Università degli Studi di Bergamo

SCUOLA DI INGEGNERIA
Corso di Laurea Magistrale in Ingegneria Informatica

Sistema di gestione prenotazioni

Progetto corso di Programmazione Avanzata

Candidato
Wasim Essbai
Matricola 1060652

Anno Accademico 2021–2022

Indice

Introduzione **1**

1 C++ **3**

C++ **3**
1.1 Features 3
1.2 Funzionamento 3
1.2.1 Entità principali 3
1.2.2 Dinamica 4

2 Haskell **5**
2.1 Algoritmo di learning 5
2.2 Funzionamento 5
2.2.1 Dinamica 5

Elenco delle figure

1.1 Diagramma delle classi. 4
1.2 Diagramma delle attività. 4

2.1 Diagramma delle attività. 6

Introduzione

Il progetto realizzato consiste in due programmi: Il primo sviluppato con il linguaggio C++, basato sul paradigma della programmazione ad oggetti, mentre il secondo in Haskell, seguendo il paradigma della programmazione funzionale. In seguito vengono descritti entrambi i programmi con le loro funzionalità.

Capitolo 1

C++

Il programma scritto con il linguaggio C++ consiste in un sistema di gestione per la prenotazione di tutorati. Il software usa diversi costrutti del linguaggio come la coppia costruttore/distruttore, ereditarietà pubblica/private, modificatori di visibilità dei membri di una classe, template e *smart pointers*. Inoltre, si fa uso anche della libreria *Standard Template Libray*, offerta dal linguaggio.

1.1 Features

Le lezioni vengono classificate in tre categorie principali:

1. Singola: Lezione tenuta singolarmente, dove viene affiancato un tutor ad uno e un solo studente;
2. Collettiva: Lezione collettiva, dove il tutor può seguire uno o più studenti;
3. Ricevimento aperto: Si tratta di una lezione "in senso lato", ovvero vengono messi a disposizione uno o più tutor e gli studenti si possono recare per chiedere supporto senza prenotazione;

Le funzionalità offerte dal programma sono:

1. Stampa di tutte le lezioni;
2. Stampa delle lezioni filtrate per categoria;
3. Inserimento di una nuova lezione;

1.2 Funzionamento

Di seguito viene descritto il funzionamento del programma e la sua struttura interna.

1.2.1 Entità principali

Le tre tipologie di lezione rappresentano le entità fondamentali del programma. In figura Fig. 1.1 viene riportato il diagramma delle classi con descritte le interfacce. In particolare, è presente una classe base che è Lezione e che contiene i campi comuni a tutte le lezioni. Dopodiché ci sono le classi che rappresentano effettivamente le entità di interessa e che sono derivate da Lezione. In particolare, LezioneSingola e LezioneCollettiva eredita in modo pubblico da Lezione mentre RicevimentoAperto in modo private, questo perché quest'ultima non ha tutti gli attributi di Lezione, quindi anche non tutte le funzionalità e pertanto restringe la sua interfaccia. In fine, sono indicate anche le relazioni tra le classi e insieme alle molteplicità.

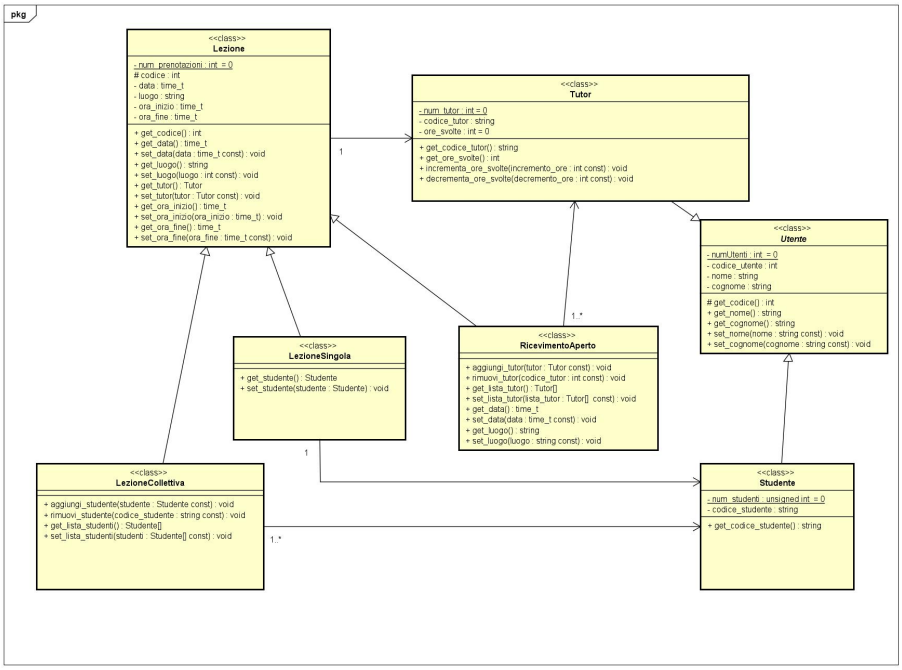

Figura 1.1 – Diagramma delle classi.

1.2.2 Dinamica

La dinamica del programma viene descritta in figura Fig. 1.2. All'avvio vengono presentato il menu da cui l'utente può scegliere l'operazione da svolgere, la scelta viene fatta inserendo un numero, che corrisponde all'operazione. La descrizione nel diagramma è ad alto livello, ovvero viene riportata la dinamica globale senza dettagliare l'algoritmo che implementa le specifiche attività. Alla scelta dell'operazione consegue l'invocazione di una procedura che la soddisfa, previa controllo dell'input, ovvero che il numero scelto corrisponda effettivamente ad un'operazione. Le procedure sono implementate in una classe dedicata chiamata Gestore.

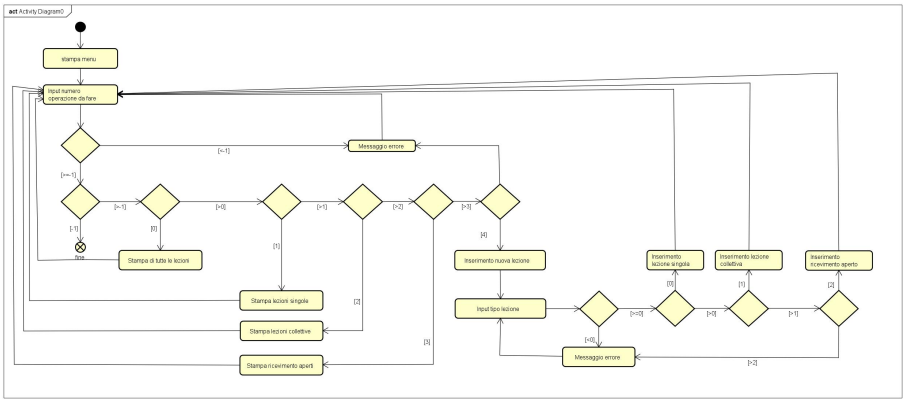

Figura 1.2 – Diagramma delle attività.

Questa non è rappresenta un'entità bensì un *controller*, che quindi implementa ed offre le varie funzionalità. Queste procedure vengono invocate dal modulo *main* del programma, dove infatti viene effettuata la scelta.

Capitolo 2

Haskell

Il secondo programma è sviluppato in Haskell ed consiste in un semplice modello *learning* che prende in input un file *.txt* con il dataset di traning. In particolare, i dati sono relativi ai pesi e altezze due categorie di animali che sono cani e gatti. L'obiettivo del modello è quello di apprendere un classificatore per distinguere tra cani e gatti in funzione di peso e altezza. Il modello di *learning* si basa su una semplice ***regressione logistica***. Le due classi sono codificate assegnando 0 se l'animale è un gatto ed 1 se è un cane.

2.1 Algoritmo di learning

Per l'apprendimento è stato implementato l'algoritmo noto come *gradient descent* che è un algoritmo iterativo usato l'ottimizzazione, in particolare la minimizzazione, per funzioni di costo con minimo globale unico. Questo algoritmo ha il vantaggio di non operare l'inversione di matrici, operazione molto costosa dal punto di vista computazionale. Se $\boldsymbol{\vartheta}$ è il vettore dei parametri da stimare allora la stima $\hat{\boldsymbol{\vartheta}}$ all'iterazione i+1 è data dalla 2.1

$$\hat{\boldsymbol{\vartheta}}_{i+1} = \hat{\boldsymbol{\vartheta}}_i - \alpha \cdot \nabla J(\boldsymbol{\vartheta})|_{\boldsymbol{\vartheta}=\hat{\boldsymbol{\vartheta}}_i} \tag{2.1}$$

2.2 Funzionamento

Di seguito viene descritto il funzionamento del programma.

2.2.1 Dinamica

In figura 2.1 viene riporta il diagramma delle attività del programma che mostra la sua dinamica e le operazioni svolte. Il flusso di controllo è molto semplice, all'avvio viene letto il file in input e rappresentato mediante una matrice. A tal fine viene usata la libreria **Data.Matrix**. Una volta gestito l'input viene eseguita la fase di *learning* che consiste nella stima dei parametri necessari alla classificazione. In particolare, viene applicato l'algoritmo *gradient descent* per minimizzare la funzione di costo. Una volta ottenuta stimati i parametri inizia la parte di interazione con l'utente, dove è possibile inserire dei valori di peso e altezza, viene quindi fatta la classificazione e restituito il risultato in output. Il programma termina quando viene inserito un valore non positivo.

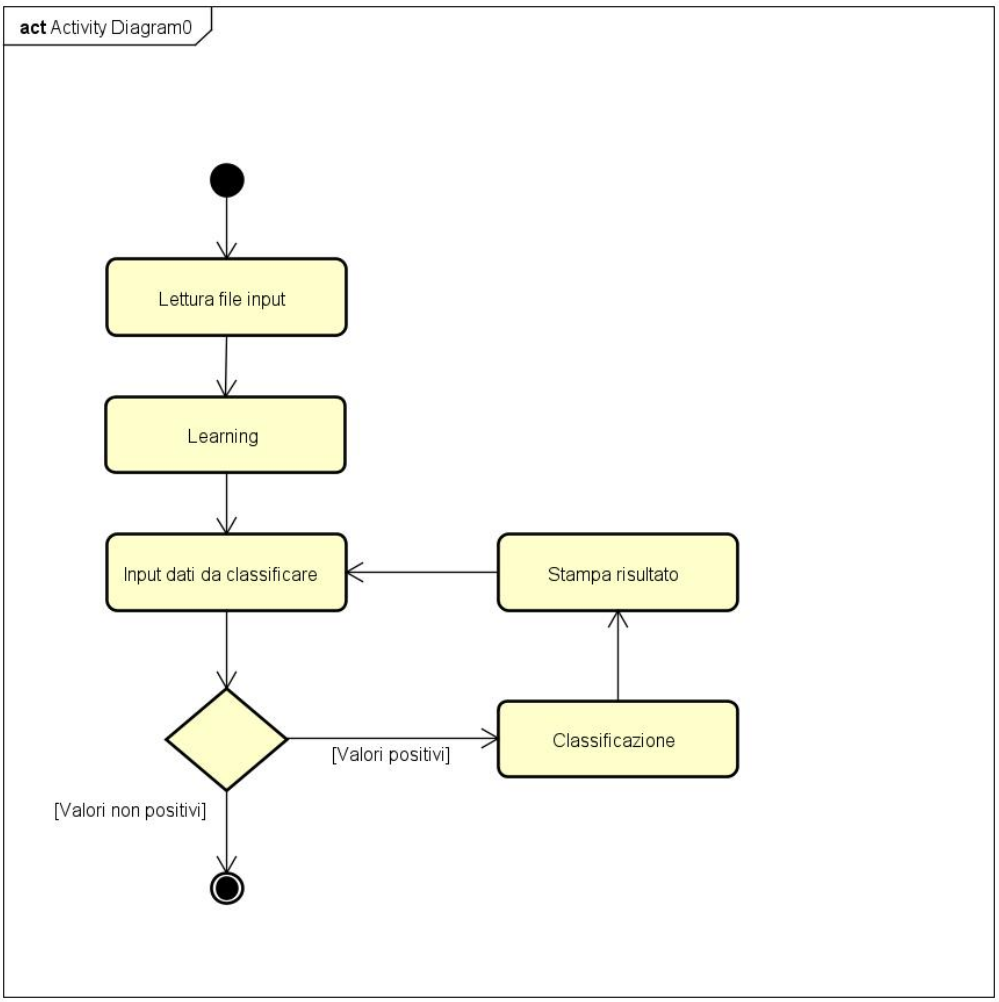

Figura 2.1 – Diagramma delle attività.