**Regression Tree Miner**

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

Progetto realizzato da: Gianluca Colasuonno – gcolasuonno14@studenti.uniba.it

**Indice**

1. Introduzione
2. Guida di installazione
3. Casi di test
4. Javadoc
5. **Introduzione**

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

* Un applicativo *server*: contiene la logica per la costruzione dell’albero di regressione, interagendo con la base di dati e comunica con il client;
* Un applicativo *client*: costituisce l’interfaccia utente, veicola le richieste dell’utente verso il server

Un albero di regressione è una struttura che viene costruita “intelligentemente” a partire da un dataset.

Il dataset è composto da dati che vengono contenuti in una tabella le cui colonne rappresentano gli 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 un insieme finito di valori, si dice invece continuo se può assumere un numero infinito di valori all’interno di un range.

L’algoritmo analizza i dati forniti come input e crea una struttura informatica detta albero. Ad ogni iterato sui dati, l’algoritmo stabilisce se generare **un nodo di split** o un **nodo foglia**.

*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 poi decidere che tipo di split applicare si punta a trovare la divisione in subset che vada a minimizzare un certo parametro, in questo caso la varianza dei dati. Si continua a generare nodi di split finché il numero di esempi coperti da uno split è sufficientemente ampio.

*Un nodo foglia* viene creato quando non è più possibile generare nodi di split o quando il numero di esempi coperti dallo split è sotto una certa soglia. Un nodo foglia rappresenta il valore dell’attributo target per quelle istanze che lo raggiungono attraverso lo split.

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

Un esempio di applicazione del programma potrebbe essere il caso in cui venga utilizzato da una banca per predire la somma di denaro da prestare a un nuovo cliente.

Si pensi a un dataset già presente nel sistema bancario formato dagli attributi: età (continuo), professione (discreto), reddito (continuo), somma prestata (continuo) e in questo caso rappresenta il nostro target.

Nel momento in cui la banca deve concedere un prestito a un nuovo cliente, il programma elabora i vari dati del dataset costruendo l’albero che ci permette di predire la somma di denaro che si potrà prestare.

**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:

**Legenda**: 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**: ultima versione di MySQL e JRE 14+ 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.

Eseguire lo script ./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 ./script/runServer.bat e ./script/runClient.bat in questo ordine.

**3. Casi di test**

**Legenda**: CT rappresenta il caso di test che viene eseguito, seguito da un numero progressivo. Ogni caso di test è costituito da delle pre-condizioni, che indicano i passi che bisogna eseguire per arrivare alla parte del programma che viene testata, il flusso di esecuzione che rappresenta la normale esecuzione del codice, le post-condizioni che costituiscono la parte di programma che in normali condizioni di funzionamento si raggiungerebbe, infine vi sono gli scenari alternativi che costituiscono alcune probabili situazioni di errore che si possono verificare e il relativo comportamento del programma.

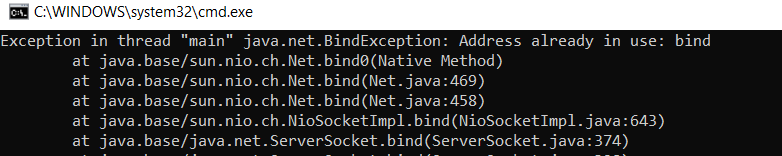
**CT1: Creazione della connessione client-server**

Pre-condizioni: E' stato eseguito almeno una volta con successo lo script setup.bat

Flusso di esecuzione: Eseguire il file runServer.bat, successivamente eseguire il file runClient.bat

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

Scenari alternativi: si potrebbe verificarsi il seguente errore:

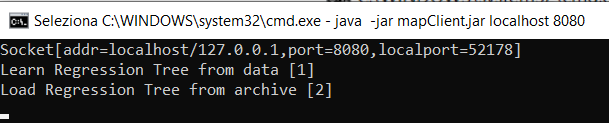


Esso indica che la porta su cui dovrebbe mettersi in ascolto il server è già utilizzata da qualche altro processo; individuare quel processo e terminarlo (si può eseguire come amministratore l’utility script/free8080.bat).

**CT2: Creazione di un albero di regressione a partire da un dataset**

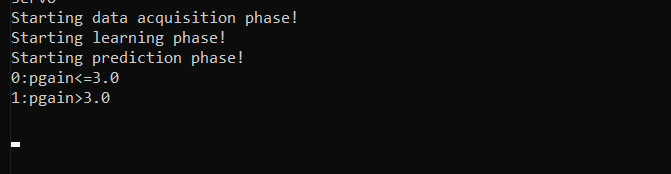
Pre-condizioni: Portare a termine CT1

Flusso di esecuzione: sul client viene riportata la seguente scelta

* 

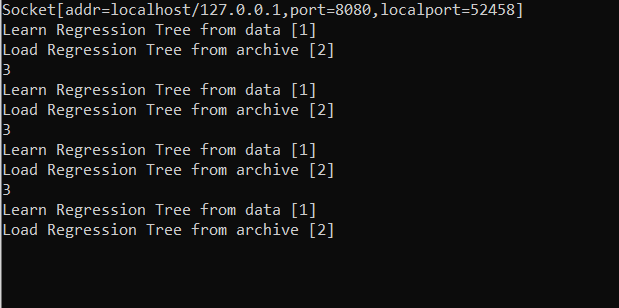
Digitare "1" e premere invio. Il client riporta l'output "tablename". Digitare "servo" e premere invio.

Post-condizioni: Il client riporta il seguente output:

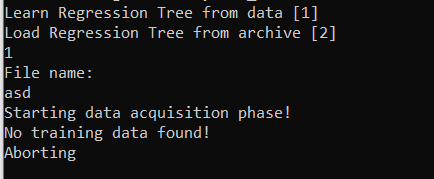


E' stato creato il file ./servo.dmp

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



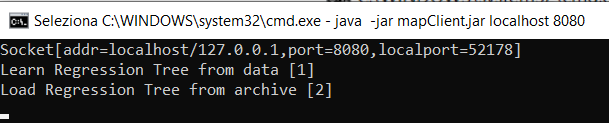
* Se al punto si digita il nome di un dataset non presente, si ha l'output seguente e il processo client viene terminato.



**CT3: Creazione di un albero di regressione a partire da un dmp**

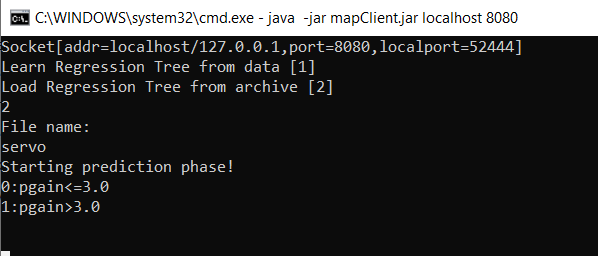
Pre-condizioni: Portare a termine lo CT2 selezionando il dataset “servo”

Flusso di esecuzione: Sul client viene riportata la seguente scelta

* 

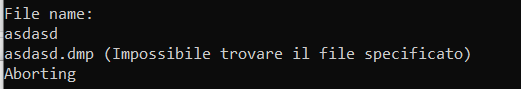
Digitare "2" e premere invio. Il client riporta l'output "filename" Digitare "servo" e premere invio.

Post-condizioni: Il client riporta il seguente output



Scenari alternativi:

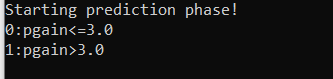
Un tipico errore, già affrontato nel CT1, è quello in cui l’utente non inserisce la scelta giusta. Se al punto 2 si digita il nome di un file non presente, si ha l'output seguente e il processo client viene terminato.



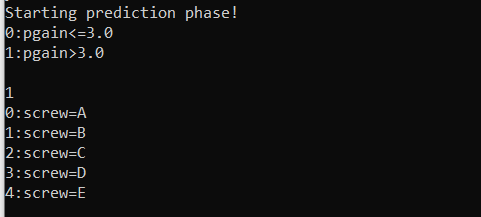
**CT4: Predizione di un nuovo valore target**

Pre-condizioni: Portare a termine lo CT2 o lo CT3

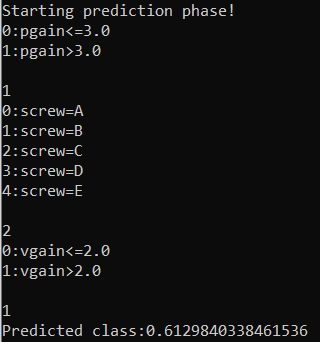
Flusso di esecuzione: Sul client viene riportata la seguente scelta



Digitare una scelta prevista tra quelle indicate dal client e premere invio

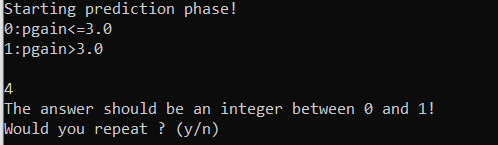


Ripetere 2 finché l'output non riporta "Predicted class:..."



Post-condizioni: Il client stampa in output il valore predetto.

Scenari alternativi: Se al punto si digita una scelta non indicata dal client, la computazione viene terminata e viene chiesto all'utente se ricominciare con la predizione



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 non è presente, 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.