International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2020): 7.803

Restaurant Recommender System for VIT Students

S. M. Jaisakthi[1], Prafful Mundra[2], Vartika Trivedi[3], Anukriti Baijal[4], Peri Nagasri Anusha[5], Mridula Menon[6]

[1]Associate Professor Grade 2 Vellore Institute of Technology, Vellore, India

[2]Student of SCOPE Vellore Institute of Technology Vellore, India
praffulmundra30[at]gmail.com

[3]Student of SCOPE Vellore Institute of Technology Vellore, India
vartika.trivedi2018[at]vitstudent.ac.in

[4]Student of SCOPE Vellore Institute of Technology Vellore, India
anukriti.baijal2018[at]vitstudent.ac.in

[5]Student of SCOPE Vellore Institute of Technology Vellore, India
perinagasri.anusha2018[at]vitstudent.ac.in

[6]Student of SCOPE Vellore Institute of Technology Vellore, India
pmridula.menon2018[at]vitstudent.ac.in

Abstract: ***When students go to a university, they tend to try different. Restaurants, even messes. When they order food, they crave a satisfaction, which is offered by the taste of food, amidst all the classes and assignments. If the food isn't satisfactory, they tend to get disappointed. We created a restaurant data mining system for the students of Vellore Institute of Technology, Vellore. It categorizes restaurants into three categories: Inside VIT, Outside VIT, Messes. It displays the menu of each restaurant along with their categories as vegetarian and non vegetarian. It also contains the price and rating of the food item. The user can view, update, insert or delete restaurants and its corresponding food items. The system displays the top 2 recommended foods from both veg and non - veg, which are refreshed in real - time upon updating, insertion or deletion of food items. The recommendations are made by mining through the data using hash - maps, array lists and observable lists. The recommendations are made taking into account both the price and rating of a food item. A low - priced item, say tea, or soda, is dealt with separately.***

Keywords: Restaurant Recommender, JAVA, hashmaps

1. Introduction

Creating a user - friendly database of restaurants available in and around VIT University using data mining with the help of data structures. This program identifies the popular food items from a wide variety of food items from various restaurants. Data mining is the process of discovering patterns in large data sets. It is a process used by companies to turn raw data into useful information. Data mining depends on effective data collection, warehousing and computer processing. In the current technology evolution of various applications of data mining requires efficiency which can be achieved in various ways by improving the steps of the data mining techniques. Current research shows a wide opening in the step that uses data structure of data mining methodology.

sing data structures, the computation operations involved in the data mining techniques can be improved. Efficient data structures make a data mining methodology more effective. The data mining process breaks down into five steps:

1) Organizations collect data and load it into their data ware - houses.
2) They store the data.
3) Then they analyze it.
4) The application software sorts the data based on the user's results.

The project has several attention - to - detail features for user convenience. If the user wants to view a restaurant, instead of entering all the details he can just click on the restaurant on the table view and display the restaurant. Similarly, when viewing or updating a food item, the user can just click on the item in the table view to fill in the textboxes with the details of the item.

The programming efficiently to reduce response time and manage space efficiently. If suppose a user deletes a restaurant, the corresponding food table in the database is also deleted. If a new restaurant is added, a corresponding food table is created in the data - base.

The functionalities of the project include:

1) Check out the restaurants and messes in and outside of VIT.
2) Add new restaurants.
3) Delete existing restaurants.
4) Add/delete food for restaurants.
5) Get recommended veg and nonveg food.
6) The food or restaurant added are refreshed upon addition.

The recommendations are also refreshed in real - time on updating food or restaurants.

2. Literature Survey

[1] When compared to the Decision Tree and Random

Forest, the evaluation of rating predictions using Deep Learning Models revealed great accuracy in performance. When compared to other models, classification errors in this model were the smallest. The highest performance was found in the evaluation of input optimization for prescriptive analytics for class rating predictions. The study adds to the knowledge on how to develop a predictive and prescriptive analytics method to a context - based recommender model. A context - based recommender system can be employed in the prediction model not only to predict rating values but also to optimize the input depending on the class rating decision. Knowing what users should do when they visit (prescriptive analytics) based on man - aged input data optimization is important and beneficial.

[2] In mobile environments, a recommender agent should be con - text - aware to assist users while they are on the move. A recommender agent can employ a variety of settings, including weather, route conditions, time and location, and so on. In a mobile environment, researchers show a prototype design of a software agent that recommends travel - related information based on the user's surroundings. The discussion between the user and the agent for altering the limitations given to the agent is part of the recommendation method. They demonstrate with a scenario in which a tailored agent in a mobile device is used to propose a restaurant to a tourist in Taipei city.

[3] Collaborative Filtering is becoming increasingly popular in the development of a simple yet effective recommender system. A Collaborative Filtering- based recommender system simply fore - casts a user's interest in a certain item based on the scores generated and the correlation calculated between the users. In this re - search, they offer a basic structure and processes for creating a recommender system that employs Collaborative Filtering (user - based) as well as data partitioning and clustering applications, resulting in the creation of a Restaurant Recommender System. The proposed system simplifies the process and provides a clear picture of how to design a recommender system from the ground up.

[4] In a world of ever - expanding information resources, knowledge - based recommender systems fill a critical gap. They don't rely on enormous amounts of statistical data on specific rated items or users, unlike other recommender systems. Their experience has proven that the knowledge component of these systems does not need to be unreasonably large, as they only require enough knowledge to judge items as being comparable. Further - more, knowledge - based recommender systems assist users in exploring and comprehending a data area. Users are an important element of the knowledge discovery process because they define their information requirements as they engage with the system. The system understands tradeoffs, category borders, and useful search algorithms in the domain, so all that is required is a general understanding of the set of things and an informal understanding of one's wants. Other forms of recommender systems benefit from the addition of knowledge - based recommender systems. They've shown one technique to build a hybrid knowledge - based/collaborative system, but this is a fruitful study area with plenty of space for further exploration.

[5] Collaborative filtering and knowledge - based filtering strategies are two very old techniques often utilized for delivering automated suggestions. When employed alone, however, each of these strategies have some downsides. In this research, researchers suggest an architecture for creating a hybrid recommender system that combines the benefits of both methodologies, resulting in increased accuracy. The suggested method combines personalized suggestions (based on an individual's past behavior), social recommendations (based on comparable users' past behavior), and item - based recommendations (based on restaurant database). When these approaches are applied together, they overcome all of the disadvantages that they have when utilized independently. They have presented the application of such a system in the restaurant industry in this study.

[6] This paper examines recent developments in recommender systems, classifies them into eight categories, including e- government, e- business, e- commerce/e-shopping, e- library, e- learning, e- tourism, e- resource services, and e - group activities, and summarizes the related recommendation techniques used in each category. It looks at the reported recommender systems from four perspectives: recommendation methods (like CF), recommender systems software (like BizSeeker), real - world application domains (like e- business), and application platforms (such as mobile based platforms). A number of important new issues have been found and are being included as new avenues. This survey will directly assist scholars and practitioners in their understanding of developments in recommender system applications by giving state - of - the - art knowledge.

[7] They offer a restaurant recommender system in a mobile context in this research. This recommender system uses a user preference model based on the features of the user's previously visited restaurants, as well as the user's and restaurants' geographical information to dynamically provide recommended results. The pro - posed recommender system is implemented using the Baidu map cloud service. The results of a case study reveal that the proposed restaurant recommender system can effectively leverage user preferences and location information to recommend tailored and appropriate eateries to various users.

[8] In this research, researchers develop a recommender system for restaurants in the Bandung area. However, today's consumers want a restaurant that has a strong reputation and caters to their preferences, therefore restaurant reviews from other users are essential in the restaurant recommendation process. They employ a user based collaborative filtering mechanism to personally recommend a restaurant based on other users' evaluations. To find the proximity between users, they employ two similarities: user rating similarity and user attribute similarity. To assess the accuracy of rating prediction, they utilize the Mean Absolute Error (MAE). For calculations without user attributes, the best MAE result is 1.492, whereas for calculations with user attributes, the best MAE result is 2.166.

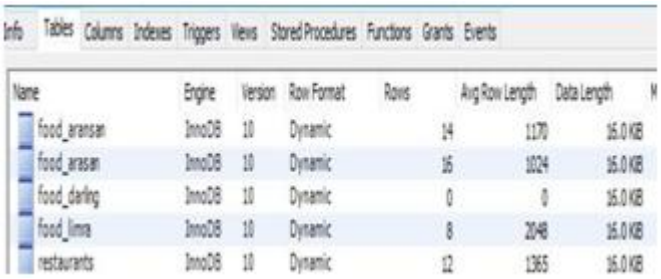
[9] Choosing a restaurant can be tough, especially in cities like New York, where there are so many options. Traveltant

is a social network - based recommender system that recommends restaurants based on the user's choices, their friends' preferences, and the restaurant's popularity. To recommend dining venues across the two social networks, Traveltant mines the interests of users and their friends from their Facebook accounts, as well as a list of restaurants and their popularity from the location - based social network Yelp. Volunteer subjects rated the recommendation results using 14 distinct models representing different combinations of factors to evaluate their approach. The findings revealed that personal preferences are the most important factor in most people's restaurant selection processes; however, friends' preferences can help a recommender system overcome some scenarios in recommender systems where personal preferences are lacking, such as the cold start problem.

3. Methodology

All hardware and software components:
H/W specifications: Any system with minimum GPU to support JavaFX
S/W specifications:
Netbeans IDE/ Any Java IDE JDBC
MySql JavaFX

Modules:
MySQL Module: This module contains the database. It con - tains a table for all the restaurants. It then contains a table for each restaurant containing the menu with price and rat - ings. It connects to the NetBeans IDE via Java Database Connectivity (JDBC). Some of the tables in the database:



Restaurants table with food table of one of the restaurants:

id	restaurantName	location
1	aransan	I
2	limra	O
3	darling	M
4	wrapsandfries	O
5	qblockshop	I
6	buddiesandbites	M
7	necafe	I
8	chickin	O
9	rangalya	O
10	benzpark	O
11	tarama	O
12	oldtown	O

item	type	price	rating
Bread Omlet	N	30	4
Chicken Tikka	N	80	4.2
Coffee	V	15	4
Egg Dosa	N	40	4
Honey Chilli Potato	V	60	4.2
Lemon Soda	V	20	4.1
Maggi	V	30	3.8
Masala Dosa	V	35	4.2
Milkshake	V	40	3.8
Naan	V	15	3.5
Pav Bhaji	V	55	3.9
Plain Dosa	V	30	4.5
Podi Dosa	V	50	4.5

- JavaFX Module: It creates an interactive GUI which is visible to the user. The user Interacts with the program dynamically via JavaFX. It has simplified the GUI creation via FXML files which can then be converted to controllers. We can create scenes (like a webpage) and then navigate between different scenes. It also has inbuilt data structures to detect events like observable lists. We can build scenes using the drag and drop actions in the FXML window. The FXML files of the main window of the project is as shown:

- JDBC Module: This module connects a database application like MySQL with the Java IDE. It makes a connection to the database and then processes queries and displays results. It can also be used to create/modify databases. If the connection is not successful, it displays the corresponding error messages.
- Recommendation Module: This module fetches data from the MySQL database via JDBC, makes recommendations for users based on the restaurant food items' price and ratings, then displays it to the user via JavaFX. The recommendations are made by making an observable list which is passed to a HashMap after calculating the recommendation parameters for sorting. Then the required number of food items are sent to an array - list for displaying separately as veg and non - veg food. Whenever an update occurs, the recommendations are refreshed. The price of an item, if less than 20, is incremented by 20 for balancing purposes. Then, the rating is divided by the price. Hash Maps are used to store

both the veg and nonveg items separately, then the items are sorted in descending order of the ratio and displayed.

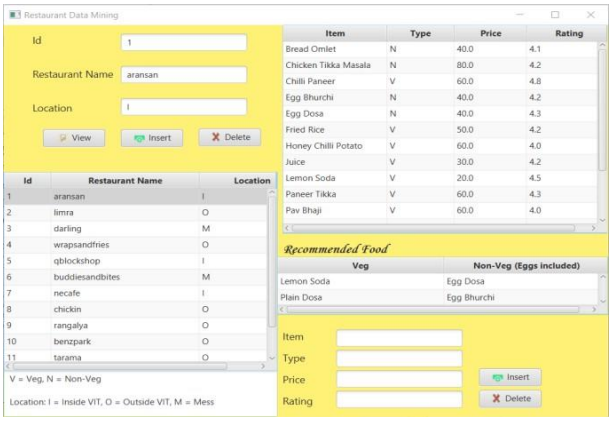
4. Implementation

- As Home page when running the app. Clicking on the

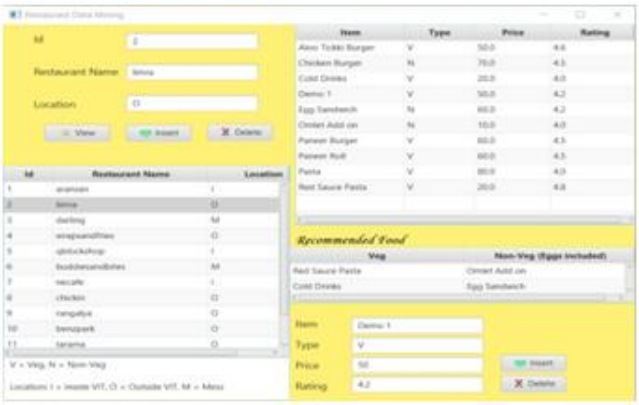
Welcome button will redirect to the next scene.

- Project functionality scene. Clicking on the check it out button will redirect to the main scene of the project.

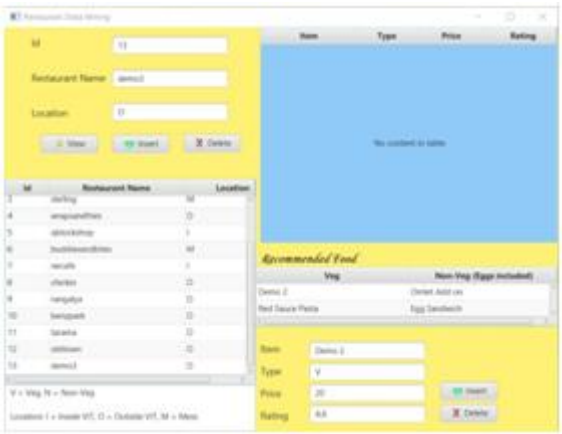
- Main scene of the project. Clicking on the table view row in the restaurants section will fill the text fields automatically. Then clicking the view button will display the menu of the selected restaurant. The recommendations for veg and non - veg food will also appear below the menu section.



- Inserting a new food item in the menu of the selected restaurant.

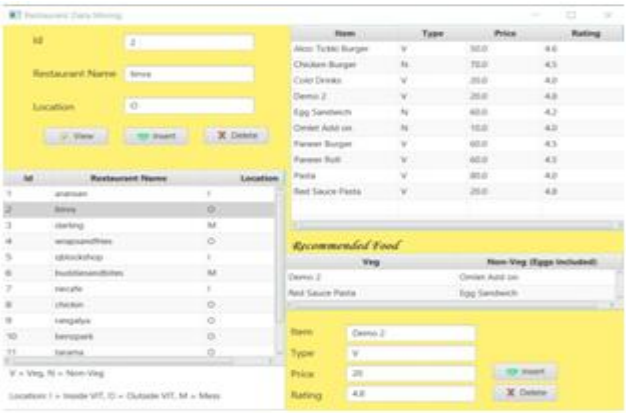


- Showing a real time example of automatic update of the recommendation upon adding a new food item.
- Adding a new restaurant.

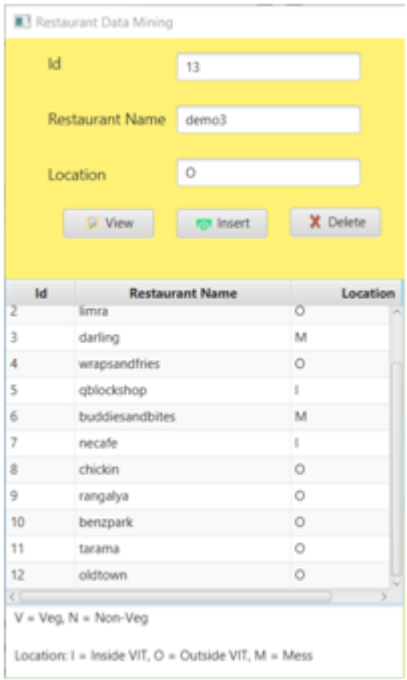


- Creation of food table for the added restaurant in the data - base.

- Adding food items in the newly added restaurant.



- Deleting a restaurant.



Key Code Snippets:

1) Price Adjustment:

```
if (price < 20) {
temp price = price + 20; ratio = rating/temp price;
}else{
ratio = rating/price;
}
```

2) Getting observable lists of restaurants:

```
ObservableList<Restaurants> getRestau - rantsList () {
ObservableList<Restaurants> res - taurantsList = FXCollec -
tions. observableArrayList ();
Connection con = getConnection ();
String query = "SELECT * FROM restaurants";
Statement st; ResultSet rs;
try{
```

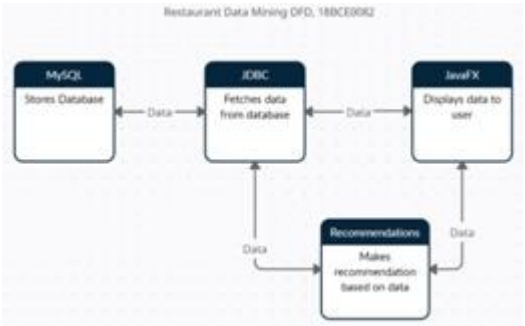
- Deleted the food table of the restaurant from the database to save residual space in the memory.


```
st = con. createStatement (); rs = st. executeQuery (query);
Restaurants restaurants; while (rs. next ()) {
restaurants = new Restau - rants (rs. getInt ("id"), rs.
getString ("restaurantName"), rs. getString ("location"));
restau - rantsList. add (restaurants);
}
}catch (SQLException e) { System. out. println ("Error: "+
e. getMessage ());
}
return restaurantsList; }
```

3) Retreiving Recommendations:

```
public ArrayList<String> sortByValue (boolean order, Map
map, int s)
```

5. Flowchart




```
{
//convert HashMap into List List<Map. Entry<String,
Float>>
list = new LinkedList<Map. Entry<String, Float>> (map.
entrySet ());
//sorting the list elements Collections. sort (list, new Com -
parator<Map. Entry<String, Float>> ()
{
public int com - pare (Map. Entry<String, Float> o1, Map.
Entry<String, Float> o2)
{
```

```
if (order)
{
//compare two object and return an inte - ger
return o1. getValue (). compareTo (o2. getValue ()); }
else
{
return o2. getValue (). compareTo (o1. getValue ());
}
}
});
ArrayList<String> arr = new Ar - rayList<String> ();
Map<String, Float> sortedMap = new
LinkedHashMap<String, Float> ();
for (Map. Entry<String, Float> en - try: list)
{
sorted - Map. put (entry. getKey (),
\entry. getValue ());
arr. add (entry. getKey ());
}
return arr;
}
```

6. Results and Discussion

If you are using Word, use either the Microsoft Equation The HashMaps and array lists used together to give recommendations is an effective use of data structures. Observable list in Java FX's module to allow listeners to track changes when they occur. The items at low price were at an advantage as a low price in division of the ratio calculated to select best food items created more impact than ratings in the numerator. So, to avoid these items from showing up in the recommendations all the time, if the price was less than or equal to Rs 20, 20 was added to calculate the ratio, this made the selection balanced.

To check if the recommendations were updating themselves once a new item is added, a test item with high rating and low price was added so that it would be favored by the algorithm. The time and space complexity were considered while taking care of details for a better and interactive user experience.

7. Conclusion and Future Work

The project worked as intended, all the desired functionalities were achieved. The restaurants were added, and the food items were updated seamlessly. The recommendations were made close to what actually is famous in the listed restaurants in the database. The problem of low - priced beverages recurring was taken care of. The updates in all the three table views were refreshed in real time as any change was made.

With regards to future work, if expanded and worked upon by a team rather than an individual, is very wide as there are about 10k freshers which come to VIT every year and are unaware of the res - taurants inside and outside VIT. They can also use it to select the mess for their hostels. If made on a larger scale, the rest of the population can also be targeted and it can be adapted and used for other campuses and colleges as well.

References

[1] Adam B. Brody and Edward J. Gottsman.1999. POCKET BARGAINFINDER: A Handheld Device for Augmented Commerce. In Proceedings of the International Symposium on Handheld and Ubiquitous Computing (HUC '99). Karlsruhe, Germany.

[2] Mark Claypool, Anuja Gokhale, Tim Miranda, Pavel Murnikov, Dmitry Netes and Matthew Sartin.1999. Combining Content - Based and Collaborative Filters in an Online Newspaper. In Proceedings of the ACM SIGIR Workshop on Recommender Systems, Berkeley, CA, August, 1999.

[3] Anatole V. Gershman, Joseph F. McCarthy and Andrew E. Fano.1999. Situated Computing: Bridging the Gap between Intention and Action. In Proceedings of the Third International Symposium on Wearable Computing (ISWC ’99), San Francisco, CA.

[4] Will Hill, Larry Stead, Mark Rosenstein and George Furnas. Recommending and Evaluating Choices in a Virtual Community of Use.1995. In Proceedings of the 1995 ACM Conference on Human Factors in Computing Systems (CHI ’95), Denver, CO, pp.194 - 201.

[5] Joseph A. Konstan, Bradley N. Miller, David Maltz, Jon Herlocker, Lee R. Gordon and John Riedl. GroupLens: Applying Collaborative Filtering to Usenet News. Communications of the ACM 40, 3 (1997), pp.77 - 87.

[6] Joseph F. McCarthy.2000. Active Environments: Sensing and Responding to Groups of People. In Proceedings of the 2000 Conference on Human Factors in Computer Systems (CHI 2000), Extended Abstracts, The Hague, pp.51 - 53.

[7] Joseph F. McCarthy and Theodore D. Anagnost.1998. MUSICFX: A Group Preference Arbitration System for Computer - Supported Collaborative Workouts. In Proceedings of the ACM 1998 Conference on Computer Supported Cooperative Work (CSCW ’98), Seattle, WA, pp.363 - 372.

[8] Joseph F. McCarthy and Eric S. Meidel.1999. ACTIVEMAP: A Visualization Tool for Location Awareness to Support Informal Interactions. In Proceedings of the First International Symposium on Handheld and

[9] Ubiquitous Computing (HUC ’99), Karlsruhe, Germany. Published in Hans - W. Gellerson (ed), Lecture Notes in Computer Science 1707, Springer, pp.158 - 170.

[10] Raymond J. Mooney and Loriene Roy.1999. Content - Based Book Recommending Using Learning for Text Categorization. In Proceedings of the ACM SIGIR Workshop on Recommender Systems, Berkeley, CA, August, 1999.

[11] Mark O’Connor, Dan Cosley, Joseph A. Konstan and John Riedl.2001. PolyLens: A Recommender for Groups of Users. In Proceedings of the Seventh European Conference on Computer Supported Cooperative Work (ECSCW 2001), Bonn.

[12] M. V. Nagendra Prasad and Joseph F. McCarthy.1999. A Multi - Agent System for Meting out Influence in an Intelligent Environment. In Proceedings of the Eleventh Conference on Innovative Applications of

Artifical Intelligence (IAAI - 99), Orlando, FL, pp.884 - 890.

[13] Upendra Shardanand and Pattie Maes.1995. Social Information Filtering: Algorithms for Automating Word of Mouth. In Proceedings of the 1995 ACM Conference on Human Factors in Computing Systems (CHI '95), Denver, CO, pp.210 - 217.