School of Computer Science and Electronic Engineering

"Final Report"

Under the subject of CE903 Group Project

Web APP Based Recommendation System for Food Purchase Using Neural Networks

Version: (1.1) March 2023

Supervised by: Mr. Somdip Dey

Submitted by: Team 18

NAIR	AKSHAY MOHAN	2205124
PIMPARKAR	REVATI CHANDRASHEKHAR	2205450
RADFAR	SOBHAN	2202444
SAAD	MUHAMMAD	2201197
SAZESH	PAYMAN	2209425
SHAH	ANKUR YUKESHKUMAR	2201475
SHAH	PARIN JUGALBHAI	2201919
SHUKLA	TEJAS AKSHAYKUMAR	2200518
SIDDIQUIE	RIFFAT FATIMA	2201532
YOUSAF	MUHAMMAD FAIZAN	2201281

Contents:

1. Abstract----------3
2. Introduction----------5
3. The project started and gradually developed----------6
4. Recommendation System using neural network----------6
   - 4.1 Why Neural Networks, but not other technologies for recommendation systems?----------7
5. System Design----------8
   - 5.1 Synthetic Data----------8
   - 5.2 Models----------10
     - 5.2.1 Network Collaborating Filtering (NCF)----------10
     - 5.2.2 xDeepFM Model----------11
   - 5.3 WEB Application----------13
6. Implementation----------15
   - 6.1 Implementation of Synthetic Data----------15
   - 6.2 Implementation of Model----------19
     - 6.2.1 Implementation of NCF----------19
     - 6.2.2 Implementation of xDeepFM Model----------21
     - 6.2.3 Implementation of DQN----------21
     - 6.2.4 Implementation of WEB Interface----------24
     - 6.2.5 Final Output----------26
7. Software Testing and Risk Management----------27
   - 7.1 Neural Network-Based Recommendation System for Food Purchase Testing----------27
   - 7.2 Testing Process of Neural Network-Based Recommendation System for Food Purchase----------28
8. Conclusion----------28
9. References----------29

1 Abstract

This content discusses the application of neural networks in artificial intelligence (AI), specifically in developing a product suggestion web application that uses neural networks to provide personalized product recommendations to users. Neural networks are described as a key building block in AI, capable of pattern recognition, speech and picture recognition, and natural language processing. The project aims to create a user-friendly and efficient product suggestion system that takes into account a user's browsing and purchase history to suggest products that are most likely to be of interest to the user. The content further describes the practical project management approach adopted for the project, which proved effective in ensuring the successful completion of the project. Additionally, the content provides an overview of how a neural network-based recommendation system for food purchase operates, using a multi-layer perceptron neural network to suggest food items to consumers based on their tastes and food buying history. The article highlights the need to take into account various factors that impact a user's food choices, such as cultural background, dietary limitations, and personal taste preferences. Overall, the content highlights the advantages of using neural networks in AI and their potential to revolutionize the way businesses suggest products to their customers. The practical project management approach used for the "Product Suggestion WEB Application" project, resulted in the successful completion of the project. The project manager fostered trust and collaboration among team members, and each member had clear responsibilities. The system analyzes food-related data and provides suggestions to consumers based on their tastes and purchasing history. The neural network uses a multi-layer perceptron (MLP) architecture and is trained using food-related data, such as user preferences, ingredient lists, recipes, and nutritional information. The article explores why neural networks are preferred over other technologies for recommendation systems, including their ability to handle complex data, model non-linear interactions, the scale for large datasets, personalize recommendations, and continuously learn and improve over time. The project uses synthetic data to provide personalized product recommendations to users by analyzing their historical data. Synthetic data is artificially generated data designed to mimic real-world data, and it is used to augment the limited or insufficient user data in the networks. Various techniques, including GANs, VAEs, and Bayesian networks, can be used to generate synthetic data. GANs have proven to be especially useful in generating accurate synthetic data for recommendation systems. Synthetic data generation provides an efficient solution to collecting real-world user data, generating large amounts of data quickly and inexpensively while addressing privacy concerns. The article outlines a method for generating synthetic data using Python's Faker library to mimic the original data and provide recommendations for a more extensive dataset. The synthetic data was later modified to add more features like location, gender, and user information. Multiple models such as MLP, LSTM, NFC, transformer, and DQA were researched and implemented, each with their strengths and weaknesses. MLP was found to be a good multipurpose model for classification and regression but not suitable for sequential data or sparse datasets. LSTM was computationally expensive and difficult to evaluate. Transformers showed potential in capturing long-range dependencies and intricate relationships within input data. The NCF model, which is a deep learning-based method for collaborative filtering, was found to excel in learning and modeling complex relationships between users and items, leading to improved recommendation accuracy. The NCF model consists of an input layer, an embedding layer, an interaction layer, a multi-layer neural network, and an output layer. It can be enhanced with additional layers or components to improve its predictive capabilities. The project discusses two other machine learning algorithms: xDeepFM and DQN. xDeepFM is a model that combines a linear model, a plain deep neural network (DNN), and a

compressed interaction network (CIN) to learn both implicit and explicit feature interactions in data. The CIN learns high-order feature interactions at a vector-wise level and explicitly assesses them, resulting in a particular form of an image. On the other hand, DQN is a reinforcement learning algorithm that uses a neural network to approximate the optimal action-value function. In a grocery recommendation system, DQN can learn to make recommendations based on the user's current preferences and the products by observing and learning the user's behavior through states and actions. The Q-values generated by the DQN neural network represent the expected cumulative reward for performing an action in a given state, which are updated using the Bellman equation during training. The project improves the development of a web application using Django as the front-end framework and Python as the back-end lang the front-end will use Jinja formatting whiwhiche ack-enddack-end includes an Admin Panel and an Application Section. The project structure includes folders for URLs, Models, and Administration. The default database provided by Django-Admin is SQLite3, a NoSQL database. The Admin Panel contains components like templates, models, views, and URLs, and the User Table, Menu Management, User Menu Access Controller, Category, Subcategory, and Product make up the Data Dictionary. The User Panel includes components such as User, Menu Management, Category, Subcategory, Product, and Cart. The project implements three different predictive models: NCF, xDeepFM, and DQN, using a dataset of user-item purchase information. For NCF, the dataset was prepared by adding a 'Purchase' column and normalizing the dense features while encoding the sparse features. Negative sampling was done to represent non-purchase events. The model architecture included eight embedding layers and fully connected layers. Similarly, for xDeepFM, the data was prepared, features were engineered, and the model architecture was defined, compiled, trained, evaluated, and used for prediction. The implementation of DQN involved defining the environment, action space, state space, reward signal, and neural network. The neural network was trained using the DQN algorithm, and the performance of the recommendation system was evaluated. Finally, the three models were integrated for a better output. However, DQN was not successful for dynamic value, and the issue is being worked on. Software testing and risk management are crucial for delivering high-quality software products that satisfy end-user expectations. The adoption of a thorough testing and risk management approach is necessary to minimize the likelihood of mistakes and defects. A risk-based testing methodology and automated testing tools are popular strategies for managing risks in software testing. In the case of neural network-based recommendation systems for food purchases, it is necessary to test the system's accuracy and dependability using various testing techniques, such as functional testing, usability testing, and performance testing. The testing process for neural network-based recommendation systems includes training, validation, and final testing, with measures such as accuracy, precision, recall, and F1 score used to evaluate their effectiveness.

Index Terms: *Neural Network, Recommendation System, XDeepFM, DQN, NCF, Django, Python, Jinja Formatting, Black Box and White Box Testing, APA Formatting*

2 Introduction

In artificial intelligence (AI), which includes building intelligent machines that can learn from data and make decisions based on it, neural networks are a key building block. Neural networks, which are made up of linked nodes with the ability to carry out sophisticated calculations, are designed to mimic the structure and operation of the human brain.

Data is sent into a neural network and processed in various ways by each layer as it travels through the network's linked layers of nodes. The network's prediction or choice is represented by the output of the top layer.

For jobs requiring pattern recognition, including speech and picture recognition, neural networks are very helpful. They may also be applied in applications like anomaly detection and natural language processing.

The capacity of neural networks to learn from data is one of its main advantages. The network modifies its weights and biases throughout training to perform better on a given task. During this procedure, called backpropagation, the weights and biases are adjusted by the error between the network's output and the desired output.

Overall, neural networks are a potent artificial intelligence technology that has been utilized to develop a variety of applications, from virtual assistants to self-driving automobiles. Neural networks are going to become more crucial in determining the direction of AI as technology develops.

In recent years, the field of artificial intelligence has made remarkable progress in solving a wide range of problems. One of the areas where AI has shown significant potential is recommendation systems. These systems are used to suggest products to customers based on their preferences and historical data. To improve the accuracy of these systems, neural networks have been increasingly used as they can learn from large amounts of data and make predictions with high accuracy.

In this project, we aim to develop a product suggestion web application that uses neural networks to provide personalized product recommendations to users. The application will take into account a user's browsing history, purchase history, and other relevant data to suggest products that are most likely to be of interest to the user. The neural network model will be trained on a large dataset of product information and user behavior to ensure the accuracy of the recommendations.

The goal of this project is to create a user-friendly and efficient product suggestion system that will enhance the shopping experience for customers and increase sales for businesses. The web application will be accessible from desktop and mobile devices, providing users with a seamless shopping experience across different platforms. Overall, this project has the potential to revolutionize the way businesses suggest products to their customers, leading to increased customer satisfaction and loyalty.

3 The project started and gradually developed

At the outset of the "Product Suggestion WEB Application" project, the project manager fostered a sense of trust and collaboration among the team members. The project was thoroughly analyzed and comprehended by the group before breaking it down into six subgroups. The project manager then assigned tasks to each subgroup, ensuring that each team member had clear responsibilities. To ensure continuous progress, regular project meetings were held, and each member actively participated in developing the project. This practical project management approach allowed for effective delegation of tasks and clear communication among team members, leading to successful project completion.

Furthermore, a timeline and deadline for each task were implemented, ensuring that the project was completed within the expected timeframe. This helped the team members stay focused and motivated, as they had clear objectives to achieve. Additionally, the project manager regularly checked in with each subgroup to ensure that they were on track and provided necessary guidance and support.

To facilitate collaboration and transparency, the project manager also utilized Jira, which allowed for easy tracking of progress and clear communication among team members. Any challenges or issues that arose during the project were immediately addressed and resolved by the project manager and all the team members.

In conclusion, the practical project management approach adopted for the "Product Suggestion WEB Application" project proved to be effective in ensuring that the project was completed on time and to the satisfaction of all stakeholders. Through clear communication, effective delegation, and the use of project management tools, the project manager was able to foster a sense of teamwork and collaboration among team members, resulting in the successful completion of the project.

4 Recommendation System using neural network

A neural Network Based Recommendation System for Food Purchase is a sort of artificial intelligence that analyses food-related data, and food item purchases and provides suggestions to consumers based on their tastes as well as a food buying history. This kind of method may be applied in a variety of situations, including grocery shopping, restaurant suggestions, and meal planning.

In general, a food recommendation system employing neural networks operates by training a model using a lot of food-related data, including user preferences, ingredient lists, recipes, and nutritional data. The neural network analyses this information to identify trends and suggest which foods a user would likely like.

Using a multi-layer perceptron neural network is one typical method of creating a meal recommendation system (MLP). Several layers of nodes make up MLPs, and each one applies a mathematical operation on the incoming data. The suggestion of the system is produced by the last layer, which takes its input from the output of the previous layer.

Data is often preprocessed and input into the network in the form of vectors to train an MLP-based food recommendation system. Each element of the vector represents a distinct component of the data, such as the history of buying food items based on the search history of users. In forcing the discrepancy between its predictions and the actual user preferences, the system then modifies the weights of the nodes in each layer.

The necessity to take into account the vast range of elements that impact a user's food choices, such as cultural background, dietary limitations, and personal taste preferences, is one problem in developing a Neural Network Based Recommendation System for Food Purchases utilizing neural

networks. Some systems include user feedback and continuously revise their suggestions in response to user interactions to address this.

4.1 Why Neural Networks, but not other technologies for recommendation systems?

Although various technologies may be utilized for recommendation systems, neural networks are a preferred option due to their many benefits.

1. Ability To Handle Complex Data: The capacity to process and evaluate complex data kinds, such as text, pictures, and audio, is a strength of neural networks. This is especially helpful for recommendation systems that may draw information from several different sources.

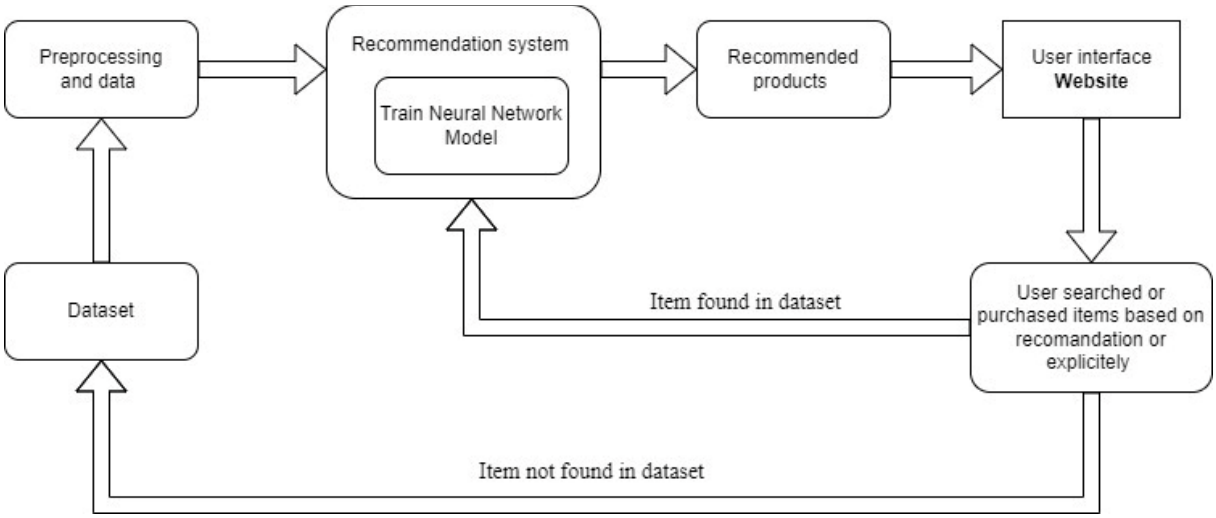
2. Non-linear modeling: Neural networks can simulate nonlinear interactions between consumers and products, which can be crucial for correctly forecasting user preferences.

3. Scalability: Neural networks are particularly suited for recommendation systems with a high number of users and products since they can be trained on enormous datasets.

4. Personalization: The probability that users will interact with the recommended goods may be increased by using neural networks to develop customized suggestions for specific users.

5. Continuous learning: Recommendation systems may adapt and get better over time because neural networks can be trained on fresh data as it becomes available.

In general, neural networks provide a potent tool for developing precise and efficient recommendation systems that can learn from enormous volumes of data and deliver user-specific recommendations. Even though there are alternative technologies that may be utilized for recommendation systems, neural networks have excelled in this area.



5 System Design

5.1 Synthetic Data

The Product Suggestion WEB Application based on neural networks is a software tool designed to provide personalized product recommendations to users. This application uses neural networks to analyze the user's historical data, such as their purchase history and browsing behavior, to generate relevant product recommendations. One of the key components of this application is the use of synthetic data. Synthetic data refers to artificially generated data that is designed to mimic real-world data. In the context of the Product Suggestion WEB Application, synthetic data is used to augment the existing user data, which may be limited or insufficient to train the neural networks. The use of synthetic data is important because it allows neural networks to learn from a larger and more diverse dataset. In other words, by generating additional data, the neural networks can become more accurate and effective at providing personalized product recommendations.

To generate synthetic data for the Product Suggestion WEB Application, various techniques can be employed. One common method is to use a generative adversarial network (GAN). A GAN is a type of neural network that consists of two networks - a generator and a discriminator. The generator is tasked with generating synthetic data, while the discriminator is trained to distinguish between real and synthetic data. The two networks are trained together, with the generator continually generating new synthetic data until the discriminator can no longer distinguish between real and synthetic data. Once the synthetic data is generated, it is combined with the real user data and used to train the neural networks. The resulting model is then used to make personalized product recommendations to users.

Recommendation systems have become an important component of many online platforms, including e-commerce, social media, and streaming services. These systems use user data to provide personalized recommendations for goods, content, and services. However, gathering user data can be time-consuming, costly, and raise privacy issues. Synthetic data generation provides a solution to these problems by generating artificial data that replicates real-world user behavior.

The process of creating synthetic data for recommendation systems includes using machine learning algorithms to simulate user behavior. These algorithms acquire patterns from real-world user data and generate synthetic data that closely resembles real data. The data produced can be used to train recommendation algorithms, which can then provide users with accurate and personalized recommendations.

One advantage of synthetic data generation is that it enables businesses to generate large amounts of data rapidly and at a lower cost than real-world data collection. Furthermore, because the generated data is not real, it does not raise privacy concerns, making it an appealing choice for businesses dealing with sensitive user information.

Faker python library, generative adversarial networks (GANs), variational autoencoders (VAEs), and Bayesian networks are some methods for generating synthetic data. GANs have proven especially useful in generating accurate synthetic data for recommendation systems. For example, Facebook researchers used GANs to create synthetic data for a movie recommendation system, which resulted in a significant increase in the recommendation algorithm's accuracy (Yesmin).

To summarize, synthetic data generation is an efficient solution to the challenges of collecting real-world user data for recommendation systems. It enables businesses to generate large amounts of data quickly and at a lower expense while addressing privacy concerns. Techniques such as GANs have shown encouraging outcomes in generating realistic synthetic data for recommendation systems.

The data generated in our system is entirely built on Python's Faker library (Welcome to Faker’s documentation! — Faker 5.0.1 documentation. (n.d.). It aided in the creation of synthetic data that resembles real-world data. The original real-world data sample was provided, and data generation was then implemented. The method that we used is outlined below.

- Every good recommendation system relies on good data to work upon.
- Considering the fact, the initial dataset provided had only data for 4 customers (21 rows), which wasn’t enough for the system to provide better recommendations.
- Therefore, the need for synthetic data rose.
- The data was generated to mimic the already provided dataset and have a way lot more customers.
- To do so, the python library functionary was used to generate the data for 2100 customers and had 125000 (approx.) rows of data for the system to work upon.
- This was achieved using the Faker library and newly generated synthetic data had the same fields as the original data.
- The results achieved for the initial synthetic data were found too scattered. Therefore, the dataset was later modified to make the results a bit more clustered.

- The dataset was later altered to add more attributes/ features – Location, gender, user information, etc.

5.2 Models

The next step is to prepare the data model, for this step I researched and implemented multiple models such as MLP, LSTM, NFC, transformer, and DQA. These models are state of art models and performed well under certain conditions. DQN and xDeepFM can achieve a high reward through each epoch. Here I will give basic details of all about the model and then start with the more actual model

Multi-Layer Perceptron is a feedforward neural network that is a universal approximator of any function. It is a multipurpose model that can handle tasks such as classification and regression. In a grocery recommendation system, MLP predicted which products a user is likely to purchase based on their previous purchase history, demographics, and other features.

However, MLP was not modeling sequential data or handling sparse datasets of grocery datasets. Synthetically generated data was mimicking the real world but was not as perfect and the model was overfitting.

LSTM was a good choice to choose for recommending finding the patterns in the user's behavior over past purchase history. However, it is computationally expensive and requires a huge dataset to learn the sequences. As well as evaluating the model was difficult, which is why the team did not pursue this further.

Transformers, an innovative deep learning architecture initially developed for natural language processing tasks, have demonstrated considerable potential in multiple domains, including recommender systems. By employing self-attention mechanisms, transformers effectively capture long-range dependencies and intricate relationships within input data, making them ideal for modeling user-item interactions in recommendation tasks.

5.2.1 Network Collaborating Filtering

Neural Collaborative Filtering (NCF) is an advanced, deep learning-based method for collaborative filtering, aiming to accurately predict user preferences and deliver tailored recommendations. By integrating the benefits of conventional collaborative filtering techniques with neural network capabilities, NCF excels at learning and modeling complex relationships between users and items, leading to improved recommendation accuracy.

The NCF model is built on user and item latent features, which are depicted as embeddings. These embeddings are then processed through a multi-layer neural network, enabling the model to identify and learn more sophisticated patterns compared to traditional matrix factorization methods. A key reason for NCF's increased accuracy is its capacity to capture nonlinear relationships between users and items, thus revealing intricate patterns and enhancing prediction accuracy.

The main components of a neural collaborative filtering model include:

1. Input: Identifying users and items.
2. Embedding Layer: This layer converts user and item IDs into dense vectors, or embeddings, representing latent characteristics.

3. Interaction Layer: This layer combines user and item embeddings using various techniques, such as concatenation or element-wise multiplication, to capture their interactions.
4. Multi-Layer Neural Network: This section consists of several fully connected layers that learn and model nonlinear relationships in user-item interactions.
5. Output Layer: The final layer produces a prediction score indicating the user's preference for a particular item.

NCF models can be developed using different loss functions and optimization methods. Once trained, they enable personalized recommendations by predicting user preferences for unseen items. Furthermore, NCF's deep learning foundation offers adaptability to various data types and the ability to incorporate a range of information sources, ensuring that the model effectively learns from both explicit and implicit feedback.

Additionally, NCF can be enhanced with extra layers or components to further improve its predictive capabilities. For example, integrating attention mechanisms or convolutional layers can help the model concentrate on specific, relevant features or detect local patterns in the data. These enhancements contribute to the model's ability to generate more accurate recommendations.

In conclusion, the increased accuracy of Neural Collaborative Filtering can be attributed to its proficiency in modeling complex nonlinear relationships, its adaptability to diverse data types, and its potential for expansion with additional components. This approach has shown significant improvements in accuracy and performance compared to traditional collaborative filtering techniques, underscoring its importance in the realm of recommendation systems (He et al.,2017 Zhang et al.,2017).

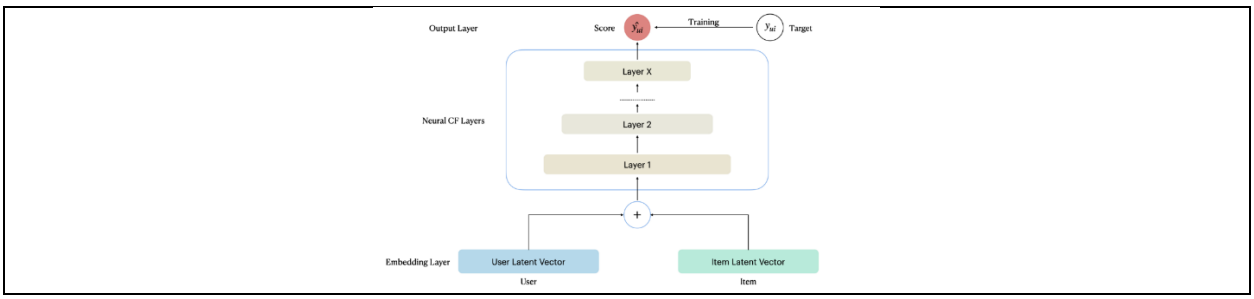


Fig1. Neural collaborative filtering framework

5.2.2 xDeepFM Model

To learn both implicit and explicit feature interactions in data, xDeepFM combines a linear model, a plain deep neural network (DNN), and a compressed interaction network (CIN). The linear model operates directly on raw input features, the plain DNN operates on dense feature embeddings, and the CIN learns explicit high-order feature interactions via its hidden layers.

The CIN learns feature interactions at a vector-wise level rather than a bitwise level, and it explicitly assesses high-order feature interactions. Unlike DNN-based systems, where high-order interactions are learned implicitly, their complexity does not expand exponentially with the degree of interaction. Similar to RNN-based models, the CIN's hidden layers are specified by a learnable

parameter, and its output is reliant on the last hidden layer and extra input. The output of the CIN is treated as a particular form of an image, and the learnable parameter is utilized as a filter, as in CNN-based models.

The hidden layer results are totaled row by row and then concatenated before being supplied to the output layer. The output equation is used to train the linear, CIN, and DNN layers in parallel. The model can learn both low-order and high-order feature interactions, and it can supplement both implicit (DNN) and explicit (CIN) feature interactions.

Sum pooling is used at each hidden layer to generate the following pooling vector (Sharma, 2019):

$$p_i^k = \sum_{j=1}^{D} \mathbf{X}_{i,j}^k$$

The output equations of xDeepFM depend on Linear, CIN, and DNN models and are all trained in parallel with the following output equation (Sharma, 2019):

$$\hat{y} = \sigma(\mathbf{w}_{linear}^T \mathbf{a} + \mathbf{w}_{dnn}^T \mathbf{x}_{dnn}^k + \mathbf{w}_{cin}^T \mathbf{p}^+ + b)$$

CIN has a space complexity of O (mHH*T), where m is the number of rows in the original feature matrix, H is the number of rows in the feature matrix at hidden layer k, and T is the network depth. The temporal complexity is O (mHD * H * T), where MHD is the cost of computing Z(k+1,h) for one row, H is the number of feature vectors (rows) at layer H, and T is the total number of hidden layers in CIN.

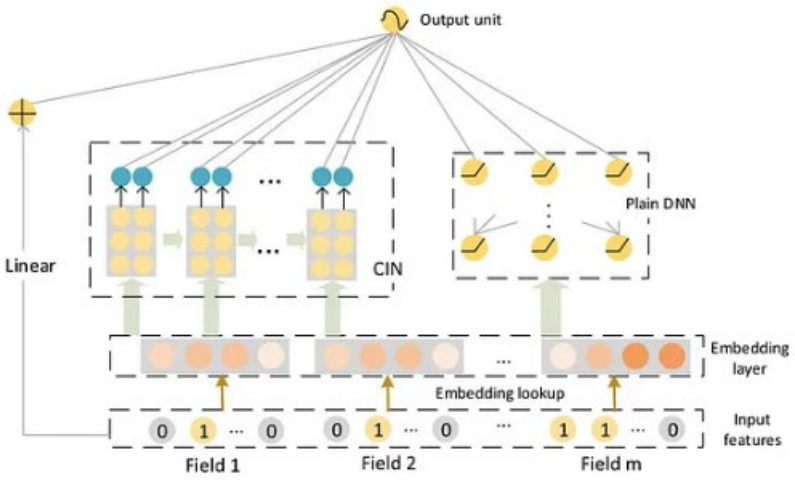
The main components of an xDeepFm model include:
Input Layer: xDeepFM's input layer accepts one-hot encoded sparse features as input. An embedding layer is used to turn these traits into dense embeddings.

The CIN (Compressed Interaction Network) Layer is in charge of recording higher-order interactions between features. It takes dense embeddings from the input layer as input and computes feature cross-interaction at various levels of interaction. The CIN layer produces a set of feature maps.

The FM (Factorization Machine) Layer is in charge of recording the pairwise interactions between features. It takes as input the dense embeddings from the input layer and computes the dot product of all pairwise feature pairings. The FM layer produces a scalar result.

The DNN (Deep Neural Network) Layer is a stack of completely connected layers. It accepts as input the concatenation of the input layer's dense embeddings and the CIN layer's feature maps. The DNN layer is in charge of capturing feature non-linear relationships.

Output Layer: The xDeepFM output layer is a sigmoid activation function that converts the DNN layer output to a value between 0 and 1, representing the anticipated probability of the positive class (Sharma, 2019).



2. DQN is a type of reinforcement learning algorithm that uses a neural network to approximate the optimal action-value function. It can learn to make rational decisions in a changing environment. In a
grocery recommendation system, DQN can learn to make recommendations based on the user's current preferences and the products. Through states and actions, the user's behavior can be observed and learned and the products would be recommended according to learned behavior.

The DQN neural network has been trained to predict the Q-values for every action that might be taken in a particular condition. The state is entered into the neural network, which then generates a set of Q-values, one for each action. The expected cumulative reward for performing that action in the current state and then implementing the best course of action is represented by the Q-values. The agent interacts with the environment while learning, choosing actions based on the greatest projected Q-value with a chance of trying out novel ones. Each action the agent does is rewarded, and the new state and reward are saved in a replay buffer. The Bellman equation is used to update the Q-value targets while the neural network is trained on a collection of transitions taken from the replay buffer (Chen and Haicheng. 2021).

5.3 WEB Application

- Front-End: -
    a. We are using Django as a framework for creating the web app front-end designs.
    b. In that part, we will be going to use Jinja formatting for all the pages.

Django is a popular web framework that allows developers to quickly and easily create complex web applications. It comes with a built-in templating language for creating dynamic web pages, but you may also use other templating languages such as Jinja.

Jinja is a modern and designer-friendly Python templating language that is comparable to Django but provides additional features and flexibility. It offers capabilities like template inheritance, filters, and macros and allows you to construct reusable templates with dynamic content.

You may still use all of Django's built-in features and capabilities while utilizing Jinja with Django, such as database models, URL routing, and form handling. Jinja templates can be used in conjunction with Django views to build HTML pages and display data from databases.

- Back-End: -
    c. As a back-end, we will go to implement python.
    d. Specifically, we will be going to develop below formatting: -
        i. Admin Panel
        ii. Application Bar

Python is a popular programming language that is extensively used for back-end development due to its simplicity, versatility, and extensive library and framework ecosystem.

The back-end development in the given project structure is divided into two parts: an Admin Panel and an Application Component. The Admin Panel is most likely used to manage the application's backend, whilst the Application Section is used to construct the web app's major features and functionalities.

The following elements are included in the project structure:

Project: The top-level folder containing the entire project. It has subfolders for URLs, Models, and Administration.

URLs: This folder includes the project's URL routing configuration.

Models: This folder includes the project's database models.

Admin: This folder contains the setup for the Django Admin Panel, which is used to manage the application's backend.

Admin App: This section contains the Admin Panel components, such as templates, models, admin.py, views, and URLs.

Templates: The HTML templates for the web app are stored in this folder.

Views: The functions in this folder handle HTTP requests and generate HTTP responses.

Web App: This folder contains the components required for the web app's main functionality, such as templates, models, admin.py, views, and URLs.

- Data Base: - As a default database provided by Django-Admin, we will be going to use SQlite3 as a database of our system. This database is the NoSQL database. Where there is no need to run any queries to add, edit, delete, or fetch the data from the database.

Data Dictionary: -

- Admin Panel: -
    a. User Table: -
    b. Menu Management: -
    c. User Menu Access Controller: -
    d. Category: -
    e. Subcategory: -
    f. Product: -
    g.

Field Name	Description

- User Panel: -
    a. User: -
    b. Menu Management: -
    c. Category: -
    d. Subcategory: -
    e. Product: -
    f. Cart: -

6 Implementation

6.1 Implementation of Synthetic Data

A methodology on synthetic data generation using Python's Faker library for a food items recommendation system:

- Define the schema: First, define the schema for the food items dataset. This may include attributes such as name, description, ingredients, category, price, etc.

- Install and import the Faker library: The Faker library is a Python library that generates fake data, including names, addresses, and other attributes. Install the Faker library by running "pip install Faker" and importing it into your Python script.

- Generate fake data: Use the Faker library to generate fake data for each attribute in the schema. For instance, use the "name" method to generate a fake name for each food item, and the "text" method to generate a fake description.

- Create a pandas Data Frame: Use the Pandas library to create a Data Frame for the generated data. Each row in the data frame represents a food item, and each column represents an attribute.

- Save the synthetic dataset: Save the synthetic dataset as a CSV or JSON file for later use in the recommendation system.

The initially available dataset included features under 2 categories:

1. Product Data
2. Purchase Data

Then we try to update our Synthetic data using Faker and Python script and first we got 5-column data which is similar to our original data with some added column
Then we got a more updated database. All result of our database is mentioned in n images.

A	B	C	D	E	F	G
customer_id	customer_name	product_ic	purchase_date_time	invoice_id		
1	Gwendolyn Bennett	38	51:24.4	1		
1	Gwendolyn Bennett	55	24:06.2	2		
1	Gwendolyn Bennett	159	22:21.5	3		
1	Gwendolyn Bennett	163	33:18.7	4		
1	Gwendolyn Bennett	156	20:06.1	5		
2	Cheryl Williams	134	30:23.4	6		
2	Cheryl Williams	23	34:20.4	7		
2	Cheryl Williams	133	29:54.0	8		
2	Cheryl Williams	138	39:43.9	9		
2	Cheryl Williams	199	48:06.9	10		
2	Cheryl Williams	28	43:32.3	11		
3	Mallory Murray	85	40:18.2	12		
3	Mallory Murray	43	46:25.7	13		
3	Mallory Murray	15	31:33.7	14		
3	Mallory Murray	59	43:59.3	15		
3	Mallory Murray	100	52:27.3	16		
3	Mallory Murray	56	21:50.7	17		
3	Mallory Murray	46	48:34.6	18		
4	Kenneth Brady	33	47:26.7	19		
4	Kenneth Brady	27	43:39.9	20		
4	Kenneth Brady	197	43:13.5	21		
4	Kenneth Brady	156	00:48.0	22		
4	Kenneth Brady	54	34:55.3	23		
5	Marc Gallagher	197	52:06.8	24		
5	Marc Gallagher	153	28:20.1	25		
5	Marc Gallagher	125	38:12.9	26		

	A	B	C	D	E	F	G	H	I	J	K	L	M
1		First name	Last Name	Gender	E-mail	Address	City	Contact	Product ID	Invoice Id	Date	Time	price in pounds
2	0	Kimberly	Barrett	M	Kimberly.Barrett@wilson.com	Studio 53 Lane port	Wembley	+44(0)1144960737	126	1348	22/02/1971	22:36:04	3
3	1	Shirley	Gonzalez	M	Shirley.Gonzalez@hughes-wright.biz	519 Joshua run	Southwark	44292018185	154	331	01/01/1983	03:54:22	45
4	2	Rebecca	Stanley	F	Rebecca.Stanley@huerta-simmons.info	Studio 1 Keith landing	Bourton-on-the-W	1174960244	136	733	26/11/1992	07:36:52	39
5	3	Misty	Anderson	M	Misty.Anderson@yu.com	462 Taylor springs	Bromley	(020) 74960541	67	386	19/11/1979	20:12:48	38
6	4	Heather	Hartman	F	Heather.Hartman@kemp.com	Flat 73 Chan pike	Stratford-upon-Avc	+44(0)28 9018 084	127	93	28/12/1998	06:22:25	47
7	5	Carol	Williams	F	Carol.Williams@higgins.com	5 Kim gardens	Oxford	1174960463	136	302	07/10/1974	14:19:57	25
8	6	Natalie	Baldwin	M	Natalie.Baldwin@hoffman.info	Flat 3 Pratt square	Portsmouth	(01632) 960 909	164	913	20/05/2019	03:50:29	22
9	7	Elizabeth	Mendoza	F	Elizabeth.Mendoza@hall.com	Studio 36q Paula cliffs	Salisbury	4;41165E+11	167	643	24/04/2019	21:16:46	27
10	8	Crystal	Mcgrath	F	Crystal.Mcgrath@barnes.com	Flat 8 May ridge	Stratford-upon-Avc	2074960739	91	603	17/01/1992	09:37:50	6
11	9	Brianna	Chan	M	Brianna.Chan@lozano.com	Studio 51 Harding track	Greenwich	(0118)4960700	122	148	23/01/2023	11:47:13	13
12	10	Janice	Gallagher	M	Janice.Gallagher@stewart.com	9 Richard mission	Windsor	0117 496 0091	32	1050	25/02/2015	04:27:35	34
13	11	Kelsey	Terry	F	Kelsey.Terry@mendoza.com	6 Richards corners	Scotland	(0113) 496 0893	29	192	07/12/1977	22:29:01	42
14	12	Kelly	Davis	F	Kelly.Davis@ruiz.net	Studio 7 Power pass	Scotland	(0116) 496 0383	76	1101	26/11/1977	01:25:34	15
15	13	Linda	Crawford	M	Linda.Crawford@williams-white.com	7 Byrne vista	Brighton	+44808 1570261	127	1842	19/02/1994	14:03:08	12
16	14	Caitlyn	Buck	F	Caitlyn.Buck@cline.com	39 Goodwin bypas	Stratford-upon-Avc	0114 4960255	168	1767	07/04/2017	04:26:20	44
17	15	Sylvia	Peterson	M	Sylvia.Peterson@cunningham-kidd.com	Studio 54 Flynn row	Stratford-upon-Avc	(0131)4960672	58	900	25/03/2015	10:27:53	31
18	16	Leslie	Rogers	M	Leslie.Rogers@richardson-moreno.net	Studio 67 Sally glens	Oxford	+4428 9018 0003	177	140	07/02/1989	03:19:01	28
19	17	Rhonda	Jackson	F	Rhonda.Jackson@stephens-hubbard.info	Studio 06K Margaret landing	Wembley	4;4307E+11	118	61	16/07/1995	02:34:57	21
20	18	Carla	Andrews	F	Carla.Andrews@dixon.info	1 Williams fall	Bristol	(0116)4960819	19	1130	09/10/1975	12:41:41	31
21	19	Autumn	Marshall	M	Autumn.Marshall@baker.com	0 Olivia knolls	Bromley	8081570333	165	416	22/06/1984	04:04:20	2
22	20	Linda	Alvarez	F	Linda.Alvarez@stewart.biz	Studio 05 Edwards fort	Edinburgh	+44(0)808157060	66	1996	03/04/1983	12:10:37	9
23	21	Lauren	Johnson	M	Lauren.Johnson@smith-pineda.com	65 Ross crescent	Scotland	0161 4960974	164	463	06/04/2017	19:40:57	28
24	22	Cindy	Brown	F	Cindy.Brown@fleming-rodriguez.com	4 Steele mill	Scotland	(0161) 4960706	17	1093	28/09/1996	10:55:37	49
						Studio 0							

Product data included: Product id, product, shelf life, and usual storage, whereas

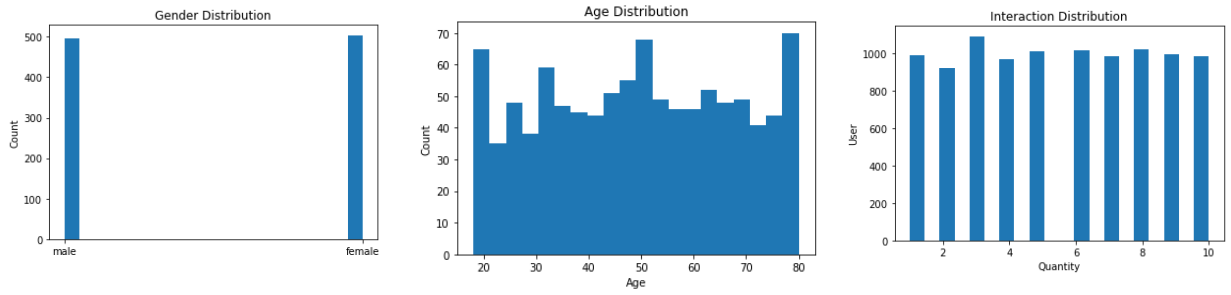
Purchase data included: Customer ID, customer name, product id purchased, purchase date and time, and invoice id.

The updated version of the data created included more features to get better recommendations along with the personal details of the customers. It included features such: as user details, product details, the quantity bought, time, age, gender, location, category, price, brand, shelf life, usual storage, day and time of the week bought, and the weather. The interesting thing about this approach is that not many have included the situation features such as time of the day or weather into consideration for learning the user behavior. The data generated had 10000 rows and 16 columns.

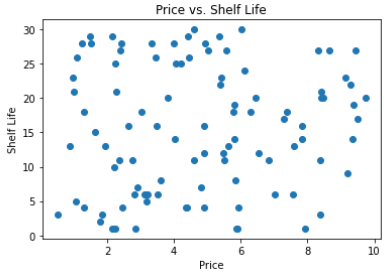
	user_id	product_id	quantity	timestamp	age	gender	location	product_name	category	price	brand	shelf_life	usual_storage	day_of_w
0	548	98	2	2022-01-01 00:00:00.000000000	43	female	Nottingham	Product 98	Bakery	6.677217	Brand C	2	Pantry	Satur
1	452	98	4	2022-01-25 23:20:58.325832583	31	male	Edinburgh	Product 98	Bakery	6.677217	Brand C	2	Pantry	Tues
2	845	98	2	2022-07-26 07:11:10.747074706	55	female	Belfast	Product 98	Bakery	6.677217	Brand C	2	Pantry	Tues
3	752	98	3	2022-12-28 12:35:21.332133212	49	male	Bristol	Product 98	Bakery	6.677217	Brand C	2	Pantry	Wednes
4	660	98	5	2022-06-18 08:48:14.689468946	72	female	Sheffield	Product 98	Bakery	6.677217	Brand C	2	Pantry	Satur
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
9995	221	49	2	2022-05-31 11:49:24.896489648	60	female	Manchester	Product 49	Bakery	9.008012	Brand B	24	Refrigerator	Tues
9996	888	49	4	2022-08-26 21:32:49.036903688	40	female	Liverpool	Product 49	Bakery	9.008012	Brand B	24	Refrigerator	Fri
9997	506	49	2	2022-12-27 04:15:46.174617460	44	male	Leeds	Product 49	Bakery	9.008012	Brand B	24	Refrigerator	Tues
9998	436	49	1	2022-07-10 14:37:37.425742574	44	female	Nottingham	Product 49	Bakery	9.008012	Brand B	24	Refrigerator	Sun
9999	509	49	2	2022-09-29 04:58:15.229522952	28	female	Bristol	Product 49	Bakery	9.008012	Brand B	24	Refrigerator	Thurs

10000 rows × 16 columns

The generation of the data is done through random selection in a range of certain features. The users and products were created in sets and then merged through interaction sets. In which the details such as quantity bought, time, weather, etc. were merged. The image shows the distribution of each feature distribution.



So, changing the data from the previously generated to new data by adding more features and attaining a statistical relationship between them. The gender distribution between males and females is equal, and the data is unbiased in this feature.



As mentioned above, the initial draft of our synthetic data did not give clustered results refer to fig.1., we modified our data and got more clustered data, refer to fig.2.

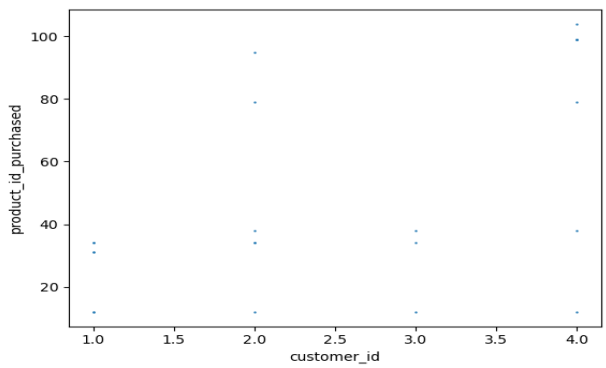


Fig1

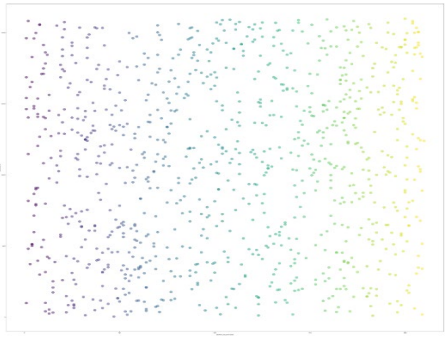


Fig2

It was seen that the initial dataset which was altered to accommodate a loosely bound clustering structure wasn't giving out the best results for the models that were been tried. This dataset was working fine with the base recommendation model (without any neural networks). Therefore, this dataset was also altered to accommodate more features and to work properly with the selected Neural Network models.

6.2 Implementation of Model

6.2.1 Implementation of NCF

In this study, we aim to develop a predictive model using a dataset containing user-item purchase information. The following steps were undertaken to prepare the dataset and develop the model.

1. Importing Essential Libraries: The required libraries were imported to facilitate data manipulation, feature engineering, and model training.

2. Dataset Preparation: A new column named 'Purchase' was added to the dataset, with all rows initially set to 1, indicating that a purchase was made for the respective item. The dataset contained two types of features:

   a. Dense Features: quantity, age, price b. Sparse Features: User_id, Product_id, Gender, Location, Product_name, Category, Brand, Day_of_week, Time_of_day

3. Feature Engineering: The Dense Features were normalized to ensure equal contribution to the model. Sparse Features were encoded using appropriate encoding techniques to convert them into a format suitable for the model.

4. Negative Sampling: To better represent the non-purchase events, four random items that the user had not purchased were added to the dataset. The 'Purchase' column for these items was set to 0.

5. Vectorization: All features were vectorized in preparation for input to the embedding layers of the model.

6. Model Architecture: The proposed model consisted of the following components:

   a. Eight separate embedding layers for each feature, each with an embedding dimension of eight.

   b. The outputs of the embedding layers were concatenated and passed to a series of fully connected layers for further processing.

This model can serve as a valuable tool for understanding user behavior and preferences, ultimately leading to more effective marketing strategies and improved customer satisfaction.

```
self.user_embedding = nn.Embedding(num_embeddings=data['user_id'].max()+1, embedding_dim=8)
self.item_embedding = nn.Embedding(num_embeddings=data['product_id'].max()+1, embedding_dim=8)
self.age_embedding = nn.Embedding(num_embeddings=data['age'].max()+1, embedding_dim=8)
self.gender_embedding = nn.Embedding(num_embeddings=data['gender'].max()+1, embedding_dim=8)
self.location_embedding = nn.Embedding(num_embeddings=data['location'].max()+1, embedding_dim=8)
self.category_embedding = nn.Embedding(num_embeddings=data['category'].max()+1, embedding_dim=8)
self.price_embedding = nn.Embedding(num_embeddings=data['price'].max()+1, embedding_dim=8)
self.brand_embedding = nn.Embedding(num_embeddings=data['brand'].max()+1, embedding_dim=8)

self.fc1 = nn.Linear(in_features=64, out_features=128)
self.fc2 = nn.Linear(in_features=128, out_features=256)
self.fc3 = nn.Linear(in_features=256, out_features=512)
self.fc4 = nn.Linear(in_features=512, out_features=64)
self.output = nn.Linear(in_features=64, out_features=1)
self.data = data
```

6.2.2 Implementation of xDeepFM Model

Data preparation: It entails cleaning and altering the data as needed. Normalize numerical characteristics and convert categorical features to one-hot encoding.

Feature Engineering: Create embeddings for the categorical features. Calculate the input layer's field embedding vectors and the CIN layer's feature maps.

Model Architecture: Using the Keras API, define the XdeepFM model architecture. Separate layers should be created for the input, CIN, FM, and DNN layers. Concatenate the outputs of the CIN and DNN layers and feed them via the output layer.

Model Compilation: Set the loss function, optimizer, and metrics to compile the model. As the loss function, use binary cross-entropy, Adam as the optimizer, and accuracy as the measure.

Model Training: Using the training data, train the model. To train the model, use the Keras API's fit () function. To avoid overfitting, keep an eye on the validation accuracy throughout training.

Model Evaluation: Using the test data, evaluate the model. To evaluate the model on the test data, use the Keras APIs evaluate () method.

Model Prediction: Make predictions on new, unseen data using the trained XdeepFM model.

```
# Defining the xDeepFM model
model = xDeepFM(linear_feature_columns, dnn_feature_columns, dnn_hidden_units=(512, 512),\
    cin_layer_size=(512, 512), \
    cin_split_half=True, cin_activation='relu'\
    ,l2_reg_linear=1e-05,\
    l2_reg_embedding=1e-05, l2_reg_dnn=0, l2_reg_cin=0, \
    # init_std=0.0001,\
    seed=1024, dnn_dropout=0.2,dnn_activation='relu', \
    dnn_use_bn=True, task='binary')

# Compiling the model
model.compile("adam", "binary_crossentropy",metrics=['binary_crossentropy'],)

# Training the model
history = model.fit(train_model_input, train[target].values,
                    batch_size=32, epochs=20, verbose=2, validation_split=0.1, )
```

6.2.3 Implementation of DQN

Step of implementation of the DQN Model
Define the environment: Define the set of items that the recommendation system can recommend. Each item should have a set of features that the neural network can use to make recommendations.

Define the action space: Define the set of actions that the neural network can take, which is the set of items that can be recommended at any given time.

Define the state space: Define the current state of the recommendation system, which includes the items that have already been recommended and any user feedback that has been received.

Define the reward signal: Define the reward signal based on how well the recommended items match the user's preferences. For example, if the user clicks on an item that was recommended, the reward would be positive. If the user ignores the recommended items, the reward would be negative.

Define the neural network: Define the neural network architecture, which should take the current state of the recommendation system as input and output a set of Q-values for each possible action. The neural network should be trained using a variant of the DQN algorithm, such as Double DQN or Dueling DQN.
Initialize the replay buffer: Initialize a replay buffer, which is a data structure that stores transitions consisting of the current state, action, reward, next state, and terminal flag.

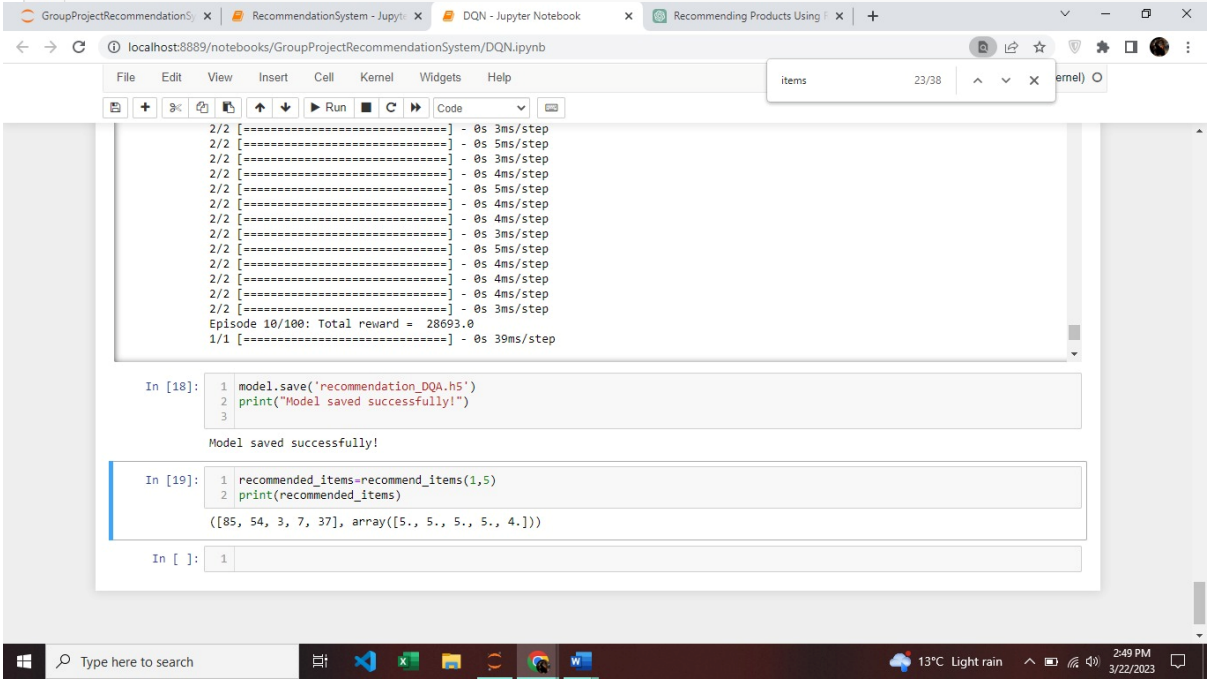
Train the neural network: Train the neural network using the transitions stored in the replay buffer. During training, the neural network should select actions based on the highest Q-value, with some probability of exploring new actions.

Evaluate the performance: Evaluate the performance of the recommendation system by measuring the average reward received over a set of test users.
Repeat steps 7-8 until convergence: Continue training and evaluating the recommendation system until the average reward received stabilizes or reaches a predetermined threshold.

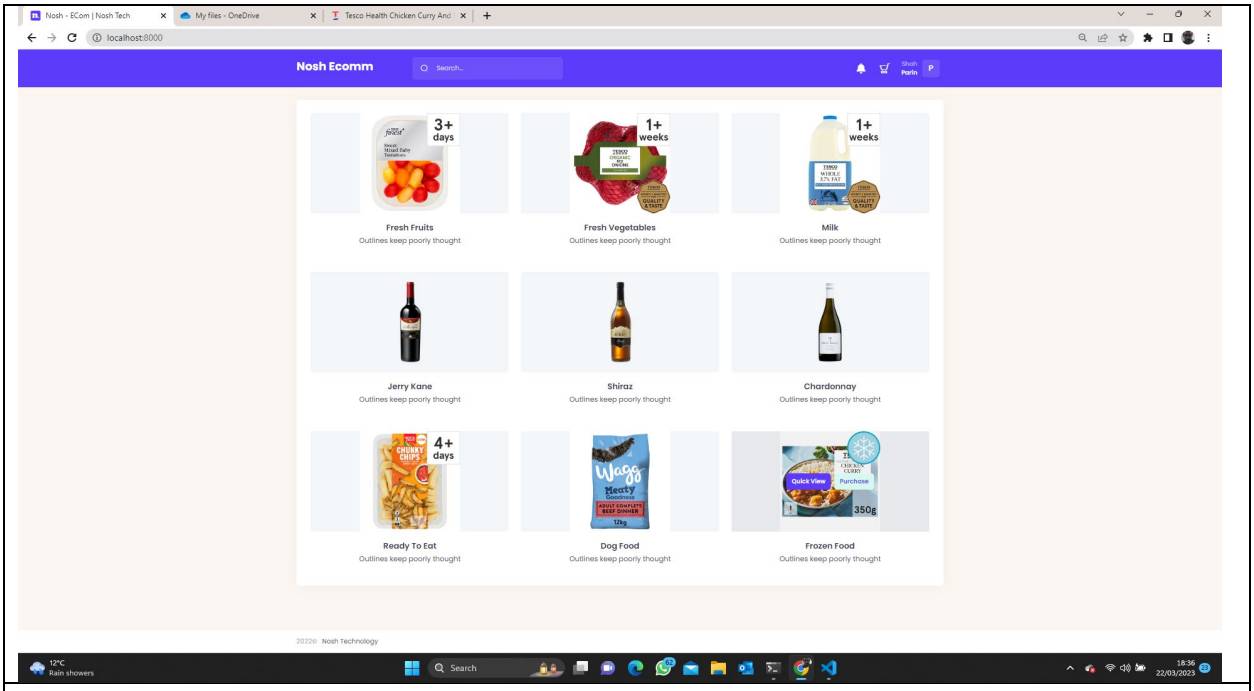
```
import pandas as pd
import numpy as np
import random

# Generate user data
users = pd.DataFrame({
    'user_id': range(1000),
    'age': [random.randint(18, 80) for i in range(1000)],
    'gender': [random.choice(['male', 'female']) for i in range(1000)],
    'location': [random.choice(['Belfast', 'Birmingham', 'Bristol', 'Cardiff', 'Edinburgh', 'Glasgow', 'Leeds', 'Liverpool',
})

# Generate item data
items = pd.DataFrame({
    'product_id': range(100),
    'product_name': ['Product {}'.format(i) for i in range(100)],
    'category': [random.choice(['Produce', 'Dairy', 'Bakery', 'Meat', 'Snacks']) for i in range(100)],
    'price': [random.uniform(0.5, 10) for i in range(100)],
    'brand': [random.choice(['Brand A', 'Brand B', 'Brand C']) for i in range(100)],
    'shelf_life': [random.randint(1, 30) for i in range(100)],
    'usual_storage': [random.choice(['Pantry', 'Refrigerator', 'Freezer']) for i in range(100)]
})

# Generate interaction data
interactions = pd.DataFrame({
    'user_id': [random.randint(0, 999) for i in range(10000)],
    'product_id': [random.randint(0, 99) for i in range(10000)],
    'quantity': [random.randint(1, 10) for i in range(10000)],
    'timestamp': pd.date_range(start='2022-01-01', end='2022-12-31', periods=10000)
})

# Combine data into a single dataframe
data = interactions.merge(users, on='user_id').merge(items, on='product_id')

# Generate contextual data
```

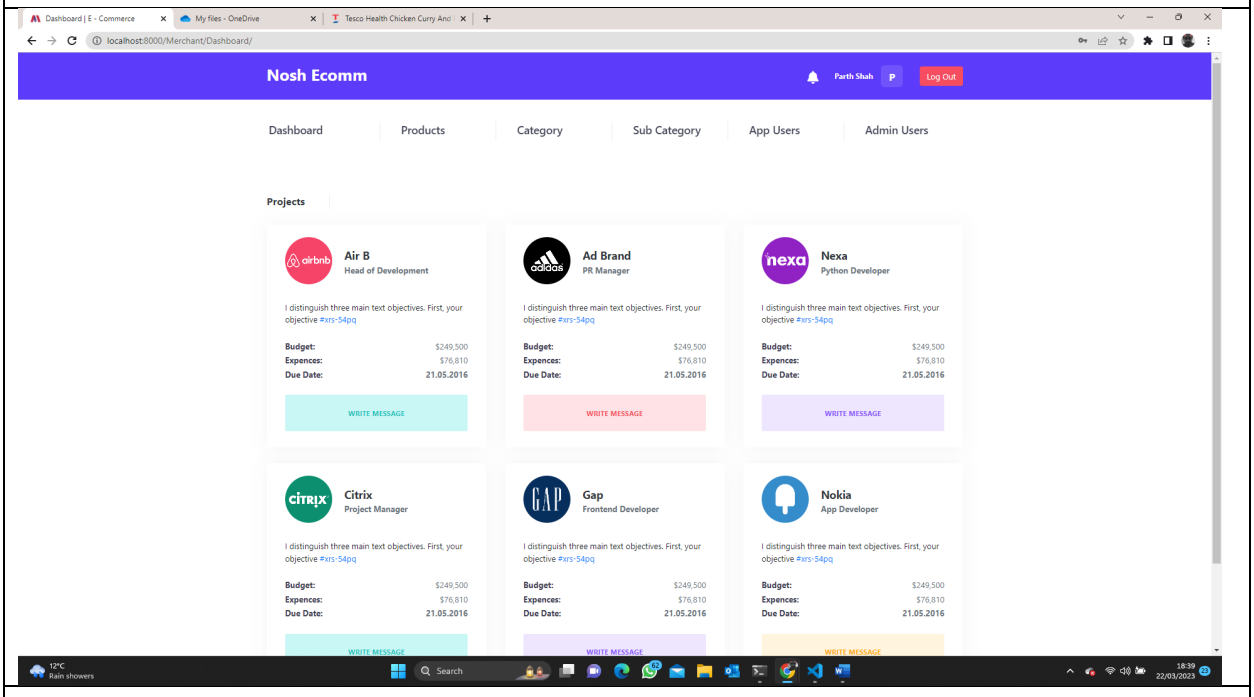
```
# Convert data to user-item matrix
user_item_matrix = pd.pivot_table(data, values='quantity', index='user_id', columns='product_id', fill_value=0)

# Define hyperparameters
NUM_USERS = len(user_item_matrix)
NUM_ITEMS = len(user_item_matrix.columns)
EMBEDDING_SIZE = 32
BATCH_SIZE = 64
EPOCHS = 10
LEARNING_RATE = 0.001

# Split data into training and validation sets
train_size = int(0.8 * len(data))
train_data = data.iloc[:train_size]
val_data = data.iloc[train_size:]

# Define neural network model
input_user = tf.keras.layers.Input(shape=(1,))
embedding_user = tf.keras.layers.Embedding(input_dim=NUM_USERS, output_dim=EMBEDDING_SIZE)(input_user)
flatten_user = tf.keras.layers.Flatten()(embedding_user)

input_item = tf.keras.layers.Input(shape=(1,))
embedding_item = tf.keras.layers.Embedding(input_dim=NUM_ITEMS, output_dim=EMBEDDING_SIZE)(input_item)
flatten_item = tf.keras.layers.Flatten()(embedding_item)

dot_product = tf.keras.layers.Dot(axes=1)([flatten_user, flatten_item])
model = tf.keras.Model(inputs=[input_user, input_item], outputs=dot_product)

```

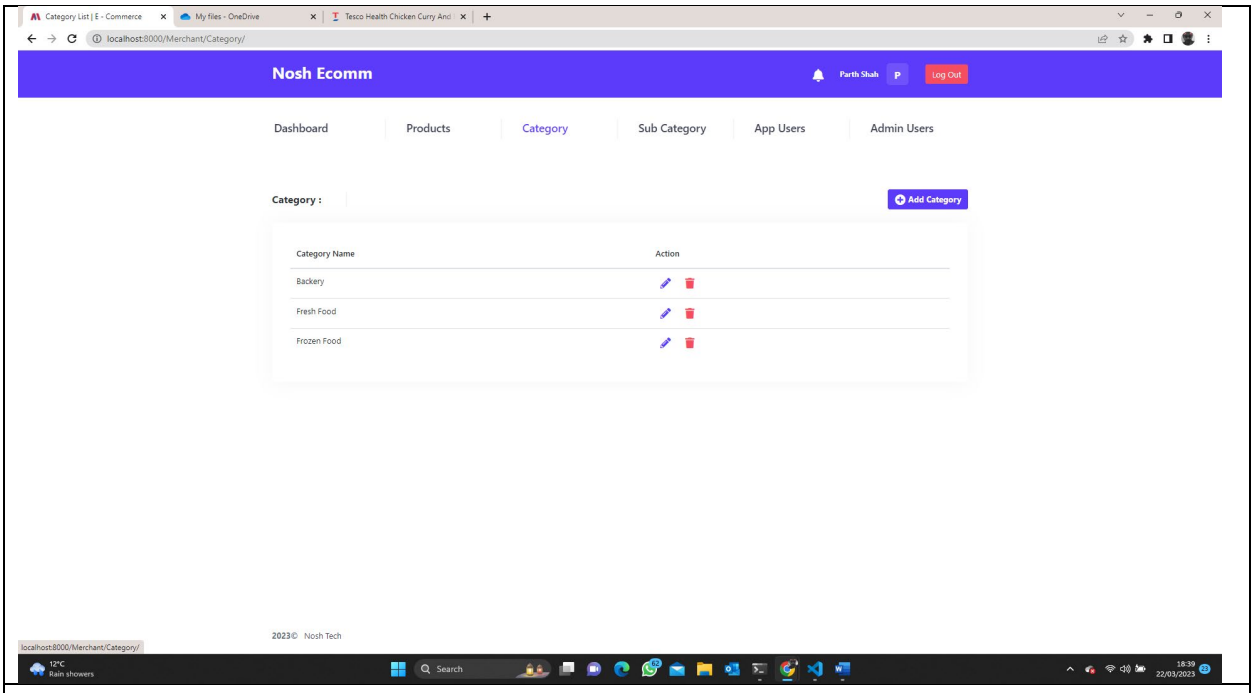
```python
# Compile model
model.compile(optimizer=tf.keras.optimizers.Adam(lr=LEARNING_RATE), loss='mean_squared_error')

# Define function to generate batches of training data
def generate_batches(data):
    num_batches = len(data) // BATCH_SIZE
    for i in range(num_batches):
        batch_data = data.iloc[i * BATCH_SIZE:(i + 1) * BATCH_SIZE]
        user_ids = np.array(batch_data['user_id'])
        item_ids = np.array(batch_data['product_id'])
        ratings = np.array(batch_data['quantity'])
        yield [user_ids, item_ids], ratings

# Train model
for epoch in range(EPOCHS):
    for batch in generate_batches(train_data):
        inputs, targets = batch
        model.train_on_batch(inputs, targets)
    # Evaluate model on validation set
    val_loss = model.evaluate([val_data['user_id'], val_data['product_id']], val_data['quantity'], verbose=0)
    print(f"Epoch {epoch+1}/{EPOCHS}: Validation loss = {val_loss:.4f}")
```

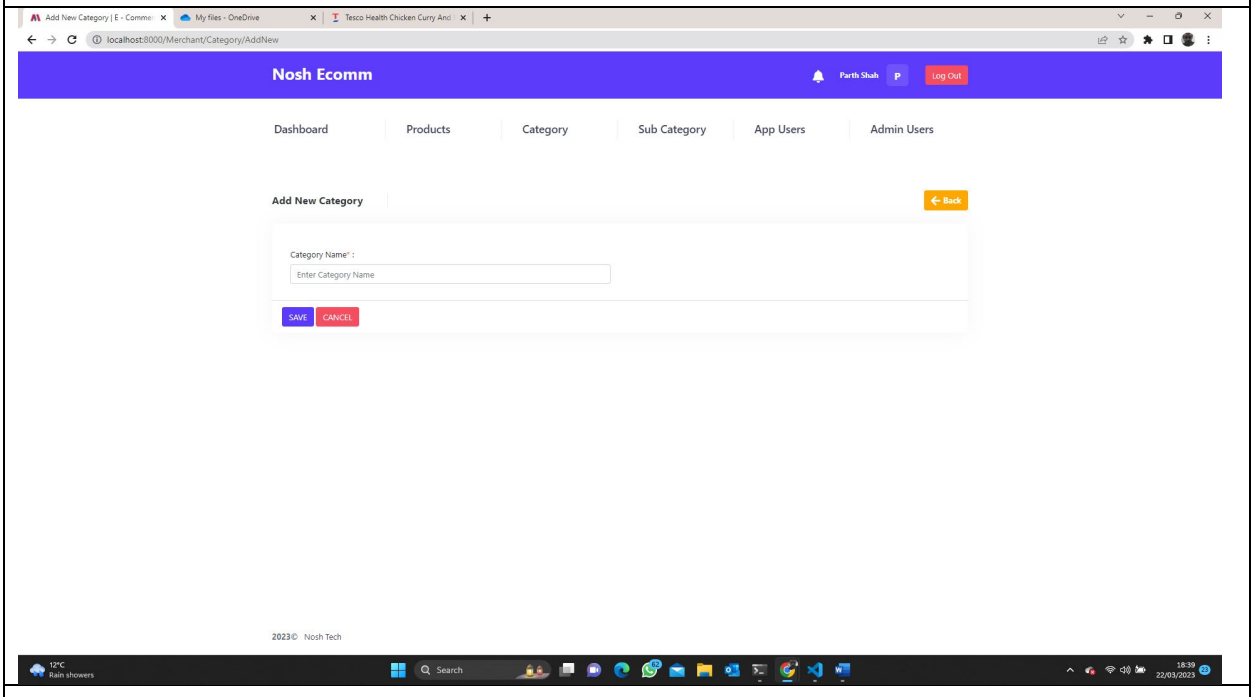


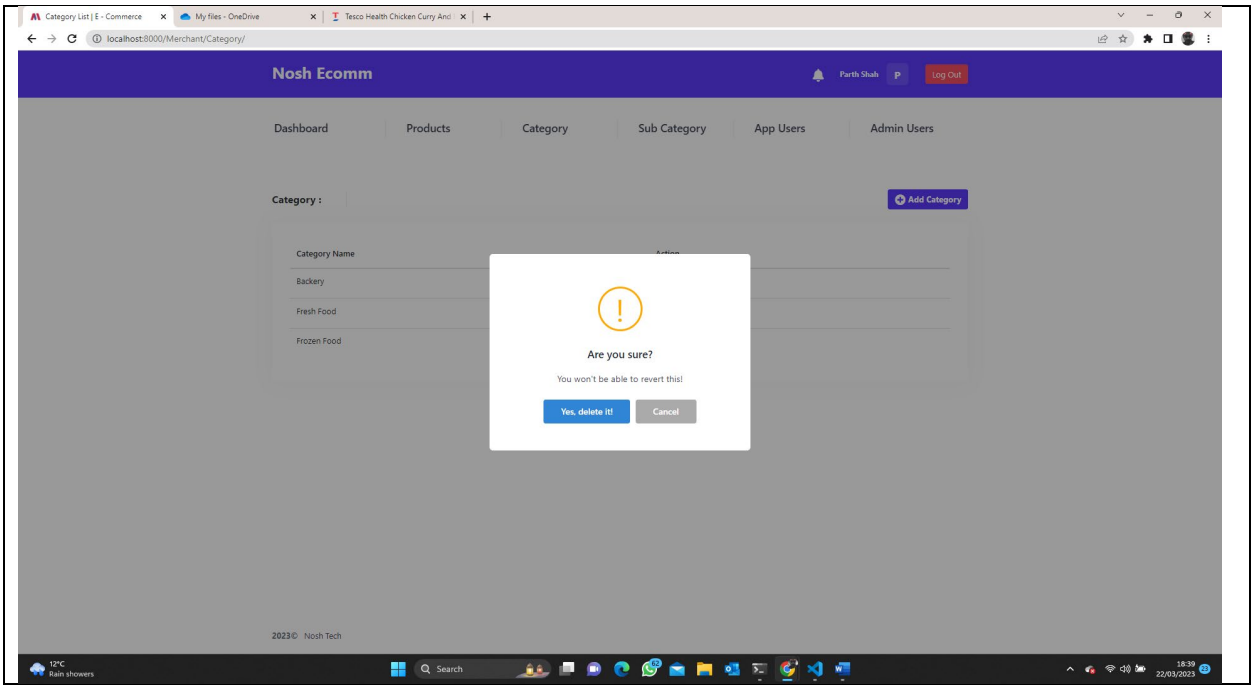
6.2.4 Implementation of WEB Interface





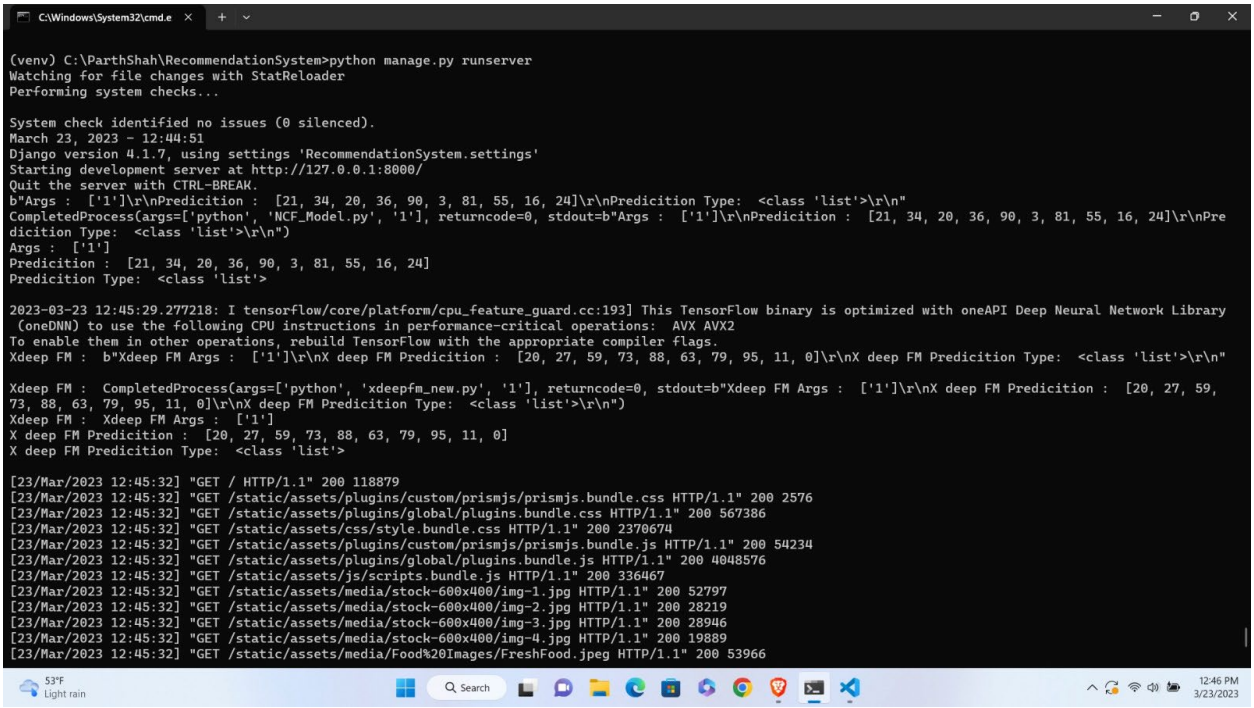

Nosh Ecomm
Parth Shah
P
Log Out
Dashboard
Products
Category
Sub Category
App Users
Admin Users
Category :
Add Category
Category Name
Action
Backery
Fresh Food
Frozen Food
2023© Nosh Tech


Nosh Ecomm
Parth Shah
P
Log Out
Dashboard
Products
Category
Sub Category
App Users
Admin Users
Add New Category
Back
Category Name* :
Enter Category Name
SAVE
CANCEL
2023© Nosh Tech



6.2.5 Final Output

We are integrating three models NCF, DQN, and XdeepFM for better and unique output. We are successful to integrate two models NCF and XdeepFm. As well for DQN, we got static value but it's not working for Dynamic Value. We try to solve this issue very Soon.



7 Software Testing and Risk Management

To guarantee the delivery of high-quality software products, software testing, and risk management are crucial elements of software development. Software testing is the process of examining a software program or system to find any mistakes, faults, or flaws that might impair its performance, usability, or functioning. The process of detecting, evaluating, and managing any risks that might influence a software project's success is known as risk management. As software bugs or errors can result in hazards that could have a detrimental impact on the project, software testing and risk management work hand in hand. For instance, system breakdowns, data loss, security breaches, and customer discontent may occur if a significant issue is not found and rectified. To guarantee that software products are of high quality and satisfy end-user expectations, it is essential to adopt a thorough testing and risk management approach.

The usage of a risk-based testing methodology is one of the most popular methods for managing risks in software testing. This strategy entails ranking testing efforts by the most important risks that might have an impact on the software project. To guarantee that the biggest risks are minimized, testing efforts should be concentrated on the software application's or system's most crucial components. The usage of automated testing tools is another strategy for risk management and software testing. These technologies can aid in lowering the possibility of human mistakes and enhancing the efficacy and efficiency of the testing procedure. A variety of software components, including functional requirements, performance, and security, may be tested via automated testing.

In conclusion, risk management and software testing are essential elements of software development that work to guarantee the delivery of high-quality software products. Software development teams may lower the likelihood of mistakes and defects, increase the quality of their work, and make sure their final products satisfy end users' needs and expectations by putting in place a thorough testing and risk management approach.

7.1 Neural Network-Based Recommendation System for Food Purchase Testing

It is essential to test a neural network-based recommendation system for food purchases to make sure it operates precisely and consistently when giving clients tailored recommendations. Testing may assist in identifying and resolving possible problems with the system's data input, processing, and output as well as with its general functionality and ability to satisfy end-user demands and preferences.

Using various testing techniques, such as functional testing, usability testing, and performance testing, is one way to evaluate a neural network-based recommendation system for food purchases. The functionality of the system is tested during functional testing, including its ability to forecast user preferences properly and produce useful recommendations. Usability testing evaluates the system's usability, including whether it has an intuitive user interface and can be used by users to navigate and interact with the system with ease. Performance testing evaluates the system's speed and scalability, checking that it can process a lot of requests and offer suggestions in real-time, for example. A neural network-based recommendation system for purchasing food may also be tested using a variety of testing methods, including black-box testing, white-box testing, and gray-box testing. Testing a system's inputs and outputs in a black-box environment includes doing so without being aware of how the system interprets the data. White box testing entails dissecting the system's internal operations and assessing its functionality in light of its code and structure. As used in

neural networks, white box testing may entail examining the network's weights and biases to make sure they are suitable for the job at hand. It could also entail assessing the activation functions and other network properties. In contrast, grey box testing combines elements of both black box and white box testing. In grey box testing, the tester gets partial access to the internal workings of the system. In the case of neural networks, for instance, the tester could have access to the network's design and the data preparation techniques employed, but not to the weights and biases. (Aebarsold, k. 2019)

7.2 Testing Process of Neural Network-Based Recommendation System for Food Purchase

It is crucial to test neural network-based recommendation systems for food purchases to make sure they are accurate and dependable and give clients recommendations that are pertinent to their needs. Training, validation, and testing are the three steps that commonly make up the testing process.

The training stage entails supplying the system with a big dataset of consumer information and food purchase history to train the neural network to recognize trends and provide reliable suggestions. The system is tested on a smaller dataset during the validation step to make sure it can generalize effectively and provide correct suggestions for new clients.

In the final testing phase, the system's performance is assessed using a fresh dataset that wasn't utilized during training or validation. This step attempts to assess the system's capability to provide clients with accurate and pertinent suggestions in real-world circumstances.

The effectiveness of neural network-based recommendation systems for food purchases may be assessed using a variety of measures. They include accuracy, precision, recall, and F1 score. Recall measures the proportion of relevant products that are recommended, whereas precision measures the proportion of recommended items that are relevant to the client. Accuracy quantifies the percentage of suggested items that are successfully chosen, while precision and recall are combined to get the F1 score.

8 Conclusion

In conclusion, this project highlights the application of neural networks in the development of a product suggestion web application that provides personalized product recommendations to users based on their browsing and purchase history. The essay outlines the practical project management approach adopted for the project, including the use of synthetic data to generate recommendations efficiently while addressing privacy concerns. The essay further discusses various machine learning algorithms, including MLP, LSTM, transformer, NCF, xDeepFM, and DQN, each with t own strengths and weaknesses, which were implemented in the project to improve recommendation accuracy. The project is implemented using Django as the front-end framework and Python as the back-end language, and the NCF, xDeepFM, and DQN models were trained using a dataset of user-item purchase information. Overall, the essay emphasizes the advantages of using neural networks in AI, such as their ability to handle complex data, model non-linear interaction ons, a scale for large datasets, personalize recommendations, and continuously learn and improve over time, and highlights the potential for neural networks to revolutionize the way businesses suggest products to their customers.

9 References

- *Welcome to Faker's documentation! — Faker 5.0.1 documentation*. (n.d.). Faker.readthedocs.io. https://faker.readthedocs.io/en/master/
- Yesmin, F. (n.d.). *How to Use Python Faker to Generate Dummy Data*. https://linuxhint.com/python-faker-generate-dummy-data/
- Nandakumar, S. (2022, March 29). *Faker library in python - An intriguing expedient for data scientists*. Medium. https://towardsdatascience.com/faker-library-in-python-an-intriguing-expedient-for-data-scientists-7dd06f953050
- Axelrod, A. (2021, June 21). *Why Use Synthetic Data vs Real Data?* Datomize. https://www.datomize.com/why-use-synthetic-data-versus-real-data/#:~:text=Synthetic%20data%20allows%20data%20scientists
- Naber, M. (2022, July 21). *The Advantages of Synthetic Data Over Real Data*. Neptune.ai. https://neptune.ai/blog/the-advantages-of-synthetic-data-over-real-data
- He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T. (2017). Neural Collaborative Filtering. *Proceedings of the 26th International Conference on World Wide Web*.
- Zhang, S., Yao, L., Sun, A., Tay, Y., Zhang, S., Yao, L., & Sun, A. (2017). Deep Learning based Recommender System: A Survey and New Perspectives. *ArXiv, abs/1707.07435*.
- Vaswani, A., Shazeer, N.M., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., & Polosukhin, I. (2017). Attention is All you Need. *ArXiv, abs/1706.03762*.
- Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W., & Jiang, P. (2019). BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*.
- Sharma, A. (2019, December 16). *eXtreme Deep Factorization Machine(xDeepFM)*. Medium. https://towardsdatascience.com/extreme-deep-factorization-machine-xdeepfm-1ba180a6de78
- Loy, J. (2022, August 25). *Deep Learning-based Recommender Systems*. Medium. https://towardsdatascience.com/deep-learning-based-recommender-systems-3d120201db7e
- Loni, B., Gu, Q., Sun, L., & Zhang, Z. (2019). A Survey on Deep Learning for Recommender Systems: Challenges and Remedies. *arXiv preprint arXiv:1906.04343*
- Chen, Haicheng. (2021). A DQN-based Recommender System for Item-list Recommendation. *5699-5702.* 10.1109/BigData52589.2021.9671947.
- Django homepage. http://www.djangoproject.com/.
- Python documentation. http://www.python.org/doc.
- Django(web framework). http://en.wikipedia.org/wiki/Django.
- Django documentation. http://docs.djangoproject.com.
- Aebersold, K. (2019). *Software Testing Methodologies*. Smartbear.com. https://smartbear.com/learn/automated-testing/software-testing-methodologies/