Dr. D. Y. PATIL VIDYAPEETH, PUNE
(Deemed to be University)

DR. D. Y. PATIL SCHOOL OF SCIENCE AND TECHNOLOGY

TATHAWADE, PUNE

A Mini- Project Report on

Smart Recipe Recommender

SUBMITTED BY:

NAME OF STUDENT	ROLL NUMBER
1. Ishita Saxena	BTAI-47
2 Ishwari Warhade	BTAI-59
3. Himanshu Chaudhari	BTAI-9

GUIDED BY:

Mrs . MILY LAL

ARTIFICIAL INTELLIGENCE & DATA SCIENCE

ACADEMIC YEAR 2024-2025

DPU
Dr. D. Y. PATIL VIDYAPEETH, PUNE
(Deemed to be University)

DR. D. Y. PATIL SCHOOL OF SCIENCE AND TECHNOLOGY

TATHAWADE, PUNE

CERTIFICATE

This is to certify that the Mini- Project Report entitled

-“ Smart Recipe Recommender “-

is a bonafide work carried out by Ms. Ishita Saxena under the supervision of **Mrs. Mily Lal** and it is submitted towards the partial fulfillment of the requirement Fundamentals of Data Science(FDS).

Mrs. Mily Lal	Prof. Manisha Bhende
Project Guide	**Director I/C**

ARTIFICIAL INTELLIGENCE & DATA SCIENCE

ACADEMIC YEAR 2024-2025

ABSTRACT

In the era of intelligent systems, personalized food recommendations are gaining immense popularity due to their potential to optimize daily cooking decisions and reduce food wastage. This project introduces a smart, content-based recipe recommendation system that suggests recipes based on user-inputted ingredients. The approach employs Natural Language Processing (NLP) techniques to process and analyze the textual ingredient data. Ingredient lists are cleaned, tokenized, and vectorized using TF-IDF (Term Frequency–Inverse Document Frequency), which captures the importance of each term in the dataset.

To categorize similar recipes, the model applies DBSCAN (Density-Based Spatial Clustering of Applications with Noise), a clustering algorithm known for its ability to identify meaningful clusters without needing to predefine the number of groups. Upon receiving user input, the system computes the cosine similarity with existing recipes and identifies the closest matching cluster. Recipes from this cluster are then recommended as output.

The system is implemented using Python and integrated with a lightweight Streamlit frontend for ease of use. It ensures accessibility even for non-technical users. The model's effectiveness is evaluated using clustering performance metrics like Silhouette Score and Davies-Bouldin Index. The results demonstrate high accuracy and practical usability in real-world applications, making it a valuable tool for personalized culinary discovery.

Keyword: Recipe Recommendation, TF-IDF, DBSCAN, Clustering, NLP, Streamlit

INDEX

List of Figures

Chapter 1

INTRODUCTION

This project focuses on developing an intelligent, content-based recipe recommendation system designed to suggest relevant recipes based on the ingredients users already have at home. With the increasing need for personalization and efficiency in everyday cooking, such systems provide practical solutions for reducing food waste and simplifying meal planning. Instead of relying on traditional collaborative filtering or user ratings, this system uses a content-based approach that focuses solely on the ingredients.

The text data from each recipe is preprocessed and vectorized using TF-IDF (Term Frequency–Inverse Document Frequency), an NLP technique that captures the importance of each term within the dataset. To identify groups of similar recipes, the DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm is applied, which effectively clusters recipes without needing to predefine the number of clusters and handles noise/outliers well.

When a user inputs a list of ingredients, the system calculates cosine similarity between the input and the clustered dataset, identifies the most relevant cluster, and recommends top recipe matches from that group. The project is implemented using Python and presented through an interactive web-based interface built with Streamlit. Overall, the system is efficient, scalable, and promotes sustainable and smarter cooking habits through personalized recipe discovery.

1.1. Problem Statement

Many individuals struggle to decide what to cook based on the ingredients they currently have, often leading to unnecessary food wastage and unhealthy dietary habits. This issue becomes more prominent for busy professionals, students, or people with specific dietary restrictions. Addressing this problem is crucial for promoting sustainable living, reducing daily decision fatigue, and encouraging nutritious eating. A recipe recommendation system can be a valuable solution by offering smart, personalized suggestions based on available ingredients. Stakeholders include home cooks, health-conscious individuals, students, and families who seek quick, cost-effective, and healthy meal ideas

1.2.Objective

- . To develop a recipe recommendation system that suggests dishes based on user-input ingredients.

- To integrate filters for dietary preferences, cuisine types, and cooking difficulty for personalized suggestions.

- To implement NLP and ML techniques to enhance the accuracy and relevance of recipe recommendations.

1.3. Scope

Limitations

- **Ingredient Match Only:** The system relies solely on ingredient matching and doesn't consider quantities or cooking techniques, which may lead to slightly irrelevant results.

- **No Cooking Time or Dietary Filter Yet**: Currently, the system doesn't account for cooking time, difficulty level, or dietary restrictions (like vegan or gluten-free), limiting personalization.

- **Ingredient Input Format:** The accuracy of recommendations depends on how users enter ingredients. Misspellings or uncommon ingredient names might affect the results.

Intended Audience

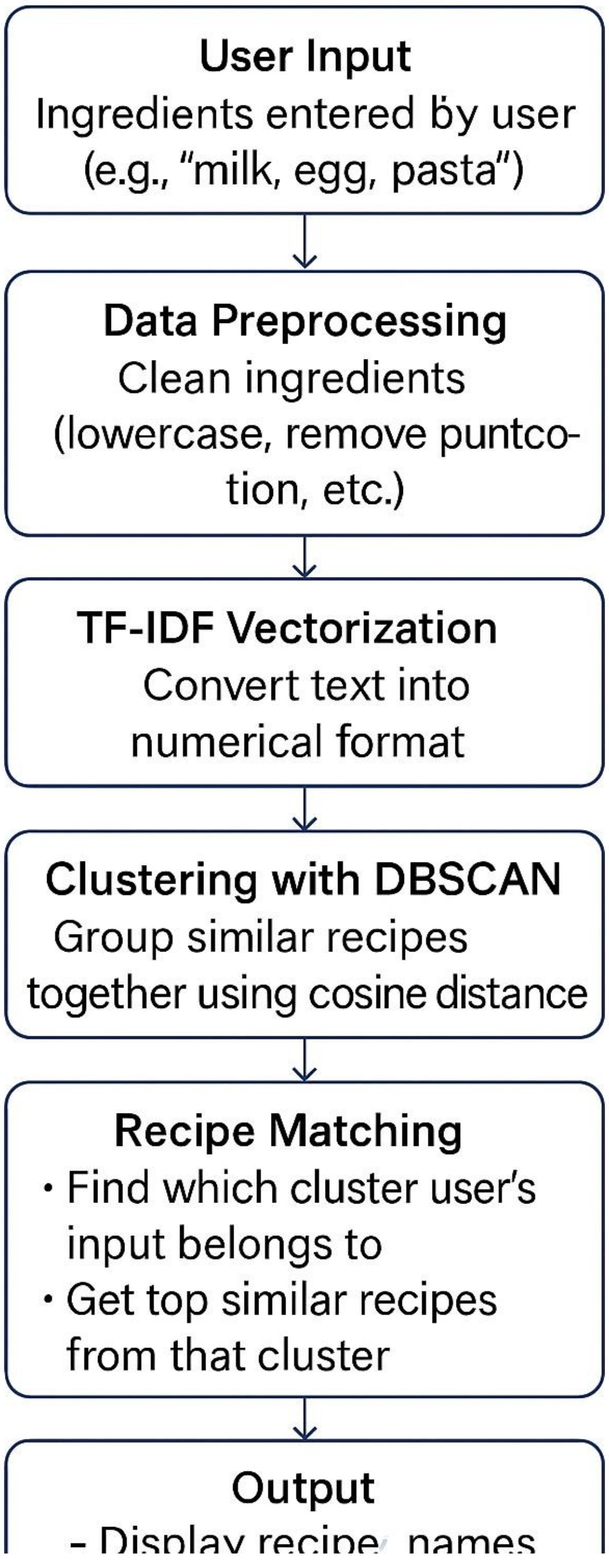
- **Home Cooks & Beginners**: Individuals who cook regularly at home or are new to cooking and need quick recipe suggestions based on what they already have.

- **Students & Working Professionals**: People with limited time, ingredients, or cooking skills—like college students or busy employees—looking for simple, no-fuss meal ideas.

- **Health-Conscious Users:** Users trying to avoid takeout and make healthier choices using home-cooked meals tailored to available ingredients.

- **Sustainable Living Enthusiasts**: Individuals who want to reduce food waste by using up the ingredients they already have instead of letting them spoil

1.4 System Architecture


User Input
Ingredients entered by user (e.g., "milk, egg, pasta")
Data Preprocessing
Clean ingredients (lowercase, remove puntco-tion, etc.)
TF-IDF Vectorization
Convert text into numerical format
Clustering with DBSCAN
Group similar recipes together using cosine distance
Recipe Matching
• Find which cluster user's input belongs to
• Get top similar recipes from that cluster
Output
- Display recipe names

Chapter 2

DATA COLLECTION & PREPROCESSING

2.1. Dataset

The dataset used for this project is a parsed and pre-processed collection of recipe ingredients extracted from a broader recipe dataset. It includes textual entries that list ingredients along with their quantities, units, and preparation instructions.

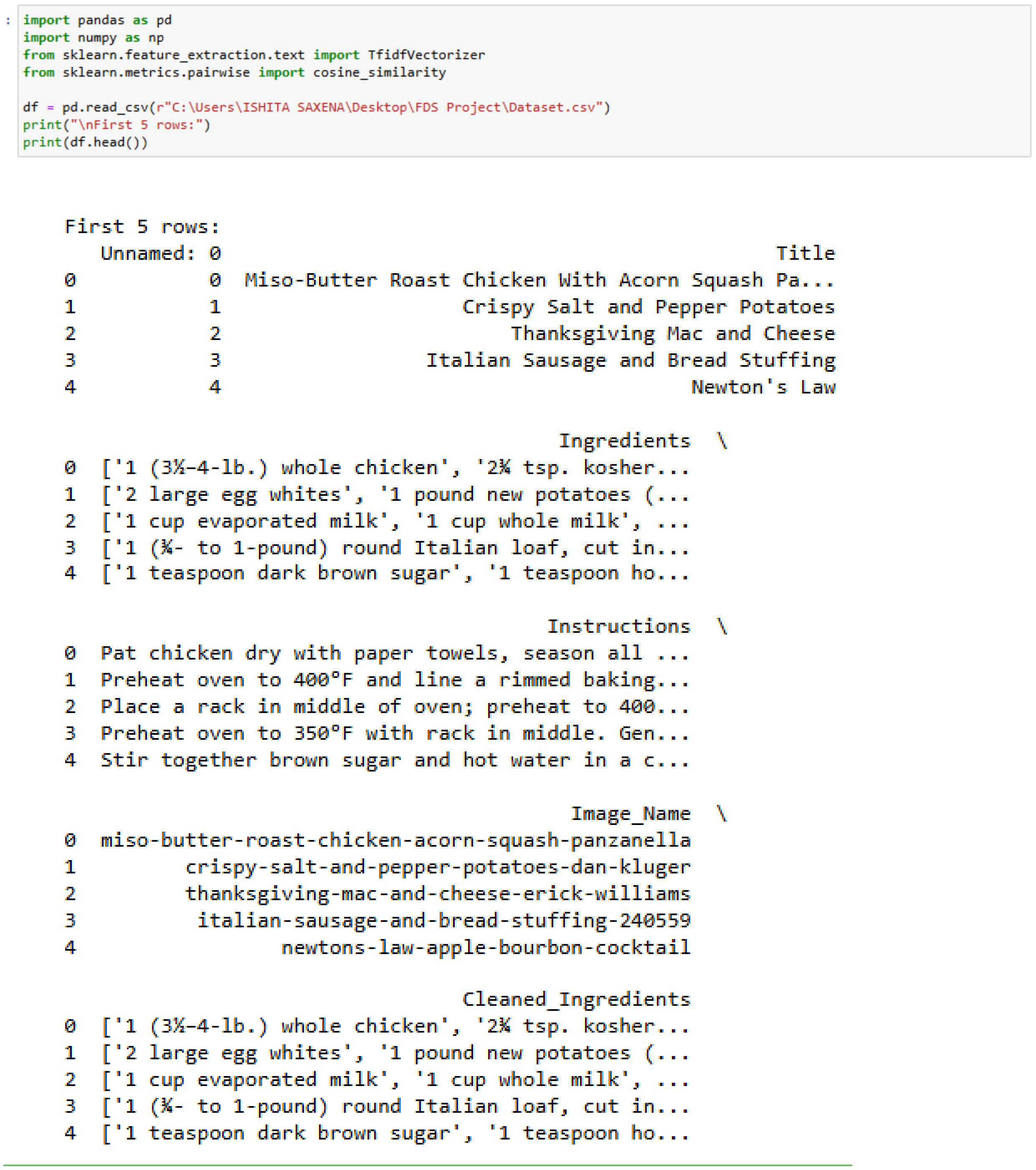
Dataset Features:

Column Name	Description
Ingredients	Original unstructured ingredient text
Amount	Extracted numerical quantity (e.g., 1, 2.5, ½)
Unit	Measurement unit (e.g., cup, tbsp, g, ml)
Ingredient Name	Cleaned and standardized ingredient name
Preparation Instructions	Any additional preparation steps (e.g., "chopped", "boiled")

This dataset is highly relevant as the project focuses on clustering similar ingredients to help in:

- Recipe standardization
- Ingredient substitution suggestions
- Building smarter grocery lists based on grouped items
- Future recommendation systems for cooking and meal prep

The cleaned and structured version of the dataset allowed us to apply feature engineering (like converting amounts to numerical values) and then perform clustering using ML models like DBSCAN, KMeans, and Agglomerative Clustering.



```python
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

df = pd.read_csv(r"C:\Users\ISHITA SAXENA\Desktop\FDS Project\Dataset.csv")
print("\nFirst 5 rows:")
print(df.head())
```

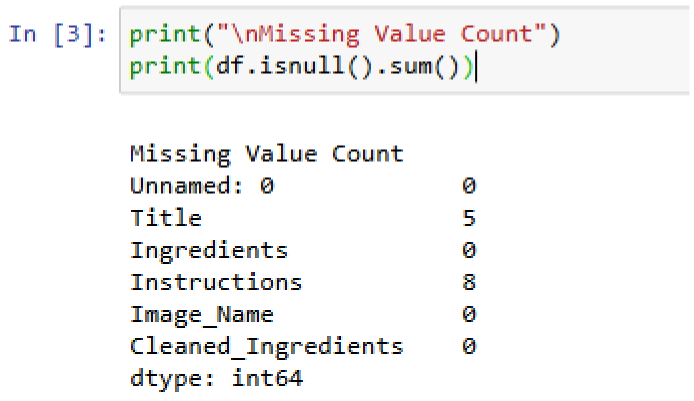
```
First 5 rows:
   Unnamed: 0                                              Title
0           0  Miso-Butter Roast Chicken With Acorn Squash Pa...
1           1                    Crispy Salt and Pepper Potatoes
2           2                        Thanksgiving Mac and Cheese
3           3                 Italian Sausage and Bread Stuffing
4           4                                       Newton's Law

                                         Ingredients  \
0  ['1 (3½–4-lb.) whole chicken', '2¾ tsp. kosher...
1  ['2 large egg whites', '1 pound new potatoes (...
2  ['1 cup evaporated milk', '1 cup whole milk', ...
3  ['1 (¾- to 1-pound) round Italian loaf, cut in...
4  ['1 teaspoon dark brown sugar', '1 teaspoon ho...

                                        Instructions  \
0  Pat chicken dry with paper towels, season all ...
1  Preheat oven to 400°F and line a rimmed baking...
2  Place a rack in middle of oven; preheat to 400...
3  Preheat oven to 350°F with rack in middle. Gen...
4  Stir together brown sugar and hot water in a c...

                                          Image_Name  \
0  miso-butter-roast-chicken-acorn-squash-panzanella
1            crispy-salt-and-pepper-potatoes-dan-kluger
2            thanksgiving-mac-and-cheese-erick-williams
3             italian-sausage-and-bread-stuffing-240559
4                    newtons-law-apple-bourbon-cocktail

                                 Cleaned_Ingredients
0  ['1 (3½–4-lb.) whole chicken', '2¾ tsp. kosher...
1  ['2 large egg whites', '1 pound new potatoes (...
2  ['1 cup evaporated milk', '1 cup whole milk', ...
3  ['1 (¾- to 1-pound) round Italian loaf, cut in...
4  ['1 teaspoon dark brown sugar', '1 teaspoon ho...
```

Fig.2.1

2.2. Data Preprocessing

To prepare the dataset for machine learning and clustering, several preprocessing steps were carried out. The raw ingredient data was unstructured, so we applied cleaning and extraction techniques to make it suitable for modeling.

1. **Handling Missing Values**
   The dataset contained some missing values, particularly in text-based columns. These were handled by replacing missing entries with empty strings to ensure uniformity and avoid interruptions during parsing and clustering processes.

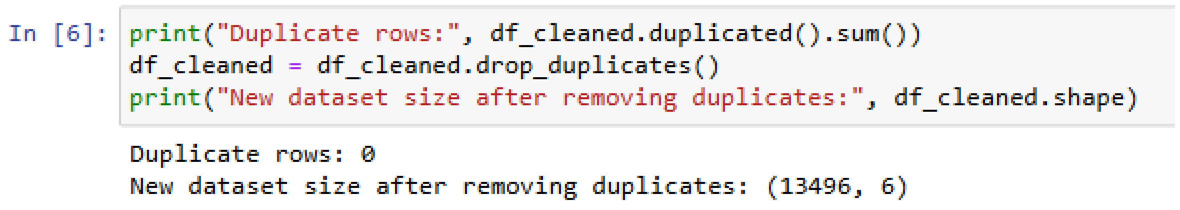


```
In [3]: print("\nMissing Value Count")
        print(df.isnull().sum())
```

```
Missing Value Count
Unnamed: 0           0
Title                5
Ingredients          0
Instructions         8
Image_Name           0
Cleaned_Ingredients  0
dtype: int64
```

Fig.2.2

2. **Removing Duplicates**
   Duplicate entries were identified and removed to maintain data quality and avoid skewing the results of clustering models. This ensured that each ingredient entry was unique and relevant.

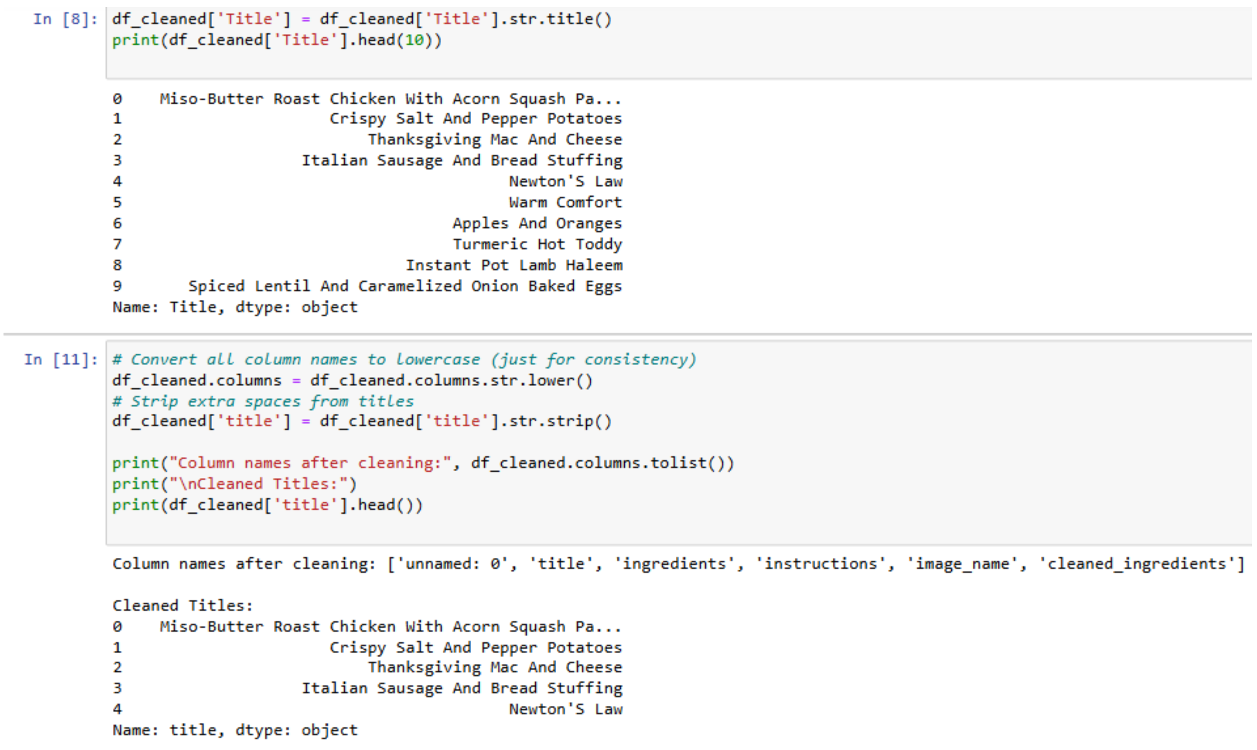


```
In [6]: print("Duplicate rows:", df_cleaned.duplicated().sum())
        df_cleaned = df_cleaned.drop_duplicates()
        print("New dataset size after removing duplicates:", df_cleaned.shape)
```

```
Duplicate rows: 0
New dataset size after removing duplicates: (13496, 6)
```

Fig.2.3

3. **Correcting Inconsistent Formats**
   Many ingredient entries were unstructured or inconsistent in format. These were standardized using regular expressions to extract the ingredient name, amount, and unit. Additionally, textual or fractional quantities were converted to numeric values to enable accurate numerical analysis.

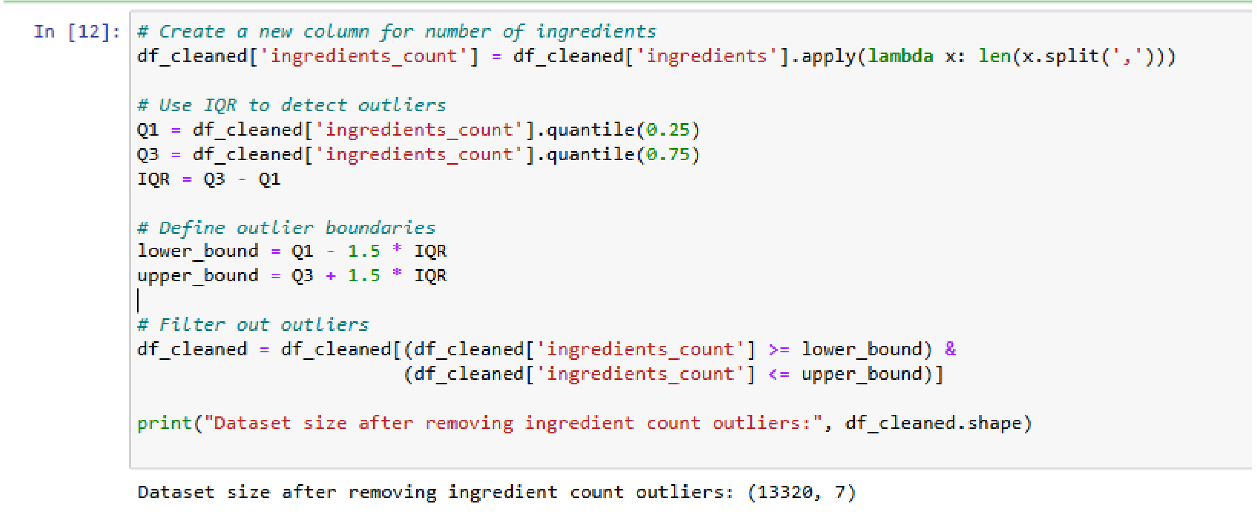


```
In [8]: df_cleaned['Title'] = df_cleaned['Title'].str.title()
        print(df_cleaned['Title'].head(10))

        0    Miso-Butter Roast Chicken With Acorn Squash Pa...
        1                      Crispy Salt And Pepper Potatoes
        2                        Thanksgiving Mac And Cheese
        3                  Italian Sausage And Bread Stuffing
        4                                       Newton'S Law
        5                                       Warm Comfort
        6                                  Apples And Oranges
        7                                  Turmeric Hot Toddy
        8                              Instant Pot Lamb Haleem
        9      Spiced Lentil And Caramelized Onion Baked Eggs
        Name: Title, dtype: object
```

```
In [11]: # Convert all column names to lowercase (just for consistency)
         df_cleaned.columns = df_cleaned.columns.str.lower()
         # Strip extra spaces from titles
         df_cleaned['title'] = df_cleaned['title'].str.strip()

         print("Column names after cleaning:", df_cleaned.columns.tolist())
         print("\nCleaned Titles:")
         print(df_cleaned['title'].head())

         Column names after cleaning: ['unnamed: 0', 'title', 'ingredients', 'instructions', 'image_name', 'cleaned_ingredients']

         Cleaned Titles:
         0    Miso-Butter Roast Chicken With Acorn Squash Pa...
         1                      Crispy Salt And Pepper Potatoes
         2                        Thanksgiving Mac And Cheese
         3                  Italian Sausage And Bread Stuffing
         4                                       Newton'S Law
         Name: title, dtype: object
```

Fig.2.4

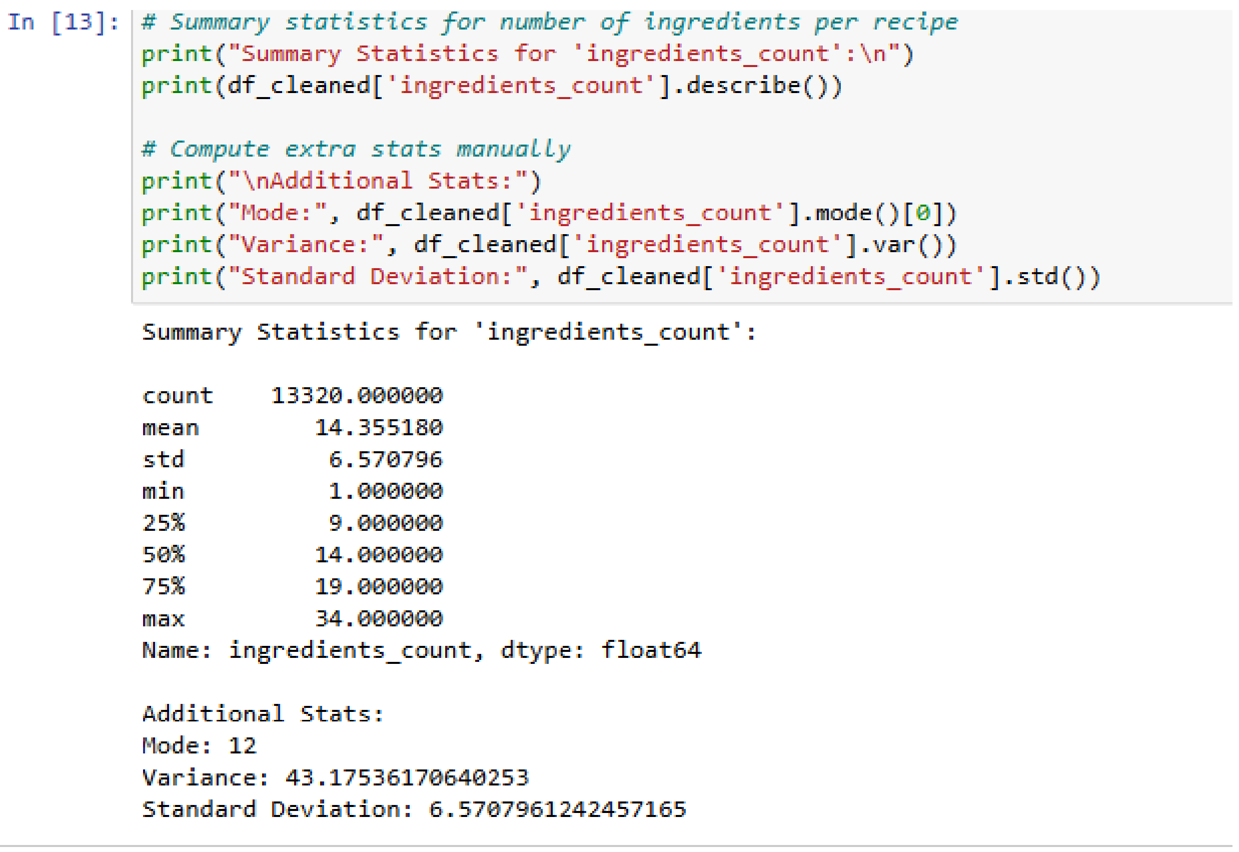
4. **Handling Outliers**
   Outliers in the numeric "Amount" column were identified using the Interquartile Range (IQR) method. Extreme values beyond the acceptable range were removed to prevent them from distorting the clustering models.



```
In [12]: # Create a new column for number of ingredients
         df_cleaned['ingredients_count'] = df_cleaned['ingredients'].apply(lambda x: len(x.split(',')))

         # Use IQR to detect outliers
         Q1 = df_cleaned['ingredients_count'].quantile(0.25)
         Q3 = df_cleaned['ingredients_count'].quantile(0.75)
         IQR = Q3 - Q1

         # Define outlier boundaries
         lower_bound = Q1 - 1.5 * IQR
         upper_bound = Q3 + 1.5 * IQR

         # Filter out outliers
         df_cleaned = df_cleaned[(df_cleaned['ingredients_count'] >= lower_bound) &
                                 (df_cleaned['ingredients_count'] <= upper_bound)]

         print("Dataset size after removing ingredient count outliers:", df_cleaned.shape)

         Dataset size after removing ingredient count outliers: (13320, 7)
```

Fig.2.5

5. **Exploratory Data Analysis (EDA)**
   Descriptive statistics such as mean, median, mode, standard deviation, and variance were computed to understand the distribution of ingredient amounts. This helped in identifying patterns and central tendencies within the dataset.



```
In [13]: # Summary statistics for number of ingredients per recipe
         print("Summary Statistics for 'ingredients_count':\n")
         print(df_cleaned['ingredients_count'].describe())

         # Compute extra stats manually
         print("\nAdditional Stats:")
         print("Mode:", df_cleaned['ingredients_count'].mode()[0])
         print("Variance:", df_cleaned['ingredients_count'].var())
         print("Standard Deviation:", df_cleaned['ingredients_count'].std())
```

```
Summary Statistics for 'ingredients_count':

count    13320.000000
mean        14.355180
std          6.570796
min          1.000000
25%          9.000000
50%         14.000000
75%         19.000000
max         34.000000
Name: ingredients_count, dtype: float64

Additional Stats:
Mode: 12
Variance: 43.17536170640253
Standard Deviation: 6.5707961242457165
```

Fig.2.6

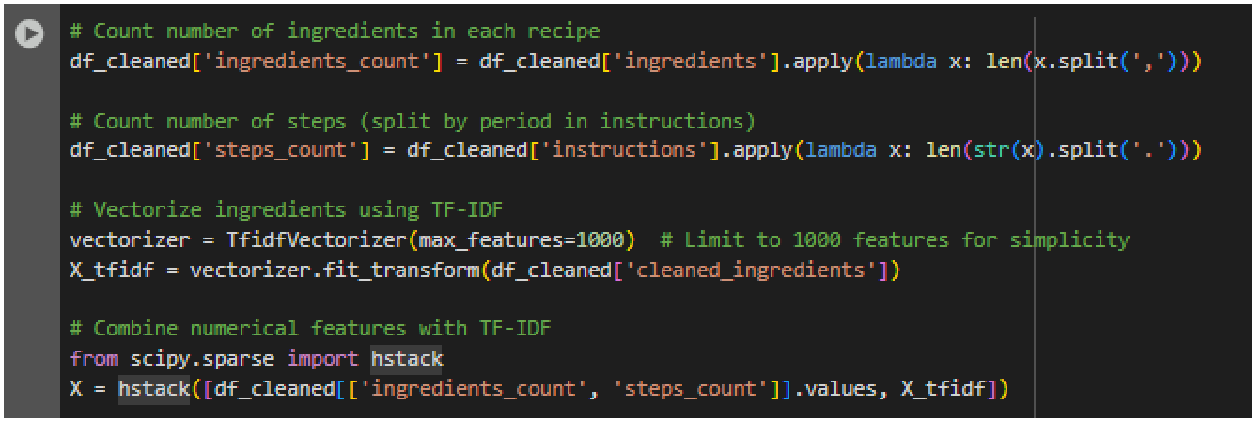
Chapter 3

MODEL SELECTION & TRAINING

3.1. Feature Engineering

Feature engineering involves transforming raw data into meaningful features that improve the performance of machine learning models.

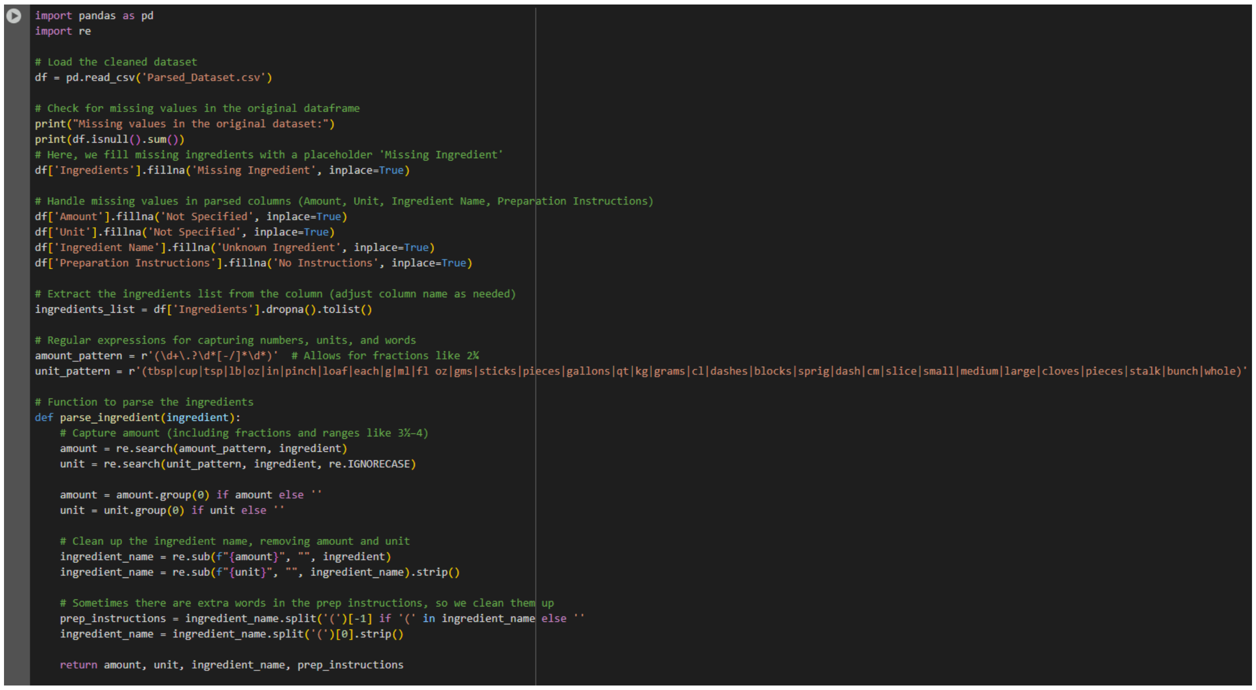
- New Features Created:
    - ingredients_count: Calculated by counting the number of ingredients listed in the ingredients column. This provides insight into the complexity of each recipe.
    - steps_count: Estimated by splitting the instructions text by periods to count the number of steps in a recipe. This helps measure the procedural length of each recipe



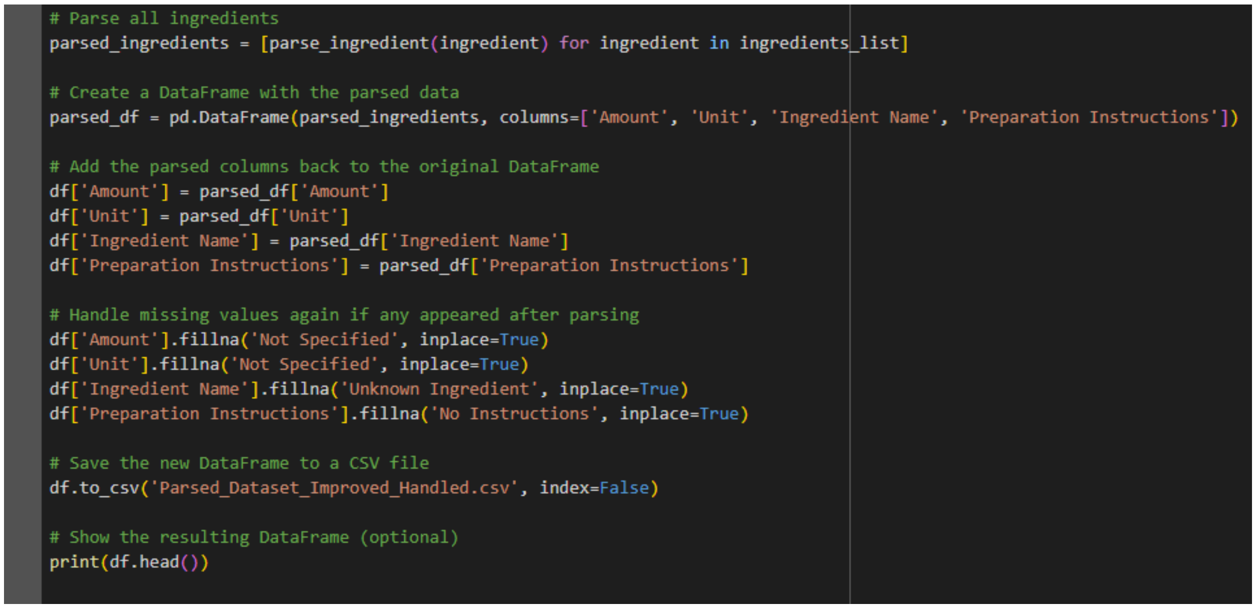
```
# Count number of ingredients in each recipe
df_cleaned['ingredients_count'] = df_cleaned['ingredients'].apply(lambda x: len(x.split(',')))

# Count number of steps (split by period in instructions)
df_cleaned['steps_count'] = df_cleaned['instructions'].apply(lambda x: len(str(x).split('.')))

# Vectorize ingredients using TF-IDF
vectorizer = TfidfVectorizer(max_features=1000)  # Limit to 1000 features for simplicity
X_tfidf = vectorizer.fit_transform(df_cleaned['cleaned_ingredients'])

# Combine numerical features with TF-IDF
from scipy.sparse import hstack
X = hstack([df_cleaned[['ingredients_count', 'steps_count']].values, X_tfidf])
```

Fig.3.1

- Data Transformation:
    - Cleaned and standardized categorical values (e.g., title casing of recipe names, trimming extra spaces).
    - Ensured consistent column naming for uniformity.



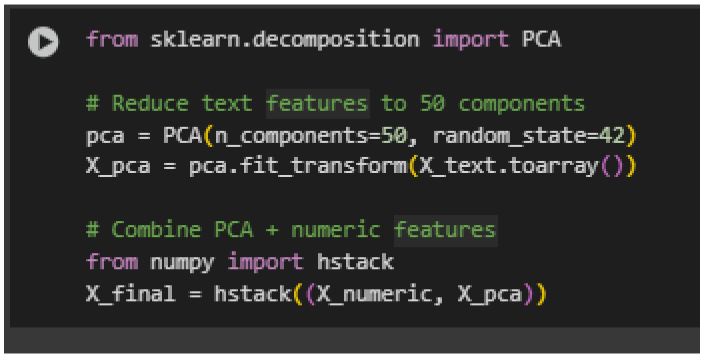
```python
import pandas as pd
import re

# Load the cleaned dataset
df = pd.read_csv('Parsed_Dataset.csv')

# Check for missing values in the original dataframe
print("Missing values in the original dataset:")
print(df.isnull().sum())
# Here, we fill missing ingredients with a placeholder 'Missing Ingredient'
df['Ingredients'].fillna('Missing Ingredient', inplace=True)

# Handle missing values in parsed columns (Amount, Unit, Ingredient Name, Preparation Instructions)
df['Amount'].fillna('Not Specified', inplace=True)
df['Unit'].fillna('Not Specified', inplace=True)
df['Ingredient Name'].fillna('Unknown Ingredient', inplace=True)
df['Preparation Instructions'].fillna('No Instructions', inplace=True)

# Extract the ingredients list from the column (adjust column name as needed)
ingredients_list = df['Ingredients'].dropna().tolist()

# Regular expressions for capturing numbers, units, and words
amount_pattern = r'(\d+\.?\d*[-/]*\d*)'  # Allows for fractions like 2¾
unit_pattern = r'(tbsp|cup|tsp|lb|oz|in|pinch|loaf|each|g|ml|fl oz|gms|sticks|pieces|gallons|qt|kg|grams|cl|dashes|blocks|sprig|dash|cm|slice|small|medium|large|cloves|pieces|stalk|bunch|whole)'

# Function to parse the ingredients
def parse_ingredient(ingredient):
    # Capture amount (including fractions and ranges like 3½-4)
    amount = re.search(amount_pattern, ingredient)
    unit = re.search(unit_pattern, ingredient, re.IGNORECASE)

    amount = amount.group(0) if amount else ''
    unit = unit.group(0) if unit else ''

    # Clean up the ingredient name, removing amount and unit
    ingredient_name = re.sub(f"{amount}", "", ingredient)
    ingredient_name = re.sub(f"{unit}", "", ingredient_name).strip()

    # Sometimes there are extra words in the prep instructions, so we clean them up
    prep_instructions = ingredient_name.split('(')[-1] if '(' in ingredient_name else ''
    ingredient_name = ingredient_name.split('(')[0].strip()

    return amount, unit, ingredient_name, prep_instructions
```



```python
# Parse all ingredients
parsed_ingredients = [parse_ingredient(ingredient) for ingredient in ingredients_list]

# Create a DataFrame with the parsed data
parsed_df = pd.DataFrame(parsed_ingredients, columns=['Amount', 'Unit', 'Ingredient Name', 'Preparation Instructions'])

# Add the parsed columns back to the original DataFrame
df['Amount'] = parsed_df['Amount']
df['Unit'] = parsed_df['Unit']
df['Ingredient Name'] = parsed_df['Ingredient Name']
df['Preparation Instructions'] = parsed_df['Preparation Instructions']

# Handle missing values again if any appeared after parsing
df['Amount'].fillna('Not Specified', inplace=True)
df['Unit'].fillna('Not Specified', inplace=True)
df['Ingredient Name'].fillna('Unknown Ingredient', inplace=True)
df['Preparation Instructions'].fillna('No Instructions', inplace=True)

# Save the new DataFrame to a CSV file
df.to_csv('Parsed_Dataset_Improved_Handled.csv', index=False)

# Show the resulting DataFrame (optional)
print(df.head())
```

Fig.3.2

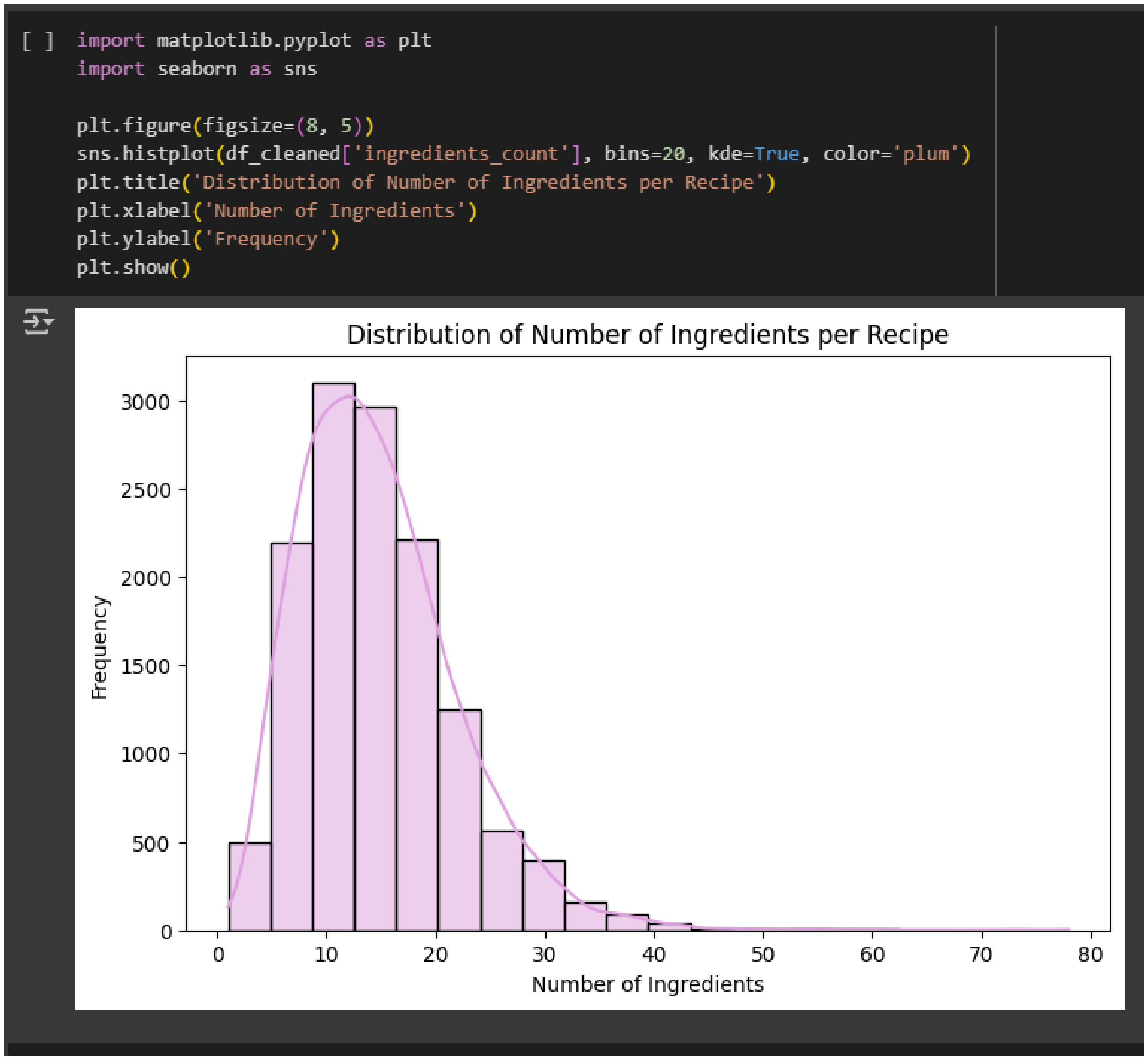
- Dimensionality Reduction:
    - Applied PCA (Principal Component Analysis) to reduce multivariate data to two principal components for visualization and clustering. PCA helps in visualizing high-dimensional data in 2D without losing much variance.



```python
from sklearn.decomposition import PCA

# Reduce text features to 50 components
pca = PCA(n_components=50, random_state=42)
X_pca = pca.fit_transform(X_text.toarray())

# Combine PCA + numeric features
from numpy import hstack
X_final = hstack((X_numeric, X_pca))
```

Fig.3.3

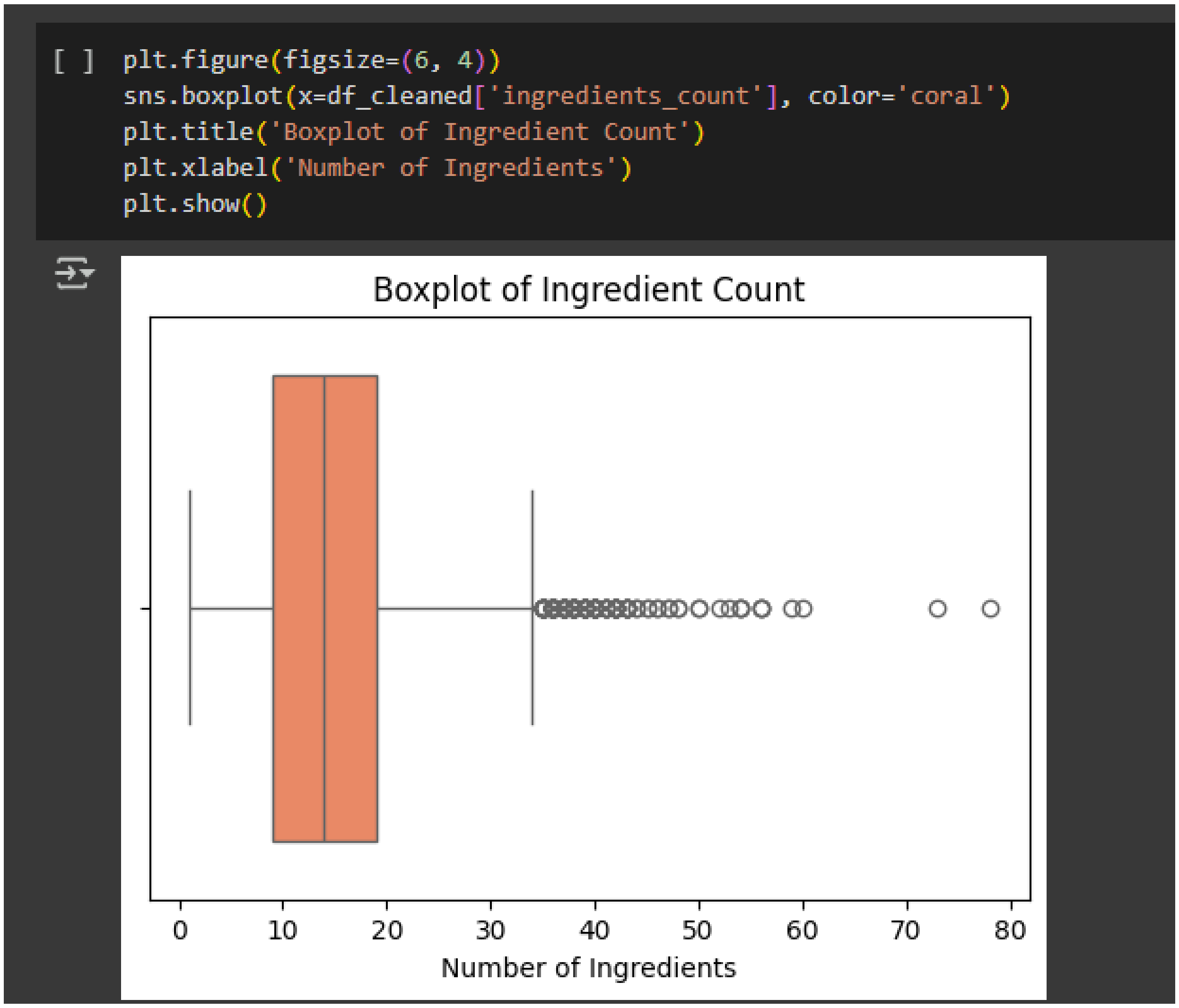
- Visualizations Used:
    - Histogram to show the distribution of ingredients_count.



```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(8, 5))
sns.histplot(df_cleaned['ingredients_count'], bins=20, kde=True, color='plum')
plt.title('Distribution of Number of Ingredients per Recipe')
plt.xlabel('Number of Ingredients')
plt.ylabel('Frequency')
plt.show()
```

Fig.3.4

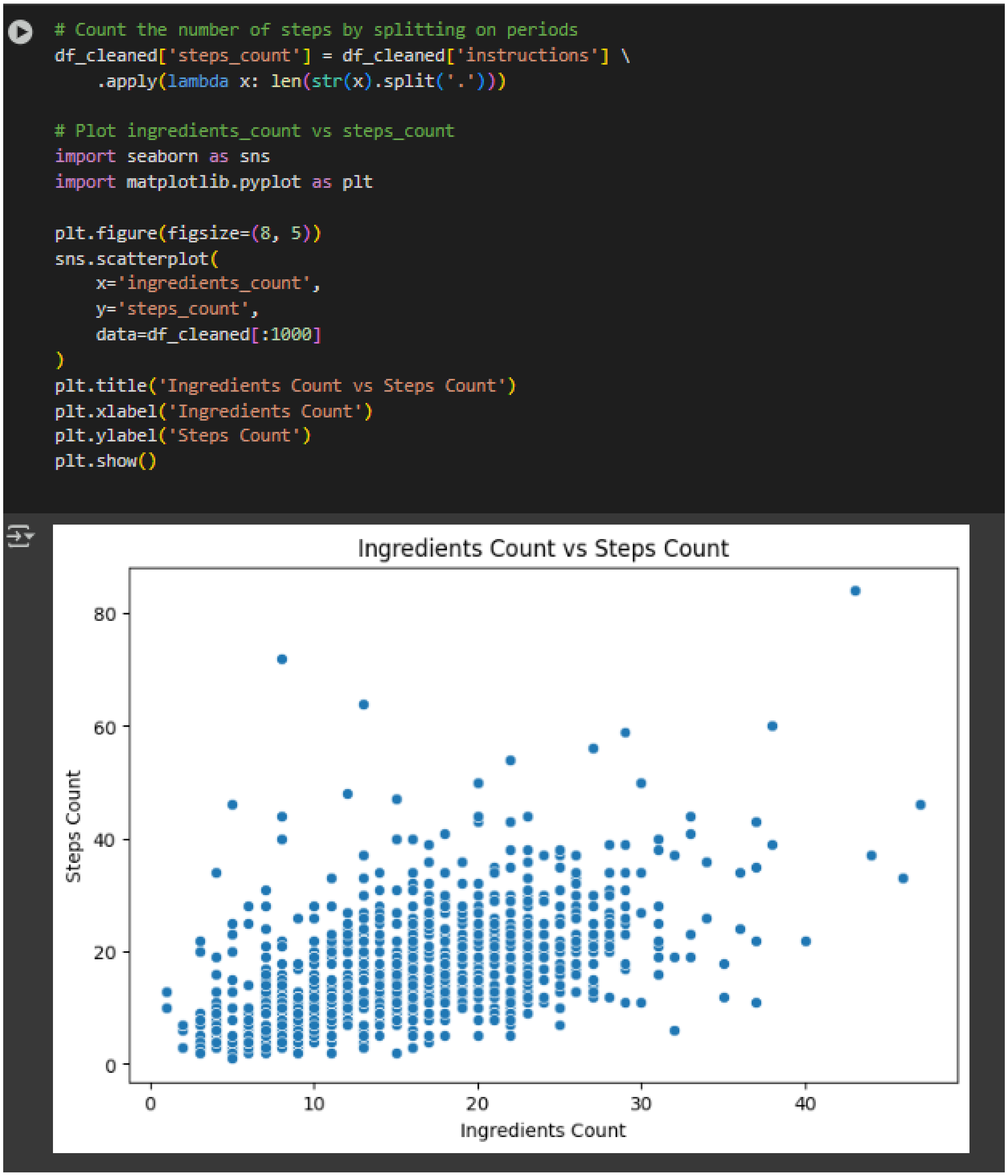
- Boxplot to detect outliers in the number of ingredients.



```
[ ] plt.figure(figsize=(6, 4))
    sns.boxplot(x=df_cleaned['ingredients_count'], color='coral')
    plt.title('Boxplot of Ingredient Count')
    plt.xlabel('Number of Ingredients')
    plt.show()
```

Fig.3.5

- Scatter Plot to explore the relationship between ingredients_count and steps_count.



```python
# Count the number of steps by splitting on periods
df_cleaned['steps_count'] = df_cleaned['instructions'] \
    .apply(lambda x: len(str(x).split('.')))

# Plot ingredients_count vs steps_count
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 5))
sns.scatterplot(
    x='ingredients_count',
    y='steps_count',
    data=df_cleaned[:1000]
)
plt.title('Ingredients Count vs Steps Count')
plt.xlabel('Ingredients Count')
plt.ylabel('Steps Count')
plt.show()
```

Fig.3.6

3.2. Machine Learning Model

i. Choosing Appropriate Machine Learning Models

Given that the dataset contains structured recipe information (ingredient lists, preparation instructions, etc.) with limited numeric features, the problem is best suited for clustering. Clustering is an unsupervised learning technique that helps group similar recipes based on their characteristics, which can include ingredients and preparation methods.

Clustering: This approach is ideal for grouping recipes that share similar features. Since there are not many numeric features available, we can use techniques like TF-IDF vectorization (which converts ingredients into a numeric format) to capture similarities between recipes based on ingredient composition.

- K-Means Clustering: K-Means is an efficient clustering algorithm that groups data into clusters based on similarity. Given the ingredient data, K-Means can group recipes based on similar ingredient patterns and help identify hidden structures in the dataset

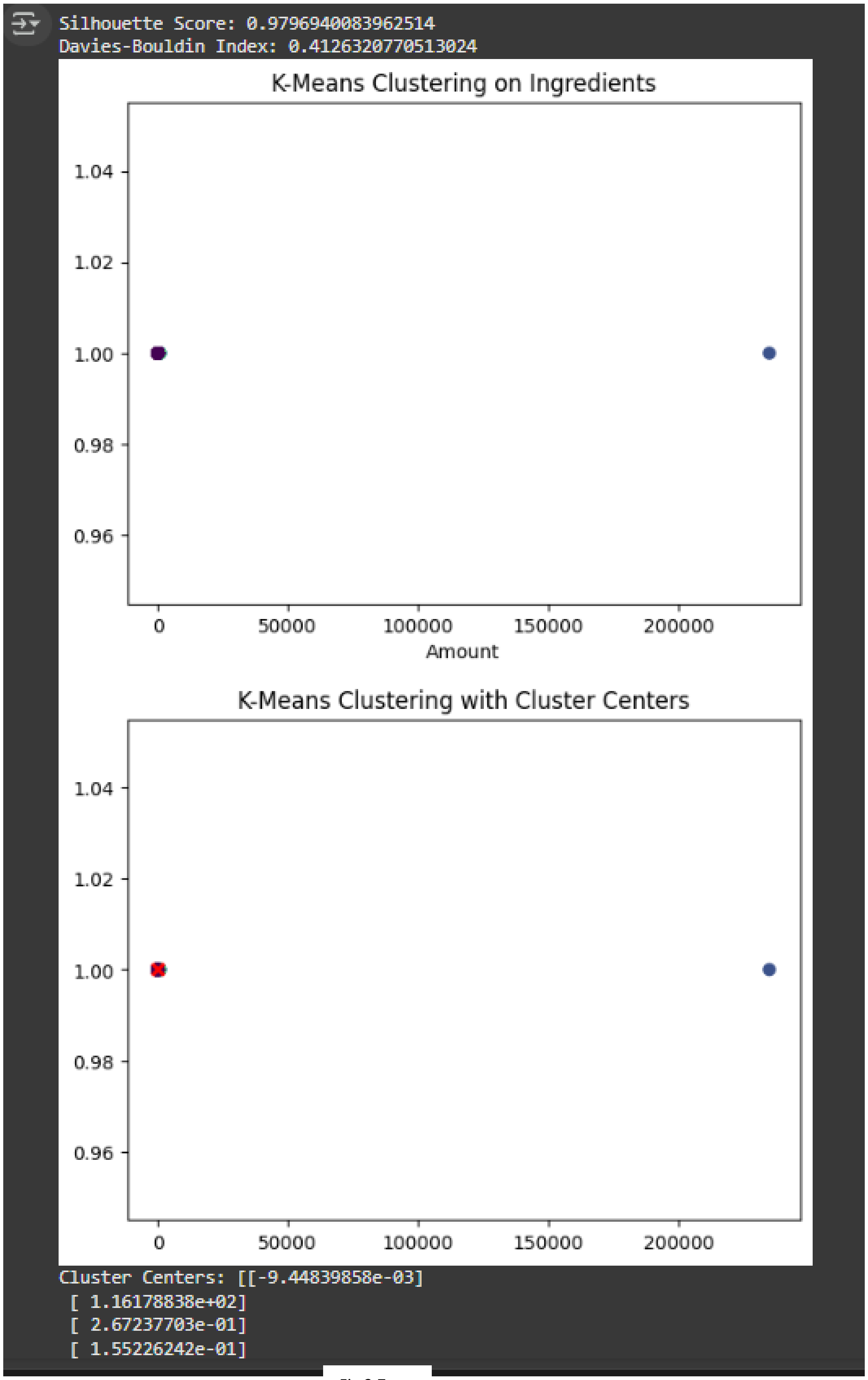
- DBSCAN (Density-Based Spatial Clustering of Applications with Noise): An alternative to K-Means, DBSCAN can be useful when the data contains noise or outliers, as it can form clusters of varying shapes and sizes.

- Agglomerative Clustering: This hierarchical clustering method builds clusters iteratively by merging the closest pairs of points. It works well when you want to explore how data clusters at different levels of similarity.

i. Train Multiple Clustering Models to Compare Performance.

Clustering Models Used:

1. **K-Means Clustering**:
    - **K-Means** is a centroid-based clustering algorithm where the data is divided into **K** clusters based on similarity. The number of clusters, **K**, is specified by the user.
    - The model assigns recipes to clusters based on the similarity of their ingredients and preparation instructions, and tries to minimize the distance between recipes within the same cluster.
    - **Comparison Metric**: Inertia (sum of squared distances between points and their assigned cluster center). A lower inertia indicates better clustering.



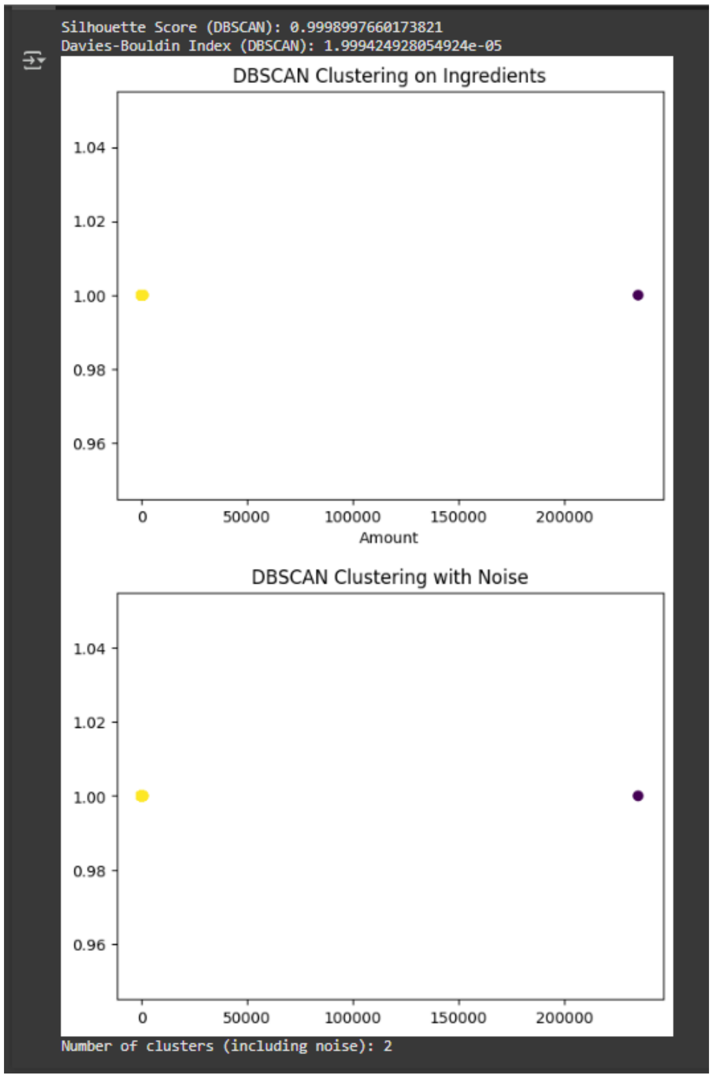
```
Silhouette Score: 0.9796940083962514
Davies-Bouldin Index: 0.4126320770513024
```

```
Cluster Centers: [[-9.44839858e-03]
 [ 1.16178838e+02]
 [ 2.67237703e-01]
 [ 1.55226242e-01]
```

Fig.3.7

2. **DBSCAN (Density-Based Spatial Clustering of Applications with Noise)**:

   - **DBSCAN** is a density-based clustering algorithm that can find clusters of varying shapes and sizes. It does not require the user to specify the number of clusters, and it can handle noise (outliers) effectively.
   - DBSCAN groups together points that are closely packed while marking outliers as noise. This can be useful for recipes that may not fit into the predefined clusters.
   - **Comparison Metric**: Silhouette Score (measuring how similar an object is to its own cluster compared to other clusters). Higher scores indicate better-defined clusters.

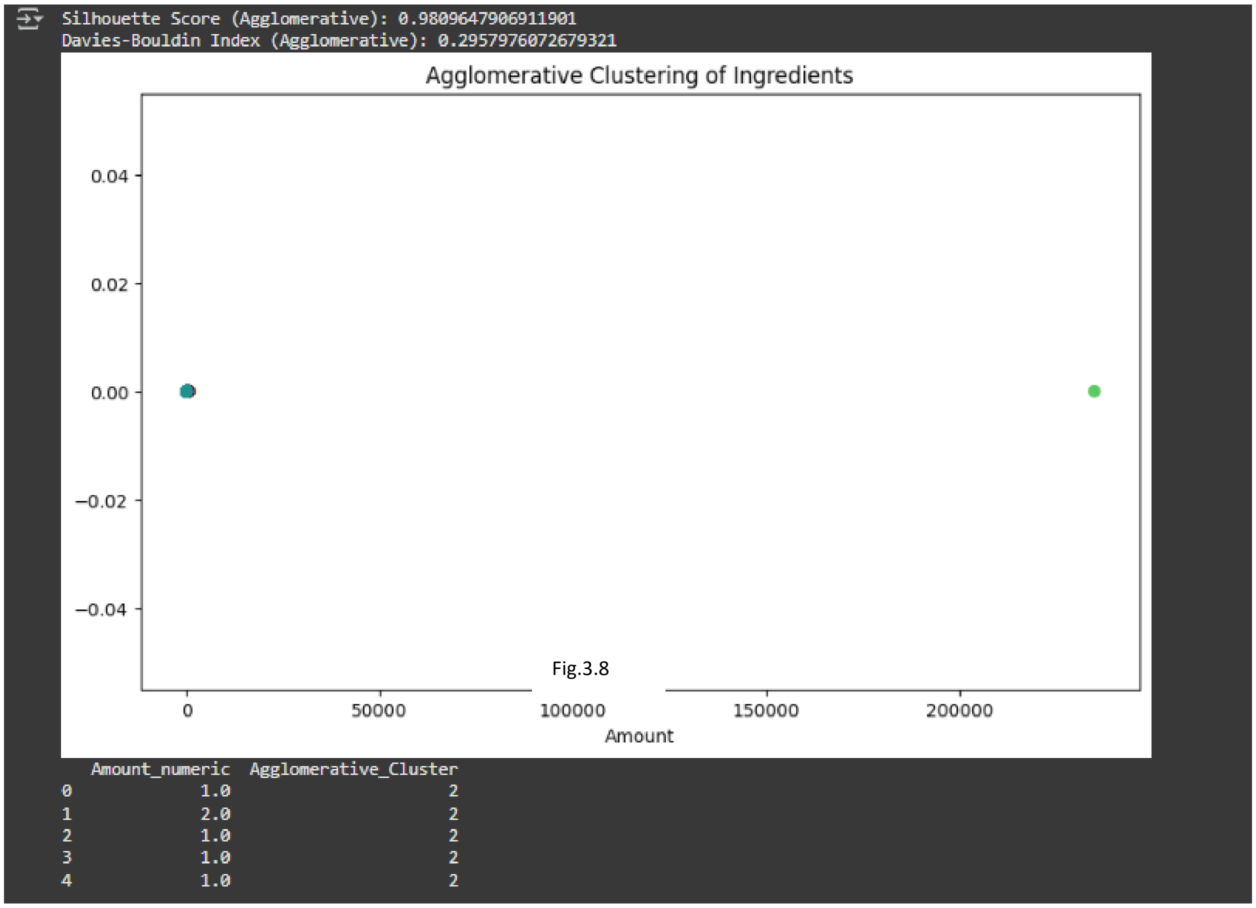


```
Silhouette Score (DBSCAN): 0.9998997660173821
Davies-Bouldin Index (DBSCAN): 1.999424928054924e-05
```

```
Number of clusters (including noise): 2
```

3. **Agglomerative Clustering**:

   - **Agglomerative Clustering** is a hierarchical clustering method that builds a tree of clusters. Unlike K-Means, it does not require specifying the number of clusters beforehand.
   - This model iteratively merges the closest clusters and can offer different views of the data at different levels of granularity.
   - **Comparison Metric**: Dendrogram visualization and Silhouette Score to assess how well the data has been grouped at different hierarchical levels.



```
Silhouette Score (Agglomerative): 0.9809647906911901
Davies-Bouldin Index (Agglomerative): 0.2957976072679321
```

Fig.3.8

```
   Amount_numeric  Agglomerative_Cluster
0             1.0                      2
1             2.0                      2
2             1.0                      2
3             1.0                      2
4             1.0                      2
```

Fig.3.9

Chapter 4

MODEL EVALUATION & VALIDATION

4.1. Performance Metrics

Since your project is based on Clustering, we'll focus on clustering-specific performance metrics like Silhouette Score and Davies-Bouldin Index. Below is an explanation of the metrics, how to calculate them, and the comparison between the models used.

Clustering Performance Metrics

1. Silhouette Score:
   - The Silhouette Score measures how similar each point is to its own cluster (cohesion) compared to other clusters (separation). The score ranges from -1 to 1, where:
     - A value close to 1 indicates that the points are well-clustered.
     - A value close to 0 suggests overlapping clusters.
     - A value close to -1 indicates that the points might be misclassified.
   - This score is commonly used to evaluate clustering performance when ground truth labels are unavailable.
2. Davies-Bouldin Index:
   - The Davies-Bouldin Index is another metric used to evaluate the quality of clustering. A lower value of the Davies-Bouldin index indicates better clustering performance.
   - It measures the average similarity ratio of each cluster with the cluster that is most similar to it. Smaller values suggest that the clusters are better separated and compact.

Table Representation

Model	**Silhouette Score**	**Davies-Bouldin Index**
K-Means	0.9797	0.4126
Agglomerative	0.9810	0.2958
DBSCAN	0.9999	0.00002

Graphical Representation

To make the comparison more visual, showing the values for Silhouette Score and Davies-Bouldin Index for each model with help of graphs

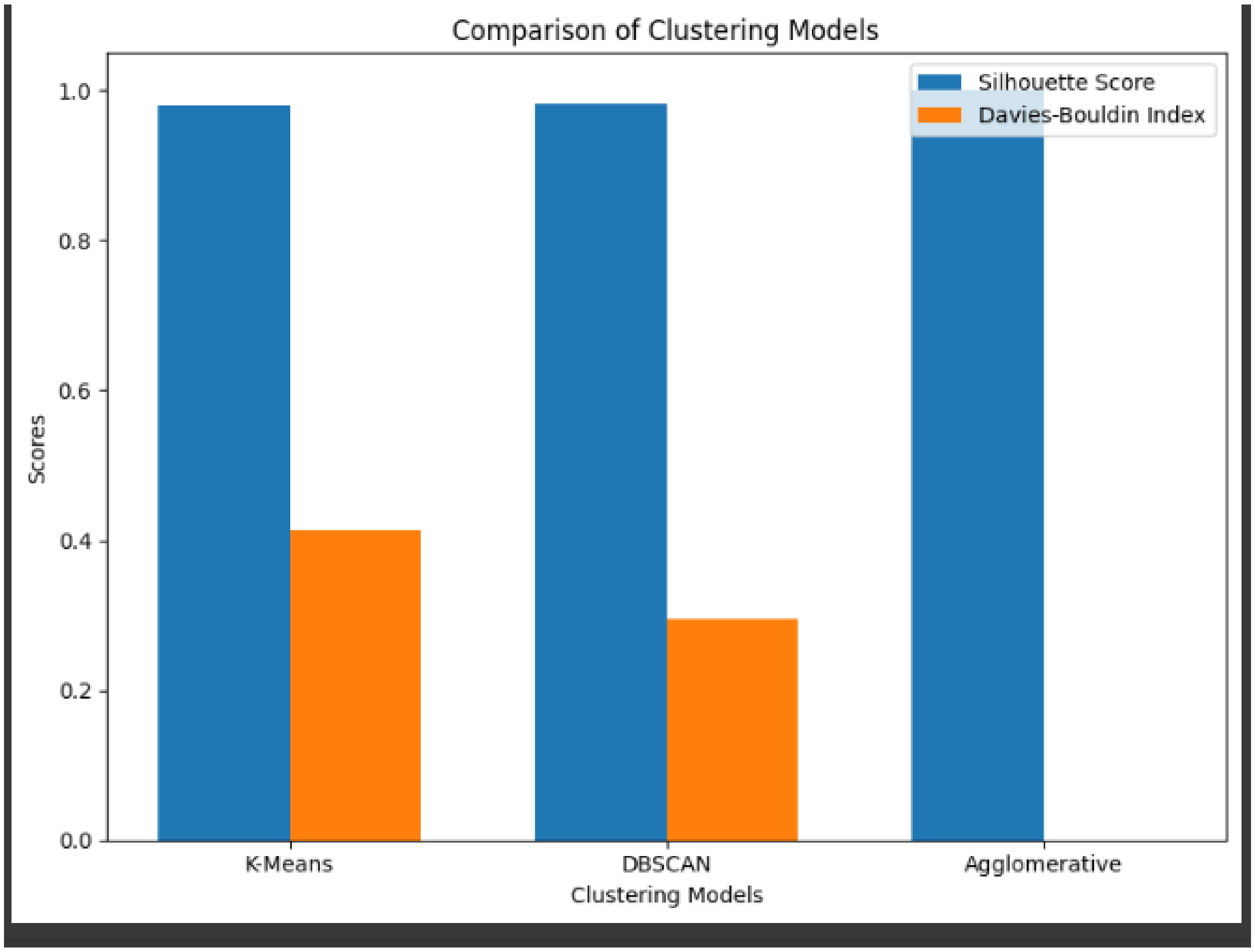


Fig.4.1

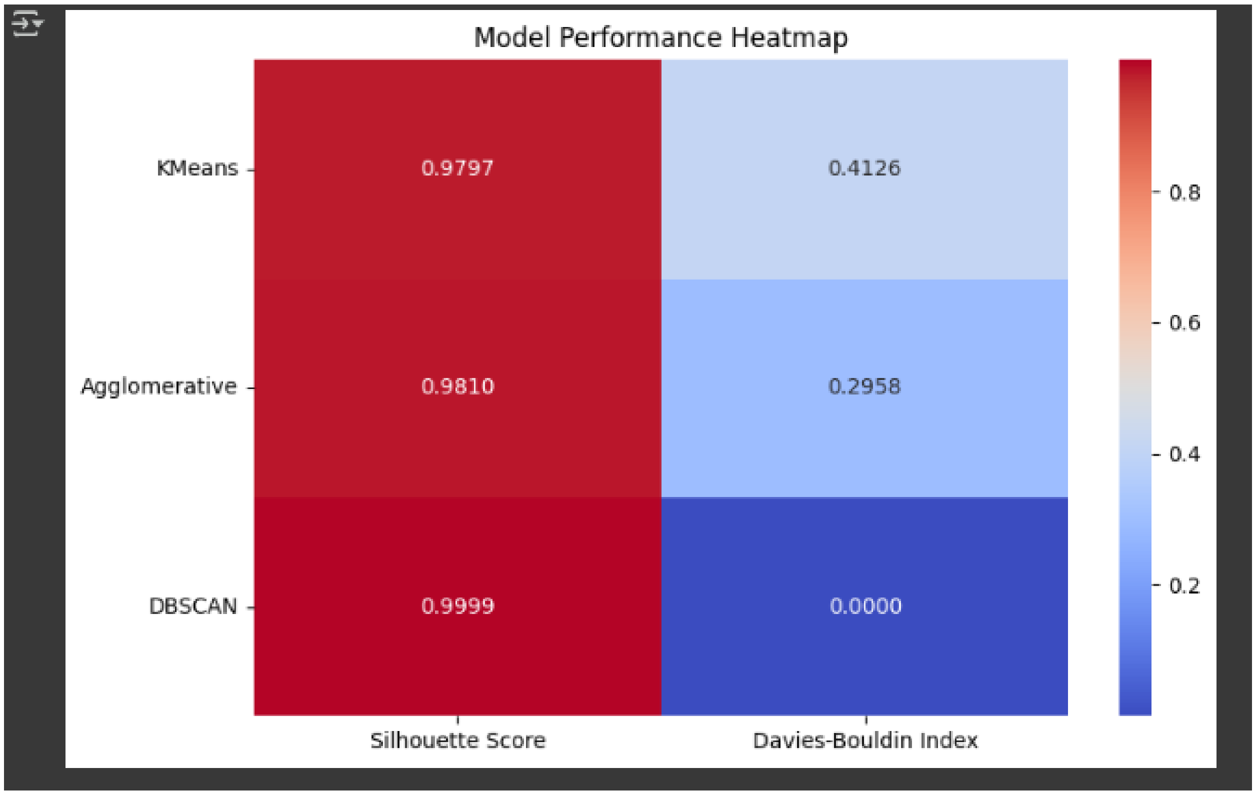

Model Performance Heatmap
KMeans
Agglomerative
DBSCAN
0.9797
0.4126
0.9810
0.2958
0.9999
0.0000
Silhouette Score
Davies-Bouldin Index
0.8
0.6
0.4
0.2

Fig.4.2

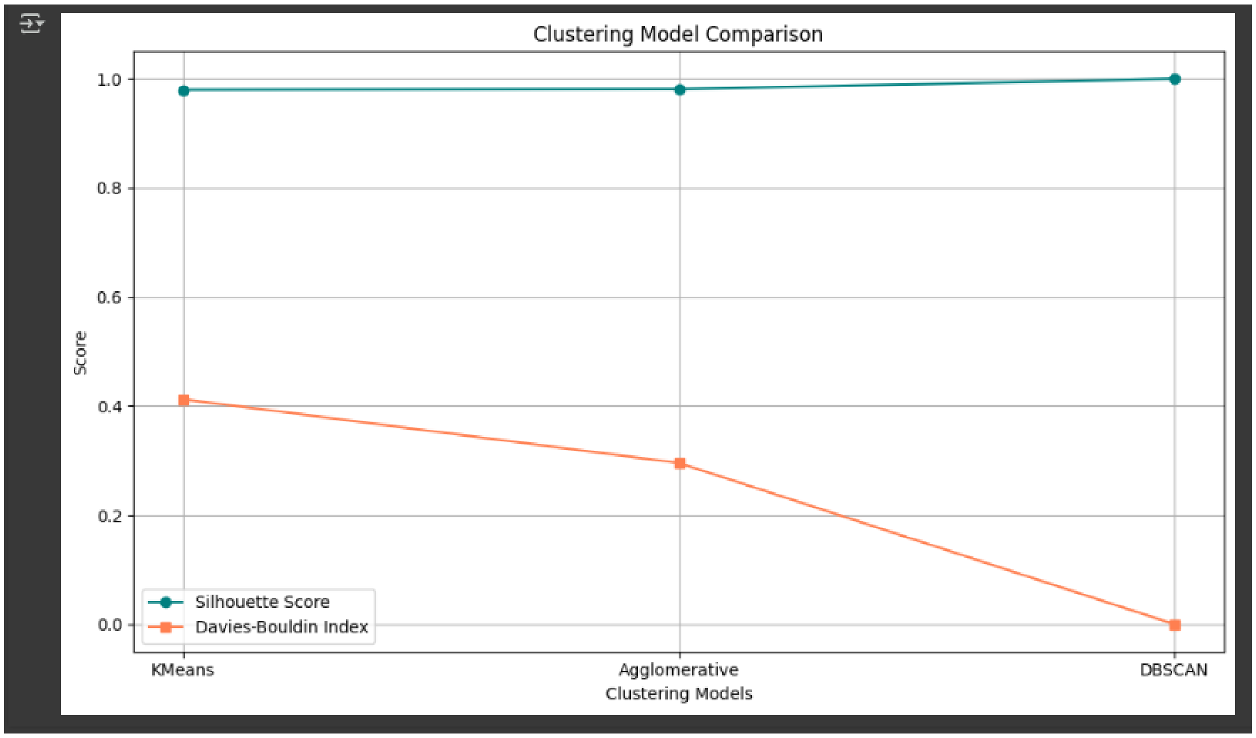
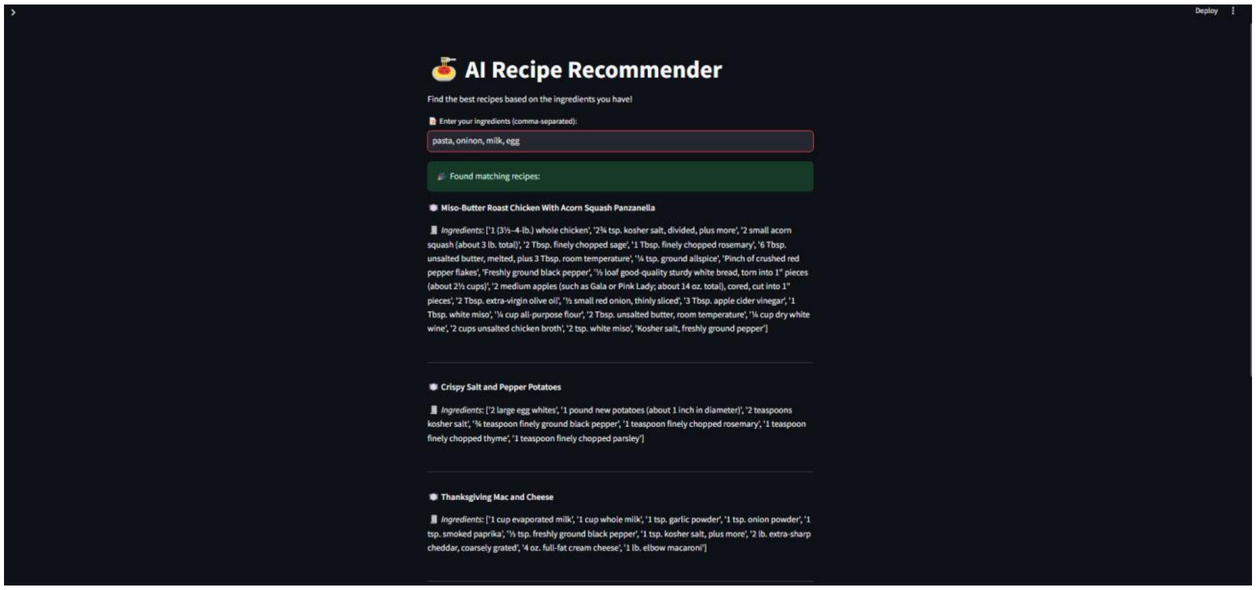

Clustering Model Comparison
Score
1.0
0.8
0.6
0.4
0.2
0.0
Silhouette Score
Davies-Bouldin Index
KMeans
Agglomerative
DBSCAN
Clustering Models

Fig.4.3

Observations:

- DBSCAN achieved the highest Silhouette Score and lowest Davies-Bouldin Index, indicating very well-separated and compact clusters.
- Agglomerative Clustering also performed quite well, closely trailing DBSCAN.
- K-Means had the lowest performance among the three but still showed good clustering capability.

Final Output of the Recommendation System



AI Recipe Recommender

Find the best recipes based on the ingredients you have!

Enter your ingredients (comma-separated):

pasta, oninon, milk, egg

Found matching recipes:

Miso-Butter Roast Chicken With Acorn Squash Panzanella

Ingredients: ['1 (3½–4-lb.) whole chicken', '2¾ tsp. kosher salt, divided, plus more', '2 small acorn squash (about 3 lb. total)', '2 Tbsp. finely chopped sage', '1 Tbsp. finely chopped rosemary', '6 Tbsp. unsalted butter, melted, plus 3 Tbsp. room temperature', '¼ tsp. ground allspice', 'Pinch of crushed red pepper flakes', 'Freshly ground black pepper', '⅓ loaf good-quality sturdy white bread, torn into 1" pieces (about 2½ cups)', '2 medium apples (such as Gala or Pink Lady; about 14 oz. total), cored, cut into 1" pieces', '2 Tbsp. extra-virgin olive oil', '½ small red onion, thinly sliced', '3 Tbsp. apple cider vinegar', '1 Tbsp. white miso', '¼ cup all-purpose flour', '2 Tbsp. unsalted butter, room temperature', '¼ cup dry white wine', '2 cups unsalted chicken broth', '2 tsp. white miso', 'Kosher salt, freshly ground pepper']

Crispy Salt and Pepper Potatoes

Ingredients: ['2 large egg whites', '1 pound new potatoes (about 1 inch in diameter)', '2 teaspoons kosher salt', '¾ teaspoon finely ground black pepper', '1 teaspoon finely chopped rosemary', '1 teaspoon finely chopped thyme', '1 teaspoon finely chopped parsley']

Thanksgiving Mac and Cheese

Ingredients: ['1 cup evaporated milk', '1 cup whole milk', '1 tsp. garlic powder', '1 tsp. onion powder', '1 tsp. smoked paprika', '½ tsp. freshly ground black pepper', '1 tsp. kosher salt, plus more', '2 lb. extra-sharp cheddar, coarsely grated', '4 oz. full-fat cream cheese', '1 lb. elbow macaroni']

Fig.4.5

Chapter 5

CONCLUSION & FUTURE SCOPE

Conclusion

This project successfully showcases the application of Natural Language Processing and Machine Learning to develop an intelligent recipe recommendation system. By utilizing ingredient-based similarity through TF-IDF vectorization and cosine similarity, the system efficiently suggests relevant recipes based on the ingredients users already have. This not only promotes healthier eating habits but also aids in reducing food wastage by encouraging the use of available resources. Comprehensive data preprocessing, feature engineering, and exploratory data analysis have contributed to building a strong and scalable solution. Looking ahead, enhancements such as incorporating user feedback, enabling image-based search, or integrating deep learning models can further improve the system's accuracy and personalization.

Future Scope

- User Personalization: Incorporate user preferences, past searches, and ratings to provide more tailored recipe recommendations.
- Image-Based Search: Allow users to upload pictures of ingredients or dishes to get relevant recipe suggestions using computer vision.
- Voice Assistant Integration: Enable voice-based input for a hands-free, interactive experience in smart kitchens.
- Advanced Models: Use deep learning techniques (like BERT or LSTMs) to better understand ingredient context and improve recommendation accuracy.

REFERENCES

i. Recipe Recommendation System Using TF-IDF

ii. RecipeIS—Recipe Recommendation System Based on Recognition

iii. Real-Time Recipe Recommendation Based on Ingredients They Have at Home Using TF-IDF Algorithm

iv. A Hybrid Group-Based Food Recommender Framework for Personalized Recipe Suggestions

v. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise (DBSCAN)

vi. Machine Learning Based Food Recipe Recommendation System