**Recipe Recommendation System Report**

*Blessy Kuriakose, Amrutha Veeramachaneni, Rohith Reddy Bollareddy, Nikhil Reddy Vemireddy, Brijesh Rao Pamara*

**Abstract**

The aim of this project is to create a web application that outputs food recipe recommendations given an input of ingredients. The user can select the type of foods they are interested in, then enter the ingredients they have at their disposal. From there, an algorithm will identify recipes from within a predetermined repository that best suits the user requirements. All the possible, or best conformed, recipes will be ranked by their rating and the top ones will be made available to the user via the web platform.

**Data specification**

The repository of recipes for this system is sourced from a data set on Kaggle. It is a document containing 28 features (columns) and 522,517 recipes (rows). For the purposes of this project, we did not need all 28 features, but there were some features that were pertinent and could not be compromised. Upon analysis, we found that some of the recipes contained null values for these required features, so in the data preprocessing stage of our program, we dropped these recipes together with the unnecessary features.

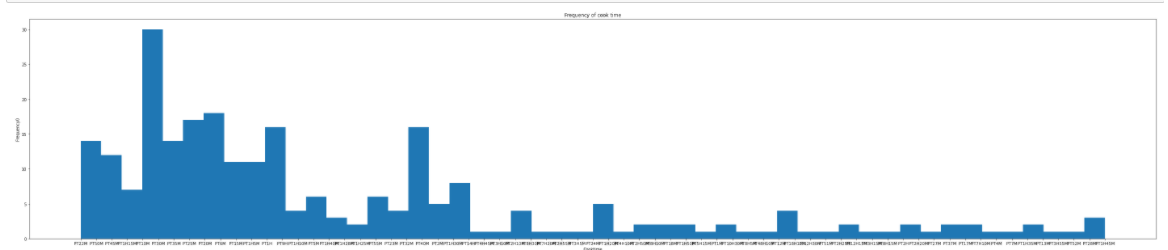
The remaining 17 features include:

['Name', 'TotalTime', 'Description', 'RecipeCategory', 'Keywords', 'RecipeIngredientParts', 'AggregatedRating', 'Calories', 'FatContent', 'SaturatedFatContent', 'CholesterolContent', 'SodiumContent', 'CarbohydrateContent', 'FiberContent', 'SugarContent', 'ProteinContent', 'RecipeInstructions']

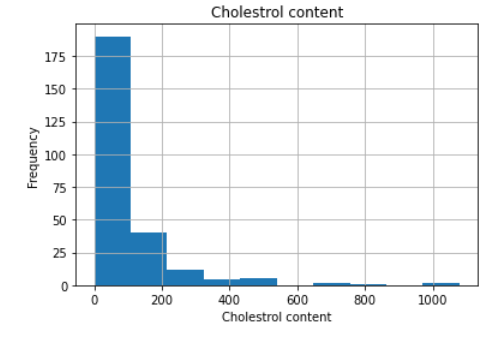
Of these the predicted column is ‘Name’.

In this project we use key terms found directly in the csv (namely 'RecipeCategory', 'Keywords', and 'RecipeIngredientParts') in conjunction with user input in a cosine similarity. As the recipe output depends on the ingredients, we have taken this system as being supervised.

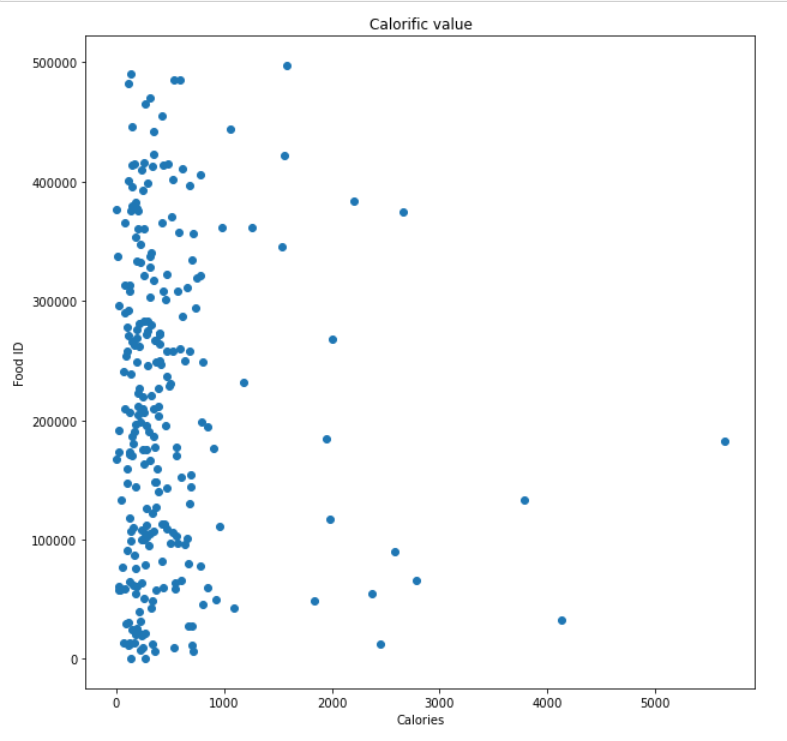
**Figures 1-4** convey information derived during feature exploration:



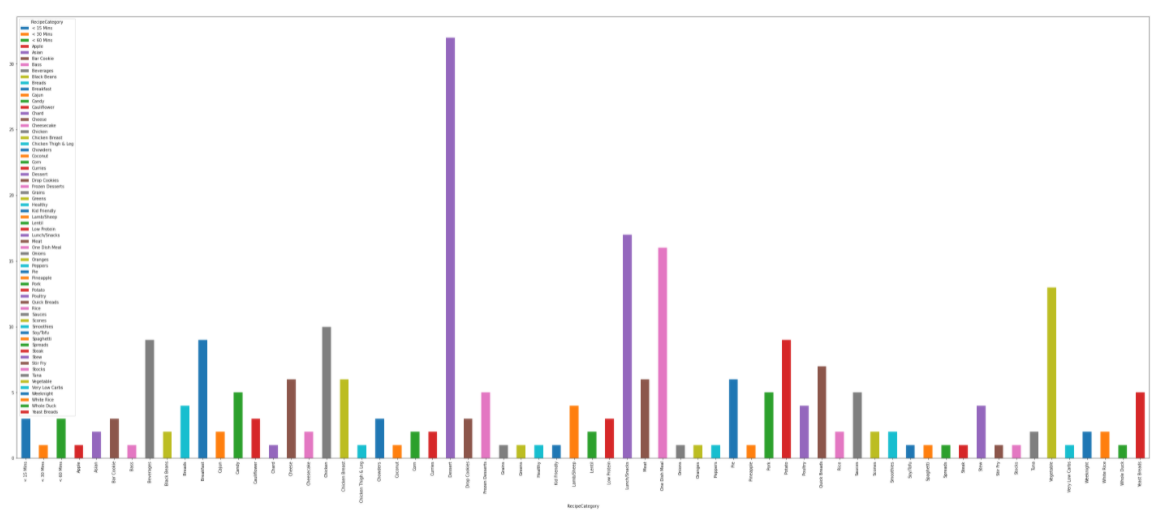
**Figure 1: Total Time.** Shows the frequency distribution of the total cooking time for a recipe.



**Figure 2: Cholesterol Content.** Shows the frequency of cholesterol (numerical value) in the dataset.



**Figure 3: Calories.** Shows the distribution of calorific value as examined linearly down the recipe list.



**Figure 4: Categories.** This shows the distribution of recipe count for each category. (Note: the highest frequency category is ‘Dessert’.)

**Figure 5** (below) shows a closer look at the category labels in **Figure 4**:

**Figure 5: Recipe categories.** Gives a full list of the categories within the dataset.

**Design and Milestones**

This project started with a few milestones, outlined below:

1. Data collection from Kaggle.

This was briefly mentioned in the prior section. The link for the Kaggle site is given within the GitHub page, included in the ***Repository/Archive*** section of this report.

1. Data pre-processing.

(Also outlined in the prior section.)

1. Building the Training model (Building model with Model based algorithm, Validation of models, Accuracies)
2. Developing the final algorithm.

Tools and frameworks used in the algorithm include:

numpy - used for arrays and numerical operations.

pandas - used for easy handling and manipulation of dataset. (The original dataset is pulled into a pandas dataframe.)

matplotlib - used to create graphs that convey patterns and distribution.

warnings - used to bypass errors for temporary usage.

sentence\_transformers (BERT) – used to embed key terms before preforming cosine\_similarity.

sklearn - cosine\_similarity, from sklearn, is used to find similarity between the user input and the dataset’s selected key terms.

More on BERT:

BERT (Bidirectional Encoder Representations from Transformers) is a pretrained ML framework for Natural Language Processing. It works to embed both images and text, in our case: text.

No training was required for this project, and the outputs were not measured against a standard in the analysis. Success was determined by how much the recipe matched with the given ingredient list.

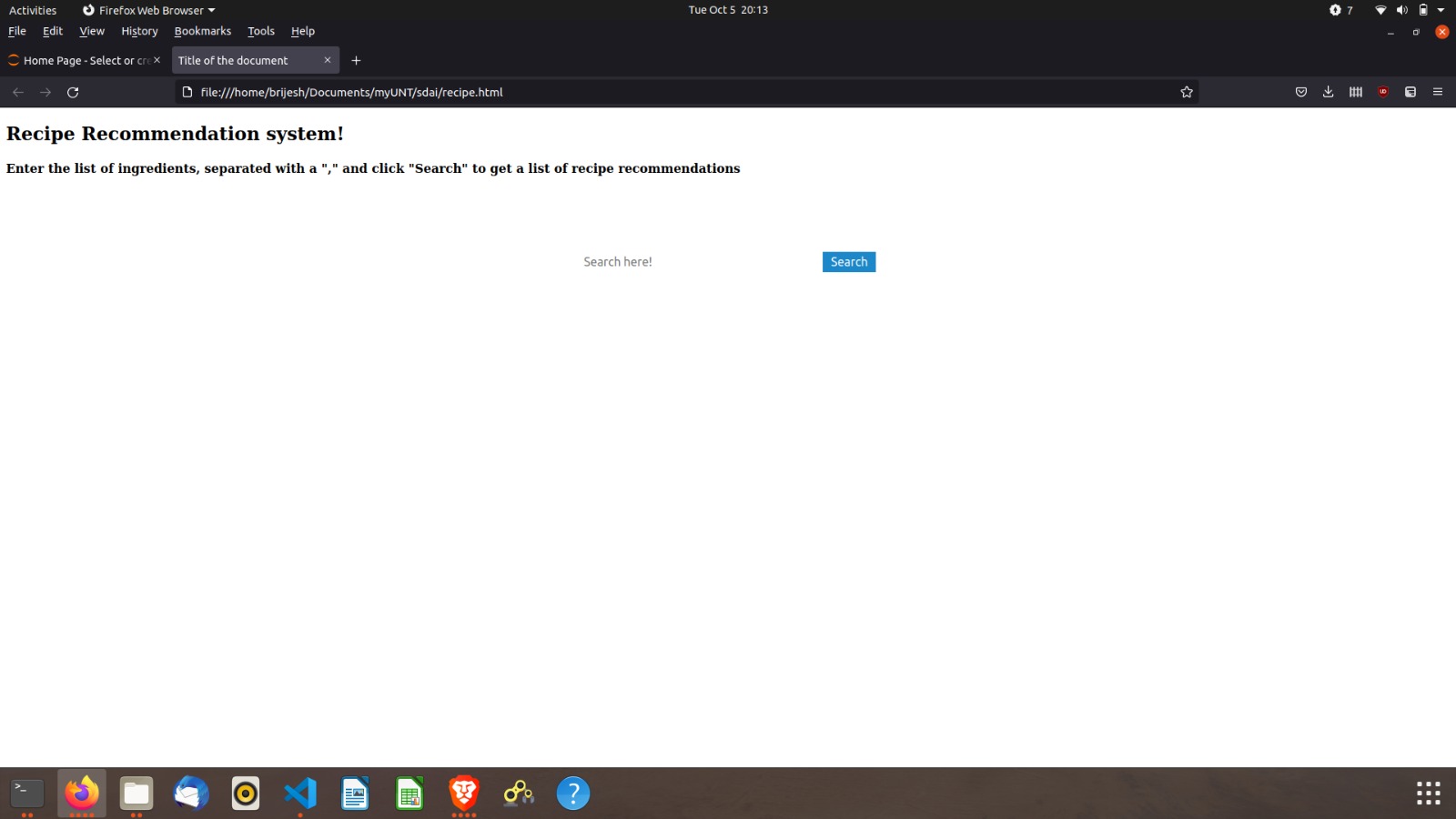
1. Create a web page.
2. Deploying deployment tools.

(5 and 6) System Design: The users interact with the recommendation system using a web application, the Apache webserver, will be used to serve the webpage to the user. The web application uses html, CSS, and JavaScript for presenting information, accepting user input, and displaying the output. The user input will be routed to the recommendation model and the model output will be displayed to the user. The model is deployed using a docker image and interacts with the web-framework using a simple API.

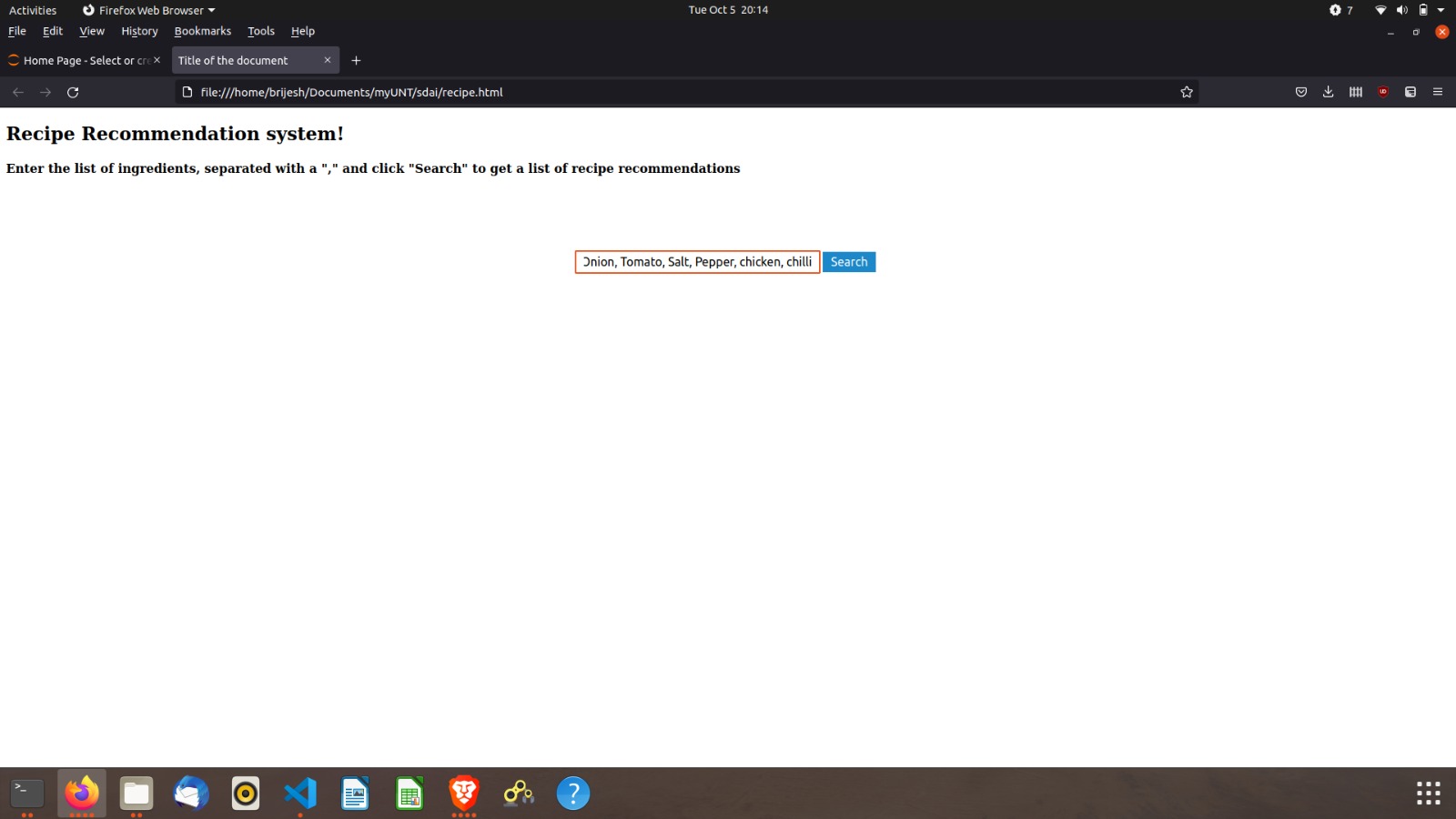
(Note that several of these milestones were done in parallel and integrated together thereafter in order to meet time constraints.)

**Results and Analysis**

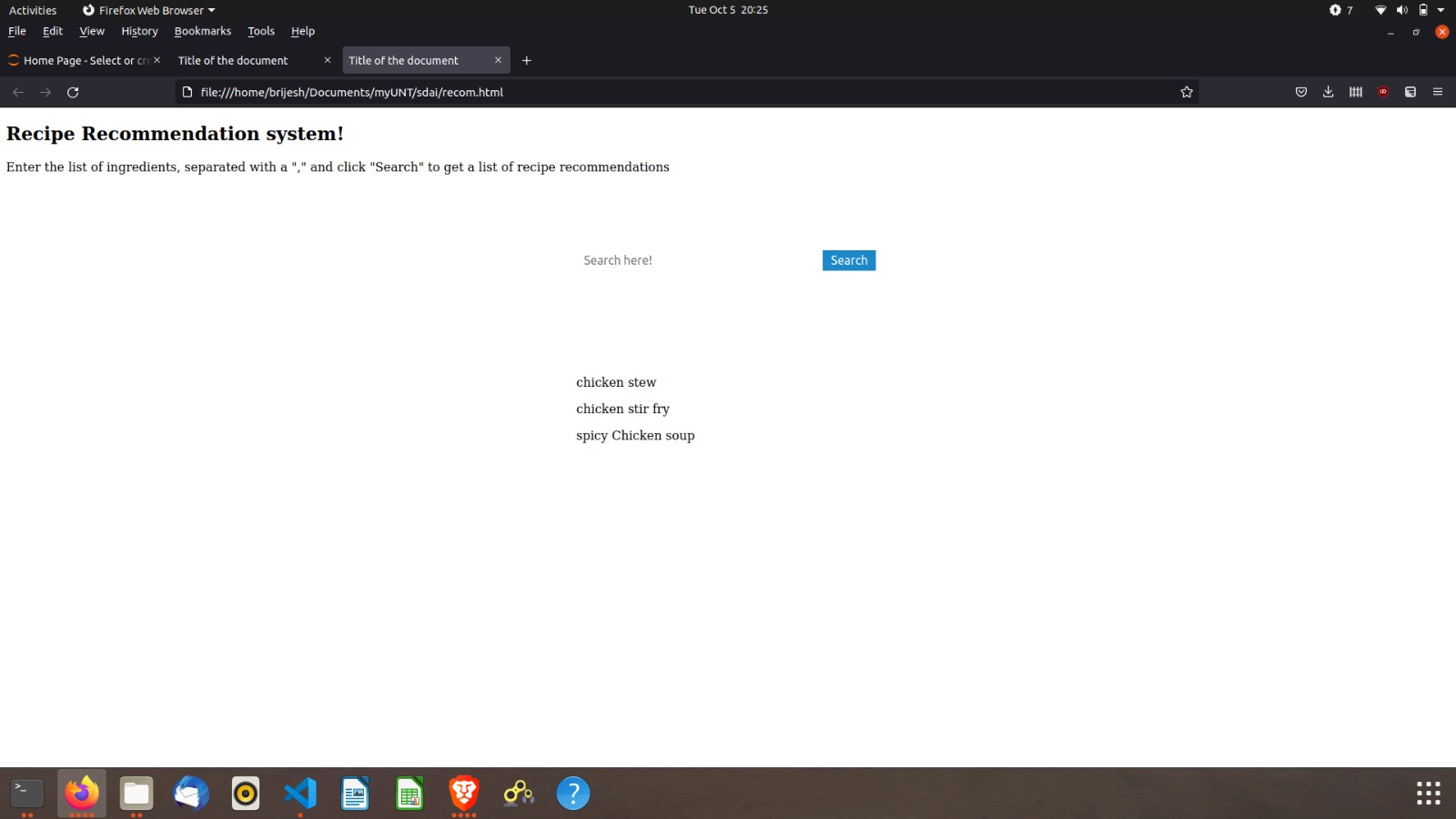
**Figures 6a-6b** (below) show images of the user interface (a cropped view):



**Figure 6a: Starting page.** This is what the user would first see when they open the webpage.



**Figure 6b: User enters ingredients.** This shows where and how the user would provide the ingredients input to the program.



**Figure 6c: Recommendation output.** Page gives the user an output of the recipe names that are recommended based on the ingredients input.

Note that as of now, the page only acts as a recommender, giving the recipe titles for the recommended matches rather than the full recipe from a selection. It takes less than one second from hitting search for the program to output the recipe recommendations to the user via the webpage. (Disclaimer: internet strength may affect this.) As the program runs on a similarity system, the recommendations are not 100% exact, but the most well-matched recipes are outputted. As it is difficult to run an analysis on if this criterion met, we have only looked at the general focus of the recipe to subjectively determine if the program is working well.

**Repository / Archive**

The GitHub page for this project is linked [here](https://github.com/NikhilReddy0608/Food-Recipe-Recommendation). (Alternative: https://tinyurl.com/RecipeRS.) It contains a link to the Kaggle page from which the original dataset was sourced, as well as a link to this report. The repository contains the code for the Recipe Recommendation System as well as other relevant files/links for this project.

**Appendix**

Code as a pdf (with outputs):



A copy of the code (in text) is here:

|  |
| --- |
| **import** numpy **as** np  **import** pandas **as** pd  **import** matplotlib.pyplot **as** plt  **import** warningswarnings**.**filterwarnings('ignore')  ip **=** pd**.**read\_csv('recipes.csv')  ip**=**ip**.**sample(500)  Ip**.**columns  ip**.**describe()  ip.drop([('RecipeId'),('AuthorId'),('AuthorName'),('DatePublished'),('Images'),('RecipeIngredientQuantities'),('CookTime'),('PrepTime'),('RecipeYield'),('ReviewCount'),('RecipeServings')],axis=1,inplace=True)  ip.head()  ip**.**RecipeCategory**.**nunique()  ip[ip**.**columns[ip**.**isnull()**.**any()]]**.**isnull()**.**sum()  ip**.**dropna(inplace**=True**)  ip**.**isnull()**.**any()  ip[ip**.**columns[ip**.**isnull()**.**any()]]**.**isnull()**.**sum()  Ip**.**columns  plt**.**figure(figsize**=**(50,10))  plt**.**hist(ip['TotalTime'],bins**=**50)  plt**.**xlabel('Cooktime')  plt**.**ylabel('Frequency0')  plt**.**title('Frequency of cook time')  plt**.**show()  ip['CholesterolContent']**.**hist()  plt**.**title('Cholestrol content')  plt**.**xlabel('Cholestrol content')  plt**.**ylabel('Frequency')  plt**.**show()  plt**.**figure(figsize**=**(10,10))  plt**.**scatter(ip['Calories'],ip**.**index)  plt**.**xlabel('Calories')  plt**.**ylabel('Food ID')  plt**.**title('Calorific value')  plt**.**show()  x **=** pd**.**crosstab(index**=**ip['RecipeCategory'],columns**=**ip['RecipeCategory'])  x**.**plot(kind**=**"bar", figsize**=**(50,20),stacked**=True**)  ip['list\_of\_keywords']**=**(ip['Keywords']**+**ip['RecipeIngredientParts']**+**ip['RecipeCategory'])  ip['list\_of\_keywords']**.**iloc[0]  ip['list\_of\_keywords']**=**ip['list\_of\_keywords']**.**str**.**replace("c\(",'')ip['list\_of\_keywords']**=**ip['list\_of\_keywords']**.**str**.**replace("\)",'')ip['list\_of\_keywords']**=**ip['list\_of\_keywords']**.**str**.**replace('"',' ')  ip['list\_of\_keywords']**.**iloc[0]  ip['list\_of\_keywords']**=**ip['list\_of\_keywords']**.**str**.**split(',')  ##---preprocessing ends here---##  **from** sentence\_transformers **import** SentenceTransformer  model**=**SentenceTransformer('all-mpnet-base-v2')  model  ip['list\_of\_keywords']**.**iloc[0]  x **=** model**.**encode(ip['list\_of\_keywords']**.**tolist())  len(x),ip**.**shape  ip['embeddings']**=**list(x)  ip**.**embeddings**.**iloc[0]**.**reshape(1,**-**1)**.**shape  input\_list\_of\_keywords**=**['chicken','Rice','paparika','pepper','cereals'] #this is where the user input is taken  input\_embedding**=**model**.**encode(''**.**join(input\_list\_of\_keywords))  input\_embedding**.**reshape(1,**-**1)**.**shape  **from** sklearn.metrics.pairwise **import** cosine\_similarity  ip['similarity']=ip.apply(lambda x: cosine\_similarity(x['embeddings'].reshape(1,-1),input\_embedding.reshape(1,-1))[0][0],axis=1)  max\_index**=**ip**.**similarity**.**idxmax()  ip**.**loc[max\_index]['list\_of\_keywords']  ip**.**loc[max\_index] #this is where the output is given  **!**pip install sentence\_transformers |