A Report on

Recipe Recommender

A Nutritional Distance-Based Approach

Submitted by :

Benitha Gadupudi - G31142871

Divya Sree Vadlamudi - G48698217

Radhika Raghuwanshi - G32395729

Submitted to:

Dr. Armin Mehrabian

Dr. Sardar Hamidian

Course Code: CSCI6364

Course Name: Machine Learning

Date: 26 April, 2024

Objective:

Our objective is to create a recipe recommendation system utilizing a comprehensive dataset sourced from Allrecipes.com, a prominent food-oriented social network boasting 1.5 billion annual visits. The dataset encompasses a wide array of recipes spanning from the years 2000 to 2018, including detailed information on ingredients, user ratings, and interactions. This kaggle data was collected by crawling Allrecipes.com, yielding a dataset comprising 52,821 recipes across 27 distinct categories. Our project focuses on using this data to find the main bits of nutritional info, like how much protein, fat, or carbs are in a recipe, which will then help us see which recipes are similar in terms of their healthiness and dietary content. We implemented Cosine, Euclidean, levenshtein and Hamming distances to quantify the similarity between recipes, leveraging nutritional data for recommendation purposes.

Tasks:

1. Data Loading and Preprocessing

Dataset information:

Our Data set consists of 49698 images and recipe information that needs to be processed. Each recipe's information is meticulously organized within a singular line, encapsulating several essential attributes. These attributes encompass the recipe's unique identifier, its name, the average ratings bestowed upon it by reviewers, a URL linking to its image, the count of reviews it has garnered, a detailed list of ingredients, step-by-step cooking instructions, nutritional data, and the textual content of reviews.

Loading the data:

Raw-data_recipe.csv file is loaded into a data frame named recipe using pd.read_csv.

The image illustrates the structure of data loaded into recipe .

	recipe_id	recipe_name	aver_rate	image_url	review_nums	ingredients	cooking_directions	nutritions	reviews
0	222388	Homemade Bacon	5.000000	https://images.media-allrecipes.com/userphotos...	3	pork belly^smoked paprika^kosher salt^ground b...	{'directions': u'Prep\n5 m\nCook\n2 h 45 m\nRe...	{u'niacin': {u'hasCompleteData': False, u'name...	{8542392: {'rating': 5, 'followersCount': 11, ...
1	240488	Pork Loin, Apples, and Sauerkraut	4.764706	https://images.media-allrecipes.com/userphotos...	29	sauerkraut drained^Granny Smith apples sliced^...	{'directions': u'Prep\n15 m\nCook\n2 h 30 m\nR...	{u'niacin': {u'hasCompleteData': False, u'name...	{3574785: {'rating': 5, 'followersCount': 0, '...
2	218939	Foolproof Rosemary Chicken Wings	4.571429	https://images.media-allrecipes.com/userphotos...	12	chicken wings^sprigs rosemary^head garlic^oliv...	{'directions': u"Prep\n20 m\nCook\n40 m\nReady...	{u'niacin': {u'hasCompleteData': True, u'name'...	{13774946: {'rating': 5, 'followersCount': 0, ...
3	87211	Chicken Pesto Paninis	4.625000	https://images.media-allrecipes.com/userphotos...	163	focaccia bread quartered^prepared basil pesto^...	{'directions': u'Prep\n15 m\nCook\n5 m\nReady ...	{u'niacin': {u'hasCompleteData': True, u'name'...	{1563136: {'rating': 5, 'followersCount': 0, '...
4	245714	Potato Bacon Pizza	4.500000	https://images.media-allrecipes.com/userphotos...	2	red potatoes^strips bacon^Sauce:^heavy whippin...	{'directions': u'Prep\n20 m\nCook\n45 m\nReady...	{u'niacin': {u'hasCompleteData': True, u'name'...	{2945555: {'rating': 5, 'followersCount': 6690...

Preprocessing the data:

The columns image_url, cooking_directions, reviews are dropped from the dataset and loaded into the recipe.

The average rating of the recipe is rounded to 2 decimal places and converted the data type of the column to float.

Total number of unique recipes are found so as to avoid repetitions in the data.

recipe id and recipe name is stored into a data frame recipe_names to be used by the model.

Nutritions column from the data are converted from string to dictionary.

Then percent daily values are extracted for selected important nutritions like calories, fat, carbohydrates, protein, cholesterol, sodium, fiber and all the nutritions data is grouped into a data frame.

This is what the nutritions data looks like after being grouped into a data frame

```
# grouping all the nutritions data into dataframe
data = {'calories': calories_list, 'fat': fat_list, 'carbohydrates': carbohydrates_list,
       'protein': protein_list, 'cholesterol': cholesterol_list, 'sodium': sodium_list,
       'fiber': fiber_list}

df = pd.DataFrame(data)
df.index = recipe['recipe_id']
df.head()
```

recipe_id	calories	fat	carbohydrates	protein	cholesterol	sodium	fiber
222388	15	36	< 1	42	21	81	2
240488	19	18	10	73	33	104	41
218939	17	36	2	48	24	31	4
87211	32	45	20	65	20	43	18
245714	8	12	5	14	7	8	3

Handling missing values:

Then the data frame is examined for null values

As there are only 963 null values in 49698 data values the null values are removed.

The string data in the data frame are imputed into numeric values to clean the text from all columns.

Normalizing the data:

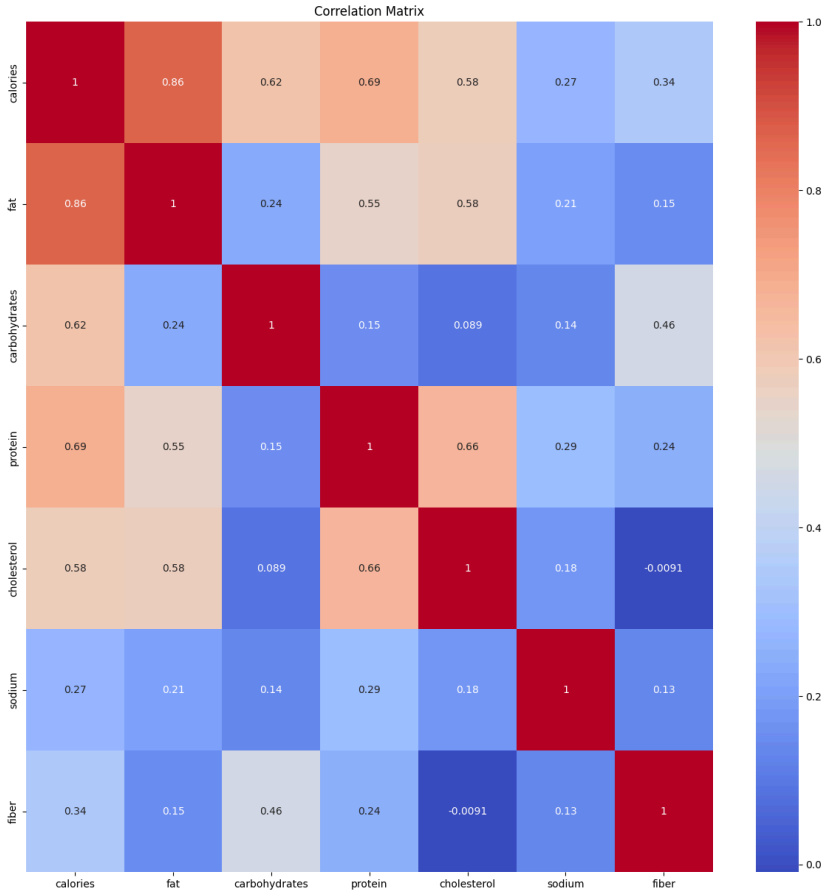
The nutrition data values are normalized by columns**.** By standardizing the range of independent variables or features within a dataset, data normalization is done to enhance consistency and comparability. Consequently, it gave more stable and reliable outcomes.

The below is the representation of the data frame after normalization of data

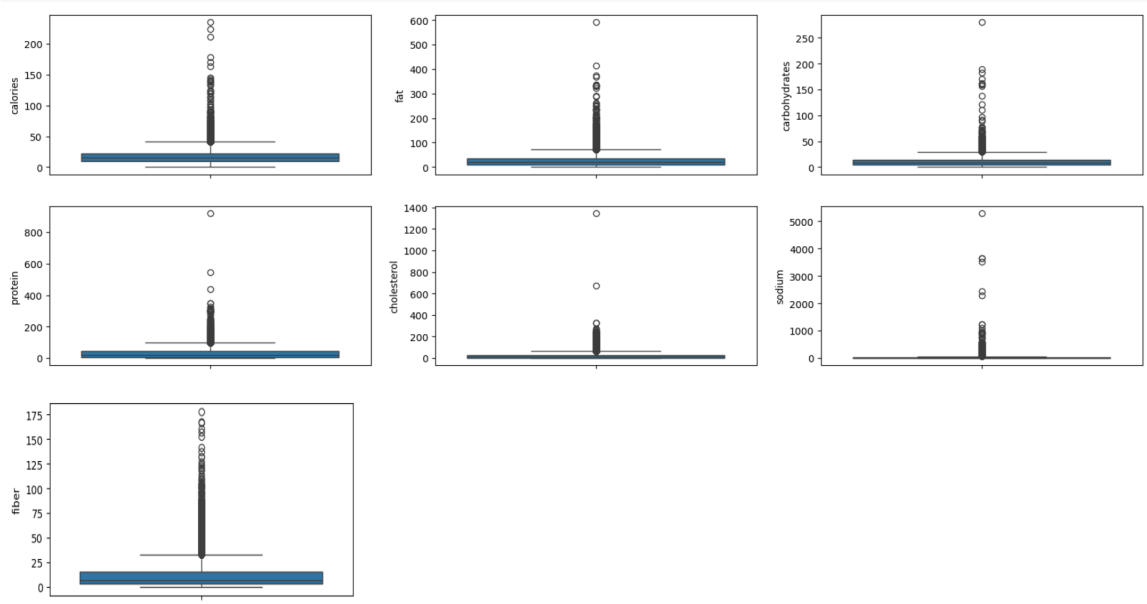
recipe_id	calories	fat	carbohydrates	protein	cholesterol	sodium	fiber
222388	0.003457	0.004775	0.000352	0.004739	0.002997	0.006572	0.000560
240488	0.004378	0.002387	0.003517	0.008236	0.004710	0.008438	0.011477
218939	0.003917	0.004775	0.000703	0.005416	0.003425	0.002515	0.001120
87211	0.007374	0.005969	0.007034	0.007334	0.002855	0.003489	0.005039
245714	0.001844	0.001592	0.001759	0.001580	0.000999	0.000649	0.000840

2. Data Visualization:

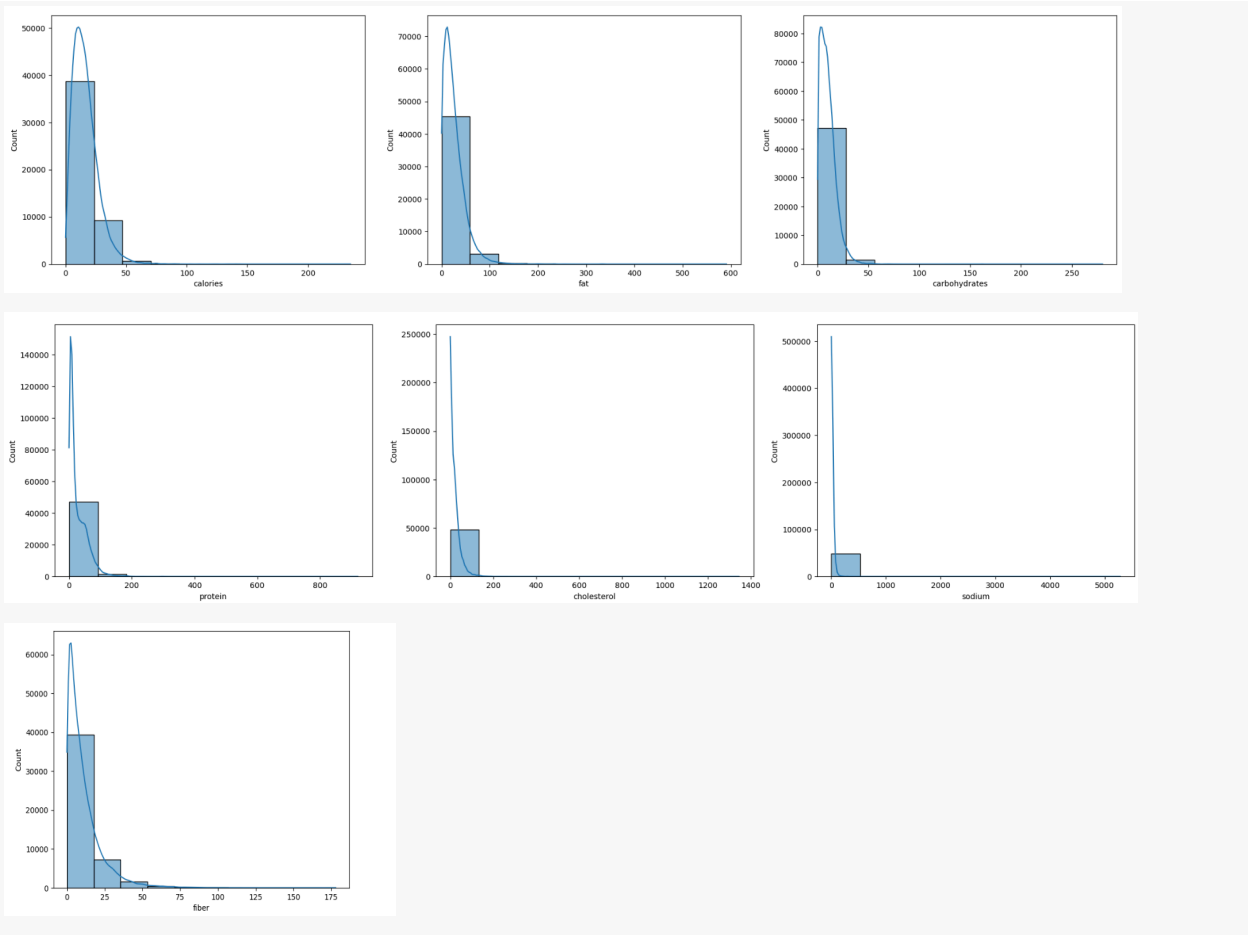
We visualized correlation between the nutrition values with the help of a correlation matrix.



We also obtained box plots to get better understanding of the data



We obtained a histogram to visualize the descriptive statistics of each variable before normalizing the data



3. Model Implementation

KNN Model:

K-Nearest Neighbors is a simple yet effective algorithm used for both classification and regression tasks in machine learning. It's a non-parametric and instance-based learning method where the model makes predictions based on the majority class (for classification) or average value (for regression) of its k nearest neighbors in the feature space. KNN relies on a distance metric (e.g., Euclidean distance, Manhattan distance, cosine similarity) to measure the similarity between instances.

The k-Nearest Neighbors (KNN) recommender is used for our model to find similar recipes based on a selected recipe (recipe_id) it calculates the distance between nutrition values of the selected recipe and all the members of the dataset. The KNN recommender returns the nutrition data of the selected recipe (recipe_id) and the top recommended recipes. We mention the number of recipes to be recommended and which distance metric to be used from the list of Euclidean, Hamming and Cosine similarity along with the selected recipe id.

```
selected_recipe(79774)
```

79774 All-Purpose Marinara Sauce

The output below shows the recommended recipes using cosine similarity as a distance metric and recommending three recipes.

```
knn_recommender(cosine, 79774, 3)
```

```
time cost: 9.43067 sec
```

recipe_id	calories	fat	carbohydrates	protein	cholesterol	sodium	fiber
79774	0.000461	0.000398	0.000703	0.000338	0.0	0.001298	0.001680
159723	0.001383	0.001061	0.002110	0.000790	0.0	0.003570	0.005039
25725	0.000922	0.000663	0.001407	0.000451	0.0	0.002434	0.003079
88986	0.000691	0.000398	0.001055	0.000451	0.0	0.001623	0.002239

159723 Richard and Suzanne's Famous Spaghetti Sauce

25725 Garden Tomato Soup

88986 Gazpacho IV

The output below shows the recommended recipes using euclidean distance as a distance metric and recommending three recipes.

```
knn_recommender(euclidean, 79774, 3)
```

```
time cost: 6.05489 sec
```

recipe_id	calories	fat	carbohydrates	protein	cholesterol	sodium	fiber
79774	0.000461	0.000398	0.000703	0.000338	0.000000	0.001298	0.00168
244415	0.000461	0.000265	0.000703	0.000338	0.000143	0.001298	0.00168
23967	0.000691	0.000531	0.000703	0.000338	0.000000	0.001460	0.00168
38507	0.000461	0.000265	0.000703	0.000338	0.000000	0.001217	0.00140

244415 Broccomoli Dip

23967 Chunky Marinara Sauce

38507 Green Papaya Salad

The output below shows the recommended recipes using hamming distance as a distance metric and recommending three recipes.

```
knn_recommender(hamming, 79774, 3)
```

```
time cost: 8.35446 sec
```

recipe_id	calories	fat	carbohydrates	protein	cholesterol	sodium	fiber
79774	0.000461	0.000398	0.000703	0.000338	0.0	0.001298	0.001680
78545	0.000461	0.000398	0.000703	0.000451	0.0	0.000162	0.001680
90547	0.000461	0.000133	0.000703	0.000338	0.0	0.000649	0.001680
140294	0.000461	0.000265	0.000703	0.000338	0.0	0.001298	0.002239

78545 Sprouted Lentil Salad

90547 Shirazi Salad

140294 Smoky Four-Pepper Salsa

Hybrid Nutrition Recommender:

This model serves as a comprehensive recommender system that leverages multiple distance metrics to generate diverse and potentially more accurate recipe recommendations based on nutritional data. It combines the strengths of different distance approaches and allows users to prioritize recommendations based on specific criteria such as rating and review count. This hybrid recommender system provides a flexible and customizable solution for recommending recipes based on nutritional similarity and user-defined preferences. We implemented four distance metrics where two recommendations from each metric are taken into account and based on the average rating or review numbers the user specified number of recommendations would be chosen from the eight recommended.

Metrics used for Distance calculation:

1. Cosine Distance: The cosine distance measures the cosine of the angle between two vectors in a multidimensional space.
2. Euclidean Distance: The Euclidean distance (L2 norm) measures the straight-line distance between two points in a multidimensional space.
3. Levenshtein Distance : The Levenshtein distance (edit distance) measures the minimum number of single-character edits (insertions, deletions, substitutions) required to transform one string into another.
4. Hamming Distance: The Hamming distance measures the number of positions at which corresponding bits differ.

```
selected_recipe(22886)
```

22886 Grilled Mediterranean Vegetable Sandwich

The output below shows the recommended recipes by a hybrid nutrition recommender using review numbers as a metric to select top three recipes.

```
nutrition_rn, topN_rn = nutrition_hybrid_recommender4(22886, ['review_nums'], 3)
```

```
78007 Mediterranean Vegetable Stew
22886 Grilled Mediterranean Vegetable Sandwich
231104 Sandy's Greek Pasta Salad
time cost: 80.40569 sec
```


```
nutrition_rn
```

recipe_id	calories	fat	carbohydrates	protein	cholesterol	sodium	fiber
22886	0.004148	0.003051	0.005627	0.002031	0.000143	0.001623	0.006158
22886	0.004148	0.003051	0.005627	0.002031	0.000143	0.001623	0.006158
78007	0.002535	0.001724	0.003869	0.001467	0.000000	0.001623	0.008958
231104	0.003917	0.002520	0.005627	0.002369	0.000428	0.001298	0.005599

```
topN_rn
```

	recipe_id	aver_rate	review_nums
30897	22886	4.58	108
34962	78007	4.22	39
35766	231104	4.25	8

Model Evaluation:

The evaluation of Recipe recommendation system is done by establishing ground truth based on predefined criteria (average rating and review numbers) and computing key evaluation metrics to assess the quality and coverage of the recommendations compared to the ground truth dataset. The computed metrics provide insights into the effectiveness and relevance of the recommendation system based on real-world data and user preferences.

We used metrics like Precision, Recall, Coverage, True positives, False positives and False Negatives to get accuracy of our model.

Precision: Precision measures the accuracy of positive predictions made by the model. It is the ratio of true positive predictions to all positive predictions made by the model (true positives + false positives).

Recall (or Sensitivity): Recall quantifies the ability of the model to identify all relevant instances of a positive class. It is the ratio of true positive predictions to all actual positive instances (true positives + false negatives).

Coverage (or True Positive Rate): Coverage is synonymous with recall and represents the proportion of actual positive cases that the model correctly identifies.

Challenges:

Handling Large Datasets Efficiently:

Working with a large dataset posed significant challenges, particularly with uploading and managing images. Uploading all images initially was not feasible due to resource constraints, requiring iterative approaches for data processing and image handling.

Evaluation Model Selection and Implementation:

Determining the appropriate methods to evaluate the performance of the recommendation model proved to be a complex task. Identifying suitable evaluation metrics and understanding how to interpret them accurately were ongoing challenges during the development and testing phases of the project.

Conclusion:

In conclusion, this project aimed to develop a recipe recommender system leveraging machine learning techniques and large-scale data processing. Through the implementation of models like KNN and hybrid nutrition recommender , we successfully created a functional recommendation engine. In this project, we focused on developing a recipe recommendation system that leverages nutritional information for similarity analysis and recommendation purposes. Key components of the project included feature extraction, distance calculation using various methods, and evaluation of system performance.