Personalized Deep Neural Network Application-Food Recommendation System

Submitted in partial fulfillment of the requirements

For the degree of

Bachelor of Technology

Computer Science and Engineering

Submitted By

Pratima Chinta (A704132519007)

Sanjana Paninjayath (A70405219005)

Gautam Reddy (A70405219029)

Under the Guidance of

Dr. Satheesh Abimannan,

Deputy Director & Professor, ASET

AMITY SCHOOL OF ENGINEERING AND TECHNOLOGY

AMITY UNIVERSITY MUMBAI

2019-23

Declaration of Academic Integrity

We declare that this written submission conveys our ideas in our own words. We have adequately cited and referenced the original sources. We also declare that we have adhered toall principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/date/fact/source in our submission.

We understand that any violation of the above will be cause for disciplinary action by the institute and they can evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Student Signature

Pratima Chinta, A704132519007
Sanjana Paninjayath, A70405219005
Gautam Reddy, A70405219029

DATE:

Approval

This is to certify that Ms. Pratima Chinta, Ms. Sanjana Paninjayanth and Mr. Gautam Reddy have satisfactorily completed their project (final stage) on "Personalized Deep Neural Network application- Food Recommendation System" during the academic term 2022-23 and their report is approved for final submission.

Examiners

………………………

……………………….

……………………….

Date:

Place:

CERTIFICATE

This is to certify that the project entitled "Personalized Deep Neural Network application- Food Recommendation System" is a bonafide work of Ms. Pratima Chinta, Ms. Sanjana Paninjayanth and Mr. Gautam Reddy submitted to the Amity School of Engineering and Technology, Amity University Mumbai in partial fulfilment of the requirement for the degree of B. Tech Computer Science & Engineering.

Supervisor Name

Dr. Deepa Parasar

Dr. Shrikant Charhate, Director, ASET

ACKNOWLEDGEMENT

This project was a great experience of learning and professional development.

We take this opportunity to express our deepest gratitude and special thanks to our mentor, Dr. Satheesh Abimannan, Deputy Director & Professor of Amity School of Engineering and Technology at Amity University, Mumbai. Despite being occupied with their schedule, our mentor took out the time to hear, guide and keep us on the correct path. His constant guidance and support throughout the course of this project have led to its timely and successful completion.

We would also like to express our deepest thanks to all the faculty members of Amity School of Engineering and Technology (ASET) for encouraging and motivating us throughout the B.Tech Program.

Lastly, we would like to express a wholehearted thanks to our family members, for supporting us with their love and guidance in whatever we pursue.

We perceive this opportunity as a big milestone in our career development. We shall strive to use the gained skills and knowledge in the best possible way and shall continue to work on their improvement, to attain the desired objectives.

ABSTRACT

Even though eating is a must, at times people are unsure of what to choose. In reality, there are too many alternatives while cooking or purchasing meals for one to consider them all. The nutritional needs of each individual are unique, as are their tastes. Because of this, knowing the individual is the only way to satisfy their requirements. The final decision will be influenced by the recommendation, regardless of whether it is for a simple hungry user, a cooking enthusiast, a health-conscious dieter, or a sick person wanting to improve his medical state.

Additionally, the product being promoted also has a significant effect: a straightforward ingredient substitution, recipe, dish, dining establishment, or even a cuisine. When the recommendation is made: immediately or in a newsletter. It could determine the user's location and recommend the closest locations. Therefore, a platform where suggestions can be made is necessary. The gadget may need different capabilities depending on the suggestions' generation method (collaborative filtering, content-based, neural networks, or embeddings). They can then be shared in plain text (SMS), on a website, or through an application. The source of the data is crucial for developing these systems. Data can be collected from previous purchases, user responses to posts (such as likes or dislikes), community ratings, viewed photos or videos, or other social network-related activities like posts, shares, searches, comments, or follows.

Food recommendation systems have been developed to assist users in making dietary choices based on their specific diets, recipes, and preferences and they aim to promote healthier eating habits that align with the user's individual preferences. However, many existing food recommendation systems do not take into account the health and nutritional aspects of food, limiting their ability to generate recommendations that prioritize user well-being.

To address this limitation, our project focuses on developing a food recommendation system that explicitly considers food ingredients and ratings. By incorporating time-aware collaborative filtering and a food ingredient content-based model, our model predicts user preferences and provides personalized recommendations. By making these adjustments, our food recommendation system ensures that it generates recommendations that are both unique to each user and promote a healthy diet.

TABLE OF CONTENTS

LIST OF FIGURES & IMAGES

CHAPTER 1: INTRODUCTION

1.1 About the project

The purpose of this project was to develop a recommendation system for meals or recipes based on the similarity of their ingredients as well as user ratings using the publicly available collection of recipe data.

Food recommendation systems are software applications or algorithms that offer personalized suggestions and recommendations for food and dining options to users. These systems utilize user preferences, historical data, and other relevant factors to provide tailored recommendations, with the goal of enhancing the user's dining experience. Food recommendation systems are available in various forms, such as mobile apps, websites, and online platforms, and they play a vital role in helping users make informed decisions about their food choices.

The primary objective of food recommendation systems is to streamline the overwhelming variety of food options and present users with relevant and appealing choices. By taking into account factors like dietary preferences, allergies, location, budget, and past behaviour, these systems strive to deliver recommendations that align with each user's unique preferences and requirements.

A fundamental aspect of food recommendation systems involves collecting and analyzing user data. This process entails gathering information about users' taste preferences, dietary restrictions, eating habits, and feedback on previously tried foods. The system then processes this data to build user profiles and generate accurate recommendations based on individual preferences. Data can be collected explicitly through surveys or questionnaires, or implicitly by analyzing user behavior and interactions within the system.

Collaborative filtering is a widely used technique in food recommendation systems. It involves analyzing the preferences and behavior of multiple users to identify patterns and make recommendations based on similar tastes. By comparing a user's preferences with those of other individuals who have similar profiles or expressed similar interests, the system can suggest food items or dining establishments that the user is likely to enjoy. Collaborative filtering can be further categorized into user-based filtering, which recommends based on similar users, and item-based filtering, which recommends based on similar food items or restaurants.

Content-based filtering is another technique employed by food recommendation systems. It involves analyzing the attributes and characteristics of food items and matching them with the user's preferences. For instance, if a user has a preference for vegetarian dishes, the system would recommend food options that align with this preference. Content-based filtering relies on item features such as ingredients, cuisine type, preparation methods, and nutritional information to make accurate recommendations. By combining these features with user preferences, the system can generate personalized suggestions that cater to the user's unique tastes.

Hybrid recommendation systems combine collaborative filtering and content-based filtering techniques to enhance the accuracy and quality of recommendations. By leveraging the strengths of both approaches, hybrid systems aim to overcome the limitations and challenges associated with each technique. These systems can provide more diverse and precise recommendations by considering both user preferences and item attributes simultaneously.

Contextual factors also play a crucial role in food recommendation systems. Contextual information includes variables such as the time of day, weather conditions, location, and social context. By considering these factors, the system can offer recommendations that are relevant to the user's current situation. For example, on a hot summer day, the system may suggest refreshing beverages or nearby ice cream shops. By incorporating contextual information, food recommendation systems can enhance the user experience and provide suggestions that are more timely and suitable.

Machine learning and artificial intelligence techniques can further enhance food recommendation systems. These technologies can analyze vast amounts of data and learn from user interactions to improve the accuracy and effectiveness of recommendations over time. By continuously adapting and updating their algorithms, these systems can refine their suggestions and deliver personalized recommendations that align with evolving user preferences.

Ethical considerations are crucial in the design and implementation of food recommendation systems. User privacy and data protection should be prioritized to ensure secure management and prevent misuse of user information. Transparency in data collection and utilization is also essential to build trust with users. Additionally, food recommendation systems should be mindful of potential biases in the data and algorithms to avoid reinforcing stereotypes or excluding certain demographics.

In conclusion, food recommendation systems are software applications or algorithms that utilize user data, preferences, and contextual factors to provide personalized suggestions and recommendations for food and dining.

1.2 Problem statement

This research study is aimed to develop a personalized food recommendation system based on food items having same or similar ingredients, it uses collaborative filtering and aims to use a deep neural network to predict the ratings of food items.

1.3 Objectives

1. To make a personalized food recommendation system which uses ingredients of food items and similarity measures to suggest food items that are similar and contain similar nutritional value to the users.
2. To make a recommendations of food items based on ratings of users collected in a dataset
3. To compare various methods: Content Base Filtering, Collaborative filtering with Machine Learning model called K-Nearest Neighbor and Neural Network for food recommendation.

CHAPTER 2: LITERATIVE SURVEY

2.1 Background

Fundamental Concepts:

1) Personalized systems:

Personalized systems refer to technological or computational systems that are designed to tailor their output or behaviour based on individual preferences, characteristics, or needs of users. These systems aim to provide a unique and customized experience to each user, taking into account their specific requirements, interests, or past interactions.

Personalized systems can be found in various domains and applications, including:

Personalized recommendations: Many online platforms, such as streaming services, e-commerce websites, and social media platforms, use personalized recommendation systems. These systems analyze a user's browsing history, purchase behavior, or social connections to suggest content, products, or services that are likely to be of interest to them.

2) Recommendation Systems:

Based on how recommendations are made, recommendation systems are usually classified into the following categories: Knowledge-based recommendation systems; Content-based recommendation systems; Collaborative recommendation systems; and Hybrid recommendation systems.

- Collaborative filtering is currently the most popular approach for developing recommendation systems. Collaborative methods focus more on rating-based recommendations.

- Content-based approaches, instead, relate more to classical Information Retrieval based methods and focus on keywords as content descriptors, to generate recommendations. Because of this, content-based methods are very popular when recommending documents, news articles or web pages, for example.

- Knowledge-based systems suggest products based on inferences about the user's needs and preferences.

3) Collaborative methods, or collaborative filtering systems:

They try to predict the utility of items for a particular user, based on the items previously rated by other users. This approach is also known as the wisdom of the crowd and assumes that users who had similar tastes in the past, will have similar tastes in the future. In order to better understand the users' tastes, or preferences, the system has to be given item ratings either implicitly or explicitly. Content-based and collaborative methods have many positive characteristics but also several limitations. The idea behind hybrid systems is to combine two or more different elements in order to avoid some shortcomings and even reach desirable properties not present in individual approaches.

4) Neural networks:

A computing model called a neural network is modelled after how the human brain functions and is organised. It is made up of layers of neurons, which are interconnected nodes. Through a process known as training, neural networks are made to learn patterns and relationships from data and then make predictions or judgements based on that knowledge.

Here is a general description of how a neural network functions:

• Input Layer: The input layer is where the initial data is received. This data can take any form, including text, graphics, or numerical numbers.

• Hidden Layers: There may be one or more hidden layers between the input and output layers. A hidden layer's neurons each compute a weighted sum of their inputs, apply an activation function, and generate an output. The network can model complicated interactions because of the activation function's non-linearity.

• Weight Modification: During training, the network modifies the weights corresponding to each neuronal link. Backpropagation, a technique used to determine the gradient of the network's error with respect to the weights, is the basis of this modification. The network develops the ability to reduce its prediction error by iteratively adjusting the weights.

• Output Layer: The network's prediction or output is created by the final layer, which is referred to as the output layer. The task's requirements determine how many neurons are present in this layer.

The prediction of ratings is one of several tasks for which neural networks are utilised. A neural network can be trained to predict a rating value given a collection of input features in the context of rating prediction. For instance, user preferences, movie genres, directors, or stars could all be input features in a recommendation system for movies. Through training on a tagged dataset, the network discovers the connections between these features and the related ratings.

During training, input samples with known ratings are given to the network. The prediction error is then calculated, and weights are changed to reduce the error. Once trained, the network may predict outcomes based on new, unobserved data by sending inputs through the network and receiving the projected rating as an output.

The ability of a neural network to predict ratings is influenced by a number of variables, including the calibre and representativeness of the training data, the network's architecture and size, the activation functions selected, and the optimisation algorithm for weight adjustment. Rating prediction performance can be enhanced by tweaking these elements.

5) PDNN to predict ratings of food items:

PDNN (Personalized Deep Neural Network) can be used to predict ratings of food items by leveraging user preferences and food features. Here's a general approach to using PDNN for this task:

1. Data Collection: Gather a dataset that includes user ratings of food items along with corresponding features of the food items. The features could include attributes such as ingredients, nutritional values, cuisine, preparation methods, or any other relevant information.
2. Data Pre-processing: Pre-process the dataset by normalizing the ratings and encoding categorical features. It's also important to split the data into training and testing sets to evaluate the performance of the model.
3. Model Architecture: Design a PDNN architecture suitable for the rating prediction task. The architecture typically includes an embedding layer to represent categorical features, one or more hidden layers, and an output layer that produces the predicted rating. The number of neurons and layers can vary based on the complexity of the dataset and the desired model capacity.
4. Training: Train the PDNN model using the training set. During training, the model learns to capture the relationships between the food features and the corresponding

ratings. This is done by minimizing the prediction error using optimization algorithms such as stochastic gradient descent (SGD) or Adam.

5. Validation: Monitor the model's performance on a validation set to ensure it is not overfitting or underfitting the data. Adjust hyperparameters, such as learning rate or regularization strength, if needed to improve performance.
6. Testing and Evaluation: Evaluate the trained PDNN model on the testing set to assess its ability to predict ratings for unseen food items. Calculate evaluation metrics such as mean squared error (MSE) or root mean squared error (RMSE) to quantify the prediction accuracy.
7. Prediction: Once the PDNN model is trained and evaluated, it can be used to predict ratings for new food items. Provide the relevant features of the food item as input to the model, and obtain the predicted rating as the output.
8. Personalization: To make the predictions personalized, incorporate user preferences into the model. This can be achieved by including user-specific features or by training separate models for each user. The personalized information can help the PDNN model adapt its predictions based on the preferences and past ratings of individual users.

6) Collaborative filtering’s Working

By using the actions of people who share characteristics with the users, collaborative filtering determines how similar users are to one another. In employing the food recommendation system based on collaborative filtering, users' data are determined as a database in working on prediction and recommendation of lists to users in accordance, which can be divided into 2 parts as follows:

i) Calculating the degree to which users are similar: To determine whether new and old user ratings are similar based on the correlation, this approach will compute Pearson's correlation coefficient:

$$s(u,v) = \frac{\sum i \in I_u \cap I_v\ (r_{u,i} - r_u)(r_{v,i} - r_v)}{\sqrt{\sum i \in I_u \cap I_v (r_{u,i} - r_u)^2}\ \sqrt{\sum i \in I_u \cap I_v (r_{v,i} - r_v)^2}}$$

where:	$s(u,v)$	refers to similarity between new users and old users.
	$I_u \cap I_v$	refers to similarity of the rating given by new users and old users.
	$r_{u',i}$	refers to the rating of new users to the lists.
	$\bar{r}_{u'}$	refers to the average rating given by new users.
	$r_{v,i}$	refers to the rating of old users to the lists.
	$\bar{r}_v$	refers to the average rating given by old users.

Fig 1: Pearson's correlation coefficient [7]

Calculated values were employed to ascertain the statistical relationship between the scores of new and veteran users.

ii) Calculating to figure out prediction value: In terms of prediction or recommendation for new users based on the collaborative filtering between new and old users, a computation is done to determine the new users' closest value provided to the lists that match those of the old users. The system then rates user scores for anticipating new users after the calculation is complete.

2.2 Existing methodologies

Paper 1) Food Proposal System for Diabetic Patients Using Clustering Examination.

In the context of nutrition and dietary features, this method will suggest the appropriate replacement foods. Nutrition is the main factor in diabetic diet management and control. The classification system in place, however, is ineffective for identifying the dietary category for diabetes patients. The ontology has been proposed for previous study on diabetic diet care and automated food mechanisms, but findings have not been able to provide fair food categories. They used the SOM method coupled with K-mean clustering in the research to show the next step in classification. Contrary to previous study, the SOM algorithm will classify foods by taking eight important nutrients into account as the primary characteristics that affect diabetes patients.

Paper 2) A context-aware food recommendation system

This suggested system gets real-time access to a user's profile, physiological signals, and surrounding environmental data. In this study, they introduced a new method, which uses each individual's profile, physiological signals, and perceived ambient data to recommend appropriate meals in real time to each individual. The benefit of the suggested system is that by fusing client settings with other settings at run time and translating these settings in a client centric approach, it enables the management of personalised health care. More specifically, the suggested approach may assess a person's wellbeing based on their unique circumstances (such as physiological state, eating habits, climate, locality, etc.). This is especially useful for persons who have a specific clinical history.

Paper 3) A Food Recommender system considering nutritional information and user Preferences

This paper's objective is to offer a comprehensive solution for this architecture's layer of intelligent systems. This method combines a short-term intelligent model based on an optimisation scenario that takes into account both nutritional and preference-aware information with the nutritional context determination based on an MCDA technique for filtering out improper meals. In the current study, a food recommendation methodology that creates daily personalised meal plans for individuals based on their dietary requirements and past food choices is provided. A review of the most important recent related research reveals that while some studies have been focused on developing computational tools for dietary admittance advising, the majority of them don't precisely address both customer preferences and wholesome data.

Paper 4) The Utilize of Machine Learning Calculations in Recommender Systems:

Based on information about the user or the proposed item, recommender systems (RS) are used to help customers find underutilised goods or services, such as books, music, transportation, or even people.

Recommender systems (RS) are widely used in social networks, e-commerce, and a few other fields. Since its introduction in the middle of the 1990s, research on RSs has advanced. The use of machine learning (ML) computations, which allow computers to memorise based on client data and to personalise suggestions in advance, is a dynamic development in the history of RS. Machine learning is an Artificial Intelligence (AI) investigation into a sector that uses mathematics to predict the outcome of information preparation. Major advances in ML have been achieved in the fields of security, look engines, and image recognition. However, a few computations from the ML field that have altered features are shown in the literature. A categorization framework is required in the writing so that computations can occur in the contexts in which they are most relevant. Therefore, selecting an ML computation to be employed in RSs may be a challenging task. Additionally, researchers in RSs lack a clear understanding of the trends in the use of ML calculations, making it difficult for them to decide where to focus their research efforts. Software development (SE) is a profession that considers the evolution of computer programmes from conception through usage and maintenance.

Paper 5) Profound Learning based Recommender System: A Study and Modern Perspectives.

This article provides a detailed audit of subsequent enquiries regarding reports on recommender systems based on deep learning. One of the most remarkable efforts on deep learning-based recommender systems to date. They highlighted many inertial research prototypes and presented a categorization approach for grouping and organising existing literature. They also discussed the benefits and drawbacks of applying advanced learning techniques to suggestion tasks.

2.2 Comparative analysis

In the research papers, a Multiple criteria decision analysis approach was utilized to determine the nutritional context and filter out unsuitable food items. However, our project made use of PDNN methodology which employs user preferences and relevant food features to predict the ratings of various food items. By leveraging this approach, we can better understand how different food characteristics and user choices influence food ratings.

CHAPTER 3: PROPOSED SYSTEM

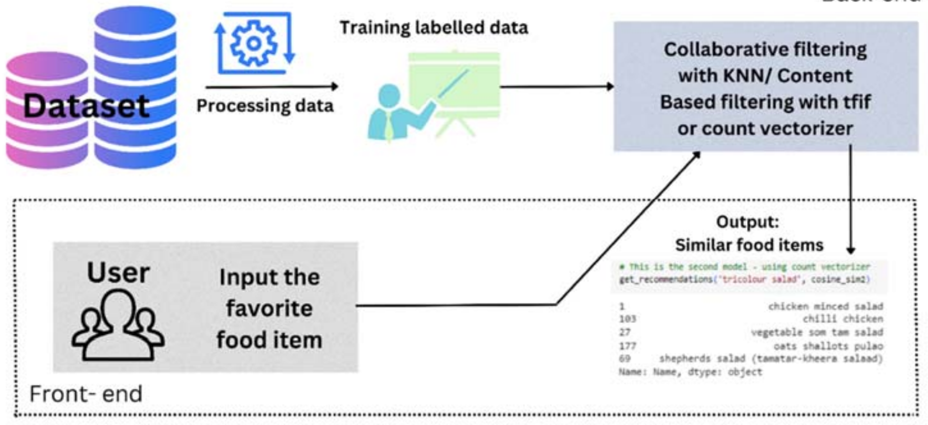
4.1 Methodology

To provide food recommendations, we begin by collecting food items data and processing it to create noise-free labelled data. We then employ content based filtering or collaborative filtering with k-nearest neighbours (KNN) algorithm to generate recommendations. To personalize recommendations, we gather input from users and incorporate it into the trained model. From the model, we suggest similar food items to the user. During the data collection stage, we gather data from various sources such as online recipe websites, cookbooks, and professional chefs. The data is then processed by removing any irrelevant information and transforming it into a structured format that can be used effectively for training.

4.2. System Design and Architecture

In the below diagram we can see the dataset containing the ingredients and food items is taken and processed and Algorithm either content-based similarity measures or collaborative filtering with K-Nearest Neighbours algorithm (to be explained in implementation part) is used to generate recommendations to the users. For two types of content-based methods: with TF-IDF (Term Frequency - Inverse Document Frequency) vector transform and count vectorizer, we are using only ingredients as the features. For collaborative filtering we are used both ingredients and ratings by the users. This is done in python i.e jupyter notebook or colab notebook.

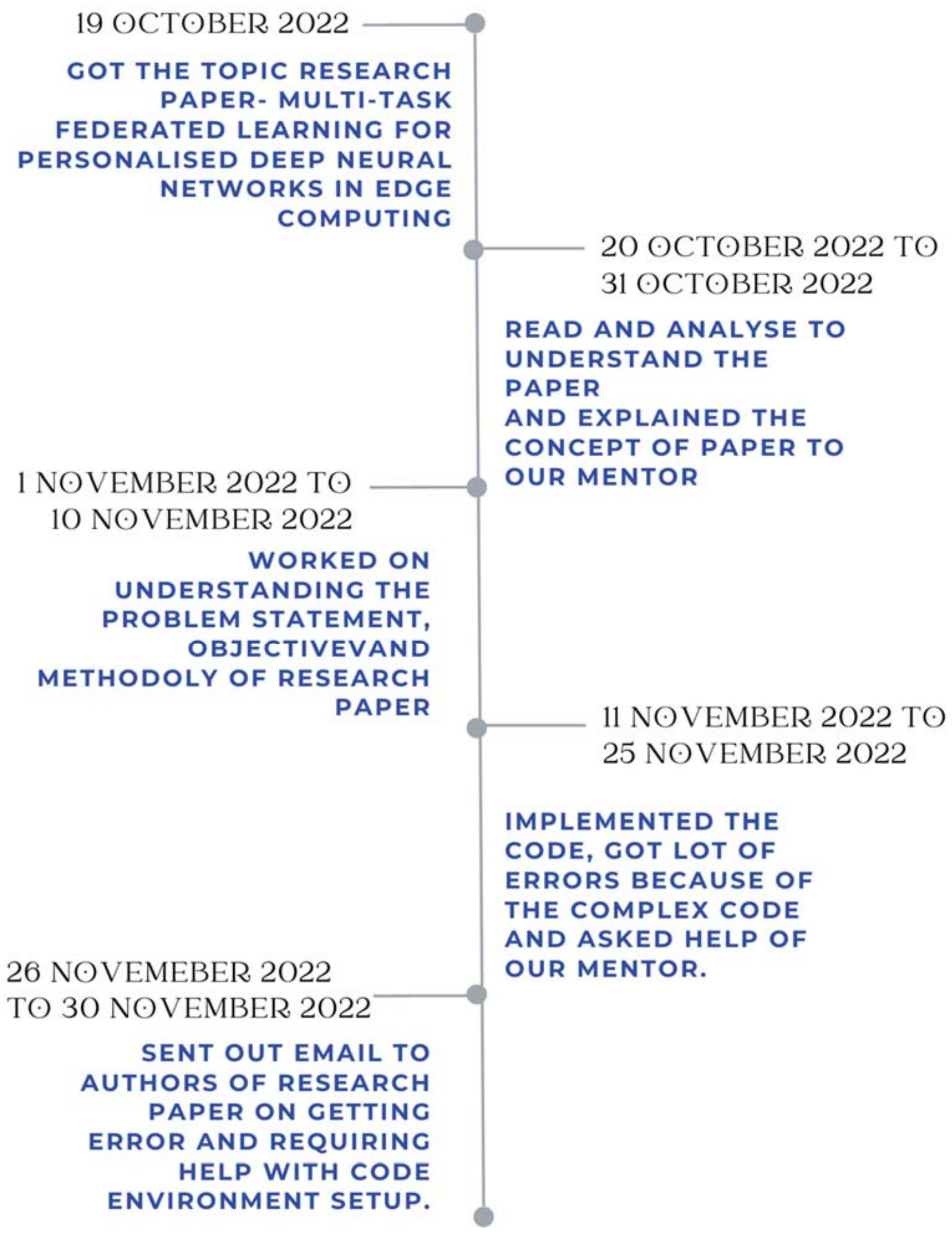
Another method is used where a deep neural network model is made using keras in python and a labelled ratings dataset is used to predict ratings of a food item and that can be further used to make recommendations using its embeddings (explained in implementation part).


Back-end
Dataset
Processing data
Training labelled data
Collaborative filtering with KNN/ Content Based filtering with tfif or count vectorizer
Output: Similar food items
User
Input the favorite food item
Front- end

```
# This is the second model - using count vectorizer
get_recommendations('tricolour salad', cosine_sim2)

1                             chicken minced salad
103                                 chilli chicken
27                         vegetable som tam salad
177                             oats shallots pulao
69     shepherds salad (tamatar-kheera salaad)
Name: Name, dtype: object
```

Pg-11) Fig 2: Architecture of food recommendation system

CHAPTER 4: PROJECT TIMELINE



19 OCTOBER 2022

GOT THE TOPIC RESEARCH PAPER- MULTI-TASK FEDERATED LEARNING FOR PERSONALISED DEEP NEURAL NETWORKS IN EDGE COMPUTING

20 OCTOBER 2022 TO 31 OCTOBER 2022

READ AND ANALYSE TO UNDERSTAND THE PAPER
AND EXPLAINED THE CONCEPT OF PAPER TO OUR MENTOR

1 NOVEMBER 2022 TO 10 NOVEMBER 2022

WORKED ON UNDERSTANDING THE PROBLEM STATEMENT, OBJECTIVEVAND METHODOLY OF RESEARCH PAPER

11 NOVEMBER 2022 TO 25 NOVEMBER 2022

IMPLEMENTED THE CODE, GOT LOT OF ERRORS BECAUSE OF THE COMPLEX CODE AND ASKED HELP OF OUR MENTOR.

26 NOVEMEBER 2022 TO 30 NOVEMBER 2022

SENT OUT EMAIL TO AUTHORS OF RESEARCH PAPER ON GETTING ERROR AND REQUIRING HELP WITH CODE ENVIRONMENT SETUP.

1 DECEMBER 2022 TO
23 DECEMBER 2022

SHOWED PROGRESS TO SIR
RESEARCHED ON FEDERATED LEARNING, PERSONALIZED FEDERATED LEARNING, MULTITASK FEDERATED LEARNING

24 DECEMBER TO 20
JANUARY 2023

WORKED ON MAKING THE REPORT AND RESEARCHED THE LITERATURE OF THE PAPER TOPIC

21 JANUARY TO 25
FEBRUARY 2023

SHOWED PROGRESS TO OUR MENTOR AND SEARCHED FOR INDUSTRIAL EXPERTS ON LINKEDIN TO HELP US UNDERSTAND THE TOPICS
IMPLEMENTED THE CODE, STILL GOT ERRORS

26 FEBRUARY TO
18 MARCH 2023

RESEARCHED ON TOPICS- EDGE COMPUTING, PERSONALIZED DEEP NEURAL NETWORKS
GAVE PRESENTATION ON THE SAME TOPICS AND PROGRESS SO FAR

19 MARCH TO
29 APRIL 2023

DEBBUGING THE CODE SEVERAL TIMES AND WERE FINALLY ABLE TO RUN THE CODE SUCCESSFULLY
SHOWED PROGRESS TO SIR

30 APRIL TO
16 MAY 2023

WORKED ON MAKING PRESENTATION AND REPORT
GAVE PRESENTATION FOR REVIEW AND PROGRESS SO FAR

17 MAY TO 20 MAY 2023

OUR MENTOR ASKED US TO CHOOSE A DOMAIN, WE CHOSE APPLICATIONS OF PDNN, AND GAVE US THE TASK TO IMPLEMENT A MODULE OF THE PROJECT AND WE DECIDED ON THE TOPIC-FOOD RECOMMENDATION SYSTEM USING PDNN

21 MAY TO 6 JUNE 2023

PREPARED A PROPOSAL OF THE TOPIC AND GAVE PRESENTATION TO OUR MENTOR AND STARTED THE WORK ON CODE(BACKEND) OF THE PROJECT AND ALSO DID THE RESEARCH ON PROJECT REPORT

7 JUNE– 9 JUNE 2023

FINISHED THE REPORT AND SHOWED THE REPORT TO OUR MENTOR FOR ANY CHANGES REQUIRED.

CHAPTER 5: IMPLEMENTATIONS, RESULTS AND DISCUSSION

5.1 Problem at hand

There are many complex methods being used for food recommendation. We implemented a simpler model that doesn't use a lot of processing time making the responses be available quicker. The main problem is the source of data. Here we have used publicly available data but in the future scope of this project if we extract data from various websites, it would include more variety of food items respective to different cuisines. Accuracy of each method used is also judged.

5.2 Methods used and Results

Software requirements:

Python version>3, numpy, pandas, Sklearn, seaborn, matplotlib, keras for tensorflow

Hardware Specification:

Operating System – Windows 7,8, 10 or Professional editions

Processor – dual core 2.4 GHz+(i5 or i7 series Intel processor or equivalent AMD)

RAM – 8GB

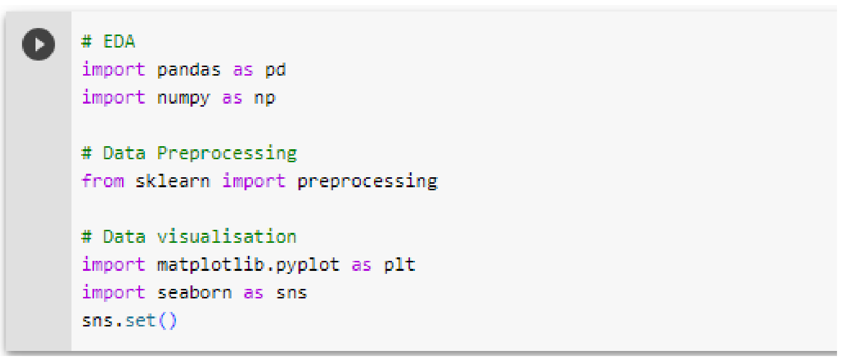
Hard Drive – 256 GB or larger solid state hard drives

Dataset used:

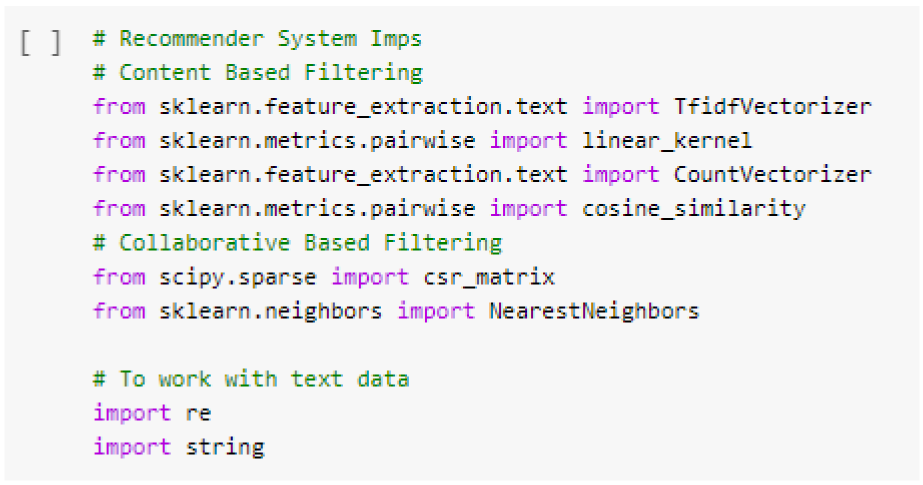
The data for the food recommendation system are represented in this dataset. This dataset file contains two datasets. The first dataset consists of information about the involved foods, ingredients, and cuisines. The second dataset contains the dataset for the recommendation system's rating system.

Steps followed:

1) Importing Libraries:



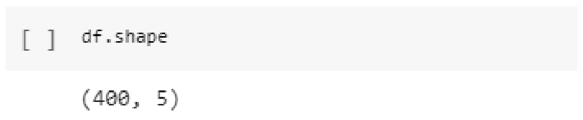
```
# EDA
import pandas as pd
import numpy as np

# Data Preprocessing
from sklearn import preprocessing

# Data visualisation
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
```



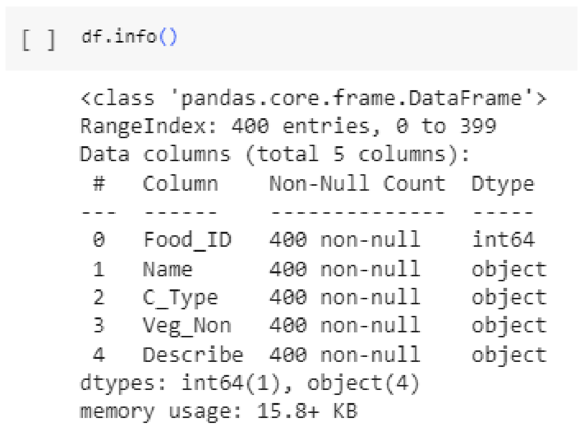
```
[ ]  # Recommender System Imps
     # Content Based Filtering
     from sklearn.feature_extraction.text import TfidfVectorizer
     from sklearn.metrics.pairwise import linear_kernel
     from sklearn.feature_extraction.text import CountVectorizer
     from sklearn.metrics.pairwise import cosine_similarity
     # Collaborative Based Filtering
     from scipy.sparse import csr_matrix
     from sklearn.neighbors import NearestNeighbors

     # To work with text data
     import re
     import string
```

2) Importing dataset using pd.read_csv

Dataset Information:



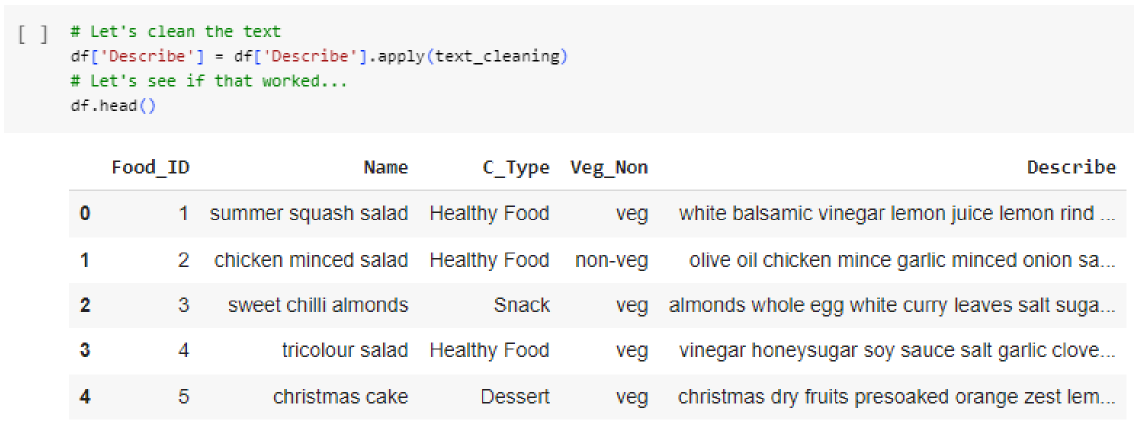
```
[ ]  df.shape

     (400, 5)
```



```
[ ]  df.info()

     <class 'pandas.core.frame.DataFrame'>
     RangeIndex: 400 entries, 0 to 399
     Data columns (total 5 columns):
      #   Column    Non-Null Count  Dtype 
     ---  ------    --------------  ----- 
      0   Food_ID   400 non-null    int64 
      1   Name      400 non-null    object
      2   C_Type    400 non-null    object
      3   Veg_Non   400 non-null    object
      4   Describe  400 non-null    object
     dtypes: int64(1), object(4)
     memory usage: 15.8+ KB
```

3) Data Cleaning and Pre-processing: Removing punctuations:

Pre-processing data can increase the accuracy and quality of a dataset, making it more dependable by removing missing or inconsistent data values brought on by human or computer mistake. It ensures consistency in data.



```
[ ] # Let's clean the text
    df['Describe'] = df['Describe'].apply(text_cleaning)
    # Let's see if that worked...
    df.head()
```

	Food_ID	Name	C_Type	Veg_Non	Describe
0	1	summer squash salad	Healthy Food	veg	white balsamic vinegar lemon juice lemon rind ...
1	2	chicken minced salad	Healthy Food	non-veg	olive oil chicken mince garlic minced onion sa...
2	3	sweet chilli almonds	Snack	veg	almonds whole egg white curry leaves salt suga...
3	4	tricolour salad	Healthy Food	veg	vinegar honeysugar soy sauce salt garlic clove...
4	5	christmas cake	Dessert	veg	christmas dry fruits presoaked orange zest lem...

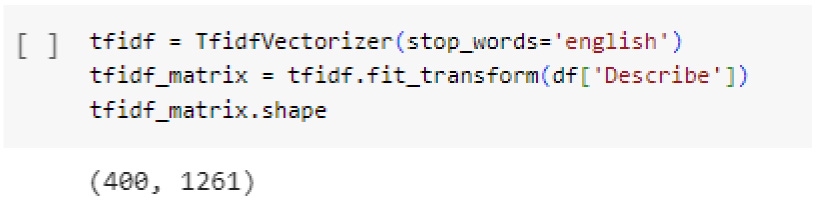
Checking for any duplicate or null values is also done.

4) Using Simple Content Based Filtering using Tfidf and linear_kernel function

What is TF-IDF?

Some words, like "the," "a," and "is" in English, will appear frequently in a huge text corpus, providing very little useful information about the substance of the text as a whole. These very common terms would overshadow the frequencies of less common but more intriguing terms if we were to send the direct count data directly to a classifier.

It is quite usual to employ the Tf-idf transform to re-weight the count features into floating point values appropriate for use by a classifier.

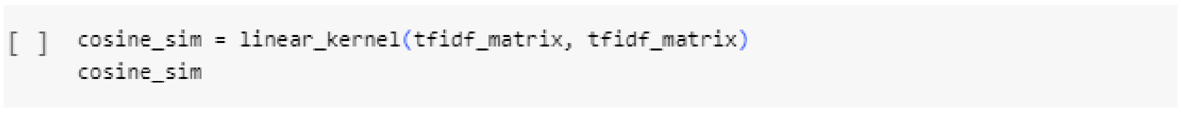


```
[ ] tfidf = TfidfVectorizer(stop_words='english')
    tfidf_matrix = tfidf.fit_transform(df['Describe'])
    tfidf_matrix.shape

    (400, 1261)
```

What is Linear Kernel?

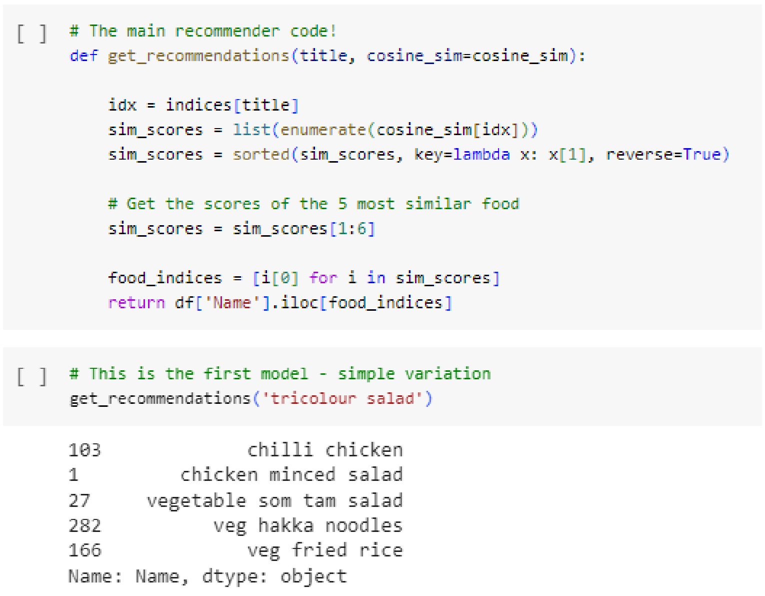
Kernels are measurements of similarity, so if items a and b are thought to be "more similar" to objects a and c, then s(a, b) > s(a, c). Additionally, a kernel must be positively semi-definite. The linear kernel, which is a special instance of the polynomial_kernel with degree=1 and coef0=0 (homogeneous), is computed using the linear_kernel function.

As a result, the following code is used to determine whether two food items' constituents are similar:



```
[ ]  cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)
     cosine_sim
```

5) <u>Results:</u> Recommender model and getting recommendations:



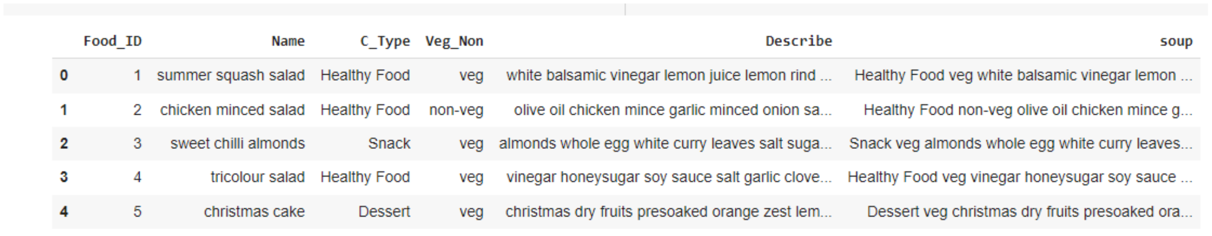
```
[ ]  # The main recommender code!
     def get_recommendations(title, cosine_sim=cosine_sim):

         idx = indices[title]
         sim_scores = list(enumerate(cosine_sim[idx]))
         sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)

         # Get the scores of the 5 most similar food
         sim_scores = sim_scores[1:6]

         food_indices = [i[0] for i in sim_scores]
         return df['Name'].iloc[food_indices]
```

```
[ ]  # This is the first model - simple variation
     get_recommendations('tricolour salad')
```

```
103             chilli chicken
1          chicken minced salad
27      vegetable som tam salad
282           veg hakka noodles
166              veg fried rice
Name: Name, dtype: object
```

6) <u>Using Advanced Content Based Filtering: Count vectorizer and cosine_similarity function:</u>

Make a column that will have all important features:



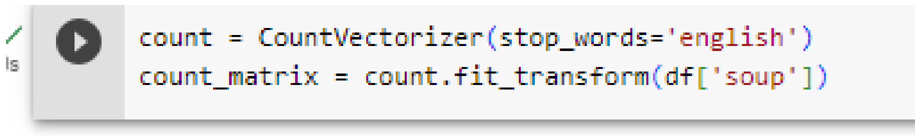
	Food_ID	Name	C_Type	Veg_Non	Describe	soup
0	1	summer squash salad	Healthy Food	veg	white balsamic vinegar lemon juice lemon rind ...	Healthy Food veg white balsamic vinegar lemon ...
1	2	chicken minced salad	Healthy Food	non-veg	olive oil chicken mince garlic minced onion sa...	Healthy Food non-veg olive oil chicken mince g...
2	3	sweet chilli almonds	Snack	veg	almonds whole egg white curry leaves salt suga...	Snack veg almonds whole egg white curry leaves...
3	4	tricolour salad	Healthy Food	veg	vinegar honeysugar soy sauce salt garlic clove...	Healthy Food veg vinegar honeysugar soy sauce ...
4	5	christmas cake	Dessert	veg	christmas dry fruits presoaked orange zest lem...	Dessert veg christmas dry fruits presoaked ora...

What is Count Vectorizer?

Create a token count matrix out of a group of text documents. Using scipy.sparse.csr_matrix, this implementation creates a sparse representation of the counts.

How does fit_transform work?

Explore the document-term matrix and the vocabulary dictionary.



```
count = CountVectorizer(stop_words='english')
count_matrix = count.fit_transform(df['soup'])
```

Regardless of the size of the documents, cosine similarity is a statistic used to determine how similar they are. It calculates the cosine of the angle formed by two vectors that are projected onto a multidimensional space.

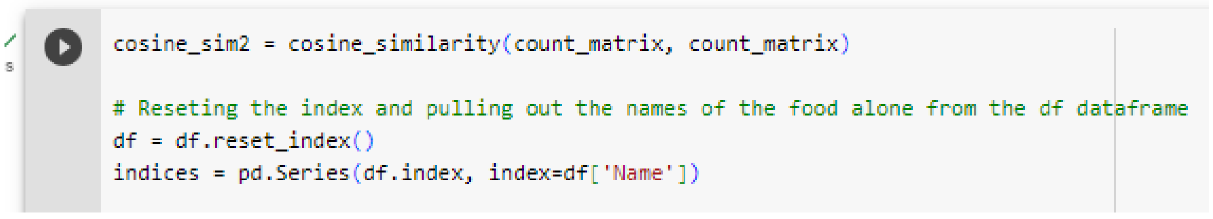
The two vectors in this situation are arrays that hold the word counts of two different documents. The cosine similarity captures the orientation (the angle) but not the magnitude of the documents when plotted on a multi-dimensional space where each dimension represents a word in the document. Calculate the Euclidean distance if you're after the magnitude.

If the word "cricket" appeared 50 times in one document and 10 times in another, for example, yet the two comparable texts are far away by the Euclidean distance due to size, they could still have a lower angle between them because to the cosine similarity. The resemblance increases with decreasing angle.

What is cosine_similarity?

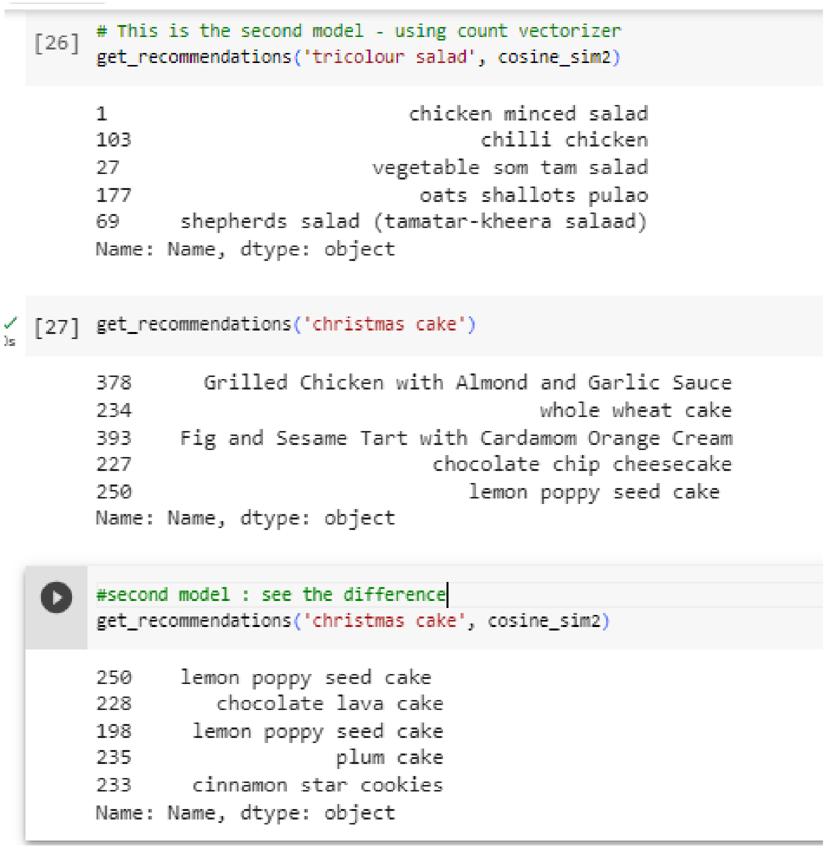
The function computes cosine similarity between samples in X and Y.

cosine_similarity computes the L2-normalized dot product of vectors. This is called cosine similarity, because Euclidean (L2) normalization projects the vectors onto the unit sphere, and their dot product is then the cosine of the angle between the points denoted by the vectors.



```
cosine_sim2 = cosine_similarity(count_matrix, count_matrix)

# Reseting the index and pulling out the names of the food alone from the df dataframe
df = df.reset_index()
indices = pd.Series(df.index, index=df['Name'])
```

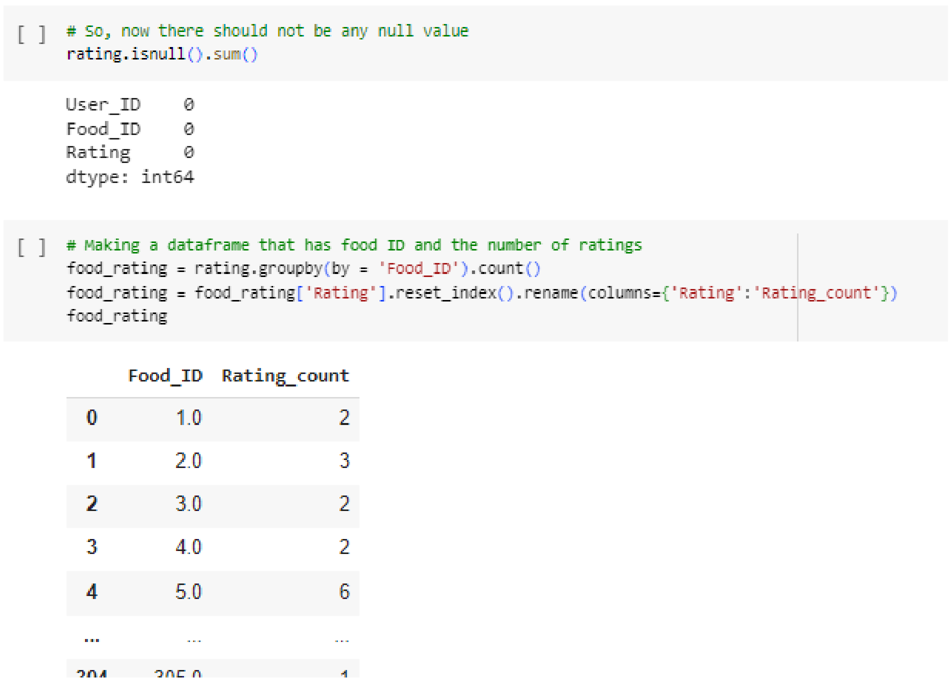
7) Results: Recommending using advanced content based filtering



```
[26] # This is the second model - using count vectorizer
     get_recommendations('tricolour salad', cosine_sim2)

     1                          chicken minced salad
     103                              chilli chicken
     27                        vegetable som tam salad
     177                           oats shallots pulao
     69      shepherds salad (tamatar-kheera salaad)
     Name: Name, dtype: object

[27] get_recommendations('christmas cake')

     378      Grilled Chicken with Almond and Garlic Sauce
     234                                  whole wheat cake
     393    Fig and Sesame Tart with Cardamom Orange Cream
     227                           chocolate chip cheesecake
     250                                lemon poppy seed cake
     Name: Name, dtype: object

     #second model : see the difference
     get_recommendations('christmas cake', cosine_sim2)

     250    lemon poppy seed cake
     228       chocolate lava cake
     198     lemon poppy seed cake
     235                 plum cake
     233     cinnamon star cookies
     Name: Name, dtype: object
```

8) Using Collaborative Filtering and KNN:

Import the raings.csv file and do pre-processing i.e remove rows that are not needed if they are empty.



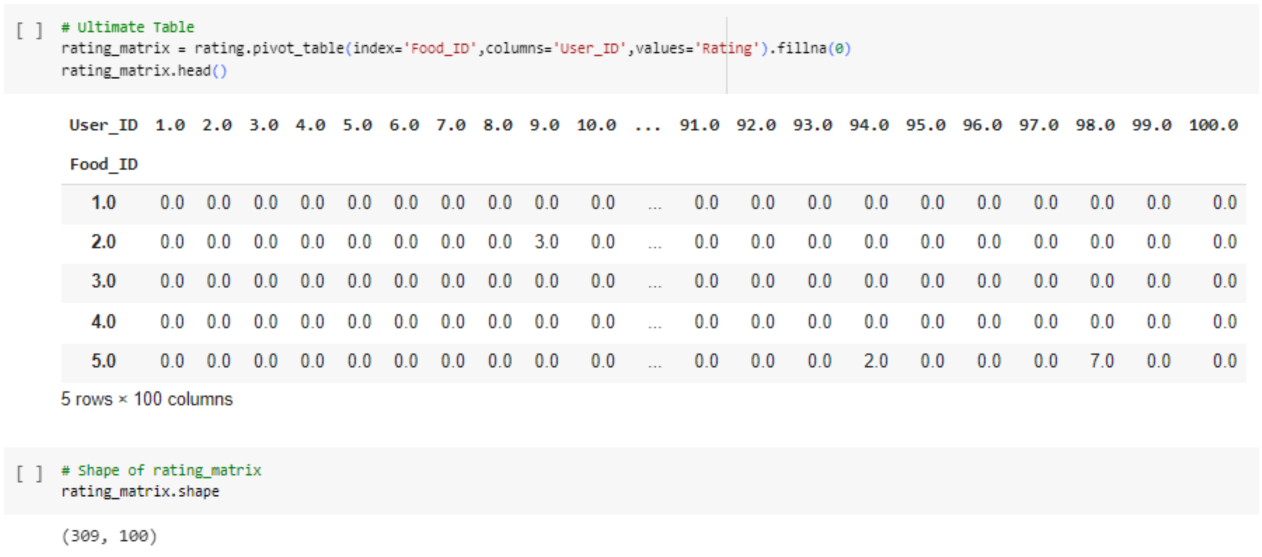
```
[ ] # So, now there should not be any null value
    rating.isnull().sum()
```

```
User_ID    0
Food_ID    0
Rating     0
dtype: int64
```

```
[ ] # Making a dataframe that has food ID and the number of ratings
    food_rating = rating.groupby(by = 'Food_ID').count()
    food_rating = food_rating['Rating'].reset_index().rename(columns={'Rating':'Rating_count'})
    food_rating
```

	Food_ID	Rating_count
0	1.0	2
1	2.0	3
2	3.0	2
3	4.0	2
4	5.0	6
...	...	...
204	205.0	1

Make user and rating per user dataframe and make a rating matrix as shown:



```
[ ] # Ultimate Table
    rating_matrix = rating.pivot_table(index='Food_ID',columns='User_ID',values='Rating').fillna(0)
    rating_matrix.head()
```

User_ID Food_ID	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0	10.0	...	91.0	92.0	93.0	94.0	95.0	96.0	97.0	98.0	99.0	100.0
1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	2.0	0.0	0.0	0.0	7.0	0.0	0.0

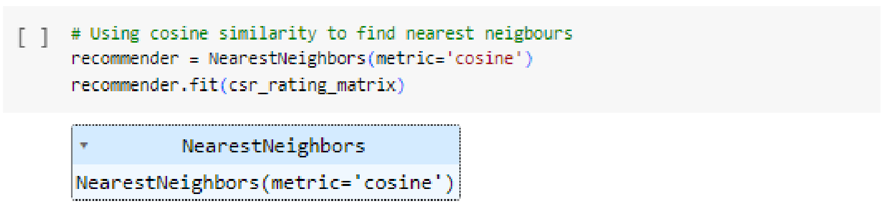
5 rows × 100 columns

```
[ ] # Shape of rating_matrix
    rating_matrix.shape
```

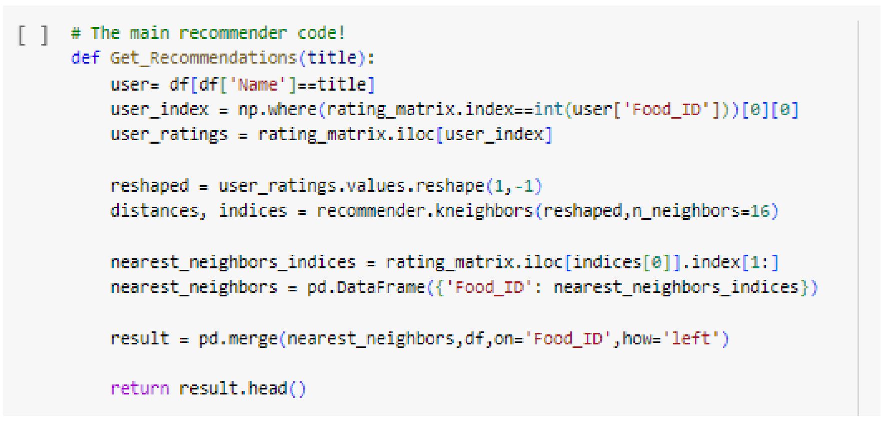
```
(309, 100)
```

Now Using Nearest Neighbours model:

Many other learning techniques, such spectral clustering and manifold learning, are built on the foundation of unsupervised nearest neighbours. The idea behind nearest neighbour methods is to select a set number of training samples that are closest in distance to the new point and then estimate the label based on them. In the case of radius-based neighbour learning, the number of samples can either vary depending on the local density of points or be a user-defined constant (k-nearest neighbour learning). In general, the distance can be measured in any metric unit; the most popular option is the conventional Euclidean distance.

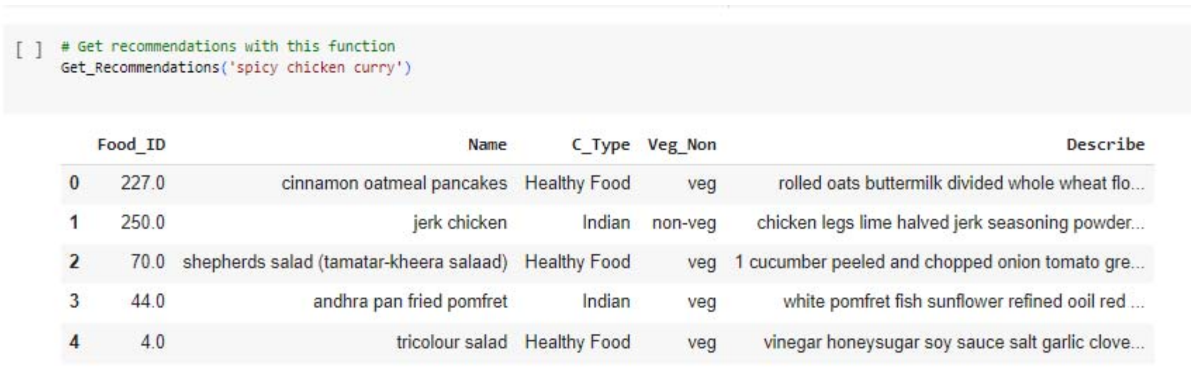


```
[ ]  # Using cosine similarity to find nearest neigbours
     recommender = NearestNeighbors(metric='cosine')
     recommender.fit(csr_rating_matrix)

          NearestNeighbors
     NearestNeighbors(metric='cosine')
```



```
[ ]  # The main recommender code!
     def Get_Recommendations(title):
         user= df[df['Name']==title]
         user_index = np.where(rating_matrix.index==int(user['Food_ID']))[0][0]
         user_ratings = rating_matrix.iloc[user_index]

         reshaped = user_ratings.values.reshape(1,-1)
         distances, indices = recommender.kneighbors(reshaped,n_neighbors=16)

         nearest_neighbors_indices = rating_matrix.iloc[indices[0]].index[1:]
         nearest_neighbors = pd.DataFrame({'Food_ID': nearest_neighbors_indices})

         result = pd.merge(nearest_neighbors,df,on='Food_ID',how='left')

         return result.head()
```

9) <u>Results: Recommendation of collaborative filtering</u>



```
[ ]  # Get recommendations with this function
     Get_Recommendations('spicy chicken curry')
```

	Food_ID	Name	C_Type	Veg_Non	Describe
0	227.0	cinnamon oatmeal pancakes	Healthy Food	veg	rolled oats buttermilk divided whole wheat flo...
1	250.0	jerk chicken	Indian	non-veg	chicken legs lime halved jerk seasoning powder...
2	70.0	shepherds salad (tamatar-kheera salaad)	Healthy Food	veg	1 cucumber peeled and chopped onion tomato gre...
3	44.0	andhra pan fried pomfret	Indian	veg	white pomfret fish sunflower refined ooil red ...
4	4.0	tricolour salad	Healthy Food	veg	vinegar honeysugar soy sauce salt garlic clove...

10) <u>Using neural network to predict ratings of food items:</u>

Preprocessing:

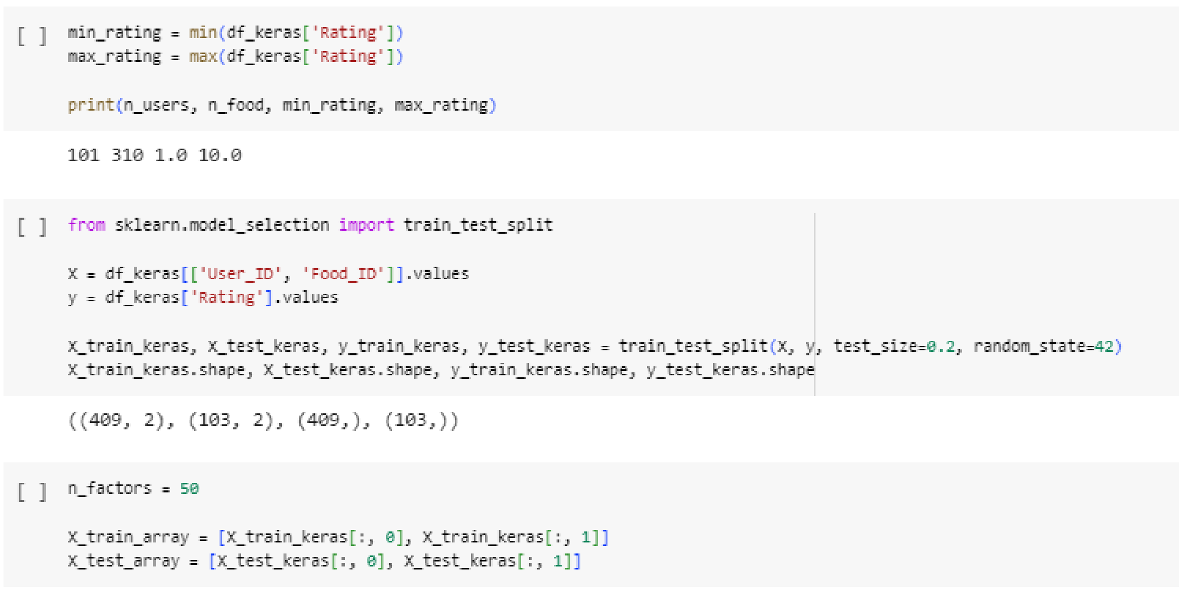
We are using LabelEncoder from sklearn to encode food and user id's. We will create variables that store unique users, food items, min_rating, and max_rating.

We will need another variable that stores the number of factors per user/food for the model. This number can be arbitrary. But for the Collaborative filtering model it needs to be the same size for both users and foods.

Finally, we will store users and foods into separate arrays for the train and test set. It is because in Keras they'll each be defined as distinct inputs.

Label encoding is a technique used in machine learning and data analysis to convert categorical variables into numerical format. It is particularly useful when working with algorithms that require numerical input, as most machine learning models can only operate on numerical data.

Keras is an open-source library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library. Here we first split data into train and test.



```
[ ] min_rating = min(df_keras['Rating'])
    max_rating = max(df_keras['Rating'])

    print(n_users, n_food, min_rating, max_rating)

    101 310 1.0 10.0
```

```
[ ] from sklearn.model_selection import train_test_split

    X = df_keras[['User_ID', 'Food_ID']].values
    y = df_keras['Rating'].values

    X_train_keras, X_test_keras, y_train_keras, y_test_keras = train_test_split(X, y, test_size=0.2, random_state=42)
    X_train_keras.shape, X_test_keras.shape, y_train_keras.shape, y_test_keras.shape

    ((409, 2), (103, 2), (409,), (103,))
```

```
[ ] n_factors = 50

    X_train_array = [X_train_keras[:, 0], X_train_keras[:, 1]]
    X_test_array = [X_test_keras[:, 0], X_test_keras[:, 1]]
```

We build a neural network model: with input layer, 2 embedding layers, activation layer and output.

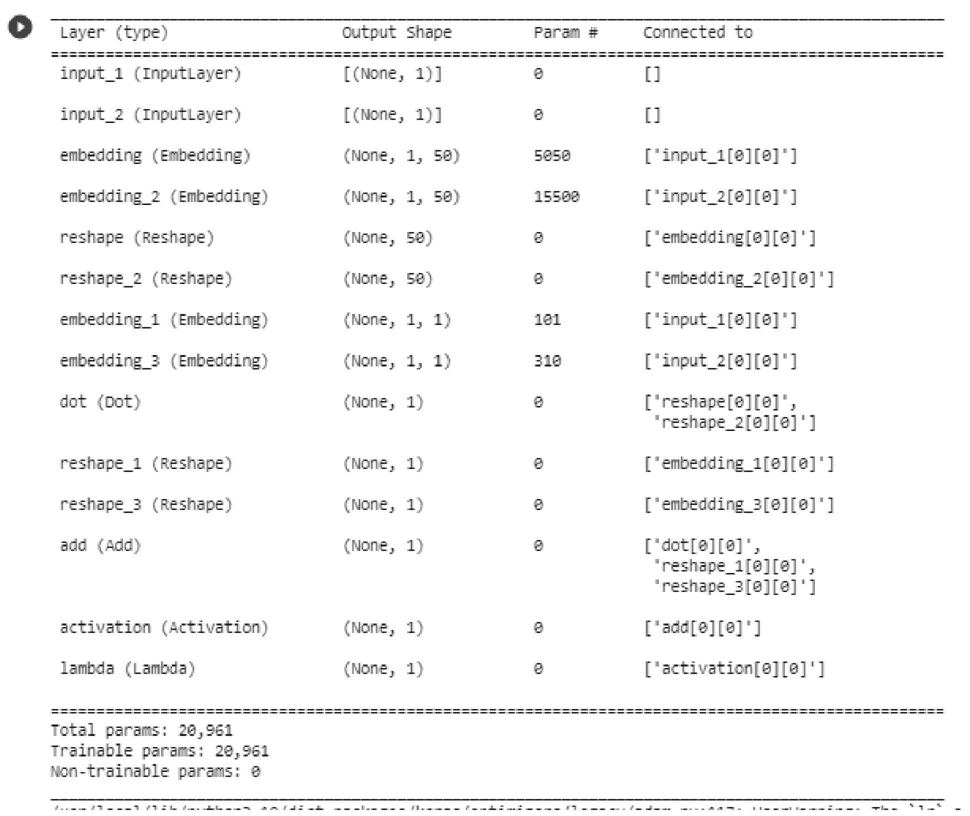
<u>What does Keras' embedding layer do?</u>

Each word is transformed into a fixed-length, predefined vector by means of an embedding layer offered by Keras. When using the one-hot-encoding technique, each word is represented by a huge, sparse matrix, but in embedding layers, each word has a real-valued vector with a set length.

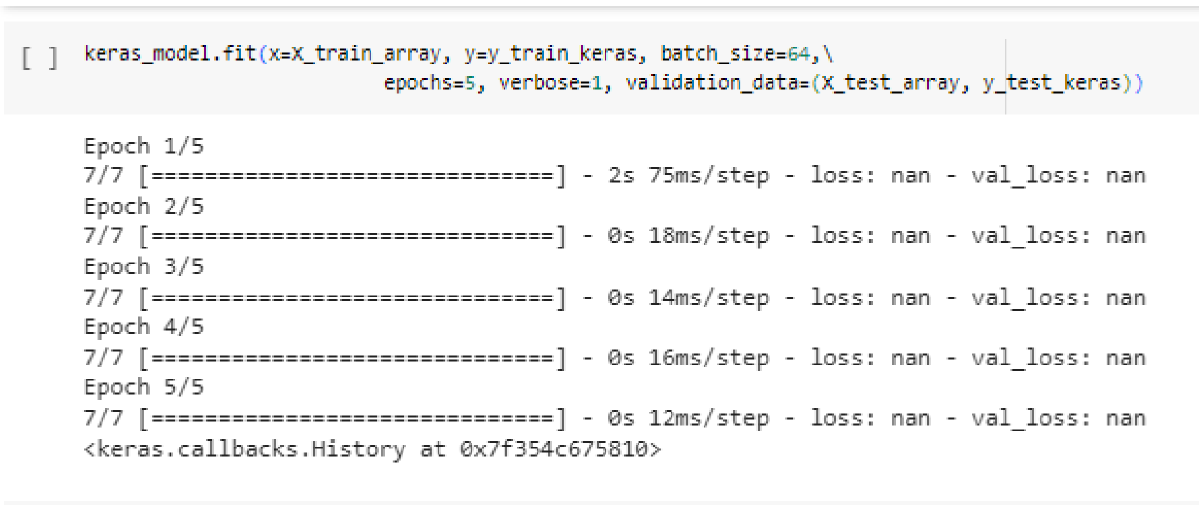
To represent each individual and each food item in the data, embeddings will be used in this instance. We must perform the dot product between the user vector and food vector in order to obtain these embeddings. We'll have vectors of size n factors to record the weights associated with each user per food item as a result.

In order to increase the model performance, we add the "bias" to each embedding. We run the output of the dot product through a sigmoid layer and then scaling the result using the min and max ratings in the data.

The summary of model:

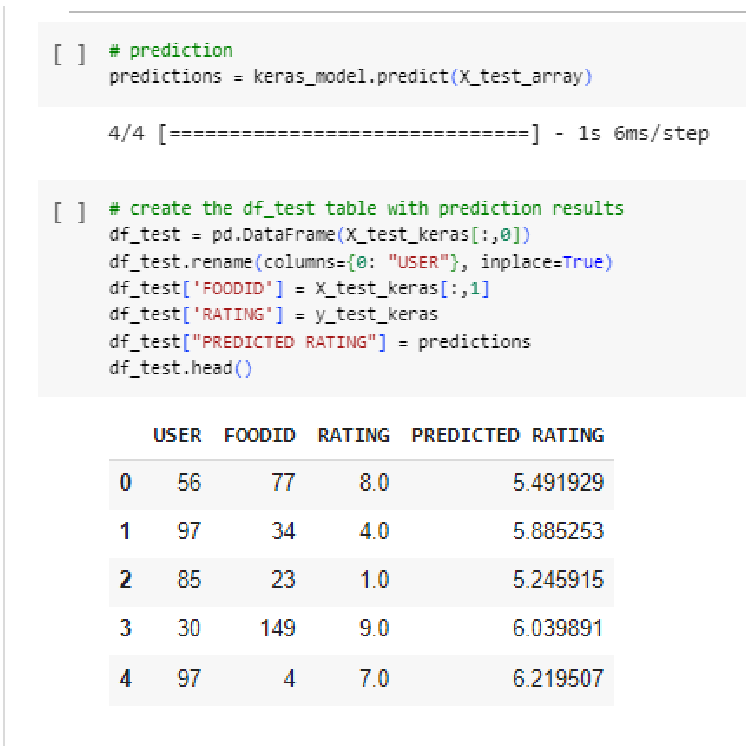


```
_____________________________________________________________________________________________________
 Layer (type)                Output Shape         Param #     Connected to
=====================================================================================================
 input_1 (InputLayer)        [(None, 1)]          0           []

 input_2 (InputLayer)        [(None, 1)]          0           []

 embedding (Embedding)       (None, 1, 50)        5050        ['input_1[0][0]']

 embedding_2 (Embedding)     (None, 1, 50)        15500       ['input_2[0][0]']

 reshape (Reshape)           (None, 50)           0           ['embedding[0][0]']

 reshape_2 (Reshape)         (None, 50)           0           ['embedding_2[0][0]']

 embedding_1 (Embedding)     (None, 1, 1)         101         ['input_1[0][0]']

 embedding_3 (Embedding)     (None, 1, 1)         310         ['input_2[0][0]']

 dot (Dot)                   (None, 1)            0           ['reshape[0][0]',
                                                               'reshape_2[0][0]']

 reshape_1 (Reshape)         (None, 1)            0           ['embedding_1[0][0]']

 reshape_3 (Reshape)         (None, 1)            0           ['embedding_3[0][0]']

 add (Add)                   (None, 1)            0           ['dot[0][0]',
                                                               'reshape_1[0][0]',
                                                               'reshape_3[0][0]']

 activation (Activation)     (None, 1)            0           ['add[0][0]']

 lambda (Lambda)             (None, 1)            0           ['activation[0][0]']

=====================================================================================================
Total params: 20,961
Trainable params: 20,961
Non-trainable params: 0
_____________________________________________________________________________________________________
```



```
[ ]  keras_model.fit(x=X_train_array, y=y_train_keras, batch_size=64,\
                                   epochs=5, verbose=1, validation_data=(X_test_array, y_test_keras))

     Epoch 1/5
     7/7 [==============================] - 2s 75ms/step - loss: nan - val_loss: nan
     Epoch 2/5
     7/7 [==============================] - 0s 18ms/step - loss: nan - val_loss: nan
     Epoch 3/5
     7/7 [==============================] - 0s 14ms/step - loss: nan - val_loss: nan
     Epoch 4/5
     7/7 [==============================] - 0s 16ms/step - loss: nan - val_loss: nan
     Epoch 5/5
     7/7 [==============================] - 0s 12ms/step - loss: nan - val_loss: nan
     <keras.callbacks.History at 0x7f354c675810>
```

11) Making Predictions using nueral network model:



```
[ ]  # prediction
     predictions = keras_model.predict(X_test_array)

     4/4 [==============================] - 1s 6ms/step
```

```
[ ]  # create the df_test table with prediction results
     df_test = pd.DataFrame(X_test_keras[:,0])
     df_test.rename(columns={0: "USER"}, inplace=True)
     df_test['FOODID'] = X_test_keras[:,1]
     df_test['RATING'] = y_test_keras
     df_test["PREDICTED RATING"] = predictions
     df_test.head()
```

	USER	FOODID	RATING	PREDICTED RATING
0	56	77	8.0	5.491929
1	97	34	4.0	5.885253
2	85	23	1.0	5.245915
3	30	149	9.0	6.039891
4	97	4	7.0	6.219507

12) Results for nueral network and discussion:

So, we have plotted bar graph to show the embedded form of the predicted ratings output.

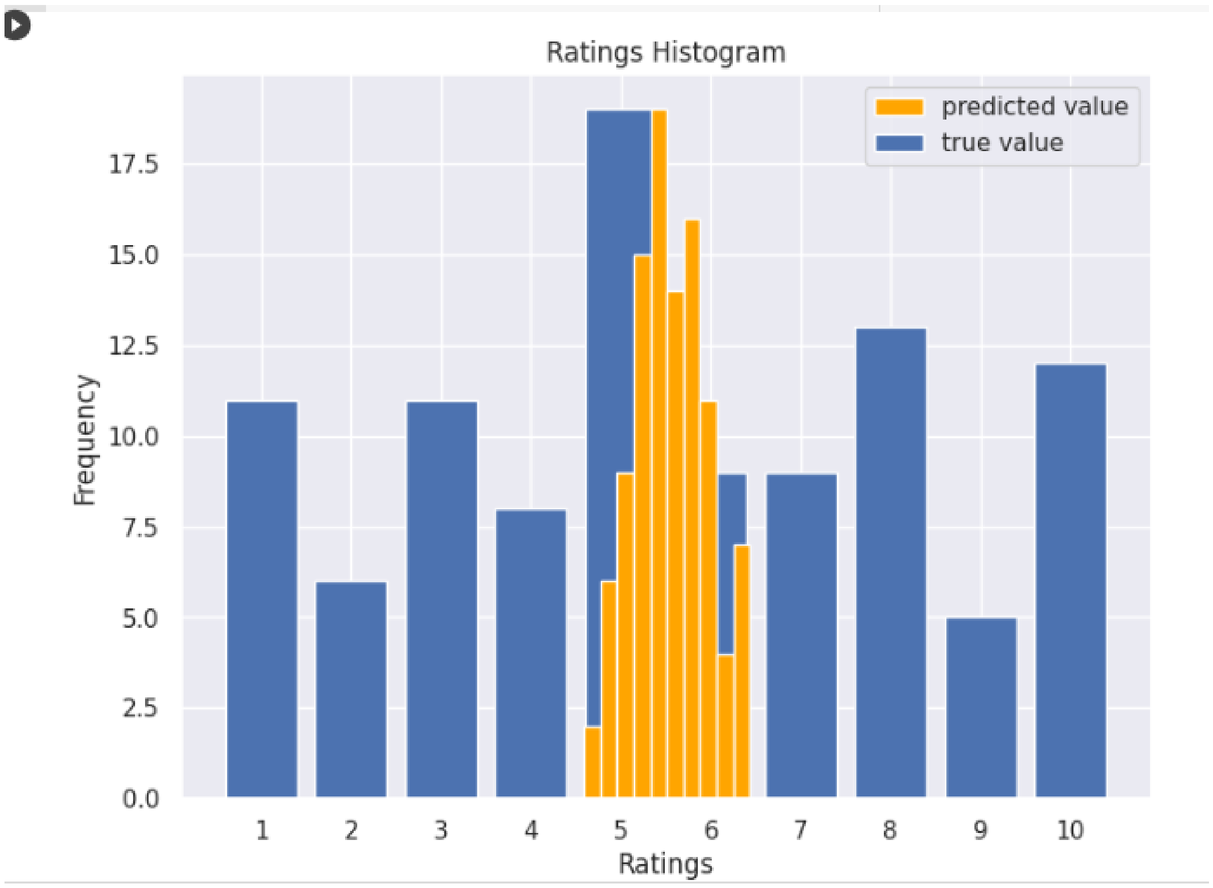


Fig 3: Food recommendation ratings graph

Here in the above graph, we can observe frequency of ratings given by users. Suppose rating 4.0 is given by average 7.5 users. Ratings is on x axis i.e ranging from 0 to 10 and Frequency is on y axis ranging from 0 to 17.5 as the maximum ratings count for rating 5.0.

After this, we can use those embeddings for Cosine similarity.

Cosine Similarity is used to calculate a numeric quantity that denotes the similarity between two food items. Therefore, we can to extract embedding layers from the Keras model to compute the cosine similarity by doing a dot product.

For recommending food we have to replace the ratings dataset with a combined dataset of previous dataset with ingredients and the ratings.

CHAPTER 7: CONCLUSION AND FUTURE SCOPE

7.1 Conclusion

In this project we have utilised the concept of personalised deep nueral networks to implement in a application of Food recommendation system based on the ingredients and ratings where we have used 2 main methods i.e. Content Based filtering using Tfidf and with count vectorizer, Collaborative filtering. We got the recommendations of food dishes according to cosine similarity functions on the ingredients column with three different methods and we have observed that collaborative filtering gives more accurate recommendations as compared to content based and content based with count vectorizer gives more accurate recommendations as compared to content based with tfidf vectors. We have also implemented a nueral network using tensorflow and predicted the ratings of the users using a labelled dataset of Ratings and got the embedded layers output in the range of 5 to 6 ratings. Those embeddings can be used to get recommendations using user ratings.

7.2 Future scope

Food recommendation is an important domain both for individuals and society. What the work described in this report shows is that despite its importance, food item recommendation, in comparison to other domains is relatively under-researched. There is a lot of future scope for this project. Natural Language Processing can be used to recover word embeddings from reviews about a food item or recipe. Different types of machine learning algorithms can be used and combined with collaborative filtering to recommend food based on nutritional values. Food recommendation can also be incorporated with Diet recommendation system for diabetic patients or any other personalised category. We can use neural collaborative filtering and sentiment analysis using deep learning methods together for the data analysis that provides the insights in a more convenient, well organized, reliable and accurate means.

The research done to date demonstrates that although user taste predictions for food can be accomplished using current techniques, their outcomes are less successful than in other domains. Another important finding in the research on preference prediction is the significance of context variables. Capturing significant context factors using various sensors and incorporating them into recommendation models would be one intriguing research area. Context, social circumstances, and group recommendations need to be taken into account more specifically. Technology systems need to take into account how social culinary experiences affect food choices and how prevalent they are.

Food recommenders for nutritional health are one particular role in food recommendation systems that, for societal and socioeconomic reasons, has become a popular research topic. Researchers have suggested a variety of approaches to integrate nutrition (meal plans, nudging, nutritional components in algorithms), but as of now, all of these recommendations are still in the early stages and it is unclear which strategy would work best.

As a final point, the community still needs to develop the evaluation of food recommenders and the techniques used to do so. The majority of literature review has been done offline with private collections. We must collaborate as a community to establish common data collection methods, baseline methodologies, and, most critically, more online research to understand how our methods function as real systems when applied to realistic situations.

REFERENCES:

[1] T. Maruyama, Y. Kawano, and K. Yanai, “Real-time mobile recipe recommendation system using food ingredient recognition,” in Proceedings of the ACM international workshop on interactive multimedia on mobile and portable devices, 2012, pp. 27–34.

[2] N. Nag, V. Pandey, and R. Jain, “Live personalized nutrition recommendation engine,” in Proceedings of the 2Nd International Workshop on Multimedia for Personal Health and Health Care, 2017, pp. 61–68.

[3] C. Trattner and D. Elsweiler, “Food recommender systems: Important contributions, challenges and future research directions,” arXiv preprint arXiv:1711.02760, 2017.

[4] Nattaporn Thongsri, Pattaraporn Warintarawej, Santi Chotkaew, Wanida Saetang , Implementation of a personalized food recommendation system based on collaborative filtering and knapsack method

[5] L. Yang, C. K. Hsieh, H. Yang, J. P. Pollak, N. Dell, S. Belongie, C. Cole, and D. Estrin, “Yum-Me: A personalized nutrient-based meal recommender system,” ACM Transactions on Information Systems, vol. 36, no. 1, p. 7, 2017.

[6] Y.-K. Ng and M. Jin, “Personalized recipe recommendations for toddlers based on nutrient intake and food preferences,” in Proceedings of the 9th International Conference on Management of Digital EcoSystems, 2017, pp. 243–250.

[7] International Journal of Electrical and Computer Engineering (IJECE)
Vol. 12, No. 1, February 2022, pp. 630~63 “Implementation of a personalized food recommendation system based on collaborative filtering and knapsack method” by Nattaporn Thongsri, Pattaraporn Warintarawej, Santi Chotkaew, Wanida Saetang

DATASET: Kaggle Link:

https://www.kaggle.com/datasets/schemersays/food-recommendation-system

CODE Available on github: https://github.com/PratimaChinta/Food-recom-system