1.INTRODUCTION

1.1 Overview

This project is a restaurant recommendation system (RRS). A Recommendation System is an information filtering system that seeks to predict the rating a user would give for the item (in this case a restaurant). RRS is an on-line system to search restaurants. Visitors can browse all restaurants in L.A, and. Get information about restaurant name, type, rating, price. The functions include searching restaurants, viewing recommendations. Recommendation systems are important for increasing business revenue and giving users the ability to find desired restaurants of their taste. The system is challenging because many users don't give ratings and we have new restaurants and users added to the system every day. In order to improve restaurant rating system, we need to predict the rating for the restaurant which are not rated. So it is important to build recommendation system for sparse rated restaurants. For this recommendation model, all that users have to input is a restaurant name that they have previously enjoyed visiting into the model and it will generate a list of the 10 most recommended restaurants based on the highest cosine similarity scores to that particular restaurant. For the content-based recommendation model**,** it works by recommending restaurants to users based on similar restaurant categories and dominant topic keywords, thus suggesting restaurants that align with a user's preferences.

1.2 Purpose

The purpose of this system is to let people get ideas about which restaurant will be great for them. This system can give people some suggestions; also you can get others' opinions from this site. Further more, you can find the best restaurants by viewing the ratings page, which gathers many members' experience and response. This system is designed for people to search the information you send, and response all those restaurants matched the customers' request. Except viewing other's opinions, you can give suggestions by rating restaurants to other people. This system is like a communication bulletin for people who love to eat. In this site, the customer need to search restaurants by their names. They will get a page describing the related names of the restaurants and their type and ratings.

2.LITERATURE SURVEY

2.1Existing problem and Existing approaches or methods

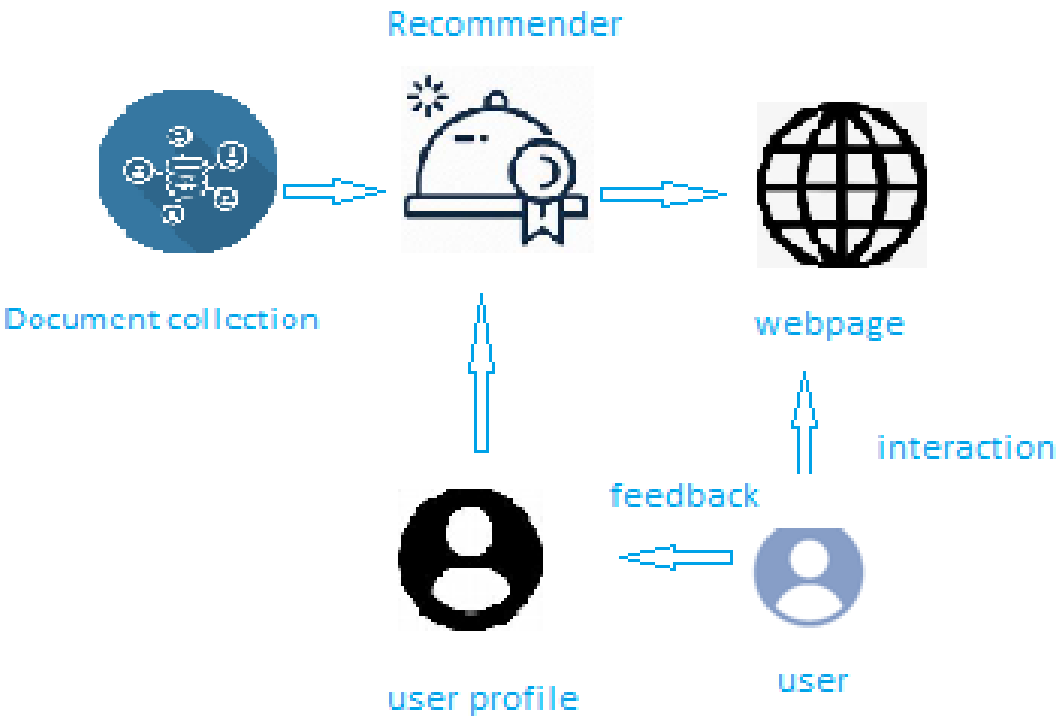
As we are users of recommendation applications, people care more about how we will like a restaurant. It is very common that we hang out with families, friends, and co-workers. when comes to lunch or dinner time. In the past, people obtained suggestions for restaurants from friends. Although this method is straightforward and user-friendly, it has some severe limitations. First, the recommendations from friends or other common people are limited to those places they have visited before. Thus, the user is not able to gain information about places less visited by their friends. Besides that, there is a chance of users not liking the place recommended by their friends.

2.2 Proposed solution

Here we are creating a content-based recommendation system. The aim is to create a content-based recommender system in which when we will write a restaurant name, the Recommender system will look at the reviews of other restaurants, and the System will recommend us other restaurants with similar reviews and sort them from the highest-rated. The main people who are going to benefit from this recommendation system are the tourists, who are new to a city. Most of the tourists always love to visit famous restaurants in a particular city during their visit. Otherwise, it can be heavily used by people belonging to the same city, to see if any new restaurant is recommended based on their activity.

3.THEORITICAL ANALYSIS

3.1 Block Diagram



3.2 Hardware and Software

Software Requirements:

To complete this project, you must require the following software's, concepts, and packages

Anaconda navigator

Python packages:

- pandas
- matplotlib
- seaborn
- plotly
- numpy
- scikit-image
- scikit-learn
- Flask

Hardware Requirements

- Processor : Intel Core i3
- Hard Disk Space : Min 100 GB
- Ram : 4 GB
- Display : 14.1 "Color Monitor(LCD, CRT or LED)
- Clock Speed : 1.67 GHz

4.EXPERIMENTAL INVESTIGATION

For developing the project the team has completed several tasks:

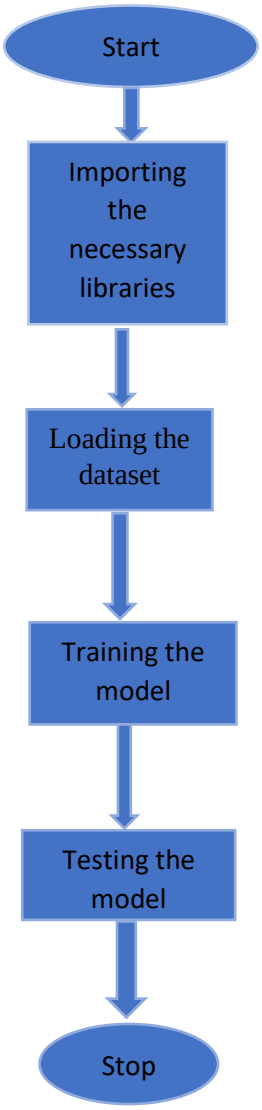
1. Data Collection.
    1. Collect the dataset or Create the dataset

1. Data Pre- processing.
    1. Import the Libraries.
    2. Importing the dataset.
    3. Exploratory Data Analysis
    4. Data Visualization.

3. Content Based Filtering
    1. Merging datasets
    2. Creating the recommender system
    3. Predicting the results

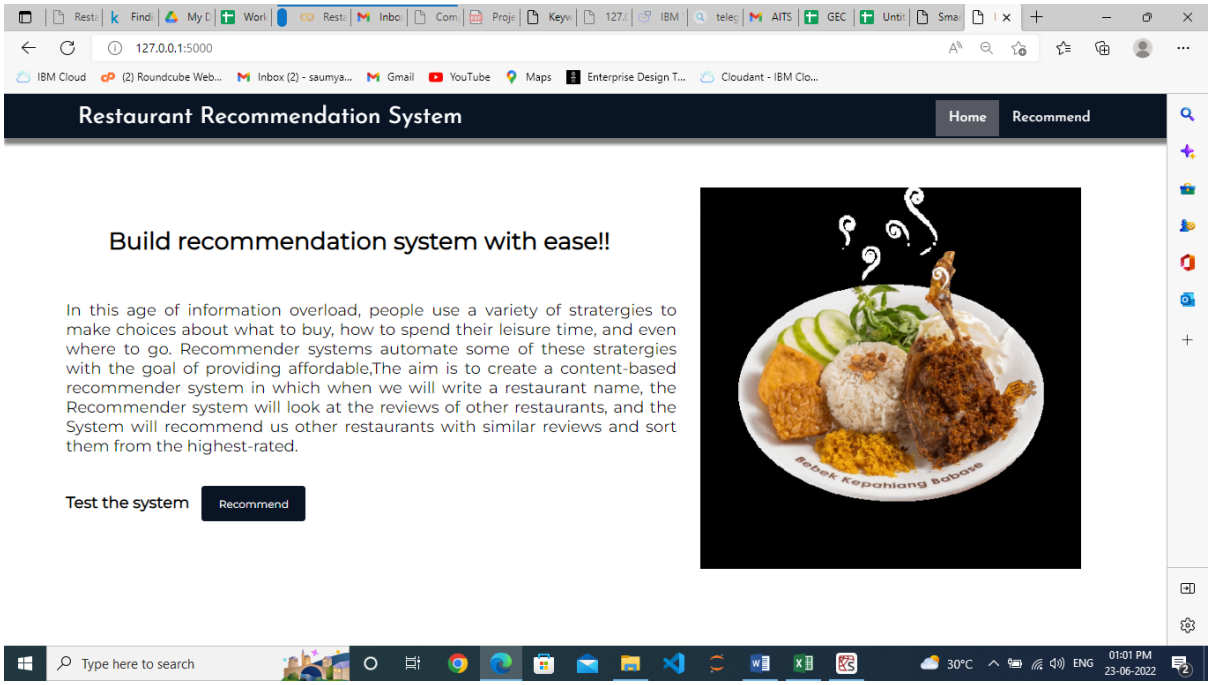
4. Application Building
    1. Create an HTML file
    2. Build a Python Code

5.FLOW CHAT

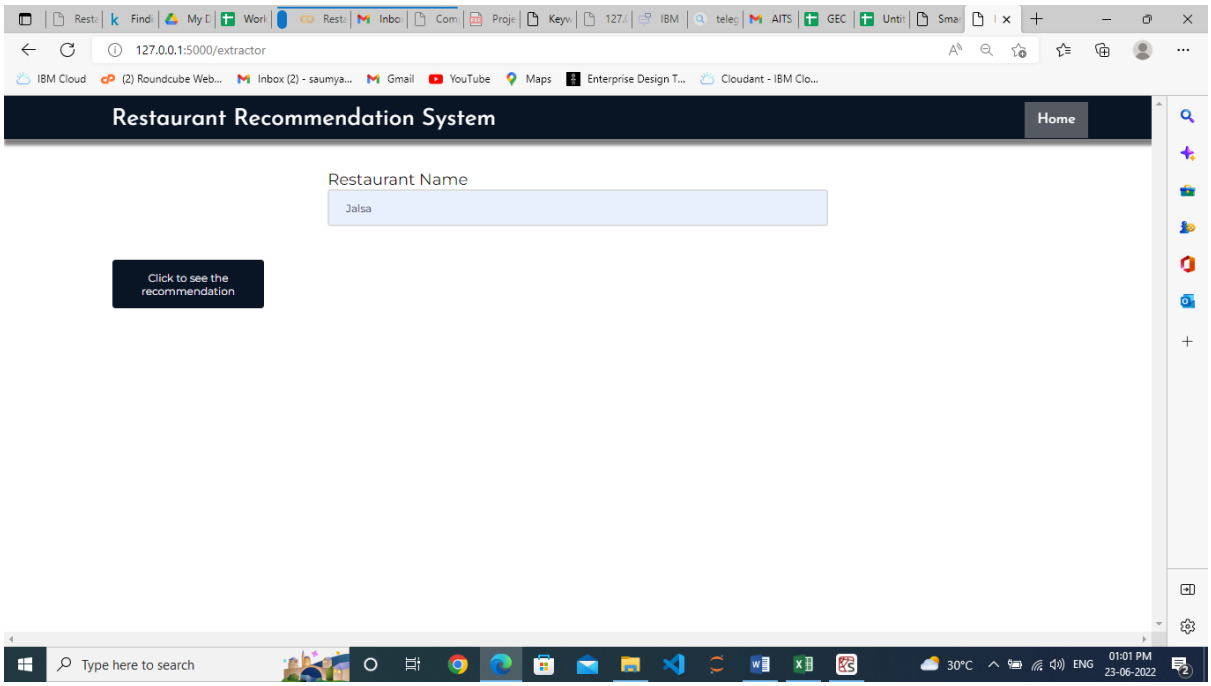

Start
Importing the necessary libraries
Loading the dataset
Training the model
Testing the model
Stop

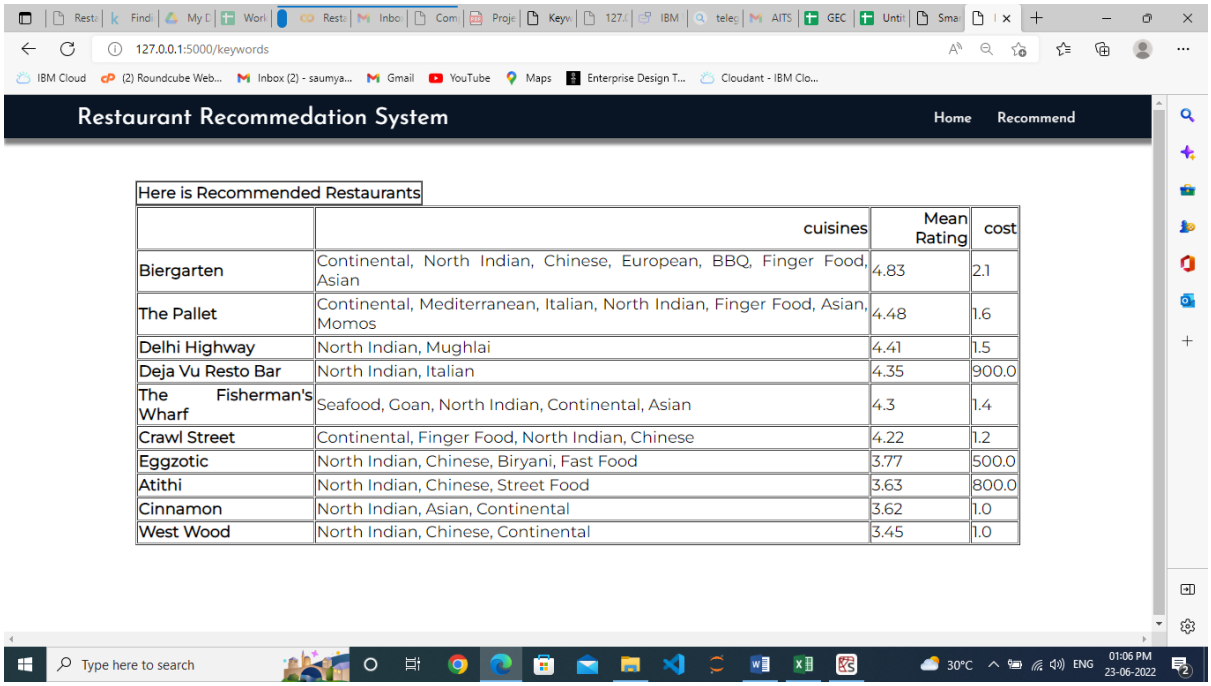
6. RESULTS

This is the home main page that describes the project and summarizes it.



Checking recommendation for the restaurant: ‘Jalsa’

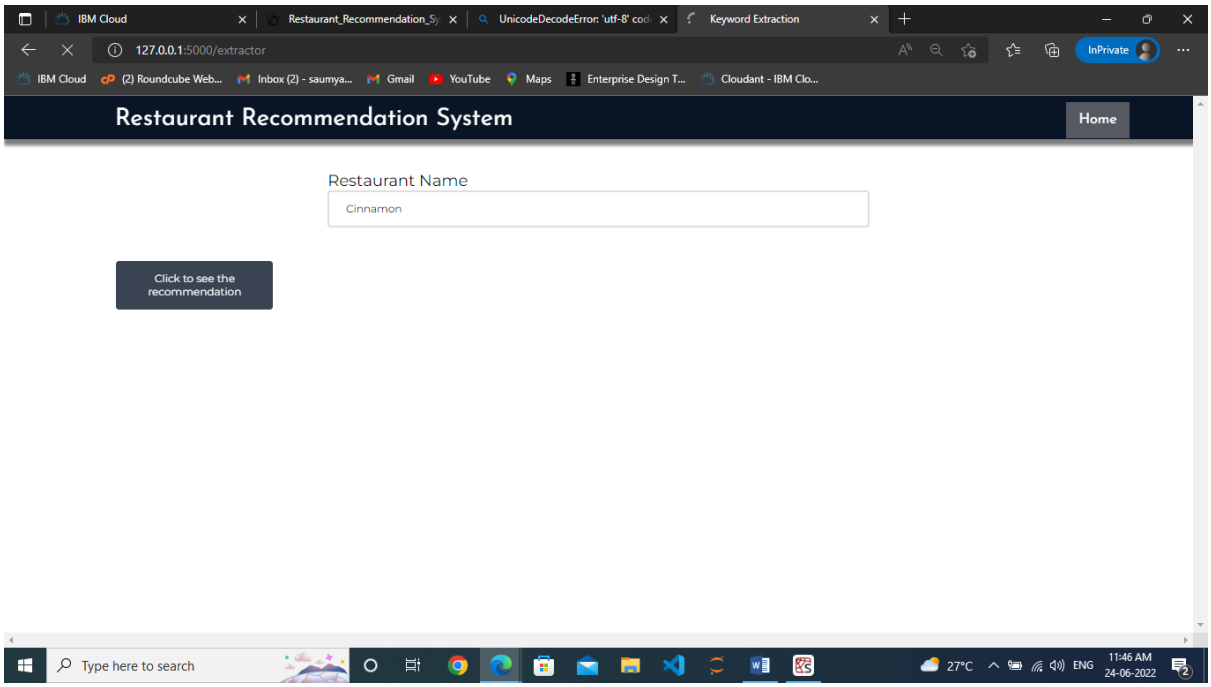


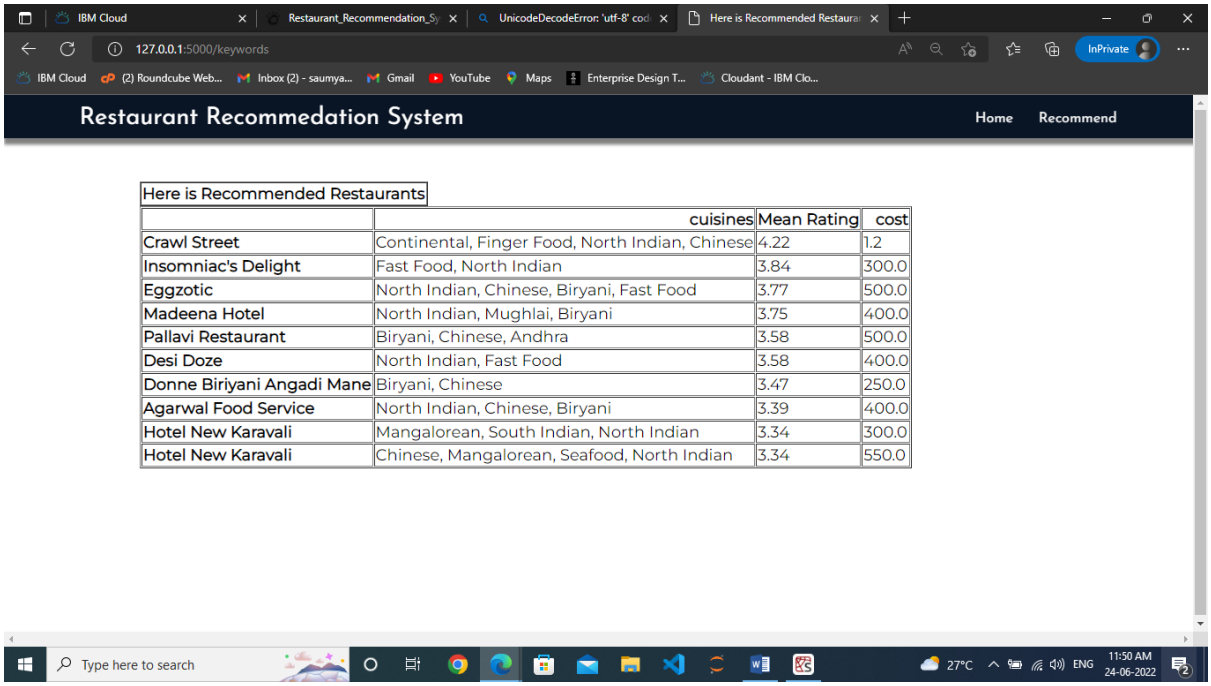


	cuisines	Mean Rating	cost
Biergarten	Continental, North Indian, Chinese, European, BBQ, Finger Food, Asian	4.83	2.1
The Pallet	Continental, Mediterranean, Italian, North Indian, Finger Food, Asian, Momos	4.48	1.6
Delhi Highway	North Indian, Mughlai	4.41	1.5
Deja Vu Resto Bar	North Indian, Italian	4.35	900.0
The Fisherman's Wharf	Seafood, Goan, North Indian, Continental, Asian	4.3	1.4
Crawl Street	Continental, Finger Food, North Indian, Chinese	4.22	1.2
Eggzotic	North Indian, Chinese, Biryani, Fast Food	3.77	500.0
Atithi	North Indian, Chinese, Street Food	3.63	800.0
Cinnamon	North Indian, Asian, Continental	3.62	1.0
West Wood	North Indian, Chinese, Continental	3.45	1.0

This is the prediction page where we will provide a restaurant name for which we will get the top recommended restaurants, which based on cuisines, mean rating (out of 5), cost in thousands.

Checking recommendation for the restaurant ‘Cinnamon’





Finally, the prediction for the given restaurant inputs is shown.

7.ADVANTAGES AND DISADVANTAGES

Advantages:

- You can try new foods.
- You do not have to cook.
- You get to spend time with family and friends.
- It's easier to feed large parties.
- Time is not wasted.

Disadvantages:

- Significant investments required.
- Too many choices.
- The complex onboarding process.
- Lack of data analytics capability.
- The 'cold start' problem.
- Inability to capture changes in user behavior.
- Privacy concerns.

9.CONCLUSION

The main objective of the study is to develop the restaurant recommendation system using machine learning with the web interface that can act as a application for the customers. This application is used for the users to predict the suitable restaurant and find out which dish is famous in region wise and in person. This application ensures the availability of ratings to the customers. The popularity based and collaborative based filtering makes the recommendation more efficient so that each user can use this application for their easy prediction of restaurant. Most the case user need the restaurant with their nearby location. We also solving that issue by adding the restaurant location in our dataset. So that our machine learning algorithm easily predicts the restaurant for the customer with their present location. This restaurant recommendation system web application will provide user a better experience in searching of restaurant with short amount of time and nearby location. This will decrease the user's effort and makes the time more precious.

10.FUTURE SCOPE

To build more friendly graphical interfaces, the next goal for a further project is to improve the performance of system and improve the member's benefit such as providing more functions for member, like online reservation function, online order menu, and website, etc.

11. BIBILOGRAPHY

https://medium.com/mlearning-ai/restaurant-recommendation-system-based-on-the-content-in-reviews-dfc3351004db

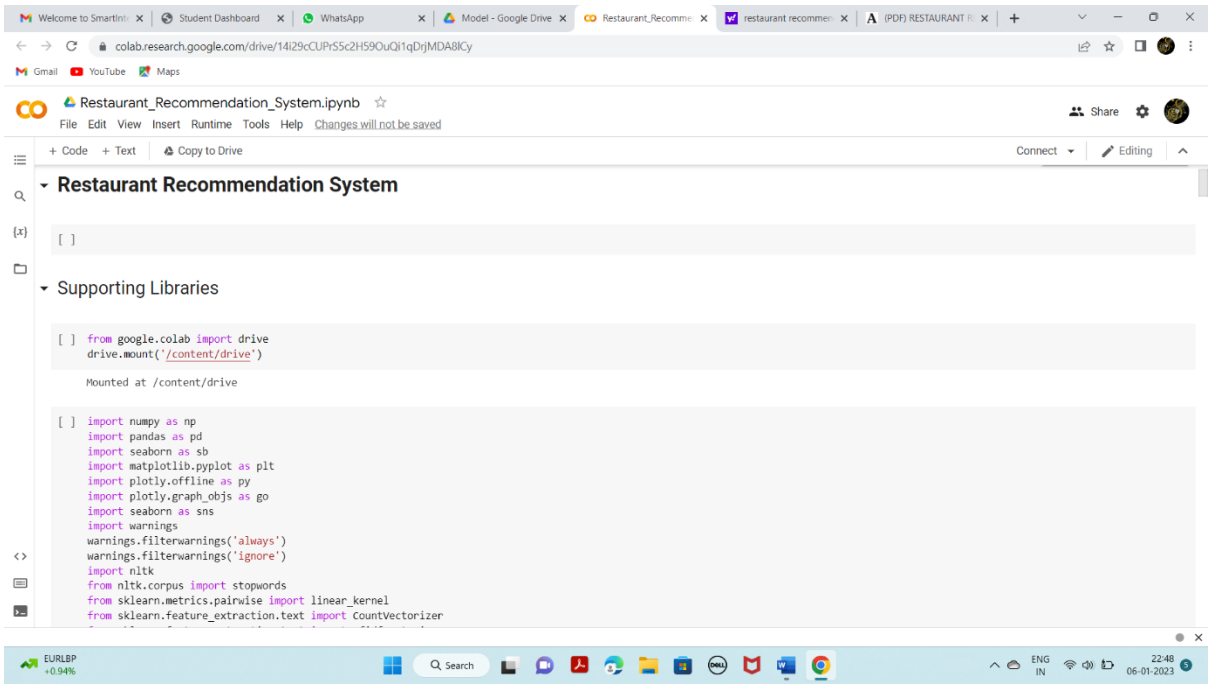
https://www.kaggle.com/code/midouazerty/restaurant-recommendation-system-using-ml

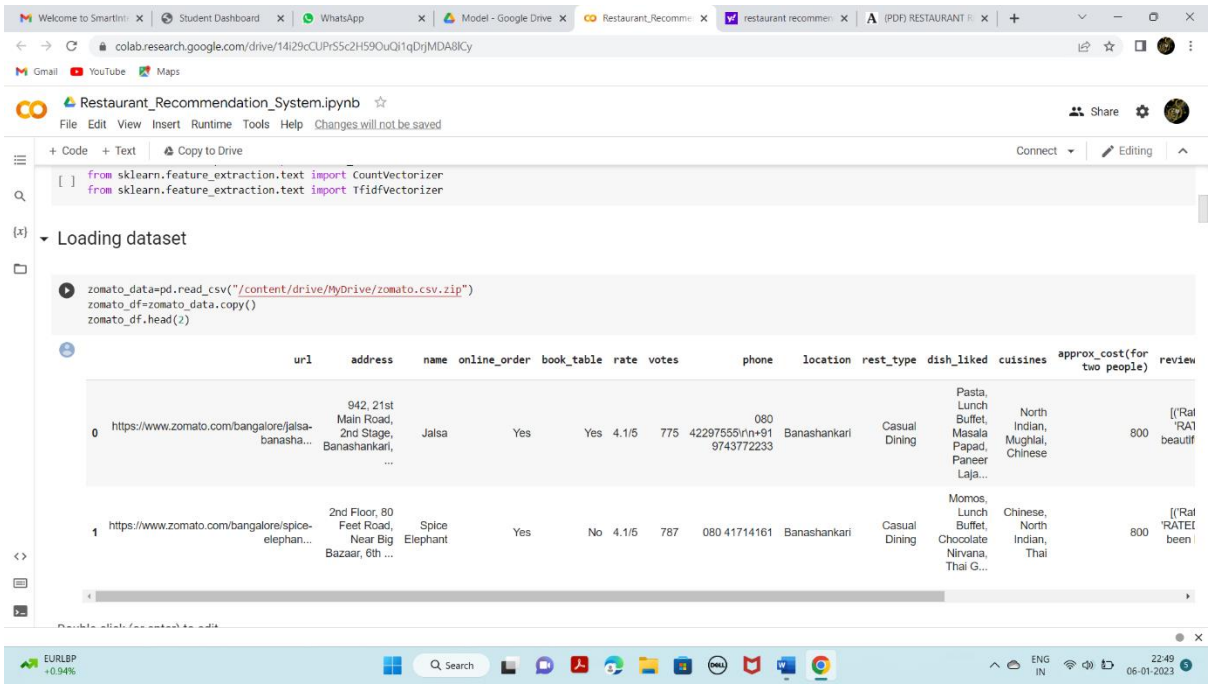
https://towardsdatascience.com/yelp-restaurant-recommendation-system-capstone-project-264fe7a7dea1

https://www.academia.edu/85069823/RESTAURANT_RECOMMENDATION_SYSTEM

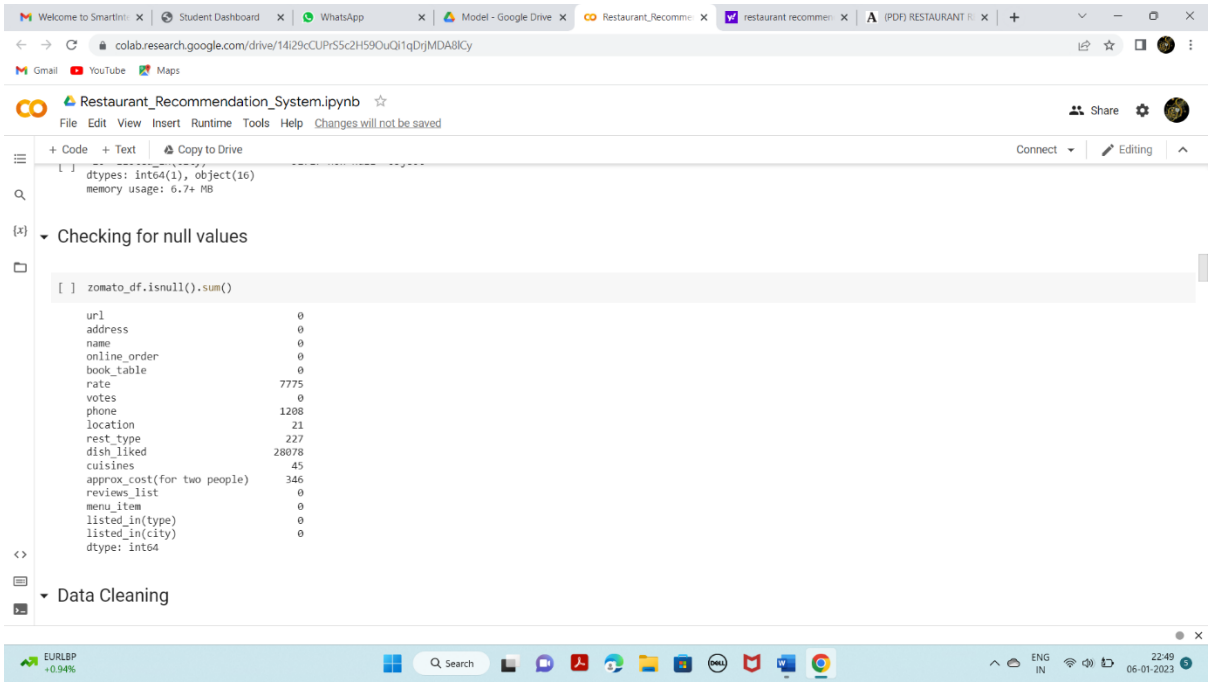
APPENDIX

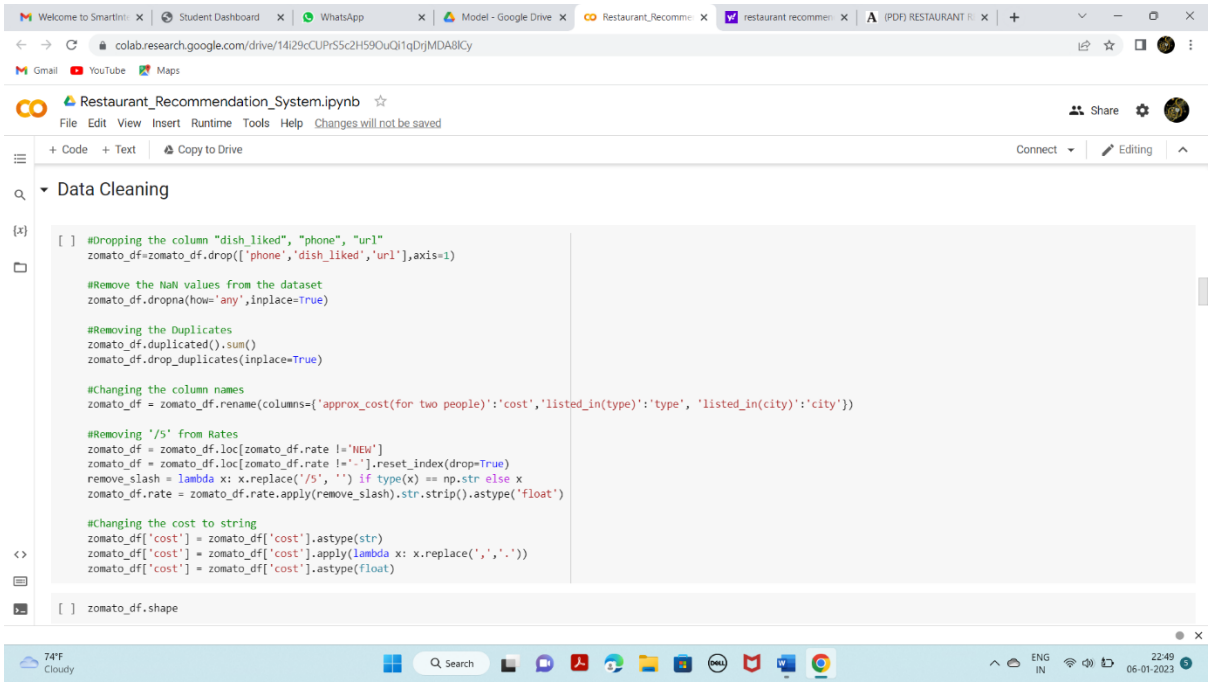
Source Code

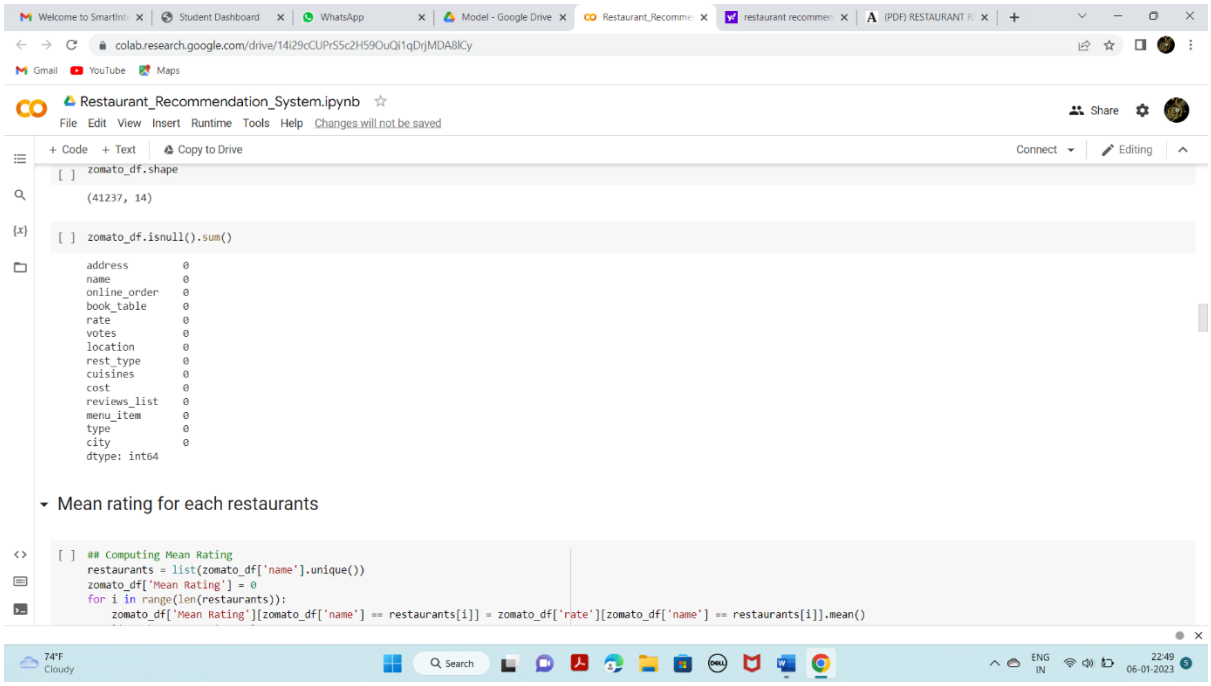




	url	address	name	online_order	book_table	rate	votes	phone	location	rest_type	dish_liked	cuisines	approx_cost(for two people)	review
0	https://www.zomato.com/bangalore/jalsa-banasha...	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	Yes	Yes	4.1/5	775	080 42297555\r\n+91 9743772233	Banashankari	Casual Dining	Pasta, Lunch Buffet, Masala Papad, Paneer Laja...	North Indian, Mughlai, Chinese	800	[('Rat 'RAT beautif
1	https://www.zomato.com/bangalore/spice-elephan...	2nd Floor, 80 Feet Road, Near Big Bazaar, 6th ...	Spice Elephant	Yes	No	4.1/5	787	080 41714161	Banashankari	Casual Dining	Momos, Lunch Buffet, Chocolate Nirvana, Thai G...	Chinese, North Indian, Thai	800	[('Rat 'RATE been


Restaurant_Recommendation_System.ipynb
File Edit View Insert Runtime Tools Help Changes will not be saved
+ Code + Text Copy to Drive
Connect Editing
dtypes: int64(1), object(16)
memory usage: 6.7+ MB
Checking for null values
[ ] zomato_df.isnull().sum()
url 0
address 0
name 0
online_order 0
book_table 0
rate 7775
votes 0
phone 1208
location 21
rest_type 227
dish_liked 28078
cuisines 45
approx_cost(for two people) 346
reviews_list 0
menu_item 0
listed_in(type) 0
listed_in(city) 0
dtype: int64
Data Cleaning


Restaurant_Recommendation_System.ipynb
File Edit View Insert Runtime Tools Help Changes will not be saved
+ Code + Text Copy to Drive
Connect Editing
Data Cleaning
[ ] #Dropping the column "dish_liked", "phone", "url"
zomato_df=zomato_df.drop(['phone','dish_liked','url'],axis=1)
#Remove the NaN values from the dataset
zomato_df.dropna(how='any',inplace=True)
#Removing the Duplicates
zomato_df.duplicated().sum()
zomato_df.drop_duplicates(inplace=True)
#Changing the column names
zomato_df = zomato_df.rename(columns={'approx_cost(for two people)':'cost','listed_in(type)':'type', 'listed_in(city)':'city'})
#Removing '/5' from Rates
zomato_df = zomato_df.loc[zomato_df.rate !='NEW']
zomato_df = zomato_df.loc[zomato_df.rate !='-'].reset_index(drop=True)
remove_slash = lambda x: x.replace('/5', '') if type(x) == np.str else x
zomato_df.rate = zomato_df.rate.apply(remove_slash).str.strip().astype('float')
#Changing the cost to string
zomato_df['cost'] = zomato_df['cost'].astype(str)
zomato_df['cost'] = zomato_df['cost'].apply(lambda x: x.replace(',','.'))
zomato_df['cost'] = zomato_df['cost'].astype(float)
[ ] zomato_df.shape



Restaurant_Recommendation_System.ipynb

File Edit View Insert Runtime Tools Help Changes will not be saved

+ Code + Text Copy to Drive

```
[ ] zomato_df.shape
```

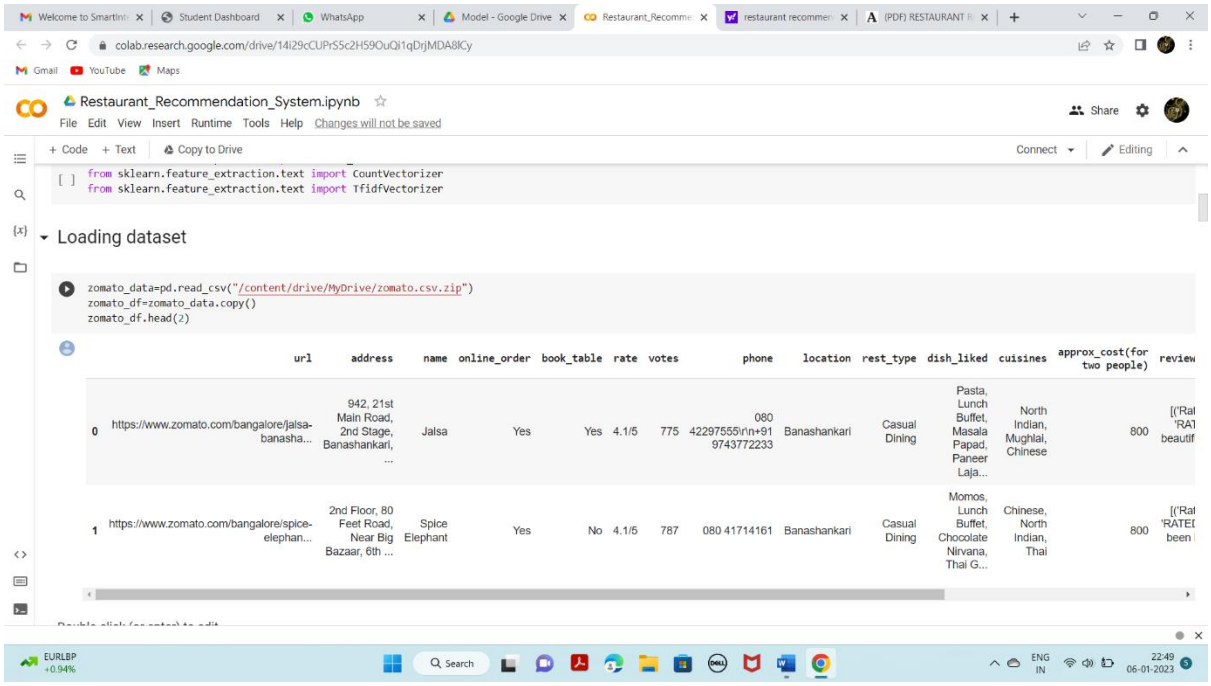
```
(41237, 14)
```

```
[ ] zomato_df.isnull().sum()
```

```
address         0
name            0
online_order    0
book_table      0
rate            0
votes           0
location        0
rest_type       0
cuisines        0
cost            0
reviews_list    0
menu_item       0
type            0
city            0
dtype: int64
```

Mean rating for each restaurants

```
[ ] ## Computing Mean Rating
    restaurants = list(zomato_df['name'].unique())
    zomato_df['Mean Rating'] = 0
    for i in range(len(restaurants)):
        zomato_df['Mean Rating'][zomato_df['name'] == restaurants[i]] = zomato_df['rate'][zomato_df['name'] == restaurants[i]].mean()
```



Restaurant_Recommendation_System.ipynb

File Edit View Insert Runtime Tools Help Changes will not be saved

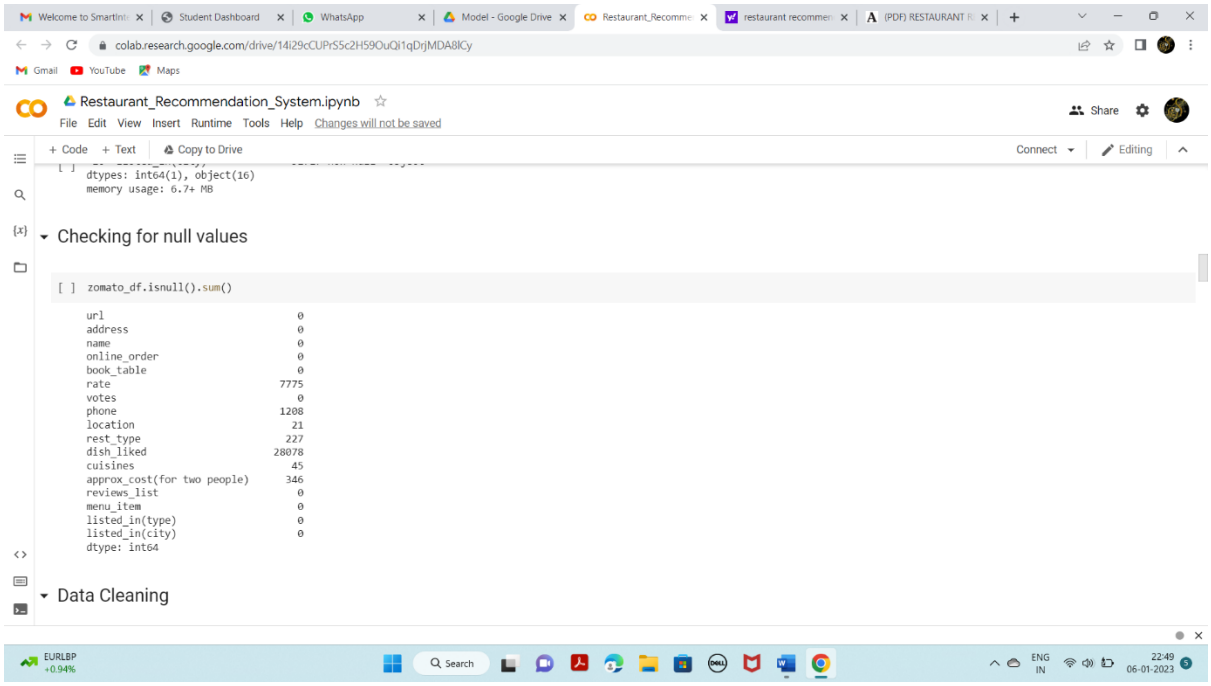
+ Code + Text Copy to Drive

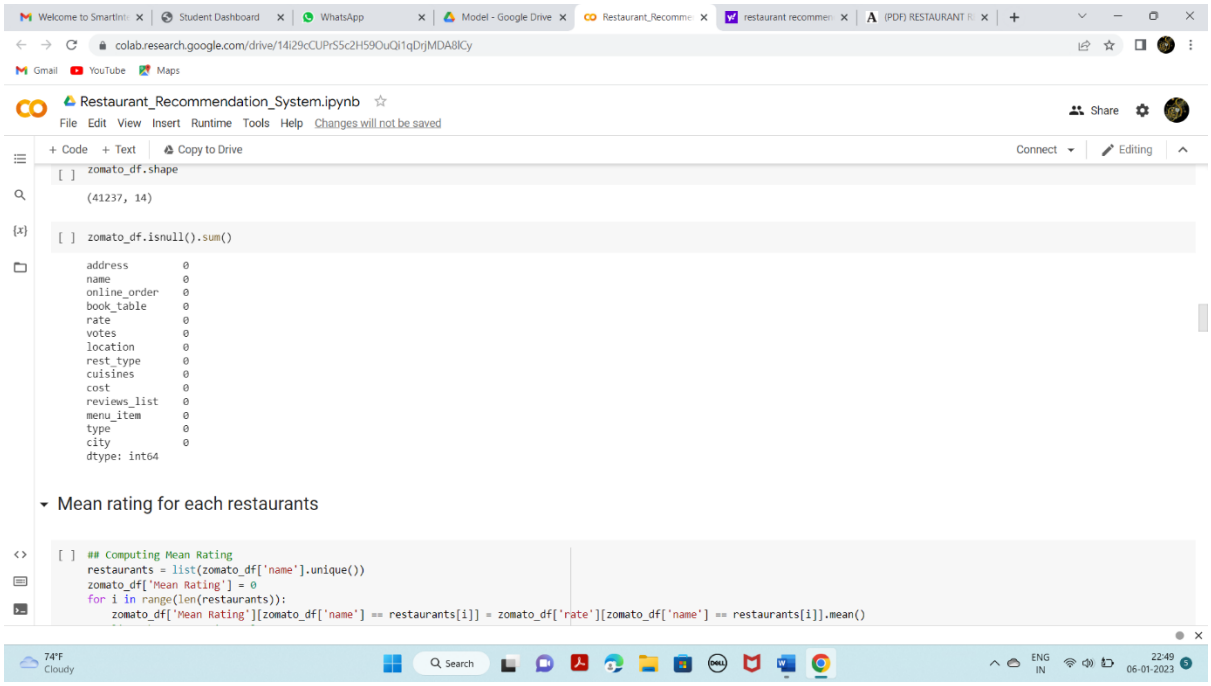
```
[ ] from sklearn.feature_extraction.text import CountVectorizer
    from sklearn.feature_extraction.text import TfidfVectorizer
```

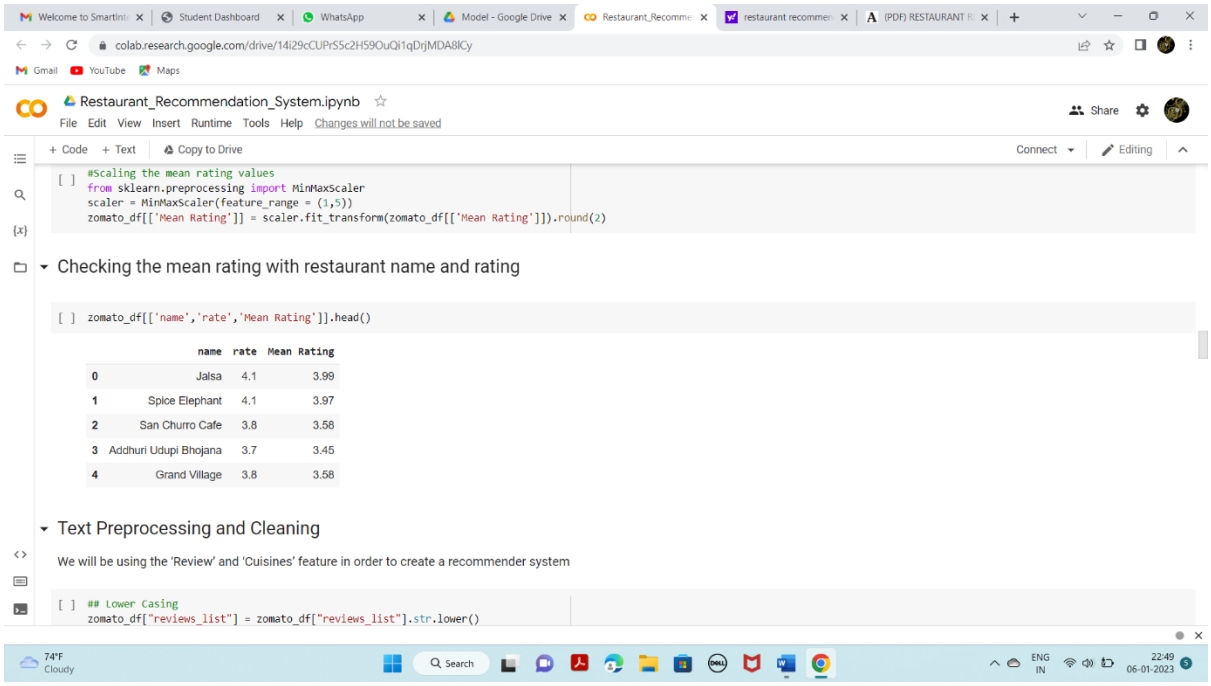
Loading dataset

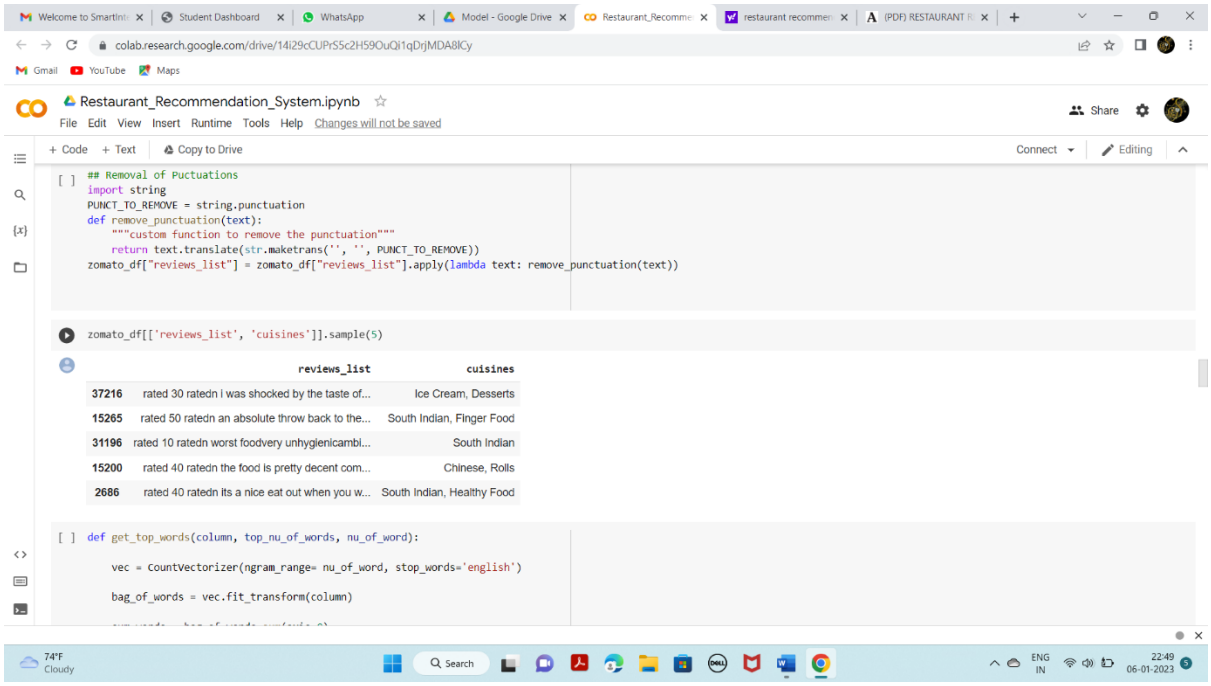
```
zomato_data=pd.read_csv("/content/drive/MyDrive/zomato.csv.zip")
zomato_df=zomato_data.copy()
zomato_df.head(2)
```

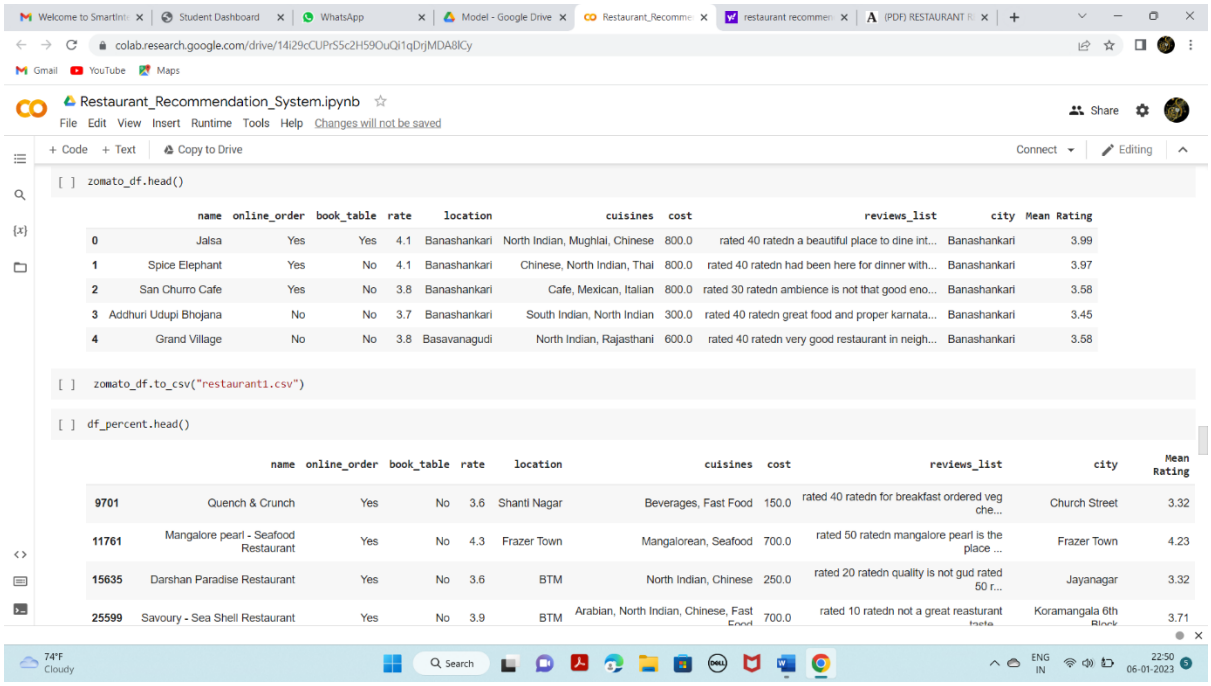
	url	address	name	online_order	book_table	rate	votes	phone	location	rest_type	dish_liked	cuisines	approx_cost(for two people)	review
0	https://www.zomato.com/bangalore/jalsa-banasha...	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	Yes	Yes	4.1/5	775	42297555\r\n+91 9743772233	Banashankari	Casual Dining	Pasta, Lunch Buffet, Masala Papad, Paneer Laja...	North Indian, Mughlai, Chinese	800	[('Rat 'RAT beautif
1	https://www.zomato.com/bangalore/spice-elephan...	2nd Floor, 80 Feet Road, Near Big Bazaar, 6th ...	Spice Elephant	Yes	No	4.1/5	787	080 41714161	Banashankari	Casual Dining	Momos, Lunch Buffet, Chocolate Nirvana, Thai G...	Chinese, North Indian, Thai	800	[('Rat 'RATEI been

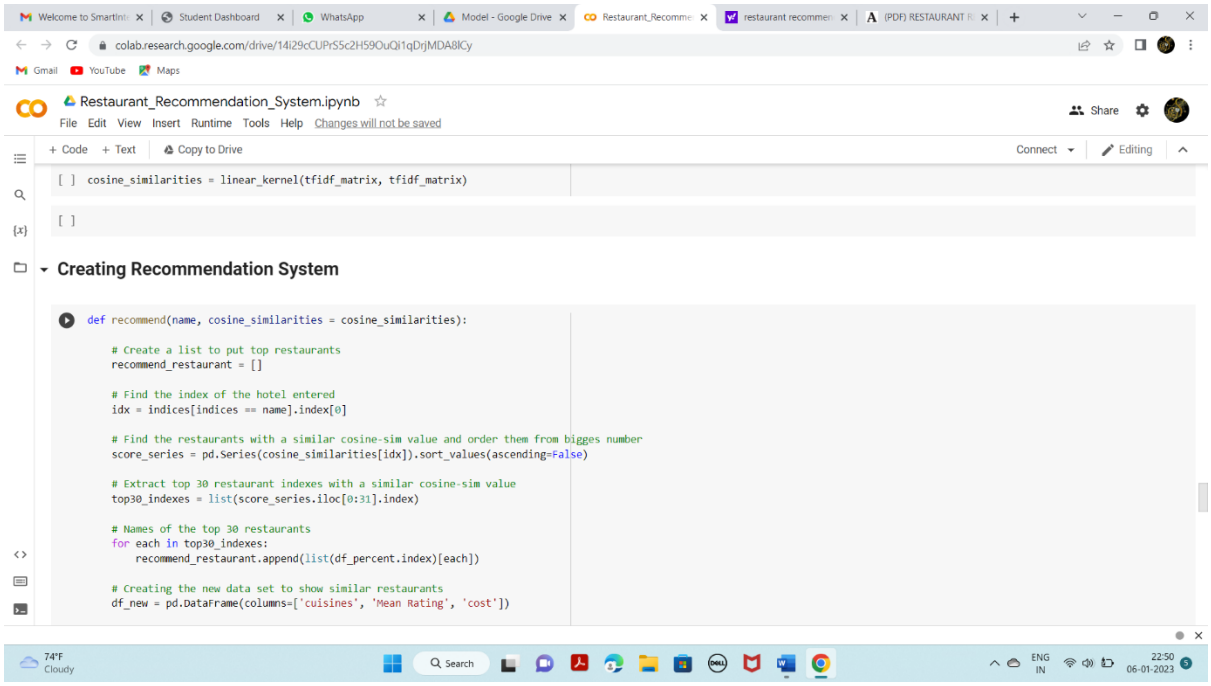

Restaurant_Recommendation_System.ipynb
File Edit View Insert Runtime Tools Help Changes will not be saved
+ Code + Text Copy to Drive
Connect Editing
dtypes: int64(1), object(16)
memory usage: 6.7+ MB
Checking for null values
[ ] zomato_df.isnull().sum()
url 0
address 0
name 0
online_order 0
book_table 0
rate 7775
votes 0
phone 1208
location 21
rest_type 227
dish_liked 28078
cuisines 45
approx_cost(for two people) 346
reviews_list 0
menu_item 0
listed_in(type) 0
listed_in(city) 0
dtype: int64
Data Cleaning

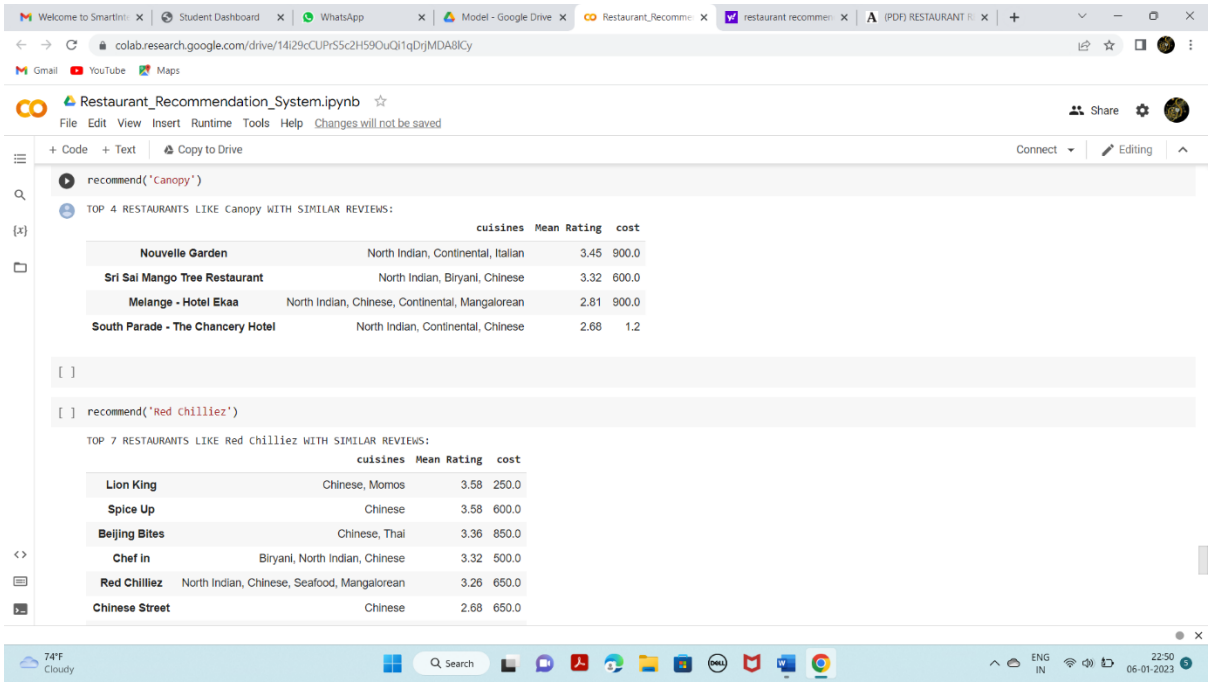

Restaurant_Recommendation_System.ipynb
File Edit View Insert Runtime Tools Help Changes will not be saved
+ Code + Text Copy to Drive
Connect Editing
[ ] zomato_df.shape
(41237, 14)
[ ] zomato_df.isnull().sum()
address 0
name 0
online_order 0
book_table 0
rate 0
votes 0
location 0
rest_type 0
cuisines 0
cost 0
reviews_list 0
menu_item 0
type 0
city 0
dtype: int64
Mean rating for each restaurants
[ ] ## Computing Mean Rating
restaurants = list(zomato_df['name'].unique())
zomato_df['Mean Rating'] = 0
for i in range(len(restaurants)):
zomato_df['Mean Rating'][zomato_df['name'] == restaurants[i]] = zomato_df['rate'][zomato_df['name'] == restaurants[i]].mean()


Restaurant_Recommendation_System.ipynb
File Edit View Insert Runtime Tools Help Changes will not be saved
+ Code + Text Copy to Drive
Connect Editing
#Scaling the mean rating values
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range = (1,5))
zomato_df[['Mean Rating']] = scaler.fit_transform(zomato_df[['Mean Rating']]).round(2)
Checking the mean rating with restaurant name and rating
zomato_df[['name','rate','Mean Rating']].head()
name rate Mean Rating
0 Jalsa 4.1 3.99
1 Spice Elephant 4.1 3.97
2 San Churro Cafe 3.8 3.58
3 Addhuri Udupi Bhojana 3.7 3.45
4 Grand Village 3.8 3.58
Text Preprocessing and Cleaning
We will be using the 'Review' and 'Cuisines' feature in order to create a recommender system
Lower Casing
zomato_df["reviews_list"] = zomato_df["reviews_list"].str.lower()


Restaurant_Recommendation_System.ipynb
File Edit View Insert Runtime Tools Help Changes will not be saved
+ Code + Text Copy to Drive
Connect Editing
Removal of Puctuations
import string
PUNCT_TO_REMOVE = string.punctuation
def remove_punctuation(text):
"""custom function to remove the punctuation"""
return text.translate(str.maketrans('', '', PUNCT_TO_REMOVE))
zomato_df["reviews_list"] = zomato_df["reviews_list"].apply(lambda text: remove_punctuation(text))
zomato_df[['reviews_list', 'cuisines']].sample(5)
reviews_list cuisines
37216 rated 30 ratedn I was shocked by the taste of... Ice Cream, Desserts
15265 rated 50 ratedn an absolute throw back to the... South Indian, Finger Food
31196 rated 10 ratedn worst foodvery unhyglenicambi... South Indian
15200 rated 40 ratedn the food is pretty decent com... Chinese, Rolls
2686 rated 40 ratedn its a nice eat out when you w... South Indian, Healthy Food
def get_top_words(column, top_nu_of_words, nu_of_word):
vec = CountVectorizer(ngram_range= nu_of_word, stop_words='english')
bag_of_words = vec.fit_transform(column)


Restaurant_Recommendation_System.ipynb
File Edit View Insert Runtime Tools Help Changes will not be saved
+ Code + Text Copy to Drive
Connect Editing
[ ] zomato_df.head()
name online_order book_table rate location cuisines cost reviews_list city Mean Rating
0 Jalsa Yes Yes 4.1 Banashankari North Indian, Mughlai, Chinese 800.0 rated 40 ratedn a beautiful place to dine int... Banashankari 3.99
1 Spice Elephant Yes No 4.1 Banashankari Chinese, North Indian, Thai 800.0 rated 40 ratedn had been here for dinner with... Banashankari 3.97
2 San Churro Cafe Yes No 3.8 Banashankari Cafe, Mexican, Italian 800.0 rated 30 ratedn ambience is not that good eno... Banashankari 3.58
3 Addhuri Udupi Bhojana No No 3.7 Banashankari South Indian, North Indian 300.0 rated 40 ratedn great food and proper karnata... Banashankari 3.45
4 Grand Village No No 3.8 Basavanagudi North Indian, Rajasthani 600.0 rated 40 ratedn very good restaurant in neigh... Banashankari 3.58
[ ] zomato_df.to_csv("restaurant1.csv")
[ ] df_percent.head()
name online_order book_table rate location cuisines cost reviews_list city Mean Rating
9701 Quench & Crunch Yes No 3.6 Shanti Nagar Beverages, Fast Food 150.0 rated 40 ratedn for breakfast ordered veg che... Church Street 3.32
11761 Mangalore pearl - Seafood Restaurant Yes No 4.3 Frazer Town Mangalorean, Seafood 700.0 rated 50 ratedn mangalore pearl is the place ... Frazer Town 4.23
15635 Darshan Paradise Restaurant Yes No 3.6 BTM North Indian, Chinese 250.0 rated 20 ratedn quality is not gud rated 50 r... Jayanagar 3.32
25599 Savoury - Sea Shell Restaurant Yes No 3.9 BTM Arabian, North Indian, Chinese, Fast 700.0 rated 10 ratedn not a great reasturant Koramangala 6th


Restaurant_Recommendation_System.ipynb
File Edit View Insert Runtime Tools Help Changes will not be saved
+ Code + Text Copy to Drive
Connect Editing
[ ] cosine_similarities = linear_kernel(tfidf_matrix, tfidf_matrix)
[ ]
Creating Recommendation System
def recommend(name, cosine_similarities = cosine_similarities):
# Create a list to put top restaurants
recommend_restaurant = []
# Find the index of the hotel entered
idx = indices[indices == name].index[0]
# Find the restaurants with a similar cosine-sim value and order them from bigges number
score_series = pd.Series(cosine_similarities[idx]).sort_values(ascending=False)
# Extract top 30 restaurant indexes with a similar cosine-sim value
top30_indexes = list(score_series.iloc[0:31].index)
# Names of the top 30 restaurants
for each in top30_indexes:
recommend_restaurant.append(list(df_percent.index)[each])
# Creating the new data set to show similar restaurants
df_new = pd.DataFrame(columns=['cuisines', 'Mean Rating', 'cost'])



app.py

```
import numpy as np
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt
import plotly.offline as py
import plotly.graph_objs as go
import seaborn as sns
import warnings
warnings.filterwarnings('always')
warnings.filterwarnings('ignore')
import nltk
from nltk.corpus import stopwords
from sklearn.metrics.pairwise import linear_kernel
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
import flask
from flask import Flask,render_template, request
import pickle


app = Flask(__name__) # initializing a flask app
model=pickle.load(open("restaurant.pkl",'rb'))  #loading the model

#loading the updated dataset
zomato_df=pd.read_csv("restaurant1.csv")
```

```
@app.route('/')# route to display the home page
def home():
    return render_template('home.html')#rendering the home page


@app.route('/extractor')
def extractor():
    return render_template('extractor.html')

#extractor page
@app.route('/keywords',  methods=['POST'])# route to show the
predictions in a web UI
def keywords():
    #typ=request.form['type']
    output=request.form['output']
    #if typ=="text":
        #output=re.sub("[^a-zA-Z.,]"," ",output)
    print(output)
    print(type(output))
    df_percent = zomato_df.sample(frac=0.5)
    df_percent.set_index('name', inplace=True)
    indices = pd.Series(df_percent.index)
    # Creating tf-idf matrix
    tfidf = TfidfVectorizer(analyzer='word', ngram_range=(1, 2),
min_df=0, stop_words='english')
    tfidf_matrix =
tfidf.fit_transform(df_percent['reviews_list'].fillna(' '))
    cosine_similarities = linear_kernel(tfidf_matrix, tfidf_matrix)

    def recommend(name, cosine_similarities = cosine_similarities):

        # Create a list to put top restaurants
        recommend_restaurant = []

        # Find the index of the hotel entered
        idx = indices[indices == name].index[0]

        # Find the restaurants with a similar cosine-sim value and
order them from bigges number
        score_series =
pd.Series(cosine_similarities[idx]).sort_values(ascending=False)

        # Extract top 30 restaurant indexes with a similar cosine-sim
value
        top30_indexes = list(score_series.iloc[0:31].index)

        # Names of the top 30 restaurants
        for each in top30_indexes:
            recommend_restaurant.append(list(df_percent.index)[each])

        # Creating the new data set to show similar restaurants
        df_new = pd.DataFrame(columns=['cuisines', 'Mean Rating',
'cost'])
```

```
        # Create the top 30 similar restaurants with some of their
columns
        for each in recommend_restaurant:
           df_new =
df_new.append(pd.DataFrame(df_percent[['cuisines','Mean Rating',
'cost']][df_percent.index == each].sample()))

        # Drop the same named restaurants and sort only the top 10 by
the highest rating
        df_new = df_new.drop_duplicates(subset=['cuisines','Mean
Rating', 'cost'], keep=False)
        pd.set_option('display.max_columns', None)

        df_new = df_new.sort_values(by='Mean Rating',
ascending=False).head(10)
        print('TOP %s RESTAURANTS LIKE %s WITH SIMILAR REVIEWS: ' %
(str(len(df_new)), name))

        return df_new

    result = recommend(output)
    print(result)
    print(type(result))
    #print(result[0])

    #print(result[0])
   # res = result.to_string(index=False)



    #showing the prediction results in a UI
    return render_template('keywords.html',keyword=result.to_html())


if __name__ == "__main__":
   # running the app
    app.run(debug=False)
```