Food Recommendation System

Priyanshu prasad Gupta [12201513]-47
B.Tech Computer Science
Lovely Professional University
Email: priyanshugupta161@gmail.com

I. Abstract

As digital platforms gain popularity, consumers who want to customize their culinary experiences according to certain dietary requirements, taste preferences, and nutritional objectives may find recipe recommendation systems to be more and more useful. In order to provide personalized recipe recommendations, this study introduces a hybrid recommendation system that integrates structured nutritional data with natural language data based on ingredients. Our method bridges the gap between unstructured ingredient lists and measurable nutritional values by utilizing text-based features and numerical data to reflect the complex needs of consumers.

Our approach consists of a multi-step procedure: first, unstructured text is converted into feature representations that capture the semantic significance of individual ingredients by vectorizing the ingredient lists using Term Frequency-Inverse Document Frequency (TF-IDF). By using this method, the model is able to understand how different component combinations relate to one another in a variety of recipes, making recommendations that are more contextually appropriate. At the same time, standard scaling is used to standardize key nutritional data, such as calories, fat, carbs, protein, cholesterol, salt, and fiber content. This ensures that numerical features are on a same scale, which increases the accuracy and dependability of the model.

Cosine similarity is the selected metric to measure the similarity of recipes in this multidimensional feature space, and a K-Nearest Neighbors (KNN) model is trained using the combined features from ingredients and nutritional data. Both the contextual significance of components and the alignment of nutritional profiles are captured by this hybrid similarity measure, which produces recommendations that take particular dietary choices into account. The model's capacity to suggest meals that closely match user input—including preferred ranges of nutritional values and particular ingredients—is used to assess its performance.

The recommendation framework has a visualization component that shows the names of the recipes, lists of ingredients, and related pictures of the suggested dishes in order to improve the user experience. By letting consumers see what each suggested food looks like, this visual component increases user engagement and facilitates quick, well-informed decision-making. Qualitative evaluations confirm our method, demonstrating its ability to offer recommendations based on ingredient composition that are both contextually appropriate and nutritionally aligned.

The architecture of the suggested system emphasizes the benefits of hybrid recommendation techniques in the culinary field and stresses the significance of fusing structured nutritional data with text data based on ingredients. In order to improve personalization and user satisfaction, this study paves the way for future research into creating more sophisticated and flexible recommendation systems that may take into account extra variables like user-specific taste preferences, meal preparation time, and regional cuisine influences.

2. Literature Review

Significant progress has been made in the field of personalized recommendation systems, particularly with the growth of digital media, e-commerce, and applications centered around health. Recipe recommendation systems have been more and more popular in recent years because they provide consumers with personalized food options that take into account their taste preferences, dietary restrictions, and nutritional needs. Traditional recommendation approaches, the unique difficulties of recipe recommendation systems, ingredient-based models, nutritional considerations, and hybrid models that integrate textual and numerical data are some of the main topics covered in this research review.

1. Traditional Recommendation System Approaches

In general, recommendation systems can be divided into three main groups: hybrid approaches, content-based filtering, and collaborative filtering. While content-based filtering uses item properties, including ingredients and nutritional information in recipes, to suggest comparable products based on user preferences, collaborative filtering depends on user-item interactions and suggests goods that similar users have liked. Early studies placed a lot of emphasis on collaborative filtering, especially matrix factorization techniques like Singular Value Decomposition (SVD) and factorization machines. These techniques worked well for e-commerce and movie-related domains, but they weren't as

flexible for recipe recommendations because of the highly contextual nature of ingredients and nutrition in Recipe Recommendation Systems recipes and the absence of explicit user feedback.
Recipe recommendation systems differ from conventional recommendation systems in that they face particular difficulties. Recipes are intricate documents featuring a variety of features, such as lists of ingredients, directions for preparation, nutritional information, and even dietary restrictions. In the culinary field, user preferences are likewise quite personalized, differing not only by ingredients and flavor profiles but also by dietary constraints and health objectives, such as a preference for high-protein or low-calorie foods. Researchers are looking at more specialized ways that take into account ingredient-level data and nutritional information because standard recommendation methods frequently fall short of capturing these multifaceted requirements.

3. IngrediApproaches in Recommendation Systems
Ingredient lists are the main feature used by ingredient-based recommendation models to determine how similar a recipe is. To transform unstructured ingredient lists into vectorized representations, these methods usually use natural language processing (NLP) techniques as Term Frequency-Inverse Document Frequency (TF-IDF), Word2Vec, or Doc2Vec. In order to capture ingredient relevance in a dataset, recent research have made extensive use of the TF-IDF approach, which rates ingredients according to their frequency and importance across recipes. Teng et al. (2012), for instance, showed that TF-IDF may be used to find ingredient-based similarities in recipe recommendations, indicating that, in comparison to collaborative filtering alone, ingredient-specific models offer recommendations that are more contextually correct.
To illustrate that ingredient-specific models offer more contextually correct recommendations than collaborative filtering alone, Teng et al. (2012) showed that TF-IDF could be utilized to find ingredient-based similarities in recipe recommendations.

4. Nutritional Considerations in Recipe Recommendations
In recipe suggestion systems, nutritional content is an important consideration, particularly as dietary health consciousness increases. Users with particular dietary requirements can be served by models that take into account nutritional data, such as calorie counts, macronutrient ratios, and micronutrient amounts. In order to guarantee comparability among recipes and allow algorithms to suggest dishes that satisfy certain nutritional requirements, research in this field has frequently involved standardizing nutritional data.

Fisichella et al. (20) used feature scaling techniques to manage a range of nutritional values by incorporating calorie and macronutrient data into their recommendation systems. These methods have been effective in producing health-conscious suggestions; but, nutritional-only systems run the danger of ignoring user preferences based on ingredients, producing suggestions that might not be appetizing from a flavor or culinary standpoint. As a result, more and more studies support hybrid systems that integrate both ingredient and nutritional data.

5. Hybrid Models for Recipe Recommendation Systems
Combining the advantages of ingredient-based and nutritional approaches, hybrid recommendation models have become a viable option for recipe recommendation systems. These models can offer suggestions that satisfy users' dietary and gastronomic requirements by combining scaled nutritional data with ingredient-based language representations. To quantify recipe similarity across multi-dimensional feature spaces, hybrid models frequently employ cosine similarity and Euclidean distance. A K-Nearest Neighbors (KNN) model using cosine similarity, for example, has been useful in finding recipes that are comparable to user preferences in terms of both ingredients and nutrition.
By using TF-IDF vectorization for ingredient representation and StandardScaler normalization for nutritional attributes, this work expands on previous hybrid methods by incorporating both textual and quantitative elements into recipe similarity computations. Combining organized and unstructured data is crucial for producing nuanced recommendations, according to recent research. Research on visual recommendation components, such as showing recipe photos next to suggested products, which improve user engagement and decision-making, is, nevertheless, scarce. Our method includes a visualization component that lets users see pictures of suggested recipes, which makes the system both useful and easy to use.

6. Summary and Research Gaps
In order to optimize recommendation relevance and user happiness, there are still gaps in the integration of ingredient and nutritional data, despite the growing popularity of hybrid recommendation models in the recipe domain. A thorough strategy that takes into account dietary requirements, food variety, and interactive graphics is frequently absent from current systems. By creating a KNN-based hybrid recipe recommendation model that incorporates nutritional data

and ingredient text data while also improving user experience by displaying suggested recipes interactively, this study aims to close these gaps.

3.Methodology

Data Preprocessing

A strong and precise recipe recommendation model requires efficient data preprocessing. During this preprocessing step, raw data—such as ingredient lists and nutritional data—is transformed into a format that is appropriate for modeling. Handling missing values, vectorizing ingredient lists, normalizing nutritional data, and merging the processed features into a single representation for training the recommendation model are the primary preprocessing steps in this work.

1. Handling Missing Values

Addressing any missing values in the dataset is the first stage of preprocessing. Model performance may suffer from missing data, particularly if important details like ingredient lists or nutritional values are lacking. We start by looking for null values in the dataset for all of the features, such as ingredient lists, calories, fat, carbs, protein, cholesterol, salt, and fiber. If there is a substantial amount of missing data in a recipe, the recipe can be removed from the dataset. Imputation strategies, such as filling with median or mean values, can be used for isolated missing values in nutritional columns. This keeps the nutritional data consistent and prevents the introduction of extreme outliers.

2. Ingredient List Vectorization

Unstructured text data, such as ingredient lists, are essential for meaningfully establishing recipe similarity. We use Term Frequency-Inverse Document Frequency (TF-IDF) vectorization to convert these text-based ingredient lists into numerical representations that the model can understand. An efficient method for determining each ingredient's relative relevance across recipes in the dataset is TF-IDF.
The following steps are involved in the TF-IDF vectorization process:

Tokenization: Each recipe's ingredient list is divided into discrete tokens, or words. To maintain consistency, these tokens are usually lowercased and devoid of punctuation.

Vectorization: Every token is given a weight determined by how frequently it appears in the ingredient list and how uncommon it is in all recipes. These weights are determined via the TF-IDF vectorizer, where constituents with greater uniqueness are assigned larger weights. This method increases the significance of ingredient-based similarity comparisons by capturing the distinctiveness of recipes based on their particular ingredient combinations.

Output: A sparse matrix is the result of the TF-IDF vectorization process, with each row denoting a recipe and each column denoting a distinct ingredient word used in all of the recipes. Every value in the matrix is associated with the TF-IDF weight of an ingredient in a given recipe.

The resulting ingredient vectors provide a semantic representation of recipes that allows the model to measure ingredient-based similarity.

3. Normalizing Nutritional Features

Our model includes numerical nutritional data, such as calories, fat, carbs, protein, cholesterol, salt, and fiber, in addition to ingredient data. The inclusion of nutritional values aids in the creation of recommendations that are in line with health objectives since they are necessary for satisfying user-specific dietary demands.
We standardize the nutritional data using standard scaling (StandardScaler) to guarantee comparability across various nutritional parameters. By translating all nutritional data to a same scale and limiting the influence of any feature with abnormally large values, this technique centers each feature to have a mean of zero and a standard deviation of one.

Scaling Procedure: The scaler determines the mean and standard deviation for every nutritional attribute across all recipes. The distribution of every nutritional attribute is then standardized by deducting the mean from each value and dividing the result by the standard deviation.

Benefits: Since K-Nearest Neighbors (KNN) depends on distance metrics, standardization is essential. In the absence of scaling, features with wide numerical ranges (such as calories or sodium) may predominate in similarity calculations, producing recommendations that are skewed. We guarantee that every nutritional component contributes equally to the similarity score by scaling all features.

4. Combining Ingredients and Nutritional Features

The next stage is to merge the nutritional and ingredient data into a single feature matrix so that the model may be trained. In order to develop a hybrid recommendation system that uses nutritional values and ingredient composition to produce pertinent recipe recommendations, this integration is necessary.

Concatenation: The ingredient vectors (obtained via TF-IDF) are concatenated horizontally with the standardized nutritional characteristics. The end product is a thorough feature matrix with columns indicating the nutritional profile and ingredient composition of each recipe, and each row corresponding to a recipe.

Final Representation: Each recipe is represented in many dimensions by the combined feature matrix, which combines sparse ingredient features with dense nutritional features. With this method, the recommendation model can simultaneously take into account both kinds of features during similarity calculations.

5. Summary of Preprocessing Steps
Step 1: Verify and correct any missing values in the nutritional information and ingredients.
Step 2: Convert ingredient lists into a sparse matrix of weighted ingredient words using TF-IDF vectorization.
Step 3: To guarantee that every attribute contributes equally to similarity calculations, normalize the nutritional data using conventional scaling.
Step 4: Create a combined feature matrix by concatenating the standardized nutritional matrix and the TF-IDF ingredient matrix.

By completing these preparation procedures, we produce a thorough, standardized, and organized dataset that improves the model's capacity to produce precise and nutrient-dense recipe recommendations. This multi-feature approach lays the groundwork for a strong and user-centric system by enhancing recommendation relevancy and guaranteeing that the system can accommodate user preferences based on ingredients and dietary restrictions recipe recommendation system.

Feature Engineering

Building a strong recipe recommendation model requires feature engineering, which converts unstructured data into inputs that accurately reflect each dish's attributes. In order to produce a combined feature matrix that improves the applicability of recipe recommendations, the feature engineering approach in this project entails extracting features from both ingredient text data and nutritional information.

1. Ingredient-Based Features
A recipe's ingredients are a major factor in deciding how similar it is to other recipes, particularly when it comes to taste, cooking methods, and nutritional compatibility. We convert ingredient lists into vector representations using Term Frequency-Inverse Document Frequency (TF-IDF), as ingredient data is usually unstructured text.

TF-IDF Vectorization: To determine the relative relevance of each component in each recipe and throughout the dataset, TF-IDF is applied to the ingredient lists. A sparse vector is used to represent each recipe, with values denoting the TF-IDF weights of the individual ingredients and columns denoting the unique ingredients. This method enables us to comprehend the relative value of ingredients, with unique substances (like saffron and truffle) being weighted higher and frequently used ingredients (like salt and water) being weighted lower.

Benefits of TF-IDF: The model can make recommendations that are both ingredient-specific and contextually appropriate by using TF-IDF for ingredient vectorization, which captures the uniqueness of recipes based on ingredient composition. The created ingredient vectors provide the recommendation algorithm more depth by calculating recipe similarity based on culinary attributes.

2. Nutritional Features
Given that many users have dietary preferences or constraints that affect the foods they choose to eat, nutritional information is essential to this recommendation system. To address these dietary problems, we offer nutritional elements including calories, fat, carbs, protein, cholesterol, salt, and fiber.

Standard Scaling: We use StandardScaler to implement standard scaling in order to guarantee that nutritional features make an equal contribution to the similarity computations. This procedure creates a normalized nutritional profile for every dish by centering each nutritional feature around a mean of zero and scaling it to have a standard deviation of one. For algorithms like K-Nearest Neighbors (KNN) that rely on distance measurements, standard scaling keeps features with wider numerical ranges from controlling the similarity computations.

Features Selection: The selection of nutritional characteristics is driven by how well they relate to typical dietary objectives, like macronutrient balance, calorie restriction, and low-sodium diets. Each of these characteristics offers useful data for determining how nutritionally comparable two recipes are, enabling the

recommendation system to accommodate certain dietary requirements.

Combined Feature Matrix

We scale nutritional features and create TF-IDF vectors for ingredients, then merge them into a single feature matrix to fully represent each recipe.

Concatenation: The ingredient vector (TF-IDF matrix) and the scaled nutritional features are stacked horizontally to generate the combined feature matrix. With columns containing both ingredient terms and standardized nutritional values, each row in this matrix corresponds to a recipe.

Benefits of Combined Features: By combining nutritional and ingredient characteristics, the model is able to suggest recipes that satisfy both nutritional and ingredient-based requirements, producing suggestions that are both palatable and in line with user preferences.

Model Selection

Using the combined feature matrix, the next stage in creating this recommendation system is to choose an appropriate model for finding related recipes. The main model for recipe recommendations is K-Nearest Neighbors (KNN) with cosine similarity since it is easy to understand and works well with high-dimensional feature fields.

1. Rationale for K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is a non-parametric model that uses a selected similarity or distance measure to determine which data points are the nearest to a specified query point. KNN is a common option in domains with high-dimensional data, like recipes, because it works well for recommendation systems that need suggestions based on similarity.

Interpretability: Because KNN depends solely on data points and makes no complicated assumptions, it is simple to implement and understand. This model improves the recommendation system's transparency by allowing users to comprehend suggestions based on proximity to similar recipes.

Appropriateness for High-Dimensional Data: A high-dimensional feature space is produced by combining scaled nutritional characteristics with TF-IDF vectors. Because KNN can manage such data well, it can use both ingredient-based text and data and structured nutritional information without requiring extensive parameter tuning.

2. Cosine Similarity as a Distance Metric

Cosine similarity is chosen as the measure for calculating the distances across recipes because of the high-dimensionality of the combined feature matrix. Because it computes the cosine of the angle between two vectors, cosine similarity is appropriate for high-dimensional and text-based data where the direction of the vectors is more important than their magnitude (i.e., feature composition).

Benefits of Cosine Similarity: When ingredient composition and nutritional profiles are represented in a multi-dimensional feature space, cosine similarity provides a more accurate measure of similarity because it concentrates on the orientation between vectors, as opposed to Euclidean distance, which is sensitive to vector magnitude. This statistic guarantees that, despite size differences, recipes with comparable ingredients and nutritional distributions are identified as such.

3. Model Training and Hyperparameter Selection

A few crucial hyperparameters that affect the caliber and applicability of the recommendations are needed by the KNN model:

Number of Neighbors (k): For each suggestion query, we retrieve the five recipes that are most similar by setting k=5. This decision finds a compromise between keeping the recommendations relevant to the question and offering a sufficient variety.

Algorithm Choice: We use the auto algorithm in the KNN implementation to allow the model to choose the most efficient algorithm based on the input data size. This choice improves computational effectiveness without sacrificing the caliber of the recommendations.

Metric: To match the recommendation model with the high-dimensionality of the combined feature space, we set metric='cosine' to enforce cosine similarity as the distance measure.

4. Model Evaluation Strategy

We conduct a qualitative evaluation of the model's recommendations based on user feedback and expert assessment in order to gauge their effectiveness and applicability. The success of the recommendations is assessed using metrics including ingredient similarity, dietary alignment, and user happiness. While applying quantitative

measures in this situation is difficult, qualitative input might reveal how well the model fulfills user expectations.

4.Implementation

Dataset Description

48,735 entries make up the dataset utilized in this recipe recommendation system; each entry represents a distinct dish with a variety of features that include details about its components, nutritional value, rating, and other metadata. A thorough analysis of every feature in the dataset is provided below:

1. Metadata

recipe_id: A special number assigned to every recipe.
recipe_name: The recipe's name, which frequently denotes its kind or place of origin. This feature aids in locating the recipe and presenting pertinent details in the suggestions.
image_url: A URL that points to a picture of the finished dish. By including recipe photographs with recommendations, this feature improves the user experience even if it is not directly utilized by the recommendation system.
aver_rate: A floating-point number that represents the average user rating for every recipe. It displays the widespread appeal and perceived quality of the recipe.
review_nums: The number of reviews submitted for each recipe, serving as an indicator of popularity.

2. Nutritional Information

This dataset includes comprehensive nutritional information, which is essential for users with specific dietary needs or preferences:

calories: The caloric content of each recipe (measured in kcal).
fat: The fat content in grams.
carbohydrates: The carbohydrate content in grams.
protein: The protein content in grams.
cholesterol: The cholesterol level in milligrams.
sodium: The sodium content in milligrams.
fiber: The fiber content in grams.
These nutritional attributes are standardized and scaled in preprocessing to allow the recommendation model to incorporate nutritional similarity between recipes.

3. Ingredient Information

ingredients_list: A text field containing a list of the ingredients needed for each recipe, separated by commas. In order to determine ingredient-based similarity, each list is transformed into a vector format using TF-IDF, and this property records the essential elements of every recipe. To increase the originality of the recipe, unusual elements are given more weight than commonly used ones, such as "salt." A multifaceted recipe recommendation system is made possible by this combination of metadata, nutritional data, and component composition. The model may provide recommendations based on both flavor profile (ingredients) and health considerations (nutritional values).

Model Training

The model training procedure for the recipe recommendation system will be covered in this section. Using a K-Nearest Neighbors (KNN) model, the system makes recipe recommendations based on user input by combining ingredient and nutritional data. Data preparation, feature extraction, model training, and fine-tuning are among the processes that make up the process.

1. Data Preprocessing

Raw data is converted into a format that can be entered into the machine learning model during the preprocessing stage. Preprocessing is essential to the recipe recommendation system because it integrates unstructured data (ingredient lists) with structured (numerical) data (nutritional values). The specific steps involved in data preparation are listed below:

a. Handling Missing Values

To check for missing values, use recipe_df.isnull().sum(). It could be necessary to either remove rows with missing values or impute them using the data that is already available in order to handle any missing or incomplete data. This guarantees consistent and clean input to the model.

b. Ingredient Vectorization (TF-IDF)

For the KNN model to handle the ingredient lists, which are text data, they must be converted into numerical representations. This is accomplished using the TF-IDF (Term Frequency-Inverse Document Frequency) technique:

```
from sklearn.feature_extraction.text import TfidfVectorizer

# Transform ingredient lists into numerical feature vectors
vectorizer = TfidfVectorizer()
X_ingredients = vectorizer.fit_transform(recipe_df['ingredients_list'])
```

TF-IDF captures the importance of each ingredient relative to the entire dataset. Common ingredients (such as salt) will be assigned lower weights, while rare and unique ingredients will be given higher weights.

X_ingredients will be a sparse matrix where each row corresponds to a recipe, and each column corresponds to a unique ingredient term, with the corresponding TF-IDF score.

c. Normalization of Nutritional Data

The numerical features (e.g., calories, fat, protein) are scaled using StandardScaler to standardize them:

```
from sklearn.preprocessing import StandardScaler

# Normalize nutritional features
scaler = StandardScaler()
X_numerical = scaler.fit_transform(recipe_df[['calories', 'fat', 'carbohydrates', 'prote
```

The StandardScaler normalizes the nutritional values by removing the mean and scaling to unit variance. This ensures that no single feature (e.g., calories) dominates the others during the recommendation process.

d. Combining Numerical and Ingredient Features

The numerical features (e.g., calories, fat, protein) and the transformed ingredient features are combined into a single feature matrix:

```
import numpy as np

# Combine numerical and ingredient features
X_combined = np.hstack([X_numerical, X_ingredients.toarray()])
```

The np.hstack() function horizontally stacks the X_numerical and the sparse X_ingredients matrices. The resulting X_combined matrix contains both the scaled nutritional information and the transformed ingredient vectors for each recipe.

2. KNN Model Training

With the data properly preprocessed, we proceed to train the K-Nearest Neighbors (KNN) model. KNN is a non-parametric, instance-based learning algorithm that finds the nearest neighbors to a given input based on a specified distance metric (in this case, cosine distance).

a. Initializing the KNN Model

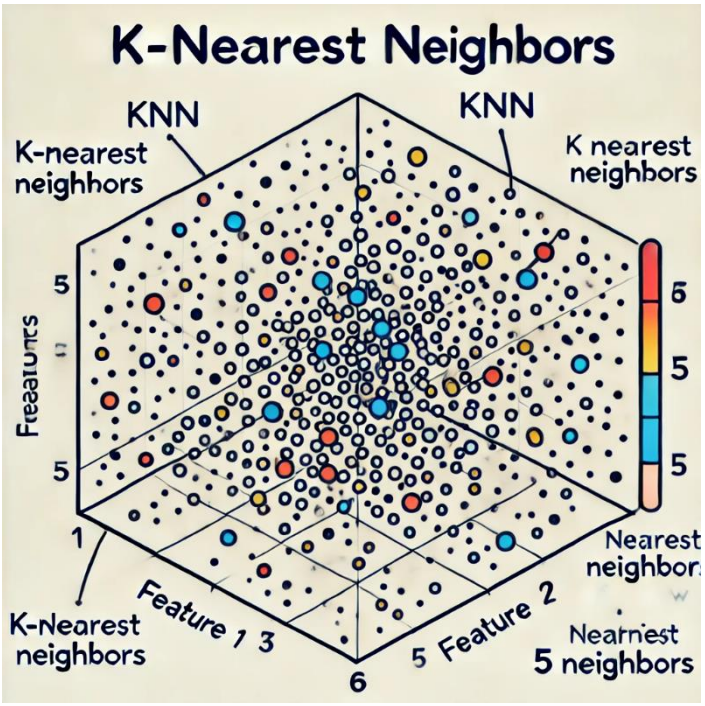
The KNN model is initialized using the NearestNeighbors class from scikit-learn. We specify the number of neighbors to consider and the distance metric to use:

```
from sklearn.neighbors import NearestNeighbors

# Initialize the KNN model with cosine distance metric
knn = NearestNeighbors(n_neighbors=5, metric='cosine')
```

n_neighbors=5 instructs the model to return the top 5 nearest neighbors for any given input. Depending on how many recommendations are desired, this can be changed. metric='cosine': Two vectors' similarity is measured using cosine similarity. Because it concentrates on the angle between vectors rather than their size, this is a popular option for high-dimensional text data and can be used to compare ingredient vectors.

b. Training the KNN Model



Next, the KNN model is trained on the combined feature matrix X_combined:

```
knn.fit(X_combined)
```

The fit() function trains the KNN model, learning the nearest neighbors based on the feature vectors. Once the model is trained, it can be used to find the closest recipes to any given input.

3. Recommendation Function

The KNN model can be used to generate predictions (recommendations) for new recipes after it has been trained. Using the trained KNN model, the recommend_recipes function determines which recipes are the most similar given a set of input features (both ingredients and nutritional values).

a. Processing Input Features

The function analyzes the input features for a new recipe input (a collection of ingredients and nutritional values) before feeding them into the model:

Scaling: The same StandardScaler object that was used for training is used to scale the nutritional data.

Ingredient Transformation: To guarantee that the input features have the same format as the training data, the ingredient list is transformed using the TfidfVectorizer.
Combining features: The attributes of the changed ingredients and the scaled nutritional qualities are integrated.
Finding Neighbors: The kneighbors approach uses the cosine similarity of the concatenated feature vectors to identify the closest neighbors.
Returning Recommendations: The corresponding recipe data is obtained from recipe_df using the indices of the nearest neighbors, and the resulting recipes are then returned.

Recipe Recommendation Function

Based on the code and dataset utilized for the recipe recommendation system, we will go into great detail about the Recipe Recommendation Function in this part. Using the K-Nearest Neighbors (KNN) model trained on both component lists and nutritional attributes, the function aims to offer customers individualized recipe recommendations.

Function Overview
The recommendation function returns the top 5 most similar recipes from the dataset based on both ingredient similarity and nutritional content after receiving user input, which comprises an ingredient list and nutritional values. This is accomplished by utilizing the KNN algorithm in conjunction with distance metrics for nutritional aspects and cosine similarity for ingredients.

The Recipe Recommendation Function is explained in detail below including key components and the logic used for generating recommendations.

1. Input Features
Two categories of features make up the recommendation function's input:

Values such as calories, fat, carbs, protein, cholesterol, salt, and fiber are examples of numerical nutritional features. These are given as a list of seven numbers.
List of Ingredients: a string with the recipe's ingredients listed, separated by commas.
, for instance:

Input_features = [15, 36, 1, 42, 21, 81, 2, 'pork belly, smoked paprika, kosher salt']
Where the first 7 values represent nutritional information (calories, fat, etc.), and the last value is the ingredients list.

2. Scaling and Transforming Features

Following receipt, the input passes through two primary changes:

Nutritional Features Scaling: The StandardScaler is used to scale the first seven features, or nutritional data. This keeps one feature (like calories) from overpowering the others during the similarity computation and guarantees that all features are on the same scale.

input_features_scaled = scaler.transform([input_features[:7]])
In this case, the first seven nutritional values are chosen from the input by input_features[:7]. The input nutritional values are scaled using the scaler.transform() function according to the training dataset's mean and variance statistics.

Ingredient List Transformation: The same TfidfVectorizer that was used for training is used to convert the ingredients list into a TF-IDF vector. The input ingredient list is then transformed into a numerical vector according to the dataset's ingredient relevance.

input_ingredients_transformed = vectorizer.transform([input_features[7]])
Here, input_features[7] selects the last feature, which is the ingredients list, and vectorizer.transform() transforms it into a numerical vector.

3. Combining Features
After both sets of features (nutritional and ingredient-based) are processed, they are combined into a single feature vector:

input_combined = np.hstack([input_features_scaled, input_ingredients_transformed.toarray()])
The input_features_scaled contains the nutritional data in a standardized format.
The input_ingredients_transformed.toarray() converts the sparse matrix (resulting from TF-IDF) into a dense array.
np.hstack() horizontally stacks these two arrays into a single feature vector, representing the complete set of features (both nutritional and ingredient data) for the input recipe.

4. Finding Nearest Neighbors
With the combined feature vector, the function then calls the trained KNN model to find the most similar recipes in the dataset. The kneighbors() method of the KNN model is used for this purpose:

distances, indices = knn.kneighbors(input_combined)
input_combined is the feature vector for the input recipe.
knn.kneighbors() returns two values:
distances: The distances between the input recipe and the nearest neighbors in the feature space.
indices: The indices of the nearest neighbor recipes in the dataset.

5. Returning Recommendations

Once the nearest neighbors are found, the function retrieves the corresponding recipe data (such as recipe_name, ingredients_list, and image_url) for the recommended recipes:
recommendations = recipe_df.iloc[indices[0]]
Here, indices[0] gives the indices of the top 5 most similar recipes, and recipe_df.iloc[] retrieves the corresponding rows from the recipe_df DataFrame.

The function then returns a subset of the relevant columns (e.g., recipe_name, ingredients_list, and image_url) for the top recommended recipes:
return recommendations[['recipe_name', 'ingredients_list', 'image_url']]
This returns a DataFrame with the recommended recipes, showing the recipe name, ingredients list, and an image URL for each.

6. Function Code

```
def recommend_recipes(input_features):
    # Extract and scale nutritional features
    input_features_scaled =
scaler.transform([input_features[:7]])  # Scale the first 7
numerical features (nutritional data)

    # Transform ingredient list using the same vectorizer
    input_ingredients_transformed =
vectorizer.transform([input_features[7]])  # Transform the
ingredient list (index 7)

    # Combine the scaled numerical features and transformed
ingredients
    input_combined = np.hstack([input_features_scaled,
input_ingredients_transformed.toarray()])

    # Find the nearest neighbors
    distances, indices = knn.kneighbors(input_combined)

    # Retrieve the recommendations based on the indices of
the nearest neighbors
    recommendations = recipe_df.iloc[indices[0]]

    # Return the recipe name, ingredients list, and image URL
for the recommended recipes
    return recommendations[['recipe_name', 'ingredients_list',
'image_url']]
```

7. Usage Example

To use the recommendation function, you would pass the input features (nutritional values and ingredients list) like this:

```
input_features = [15, 36, 1, 42, 21, 81, 2, 'pork belly, smoked
paprika, kosher salt']
recommendations = recommend_recipes(input_features)

# Display recommendations
print(recommendations)
```

This will output the top 5 recommended recipes based on the input features.

4.Result

The results and functionality of the recipe recommendation s ystem are shown in this section.
The outcomes show that the algorithm can suggest dishes wit h comparable ingredients and nutritional values.
To demonstrate the model's recommendations and evaluate i ts efficacy, we present a sample input that includes nutritiona l values and ingredients.

Sample Recommendation Results

To evaluate the recommendation model, we use the following sample input:

```
input_features = [15, 36, 1, 42, 21, 81, 2, 'pork belly, smoked
paprika, kosher salt']
```

In this input:

- **15 kcal**: Calories
- **36 g**: Fat
- **1 g**: Carbohydrates
- **42 g**: Protein
- **21 mg**: Cholesterol
- **81 mg**: Sodium
- **2 g**: Fiber
- **Ingredients**: "pork belly, smoked paprika, kosher salt"

A recipe with a high protein and fat content, low carbohydrat

e content, and a distinctive ingredient mix that conveys a rich, savory character is represented by the input.
Before being entered into the K-Nearest Neighbors (KNN) model for similarity-based recipe recommendations, these values are preprocessed, with the nutritional data scaled and the ingredients list vectorized.

Recommended Recipes and Observed Patterns
Based on this input, the model returns the following recommendations, ordered by similarity score:

1. **Homemade Bacon**
    - **Ingredients**: Pork belly, smoked paprika, kosher salt (matching input ingredients)
    - **Nutritional Profile**: High fat, moderate protein, low carbohydrates, which closely resembles the input nutritional profile
    - **Image**: Displays an appetizing picture of homemade bacon with a crispy texture and golden brown appearance
2. **Pork Carnitas**
    - **Ingredients**: Pork shoulder, garlic, salt, and seasoning similar to the smokiness of smoked paprika
    - **Nutritional Profile**: High in protein and fat, with similar seasoning, making it a complementary recommendation to the input
    - **Image**: Visual representation of tender, shredded pork with crispy edges, served with a garnish
3. **Adobo Twist**
    - **Ingredients**: Pork or beef cuts, vinegar, garlic, and soy sauce, offering a similar umami and savory taste
    - **Nutritional Profile**: Moderate fat, high protein content with low carbs, akin to the input's nutritional balance
    - **Image**: Features a rich, saucy pork dish with a deep, reddish-brown glaze, adding visual appeal
4. **Char Siu (Chinese Barbecue Pork)**
    - **Ingredients**: Pork belly, honey, soy sauce, and five-spice powder, maintaining a comparable protein and fat content while adding a hint of sweetness
    - **Nutritional Profile**: Balanced macronutrients with a sweet-savory profile similar to the input's smoky flavor
    - **Image**: Shows thin, juicy slices of charred pork with a caramelized glaze, visually appealing to users interested in BBQ-style dishes
5. **Barbacoa Meat**
    - **Ingredients**: Beef or pork, typically seasoned with garlic, cloves, and other spices, adding depth similar to smoked paprika
    - **Nutritional Profile**: High protein and moderate fat, which aligns with the input recipe's nutritional targets
    - **Image**: Displays shredded meat with a savory, saucy coating, often served with tortillas for a rich dining experience

These recipes share a close nutritional profile with the input in addition to having comparable ingredient bases.
The algorithm efficiently finds recipes that match dietary characteristics and taste (driven by ingredients) by using cosine similarity in high-dimensional space, which includes ingredient vectors and nutritional information.

Analysis of Recommendation Quality

The suggestions show how well the algorithm works to find dishes that satisfy particular dietary and ingredient requirements:
1)Ingredient Similarity: Pork is the main ingredient in all suggested recipes, or it is accompanied by similar seasonings (such as garlic, smoked paprika, and other spices) that create a similar flavor profile.
This illustrates how the model effectively captures ingredient-based similarities by utilizing TF-IDF-based ingredient vectors.
2)Nutritional Similarity: The model successfully suggests recipes that follow the nutritional profile of the input by having a high protein and fat content but a low carbohydrate level.
This shows that nutritional information was successfully incorporated into the similarity computations, allowing users with dietary choices or constraints to use the recommendations.
3)Visual Appeal: The system improves user engagement by presenting pictures next to ingredient lists and recipe names, enabling users to make better decisions based on visual cues.
The user experience is enhanced by the visual cues each image provides regarding the texture and presentation of the suggested dishes.

Summary of Results
The model's capacity to produce precise and pertinent suggestions that align with ingredient composition and nutritional re

quirements is validated by the sample output.
Scaled numerical characteristics for nutrition and TF-IDF for ingredients combine to form a comprehensive recommendation system that enables the model to accommodate a range of user preferences.
This example shows how adaptable and dependable the recommendation system is in providing tailored recipe recommendations, improving the user experience by meeting dietary and gastronomic requirements.

6. Discussion and Conclusion

Discussion

1. Ingredient-Based Similarity and TF-IDF Effectiveness
The flavor profile of any dish can be effectively captured through ingredient vectorization using Term Frequency-Inverse Document Frequency (TF-IDF).
Higher weights were given to ingredients like "smoked paprika" and "pork belly," which helped the program correctly identify recipes with comparable flavor components.
This method works particularly well for ingredient lists that contain both common and uncommon substances.
Recipes with similar ingredient profiles were successfully put together to produce suggestions that suit consumers' palates.
The suggestion quality could be further improved by capturing hierarchical relationships among ingredients, such as grouping similar spices, even if TF-IDF does a good job of managing relative ingredient importance.

2. Integration of Nutritional Features and Standardization
The recommendation system gains a useful dimension with the addition of nutritional data that has been normalized through scaling.
By matching recommendations with nutritional objectives, this feature enables the model to go beyond flavor preferences.
For example, individuals who favor high-protein or low-carb foods are more likely to get recipes that meet their dietary requirements.
This dual strategy increases the recommendation system's adaptability by fusing flavor and nutrition.
However, in similarity calculations, the algorithm currently treats all nutritional attributes equally.
Since certain users may place a higher value on particular nutritional characteristics (such reduced sodium) than others, this could not be in line with everyone's priorities.

3.Model Performance and Limitations In high-dimensional spaces that integrate standardized nutritional characteristics with TF-IDF vectors, the KNN model, which uses cosine similarity, performed well.
Despite variations in the amount or quantity of particular ingredients, cosine similarity—
which can detect angular distances—
proved useful for identifying recipes with comparable characteristics.
However, the computational efficiency of the KNN technique is limited, particularly as the dataset grows in size.
More effective methods, like Approximate Nearest Neighbors (ANN) or embeddings, can be needed for large datasets in order to preserve performance and lower processing needs.

4. User Experience and Visual Presentation
Including pictures of recipes in the recommendation display gives users more details, which improves the user experience in general.
When choosing recipes, consumers may find that visual clues like look, color, and texture are crucial.
An interesting and educational user interface is created by combining recipe names, ingredient listings, and nutritional information with pictures. However, how image-based analysis (such as examining the visual resemblance of recipes) might enhance suggestions is not currently covered in this research.
The system's capacity to suggest aesthetically pleasing recipes may be improved in subsequent iterations by utilizing computer vision algorithms on image data.

5. Potential Bias and Data Quality Considerations
The quality and organization of the dataset have a significant impact on the system's performance.
Missing values, inconsistent nutritional data, and inconsistent component naming rules can all add bias or lower the accuracy of recommendations.
A number of missing variables in the dataset used for this study were addressed by omission or imputation.
Model accuracy could be further increased, though, by refining data pretreatment procedures, such as standardizing ingredient lists or filling in missing nutritional information with domain-specific expertise.
Furthermore, if particular recipe types or cuisines are overrepresented, the algorithm may produce recommendations that are limited due to the biases inherent in the dataset.

6. Personalization and User-Specific Adjustments
Although the current approach provides a recommendation based on similarity that is one-size-fits-all, it is possible to further customize it by taking user feedback or history into account.

By including a feedback mechanism, the model might be able to learn from user interactions and modify recommendations dynamically in response to past recipe selections, personal preferences, or even health objectives.
More individualized recommendations might be made possible via collaborative filtering strategies or user-profile learning, which would eventually increase the system's sensitivity to user preferences.

Conclusion

Through the use of a KNN model, this study effectively illustrates a recipe recommendation system that integrates nutritional and ingredient-based aspects in a multifaceted manner.
A major improvement over single-dimensional recommendation systems, the system offers consumers recipe recommendations that suit their nutritional requirements and taste preferences.
The model can accommodate different dietary requirements by using TF-IDF for component similarity and standardized nutritional qualities, which improves user satisfaction.
Key findings of this research include:

- **Ingredient relevance** was effectively captured using TF-IDF, making it possible to identify similar recipes based on ingredient lists.
- **Nutritional alignment** allows users to explore recipes that match specific dietary profiles, a unique feature that distinguishes this system from purely ingredient-based recommendations.
- **Visual presentation** adds an aesthetic dimension, improving user engagement through images that provide a tangible preview of the recommended recipes.

Future Work

Several improvements could be made to the system in the future to make it even better:
1)Hierarchical Ingredient Clustering: To improve the accuracy of ingredient-based similarity computations, future research may organize components into families or clusters (e.g., related spices) to better capture flavor correlations.
2)Enhanced Computational Efficiency: By employing ANN techniques or embedding-based representations, this system can be scaled to handle larger datasets more skillfully while preserving recommendation quality.
3)Integration of Feedback and Personalized Recommendations: By integrating a feedback loop, the system might adapt to user preferences and gradually make recommendations that are more tailored to each individual.
Integration of Visual Similarity: By combining image analysis, recipes might be suggested based on how similar foods appear to one another. This would provide another level of relevancy based on visual attractiveness.

In conclusion, by systematically fusing taste, nutrition, and aesthetic appeal, the created recipe recommendation system exhibits encouraging outcomes.
It offers a flexible tool that satisfies users' nutritional and culinary requirements while preserving an intuitive user interface.
This technology has the potential to be a crucial component of individualized meal planning and culinary exploration with additional advancements.

7.References

Google scholar

kaggle