## Regression Tree Miner

*Metodi avanzati di Programmazione - Caso di studio a.a. 2019-2020*

Progetto realizzato da: **Gianfranco Demarco** (mat. 708795) - gianfranco.demarco26@studenti.uniba.it

**Indice**

1. Introduzione
2. Guida di installazione
3. Casi di test
   1. Scenari alternativi
4. Javadoc

**1. Introduzione**

Il progetto consiste nella creazione di un'architettura software composta da due parti:

* un applicativo server: contiene la logica di business; interagisce con la base di dati; comunica con il client;
* un applicativo client: costituisce l'interfaccia utente, veicola le richieste dell'utente verso il server

In breve, un albero di regressione è una struttura che viene costruita "intelligentemente" (utilizzando dei parametri matematici) a partire da un dataset (un insieme di dati).

Il dataset è composto da diverse colonne (***attributi***) contenenti valori (*discreti* o *continui*) e da una colonna ***target***, rappresentante il valore che vogliamo predire.

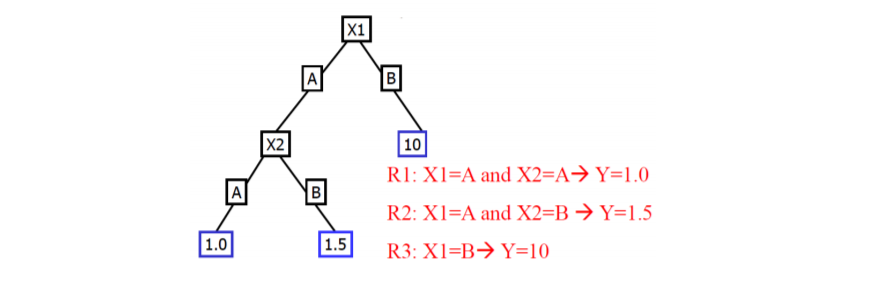
Un attributo si dice discreto se può assumere solo un insieme finito di valori; si dice continuo se può assumere un numero infinito di valori all'interno di un dato range (es. [-∞, +∞], [0, 40]...).

L'algoritmo quindi analizza i dati forniti come input e crea una struttura (***albero***).

Ad ogni iterazione sui dati, un algoritmo stabilisce se generare un nodo di *split* o un nodo *foglia*:

* nodo di ***split***: un nodo di split rappresenta una decisione sul valore di un attributo; il dataset viene diviso con i nodi di split in subset sempre più piccoli. Per decidere che tipo di split applicare si punta a trovare la divisione in subset che vada a minimizzare un certo parametro (ad esempio la varianza dei dati).

Si continua a generare nodi di split finché possibile o comunque finché il numero di esempi coperti da uno split è sufficentemente ampio (si può scegliere una soglia arbitraria)



* nodo ***foglia***: quando non è più possibile generare nodi di split o quando il numero di esempi coperti dallo split è sotto una certa soglia, si genera un nodo foglia. Un nodo foglia rappresenta il valore dell'attributo target per quelle istanze che lo raggiungono attraverso gli split.

Ad esempio un'istanza così composta: **{X1:A, X2: A}** avrà come valore target *1.0*.

Quando la costruzione dell'albero è completa, esso può essere utilizzato per predire nuovi valori target per istanze mai analizzate precedentemente.

Si pensi ad un albero di regressione riguardo gli appartamenti nella città di Milano.

Il dataset potrebbe essere composto dagli attributi **quartiere**(*discreto*), **vani**(*continuo*), **servizi**(*continuo*) e dall'attributo target **prezzo**.

Una volta generato l'albero di regressione con un dataset di dati a noi conosciuti, si può ad esempio stabilire quale sarebbe il giusto prezzo per una nuova costruzione, oppure valutare se il prezzo di un dato immobile è sovra/sottostimato.

**Flusso Applicativo**

L'applicativo server è costruito secondo un'architettura standard: una ServerSocket rimane in ascolto di connessioni da parte dei client (ad esempio sulla porta 8080); ricevuta una richiesta, viene stabilita una connessione e la comunicazione viene gestita su un thread dedicato, così che l'applicativo possa quindi restare in ascolto e in definitiva servire più client contemporaneamente.

Quando avviene una connessione, parte un canale di comunicazione tramite cui l'utente decide quali computazioni debba effettuare il server.

Di seguito uno schema riassuntivo del flusso applicativo:

***Attori***: MS - MultiServer; S - Server; C - Client; U - utente

1. MS - viene avviato l'applicativo; resta in attesa di connessioni
2. C - richiede una connessione al MS
3. MS - accetta la connessione del C; crea un'istanza di S che gestisca la comunicazione con il C.
4. **parallelamente** MS torna al punto 1; S avvia la comunicazione con C
5. ***Computazione***
6. C mostra una scelta a U;
7. U effettua la scelta;
8. C comunica la scelta a S
9. S risponde;
10. Torna a 5

**Requisiti funzionali - Server**

Il server deve esporre le seguenti funzionalità:

1. Permettere la creazione di un albero di regressione a partire da un dataset memorizzato nella base di dati;
2. Permettere il salvataggio del dump di un albero di regressione creato come descritto al punto (1) attraverso il meccanismo di ***serializzazione*** di Java;
3. Permettere la ricostruzione di un albero di regressione salvato come descritto al punto (2) attraverso il meccanismo di ***deserializzazione*** di Java;
4. Permettere la stampa delle regole generate da un albero di regressione;
5. Permettere la stampa della struttura di un albero di regressione;
6. Permettere la predizione di nuovi valori scorrendo la struttura dell'albero fino ad un nuovo foglia;
7. Permettere la comunicazione con un applicativo client, attraverso il meccanismo delle socket. Tutti gli input riguardo le scelte che comandano i flussi di esecuzione devono essere letti dal client; tutti gli output devono essere redirezionati verso il client

**Requisiti funzionali - Client**

Il client deve esporre le seguenti funzionalità:

1. Guidare l'utente ed esporre le scelte possibili;
2. Leggere gli input dell'utente e direzionarli verso l'applicativo server;
3. Leggere gli output del server e direzionarli verso l'utente.

### 2. Guida di installazione

Pre-requisiti:

- MySQL; JRE 8+ installati sulla macchina.  
- esistenza di un'utenza "root" per il server MySQL. Se la password è diversa da "root", modificare il file "setup.bat" alla riga 2 sostituendo la password corretta.

1. Eseguire lo script *setup.bat*.

Esso si occupa di creare l'utenza per l'applicativo sul database; creare e popolare le tabelle per 2 dataset di test; eseguire l'applicativo server e l'applicativo client.

Per le esecuzioni successive, eseguire gli script *runServer.bat* e *runClient.bat* in questo ordine.

### 3. Casi di test

**UC1**: Creazione della connessione client-server

***Pre-condizioni*:**

​ 1. E' stato eseguito almeno una volta con successo lo script *setup.bat*

***Flusso di esecuzione:***

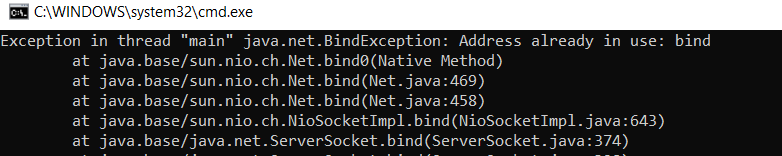
1. Eseguire il file runServer.bat
   1. Eseguire il file runClient.bat

***Post-condizioni:***

1. Entrambi gli applicativi sono stati avviati correttamente.
   1. Sull'applicativo server si legge "Connection Accepted!", indice che la connessione è stata stabilita con successo.

***Scenari alternativi***:

1. Al punto (1) potrebbe verificarsi il seguente errore:



Esso indica che la porta su cui dovrebbe mettersi in ascolto il server è già utilizzata da qualche altro processo; individuare quel processo e terminarlo.

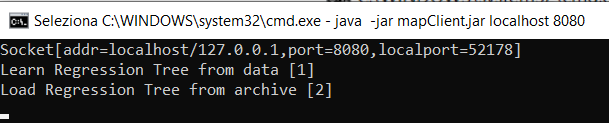
**UC2:** Creazione di un albero di regressione a partire da un dataset

***Pre-condizioni*:**

​ 1. Portare a termine lo **UC1**

***Flusso di esecuzione:***

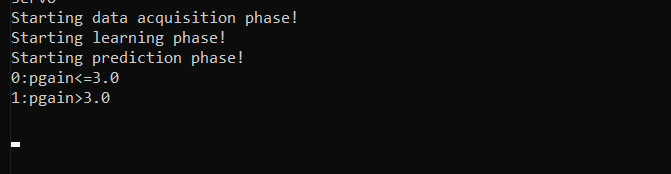
1. Sul client viene riportata la seguente scelta

* Digitare "1" e premere invio.

1. Il client riporta l'output "filename"

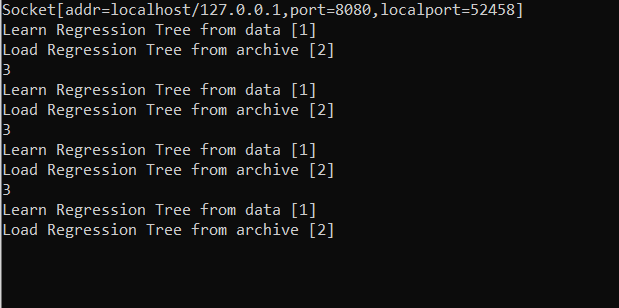
Digitare "servo" e premere invio.

***Post-condizioni:***

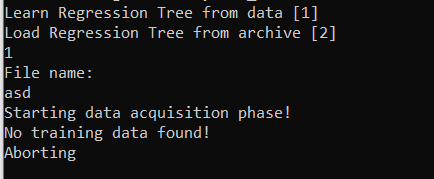
1. Il client riporta il seguente output:
2. E' stato creato il file ./servo.dmp

***Scenari alternativi***:

1. Se al punto (1) si digita una scelta non indicata dal client, essa viene ignorata e viene riproposto lo stesso output finché non si effettua una scelta corretta.



* Se al punto 2 si digita il nome di un dataset non presente, si ha l'output seguente e il processo client viene terminato.



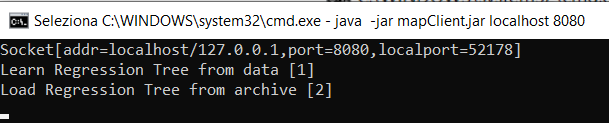
**UC3:** Creazione di un albero di regressione a partire da un dump

***Pre-condizioni*:**

​ 1. Portare a termine lo **UC2**

***Flusso di esecuzione:***

1. Sul client viene riportata la seguente scelta

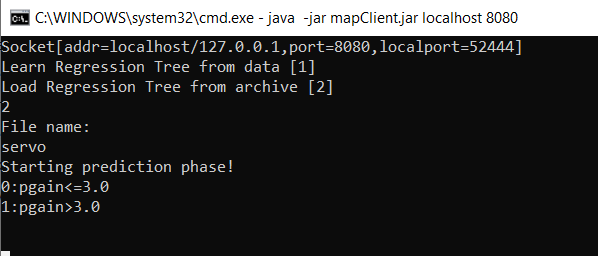
* Digitare "2" e premere invio.

1. Il client riporta l'output "filename"

Digitare "servo" e premere invio.

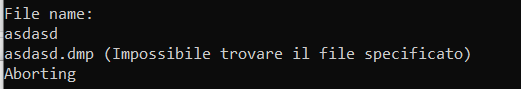
***Post-condizioni:***

1. Il client riporta il seguente output:



***Scenari alternativi***:

1. Analogo a UC1 - Scenari alternativi (1)  
2. Se al punto 2 si digita il nome di un file non presente, si ha l'output seguente e il processo client viene terminato.



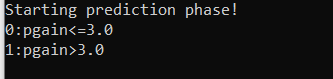
**UC4:** Predizione di un nuovo valore target

***Pre-condizioni*:**

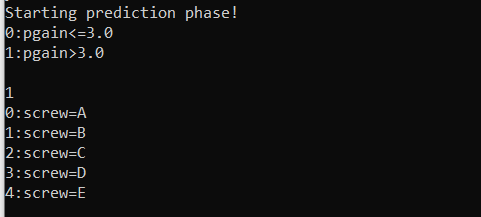
​ 1. Portare a termine lo **UC2** o lo **UC3**

***Flusso di esecuzione:***

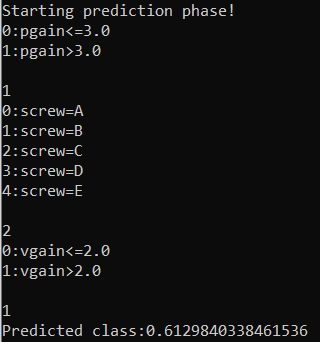
1. Sul client viene riportata la seguente scelta



1. Digitare una scelta prevista tra quelle indicate dal client e permere invio



1. Ripetere 2 finché l'output non riporta "Predicted class:..."

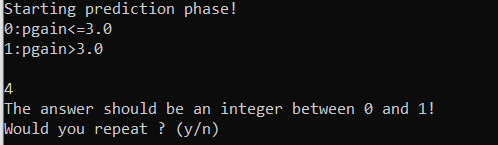


***Post-condizioni:***

1. Il client stampa in output il valore predetto.

***Scenari alternativi***:

1. Se al punto (1) si digita una scelta non indicata dal client, la computazione viene terminata e viene chiesto all'utente se ricominciare con la predizione



1. Una volta terminata la predizione, viene chiesto all'utente se ricominciare. In caso affermativo ('y'), viene riproposta la prima scelta; in caso negativo, il processo client termina la sua esecuzione.

### 4. Javadoc

Al codice sorgente è allegata la documentazione Javadoc. Per l'applicativo client è minimale, in quanto è presente una sola classe che controlla il flusso di esecuzione.

Per l'applicativo server sono documentati i package Server, Data, Tree, Database.