CULINARY CONNECT
A Smart Recipe Recommendation and Nutrition Analysis Platform.
REPORT

Project Team Members

1. **Dennis Muuo**
2. **Fiona Amugune**
3. **Caroline Kimani**
4. **Daniel Karue**
5. **Adrianna Ndubi**

Business Understanding

The goal of this project is to develop a Recipe Recommendation System, a Sentiment Analysis model, that enhances user experience by providing personalized recipe suggestions. The system will analyze user preferences, sentiment analysis, and cultural relevance to recommend recipes that align with dietary needs, taste preferences, and engagement patterns. Additionally, this project aims to preserve and promote African and international culinary heritage while making it more accessible to a wider audience.

Problem Statement

African and International cuisine platforms struggle with low engagement due to generic recommendations and limited sentiment analysis. We propose a personalized recipe recommendation system with sentiment analysis to enhance user experience, retention, and platform authority.

Objectives

Main Objective:

The primary objective is to build a Recipe Recommendation System and sentiment Analysis model using the provided datasets. The goal is to enhance user experience by offering personalized recipe suggestions and analyzing user feedback to improve content.

Secondary Objectives:

- Derive meaningful Insights for the Stakeholders such as:
- Platform Users: Seek personalized and diverse recipe recommendations to explore new dishes and enhance their cooking skills.
- Developers & Data Science Team: Responsible for model development, data integration, and ensuring the system operates smoothly.
- Business Analysts: Utilize insights from user interactions and feedback to identify popular trends, user preferences, and potential areas for growth.

Approach Methodology

The methodology involves a systematic approach to preprocessing, feature extraction, model selection, and evaluation. By iteratively testing and refining different techniques, the goal is to identify the most effective combination of features and models for accurately classifying customer sentiment and recommendation systems. This will provide actionable insights into customer opinions and help improve interactions for all stakeholders at large.
To address the key questions outlined in this project, the following approach methodology was employed.

- Data cleaning to fill in the columns with missing values, remove the metric 'g', and dropping duplicates for both international and African cuisine datasets
- Feature engineering and scaling the data for both datasets to transform it into meaningful features that improve the performance of the models.
- Vectorization to convert the data to numerical format
- Collaborative filtering recommendation
- Content-based recommendation system
- Data preprocessing for sentimental analysis
- Text normalization: to convert text, remove punctuation, and handle contractions.
- Tokenization: to split text into individual words or tokens
- Stop word removal: to eliminate common words that do not contribute to sentiment.
- Model evaluation
- Deployment

Metrics of Success

To assess the effectiveness of the sentimental analysis model and recommendation system, the following metrics will be used.

Sentimental Analysis:

- TD-IDF for feature extraction
- Logistic regression for classification
- Accuracy score and classification report for evaluation

1. Accuracy: Measures the percentage of correctly classified sentiment instances out of the total.

Target: 85% or higher.

2. Precision: Indicates the percentage of correctly predicted positive sentiments, reflecting how often the model is correct when predicting specific sentiments.

Target: 80% or higher for each sentiment class (positive, negative, or neutral).

3. Recall: Evaluate the model's ability to correctly identify all relevant instances of a specific sentiment, balancing true positives and false negatives.

Target: 75% or higher for each class.

4. F1-Score: provides a balanced measure by combining precision and recall into a single metric.

Target: 80% or higher overall.

Recommendation system:

- Cosine similarity for recommendations of recipes based on the textual descriptions.
- Collaborative filtering to leverage past user interactions to make personalized recommendations.
- Linear regression for predictive modeling based on the calories and cooking time constraints.
- Mean squared error 0.1214
- Root mean square error 0.3484

DATA UNDERSTANDING

Data Sources:

The data used for the recommendation system and sentimental analysis was scraped from websites using Selenium and Beautiful Soup Libraries.

International Recipes: Scraped from Food.com

African Recipes: Scraped from Cookpad

International Reviews: Food.com

Nutrition: Scrapped from Better Me.

The data was read using the pandas library in CSV format, key steps in data inspection included checking the dataset's shape(number of rows and columns), extracting the metadata with .info(), describing the data using the .describe(), identifying duplicates, dropping duplicates with .drop duplicates() and detecting missing values using .isna().

Data Structure

Recommendation System:

International Recipes Data

- name; Name of the recipe
- id; Unique identifier for the recipe
- minutes; Time required to prepare the recipe (in mins)
- tag; Tags associated with the recipe
- steps; Step-by-step instructions for preparation
- ingredients; List of ingredients used
- calories; Caloric content per serving
- total fat (PDV); Total fat content as a percentage of daily value
- sugar (PDV); Sugar content as a percentage of daily value
- sodium (PDV); Sodium content as a percentage of daily value
- protein (PDV); Protein content as a percentage of daily value
- saturated fat (PDV); Saturated fat content as a percentage of daily value
- carbohydrates (PDV); Carbohydrate content as a percentage of daily value

Nutrition Data

- name; Name of the food item
- serving_size; Size of the serving (in grams or milliliters)
- calories; Caloric content per serving
- total_fat; Total fat content (in grams)
- saturated_fat; Saturated fat content (in grams)
- cholesterol; Cholesterol content (in mg)
- sodium; Sodium content (in mg)
- potassium; Potassium content (in mg)
- saturated_fatty_acids; Total saturated fatty acids (in grams).

Sentimental Analysis:

Interactions Data

- user_id; Unique identifier for the user
- recipe_id; Unique identifier for the recipe
- date; Date of interaction
- rating; User's rating for the recipe
- review; User's review or feedback

DATA PREPARATION.

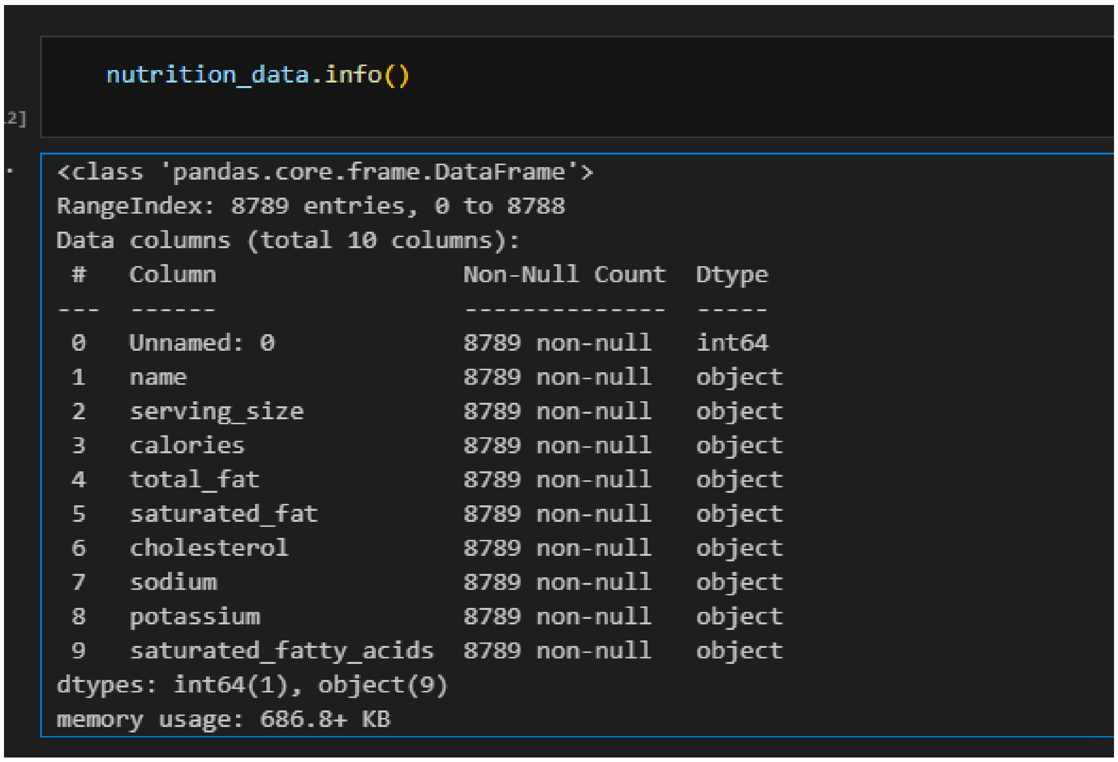
EDA AND PREPROCESSING

Data Cleaning:

- **Nutrition Data:**

We performed a series of steps to clean and preprocess the nutrition dataset to ensure accurate analysis. We started by inspecting the data;

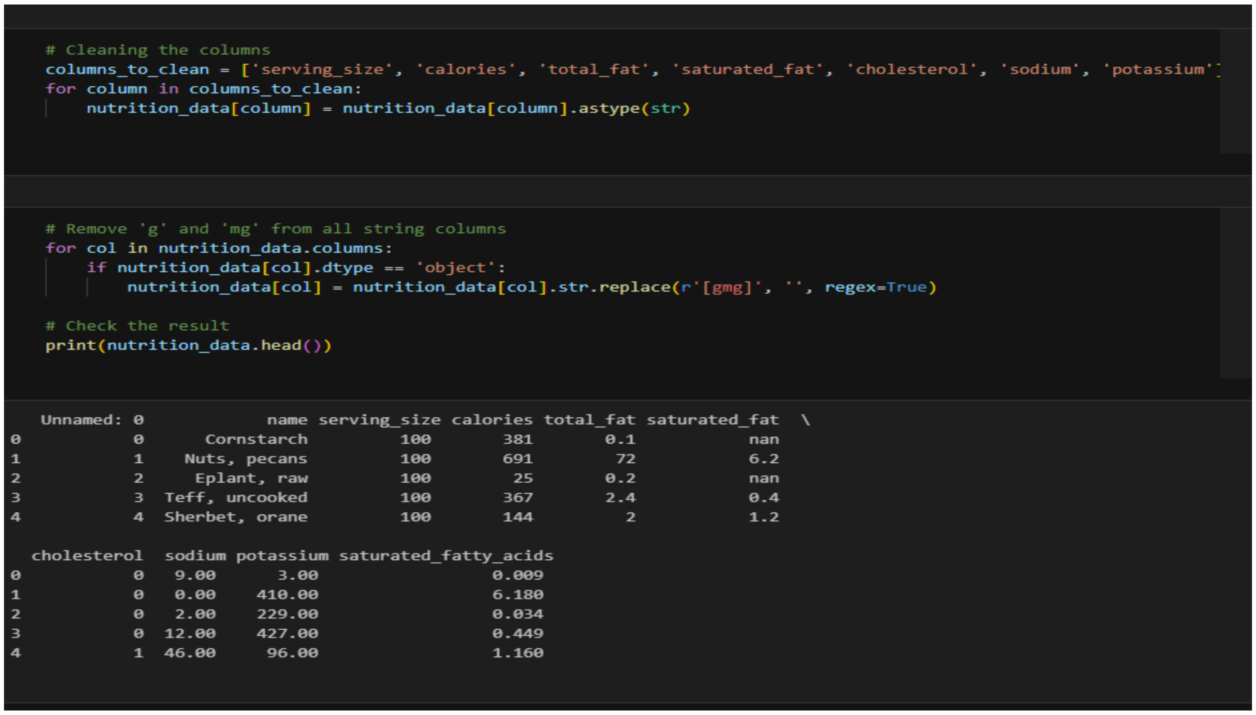
nutrition_data.info()



```
nutrition_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8789 entries, 0 to 8788
Data columns (total 10 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Unnamed: 0             8789 non-null   int64
 1   name                   8789 non-null   object
 2   serving_size           8789 non-null   object
 3   calories               8789 non-null   object
 4   total_fat              8789 non-null   object
 5   saturated_fat          8789 non-null   object
 6   cholesterol            8789 non-null   object
 7   sodium                 8789 non-null   object
 8   potassium              8789 non-null   object
 9   saturated_fatty_acids  8789 non-null   object
dtypes: int64(1), object(9)
memory usage: 686.8+ KB
```

nutrition_data.head()

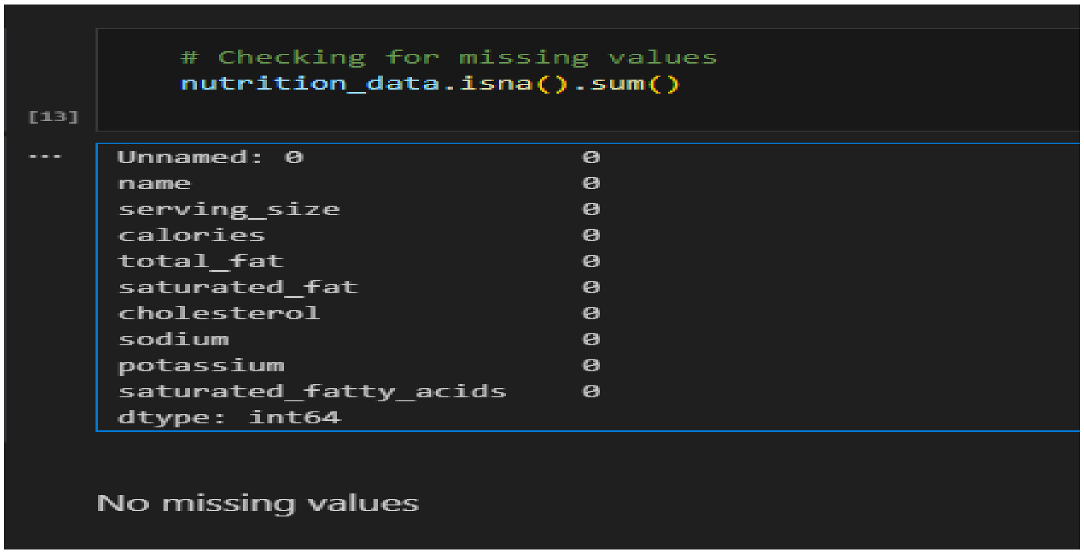


```
# Cleaning the columns
columns_to_clean = ['serving_size', 'calories', 'total_fat', 'saturated_fat', 'cholesterol', 'sodium', 'potassium']
for column in columns_to_clean:
    nutrition_data[column] = nutrition_data[column].astype(str)
```

```
# Remove 'g' and 'mg' from all string columns
for col in nutrition_data.columns:
    if nutrition_data[col].dtype == 'object':
        nutrition_data[col] = nutrition_data[col].str.replace(r'[gmg]', '', regex=True)

# Check the result
print(nutrition_data.head())
```

```
   Unnamed: 0             name serving_size calories total_fat saturated_fat  \
0           0       Cornstarch          100      381       0.1           nan
1           1     Nuts, pecans          100      691        72           6.2
2           2      Eplant, raw          100       25       0.2           nan
3           3   Teff, uncooked          100      367       2.4           0.4
4           4   Sherbet, orane          100      144         2           1.2

  cholesterol  sodium potassium saturated_fatty_acids
0           0    9.00      3.00                 0.009
1           0    0.00    410.00                 6.180
2           0    2.00    229.00                 0.034
3           0   12.00    427.00                 0.449
4           1   46.00     96.00                 1.160
```

Checked for the missing values, column types, and the general structure of the dataset.



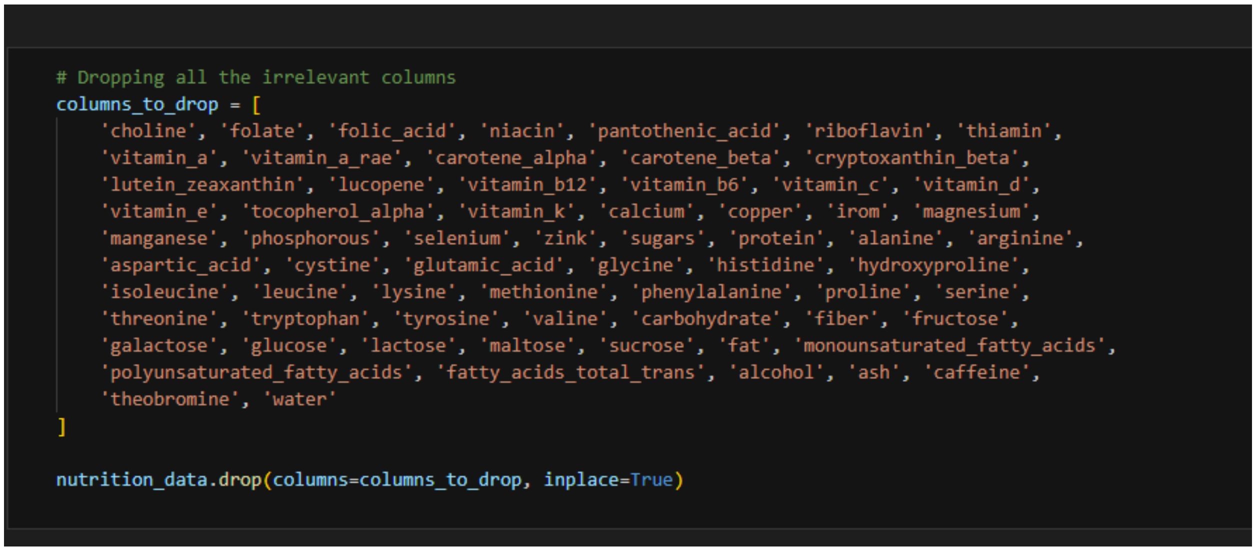
```
# Checking for missing values
nutrition_data.isna().sum()
```

[13]

```
Unnamed: 0               0
name                     0
serving_size             0
calories                 0
total_fat                0
saturated_fat            0
cholesterol              0
sodium                   0
potassium                0
saturated_fatty_acids    0
dtype: int64
```

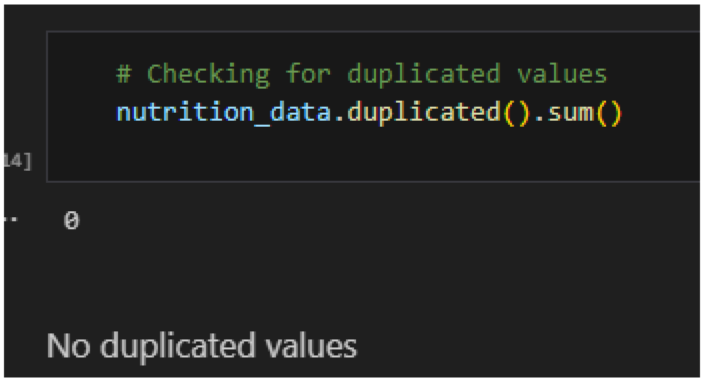
No missing values

Handled the irrelevant columns by dropping them to help us get more accurate insights.



```python
# Dropping all the irrelevant columns
columns_to_drop = [
    'choline', 'folate', 'folic_acid', 'niacin', 'pantothenic_acid', 'riboflavin', 'thiamin',
    'vitamin_a', 'vitamin_a_rae', 'carotene_alpha', 'carotene_beta', 'cryptoxanthin_beta',
    'lutein_zeaxanthin', 'lucopene', 'vitamin_b12', 'vitamin_b6', 'vitamin_c', 'vitamin_d',
    'vitamin_e', 'tocopherol_alpha', 'vitamin_k', 'calcium', 'copper', 'irom', 'magnesium',
    'manganese', 'phosphorous', 'selenium', 'zink', 'sugars', 'protein', 'alanine', 'arginine',
    'aspartic_acid', 'cystine', 'glutamic_acid', 'glycine', 'histidine', 'hydroxyproline',
    'isoleucine', 'leucine', 'lysine', 'methionine', 'phenylalanine', 'proline', 'serine',
    'threonine', 'tryptophan', 'tyrosine', 'valine', 'carbohydrate', 'fiber', 'fructose',
    'galactose', 'glucose', 'lactose', 'maltose', 'sucrose', 'fat', 'monounsaturated_fatty_acids',
    'polyunsaturated_fatty_acids', 'fatty_acids_total_trans', 'alcohol', 'ash', 'caffeine',
    'theobromine', 'water'
]

nutrition_data.drop(columns=columns_to_drop, inplace=True)
```

Checking for duplicates



```python
# Checking for duplicated values
nutrition_data.duplicated().sum()
```

```
0
```

No duplicated values

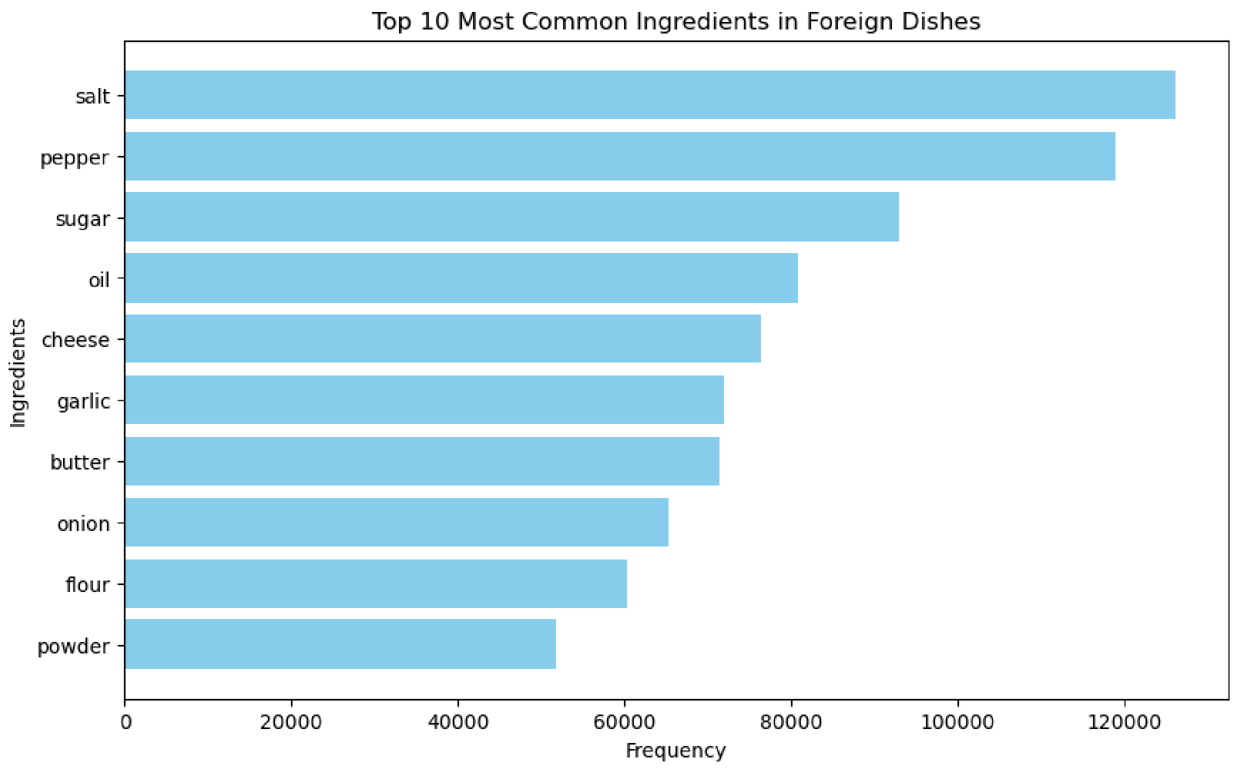
- **African Recipes, International Recipes and Interaction Data:**

The same series of steps to clean and preprocess was repeated for these two datasets to ensure accurate analysis.

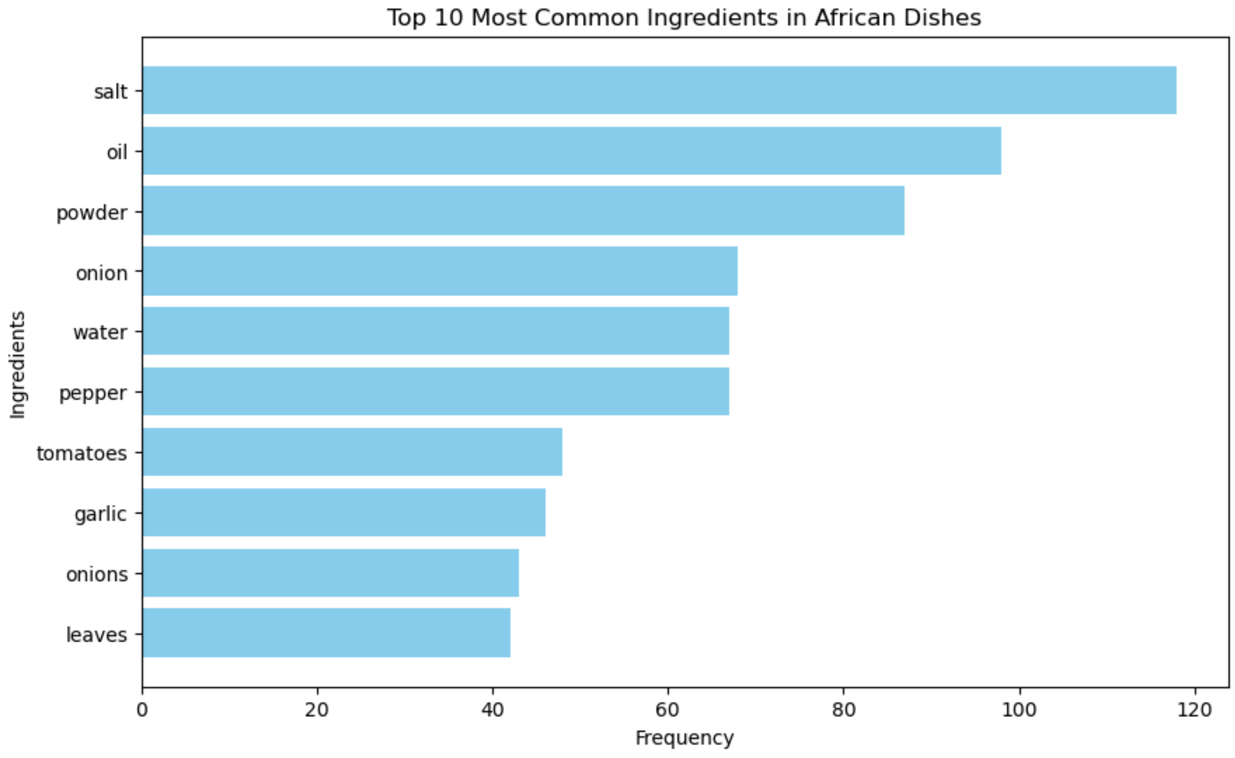
Exploratory Data Analysis

In this section, we wanted to find the answers to our business questions such as, what are the most common ingredients used across all cuisines? how does the nutritional content vary across different cuisines? what trends can be observed in user interaction over time? are there seasonal patterns in user engagement with recipes? which cuisines receive the most positive, and negative feedback?
We started by importing all the relevant libraries such as pandas, matplotlib for visualizations, counter, and re to read the data and analyze the data.

To find out the most common ingredient used across all cuisines, in this segment, we had to define the common adjectives and stop words, flatten and count ingredients, combine the adjectives, and convert the data frame for plotting. Through the visualization, we can deduce that the most common ingredients in foreign dishes are salt, pepper, sugar, and oil. They have the highest number of frequency.

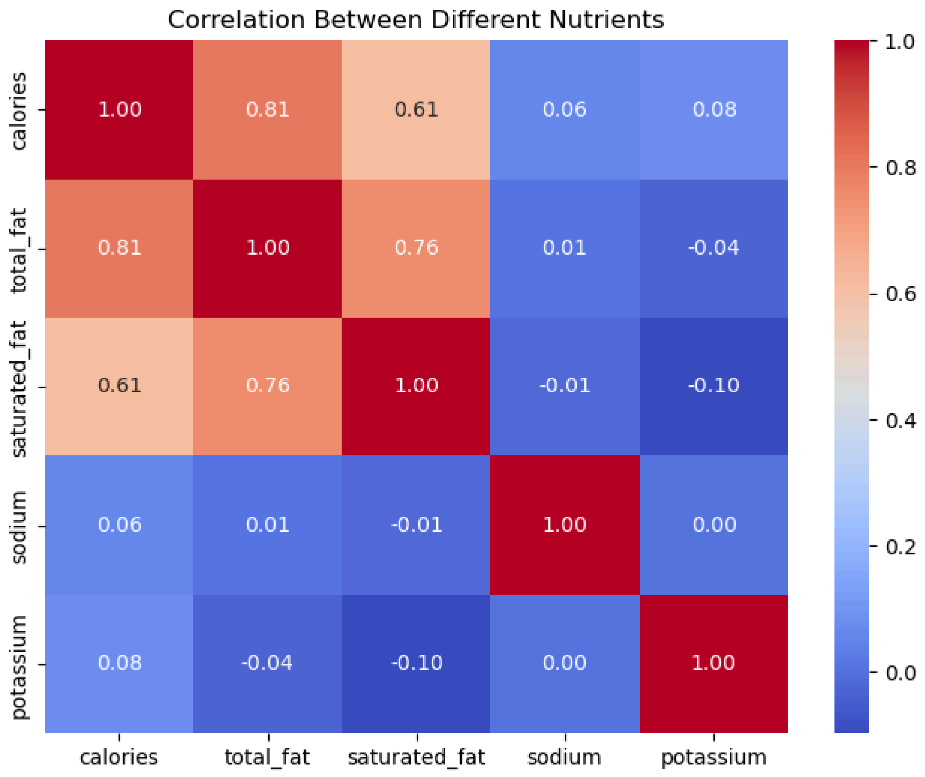


To find the most common ingredients in African dishes, the same steps were implemented and the data frame was converted for plotting. The findings are shown below.



The ingredients with the highest frequency were salt, oil, powder, and onion.

How does the nutritional content vary across different cuisines, a correlation matrix analysis was used to explore the relationships between different nutritional components.



From the heatmap, it's evident that calories, total fat, and saturated fat are closely related. Foods that are considered higher in fat tend to be higher in calories. Sodium and Potassium show little to no correlation with fat and calorie content. Sodium and Saturated fat are mostly independent meaning a food high in one does not necessarily have high amounts of the other.

Hypothesis:

Positive Relationship Hypothesis
Hypothesis: Higher levels of total fat are associated with increased calorie content and saturated fat in foods.
Rationale: The strong correlations (0.81 and 0.76) suggest that foods high in total fat tend to have higher calories and saturated fat.

Minimal Impact of Sodium and Potassium
Hypothesis: Sodium and potassium levels have limited or negligible relationships with calories, total fat, and saturated fat.
Rationale: The weak correlations (close to 0) indicate that variations in sodium and potassium do not significantly affect the levels of the other nutrients.

Caloric Density Hypothesis
Hypothesis: Calorically dense Foods are likely to contain higher amounts of total fat and saturated fat.
Rationale: The strong correlation between calories and both types of fat reinforces the idea that higher-calorie foods are often those that are higher in fat.

What are the top 10 healthiest recipes, The key metrics for a healthy recipe were as follows.
Calories: For a main meal: 400–600 kcal **For a snack:**100–250 kcal
Check: High calories with high fat and sugar might indicate unhealthy food.

Protein (PDV): Higher protein (≥15–20%)is good for satiety and muscle health.
Check: protein (PDV) ≥ 15` is often considered high.

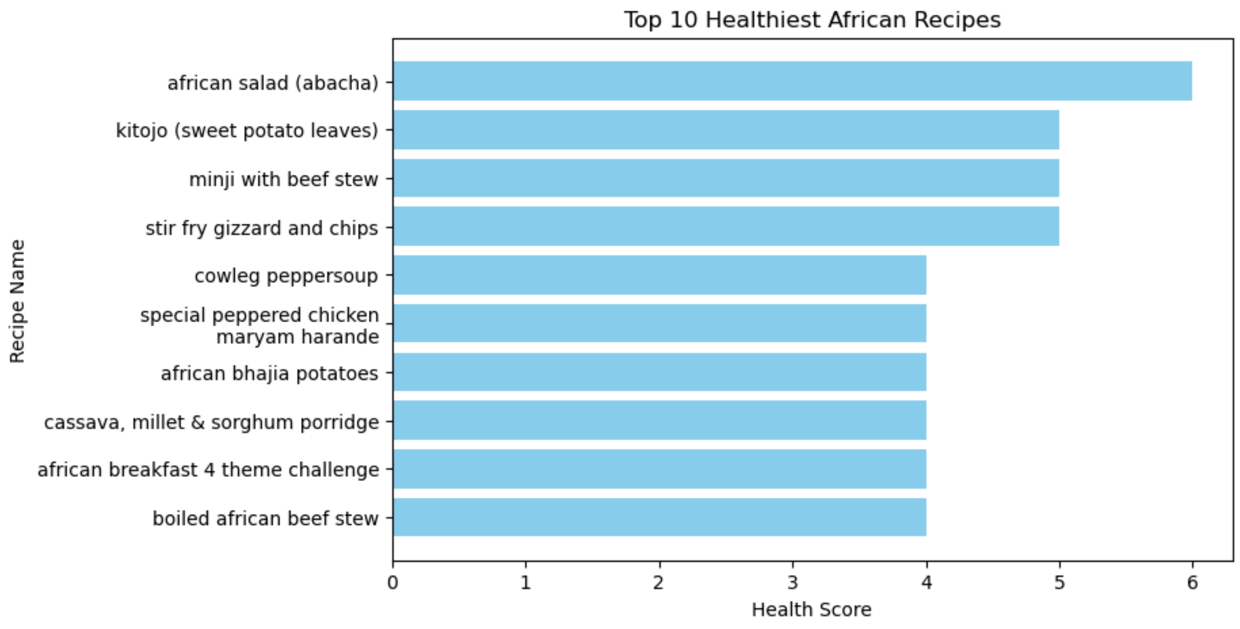
Total Fat (PDV): Aim for moderate fat, mostly unsaturated fats.
Check:`total fat (PDV) ≤ 20` for most meals; avoid high saturated fat.

Saturated Fat (PDV): Should be low (<10% of daily value)
Check:`saturated fat (PDV) ≤ 10`

Sodium (PDV): High sodium (>20%)might indicate processed or unhealthy food.
Check:`sodium (PDV) ≤ 15`

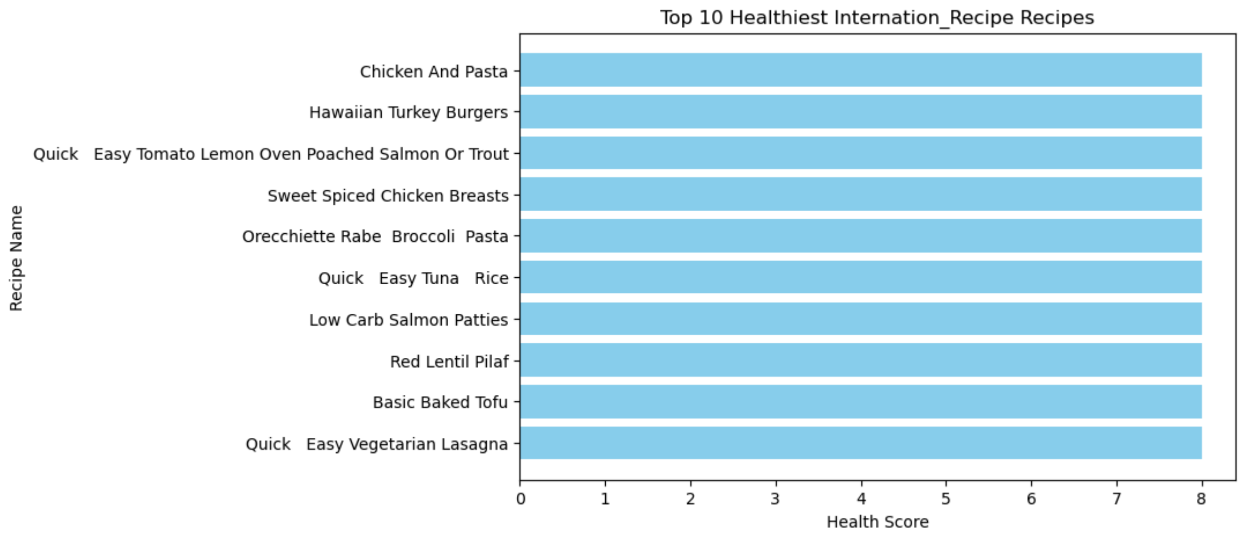
Carbohydrates (PDV): Look for complex carbs from whole ingredients.
Check: carbohydrates (PDV) ≤ 50` unless it's a carb-focused meal.

By calculating the health score and applying the health score function the data frame plotting was as follows.



From the graph, it's evident that African salad, Kitojo, and Minji with stew were at the top, having met the key metrics used to determine whether a meal is healthy.

The same metrics were applied to the international recipes however the dataset for international recipes had to undergo a standardized text formatting to enable uniformity. The visualizations were as follows.

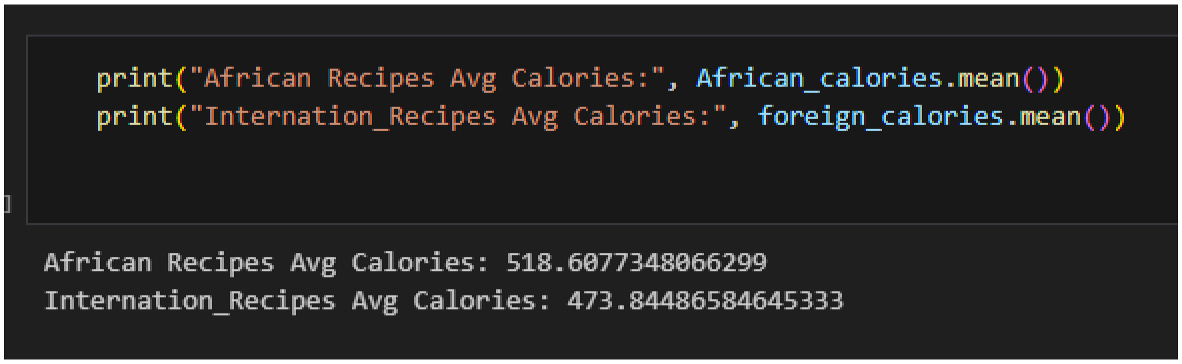


Hypothesis testing; from the visualizations, the null hypothesis of this dataset shows that African recipes generally have a lower calorie content compared to international recipes.

Scipy.stats library helps us to break this down by performing independent t-tests for the unequal variances.

The results are as follows: ***T-test Statistic: 2.120 and the P-value: 0.035*** indicating that the difference in calorie content between African and International recipes is statistically significant. With a ***p-value of 0.035,*** it means that we should reject our null hypothesis.

By comparing the means of the average calories in African recipes and International recipes, the mean of African calorie content is higher than the International meaning we still have to reject the null hypothesis.

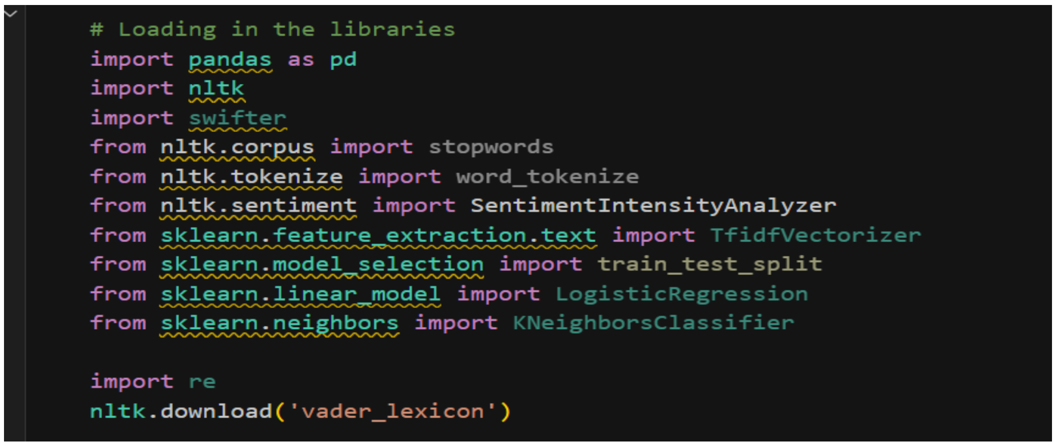


```
print("African Recipes Avg Calories:", African_calories.mean())
print("Internation_Recipes Avg Calories:", foreign_calories.mean())
```

```
African Recipes Avg Calories: 518.6077348066299
Internation_Recipes Avg Calories: 473.84486584645333
```

SENTIMENTAL ANALYSIS

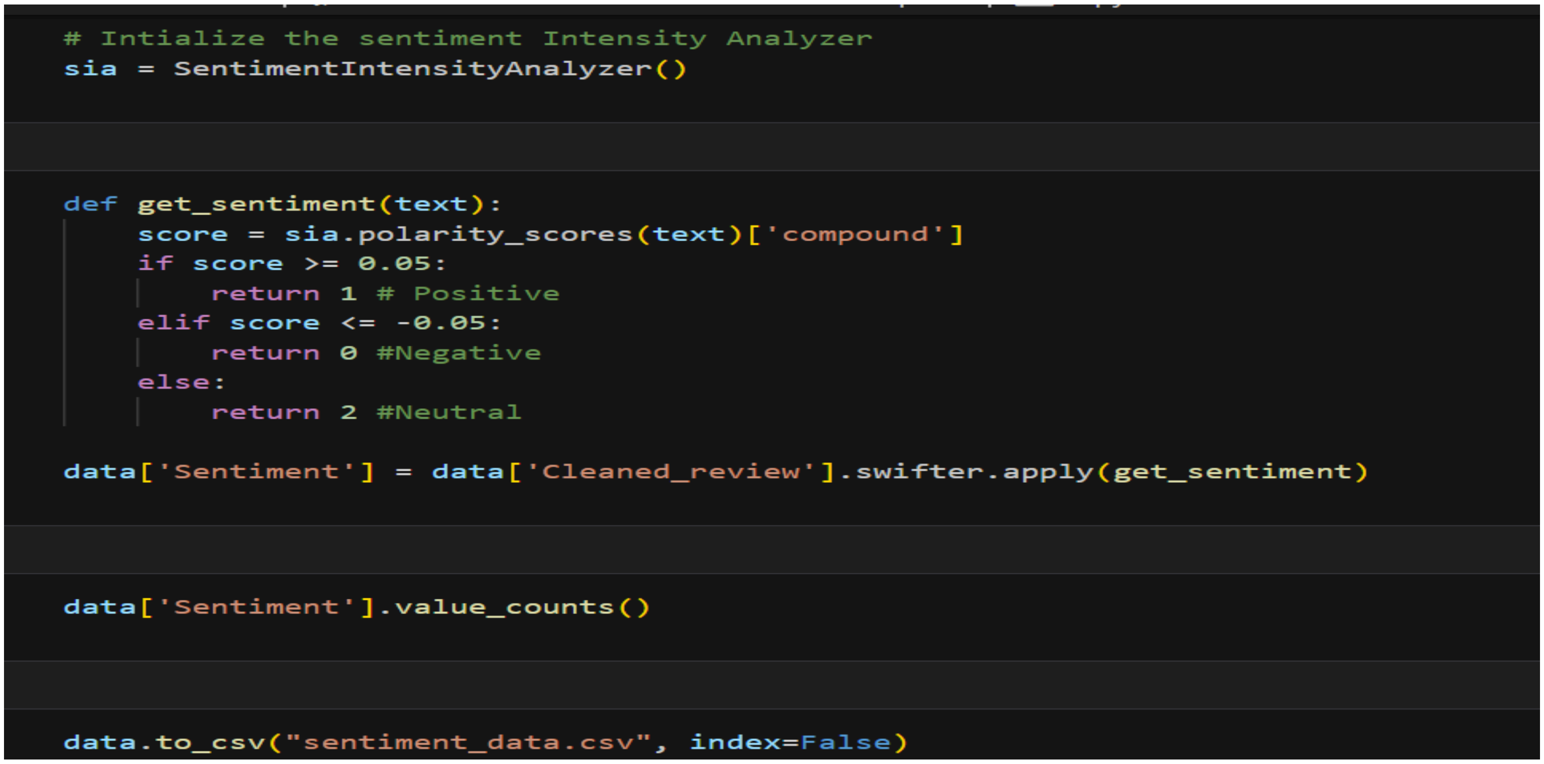
We started by loading the necessary libraries.



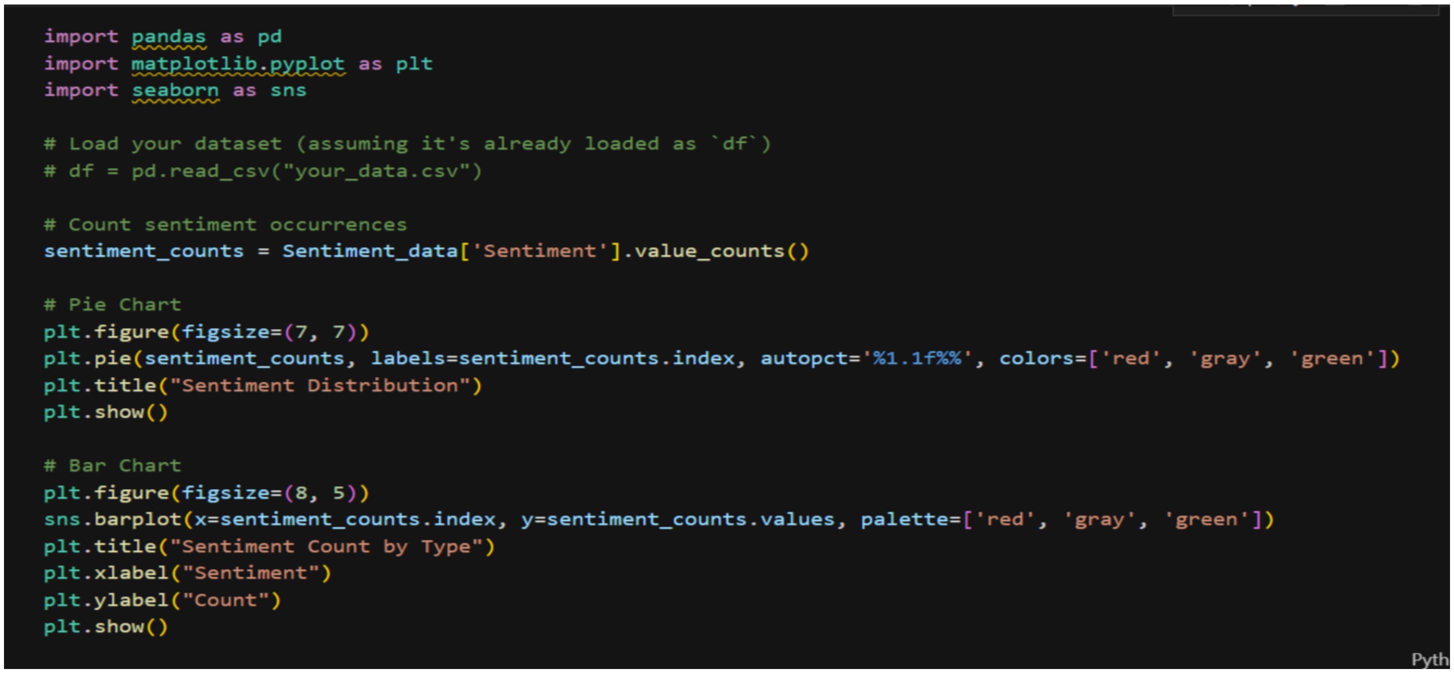
```
# Loading in the libraries
import pandas as pd
import nltk
import swifter
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.sentiment import SentimentIntensityAnalyzer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier

import re
nltk.download('vader_lexicon')
```

Then loaded the data (cleaned_combined_data .csv), checked for missing values, and dropped the missing values.

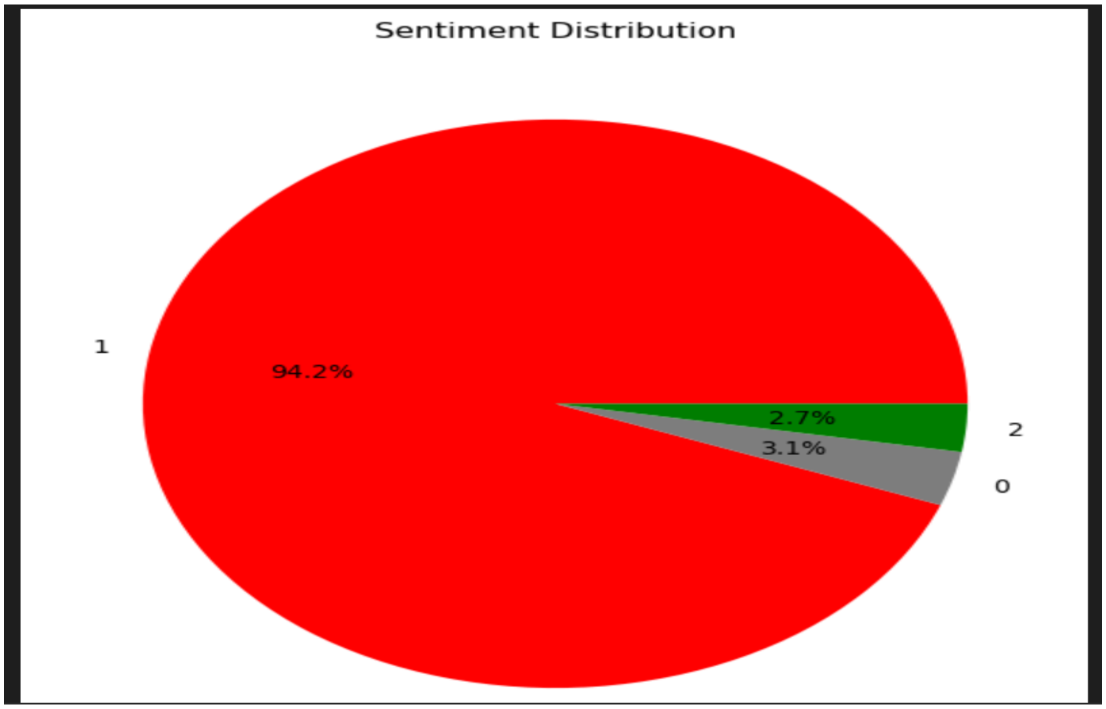
We then Initialize the sentiment Intensity Analyzer.

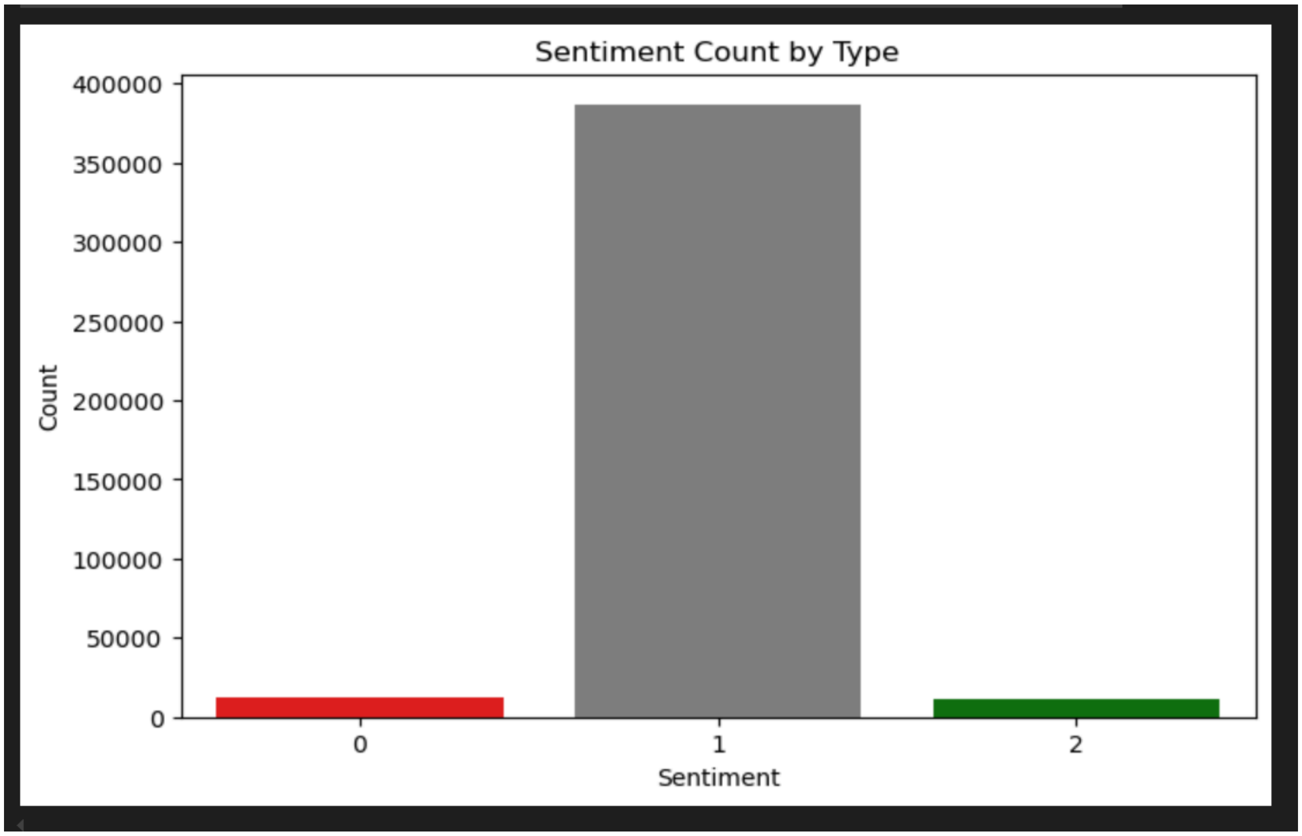


```
# Intialize the sentiment Intensity Analyzer
sia = SentimentIntensityAnalyzer()


def get_sentiment(text):
    score = sia.polarity_scores(text)['compound']
    if score >= 0.05:
        return 1 # Positive
    elif score <= -0.05:
        return 0 #Negative
    else:
        return 2 #Neutral

data['Sentiment'] = data['Cleaned_review'].swifter.apply(get_sentiment)


data['Sentiment'].value_counts()


data.to_csv("sentiment_data.csv", index=False)
```



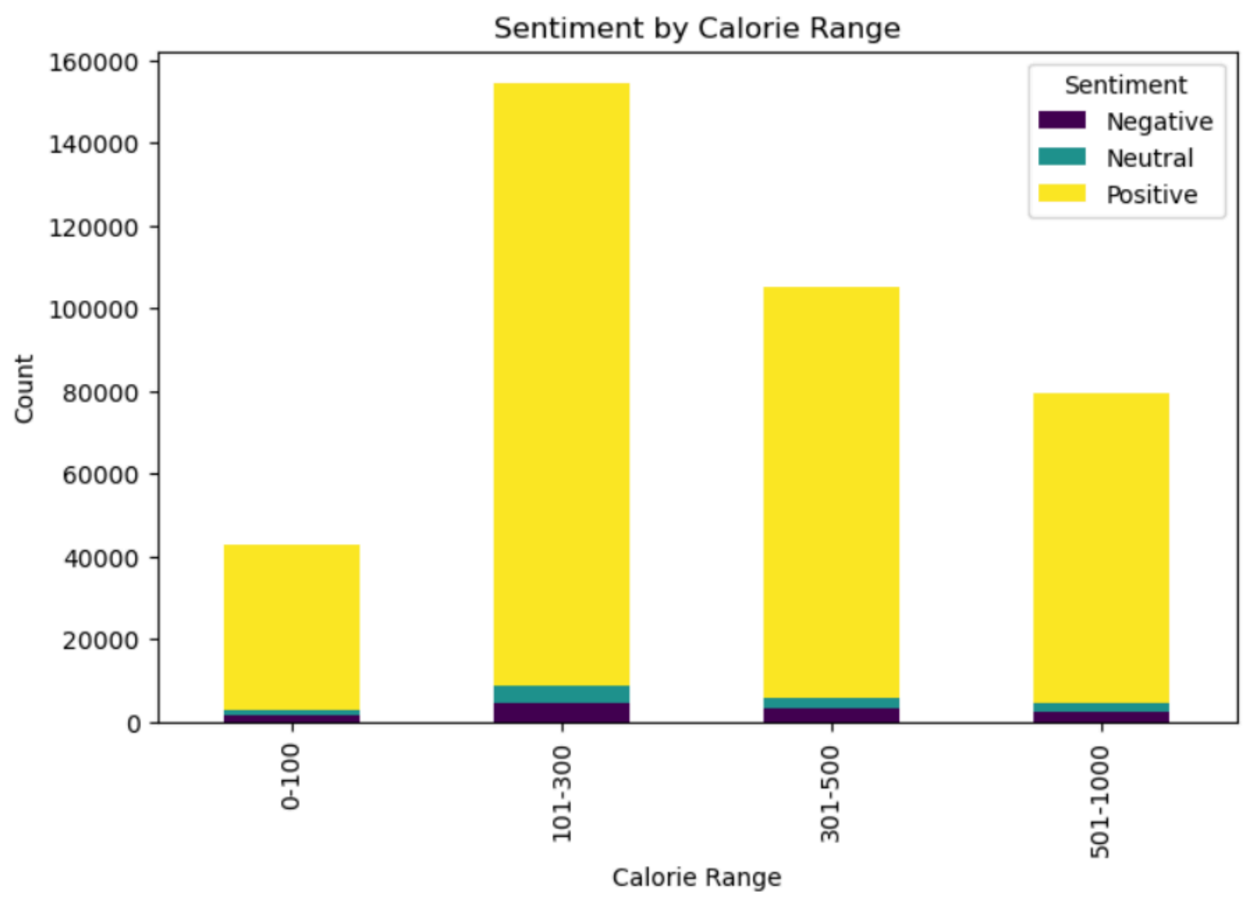
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load your dataset (assuming it's already loaded as `df`)
# df = pd.read_csv("your_data.csv")

# Count sentiment occurrences
sentiment_counts = Sentiment_data['Sentiment'].value_counts()

# Pie Chart
plt.figure(figsize=(7, 7))
plt.pie(sentiment_counts, labels=sentiment_counts.index, autopct='%1.1f%%', colors=['red', 'gray', 'green'])
plt.title("Sentiment Distribution")
plt.show()

# Bar Chart
plt.figure(figsize=(8, 5))
sns.barplot(x=sentiment_counts.index, y=sentiment_counts.values, palette=['red', 'gray', 'green'])
plt.title("Sentiment Count by Type")
plt.xlabel("Sentiment")
plt.ylabel("Count")
plt.show()
```


Sentiment Distribution
1
94.2%
2.7%
2
3.1%
0


Sentiment Count by Type
400000
350000
300000
250000
200000
150000
100000
50000
0
Count
0
1
2
Sentiment

Sentiments vs attributes

This barplot presents the distribution of sentiments (Negative,Neutral, and Positive) across different calorie ranges.



Key Observations**:**

1. **Higher-calorie recipes (101-1000 calories)** receive the most positive feedback, suggesting consumer preference for richer meals.

2. **Low-calorie recipes (0-100 calories)** also have strong positive sentiments but fewer total reviews.

3. **Negative and Neutral reviews** are consistently lower across all calorie ranges, indicating overall customer satisfaction.

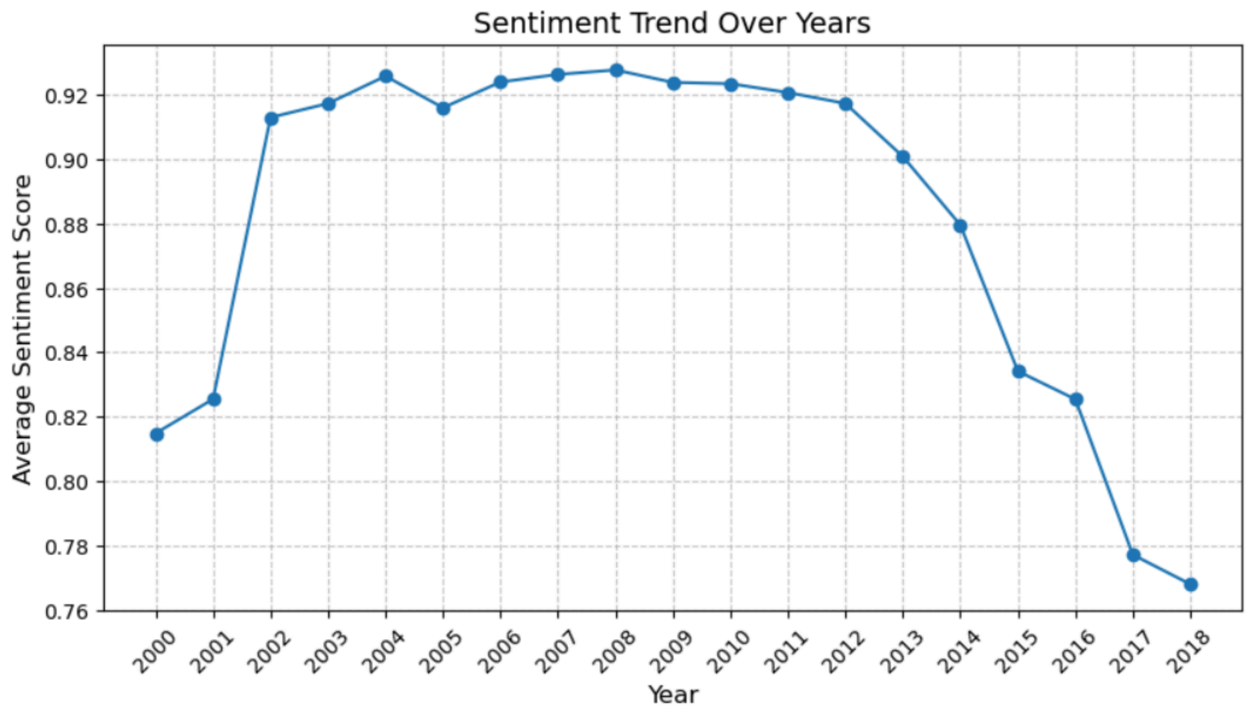
Possible Actions:

- Promote highly rated calorie-dense recipes for marketing.

- Analyze negative reviews in each range to identify improvement areas.

- Explore health-conscious options to attract calorie-conscious consumers.

Sentiment Trend over Time



Sentiment Trend Analysis (2000-2018)

Key Insights:

- **2000-2008:** Sentiment remained high and stable (~0.92), indicating strong customer satisfaction.

- **2009-2012:** Gradual decline (~0.91), suggesting shifting customer expectations or mild dissatisfaction.

- **2013-2018:** Significant drop from 0.90 to 0.76, highlighting a decline in satisfaction.

Recommendations:

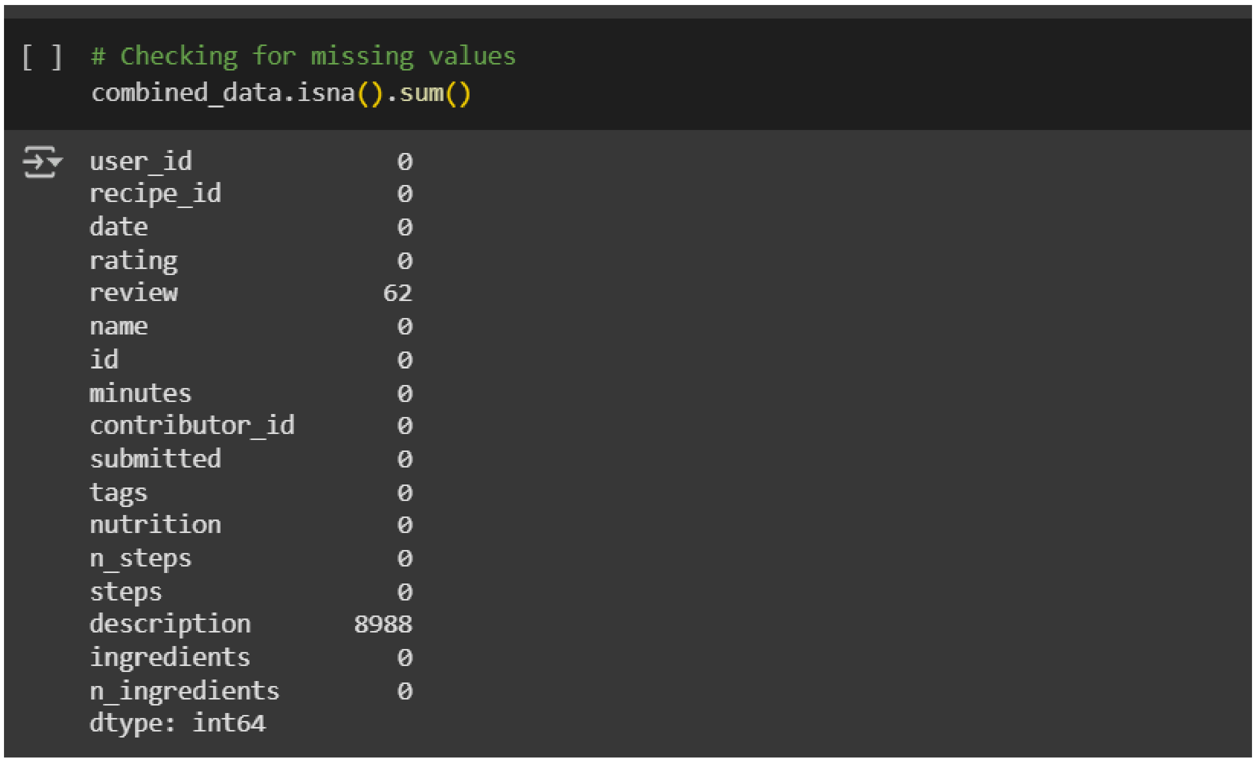
- Investigate 2013-2018 for major changes affecting sentiment.

- Address customer complaints and improve service/product quality

- Conduct a competitive analysis to stay ahead in the market.

Sentiment Analysis Cleaning

To ensure high-quality input data, missing values were addressed by removing empty review entries and dropping rows with missing values. During this process, **62 missing reviews** and **8,988 missing descriptions** were identified and dropped.

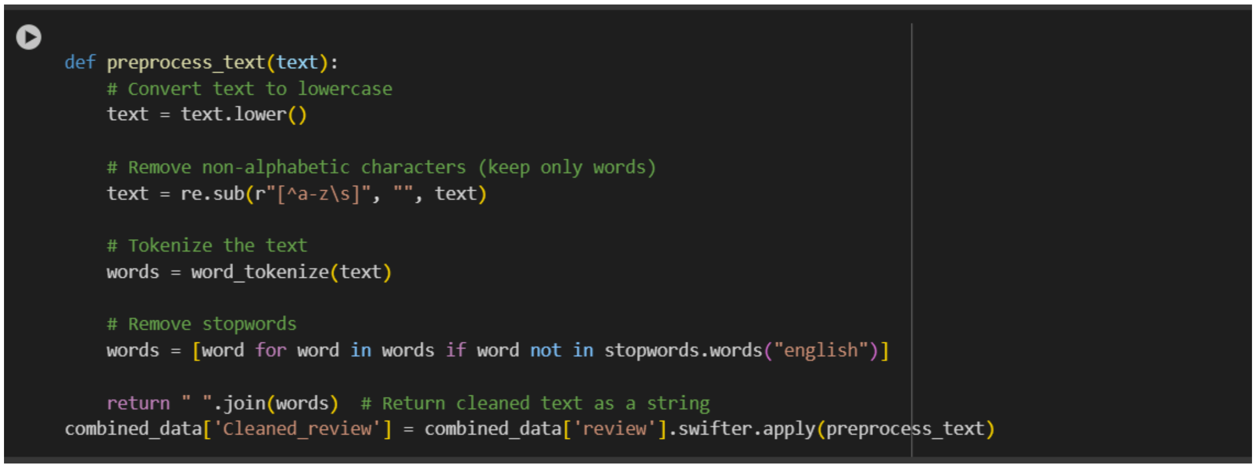


```
[ ] # Checking for missing values
    combined_data.isna().sum()

    user_id            0
    recipe_id          0
    date               0
    rating             0
    review            62
    name               0
    id                 0
    minutes            0
    contributor_id     0
    submitted          0
    tags               0
    nutrition          0
    n_steps            0
    steps              0
    description     8988
    ingredients        0
    n_ingredients      0
    dtype: int64
```

For text preprocessing, several essential steps were undertaken to standardize the data and remove unnecessary elements:

- Converted all text to lowercase for consistency.
- Removed punctuation, numbers, and extra spaces to clean the text.
- Tokenized text into individual words to facilitate further processing.

- Removed common stopwords to focus on meaningful words.



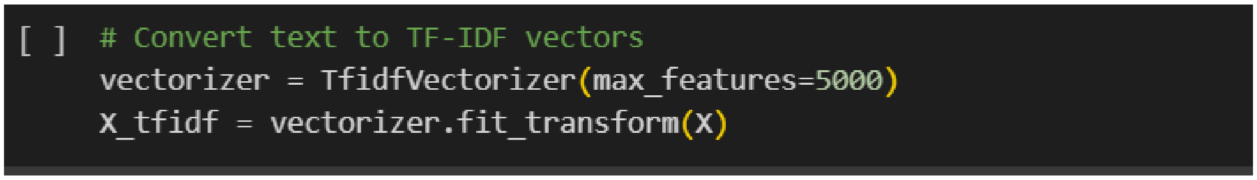
```python
def preprocess_text(text):
    # Convert text to lowercase
    text = text.lower()

    # Remove non-alphabetic characters (keep only words)
    text = re.sub(r"[^a-z\s]", "", text)

    # Tokenize the text
    words = word_tokenize(text)

    # Remove stopwords
    words = [word for word in words if word not in stopwords.words("english")]

    return " ".join(words)  # Return cleaned text as a string
combined_data['Cleaned_review'] = combined_data['review'].swifter.apply(preprocess_text)
```

Modeling

Feature Engineering & Vectorization

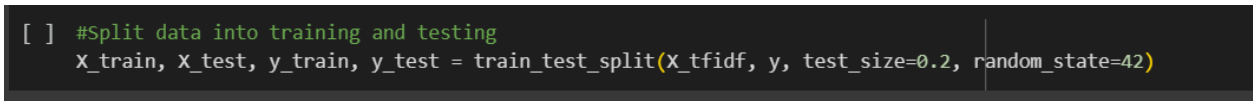
To transform textual data into a numerical format, **TF-IDF Vectorization** was applied. This technique captured the importance of words in the dataset while reducing noise.

The **top 5,000 most relevant words** were extracted for feature representation, ensuring that only meaningful terms contributed to the model.



```python
[ ]  # Convert text to TF-IDF vectors
     vectorizer = TfidfVectorizer(max_features=5000)
     X_tfidf = vectorizer.fit_transform(X)
```

The dataset was then split into **80% training and 20% testing**, ensuring a balanced approach to model evaluation.



```python
[ ]  #Split data into training and testing
     X_train, X_test, y_train, y_test = train_test_split(X_tfidf, y, test_size=0.2, random_state=42)
```

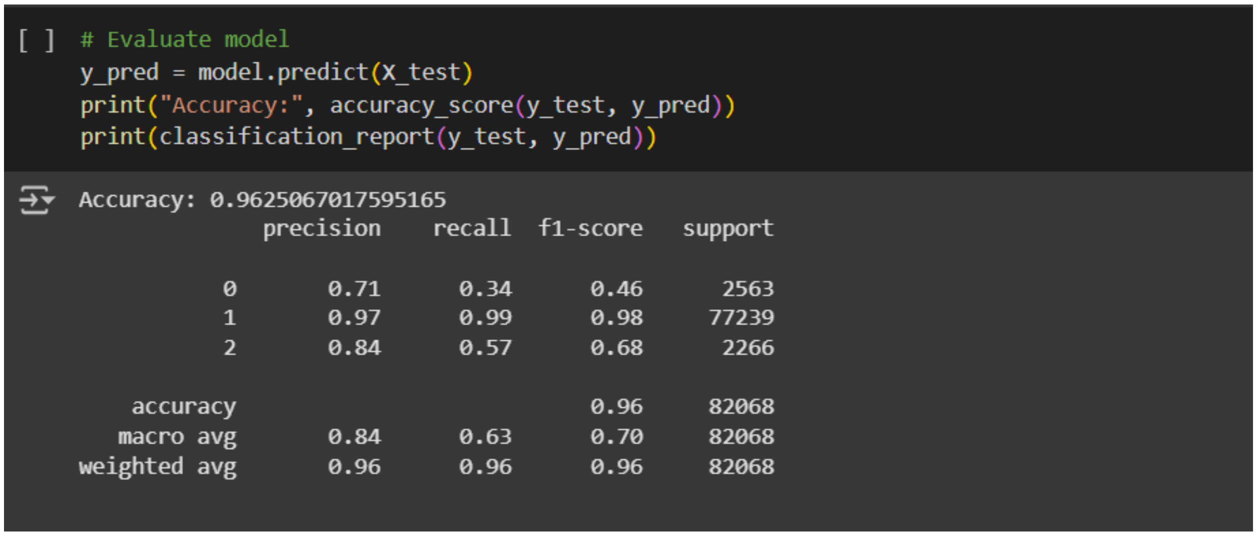
Model Selection

A **Logistic Regression** model was selected for sentiment classification due to its efficiency and strong performance in text-based classification tasks.

The model was evaluated using **accuracy, precision, recall, and F1-score** across Positive, Neutral, and Negative sentiment categories.

Overall Model Performance

The model achieved an **accuracy of 96.25%**, meaning that **96.25%** of test predictions were correct. This high accuracy indicates that the model effectively classifies user reviews into **Positive, Neutral, or Negative** sentiments.



```
[ ] # Evaluate model
    y_pred = model.predict(X_test)
    print("Accuracy:", accuracy_score(y_test, y_pred))
    print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.9625067017595165
              precision    recall  f1-score   support

           0       0.71      0.34      0.46      2563
           1       0.97      0.99      0.98     77239
           2       0.84      0.57      0.68      2266

    accuracy                           0.96     82068
   macro avg       0.84      0.63      0.70     82068
weighted avg       0.96      0.96      0.96     82068
```

RECOMMENDATION SYSTEM

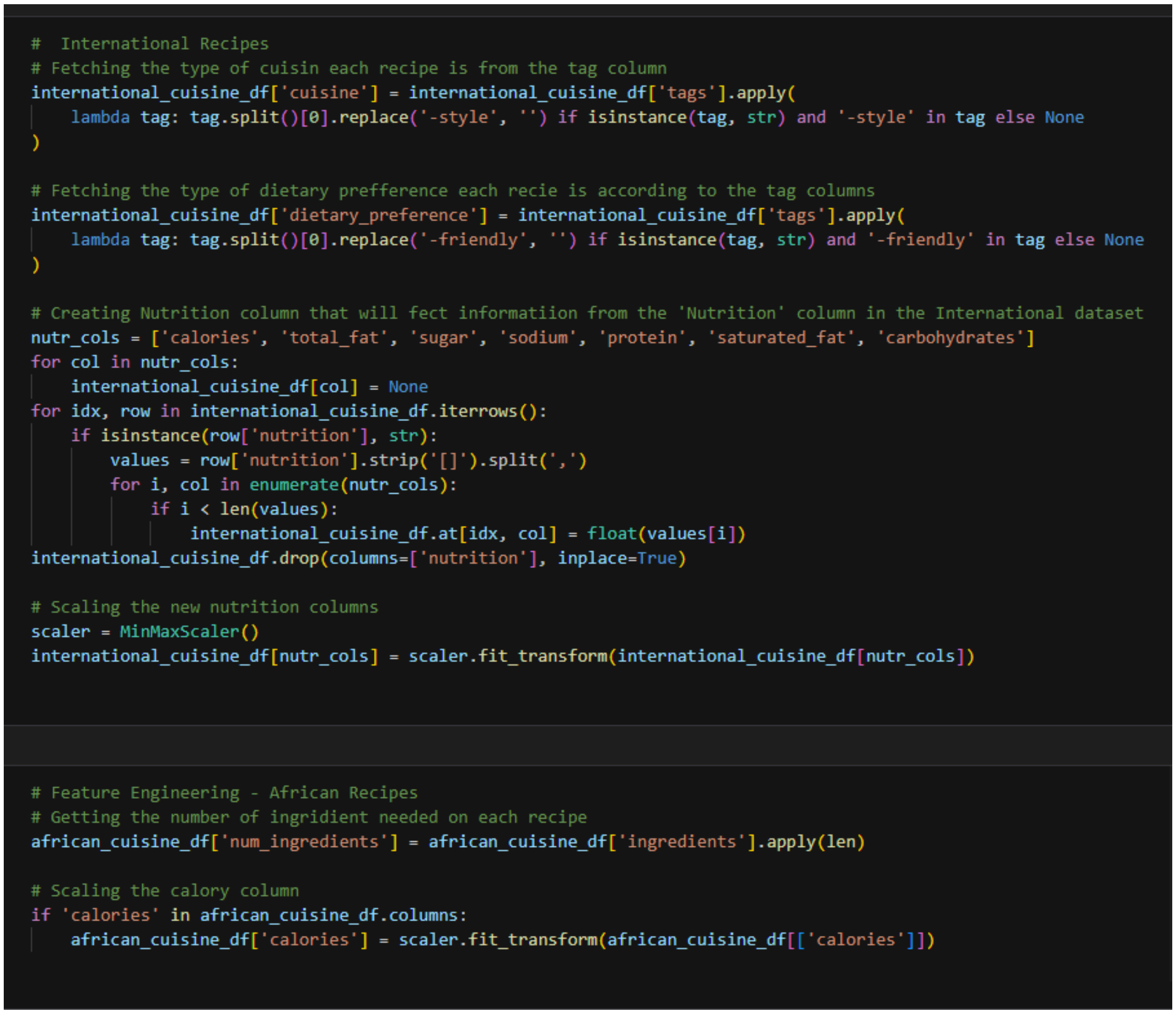
The data was read using the pandas' library in CSV format, necessary libraries for this analysis such as numpy, sklearn, and pickle were also imported.

Key steps in data inspection included handling missing values using the fillna(), dropping missing values using dropna(), data was converted using the astype(), and removing duplicates with drop_duplicates().

Feature Engineering.

Feature engineering is important in improving the effectiveness of a recommendation system, therefore the following techniques were applied.

- Creation of new features by extracting relevant details from existing data using apply() and lambda functions.
- Data was transformed into categorical variables using one-hot encoding.
- Scaling of numerical features to normalize values.
- Missing values were handled using the mean/mode for numerical/ categorical columns.



```python
#  International Recipes
# Fetching the type of cuisin each recipe is from the tag column
international_cuisine_df['cuisine'] = international_cuisine_df['tags'].apply(
    lambda tag: tag.split()[0].replace('-style', '') if isinstance(tag, str) and '-style' in tag else None
)

# Fetching the type of dietary prefference each recie is according to the tag columns
international_cuisine_df['dietary_preference'] = international_cuisine_df['tags'].apply(
    lambda tag: tag.split()[0].replace('-friendly', '') if isinstance(tag, str) and '-friendly' in tag else None
)

# Creating Nutrition column that will fect informatiion from the 'Nutrition' column in the International dataset
nutr_cols = ['calories', 'total_fat', 'sugar', 'sodium', 'protein', 'saturated_fat', 'carbohydrates']
for col in nutr_cols:
    international_cuisine_df[col] = None
for idx, row in international_cuisine_df.iterrows():
    if isinstance(row['nutrition'], str):
        values = row['nutrition'].strip('[]').split(',')
        for i, col in enumerate(nutr_cols):
            if i < len(values):
                international_cuisine_df.at[idx, col] = float(values[i])
international_cuisine_df.drop(columns=['nutrition'], inplace=True)

# Scaling the new nutrition columns
scaler = MinMaxScaler()
international_cuisine_df[nutr_cols] = scaler.fit_transform(international_cuisine_df[nutr_cols])
```

```python
# Feature Engineering - African Recipes
# Getting the number of ingridient needed on each recipe
african_cuisine_df['num_ingredients'] = african_cuisine_df['ingredients'].apply(len)

# Scaling the calory column
if 'calories' in african_cuisine_df.columns:
    african_cuisine_df['calories'] = scaler.fit_transform(african_cuisine_df[['calories']])
```

Vectorization.

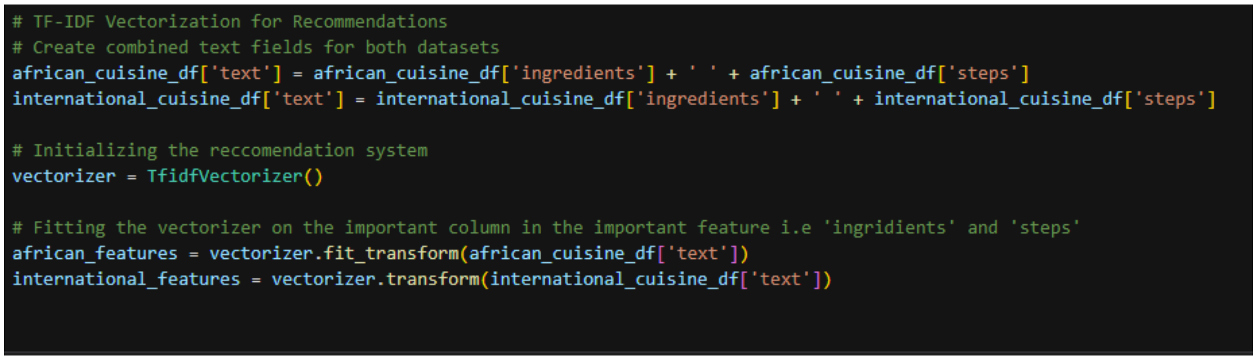
To convert textual data into numerical format, vectorization was used.

TF-IDF Vectorizatizatioon

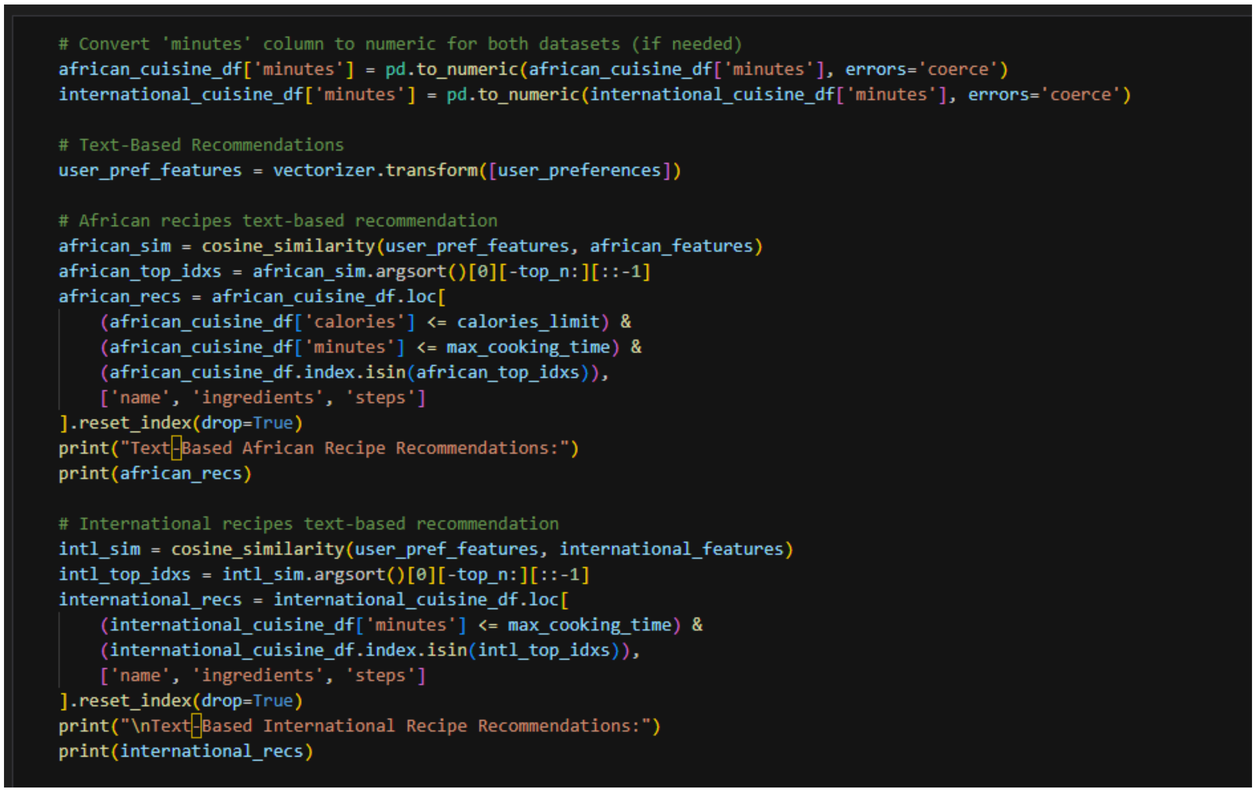
- Implemented TfidfVectorizer from sklearn.feature_extraction.text.
- Transformed the textual features (such as recipe descriptions and ingredient lists) into TF-IDF matrices.
- Removed stop words and applied n-grams to capture word dependencies.]

Cosine Similarity

- Used cosine similarity for creating bag-of-words representations of text features.
- Compared performance with TF-IDF and selected the most effective method.



```
# TF-IDF Vectorization for Recommendations
# Create combined text fields for both datasets
african_cuisine_df['text'] = african_cuisine_df['ingredients'] + ' ' + african_cuisine_df['steps']
international_cuisine_df['text'] = international_cuisine_df['ingredients'] + ' ' + international_cuisine_df['steps']

# Initializing the reccomendation system
vectorizer = TfidfVectorizer()

# Fitting the vectorizer on the important column in the important feature i.e 'ingridients' and 'steps'
african_features = vectorizer.fit_transform(african_cuisine_df['text'])
international_features = vectorizer.transform(international_cuisine_df['text'])
```



```
# Convert 'minutes' column to numeric for both datasets (if needed)
african_cuisine_df['minutes'] = pd.to_numeric(african_cuisine_df['minutes'], errors='coerce')
international_cuisine_df['minutes'] = pd.to_numeric(international_cuisine_df['minutes'], errors='coerce')

# Text-Based Recommendations
user_pref_features = vectorizer.transform([user_preferences])

# African recipes text-based recommendation
african_sim = cosine_similarity(user_pref_features, african_features)
african_top_idxs = african_sim.argsort()[0][-top_n:][::-1]
african_recs = african_cuisine_df.loc[
    (african_cuisine_df['calories'] <= calories_limit) &
    (african_cuisine_df['minutes'] <= max_cooking_time) &
    (african_cuisine_df.index.isin(african_top_idxs)),
    ['name', 'ingredients', 'steps']
].reset_index(drop=True)
print("Text-Based African Recipe Recommendations:")
print(african_recs)

# International recipes text-based recommendation
intl_sim = cosine_similarity(user_pref_features, international_features)
intl_top_idxs = intl_sim.argsort()[0][-top_n:][::-1]
international_recs = international_cuisine_df.loc[
    (international_cuisine_df['minutes'] <= max_cooking_time) &
    (international_cuisine_df.index.isin(intl_top_idxs)),
    ['name', 'ingredients', 'steps']
].reset_index(drop=True)
print("\nText-Based International Recipe Recommendations:")
print(international_recs)
```

Collaborative Filtering Recommendation.

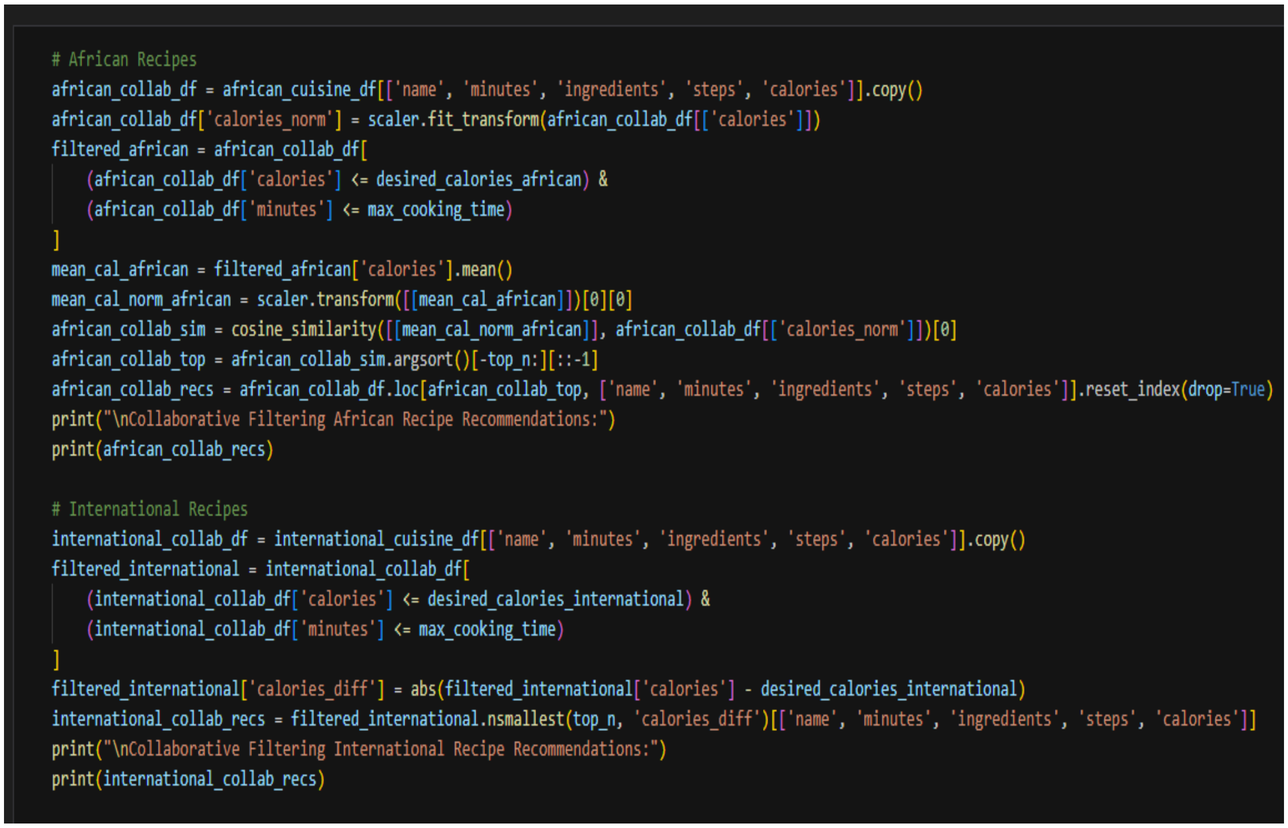
Collaborative filtering relies on user interaction data to provide personalized recommendations. Two approaches were implemented:

User-Item Matrix

- Constructed a user-item interaction matrix using **ratings and interactions**.
- Applied cosine similarity to measure user-user and item-item similarities.

Generating Recommendations

- Predicted missing ratings using the trained collaborative filtering model.
- Recommended recipes based on the **highest predicted ratings**.



```python
# African Recipes
african_collab_df = african_cuisine_df[['name', 'minutes', 'ingredients', 'steps', 'calories']].copy()
african_collab_df['calories_norm'] = scaler.fit_transform(african_collab_df[['calories']])
filtered_african = african_collab_df[
    (african_collab_df['calories'] <= desired_calories_african) &
    (african_collab_df['minutes'] <= max_cooking_time)
]
mean_cal_african = filtered_african['calories'].mean()
mean_cal_norm_african = scaler.transform([[mean_cal_african]])[0][0]
african_collab_sim = cosine_similarity([[mean_cal_norm_african]], african_collab_df[['calories_norm']])[0]
african_collab_top = african_collab_sim.argsort()[-top_n:][::-1]
african_collab_recs = african_collab_df.loc[african_collab_top, ['name', 'minutes', 'ingredients', 'steps', 'calories']].reset_index(drop=True)
print("\nCollaborative Filtering African Recipe Recommendations:")
print(african_collab_recs)

# International Recipes
international_collab_df = international_cuisine_df[['name', 'minutes', 'ingredients', 'steps', 'calories']].copy()
filtered_international = international_collab_df[
    (international_collab_df['calories'] <= desired_calories_international) &
    (international_collab_df['minutes'] <= max_cooking_time)
]
filtered_international['calories_diff'] = abs(filtered_international['calories'] - desired_calories_international)
international_collab_recs = filtered_international.nsmallest(top_n, 'calories_diff')[['name', 'minutes', 'ingredients', 'steps', 'calories']]
print("\nCollaborative Filtering International Recipe Recommendations:")
print(international_collab_recs)
```

Content-Based Recommendation System.

It relies on attributes of the recipe rather than the user behavior. We therefore were able to:

Building Similarity Matrices

- Used TfidfVectorizer to extract feature vectors from text-based attributes.
- Computed cosine similarity to measure content resemblance between recipes.

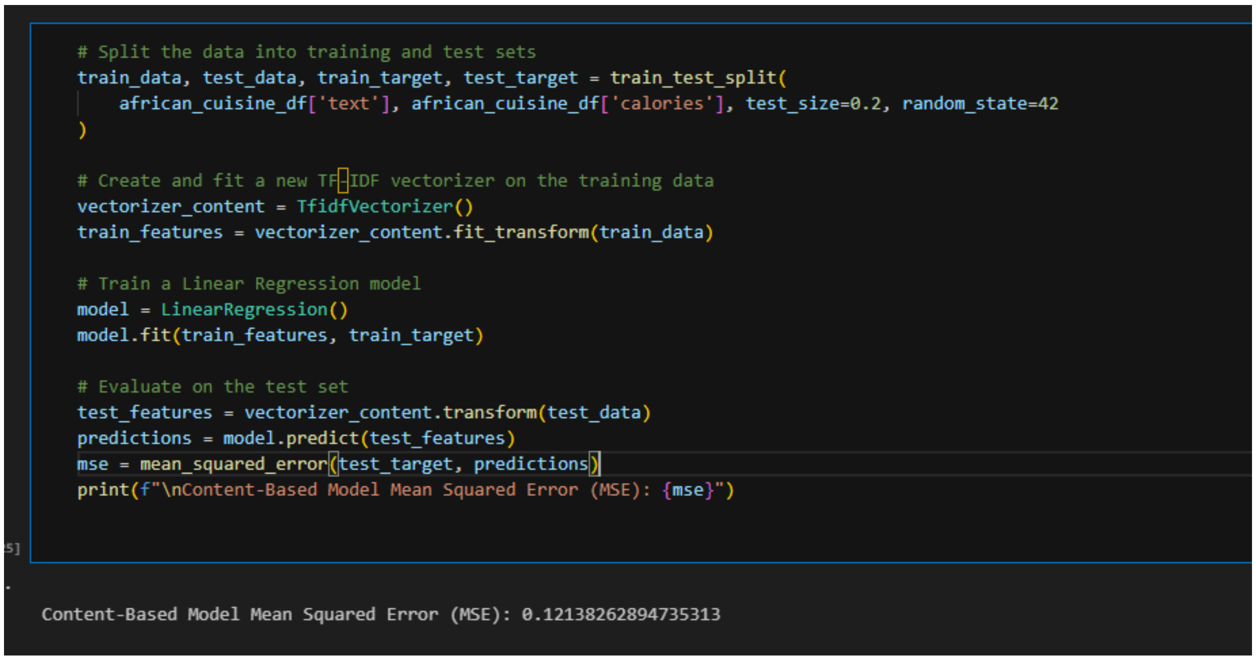
Recommendation Function

- Developed a recommend() function to retrieve the most similar recipes.
- Given a recipe input, the function returns the top-N most similar recipes.

Evaluating the System

- Compared recommended recipes to user preferences.
- Assessed precision and recall to measure recommendation effectiveness.

The MSE had a score of 0.12138262894735313



```
# Split the data into training and test sets
train_data, test_data, train_target, test_target = train_test_split(
    african_cuisine_df['text'], african_cuisine_df['calories'], test_size=0.2, random_state=42
)

# Create and fit a new TF-IDF vectorizer on the training data
vectorizer_content = TfidfVectorizer()
train_features = vectorizer_content.fit_transform(train_data)

# Train a Linear Regression model
model = LinearRegression()
model.fit(train_features, train_target)

# Evaluate on the test set
test_features = vectorizer_content.transform(test_data)
predictions = model.predict(test_features)
mse = mean_squared_error(test_target, predictions)
print(f"\nContent-Based Model Mean Squared Error (MSE): {mse}")
```

```
Content-Based Model Mean Squared Error (MSE): 0.12138262894735313
```

RECOMMENDATION

Marketing Focus

- Promote highly rated and calorie-dense recipes to attract food enthusiasts.
- Highlight these recipes in advertisements, social media, and promotional campaigns.

User Insights & Review Analysis

- Analyze negative reviews across different rating ranges to identify recurring issues.
- Focus on reviews from 2013-2018 to detect significant shifts in sentiment and trends.

Health-Conscious Options

- Introduce low-calorie and nutrient-rich alternatives to cater to health-conscious consumers.
- Provide detailed nutritional breakdowns and healthy ingredient swaps.
- Customer Satisfaction & Quality Improvement
- Address customer complaints to enhance both service and product quality.
- Implement feedback-driven improvements in recipe offerings and user experience

CONCLUSION

- The Sentiment Analysis Model provides valuable insights into customer opinions, allowing businesses to improve their offerings. However, refining text cleaning and model tuning can improve accuracy.
- The Recommendation System successfully suggests recipes but can be enhanced with personalized filtering and user feedback loops.
- Implementing these models in a real-world food app can boost user engagement, improve satisfaction, and increase retention rates.

NEXT STEPS

Build a Chatbot for User Engagement

Chatbot Features:

- Provide recipe suggestions based on user mood (using sentiment analysis results).
- Answer nutrition-related questions (calories, ingredients, dietary preferences).
- Support voice commands for an interactive experience.

Tools & Technologies:

- Use Rasa, Dialogflow, or GPT-based models for chatbot development.
- Deploy on Telegram, WhatsApp, or a web app for user access.