Analysis of a Recipe Dataset with Clustering and Classification

1. Executive Summary

This report explores the challenge of predicting recipe ratings using a dataset of 172299 recipes with 1,235 features, including nutrient contents, ingredients, keywords, season, and reviewer ratings. The initial goal was to develop a regression model to predict continuous ratings. However, after extensive experimentation with regression techniques such as K-Nearest Neighbors (KNN) or random forest it became evident that the models struggled to achieve satisfactory performance, with a best score of only ~41.6%. This poor performance can be attributed to the high dimensionality of the dataset, potential irrelevance of certain features, and the complex, non-linear relationships inherent in the data.

Given the limitations of regression models, the approach was shifted to a clustering-based classification problem. K-Means clustering was employed to group the recipes into four distinct clusters, which revealed underlying patterns in the data. This clustering step significantly improved the performance of the Random Forest classifier, achieving an accuracy of 92% on the test set. The classification results demonstrated strong precision, recall, and F1-scores for the identified clusters, indicating that clustering effectively captured meaningful structures in the dataset.

Despite the success of the clustering and classification approach, several caveats remain, including the need for further feature selection, handling class imbalance, and exploring more advanced models. Moving forward, the focus will shift to leveraging these insights to develop a recommendation system that uses reviewer ratings to suggest personalized recipes to customers. This system will combine collaborative filtering, content-based filtering, and hybrid techniques to provide accurate and relevant recommendations, ultimately enhancing user satisfaction and engagement.

In conclusion, while the initial goal of regressing recipe ratings proved unfeasible, the pivot to clustering and classification yielded promising results. These findings lay the groundwork for the next phase of the project: building a robust recommendation system tailored to user preferences.

2. Introduction

a. Background

In the modern era of digital platforms, recipe recommendation systems have become increasingly important for helping users discover new dishes tailored to their preferences. These systems rely on understanding the factors that influence recipe ratings, such as ingredient composition, nutrient content, and seasonal relevance. However, predicting recipe ratings is a complex task due to the high dimensionality of the data, the presence of noisy or irrelevant features, and the intricate relationships between variables.

b. Problem Statement

Predicting recipe ratings is a challenging task due to the inherently subjective nature of user preferences. Ratings are influenced by a wide range of factors, including personal taste, dietary restrictions, cultural background, and individual health goals. This subjectivity makes it difficult to build a universal model that accurately predicts ratings across diverse user groups. Compounding this challenge is the fact that the dataset used in this project is not tailored to a specific type of customer, such as those with dietary restrictions (e.g., diabetic or weight-loss-focused individuals). Instead, it is a general dataset encompassing 172,299 recipes with 1,235 features, including nutrient contents, ingredients, keywords, season, and reviewer ratings.

The high dimensionality of the dataset, combined with the potential irrelevance of certain features for specific user groups, further complicates the task of predicting ratings. Initial attempts to address this problem using regression models, such as K-Nearest Neighbors (KNN), yielded poor performance, with a best score of only ~41.6%. This suggests that the models struggled to capture the complex, non-linear relationships between the features and the subjective ratings. Additionally, the datasets lack of specificity to particular user segments means that the models may not generalize well to niche audiences with unique preferences or dietary needs.

Given these challenges, the project pivoted from a regression-based approach to a clustering-based classification strategy. By grouping recipes into distinct clusters using K-Means clustering, the aim was to uncover underlying patterns in the data that could improve the performance of classification models. This shift in approach allowed for a more nuanced understanding of the dataset and ultimately led to the development of a Random Forest classifier with an accuracy of 92%. However, the subjective nature of ratings and the generalizability of the dataset remain significant hurdles that need to be addressed in future work.

The ultimate goal of this project evolved to leverage these insights to develop a recommendation system that can suggest personalized recipes to users. This system must

account for the subjectivity of ratings and the diverse needs of different user groups, making it a complex but rewarding challenge.

Problem Statement: Predicting recipe ratings proves challenging due to the large and complex dataset, consisting of 172,299 recipes and 1,235 features, which capture diverse aspects of the recipes.

Objective: Develop a model that can categorize recipes into distinct types based on the 1,235 features, enabling better understanding and organization of the recipe data.

Future work: Build a recommendation system that predicts the type of recipe a given user is likely to highly rate. This system would suggest new recipes of the same type, enhancing user experience and personalization.

3. Data wrangling

The full [dataset] is 522517 recipes from food.com website. We accessed this dataset on the kaggle website as .parquet files. We converted them into panda dataframes for further analysis.

The “recipes” dataframe has 522517 entries x 28 columns. (Figure 1) and the “reviews” dataframe has 1401982 entries x 8 columns. (Figure 2)

```
#overview of the recipe dataframe
recipes.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 522517 entries, 0 to 522516
Data columns (total 28 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   RecipeId                    522517 non-null  float64
 1   Name                        522517 non-null  object
 2   AuthorId                    522517 non-null  int32
 3   AuthorName                  522517 non-null  object
 4   CookTime                    439972 non-null  object
 5   PrepTime                    522517 non-null  object
 6   TotalTime                   522517 non-null  object
 7   DatePublished               522517 non-null  datetime64[us, UTC]
 8   Description                 522512 non-null  object
 9   Images                      522516 non-null  object
 10  RecipeCategory              521766 non-null  object
 11  Keywords                    522517 non-null  object
 12  RecipeIngredientQuantities  522517 non-null  object
 13  RecipeIngredientParts       522517 non-null  object
 14  AggregatedRating            269294 non-null  float64
 15  ReviewCount                 275028 non-null  float64
 16  Calories                    522517 non-null  float64
 17  FatContent                  522517 non-null  float64
 18  SaturatedFatContent         522517 non-null  float64
 19  CholesterolContent          522517 non-null  float64
 20  SodiumContent               522517 non-null  float64
 21  CarbohydrateContent         522517 non-null  float64
 22  FiberContent                522517 non-null  float64
 23  SugarContent                522517 non-null  float64
 24  ProteinContent              522517 non-null  float64
 25  RecipeServings              339606 non-null  float64
 26  RecipeYield                 174446 non-null  object
 27  RecipeInstructions          522517 non-null  object
dtypes: datetime64[us, UTC](1), float64(13), int32(1), object(13)
memory usage: 109.6+ MB
```

Toggle output scrolling

Figure 1: recipes dataframe .info()

```
: #overview of the reviews dataframe
  reviews.info()

  <class 'pandas.core.frame.DataFrame'>
  RangeIndex: 1401982 entries, 0 to 1401981
  Data columns (total 8 columns):
   #   Column         Non-Null Count    Dtype
  ---  ------         --------------    -----
   0   ReviewId       1401982 non-null  int32
   1   RecipeId       1401982 non-null  int32
   2   AuthorId       1401982 non-null  int32
   3   AuthorName     1401982 non-null  object
   4   Rating         1401982 non-null  int32
   5   Review         1401982 non-null  object
   6   DateSubmitted  1401982 non-null  datetime64[us, UTC]
   7   DateModified   1401982 non-null  datetime64[us, UTC]
  dtypes: datetime64[us, UTC](2), int32(4), object(2)
  memory usage: 64.2+ MB
```

Figure 2: reviews dataframe .info()

a. Handling the "Image" Column

Initially, the plan was to create a new column indicating whether a recipe included an image to analyze its potential impact on ratings based on the **`Image`** column. However, this column was ultimately dropped, as 68% of the recipes lacked an associated image, making the feature less meaningful.

b. Handling Missing Values

- Converted all NumPy arrays into tuples to enable the use of isnull(), as it does not work on NumPy arrays.

- Dropped columns with a high percentage of missing values:

  - **`AggregatedRating`** and **`ReviewCount`** (contained nearly 50% missing values).
  - **`RecipeYield`** (35% missing values), as it was redundant with **`RecipeServings`** but provided less valuable information.

- Removed recipes with missing values in:

  - **`RecipeServings`**
  - **`RecipeCategory`**

c. Processing Recipe Preparation Time

- Dropped **`PrepTime`** and **`CookTime`** columns and retained **`TotalTime`** instead.
- Removed recipes where **`TotalTime`** = 0.
- Converted **ISO 8601** duration format into minutes and created a new column: **`TotalTimeMinutes`**.

- Identified extreme outliers in **TotalTimeMinutes** (e.g., a maximum of 43,552,800 minutes). Since most recipes had a total time under 3,000 minutes (~50 hours or ~2 days), we removed all recipes exceeding this threshold.

d. Handling duplicates

No exact duplicates were found, but duplicate values appeared in multiple columns but Duplicate entries in **Recipe Name** and **Author** might indicate updated versions of the same recipe rather than actual duplicates and the date of publication can help to distinguish between original and updated versions. So we dropped duplicates based on **Description, RecipeInstructions, Name, AuthorId**, and **DatePublished** to ensure data integrity.

e. Creation of the rating data

The **AggregatedRating** and **ReviewCount** columns were dropped due to missing values. To recover the ratings, we utilized the reviews dataframe, which contains **RecipeID** along with individual ratings. This allowed us to reconstruct a complete dataset with all available ratings for each recipe. Additionally, we created new columns for the average rating and rating count to provide a more comprehensive overview of recipe evaluations.
In order to match users to recipes and generate the average rating for each recipe we will merge the recipe dataframe and the review dataframe.

We were able to generate **average_rating** values for 172299 recipes by merging the 2 dataframes with an inner join on **RecipeId**.
Generate the **average_rating** column using groupby (**RecipeID)** and transform (mean) rating values. Each of the recipes will be also paired with the different users, their ratings and reviews.

Figure 3 shows the distribution of the ratings. We can observe that there is a strong class imbalance in the rating values with a majority of 4 and very few of 2 and 1. This may cause issues when we will build a model in order to predict the ratings.

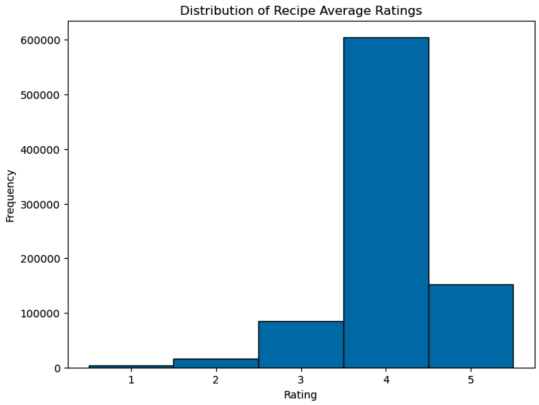

Figure 3: Frequency of the ratings

4. Data EDA

a. How do numerical features of the recipes correlate with each other?

i. Overall correlation matrix

We tested whether there was a correlation between nutrient composition, calorie content, and recipe ratings.

Additionally, we designed several numerical features to explore their relationship with the average rating. Specifically, we created the following metrics to assess the impact of recipe complexity:

- **Ingredient count per recipe** (`ingredient_count`)
- **Description length** (`description_length`)
- **Instruction length** (`instruction_length`)

These features were analyzed to determine their correlation with average ratings by visualizing the overall correlation matrix between the features.

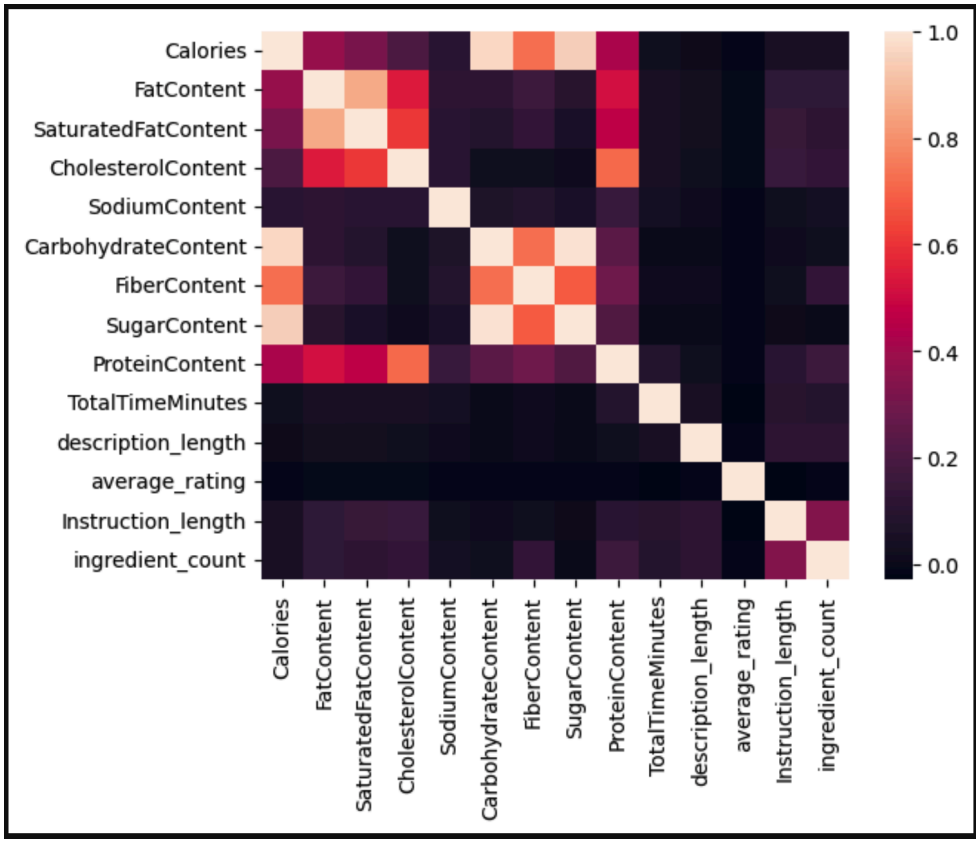

Figure 4: correlation matrix between the numerical features

We did not find any correlation with average ratings; however, we observed that:

- Calories are more strongly correlated with sugar content than with fat content.
- Calories have a stronger relationship with protein and fiber than with fat.

ii. Specific correlation between nutrient content

We simplify the data by adding the different fat contents features together and same for the sugar contents features with also the hope to reinforce the correlation trends.(Figure 5)

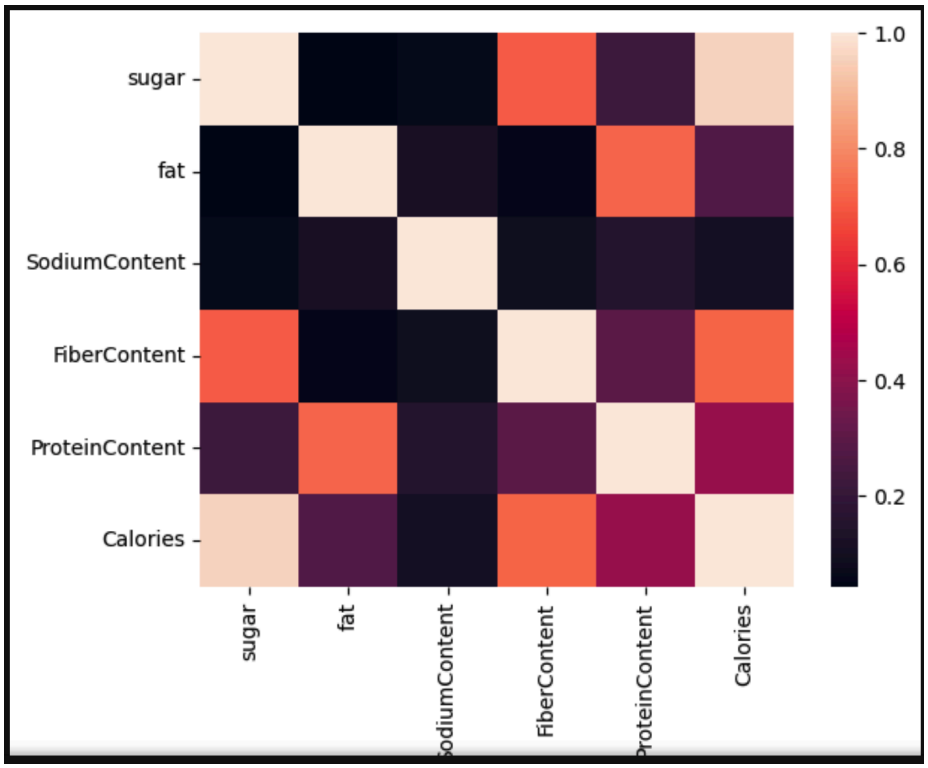

Figure 5: Correlation matrix between simplified nutrient content features

We observe varying levels of correlation with calorie content, ranked from least to most correlated as follows: **sodium < fat < protein < fiber < sugar**. As expected, **sugar content shows the strongest correlation with calories**, surpassing the other nutrients.

iii. Data distribution of the nutrient contents

The data distribution could suggest a poisson distribution (Figure 6) but the variance is much larger than the mean (Table 1). (for a poisson distribution the std should be the square root of the mean). This suggests that the data may be modeled by Negative Binomial distribution, which can account for greater variance relative to the mean.

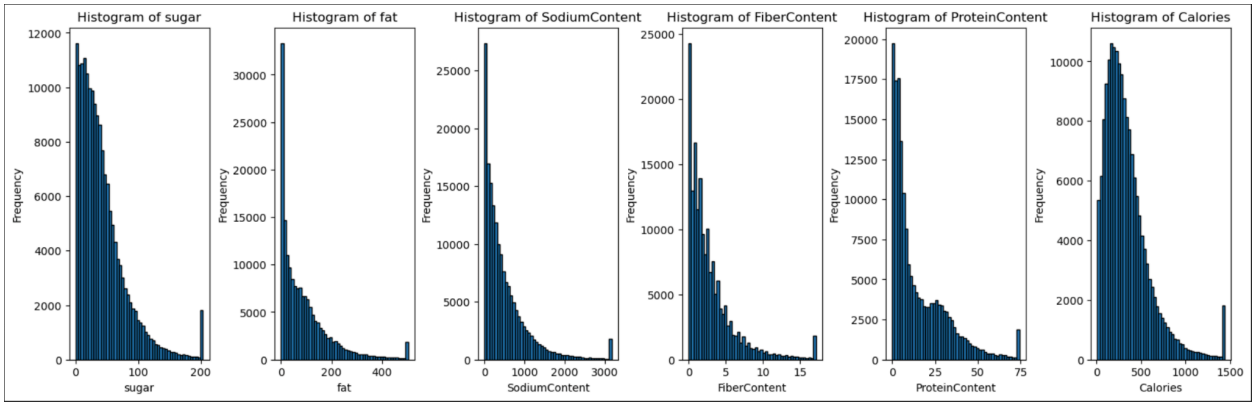

Figure 6: Nutrient data distribution

	sugar	fat	SodiumContent	FiberContent	ProteinContent	Calories
count	172299	172299	172299	172299	172299	172299
mean	46	94	563	3	16	356
std	484	133	2707	6	22	1118
min	0	0	0	0	0	0
25%	16	16	122	1	3	165
50%	34	62	325	2	9	285
75%	58	129	686	4	24	448
max	198977	15623	704130	1749	3270	434360

Table 1: Summary statistics for nutrient content.

b. Statistical analysis for nutrient correlation

We investigated the significance of the Pearson correlation coefficients between nutrient features and calorie content using a permutation strategy.

Our null hypothesis ($\mathbf{H_0}$) states that there is no difference between the Pearson correlation coefficients computed from permuted data and those from the real data. We calculated p-values using the appropriate statistical test based on a normal distribution.

The results show:

- The p-values for the differences between the Pearson correlations from real and permuted data are very small. This suggests that even low correlation values observed in the real data are statistically significant and unlikely to be due to random chance.
- The strong statistical significance confirms that recipes with higher sugar and fiber content tend to have more calories compared to those rich in protein or fat.
- We also observed varying levels of correlation between different nutrient content features. Further analysis confirmed that these differences are statistically significant (small p-values).

In Conclusion, the correlation between nutrient content and calorie counts is both statistically significant and biologically plausible. Therefore, users can confidently rely on the nutritional information provided by this database.

c. Creation of Ingredient clusters for recipe recommendation model

Due to the lack of strong correlations between numerical features and average ratings, building an effective regression or classification model to predict ratings is challenging. Instead, a more promising approach is to cluster recipes based on their ingredient lists, allowing us to group similar recipes and develop a model that predicts recipe clusters rather than individual ratings.

Methodology:

- Standardizing ingredient names using the `pyfood` library to ensure consistency.
- Vectorizing ingredient lists to convert textual data into numerical representations.
- Clustering recipes using K-Means based on ingredient compositions.
- Visualizing clusters using PCA to explore patterns in the data.

Results & Insights:

By clustering recipes based on ingredient composition, we successfully identified meaningful recipe groupings (Figure 7). This approach provides a foundation for alternative modeling strategies, shifting from direct rating prediction to **cluster-based recommendations**. Future steps would be to develop models based on clustering to recommend recipes similar to those highly rated by a given user.

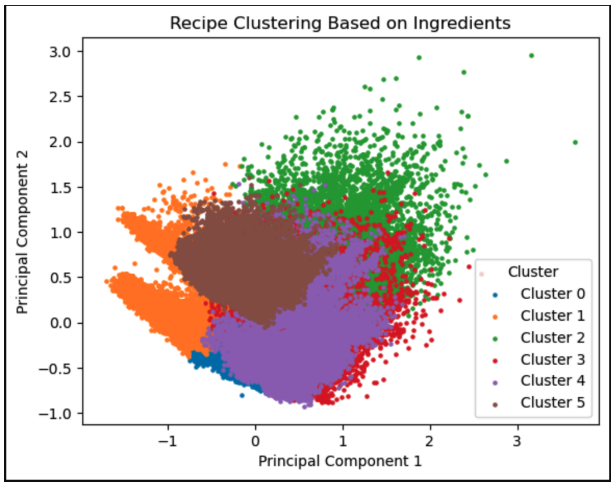

Figure 7: PCA representation of the recipes separated by clusters based on ingredient list

5. Feature engineering

a. Numerical features

- `average_rating` represents the average rating for each authors recipes. It helps evaluate whether an authors average rating is a reliable predictor for the ratings of their future recipes.

- `rating_count` captures the total number of recipes posted by an author. It allows us to analyze the relationship between the quantity of recipes and their ratings.

- `TotalTimeMinutes` is the total preparation and cooking time for a recipe, measured in minutes. It serves as a numerical feature for prediction.

- The complexity of the recipe can be captured by the length of the description (`DescriptionLength`), length of the instruction (l`nstruction_length`) and total ingredient count (`ingredient_count`). It will help assess whether the complexity of a recipe influences the recipes rating.

b. Categorical features

- We extracted seasonal information from the timestamp `DatePublished` column and encoded it as one-hot features using pd.get_dummies().

- One-Hot Encoding the `RecipeCategory` column: we transformed recipe categories into binary one-hot encoded features using OneHotEncoder() with sparse = True for memory and computational efficiency.

- One-Hot Encoding the `Keywords` column: we extracted the unique set of keywords from the `Keyword` column and used it to generate a multi-hot encoding, where each keyword is represented as a separate binary column. To do so we used a dictionary comprehension to create a new column for each unique keyword. For each row, if a given keyword is present in the keyword set, it assigns 1 (indicating presence). Otherwise, it assigns 0. This transformation allows us to capture keyword-related patterns effectively.

- Ingredient Vectorization:
    - We standardized the ingredient names using the `pyfood` library and passed the whole ingredient column through shelf.process_ingredients by chunk to make the process faster. We then joined back to the initial dataframe under `homogen_ingredient` column.

- We joined the elements of the `homogen_ingredient` column into a single string, where each ingredient is separated by a comma and a space (,) in order to prepare the data for vectorization.
- We used CountVectorizer() to tokenize the `homogen_ingredient` ingredient list and then joined it back to the original dataframe.

Our final dataframe has 172299 rows x 1235 columns. It is modeling time!

6. Modeling

a. Methods

i. Data preparation steps

- Splitting training set and testing set
- Scaling training set using StandardScaler()
- Scale Transform test data using training parameters
- PCA transformation to reducer dimensionality

ii. Supervised regression models in order to predict average ratings

- KNN regressor using KNeighborsRegressor(n_neighbors=5)
- Discretizing continuous target feature (average_rating)
- KNN classifier with grid search using GridSearchCV(estimator=knn, param_grid=param_grid, cv=5, n_jobs=-1)
- Random forest classifier using RandomForestClassifier(n_estimators=100, random_state=42)

iii. Unsupervised models in order to identify clusters based on the given features

- We used StandardScaler() on the full dataset before splitting into training and testing set
- We then used the elbow method to find the optimal cluster number. To do so we created a function that calculates the inertia for a range of cluster numbers.
- We visualized the clusters using PCA.
- We splitted the data into training and testing set
- We then used Random Forest classifier to predict the recipe clusters

b. Results

i. KNN Regressor (KNeighborsRegressor(n_neighbors=5))

- Performance metrics:
  - Mean Squared Error (MSE): 1.12

- Root Mean Squared Error (RMSE): 1.06
- Mean Absolute Error (MAE): 0.72
- R^2 Score: -0.16

Despite a relatively low MAE (0.72) and RMSE (1.06), the negative R^2 suggests that the model does not provide meaningful improvements over a simple mean prediction.

ii. KNN Classifier with Grid Search (GridSearchCV)

- Performance metrics:
  - Best Parameters: `{metric: euclidean, n_neighbors: 15}`
  - Best Cross-Validation Score: 41.6%
  - Test Set Accuracy: 41.9%

An accuracy of ~40% indicates that the model struggles to make reliable predictions.

iii. Random Forest Classifier (`RandomForestClassifier`)

- Performance metrics:
  - Precision: 0 for classes 1–4, indicating poor predictive accuracy for these categories.
  - Recall: 0 for classes 1–4, showing difficulty in correctly identifying these classes.
  - F1-Score: Low across classes 1 to 6, with marginal improvement in class 7 (0.62).
  - Overall Accuracy: 46%, meaning that fewer than half of the predictions were correct.
  - Macro & Weighted Averages: Indicate poor overall performance

iv. K Means to define recipe clusters followed by random forest classifier in order to predict clusters of a given recipe.

- Performance metrics:
  - The elbow method shows that the optimal cluster number for k-means clustering would be 4**.** Silhouette score = 0.088: A Silhouette Score of 0.0885 suggests poor clustering, meaning clusters may be overlapping
  - The random forest classification for cluster prediction overall performance accuracy is 92% of predictions indicating strong overall performance.
  - Per-Class Performance:
    - Class 0: High precision (93%) and recall (90%), showing strong performance.
    - Class 1: Extremely low recall (6%), meaning most actual class 1 samples are misclassified. Despite perfect precision (100%), the F1-score (0.11) reflects poor model effectiveness for this class.
    - Class 2: Strong balance with 92% precision and 89% recall.

        - Class 3: Highest recall (96%) and strong precision (92%), indicating excellent performance due to large sample size.
    - Macro Average:
        - Precision: 94%, Recall: 70%, F1-Score: 72%.
        - Highlights poor recall in class 1, reducing overall recall and F1-score.
    - Weighted Average:
        - Precision, Recall, and F1-Score: 92%.
        - Dominated by well-performing larger classes (2 and 3), masking class 1s poor performance.
- Key Observations
    - Class Imbalance: Class 1 has too few samples, leading to poor recall.
    - Model Strengths: High performance for majority classes (0, 2, 3).
    - Model Weaknesses:
        - Poor recognition of minority class 1, limiting its reliability for underrepresented categories.
        - We observe good recall, F1-score, and accuracy but a poor silhouette score, it suggests that the classification model is performing well, but the clustering structure is weak.

v. Clusters Identification

After analyzing the top values for each cluster, we can outline their profiles as follows:

- Cluster 0: Recipes in this cluster have a high protein content and frequently include keywords or ingredients like meat, chicken, egg, and poultry. They typically require a medium preparation time (less than 1 hour).

- Cluster 1: This cluster is notable for its higher sugar and carbohydrate content. Common keywords and ingredients include sugar, butter, egg, and dessert. While some keywords like beef and chicken appear, their frequency is lower compared to Cluster 0. Despite a class imbalance, this cluster distinctly represents dishes rich in fat and sugar.

- Cluster 2: Recipes in this cluster stand out due to longer instructions and a higher count of ingredients. Keywords like healthy, vegetable, olive oil, tomatoes, and fruits are prevalent, indicating a focus on more complex and health-conscious recipes.

- Cluster 3: This cluster features shorter preparation times and includes keywords such as beginner cook, inexpensive, and kid-friendly. These recipes also emphasize health, with lower carbohydrate and protein content, and include ingredients like vegetables and fruits.

7. Discussion

a. Key Challenges in predicting the average ratings with regression and classification models

i. **Imbalanced Rating Distribution:**

- The dataset is skewed toward certain ratings (e.g., most recipes are rated 4 or 5), leading models like Random Forest to favor majority classes.
- This imbalance poses significant challenges for both regression and classification tasks.

ii. **Subjectivity of Ratings:**

- Ratings are influenced by subjective factors such as user preferences and cooking skills, which are not well-represented in the available features.
- This makes it difficult to build an accurate predictive model.

iii. **Lack of Correlation** between Nutrient Content and Correlation analysis from the exploratory data analysis (EDA) shows that nutrient content does not strongly influence ratings. This may be due to the dataset being sourced from a general-purpose recipe collection rather than one tailored for specific dietary needs (e.g., low-carb, gluten-free).

b. Key challenges for K-Means clustering and cluster predictions

i. **Low silhouette score but good performance metrics for classification into clusters**

This could be explained by:

- Overlapping Clusters but the classification model can still perform well if it learns decision boundaries that separate classes effectively.
- The data has many features, the distance between points becomes less meaningful in K-Means, leading to poor cluster separation. But a classifier using advanced models like Random Forest can still learn non-linear patterns and perform well.

ii. **Under represented cluster 1**

To improve performance we can further push the modeling by using diverse strategies to address the class imbalance:

- **Oversampling the Minority Class**
  - *SMOTE (Synthetic Minority Over-sampling Technique)*: Generates synthetic samples for the minority class.
  - *ADASYN (Adaptive Synthetic Sampling)*: Similar to SMOTE but focuses more on difficult-to-learn samples.
  - *Random Oversampling*: Duplicates samples from the minority class.
- **Undersampling the Majority Class**
  - *Random Undersampling*: Removes some majority class samples to balance the dataset.
  - *Cluster-Based Undersampling*: Uses clustering techniques to retain diverse samples while reducing redundancy.
- **Combination of Over- and Undersampling**

    - *SMOTE + Tomek Links* or *SMOTE + Edited Nearest Neighbors (ENN)*: Balances the dataset while removing borderline/noisy samples.
- **Class Weighing**
    - Assigns higher weights to the minority class during model training.
    - Supported in models like RandomForestClassifier(class_weight="balanced") and LogisticRegression(class_weight="balanced").
- **Cost-Sensitive Learning**
    - Modifies the loss function to penalize misclassifications of the minority class more heavily.
    - Works well with algorithms like Support Vector Machines (SVMs) and Neural Networks.

8. Recommendation for future modeling projects

To enhance the utility of these insights, the focus can be shifted to creating a strategy for Personalized Recipe Recommendations using the following approaches:

- **Personalized Recipe Recommendations Using Clustering**

By leveraging clustering results, we can recommend recipes that align with a user's historical preferences. For example, a user frequently selecting high-protein recipes can be matched with Cluster 0. User feedback (ratings) is incorporated to refine and improve recommendations over time.

To validate the model, we utilize the review dataset, which contains individual user ratings for recipes. By recommending recipes from the same cluster as those a user has rated 4 or 5 stars, we ensure personalized and relevant suggestions.

- **Collaborative Filtering (CF)**

CF recommends recipes based on user interactions (e.g., ratings, clicks, purchases). It assumes users with similar past behavior will have similar preferences. We can recommend recipes preferred by users with similar tastes or **recommend recipes frequently liked together.**

- **Content-Based Filtering (CBF)**

CBF recommends recipes based on their attributes (e.g., ingredients, cuisine type, nutritional values). It builds a profile of user preferences based on previously interacted items.

In conclusion, by transitioning to unsupervised learning and leveraging clustering insights, this strategy can address the challenges posed by the dataset and offer valuable tools for recipe assignment and personalized user experiences.