# SURPLUS FOOD REDISTRIBUTION SYSTEM WITH SMART RECOMMNEDATIONS

## **A PROJECT REPORT**

*Submitted by*

**SHAH MOHD ARHUM [RA2011003010104]**

**SRISHTI SINGH [RA2011003010122]**

*Under the Guidance of*

# Dr. SWORNA KOKILA M.L

Assistant Professor, Department of Computing Technologies

*in partial fulfillment of the requirements for the degree of*

**BACHELOR OF TECHNOLOGY**

# in

**COMPUTER SCIENCE AND ENGINEERING**



**DEPARTMENT OF COMPUTING TECHNOLOGIES**

# COLLEGE OF ENGINEERING AND TECHNOLOGY

# SRM INSTITUTE OF SCIENCE AND TECHNOLOGY KATTANKULATHUR– 603 203

**MAY 2023**



# SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

# KATTANKULATHUR–603 203

## **BONAFIDE CERTIFICATE**

Certified that 18CSP107L project report titled “**SURPLUS FOOD REDISTRIBUTION SYSTEM WITH SMART RECOMMNEDATIONS**” is the bonafide work of **SHAH MOHD ARHUM [RA2011003010104]** and **SRISHTI SINGH [RA2011003010122]** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported here in does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion for this or any other candidate.

### Dr.SWORNA KOKILA ML

### SUPERVISOR

Assistant Professor

Department of Computing Technologies

### Dr. B KANISHA

### PANEL HEAD

### Associate Professor

### Department of Computing Technologies

### Dr. M. PUSHPALATHA

### HEAD OF THE DEPARTMENT

Department of Computing Technologies

 Department of Computing Technologies

# SRM Institute of Science and Technology Own Work Declaration Form

**Degree/Course :**B.Tech in Computer Science and Engineering

S**tudent Names :** Shah Mohd Arhum, Srishti Singh

**Registration Number:** RA2011003010104, RA2011003010122

**Title of Work : Surplus Food Re-distribution system with smart recommendations**

We here by certify that this assessment compiles with the University’s Rules and Regulations relating to Academic misconduct and plagiarism, as listed in the University Website, Regulations, and the Education Committee guidelines.

We confirm that all the work contained in this assessment is our own except where indicated, and that we have met the following conditions:

* + - Clearly references / listed all sources as appropriate
    - Referenced and put in inverted commas all quoted text (from books, web,etc.)
    - Given the sources of all pictures, data etc that are not my own.
    - Not made any use of the report(s) or essay(s) of any other student(s)either past or present
    - Acknowledged in appropriate places any help that I have received from others (e.g fellow students, technicians, statisticians, external sources)
    - Compiled with any other plagiarism criteria specified in the Course hand book / University website

We understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

|  |
| --- |
| **DECLARATION:** |
| I am aware of and understand the University’s policy on Academic misconduct and plagiarism and I certify that this assessment is my / our own work, except where indicated by referring, and that I have followed the good academic practices noted above.  **Student 1 Signature:**  **Student 2 Signature:**  **Date:** |
| If you are working in a group, please write your registration numbers and sign with the date for every student in your group. |

# ACKNOWLEDGEMENT

We express our humble gratitude to **Dr. C. Muthamizhchelvan**, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support.

We extend our sincere thanks to Dean-CET, SRM Institute of Science and Technology, **Dr. T. V. Gopal**, for his invaluable support.

We are incredibly grateful to our Head of the Department, **Dr. M. Pushpalatha,** Professor, Department of Computing Technologies, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

We want to convey our thanks to our Panel Members, **Dr. B. Kanisha** Associate Professor, **Mr.G.Balamurugan**, Assistant Professor and **Mrs Kanmani S** Assistant Professor, Department of Computing Technologies, SRM Institute of Science and Technology, for their inputs during the project reviews and support.

We register our immeasurable thanks to our Faculty Advisor, **Dr. Sworna Kokila M L**, Assistant Professor, Department of Computing Technologies, SRM Institute of Science and Technology, for leading and helping us to complete our course.

Our inexpressible respect and thanks again to our guide, **Dr. Sworna Kokila M L**, Assistant Professor, Department of Computing Technologies, SRM Institute of Science and Technology, for providing us with an opportunity to pursue our project under her mentorship. She provided us with the freedom and support to explore the research topics of our interest. Her passion for solving problems and making a difference in the world has always been inspiring.

We sincerely thank all the staff and students of Computing Technologies Department, School of Computing, S.R.M Institute of Science and Technology, for their help during our project. Finally, we would like to thank our parents, family members, and friends for their unconditional love, constant support and encouragement.

**SHAH MOHD ARHUM [RA2011003010104]**

**SRISHTI SINGH [RA2011003010122]**

**ABSTRACT**

# There is a need for effective help to various Non-Profit Organizations that work tirelessly to help the needy in various ways possible. One such use case is effective re-distribution of surplus foods various places like restaurants, cafes etc. produce. This study aims to address this by analyzing modern tools and ICT to build a system that simplifies search for NGOs looking for perfect volunteers to procure food/grains from. Ignoring the delivery logistics, this application will be a website which is integrated with a Random Forest Classifier model that learns to categorize restaurants based on Ratings, Distance from the user and Quantity of food they offer. This is achieved via training by a synthetic dataset with bias. The model's output is coupled with the website to display results. This study also serves as a starting point from where a more complex system can be built.

# 

# TABLE OF CONTENTS

|  |  |  |  |
| --- | --- | --- | --- |
|  | **ABSTRACT** | | **6** |
|  | **LIST OF TABLES** | | **9** |
|  | **LIST OF FIGURES** | | **10** |
|  | **LIST OF SYMBOLS AND ABBREVIATIONS** | | **11** |
| **1**. | **INTRODUCTION** | | **12** |
|  | 1.1 | General Introduction | 12 |
|  | 1.2 | Goals |  |
|  | 1.3 | APIs | 13 |
|  | 1.4 | How websites/webapps work | 14 |
|  | 1.5 | Classification in Machine Learning | 15 |
|  | 1.6 | Random Forest Classifier | 16 |
|  | 1.7 | Python modules used | 17 |
|  | 1.8 | Web-development tools used | 17 |
| **2** | **LITERATURE SURVEY**  2.1 Limitations and Challenges | | **18**  22 |
| **3** | **SYSTEM ARCHITECTURE AND DESIGN** | | **23** |
|  | 3.1 | Architecture Diagram | 23 |
| **4** | **METHODOLOGY** | | **25** |
|  | 4.1 | Introduction to approach | 25 |
|  | 4.2 | Dataset  4.2.1 Feature Consideration  4.2.2 Bias  4.2.3 Generation  4.2.4 Entropy  4.2.5 Visualization | 26  26  27  28  29  30 |
|  | 4.3 | Training  4.3.1 Hyperparameters | 32  32 |
|  | 4.4 | Integration and Flow of Control  4.4.1 Authentication  4.4.2 Redirection  4.4.3 Backend Algorithm  4.4.4 Parsing  4.4.5 Calling the Model  4.4.6 Output Capture  4.4.7 Response Generation  4.4.8 Front-end Algorithm | 34  34  35  35  36  36  36  37  37 |
| **5** | **RESULTS AND DISCUSSION**  5.1Performance Analysis using Various Metrics  5.1.1 Unit Testing  5.1.2 Integration Testing  5.1.3 Acceptance Testing | | **38**  38  38  39  40 |
| **6** | **CONCLUSION AND FUTURE SCOPE** | | **42** |
|  | 6.1 Conclusion  6.2 Future Scope | | 42  43 |
|  | **REFERENCES**  **APPENDIX A**  **APPENDIX B**  **PLAGIARISM REPORT** | | **44**  **47**    **53**         1. **Appendices** |
|  |  | |  |

# LIST OF TABLES

|  |  |  |
| --- | --- | --- |
| 1.1 | Web Tools…………………………………………………….…………….. | 17 |
| 2.1 | Survey Summary…………………………………………………………… | 19 |
| 4.1 | Dataset…………………..…….……………………………….…………… | 28 |
| 4.2 | Deterministic Generation……………………………………...……………. | 28 |
| 4.3 | Chosen Probability Matrix………………………………………………….. | 29 |
| 4.4 | Sample Output………………………………………………………………. | 34 |
| 4.5 | Common Http Codes………………………………………………………… | 34 |
| 5.1 | Unit Testing………………………………………………………………… | 38 |
| 5.2 | Integration Testing…………………………………………………………. | 39 |
| 5.3 | Acceptance Testing…………………………………………………………. | 40 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| 1.1 | Machine Learning…………………………………………………………………. | 15 |
| 1.2 | Forest Classification………………….................................................................... | 16 |
| 3.1 | Architectural Diagram………………………………………………...…………... | 24 |
| 3.2 | Proposed Dataset Generation Method………......................................................... | 24 |
| 4.1 | Zomato Bangalore Rating Dataset........................................................................... | 26 |
| 4.2 | Relationship Heatmap…………..………………………………………………… | 31 |
| 4.3 | sns pairplot………………………………………………………………………... | 31 |
| 4.4 | Output…………………………………………………………………………….. | 32 |
| 4.5 | Classification Report……………………………………………………………… | 33 |
| 4.6 | Output Trees……………………………………………………………………… | 33 |
| 4.7 | Json response……………………………………………………………………… | 37 |
|  |  |  |

**LIST OF SYMBOLS AND ABBREVIATIONS**

**CSV** Comma Separated Values

**REST** Representational State Transfer

**RFC** Random Forest Classifier

**HTTP** HyperText Transfer Protocol

**SPA** Single Page Applications

**AI** Artificial Intelligence

**ML** Machine Learning

**NGO** Non Profit Organization

**SVM** Support Vector Machine

**JSON** Javascript Object Notation

**CHAPTER 1**

**INTRODUCTION**

**1.1 General Introduction**

Nowadays every industry in the world is looking for opportunities to automate and improve workflow by incorporating ICT into their work. This usually means whenever necessary, there is a need to identify where the requirement is, what tools can be leveraged, their utilization to craft a tool and finally a proper system to measure whether investing in the production of such a tool is useful or not.

In the context of a food distribution service, it becomes apparent that there can be several AI/ML algorithms that can help ease our way into serving customers with whatever they desire. This has been done already by several big companies that offer apps for the masses to order food. Large amounts of data, therefore, is collected regarding search patterns, preferences and feedback that using this data we can develop a system that offer recommendations to better serve the customer. To leverage this, we have to think of a scenario where data analysis can be done and useful output can be taken out of it. A good scenario that comes to mind of NGOs.

There can be a scenario where an NGO that is in charge of providing food or raw grains to needy parts of the society is looking for a participant which can fulfil this requirement. Keeping logistics aside, imagine they are searching for anyone that can provide with anything, no matter how far they are (within a logical limit, say same state). Out of n different choices, they have to decide on one. How do they decide? Surely going through the details of every choice is not viable. The use of random classifier model is good in this context because of the nature of the problem. The details of how it works and how it was utilized here will be explained later.

**1.2 Objectives**

**Objective 1: Website**

Make a website that has the capability of registering NGOs and participants like restaurants along with their details like phone no, email and address. The website should also offer secure authentication and login.

**Objective 2: Storage**

Gather that information and store that in a relational manner so that it can be retrieved later.

**Objective 3: Recommendations**

From the information stored, develop a system that can take a look at factors like distance, ratings, quantity of food/type and gives a recommendation**.**

**Objective 4: Dynamic natured integration**

The recommendations must make the website dynamically show the necessary details of the concerned choice.

**Objective 5: Adaptability**

The recommender should be made in way that when trends change, it should be able to identify and adapt to it, while keeping the rest of the application intact.

**1.3 APIs**

API stands for Application Programming Interface. It is a set of rules and specifications that define how software components should interact with each other. APIs are used to allow different software programs to communicate with each other and share data.

APIs can be used for a variety of purposes, including:

• **Accessing data:** APIs can be used to access data from third-party services. For example, a weather app can use an API to get the latest weather data from a weather service.

­­­

• **Performing actions:** APIs can be used to perform actions on third-party services. For example, a social media app can use an API to post a message on your behalf.

• **Integrating different services:** APIs can be used to integrate different services together. For example, an e-commerce website can use an API to integrate with a payment processing service.

**1.4 How modern webapps work**

Webapps have backends, frontends that communicate with one another and helps one to interact with the user to gather data. Now, when it comes to exchanging data, there are a few common techniques:

• **HTTP Requests:** Websites communicate through the Hypertext Transfer Protocol (HTTP). When you click a link or submit a form, your browser sends an HTTP request to the web server. The server processes the request and sends back an HTTP response, which contains the data needed to update the web page.

• **APIs:** APIs are essential for data exchange. Many websites expose APIs that allow other services to request and receive specific data. For instance, a weather website might offer an API that provides weather data to other applications.

Essentially, the requirement is an API that in turn calls a classifier model to get preference values, forms a response that depends on the preference values, get additional details regarding the preference values then send it all to the front-end as a JSON file.

**1.5 Classification in Machine Learning**

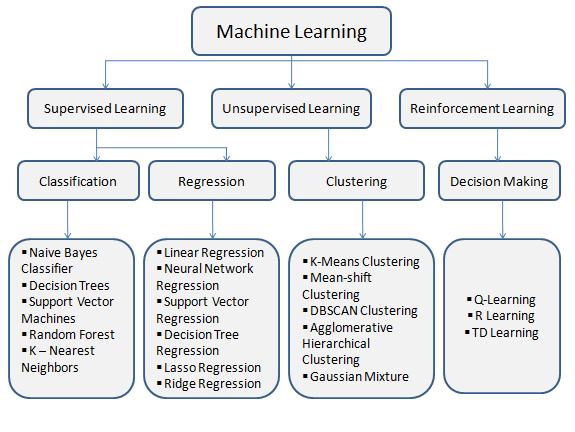
Machine learning is like teaching computers to learn from data and make predictions or decisions without being explicitly programmed. It's a subfield of artificial intelligence that involves algorithms and statistical models to recognize patterns, learn from them, and improve their performance over time. It's widely used in various applications, from image and speech recognition to recommendation systems, and it's all about training computers to get smarter through data and experience.

Classification is a fundamental task in supervised machine learning. It's like teaching a computer to categorize or label things based on their characteristics. The primary goal is to train a model to make predictions or decisions without being explicitly programmed. Here's how it works:

* **Data Preparation**: Start with labeled data, where each example has input features and a corresponding class label.
* **Model Training:** Choose an algorithm and train the model using the labeled data. The model learns to make predictions based on patterns in the data.
* **Model Evaluation:** Test the model's performance on a separate dataset to see how well it can categorize new, unseen data.
* **Prediction:** Once the model is trained and performs well, it can classify new data.

As someone can expect there can be multiple ways to classify: Binary, and Multi-class classification. For classification, there are algorithms like **SVM, Naïve Bayes, Random Forest**, etc.

Random Forest is a good choice for us because of its simplicity, traceability and relatively low computational requirements.



**Fig 1.1 Machine Learning**

**1.6 Random Forest Classifier**

Random Forest is a versatile and powerful ensemble learning method for classification and regression tasks in machine learning. It's called a "forest" because it's composed of many decision trees, and it's "random" because it introduces randomness in multiple ways to improve performance and reduce overfitting.

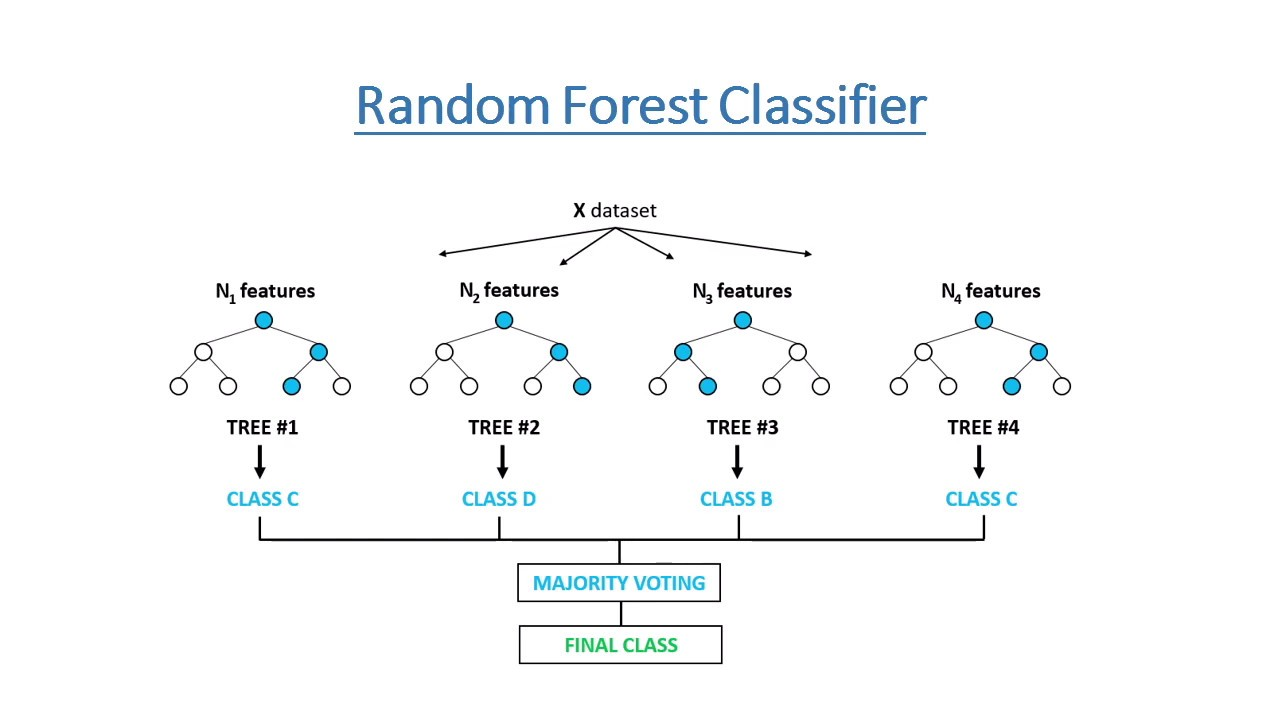
RFC can have the following features:

• It combines many decision trees.

• Each tree is trained on a random subset of the data and features.

• Bagging reduces overfitting, and random feature selection adds diversity.

• The final prediction is based on a majority vote from all the trees.



**Fig 1.2 Forest Classifications**

**1.7 Python Modules**

In the implementation of the model, the following Python modules were used to facilitate various aspects of the project. These modules are integral to the functionality and performance of the solution.

* **scikit-learn (sklearn)**

scikit-learn, a well-established and widely recognized machine learning library, was used as a cornerstone of the implementation. This library offers a comprehensive suite of tools for machine learning tasks, encompassing

classification, regression, clustering, and more. It is notable for providing an extensive array of algorithms and tools for data preprocessing and model evaluation.

* **Matplotlib**

For the creation of data visualizations, Matplotlib was relied upon. This versatile library is highly regarded for its capability to generate static, animated, or interactive visualizations. It is a primary choice for data plotting and charting, enabling meticulous control over plot details and customization.

* **Pandas**

Pandas played a pivotal role in the project by serving as a robust data manipulation and analysis library. It introduces essential data structures such as DataFrames, which were used for handling tabular data. Additionally, Pandas equips the project with an extensive set of tools for data cleaning, transformation, and exploratory data analysis.

* **Seaborn**

Complementing Matplotlib, Seaborn specializes in the realm of statistical data visualization. This library simplified the creation of aesthetically pleasing and informative statistical graphics. It offers high-level functions tailored to common data visualization tasks, which enhanced the visual representation of the project's findings.

**1.8 Web development tools**

|  |  |  |
| --- | --- | --- |
| Tool | Advantage | Disadvantage |
| Django [backend] | Built-in authentication, written in python | None |
| ReactJS[frontend] | SPA oriented, fast | Learning Curve |
| MySQL | Relational storage, integration w/ Django | None |

**Table 1.1 Web Tool**

**CHAPTER 2**

**LITERATURE SURVEY**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.No** | **Title** | **Author** | **Journal name** | **Year** | **Brief Description** |
| 1. | Scaling relational geographies of ICT-  mediated surplus food redistribution | Marion Weymes and Anna R. Davies | Journal of  Sharecity | 2018 | Discussed the role of Information and Communications Technology (ICT)in improving food systems, particularly in preventing food waste and redistributing surplus food. |
| 2. | Sustainability Assessment of Food Redistribution  Initiatives in Sweden | Pauline Bergström, Christopher Malefors, Ingrid Strid , Ole Jørgen Hanssen and Mattias Eriksson |  | 2020 | Investigates surplus food redistribution units in Sweden, which connect surplus food from retailers and the food industry with people in need, aiming to both prevent food waste and provide food to vulnerable populations. |
| 3. | Analysis and Prediction of Food Donation Behavior for a Domestic  Hunger Relief Organization | Lauren B. Davis\*1, Steven X. Jiang1, Shona Morgan2, Isaac A. Nuamah1, Jessica R. Terry1 | [International Journal of Production Economics](https://www.sciencedirect.com/journal/international-journal-of-production-economics) | 2016 | Discussed about how non-profit hunger relief organizations rely on donations to provide food and services to those in need and also develops predictive models to estimate donation quantities and explores the impact of forecast accuracy on donation behaviour |
| 4. | Machine learning application for sustainable agri-food supply  chain performance | Santoso1\*, M Purnomo2, A A Sulianto1 and A Choirun1 | Journal of IOP Conference Series | 2021 | Provides a systematic review of machine learning (ML) applications within the agri-food supply chain and emphasizes the role of ML algorithms in offering real-time analytical insights to support data-driven decision-making and to enhance productivity and sustainability in the agri-food industry. |

**Table 2.1 Survey summary**

“Scaling relational geographies of ICT-mediated surplus food redistribution” proposed by Marion Weymes and Anna R. Davies (Trinity College Dublin, Ireland) discusses the role of Information and Communications Technology (ICT) in improving food systems, particularly in preventing food waste and redistributing surplus food. Several Products (spanning 100 cities) were studied, one of them wasFoodCloud. While ICT is crucial for their success, the research finds that it's not enough on its own to address food waste and insecurity; broader systemic changes are needed. Surplus food redistribution, alternatively referred to as food rescue (Reynolds et al., 2015) or food recovery (Garrone et al., 2014), generally involves the collection of edible food that would otherwise be discarded and its subsequent relocation to individuals, organisations or communities. Through practices such as gleaning and charitable giving, such systems of redistribution have a long, if territorially diverse, lineage (Edwards and Mercer, 2007). However, the increasing accessibility of ICT - particularly through mobile digital media - has been identified as means to transform the ways in which the redistribution of surplus food takes place and to 4 extend its impacts (Lipinski et al. 2013; Ciaghi and Adolfo Villafiorita, 2016; Farr-Wharton et al. 2014; Corbo et al., 2015). Indeed, claims have been made that ICT has the potential to disrupt systemic food waste and empower communities to reduce waste and enhance food security.

“Sustainability Assessment of Food Redistribution Initiatives in Sweden” proposed by Pauline Bergström, Christopher Malefors, Ingrid Strid , Ole Jørgen Hanssen and Mattias Eriksson (Department of Energy and Technology, Swedish University of Agricultural Sciences) highlight that distributing food bags to socially vulnerable individuals leads to the most significant reduction in greenhouse gas emissions per kilogram of redistributed food. Reprocessing surplus food into high-quality products is socially valuable due to job creation, but most scenarios showed monthly financial losses, necessitating additional financial support for sustainability. Awareness of food waste is growing globally, and is regarded as such an important topic, that it is part of the United Nations (UN) Sustainable Development Goals (SDGs), where the aim is to: “By 2030, halve per capita global food waste at the retail and consumer levels and reduce food losses along production and supply chains, including post-harvest losses” [1]. The European Union (EU) has committed to implementing the SDGs, and is also working on reducing waste in general via a Waste Framework Directive 2008/98/EC (WFD), with a waste management hierarchy that ranks the prevention of waste, and is also preparing for re-use as its first and second priority, respectively [2]. However, the wastage of edible food is not just a waste management issue, but also raises critical equity questions, especially considering the large number of people globally who are living below the poverty line [3].

“Analysis and Prediction of Food Donation Behavior for a Domestic Hunger Relief Organization” by Lauren B. Davis\*1, Steven X. Jiang1, Shona Morgan2, Isaac A. Nuamah1, Jessica R. Terry1 (North Carolina Agricultural & Technical State University)aims to analyze a specific food bank supply chain with a particular emphasis on literature examine food bank usage and the challenges associated with limited and unpredictable supply (Campbell et al., 2011, Paynter et al., 2011, Tarasuk and Eakin, 2003), nutritional related initiatives (Handforth et al., 2013, Hoisington et al., 2011), and the impact of corporate donations on food bank operations (Tarasuk and Eakin, 2005). However, to the best of our knowledge, the application of statistical analysis techniques to manage supply uncertainty has not been addressed. We fill this gap by explicitly studying the nature of in-kind supply uncertainty and demonstrating that time-series forecasting techniques can provide reasonable estimates for in-kind donations. Our results generate forecast accuracy within 10% for specific instances. Our study has particular merit because it is important for non-profit organizations to leverage knowledge and technology to renew and reinvent their operational effectiveness. Food banks armed with better predictive information on supply donation behavior can refine in-kind donation strategies and make informed downstream distribution decisions, which in combination increases the potential of the supply chain to meet organizational objectives.

“Machine learning application for sustainable agri-food supply chain performance: a review” proposed by Santoso1\*, M Purnomo2, A A Sulianto1 and A Choirun1 (Universitas Brawijaya, Malang Indonesia). In this agri-food supply chain encompasses activities from farming to consumption, including agriculture, production, packaging, distribution, and marketing. Data analytics are crucial for ensuring food security, safety, and ecological sustainability. Emerging technologies like the internet of things, machine learning, and cloud computing have the potential to revolutionize how agri-food production is managed. Artificial intelligence (AI) methods are used to address these big data-related challenges. Machine learning, a subset of AI, is widely used to identify hidden patterns in the data [3,4]. Machine learning (ML) algorithms can detect data whose data patterns are unknown and direct researchers to achieve the expected goals. The application of the ML Algorithm in the supply chain has attracted the interest of researchers. So far, researchers have analyzed and interpreted big data using traditional methods.

Artificial intelligence (AI) methods are used to address these big data-related challenges. Machine

learning, a subset of AI, is widely used to identify hidden patterns in the data [3,4]. Machine learning

(ML) algorithms can detect data whose data patterns are unknown and direct researchers to achieve the

expected goals. The application of the ML Algorithm in the supply chain has attracted the interest o

* 1. **Limitations and Challenges**

The paper also mentions that only ICT is not going to solve the problem this means maintaining relationships between the donors and recipients is very essential apart from that it really doesn't mention any technical solution yet, which we aim to address in the project. Systemic Restructuring Needed: The literature emphasizes that achieving meaningful change in surplus food redistribution requires systemic restructuring within agri-food systems. This includes addressing underlying issues such as food production, distribution, and access.

The study primarily focuses on surplus food redistribution units in Sweden. Therefore, the findings and insights may not be directly applicable to different geographic regions or countries, limiting the generalizability of the results. Many surplus food redistribution efforts focus on short-term relief rather than addressing the root causes of food insecurity and food waste. Doesn’t propose any type of modern technologies like AI or Machine Learning.

Although forecasted models are shown, it is being related to external poverty data, beyond the scope of this project. This paper also has a section about distribution practices and stuff which is irrelevant to us, rendering parts of this paper irrelevant.

The paper once again isn’t specific to our application but rather gives an insight on the use of ML in the logistics of the application. Therefore, the degree of advantages ML can or cannot provide is still relatively unknown.The paper explores the possibilities of several algorithms but not implementation

**CHAPTER 3**

**SYSTEM ARCHITECTURE AND DESIGN**

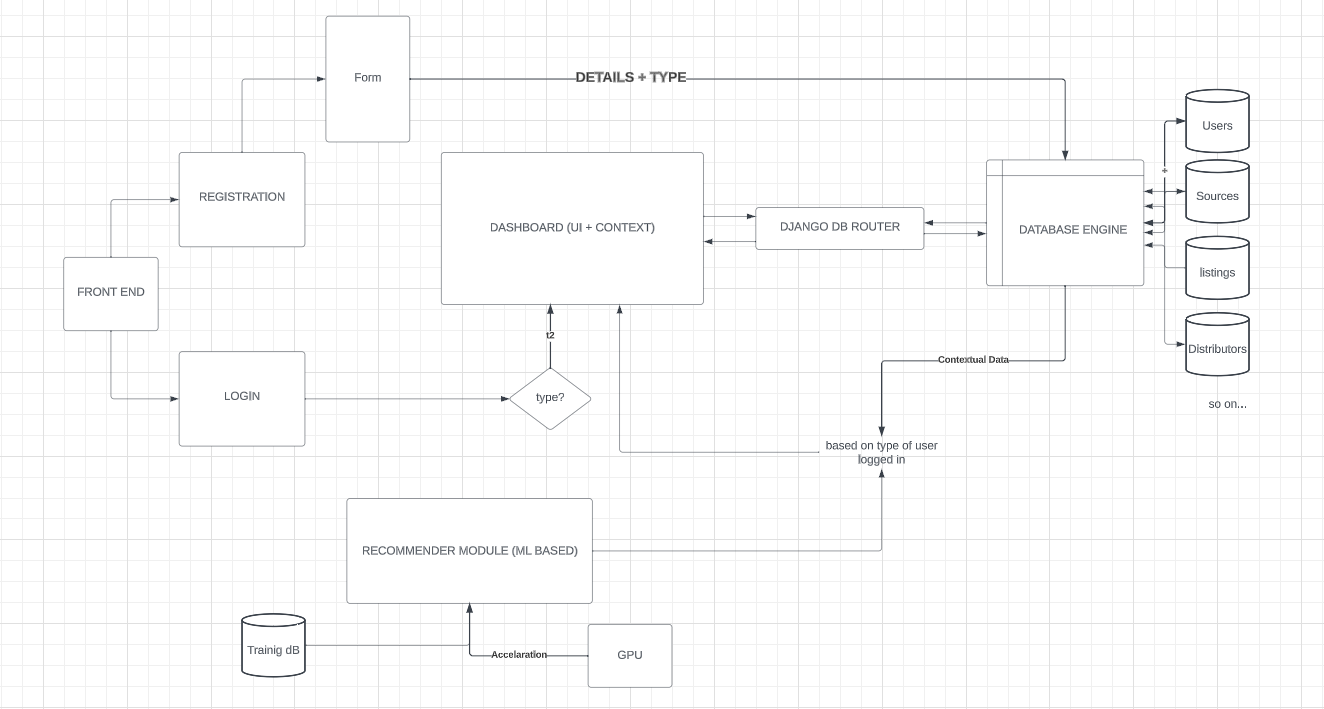
**3.1 Proposed System Architecture**

A system that recommends best possible restaurants on the basis of values of ratings, quantities and distance that is essentially a website consists of several components working together. These include REACT based dashboard that is hidden behind an authentication layer which is connected separately to the database. Users’ login via the in-built Django template engine that allows for making dynamic webpage content. After that, the system redirects to the REACT app upon successful authentication. The REACT app then exchanges data with the server to display the content. The authentication is handled by a form element that sends a POST request to the server, where a specific function in the backend gets triggered to authenticate and create session IDs

This was the main overview of the website, where the recommender module is a separate module of the system. The recommender module firstly trains itself to categorize restaurants. After categorization, the backend will ask for categorization of the restaurants in the user’s search. This search can be related to location. The recommender module is a random forest classifier which is good for categorizing the given data into 5 distinct values.

For Example, a user (an NGO) is looking for volunteers in the entire state of Chennai. For n choices, there will be a set of data points in the format [Ratings, Distance, Quantity]. For every such datapoint, the model will return the category, which will be helpful in sorting the restaurants. All of this data will be passed on as the contextual data REACT can use to dynamically render components, which will be a card in this case.

REACT router dom can also be used to add in routes within the app, to allow for more features. One such feature is added in this project is to add a listing. This creates an entry in the database corresponding to the user who created it.



**Fig 3.1 Architectural Diagram**

**Fig 3.2 Proposed Dataset Generation method**

**CHAPTER 4**

**METHODOLOGY**

**4.1 Introduction to approach**

The first stage of the application is to make the backend APIs that handle constant things like Authentication and User creation. For this, the database is also set up with the connection details intact. The models are created which correspond to the fields in the database and contains things like Address, Phoenno, Email, etc. Templates corresponding to these fields are then created using the <form> in HTML. This form sends a POST request that allows Django to store the information received.

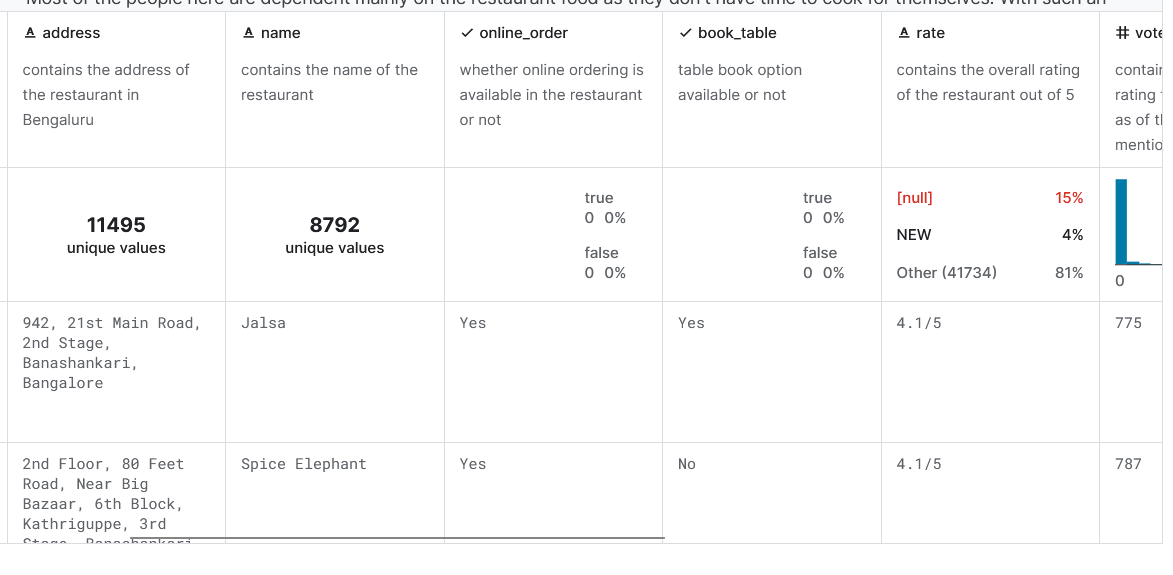
A similar form is then created for logging in, which sends a POST request to another view in Django that authenticates and logs user in. The user is then redirected to the REACT app, which is the second stage.

The REACT app has several components like sidebar, cards, etc. that are all coded separately, and are joined together in a flexbox configuration. Here the advantage of REACT really shines, because the system wants to render 5 cards displaying different data, it isn’t necessary to add 5 card components manually, the server can return information in a specific format for which the frontend can render component based on the number of entries in the response.

The third stage is the recommender system. For the recommender system, the dataset has to be set in a such a way that the target value which is category, has some dependency on the independent factors, ratings, etc. For this the dataset will be generated with a bias whose extent will be analyzed using a heatmap. After the bias is induced, the model can then be trained and it will also start to prefer values corresponding to the bias. The top few results will be returned whose data will be combined with the information retrieved from the DB. This information can then be organized and sent to the REACT app for rendering. Hence the users see the recommendations in action.

**4.2 Dataset**

Several open-source datasets were studied, and one almost conformed to the requirements here but they had no bias. One such approach could have been to edit the dataset to induce bias but it was too big to do that



**Fig 4.1 Zomato Banglalore Rating Dataset**

So, it was more practical to generate a relatively smaller dataset with intended properties using pandas.

**4.2.1 Feature Consideration**

### In a choice of n restaurants, an NGO has to make a choice, and that choice we have to “recommend” to the user.

### An NGO may look at several properties as to help them decide on restaurant. These are (and are not limited to):

### Quantity of food/grains/groceries

### What other users think of a particular source, also known as rating

### Their physical distance

### Miscellaneous factors such as Hygiene or food standards, cooking standards, political agenda etc.

### This list is not exhaustive and quite frankly limiting this list could very well be against the very goal for these NGOs but for the sake of simplicity, we can say that the most important factors that might affect the choice (or preference) is ratings, quantity & distance. This means that the choice will be made based on some combination of the three.

### Therefore, we can limit our recommendation system to look for these values and base its choice on these.

### Apart from these, there also should be a way to identify them which can be achieved by utilizing IDs. These IDs can be referenced from the database to gather other relevant details about the restaurant.

**4.2.2 Bias**

On a higher level, presence of 3 factors lead to 3 conditions:

### CONDITION A: the choice of users tends to be dependent more on ratings

### CONDITION B: the choice of users tends to be dependent more on quantity

### CONDITION C: the choice of users tends to be dependent more on distance

### Using libraries like NumPy, it is possible to induce these conditions onto the dataset to test out whether or not it learns to prefer these factors. The triplet factors, (ratings, quantity, distance) will be used to categorize them into 5 categories, 5 being the best. In the training dataset, the category will be given.

In summary, the dataset format is given below:

|  |  |  |  |
| --- | --- | --- | --- |
| **Ratings** | **Quantity** | **Distance** | **category** |
| 2 | 19 | 14 | 3 |
| 1 | 7 | 20 | 1 |

### 

**Table 4.1 Datset format**

### Where,

### 

### 

### 

### These ranges are arbitrary choices. In a real situation, all values can be rounded off to match the dataset.

**4.2.3 Generation**

The method where the data values are generated is straightforward. Among a list of possibilities, pick a random value. Optionally, the function can also take in a probability list that corresponds to the list’s index

For example, consider:

category = np.random.choice([1, 2, 3, 4, 5], p=[0.7, 0.1, 0.1, 0.05, 0.05])

it follows the following probability distribution:

|  |  |  |
| --- | --- | --- |
| **choice** | **P** | **%age** |
| **1** | 0.7 | 70 |
| **2** | 0.1 | 10 |
| **3** | 0.1 | 10 |
| **4** | 0.05 | 5 |
| **5** | 0.05 | 5 |

**Table 4.2: Deterministic generation**

In this way, to achieve a bias towards ratings, a probability matrix has to be assigned to each possibility of rating. The target variable would be category, so ratings have 5 possibilities, along with 5 possibilities for category. After a bit of trail and error, the optimal probability values were assigned This is summarized below:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Rating=>****Choice** | **1** | **2** | **3** | **4** | **5** |
| **1** | 0.7 | 0.05 | 0.05 | 0.05 | 0.025 |
| **2** | 0.1 | 0.35 | 0.1 | 0.15 | 0.025 |
| **3** | 0.1 | 0.25 | 0.4 | 0.2 | 0.05 |
| **4** | 0.05 | 0.15 | 0.2 | 0.3 | 0.05 |
| **5** | 0.05 | 0.2 | 0.25 | 0.3 | 0.85 |

**Table 4.3: chosen probability matrix**

### This algorithm therefore:

### Generates the dataset according to the specifications

### Induces condition A bias.

### Saves in a standard csv format for models to use

**4.2.4 Entropy**

### This dataset has a fundamental problem. Although the dataset generated is of the required format and with the biasing needed, this won’t be a good dataset for testing because of the fact that values are randomly generated. For example, consider the tuple (1,76,2). For this, the model will learn to give it the least category (1) but in the dataset there can be several outlier values like this which the category is another value like 2,3,4 or even 5 because of the small amount of non-zero probability of happening so.

The solution for this can be tricky but for simplicity, one has to avoid this dataset for testing and can resort to manual testing for logical solutions

**4.2.5 Visualization**

### Now there are several ways to visualize our dataset but perhaps the best one would be to use heat maps. This is because we can visually confirm that the target variable has some sort of dependency on (ratings in this case) which is pattern-based (not numerical). We can achieve this using co-relation matrix for the dataset and then plotting that onto a heatmap.

### After loading the dataset we run the following code:

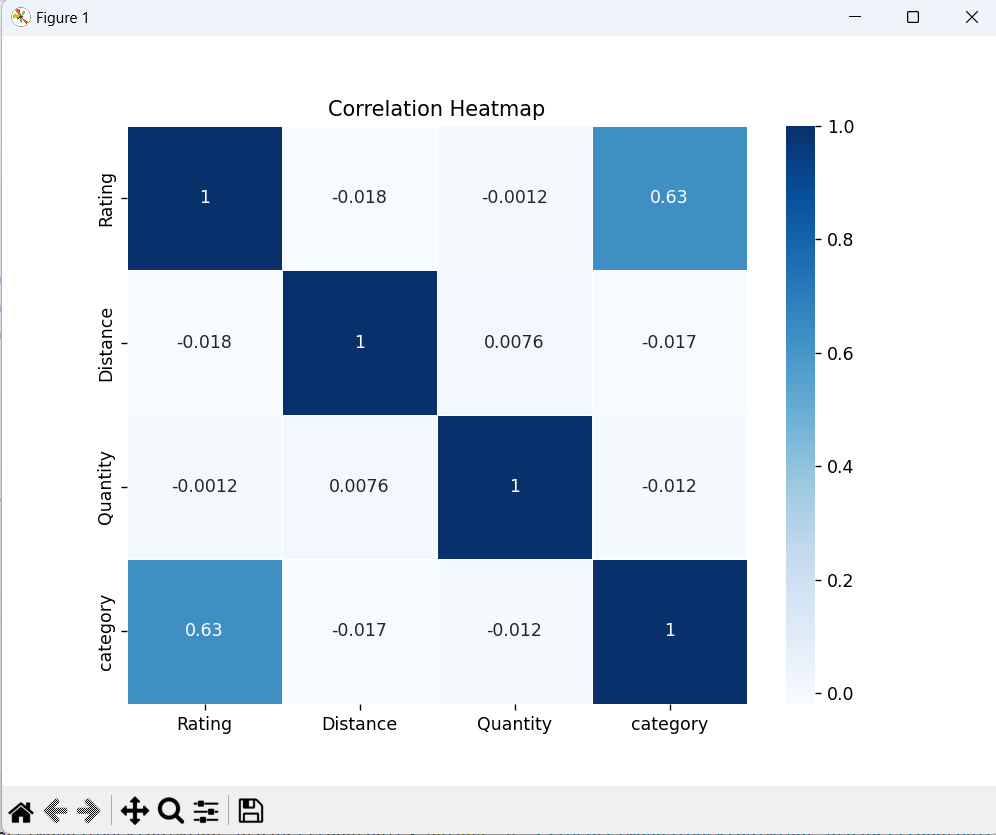
corr\_matrix = dataset.corr()

plt.figure(figsize=(8, 6))

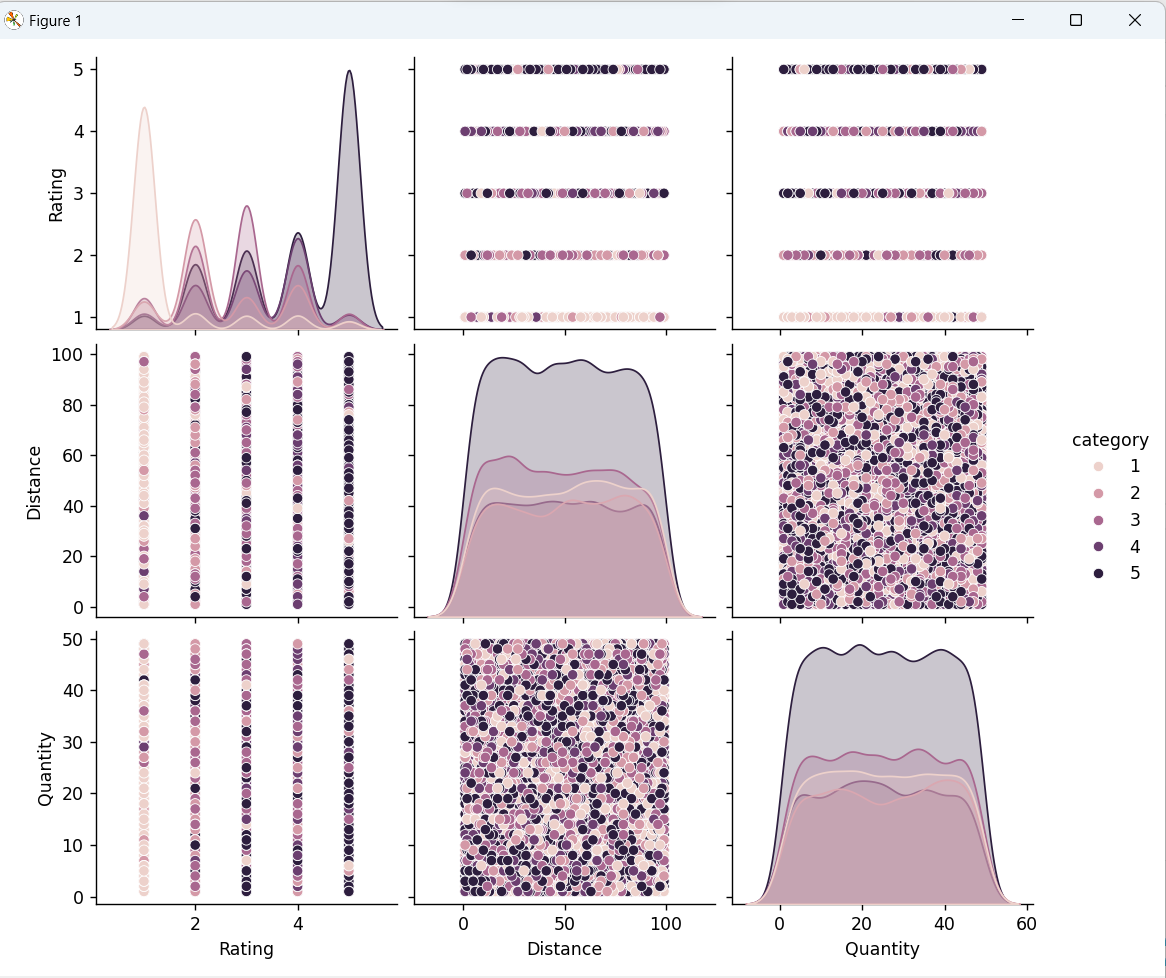
sns.heatmap(corr\_matrix, annot=True, cmap='Blues', linewidths=0.5)

### corr\_matrix = dataset.corr(): This line computes the correlation matrix for your dataset. The correlation matrix shows how each variable in your dataset is related to every other variable. It's a square matrix, and the values in it range from -1 to 1. A high positive value (close to 1) indicates a strong positive relationship, while a high negative value (close to -1) suggests a strong negative relationship.

### sns.heatmap(corr\_matrix, annot=True, cmap='Blues', linewidths=0.5): sns.heatmap is a function from the seaborn library that visualizes the correlation matrix as a heatmap. corr\_matrix is the correlation matrix we computed earlier. annot=True adds the actual numerical values on each cell of the heatmap. So, you can see the correlation coefficients. rest is theming. (ratings vs cat is 0.63)



**Fig 4.2: Relationship heatmap**



**Fig 4.3: sns pairplot**

**4.3 Training**

The code begins by importing the necessary libraries and loading a dataset from a CSV file. This dataset contains features and a target variable. The data is then divided into training and testing sets to evaluate the model's performance effectively. A Random Forest Classifier is created and trained on the training data. After training, the model makes predictions on the test set. The code calculates various evaluation metrics, such as accuracy, precision, recall, and F1-score, and displays a confusion matrix and classification report to assess the model's classification performance. Additionally, it visualizes individual trees in the Random Forest for a deeper understanding of their contributions. Lastly, a sample input is provided to the trained model for prediction.

It's important to note that hyperparameters, like the number of trees and tree depth, play a crucial role in the model's performance, and they can be fine-tuned to achieve better results. This code serves as a practical example of building and evaluating a Random Forest Classifier for a classification task and can be adapted to various real-world applications where you need to make predictions based on data.

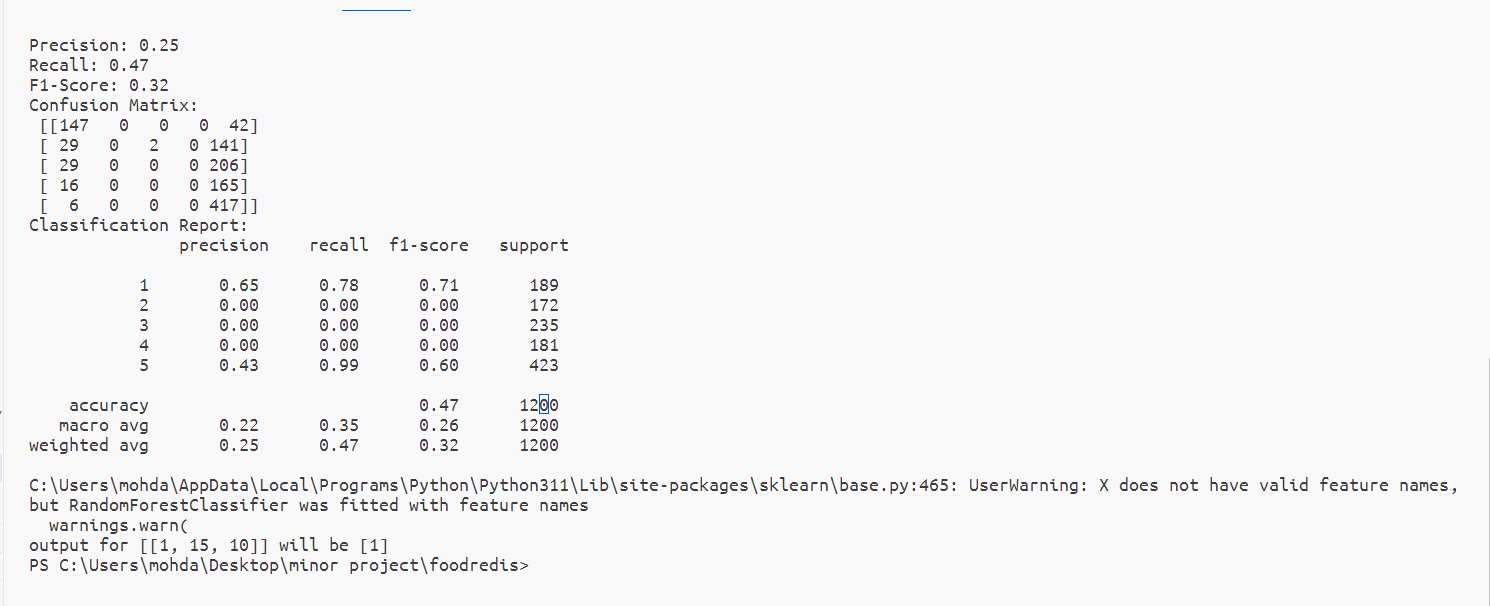
**4.3.1 Hyperparameters**

The chosen hyperparameters are as follows:

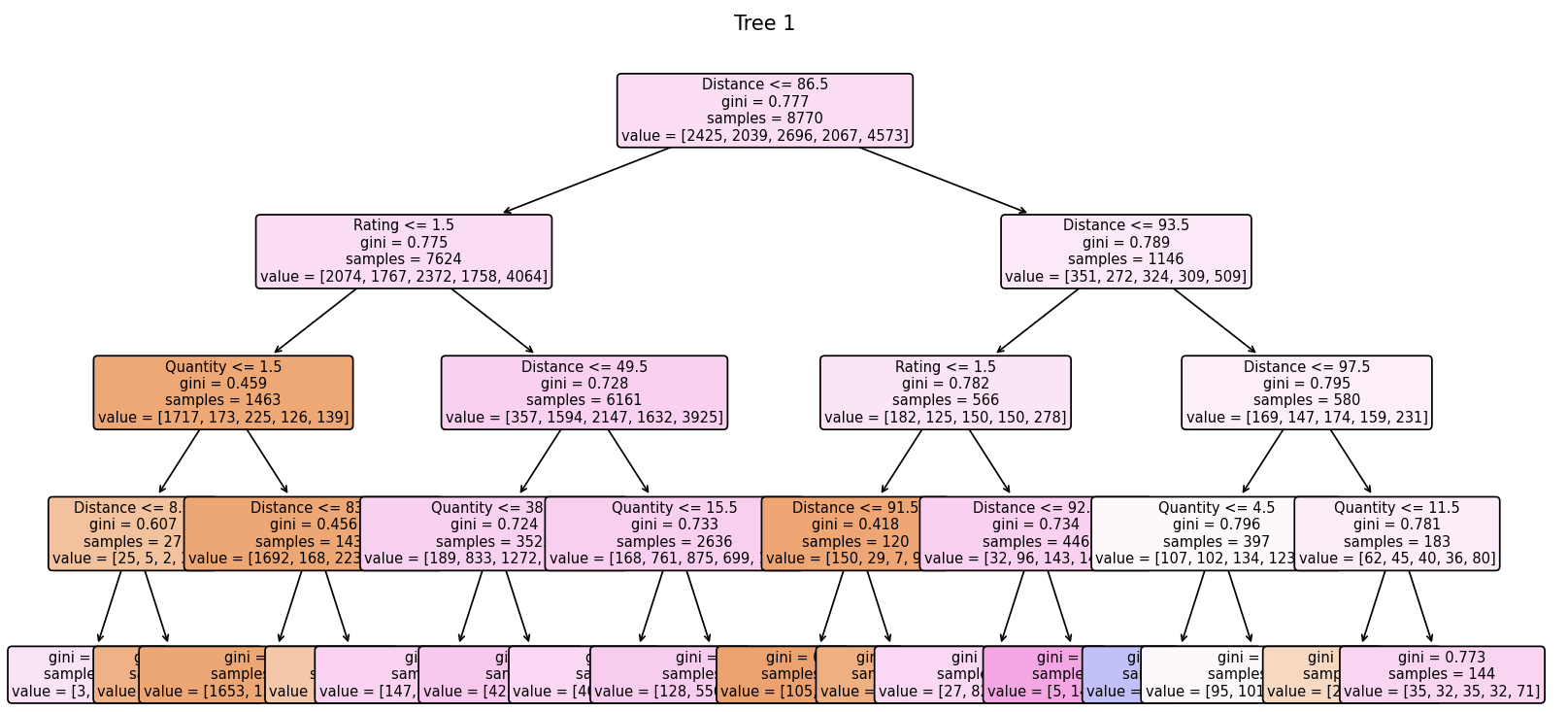
* **n\_estimators = 40:** This hyperparameter sets the number of decision trees in the Random Forest ensemble to 40
* **max\_depth = 5:** The max\_depth hyperparameter limits the maximum depth of each tree in the forest to 5 levels.
* **random\_state = 1:** The Random Forest's internal random processes are reproducible. It means that tunning the code with the same random\_state, will yeild the same results each time.



**Fig 4.4: output**



**Fig 4.5: classification report**



**Fig 4.6: output trees**

### Here is a summary of some sample outputs given by the model:

|  |  |  |  |
| --- | --- | --- | --- |
| **Rating** | **Distance** | **Quantity** | **Category (target)** |
| **2** | 91 | 1 | 5 |
| **3** | 15 | 6 | 5 |
| **1** | 1 | 6 | 1 |
| **1** | 15 | 10 | 1 |

### Table 4.4 Sample Output

### Here we can observe that the classifications are very heavily biased towards ratings, which was our intention. Obviously, there will be outliers as well.

**4.4 Integration and Flow of control**

### At this stage it is also important to point out how exactly the system will integrate with the website.

### Here is a summary of HTTP error codes and their meaning

|  |  |
| --- | --- |
| **Code** | **Meaning** |
| **200** | The request was successful, and the server has responded with the requested data or action. |
| **403** | The server understood the request, but it refuses to fulfill it due to insufficient permissions or authentication. |
| **404** | The requested resource or page was not found on the server, indicating that the URL or path is incorrect. |
| **500** | The server encountered an unexpected condition, making it unable to fulfill the request, typically due to a server-side issue. |

### Table 4.5 Common HTTP codes

### 4.4.1 Authentication

### The basic idea is

### Login in via the Django template-based system @localhost:8000

### the login() function is called where it takes username/password as arguments

### Error messages can be displayed based on the HTTP code.

### 4.4.2 Redirection

### On successful authentication, redirect to the react app which is running on localhost:3000

### This redirection is achieved using the following code snippet:

try {

        const r = await fetch('http://localhost:8000/loginauth', {

            method: 'POST',

            body: formData,

        });

        if (r.ok) {

            window.location.href = 'http://localhost:3000/';

### Error handling was also added here.

### 4.4.3 Backend Algorithm

### 

### This algorithm outlines the process of handling incoming requests for restaurant recommendations. First, it begins by parsing the request and extracting the "location" information from the request's JSON payload. Next, it establishes a connection to the database and retrieves information from the "source\_info" table where the location matches the one provided. It then invokes the Machine Learning (ML) module to obtain ratings, distance, and quantity information, which is sent as input. The ML module returns a category for each restaurant based on this input.

### Afterward, the algorithm calculates a score for each restaurant within their respective categories. These scores are used to sort the restaurants, ensuring that the most relevant ones appear at the top. The sorted restaurant data is then serialized into a JSON response format. Finally, the algorithm returns this response to the client or handles any errors if they occur during the process. It's a step-by-step guide for generating restaurant recommendations based on location and ML-driven categories.

### 4.4.4 Parsing

### Each Django view(api) has a parameter: request that it accepts.

if request.method == 'POST':

### for POST request, do something. Etc.

### We fetch the objects (outputs from the database) using:

source\_data = SourceInfo.objects.filter(loc=request.location).prefetch\_related('listings').values('name', 'address', 'region', 'phoneno', 'email', 'ratings',

'listings\_\_quantity', 'listings\_\_food\_type', 'listings\_\_description'

                )

### 4.4.5 Calling the Model

### Doing this would just involve importing the py file for the model and then calling the classifier from the views file itself.

### For every input, it will return an integer (1,5). This also assumes that the dataset is already present in the correct locations Store all responses in a single list.

### 4.4.6 Output Capture

### At this stage we have a ‘category’ value for each rest\_id. We start by sorting by category. Using this we can query the database for details for restaurants with rest\_id of say, top 5 restaurants. This introduces an issue. Say for [4,55,12] and [5,12,14] the model return the category 5. How do we determine the order? We use the formula:

### 

### Where i,j,k are balancing weights and category is the output the model gave us.

### Preferences can be sorted by score now

### 4.4.7 Response Generation

### 

**Fig 4.7 json response**

This is the JSON response the server returns. The order in which it appears is what the website renders. This means if we calculate score and sort by it and combine it with the data we get from the database we can form a response that has this format.

### 4.4.8 Front-end Algorithm

The front-end follows a simple algorithm:

* + - Wait for response from the API
    - Parse the JSON received
    - Based on the order of JSON, set the props of the card
    - The cards are rendered with the information it received.
    - Wait for user-input

const Card = ({ data }) => {// content is received by the server here}

The component is meant to be used as a reusable card element. When we provide it with the data prop, it populates the card with the information you pass, making it easy to display different cards with different data throughout the application.

Thus, in this way, the whole system seamlessly works together to achieve the output.

**CHAPTER 5**

**RESULTS AND DISCUSSION**

**5.1 Performance testing using various metrics**

For such a system, we have to conduct several methods of testing so as to measure its feasibility. It is also important to note that this application can be regarded as a very early build and can be improved substantially by more experienced personnel. Moreover, tests are not limited to the ones given below, these were possible to carry out within the stipulated time period. Nevertheless, here are the tests that were carried out:

* Unit Testing
* Integration Testing
* Acceptance Testing

**5.1.1 Unit Testing**

The primary goal of unit testing is to ensure that each unit of code, typically a function or method, behaves as expected, and that it produces the correct output for a given input. It helps identify and fix bugs early in the development process, improving software reliability and maintainability.

Here is a tabular summary of the unit testing that was done.

|  |  |  |
| --- | --- | --- |
| S.no | Test | status |
| 1. | **Model Tests:** ensuring that data is stored, retrieved, and manipulated correctly | Works as expected |
| 2. | **View Tests**: functionality of views and how they handle incoming HTTP requests. | Works as expected |
| 3. | **Form Tests**: correct interpretation of the form data | Works as expected |
| 4. | **API Endpoint test:** url patterns correspond to correct APIs | Works as expected |
| 5. | ReactApp configuration | Works as expected |
| 6. | Card Component Rendering | Works as expected |
| 7. | React-router-dom route configuration allowing the user to access different parts of the app | Partial. Several features are not implemented yet |
| 8. | Recommendation model import | Works as expected |
| 9. | Recommendation module output generation | Partial. There are outlier values which can be improved by inducing a feedback loop. |
| 10. | Successful utilization of the output | Works as expected |

**5.1.2 Integration Testing**

focuses on assessing the interactions between different components or modules within a system. It aims to verify that these components work together as expected, detecting issues such as data flow, communication, and interface compatibility. Integration testing helps ensure the seamless integration of various parts of a software system and identifies any potential problems that might arise when they interact.

Here is a tabular summary of the integration testing that was done:

|  |  |  |
| --- | --- | --- |
| S.no | Type | status |
|  | Frontend-backend | Partial. The redirection works in a janky way for now because there is no proper session management. Accessing the reactjs app directly while skipping authentication is technically possible.  Best way to solve this is to make the entirety of the app in react js and skip the Django template system. The API should be made using Django REST framework for optimal working |
|  | Backend-recommender | Works as expected. |
|  | Backend-dataset | Works as expected. |

**5.1.3 Acceptance Testing**

It involves testing the software in a real-world context to validate that it aligns with the users' needs and expectations. Acceptance testing is crucial for confirming that the software is ready for deployment and can deliver the desired value to the end-users or clients.

Earlier in this document we described some goals for the application and hence we can use those as parameters for that:

|  |  |  |
| --- | --- | --- |
| S.no | Test/goal | status |
| 1. | Make a website that has the capability of registering NGOs and participants like restaurants along with their details like phone no, email and address. The website should also offer secure authentication and login. | Partial. The website is at a very alpha stage with the bare minimum features available. This project really serves as a starting point to develop a fully functioning application |
| 2. | Gather that information and store that in a relational manner so that it can be retrieved later. | Works as expected |
| 3. | From the information stored, develop a system that can take a look at factors like distance, ratings, quantity of food/type and gives a recommendation. | Partial. The recommender system is exclusively limited to these factors. Although, reconfiguration and re-training is straightforward |
| 4. | The recommendations must make the website dynamically show the necessary details of the concerned choice. | Works as expected |
| 5. | The recommender should be made in way that when trends change, it should be able to identify and adapt to it, while keeping the rest of the application intact. | Weak. There is no feedback loop implemented in order for the system to understand the trends |

**CHAPTER 6**

**CONCLUSION AND FUTURE SCOPE**

**6.1 Conclusion**

Hence, we highlighted the entire process of making the system from scratch.

Here is a summary of the application below:

* + We researched the role of ICT in situations like surplus food re-distribution
  + We then set out to find tools that will help us to make a website
  + We also found out a suitable machine-learning algorithm that helps us achieve the goal
  + We went ahead with the implementation and found out a way to train the system using a synthetic dataset
  + We then interpreted its recommendations and then combined its output with the website/
  + We performed some testing techniques to potentially find weakness

**6.2 Future Scope**

There are various aspects that can be expanded to this application:

* Incorporation of real-location data to calculate real-life distance data. Map services can be integrated and then distances calculated using Longitude and Latitude values
* Research and development of a custom algorithm for the recommender system
* Explore more web-development tools
* Try to incorporate DevOps techniques to host the application on the cloud

**REFERENCES**

1. Choirun A, Santoso I and Astuti R 2020 Sustainability risk management in the agri-food supply chain: Literature review IOP Conf. Ser. Earth Environ. Sci. 475
2. Lezoche M, Hernandez J, Alemany M, Lezoche M, Hernandez J, Alemany M, Panetto H, Kacprzyk J, Lezoche M, Hernández J E, Eva M, Diaz A, Panetto H and Kacprzyk J 2020 Agrifood 4 . 0 : a survey of the supply chains and technologies for the future agriculture To cite this version : HAL Id : hal-02395411 Future Agriculture Comput. Ind. 117 1–15
3. Sharma R, Kamble S S, Gunasekaran A, Kumar V and Kumar A 2020 A systematic literature review on machine learning applications for sustainable agriculture supply chain performance Comput. Oper. Res. 119 104926
4. ] Oluyisola O E, Sgarbossa F and Strandhagen J O 2020 Smart production planning and control: Concept, use-cases and sustainability implications Sustain. 12
5. Sisson, L. G. (2016) Food Recovery Program at Farmers' Markets Increases Access to Fresh Fruits and Vegetables for Food Insecure Individuals, Journal Of Hunger & Environmental Nutrition, 11(3): 337-339
6. Phillips, C., Hoenigman, R., Higbee, B., Reed, T. (2013) Understanding the Sustainability of Retail Food Recovery, PLOS ONE 8(10).
7. Alexander, C., and Smaje, C. (2008) Surplus retail food redistribution: An analysis of a third sector model. Resources Conservation and Recycling, 52 (11): 1290-1298.
8. Bilska, B., Wrzosek,M., Kolozyn-Krajewska, D., Krajewski, K. (2016) Risk of food losses and potential of food recovery for social purposes. Waste Management, 52: 269-277.
9. Caraher M, Cavicchi A (2014) Old crises on new plates or old plates for a new crisis. British Food Journal 116: 1392–1404.
10. Weymes, M. and Davies, A. R. (2018) Disruptive technologies? Scaling relational geographies of ICT-mediated surplus food redistribution, SHARECITY working paper 3, Trinity College Dublin.
11. UN Goal 12: Sustainable Development Knowledge Platform. Available online: https://sustainabledevelopment.un.org/sdg12 (accessed on 25 May 2019).
12. Swedish National Food Agency. Fler Gör Mer! Handlingsplan för Minskat Matsvinn 2030; Swedish National Food Agency: Uppsala, Sweden, 2018; p. 38.
13. Karlsson, J.O.; Röös, E. Resource-efficient use of land and animals—Environmental impacts of food systems based on organic cropping and avoided food-feed competition. Land Use Policy 2019, 85, 63–72.
14. Schneider, F. The evolution of food donation with respect to waste prevention. Waste Manag. 2013, 33, 755– 763.
15. Pauline Bergström 1, Christopher Malefors 1,\*, Ingrid Strid 1, Ole Jørgen Hanssen2 and Mattias Eriksson 1 1 Department of Energy and Technology, Swedish University of Agricultural Sciences, Box 7032, 75007 Uppsala, Sweden
16. Foodloopz Så Funkar Det. Available online: https://www.foodloopz.se/sa-funkar-det (accessed on 19 June 2019).

**APPENDIX A**

This section highlights the important parts of the website’s code:

**DATASET GENERATION- python**

import pandas as pd

import numpy as np

# Create an empty DataFrame

dataset = pd.DataFrame(columns=['NGO\_ID', 'Restaurant\_ID', 'Rating', 'Distance', 'Quantity', 'Category'])

# Number of data points

n = 15000

# Define the preference distribution

preference\_weights = [0.5, 0.3, 0.2]

data\_rows=[]

for \_ in range(n):

    # Generate random data for the features

    rating = np.random.choice([i for i in range(1,6)], p=[0.2 for \_ in range(5)])

    distance = np.random.randint(1,100)

    quantity = np.random.randint(1, 50)

    # Calculate the score based on the chosen legend

    if rating == 1:

        category = np.random.choice([1, 2, 3, 4, 5], p=[0.7, 0.1, 0.1, 0.05, 0.05])

    elif rating == 2:

        category = np.random.choice([1, 2, 3, 4, 5], p=[0.05, 0.35, 0.25, 0.15, 0.2])

    elif rating == 3:

        category = np.random.choice([1, 2, 3, 4, 5], p=[0.05, 0.1, 0.4, 0.2, 0.25])

    elif rating == 4:

        category = np.random.choice([1, 2, 3, 4, 5], p=[0.05, 0.15, 0.2, 0.3, 0.3])

    else:

        category = np.random.choice([1, 2, 3, 4, 5], p=[0.025, 0.025, 0.05, 0.05, 0.85])

    # Create a new row and add it to the dataset

    new\_row = {'Rating': rating, 'Distance': distance, 'Quantity': quantity, 'Category ': category}

    data\_rows.append({'Rating': rating, 'Distance': distance, 'Quantity': quantity, 'category': category})

# Save the dataset to a CSV file

dataset = pd.DataFrame(data\_rows)

dataset.to\_csv('model\generated\_dataset.csv', index=False)

**TRAINING AND TESTING- python**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier

import matplotlib.pyplot as plt

from sklearn.tree import plot\_tree

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, confusion\_matrix, classification\_report

# Load the dataset

dataset = pd.read\_csv('model/generated\_dataset.csv')

# dataset['Distance'] = dataset['Distance'] // 10

# dataset['Quantity'] = dataset['Quantity'] // 10

# print(dataset['Distance'].value\_counts())

# print(dataset['Quantity'].value\_counts())

# Split the data into features (X) and the target variable (y)

X = dataset[['Rating', 'Distance', 'Quantity']]

y = dataset['category']

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.08, random\_state=1)

# Create a Random Forest Regressor model

rf\_regressor = RandomForestClassifier(n\_estimators=40, max\_depth=5, random\_state=1)

# Train the model on the training data

rf\_regressor.fit(X\_train, y\_train)

# Make predictions on the test set

y\_pred = rf\_regressor.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

# Calculate precision, recall, and F1-score

precision = precision\_score(y\_test, y\_pred, average='weighted')

recall = recall\_score(y\_test, y\_pred, average='weighted')

f1 = f1\_score(y\_test, y\_pred, average='weighted')

# Display the confusion matrix

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

# Display a classification report

class\_report = classification\_report(y\_test, y\_pred)

print(f'Accuracy: {accuracy:.2f}')

print(f'Precision: {precision:.2f}')

print(f'Recall: {recall:.2f}')

print(f'F1-Score: {f1:.2f}')

print('Confusion Matrix:\n', conf\_matrix)

print('Classification Report:\n', class\_report)

# for i, tree in enumerate(rf\_regressor.estimators\_):

#     plt.figure(figsize=(50, 50))

#     plot\_tree(tree, filled=True, feature\_names=X.columns, rounded=True)

#     plt.title(f'Tree {i + 1}')

#     plt.show()

sample=[[1,15,10]]

print(f"output for {sample} will be {rf\_regressor.predict(sample)}")

**REACT CARD COMPONENT- javascript(react)**

import React from "react";

import Button from "@mui/material/Button";

import Typography from "@mui/material/Typography";

import StarRateIcon from "@mui/icons-material/StarRate";

const Card = ({ data }) => {

  return (

    <div classname='cards'>

        <br />

      <img src="https://picsum.photos/300/200" alt="cardimg" />

      <Typography variant="h5">{data.listings\_\_description}</Typography>

      <Typography variant="subtitle1">

        <strong>Quantity:</strong> {data.listings\_\_quantity}

      </Typography>

      <Typography variant="subtitle2">

        <strong>Email:</strong> {data.email}

      </Typography>

      <Typography variant="subtitle2">

        <strong>Phone No:</strong> {data.phoneno}

      </Typography>

      <Typography variant="subtitle2">

        <strong>Source Name:</strong> {data.name}

      </Typography>

      <Typography variant="subtitle2">{data.address}</Typography>

      <Typography variant="caption">

        <StarRateIcon fontSize="small" /> {getStarRating(data.ratings)}

      </Typography>

      <br></br><br></br>

      <Button

        style={{

          color: "black",

          borderColor: "black",

          borderRadius: 15,

        }}

        variant="outlined"

      >

        CONTACT THEM!

      </Button>

    </div>

  );

};

// Function to calculate star ratings based on the rating value

const getStarRating = (ratings) => {

  const ratingValue = parseFloat(ratings);

  if (ratingValue <= 1) {

    return "1 Star";

  } else {

    return `${Math.round(ratingValue)} Stars`;

  }

};

export default Card;

**SEVERAL IMPORTANT DJANGO APIS- python**

@csrf\_exempt

def get\_card\_data(request):

    if request.method == 'POST':

        try:

            # Parse the request body as JSON

            data = json.loads(request.body)

            # Extract the source\_id from the JSON data

            source\_id = data.get('source\_id')

            if source\_id is not None:

                # Retrieve the source information and related listings for the specified source\_id

                source\_data = SourceInfo.objects.filter(id=source\_id).prefetch\_related('listings').values(

                    'name', 'address', 'region', 'phoneno', 'email', 'ratings',

                    'listings\_\_quantity', 'listings\_\_food\_type', 'listings\_\_description'

                )

                if source\_data:

                    # Convert the query result to a list of dictionaries

                    source\_and\_listings = list(source\_data)

                    return JsonResponse(source\_and\_listings, safe=False)  # safe=False allows returning a list

                return JsonResponse({"message": "Source not found."}, status=404)

            else:

                return JsonResponse({"message": "Invalid request data. source\_id is missing."}, status=400)

        except Exception as e:

            return JsonResponse({"message": f"Error: {str(e)}"}, status=500)

    else:

        return JsonResponse({"message": "Invalid request method."}, status=405)

**----**

@api\_view(['POST'])

@csrf\_exempt

def usrlogin(request):

    username = request.data.get('username')

    password = request.data.get('password')

    http\_request = request.\_request

    try:

        user = authenticate(http\_request, username=username, password=password)

        if user is not None:

            login(http\_request, user)

            user\_data = {

                'username': user.username,

            }

            response\_content = {

                'message': 'Success!1!',

                'user': user\_data

            }

            return HttpResponse(content=str(response\_content), content\_type='application/json', status=200)

        else:

            return HttpResponse(content='{"message": "Invalid Credentials"}', content\_type='application/json', status=401)

    except Exception as e:

        return HttpResponse(content='{"message": "500: INTERNAL SERVER ERROR", "error": "' + str(e) + '"}', content\_type='application/json', status=500)

**REDIRECTION CODE- javascript(vanilla)**

const errorMessage = document.getElementById('error-mes');

// Add an event listener to the form submission

document.getElementById('login-form').addEventListener('submit', async (e) => {

    e.preventDefault(); // Prevent the form from submitting

    const form = e.target;

    const formData = new FormData(form);

    try {

        const r = await fetch('http://localhost:8000/loginauth', {

            method: 'POST',

            body: formData,

        });

        if (r.ok) {

            window.location.href = 'http://localhost:3000/';

        } else if (r.status === 500) {

            // Display the server error message

            errorMessage.innerText = 'Server Error: Please try again later.';

            errorMessage.style.display = 'block';

        } else {

            // Display a default error message

            errorMessage.innerText = 'Invalid credentials. Please try again.';

            errorMessage.style.display = 'block';

        }

    } catch (error) {

        // Display the error message and log the error

        errorMessage.innerText = `An error occurred: ${error.message}`;

        errorMessage.style.display = 'block';

        console.error('An error occurred:', error);

    }

});

**APPENDIX B**

Screenshots of the system are provided below

