**Università degli Studi di Bari**

**Aldo Moro**



Sistema di Data Mining: KNN  
Documentazione

Corso in:

Metodi Avanzati di Programmazione

Barbaro Vanessa – matricola: 725617

Capobianco Fabio – matricola: 609266

A.A 2021-2022  
21 luglio 2022

**Indice**

**Introduzione** 3

**Requisiti di sistema** 4

**Funzionalità implementate** 5

**Estensioni, differenze dal progetto originale** 6

**KNN: K-Nearest Neighbors (Regressione)** 7

**Diagramma delle Classi** 9

**Descrizione delle classi del progetto KNN** 10

1. **Introduzione**

Il progetto KNN è un software sviluppato in linguaggio Java su architettura client-server che implementa un algoritmo di data mining relativo ad un processo di regressione chiamato k-nearest neighbors (KNN), che viene utilizzato per il riconoscimento di pattern per la classificazione di oggetti basandosi sulle caratteristiche degli oggetti vicini a quello considerato.

Il Server acquisisce i dati memorizzati in file o database e li elabora per realizzare una predizione in base alle informazioni fornite dall’utente tramite il Client.

Il data mining permette l’estrazione (semi) automatica di conoscenza nascosta in voluminose basi di dati al fine di renderla disponibile e direttamente utilizzabile.

In particolare, l’algoritmo di regressione consente di stimare una o più variabili (continue o discrete) in base ad altri attributi del set di dati.

1. **Requisiti di sistema**

Per l’esecuzione di KNN è necessaria Java SE RE (Java Runtime Environment), versione 18 o superiore. Nello specifico per il funzionamento completo del Server è richiesto il Server MySQL.

In allegato è fornito lo script SQL per la creazione del DataBase con cui il Server lavora.

1. **Funzionalità implementate**

KNN offre le seguenti funzionalità:

* Applicazione dell’algoritmo di K-Nearest Neighbors
* Regressione su set di valori discreti e continui
* Interazione con una base di dati relazionale con linguaggio SQL: acquisizione dei set dalle tabelle.
* Possibilità di acquisire i Training Set da più fonti: basi dati o file (testuali o .dmp)
* Persistenza di dati: i Training Set sono memorizzati in un file con estensione “.dmp” da cui è possibile eseguire il caricamento in un secondo momento, nel caso se ne conosca il nome.
* Interfaccia grafica: il Client dispone di una GUI realizzata mediante il Framework JavaFX.

1. **Estensioni, differenze dal progetto originale**

E’ stata realizzata un’interfaccia grafica del Client per rendere più semplice l’interazione utente con l’applicazione KNN.

La GUI è stata realizzata con il Framework JavaFX.

1. **KNN: K-Nearest Neighbors (Regressione)**

L’algoritmo KNN viene utilizzato per stimare le variabili continue e discrete.  
Esso utilizza una media pesata dei k esempi più vicini per calcolare la predizione.

L’algoritmo si sviluppa per passi:

1. Ordina gli esempi etichettati e applica il min-max scaler sulle variabili continue.
2. Calcola la distanza di Hamming (per le variabili discrete) e la distanza euclidea (per le variabili continue) dall’esempio di query agli esempi etichettati
3. Trova il numero k di vicini più prossimi.
4. Calcola una media ponderata per la distanza con i k esempi più vicini
5. Restituisce la predizione

I Training Set sono rappresentati da tabelle composte da una o più colonne di variabili discrete e variabili continue e una colonna per le variabili di target.

1. **Diagramma delle Classi**

In allegato i diagrammi delle classi con dicitura [UML diagramma classi.pdf](UML%20diagramma%20classi.pdf)

1. **Descrizione delle classi del progetto KNN**

**Progetto Server**

Il server è composto da diversi package:

* Data: contiene le classi che modella il Training Set
* Database: contiene le classi per l’acquisizione del Training Set dal database
* Example: contiene le classi per modellare le variabili del Training Set
* Mining: contiene la classe che implementa l’algoritmo KNN
* Server: contiene le classi che gestiscono la comunicazione con il Client

Package Data

* Classe Attribute: classe astratta che modella gli attributi degli esempi.

Fornisce:

* + un costruttore per inizializzare l’oggetto che prende in input una stringa e un indice.
  + Due metodi getter astratti per ricavare il nome e l’indice dell’attributo
* Classe ContinuousAttribute: classe che estende Attribute; modella attributi continui.

Fornisce:

* + Costruttore per inizializzare l’oggetto, che prende in input una stringa e un indice.
  + Metodi per impostare il valore minimo e massimo dell’attributo, che prendono in input il valore da verificare
  + Metodo che scala il Training Set in un range [0,1] e lo restituisce
  + Metodi getter, implementazioni dei metodi della superclasse
* Classe DiscreteAttribute: classe che estende Attribute; modella attributi discreti.

Fornisce:

* + Costruttore che inizializza l’oggetto, che prende input un nome e un indice.
  + Metodi getter, implementazioni dei metodi della superclasse
* classe Data: classe che modella il Training Set  
  Fornisce:
  + Due costruttori per la costruzione del Training Set a seconda della sorgente (file o database)
  + Metodo che restituisce il numero degli attributi indipendenti
  + Metodi che ordinano il Training Set usando l’algoritmo quickSort
  + Metodo che calcola la distanza media tra gli esempi più vicini alla query dell’utente
  + Metodo che genera l’esempio creato dall’utente
  + Metodo che scala l’esempio
  + Metodo di supporto alla rappresentazione del Training Set
* Classe TrainingDataException: classe che contiene metodi per la gestione delle eccezioni che riguardano il Training Set

Package Database

* Classe Column: classe che modella le colonne delle tabelle dei database

Fornisce:

* + Costruttore che inizializza l’oggetto con il nome e il tipo della colonna
  + Metodo che restituisce il nome della colonna
  + Metodo per verificare che la colonna sia di tipo numerico
  + Metodo di supporto alla rappresentazione della colonna
* Classe DbAccess: classe che gestisce al database per la lettura dei dati di training

Fornisce:

* + Costruttore che stabilisce la connessione con il database SQL
  + Metodo che restituisce l’oggetto connessione
  + Metodo che chiude la connessione
* Classe TableData : classe che modella le tabelle del database

Fornisce:

* + Costruttore che inizializza l’oggetto, prendendo in input il dbAccess e il TableSchema
  + Metodo che costruisce la tabella
  + Metodi getter che restituiscono il contenuto delle tabelle e i valori minimi e massimi delle variabili indipendenti
* Classe TableSchema: classe che modella lo schema delle tabelle

Fornisce:

* + Costruttore che inizializza l’oggetto con il nome della tabella e il DbAccess
  + Metodi getter che restituiscono i componenti della tabella
* Classe QUERY\_TYPE: enum di supporto per la scoperta del minimo e massimo delle colonne del database
* Classi di eccezioni che gestiscono eventuali problemi di connessione al database o di incongruenze tra tabelle e Training Set

Package Example

* Classe Example: classe che modella gli esempi del Training Set

Fornisce:

* + Il costruttore che inizializza la lista dell’esempio con la dimensione passata in input
  + Metodi per il popolamento della lista dell’esempio
  + Metodi getter che restituiscono il valore in una determinata posizione della lista e la lunghezza di quest’ultima.
  + Metodo di supporto all’ordinamento
  + Metodo che calcola la distanza tra l’esempio passato in Input e l’oggetto della classe
  + Metodo di supporto per la rappresentazione dell’esempio
* Classe ExampleSizeException: classe che gestisce problemi relativi alla dimensione degli esempi

Package Mining

* Classe KNN: classe che gestisce il training set, la predizione degli attributi del training set più vicini all'esempio inserito dall'utente e i metodi per salvare e caricare l'oggetto stesso in un file binario

Fornisce:

* + Costruttore che inizializza il Training Set
  + Metodo per la predizione
  + Metodi per il salvataggio e il caricamento dell’oggetto in / da file
  + Metodo di supporto per la rappresentazione del Training Set

Package Server

* Classe MultiServer: classe che gestisce le nuove connessioni con i Client

Fornisce:

* + Costruttore che inizializza la porta di rete e procede all’avvio del Server
  + Metodo che avvia il Server e gestisce i nuovi Client in arrivo
  + Metodo Main che avvia il programma
* Classe ServerOneClient:

Fornisce:

* + Costruttore che inizializza i campi e avvia la connessione con il Client, prendendo in input la Socket
  + Metodo che gestisce effettivamente la comunicazione con il client

**Progetto Client**

Package application

* Classe Client: classe che gestisce la comunicazione con il server.

Fornisce:

* + Costruttore che inizializza la connessione con l’indirizzo e la porta di rete, presi in Input.
  + Metodo che inviano e ricevono messaggi a/da Server
  + Metodo che termina la connessione con il Server
* Classe ClientApplication: classe che modella l’interfaccia grafica utente

Fornisce:

* + Metodo che avvia l’interfaccia grafica e imposta la prima scena
  + Metodi che definiscono nuove scene, tra cui: nome della tabella, ricezione Training Set, costruzione dell’esempio dell’utente, predizione.
  + Metodo Main che avvia il programma
* File application.css : file di supporto per la grafica dell’interfaccia