RECIPE RECOMMENDER ASSIGNMENT EDA

By Pratik Punyawant, Manthan Sumbhe, Kushal

CONTENT

FEATURE EXTRACTION

Part1

TASK 1: READ THE DATA

The data is read using the RAW_recipes_cleaned.csv.

The counting occurrences of the data is around 2.3 lakhs.

The data type is checked through the schema.

TASK 2: EXTRACT INDIVIDUAL FEATURES FROM THE NUTRITION COLUMN.

Objective: Transform the nutrition column in raw_recipes_df into individual columns for each nutrition attribute.

Process:

- Step 1: Remove square brackets from the nutrition column for cleaner data formatting.
- Step 2: Split the cleaned nutrition string into individual values separated by commas.
- Step 3: Create separate columns for each nutrition attribute using a loop, ensuring each value is accurately typecast to float.

Validation: Assertions confirm correct typecasting and proper splitting of values into columns.

Outcome: Seven new columns are generated for individual nutrition attributes, enhancing data accessibility and enabling detailed analysis.

TASK 3: TRANSFORMING NUTRITION DATA AND VALIDATING NULL VALUES

Goal: Break down the nutrition data to show values per 100 calories, providing a standardized way to compare different nutrients across recipes.

Process:

- Calculated new columns for each nutrition metric (e.g., fat, sugar) on a per-100-calorie basis, making it easier to analyze nutrient density without the influence of varying calorie counts.
- Excluded calories from this transformation since it serves as the reference point for the calculations.
- Filled any resulting gaps with zeros to ensure every column has complete data.

Quality Checks:

- Validated calculations against specific recipe IDs to confirm accuracy.
- Ran checks for missing values to guarantee that all per_100_cal columns are ready for reliable analysis.

Result: Successfully generated and validated nutrition columns on a per-100-calorie basis, making the data cleaner, more consistent, and ready for meaningful insights.

TASK 4: CLEANING AND PREPARING RECIPE TAGS AND RATINGS DATA

Objective: Enhance the recipe dataset by cleaning and structuring the tags column while integrating user ratings for comprehensive analysis.

Tags Processing:

- Removed unwanted characters from the tags column, including brackets and quotes, to achieve a clean string format.
- Split the cleaned tags string into an array format using a comma delimiter, allowing for easier analysis of individual tags related to each recipe.

Ratings Data Integration:

- Imported a separate dataset containing user ratings, ensuring proper schema and data types for seamless merging.
- Adjusted the review_date column for clarity by renaming it while dropping the original date column.

Validation:

- Confirmed successful transformation of the tags column into an array type.
- Verified the integrity of the ratings data, ensuring accurate record counts and correct column structures.

Outcome: The dataset is now enriched with structured tags and comprehensive ratings, setting the stage for deeper insights into recipe performance and user preferences.

TASK 5: MERGING RECIPE AND RATINGS DATA FOR INTERACTION ANALYSIS

Purpose: Bring together recipe details and user ratings to understand how users engage with each recipe.

Steps Taken:

- Merged the two datasets using an inner join on recipe IDs, ensuring that only recipes with existing user ratings are included.
- This unified view now connects each recipe with its corresponding user feedback, making it possible to explore user behavior and preferences.

Advantages:

- Allows us to see which recipes are most popular, how users rate them, and overall engagement levels.
- Sets the foundation for understanding what users like, helping to guide recommendations and identify trending recipes.

Result: The combined interaction_level_df dataset is ready for in-depth analysis, offering rich insights into user interactions with each recipe.

TASK 6: CALCULATING TIME DIFFERENCES BETWEEN RECIPE SUBMISSION AND REVIEW DATES

Objective: Add time-based insights to analyze the duration between when recipes were submitted and when users reviewed them.

Approach:

- Converted the submitted and review_date columns to date format for accurate time calculations.
- Calculated three metrics to capture time differences:
  - Days since recipe submission for each review.
  - Months since submission, allowing for a broader view of engagement over time.
  - Years since submission by dividing months by 12, useful for observing longer-term trends.

Benefits:

- Enables tracking how long it takes for users to review a recipe after submission.
- Helps identify recipes that continue to attract interest long after submission, revealing their sustained popularity.

Outcome: The dataset now includes time-based columns for deeper insights into user engagement patterns and recipe longevity.

EDA

Exploratory data analysis

CUSTOM FUNCTIONS FOR DATA ANALYSIS AND BUCKETIZATION

Purpose: Developed functions to simplify exploratory data analysis, enabling efficient quantile calculations, bucketized summaries, and custom statistics for recipes and reviews.

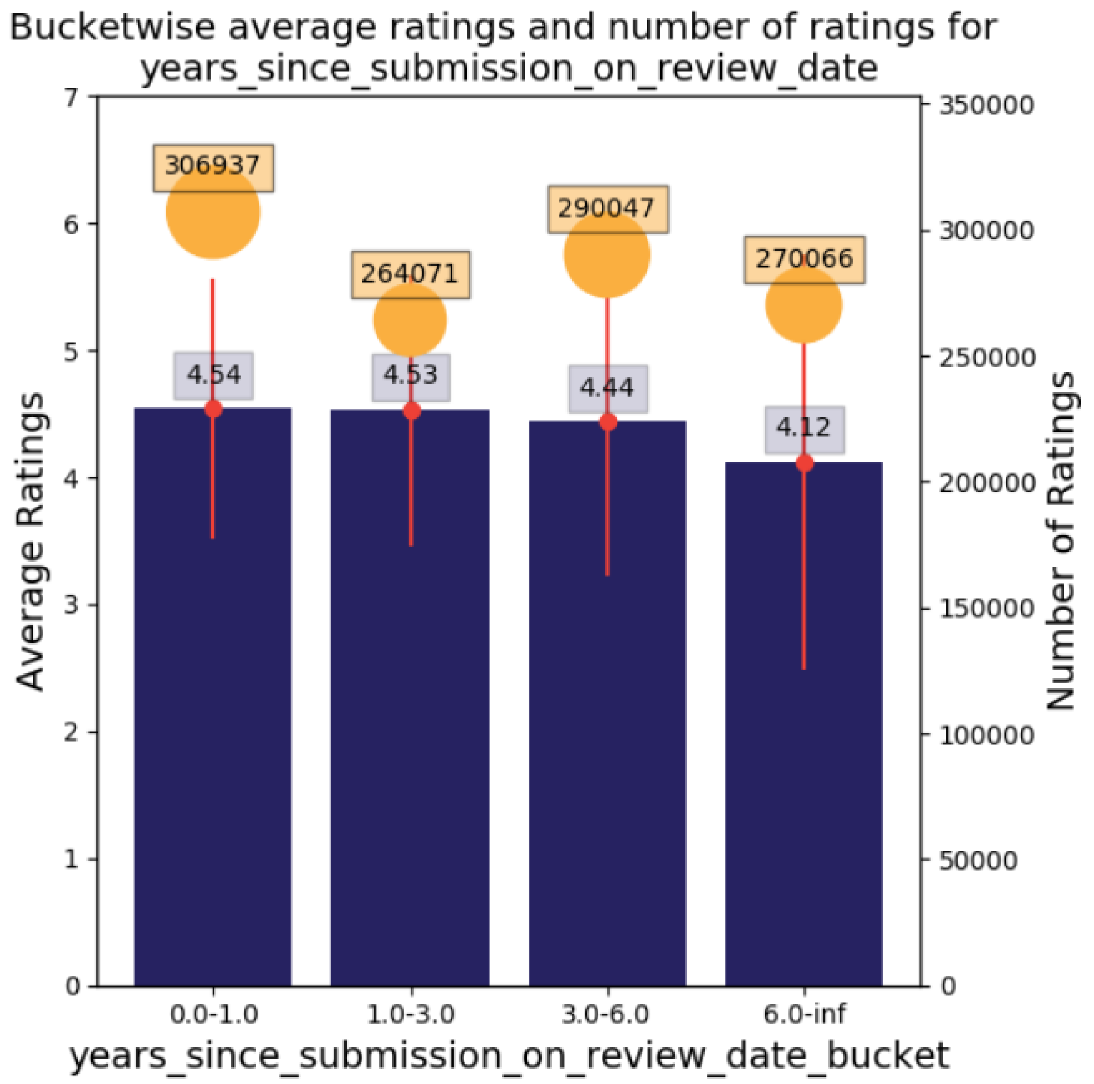
Core Functions:

- **get_quantiles**: Quickly finds key quantile values for a numerical column, helping identify distribution insights (e.g., median, 99th percentile).
- **bucket_col_print_summary**: Buckets a numerical column based on defined splits, producing summary statistics like average ratings and count for each bucket.
- **plot_bucketwise_statistics**: Visualizes average ratings and number of ratings for each bucket, aiding interpretation of distribution patterns and variability.
- **get_column_distribution_summary**: Displays statistics for unique values in a column, including average ratings, count, and distinct recipe count, offering insights on categorical data.
- **get_n_items_satisfying_condition**: Calculates and displays the number and percentage of rows satisfying a given condition, useful for filtering specific data insights (e.g., finding recipes with zero prep time).

Outcome: These functions enable deeper analysis, visualization, and condition-based filtering, streamlining data exploration in recipe and review datasets.

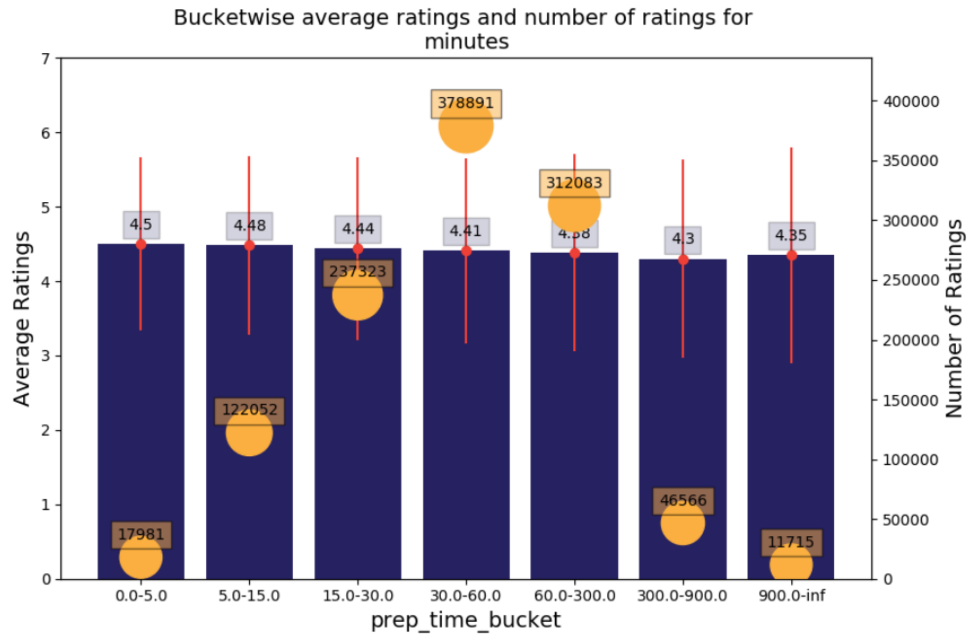
DATA LOADING, FILTERING, AND QUANTILE ANALYSIS

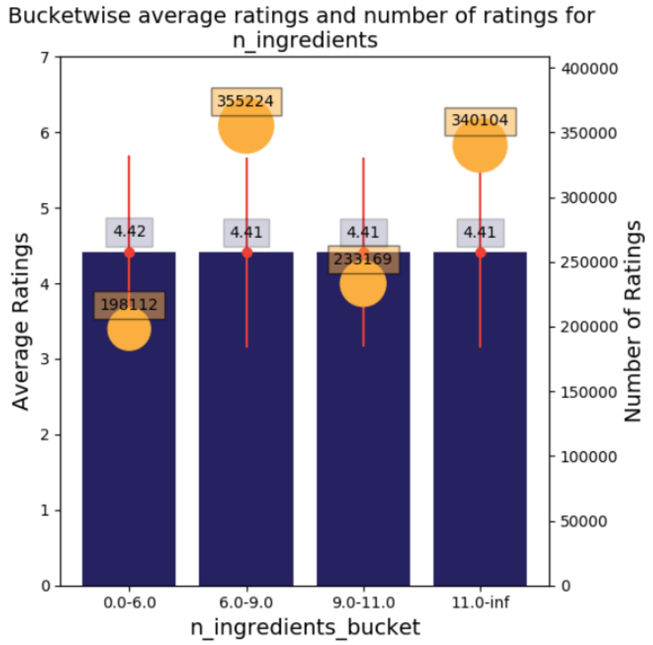
- Data Loading & Validation:Loaded interaction-level data from S3 and confirmed data structure by validating row and column counts.
- Quantile Analysis:
  - Examined quantiles for key features like years_since_submission_on_review_date, minutes, n_steps, and n_ingredients to understand data spread and identify outliers.
  - Insights guided thresholds for filtering out invalid entries.
- Condition-Based Filtering:
  - Applied filters to remove invalid entries where:
    - years_since_submission_on_review_date is negative.
    - minutes or n_steps are zero, ensuring all recipes have valid preparation times and steps.

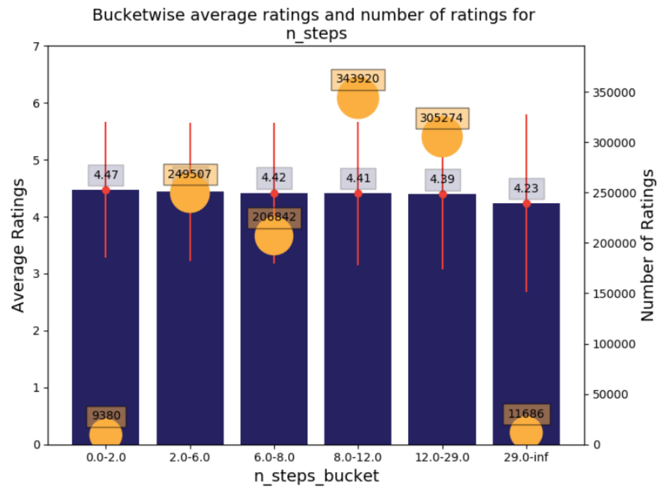


FEATURE BUCKETIZATION, NUTRITIONAL SUMMARY, AND DATA STORAGE

- Feature Bucketization:
  - Segmented key features (years_since_submission, prep_time, n_steps, n_ingredients) into specific value ranges to facilitate data analysis.
  - Used buckets to observe data distribution and visualize rating trends within each range.
- Nutritional Quantile Summary:
  - Created quantile-based buckets for various nutritional attributes.
  - Analyzed how average ratings varied across these quantiles to observe potential impact on user satisfaction.
- Data Storage:
  - Saved the post-EDA dataset back to S3, preparing a clean dataset ready for advanced analytics and further recommendations.







FEATURE EXTRACTION

Part 2

INITIAL SETUP

Setting Up Spark Environment

- Import required libraries: pyspark, matplotlib, pandas, numpy.
- Create a Spark session with SparkSession.builder.appName("Basics").getOrCreate().
- Install necessary Python packages: plotly, pandas, numpy, and matplotlib using spark.sparkContext.install_pypi_package.

Defining Helper Functions

- get_quantiles(df, col_name): Returns quantiles for a specified numerical column.
- plot_bucketwise_statistics(summary, bucketizer): Visualizes summary statistics for bucketed data.
- bucket_col_print_summary(df, splits, inputCol, outputCol): Buckets a numerical column and prints statistics.
- get_column_distribution_summary(df, col_name): Displays summary statistics for unique values in a column.

TAGS LEVEL EDA

Data Processing Steps

- Read data from Parquet format using spark.read.parquet.
- Calculate user-specific statistics (e.g., average rating, number of ratings) using window functions.

Nutrition Analysis

- Define nutrition columns and compute user averages for these metrics.
- Create one-hot encoded columns for tags with add_OHE_columns(df, n_name_list).

Condition-Based Analysis

- get_n_items_satisfying_condition(df, condition, aggregation_level): Analyzes how many recipes or reviews meet a specific condition, providing counts and percentages.

END