine di effettuare una predizione. Anche qui come prima potrebbe esistere il caso in cui vi siano più condizioni vere sugli archi e casi in cui la predizione non può avere luogo (si pensi al caso in cui le condizioni sono a<=5 e a>5 e l'insieme di coppie è {(b, 8), (c,4)}

Formato ed esempio di file di input

Il formato del file di input è molto simile a quello usato a lezione per gli alberi generici. Ovvero la prima riga del file deve contenere l'etichetta della radice e le righe seguenti devono contenere come prima etichetta quella di un nodo (che deve già essere stato elencato prima) seguita dalle coppie etichetta di uno dei suoi figli e la corrispondente l'etichetta che rappresenta la condizione dell'arco. Per non complicare troppo le cose le etichette dei nodi e le condizioni sono stringhe che non contengono il carattere di spazio. Le condizioni sono stringhe che iniziano con un'operatore relazionale tra i seguenti {=, ≠, <, >, <=, >=} seguito da un valore di una variabile (nel caso dell'esempio i valori della variabile Età sono dei numeri interi, mentre i valori della variabile Tipo sono le stringhe *Sportiva*, *Autocarro* e *Familiare*.

```
radice
radice nodo1 cond1 nodo2 cond2 nodo3 cond3
nodo1 nodo4 cond4 nodo5 cond5 nodo6 cond6 .......
nodo2 nodo7 cond7 nodo8 cond8 nodo9 cond9 .......
```

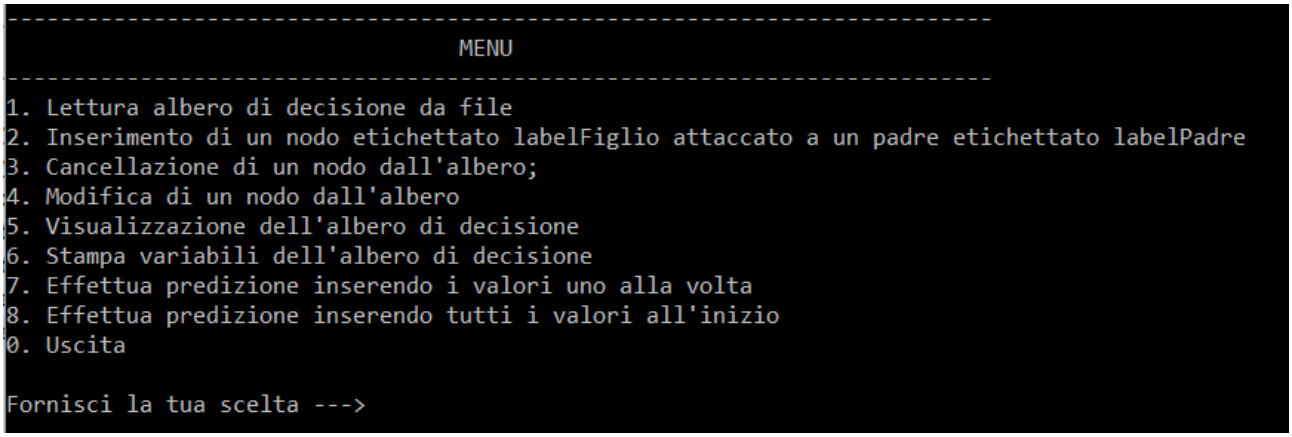
ESEMPIO (si riferisce all'esempio riportato nella figura sopra):

```
Età_1
Età_1 Rischio_1 <=23 Tipo_1 >23
Rischio_1 END_1 =A
Tipo_1 Rischio_2 =Sportiva Rischio_3 =Autocarro Rischio_4 =Familiare
Rischio_2 END_2 =A
Rischio_3 END_3 =B
Rischio_4 END_4 =B
```

Underscore seguito da un numero serve per disambiguare le label

Menu

Il programma dovrà fornire all'utente un menu simile al seguente:



Vincoli

Non è consentito utilizzare librerie specifiche per gli alberi o per i grafi (ad esempio, The Boost Graph Library - BGL) così come non è consentito utilizzare i Vector. Si consiglia di riusare il più possibile il codice sviluppato e testato durante l'anno.

Consegna

Si richiede di consegnare i file .cpp e .h (e gli eventuali file di input usati per testare il vostro software) in un unico file zip con il formato CognomePrimaLetteraNome.zip (ad esempio RiccaF.zip). **Il progetto deve essere svolto e consegnato in modo individuale**.

Punti e Valutazione

Il punteggio massimo di **2.5 punti** si raggiunge non solo se le funzioni e le strutture dati sono corrette, ma anche se sono implementate in modo efficiente e non ci sono problemi "di stile". In particolare, sono elementi apprezzati ai fini della valutazione (l'elenco non è esaustivo): corretta indentazione del codice, identificatori significativi, introduzione di funzioni ausiliarie quando appropriato e commenti significativi.