Università degli studi di Salerno
Dipartimento di Informatica
Corso di Laurea in Informatica

FONDAMENTI DI INTELLIGENZA ARTIFICIALE

Documentazione Progetto

"RistoManager".

Docenti:

Fabio Narducci
Fabio Palomba

Studente:

Nome	***Matricola***
Benitozzi Simone	0512105742

Link al repository GitHub:
https://github.com/sambrosio17/RistoManager

Anno Accademico: 2020/21

INDICE

1. Obiettivo del Sistema

L’obiettivo del sistema è quello di venir incontro alle esigenze dei ristoratori che, a seguito della diffusione del COVID-19, necessitano sempre più di sistemi sicuri, nonché una rapida gestione di quelle che sono le tipiche esigenze di un’attività ristorativa.

Lo scopo, quindi, è quello di informatizzare la gestione del locale attraverso un sistema che permetta non solo di limitare al massimo i contatti col personale di sala, ma anche di tener traccia della clientela per poi poter avvisare in caso di esposizione al contagio.
Il sistema permette al cliente di accedere al menù, comporre il proprio ordine e mandare la comanda in cucina, il tutto mediante il proprio dispositivo.

In particolare, il modulo riguardante il corso di Fondamenti di Intelligenza Artificiale consiste nell’implementazione di un sistema di profilazione degli utenti, che andrebbe a consigliare al cliente attuale, a seguito delle scelte effettuate, oltre che dei feedback rilasciati, una possibile scelta di portate o bevande, in base anche a ciò che è attualmente ordinabile dal menù, raggruppando i prodotti disponibili sulla base di caratteristiche e valori nutrizionali simili.

2. Specifica dell'Ambiente – PEAS

Questa sezione fornisce una descrizione della rappresentazione schematica dell'ambiente in esame e nel dettaglio dei 4 indicatori PEAS: Performance Measure, Environment, Actuators, Sensors.

2.1 Performance Measure

L'obiettivo del sistema è quindi quello di consigliare prodotti che abbiano la più alta probabilità di poter essere scelti dal cliente.
Pertanto la misura di performance del sistema si basa sull'accuratezza dei consigli forniti a ciascun utente. Quanto più il raggruppamento dei prodotti sarà preciso, sulla base delle loro caratteristiche, tanto più sarà probabile che i prodotti consigliati saranno effettivamente graditi al cliente.

2.2 Environment

Gli oggetti di cui l'ambiente è costituito sono innanzitutto i clienti che fanno uso del sistema.
Ci sono poi i prodotti del menu, che ciascun cliente ha la possibilità di consultare per effettuare delle scelte.
Sono inoltre tracciate le scelte di un utente, sia quelle parziali, che devono essere confermate, rappresentate dalle comande, che quelle finali, rappresentate dal riepilogo. In entrambi i casi si tratta di insiemi di prodotti.

2.3 Actuators

Gli attuatori del sistema sono rappresentati dalle operazioni che vengono svolte lato server dal programma, che, elaborando le scelte effettuate dal cliente, calcolano un indice di gradimento, sulla base del quale danno in output i consigli per il cliente.

2.4 Sensors

I sensori attraverso i quali il sistema riceve gli input percettivi sono le operazioni che il cliente effettua sull'interfaccia grafica dell'applicazione, attraverso pulsanti a cui egli può accedere per esprimere le proprie scelte.

3. Proprietà dell’ambiente

In questa sezione sono elencate le proprietà dell’ambiente in esame:

- **Completamente Osservabile:** i sensori dell’ambiente danno accesso, in ogni momento, allo stato completo dell’ambiente in ogni momento. Tutti i prodotti sono infatti accessibili, con tutte le loro caratteristiche, e allo stesso modo lo sono i feedback dell’utente a cui deve essere consigliata una possibile scelta di prodotti.

- **Deterministico:** lo stato successivo dell’ambiente è determinato dallo stato corrente e dall’azione eseguita dall’agente. I consigli variano infatti sulla base di quelle che sono le scelte e i feedback dell’utente che usufruisce del sistema.

- **Sequenziale:** l’esperienza dell’agente è costituita da una serie di azioni sequenziali, correlate tra loro e la scelta di ciascuna azione dipende dall’insieme delle scelte effettuate fino a quel momento, quali la scelta di una portata o un feedback su un prodotto già acquistato.

- **Statico:** l’ambiente viene modificato solo dalle azioni che l’agente adopera su di esso, e nel momento in cui esso sta deliberando, l’ambiente resta invariato. Gli unici casi in cui l’ambiente, e con esso gli output, variano è nel momento in cui l’agente effettua delle operazioni, andando a variare l’indice di gradimento dell’utente rispetto a determinati prodotti.

- **Discreto:** l’ambiente prevede un numero limitato di percezioni e azioni distinte e definite, che corrispondono alle scelte che l’utente intraprende nell’utilizzo del sistema.

- **Singolo Agente:** l’ambiente prevede un singolo agente che opera individualmente su di esso. Non sono previste situazioni di cooperatività o competizione con altri agenti.

4. Dati a disposizione

Per il training del sistema, è stato fatto uso di un dataset rappresentante un menu di 260 elementi, contenente per ciascuno di essi caratteristiche e dati rappresentanti diversi tipi di valori nutrizionali, senza dati mancanti.
Tale dataset è stato scaricato dal seguente link:
https://data.world/devonsepeda/mcmenu/workspace/file?filename=menu.csv#

Nonostante non si tratti di un dataset particolarmente grande in linea generale, contestualizzato al problema in questione, ovvero quello di fornire un menu per un ristorante, è stato ritenuto un numero adeguato, non sono per questo state aggiunte ulteriori istanze al dataset rispetto a quelle originariamente presenti.

Fin dalle prime esecuzioni degli algoritmi di clustering, che sarebbero poi stati implementati nel codice, risultava evidente un problema di rumore causato dai prodotti della categoria “Salads”, che pur essendo pochi (6 in totale), portavano gli algoritmi a creare sempre un cluster di piccole dimensioni esclusivamente per loro. Per questo motivo le instanze di tale categoria sono state rimosse.

Sono state inoltre effettuate operazioni sul dataset, in particolare sugli attributi, attraverso una fase iterativa, descritta di seguito, che ha portato ad una versione finale più affinata e meglio rappresentativa dei dati necessari alla risoluzione del problema.

4.1 Dataset originale

Il dataset originale è costituito da 24 attributi (colonne) elencati di seguito:

1. Category
2. Item
3. 'Serving Size'
4. Calories
5. 'Calories from Fat'
6. 'Total Fat'
7. 'Total Fat (\% Daily Value)'
8. 'Saturated Fat'
9. 'Saturated Fat (\% Daily Value)'
10. 'Trans Fat'
11. Cholesterol
12. 'Cholesterol (\% Daily Value)'
13. Sodium
14. 'Sodium (\% Daily Value)'
15. Carbohydrates
16. 'Carbohydrates (\% Daily Value)'
17. 'Dietary Fiber'
18. 'Dietary Fiber (\% Daily Value)'
19. Sugars
20. Protein
21. 'Vitamin A (\% Daily Value)'
22. 'Vitamin C (\% Daily Value)'
23. 'Calcium (\% Daily Value)'
24. 'Iron (\% Daily Value)'

4.2 Rimozione attributi ridondanti

La prima cosa che salta all'occhio è la presenza di alcuni dati, che già dal nome si capisce come siano strettamente legati tra loro, e che difatti, analizzando i valori delle varie istanze risultano crescere in maniera direttamente proporzionata tra loro.
Si tratta di quelle coppie di attributi come [*'Total Fat' ; 'Total Fat (\% Daily Value)'*], che esprimono prima il valore totale nutrizionale totale, e successivamentelo stesso valore rapportato al fabbisogno giornaliero. Nello specifico :

1. Category
2. Item
3. 'Serving Size'
4. Calories
5. 'Calories from Fat'
6. 'Total Fat'
7. 'Total Fat (\% Daily Value)'
8. 'Saturated Fat'
9. 'Saturated Fat (\% Daily Value)'
10. 'Trans Fat'
11. Cholesterol
12. 'Cholesterol (\% Daily Value)'
13. Sodium
14. 'Sodium (\% Daily Value)'
15. Carbohydrates
16. 'Carbohydrates (\% Daily Value)'
17. 'Dietary Fiber'
18. 'Dietary Fiber (\% Daily Value)'
19. Sugars
20. Protein

21. 'Vitamin A (\% Daily Value)'
22. 'Vitamin C (\% Daily Value)'
23. 'Calcium (\% Daily Value)'
24. 'Iron (\% Daily Value)'

Risulta quindi opportuno rimuovere questi attributi ridondanti, lasciando, per ciascuna coppia, uno solo degli attributi. Si è optato per mantenere, per ciascuna coppia, il valore nutrizionale rapportato al fabbisogno giornaliero. Ne risulta una nuova configurazione di attributi del dataset:

1. Category
2. Item
3. 'Serving Size'
4. Calories
5. 'Calories from Fat'
6. 'Total Fat (\% Daily Value)'
7. 'Saturated Fat (\% Daily Value)'
8. 'Trans Fat'
9. 'Cholesterol (\% Daily Value)'
10. 'Sodium (\% Daily Value)'
11. 'Carbohydrates (\% Daily Value)'
12. 'Dietary Fiber (\% Daily Value)'
13. Sugars
14. Protein
15. 'Vitamin A (\% Daily Value)'
16. 'Vitamin C (\% Daily Value)'
17. 'Calcium (\% Daily Value)'
18. 'Iron (\% Daily Value)'

4.3 Conversione dell'attributo 'Serving Size'

L'attributo serving size si presentava originariamente sotto forma di stringa, nel formato ['*4.8 oz (136 g)*'].
Dal momento che questo tipo di formato avrebbe dato problemi nell'interpretazione dei dati, rendendo il dato, che rappresenta una caratteristica di fondamentale importanza nel raggruppamento dei prodotti del dataset, inutilizzabile, si è optato per convertire il dato di ciascuna istanza da stringa a double, estraendo la prima sottostringa e convertendola in un numero reale, rappresentente proprio il peso della portata espressa in oz.
Questo intero processo è stato effettuato in maniera programmatica e automatizzata per ciascuna istanza del dataset, come descritto nelle sezioni successive del documento.

4.4 Aggiunta dell'attributo 'Prezzo'

Nel dataset originale mancava un attributo fondamentale riferito a ciascun prodotto, non tanto per il problema di raggruppamento delle istanze, quanto per il dominio del problema del sistema esistente: il prezzo di ciascun prodotto, che viene mostrato quando si scorre il menu nell'interfaccia grafica dell'applicazione.
A questo proposito, sempre in maniera automatizzata, è stata aggiunta una colonna ggiuntiva al dataset, rappresentante il prezzo, e a ciascun prodotto ne è stato assegnato uno, in funzione di categoria e peso.

4.5 Dataset finale

Si giunge in questo modo al dataset finale, costituito da 19 attributi:

1. Category*
2. Item*
3. Calories
4. 'Calories from Fat'
5. 'Total Fat (\% Daily Value)'
6. 'Saturated Fat (\% Daily Value)'
7. 'Trans Fat'
8. 'Cholesterol (\% Daily Value)'
9. 'Sodium (\% Daily Value)'
10. 'Carbohydrates (\% Daily Value)'
11. 'Dietary Fiber (\% Daily Value)'
12. Sugars
13. Protein
14. 'Vitamin A (\% Daily Value)'
15. 'Vitamin C (\% Daily Value)'
16. 'Calcium (\% Daily Value)'
17. 'Iron (\% Daily Value)'
18. 'Serving Size'
19. Price*

*Nell'esecuzione dell'algoritmo di clustering, successivamente descritto, sono stati inoltre ignorati gli attributi marcati, le motivazioni sono espresse successivamente.

5. Contestualizzazione del Problema

Sulla base delle osservazioni fatte finora, la successiva scelta da effettuare è quella di associare il problema alla specifica categoria di apprendimento a cui esso appartiene.

I dati a disposizione, analizzati nella precedente sezione, non presentano un'etichetta o una struttura ben definita, che ci consentano di definire una interpretazione precisa di ciò che rappresentano.

Siamo quindi di fronte ad un problema di apprendimento non supervisionato, le cui tecniche ci consentono di sfruttare i dati a disposizione, in maniera tale che la macchina li possa elaborare ed estrarne informazioni rilevanti. Non ci consentono, invece, di contare su una variabile nota relativa al risultato o su di una funzione di ricompensa da asseganare all'agente a seguito di un'azione intrapresa.

Il *Clustering* è una tecnica esplorativa che cerca regolarità nei dati disponibili, e che consente di aggregare all'interno di gruppi (detti *cluster*), dati dei quali non abbiamo precedente conoscenza di appartenenza a gruppi, e che presentano caratteristiche simili tra loro.

Si tratta della tecnica più comunemente utilizzata nell'apprendimento non supervisionato, e quella che nel contesto del problema corrente risulta essere la più adeguata.

6. Scelta del Software

Scelta la tecnica di analisi dei dati, c'è ora la necessità di scegliere il software con cui il tutto dovrà essere concretamente trasformato in codice ed eseguito.

Il linguaggio di programmazione più comune ed utilizzato per problemi di Machine Learning è sicuramente python, ma, dal momento che il sitema attuale, in cui il modulo descritto dovrà essere integrato, è interamente scritto in Java, attraverso l'interazione di classi Bean, Servlet e JSP, la scelta è ricaduta sull'utilizzo di librerie scritte anch'esse in Java.

Il software scelto è stato Weka, un framework open source di machine learning, accessibile sia attraverso interfaccia grafica, che API Java.

Nel corso della progettazione sono state utilizzate entrambe le opzioni, per diversi tipi di operazioni. Attraverso l'interfaccia grafica è stato infatti possibile effettuare, in maniera molto agevole, le operazioni di manipolazione del dataset, specialmente sulla rimozione di determinate colonne, descritte nella sezione 4 del documento.

7. Scelte implementative

In questa sezione sono elencate le principali scelte, motivate, che hanno portato all'implementazione del sistema.

7.1 Conversione del formato del Dataset

Il file originale, rappresentativo del dataset, era in formato *.csv*, e dal momento che le API di Weka lavorano di default su file di tipo *.arff,* il dataset è stato convertito in tale formato, attraverso l'*explorer* dell'applicazione grafica di Weka.

7.2 Scelta dell'algoritmo

La scelta dell'algoritmo di clustering più adatto al problema corrente è ricaduta su uno tra Kmeans e DBScan. Tra i due, attraverso anche un'osservazione degli output ottenuti a seguito di diverse esecuzioni, quello che maggiormente è sembrato essere adatto al problema corrente si è rilevato il Kmeans.
Il motivo principale è senz'altro il fatto che DBScan non sia in grado di gestire correttamente dataset multidimensionali, come nel caso corrente, in cui il database conta 16 variabili (che vedremo successivamente diventare 8, dopo la fase di preprocessing e l'applicazione di PCA), cosa che invece riesce bene al Kmeans.
Lo svantaggio del Kmeans è quello di trovare difficoltà con dataset che presentino dati rumorosi, ma nelle precedenti sezioni è stato già descritto come questi, in parte, siano già stati rimossi in principio.

7.3 Scelta di K

Il risultato dell'algoritmo Kmeans, dipende fortemente dall'input K, corrispondente al numero di cluster che dovranno essere formati a partire dai dati a disposizione.

La libreria Weka non mette a disposizione una funzione che permetta di valutare il risultato di un algoritmo di clustering non supervisionato.
Pertanto ho implementato io direttamente, un algoritmo che valuti il risultato dell'esecuzione di Kmeans, in termini di Silhouette Score (coefficiente di forma).

Ho eseguito ripetute chiamate all'algoritmo, passando ogni volta valori di K e Seed diversi, in maniera tale da trovare la miglior combinazione possibile, sulla base dei dati a disposizione.

Di seguito il risultato ottenuto
[Classe: *it.RistoManager.FIA.KMeansEvaluator*]:

```
System.out.println("\nMAX = " + max + "; BEST CLUSTERS = " + bestNumClusters + "; BEST SEED = " + bestSeed);

// Risultato:
// MAX = 0.3733685131882255; BEST CLUSTERS = 6; BEST SEED = 3
```

Pertanto sono stati scelti i valori K = 6 e Seed = 3.

7.4 Riduzione degli attributi e PCA

Come visto nella sezione 4 del documento, la prima fase della progettazione ha previsto lo sfoltimento dei dati del database, rimuovendo gli attributi che risultavano evidentemente correlati tra loro.

Oltre ad essi, prima dell'esecuzione dell'algoritmo Kmeans, vengono applicati dei filtri che permettono di ridurre ulteriormente il numero di variabili e quindi la dimensionalità del dataset.

Innanzitutto è stato rimosso l'attributo contenente il nome del prodotto, trattandosi di una stringa con cui Kmeans non può lavorare.

A questo punto, dalle varie esecuzioni risultava evidente come la suddivisione in cluster dipendesse troppo dall'attributo *"Category"*, il che non fa al caso di quello che è l'obiettivo del sistema, ovvero consigliare ad un cliente delle combinazioni di diversi tipi di portate, ove possibile.

Per evitare, quindi, che scelto un prodotto, si continuassero a consigliare al cliente prodotti simili a quelli già consumati, tale attributo è stato rimosso, e lo stesso è stato fatto per *"Price"*, il quale, come descritto prima, è stato calcolato in funzione di categoria e peso del prodotto.

Tolti questi 3 attributi, si passa ad un dataset di 16, che è comunque un numero abbastanza elevato per il problema in questione. Prima di eseguire l'algoritmo Kmeans, viene quindi eseguito un filtro sul dataset che gli verrà dato in input, che attraverso la Principal Component Analysis, consente una semplificazione dei dati cercando un nuovo insieme di variabili con dimensione minore dell'insieme originario e comunque rappresentativo del contenuto informativo del dataset.

Tale filtro è stato applicato attraverso le API messe a disposizione da Weka, e si nota come il numero di variabili viene dimezzato, passando da 16 ad 8, garantendo un grosso vantaggio dal punto di vista dell'esecuzione e anche dell'efficienza a runtime del sistema.

7.5 Consigli all'utente

Ottenuto così il clustering, esso va a questo punto utilizzato per quello che era lo scopo iniziale del sistema: consigliare prodotti del menu ai clienti.

Il sistema prevede, in una situazione reale, un numero limitato di scelte da parte di ciascun cliente, che verosimilmente ordinerà un piccolo sottoinsieme di quelli che sono tutti i prodotti che formano il menu.
Pertanto risulta necessario che ogni singola scelta da parte del cliente, abbia un'influenza relativamente forte su quelli che saranno i consigli che il sistema dovrà generare.

A questo proposito ho impementato un algoritmo semplice, ma che fa al caso del problema in questione
[Classe: *it.RistoManager.Control.Menu.ConsigliaProdotto*]:

- a ciascun cliente viene associato un array di interi, in cui ogni indice corrisponde ad un cluster, il cui valore di ciascuna entry rappresenta la probabilità che un prodotto del cluster *i-esimo* venga consigliato al cliente.
- inizialmente tutte le entry saranno settate a 100, il che si traduce nel fatto che per ogni cluster, se il cliente non ha effettuato ancora alcuna scelta, verrà consigliato sicuramente un prodotto.
- quando un prodotto viene scelto dal cliente, il sistema lo interpreterà come un segno di gradimento e pertanto alzerà la probabilità che i prodotti del cluster a cui esso appartiene vengano consigliati, di un valore arbitrario (+25), abbassando complessivamente, nella stessa quantità, la percentuale relativa a tutti gli altri cluster (-5 per ciascuno). In questo modo la percentuale complessiva sarà sempre K*100 = 600.
- inoltre il cliente ha la possibilità di rilasciare un feedback, positivo o negativo, che varia ulteriormente il suo indice di gradimento verso i prodotti di quel cluster.
- per un cluster che abbia una probabilità >100, potrà essere consigliato anche più di un prodotto (ad esempio, se la probabilità è

del 160%, al 100% verrà consigliato un prodotto di quel cluster, al 60% anche un secondo prodotto).

8. Classi utilizzate

In questa sezione sono elencate le classi utilizzate per l'integrazione del modulo di apprendimento, con una breve descrizione per ciascuna di esse e l'elenco delle funzionalità che mettono a disposizione, utilizzabile come guida per la consultazione del codice.

8.1 Package FIA

Questo package contenente le classi introdotte ad hoc per il modulo di FIA, è costituito da 3 classi:

KmeansExecutor	
Metodo:	**Descrizione:**

static{…}	Blocco statico, chiamato la prima volta che la classe viene chiamata, che effettua gran parte delle operazioni:
	• carica il file del dataset
	• applica dei filtri sulle istanze ottenute per:
	  ○ rimuovere gli attributi non utili per il clustering
	  ○ standardizzare gli attributi numerici
	  ○ applicare la Principal Component Analysis
	• chiama il metodo *getAllProdotti()*
	• chiama il metodo *createClusters()*

createClusters()	• Esegue l'algoritmo Kmeans, sulla base dei valori K e SEED • Restituisce un array di assegnamenti, che associa, ad ogni prodotto ordinato per posizione nel dataset, il cluster corrispondente
getProdottoByLine()	• dato in input il numero di una riga, restituisce il prodotto corrispondente a quella riga nel dataset

getProdottoById()	• dato in input un id, restituisce il prodotto corrispondente a quell'id nel database
getLineById()	• dato in input un id di un prodotto, restituisce la riga in cui è presente quel prodotto all'interno del dataset
getAllProdotti()	• restituisce tutti i prodotti del dataset
getProdottiByCategoria()	• restituisce tutti i prodotti del dataset appartenenti alla categoria data in input
getCluster()	• restituisce la lista di prodotti appartenenti ad un numero di cluster dato in input
getAssociations()	• restituisce l'array delle associazioni tra riga del dataset e id del prodotto corrispondente
getAssignments()	• restituisce l'array delle associazioni tra riga del dataset e il numero di cluster corrispondente
createProdotto()	• crea un nuovo prodotto, salvandolo nel database • chiama updateProdotto()
updateProdotto()	• aggiunge al prodotto dato in input, le informazioni aggiuntive non tracciate dal database

KmeansEvaluator	
Metodo:	**Descrizione:**
main()	• chiama ripetutamente *evaluate()* allo scopo di trovare la miglior combinazione possibile in termini di K e seed
evaluate()	• esegue un *SimpleKMeans*, utilizzando i parametri passati in input • calcola il coefficiente di forma del clustering ottenuto • restituisce il coefficiente calcolato
createData()	• carica i file dal dataset

DatasetManipulator	
Metodo:	**Descrizione:**
main()	• chiama *servingSizeConverter()* e *priceGenerator()*
servingSizeConverter()	• carica il dataset • aggiunge un attributo per la *serving size* (double) • converte per ciascuna istanza l'attributo *serving size* da stringa a double • elimina l'attributo *serving size* (stringa) • salva il nuovo dataset
priceGenerator()	• carica il dataset • aggiunge un attributo *price* (double) • per ciascuna istanza calcola il prezzo in base alla categoria a cui appartiene e al peso • salva il nuovo dataset

8.2 Package Model

Di questo package, contenente gli ogetti bean del codice, verranno illustati solamente quelle che trovino applicazione nel modulo di FIA:

Subpackage: Model.Entity	
CLASSE	**FUNZIONALITÀ**
ClienteBean	Rappresenta un cliente del locale. Mantiene un array di interi, della size del numero di cluster, in cui ciascuna entry rappresenta la probabilità che un prodotto del corrispondente cluster gli venga consigliato.
ProdottoBean	Rappresenta un prodotto del menu, mantiene quindi le informazioni di ciascuna entità del dataset.

8.3 Package Control

Di questo package, contenente le Servlet, verranno illustate solamente quelle che trovano applicazione nel modulo di FIA:

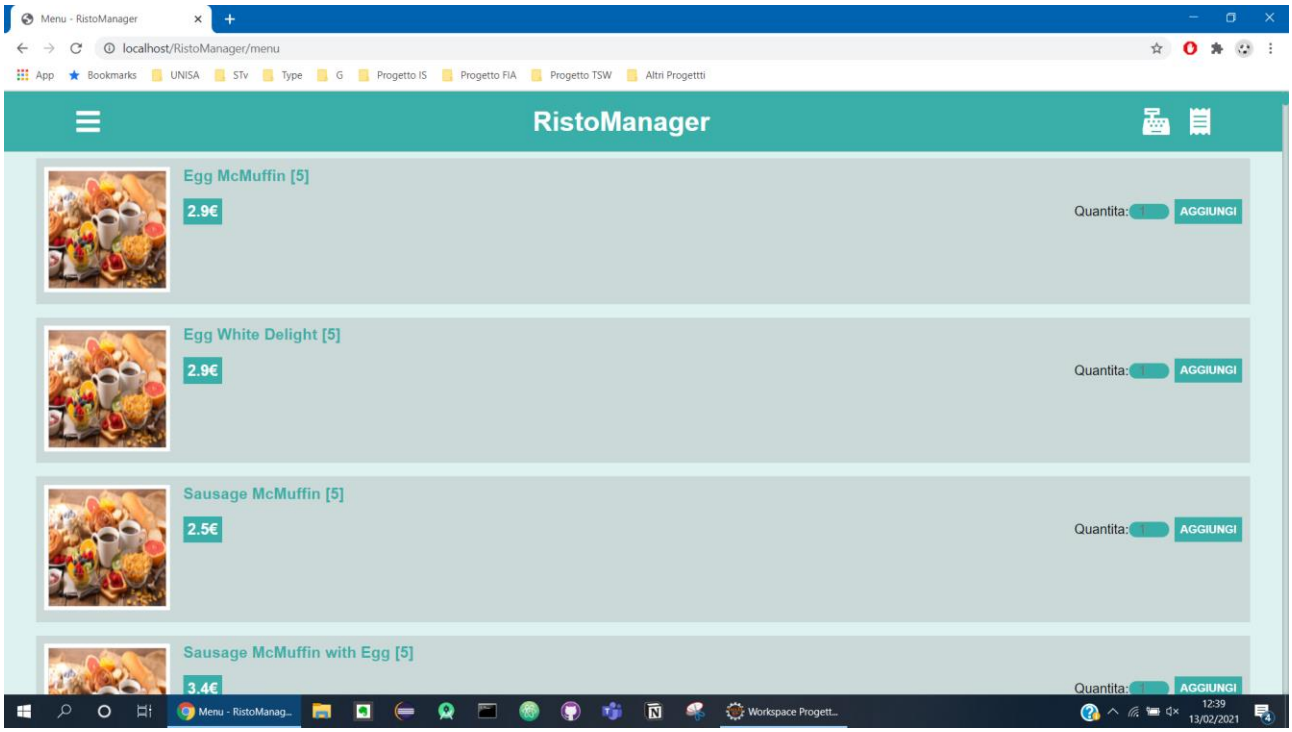
Subpackage: Control.Comanda	
CLASSE	**DESCRZIONE**
InviaComanda	Permette di inviare la comanda alla cucina. Nel contesto del modulo di FIA, quando un cliente invia una comanda, confermando quindi le scelte fatte, tale operazione verrà considerata dal sistema come un gradimento per quel prodotto, e quindi l'indice di gradimento dell'utente, riferito al cluster a cui il prodotto appartiene, verrà incrementato.
Feedback	Permette al cliente di rilasciare un feedback, positivo o negativo, su un prodotto ordinato. In base al tipo di feedback l'indice di gradimento dell'utente, riferito al cluster a cui il prodotto in questione appartiene, verrà incrementato o decrementato.

Subpackage: Control.Menu	
CLASSE	**DESCRZIONE**
ConsigliaProdotto	Sula base dell'indice di gradimento dell'utente per ciascun cluster, estre da ciascuno di essi un numero di prodotti variabile, sulla base di tali percentuali, e restituisce una lista dei prodotti così ottenuti.

Subpackage: Control.Utente.Cliente	
CLASSE	**DESCRZIONE**
DispatchCliente	Nel caso in cui un cliente che aveva già effettuato l'accesso, ritorna sulla piattaforma, in base alle ordinazioni effettuate precedentemente, gli indici di gradimento per ciascun cluster saranno aggiornati di conseguenza.

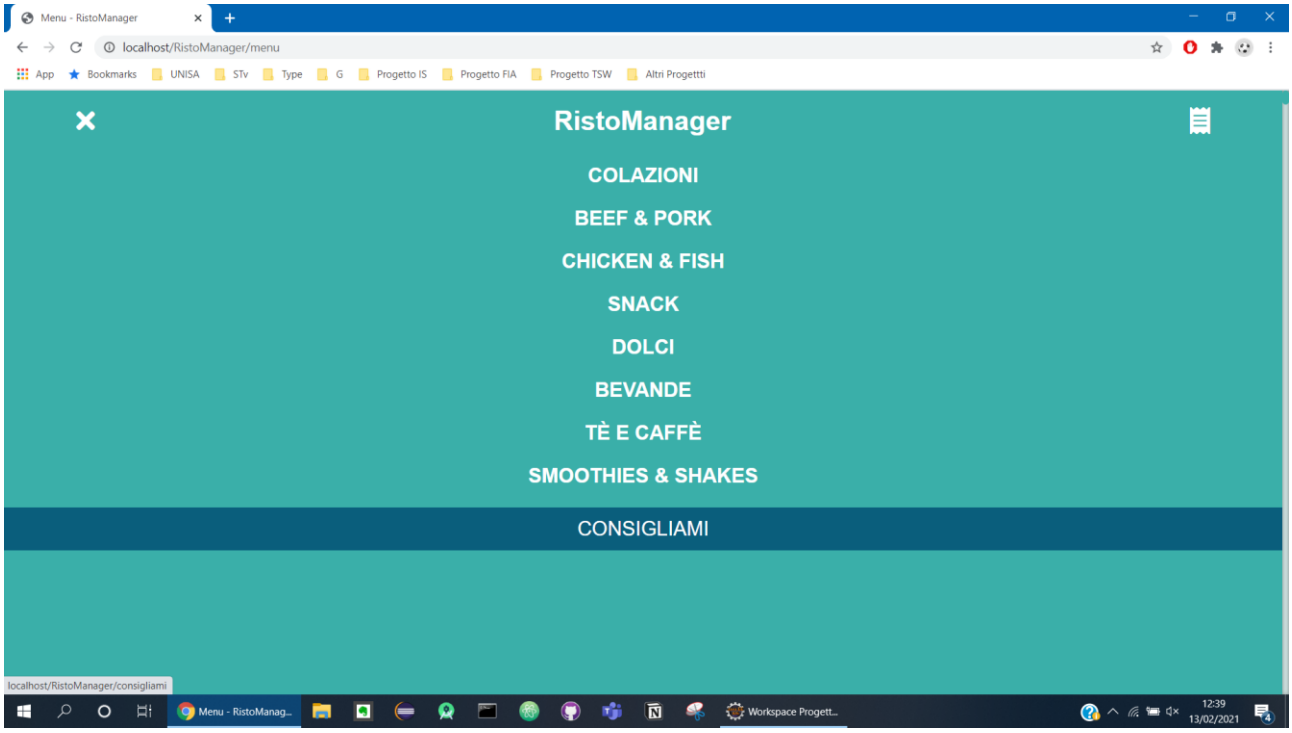
9. Esempio di utilizzo

Il cliente può scegliere i proodotti tra quelli presenti sul menu:

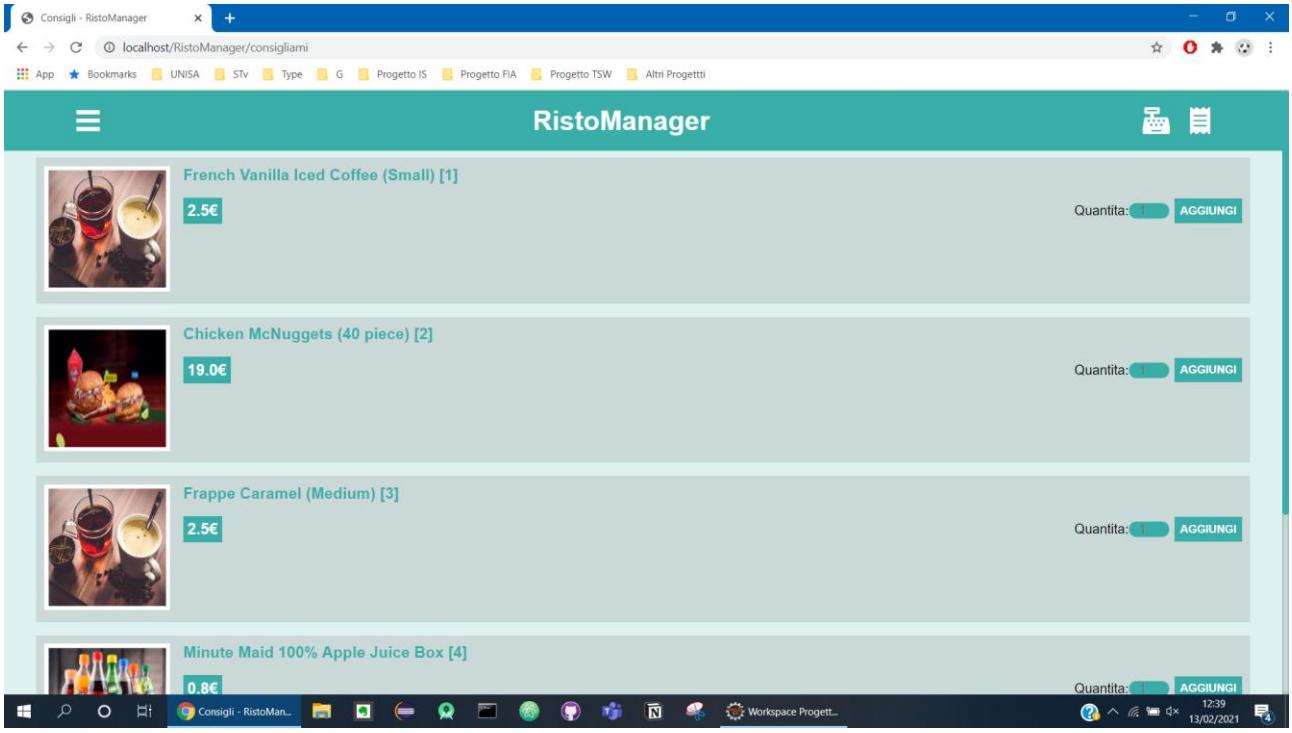


**Il numero presente tra parentesi [] presente dopo il nome del prodotto, indica a quale cluster appartiene quel prodotto. È stato inserito solo a scopo dimostrativo, per evidenziare il comportamento del sistema a seguito delle scelte del cliente*

Può inoltre filtrare il menu in base alla categoria dei prodotti, o scegliere di farsi consigliare dei prodotti dal sistema:



Il sistema consiglierà dei prodotti, a partire dai cluster disponibili, sulla base delle scelte del cliente:



Il cliente potrà rilasciare dei feedback sui prodotti consumati: