Zomato Restaurant Recommendation System with Python

I will introduce you to a machine learning project on Zomato Restaurant Recommendation System with Python programming language. There is an extended class of applications that involve predicting user responses to a variety of options. Such a system is called a recommender system.

A Restaurant recommendation system uses content-based filtering. This method only uses information about the description and attributes of items that users have previously consumed to model user preferences.

In other words, these algorithms try to recommend things similar to what a user liked in the past. The dataset I'll be using here consists of restaurants in Bangalore, India, collected from Zomato. You can download the dataset from https://www.kaggle.com/himanshupoddar/zomato-bangalore-restaurants/download.

To create the Restaurant recommendation system, I will create a content-based recommendation system where when I enter the name of a restaurant, the Restaurant recommendation system will look at reviews from other restaurants, and System will recommend us to the other restaurants with similar reviews and sort them from the top-rated.

I will start the task of Zomato Restaurant Recommendation System by installing the necessary Python Libraries using pip:

In [2]:
```
!pip install numpy
!pip install pandas
!pip install seaborn
!pip install matplotlib
!pip install sklearn
!pip install nltk
```

```
Requirement already satisfied: numpy in c:\users\asus\desktop\vaibhav\env\lib\site-packa
ges (1.18.5)
Requirement already satisfied: pandas in c:\users\asus\desktop\vaibhav\env\lib\site-pack
ages (1.1.4)
Requirement already satisfied: pytz>=2017.2 in c:\users\asus\desktop\vaibhav\env\lib\sit
e-packages (from pandas) (2020.4)
Requirement already satisfied: numpy>=1.15.4 in c:\users\asus\desktop\vaibhav\env\lib\si
te-packages (from pandas) (1.18.5)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\asus\desktop\vaibhav\e
nv\lib\site-packages (from pandas) (2.8.1)
Requirement already satisfied: six>=1.5 in c:\users\asus\desktop\vaibhav\env\lib\site-pa
ckages (from python-dateutil>=2.7.3->pandas) (1.15.0)
Collecting seaborn
  Downloading seaborn-0.11.1-py3-none-any.whl (285 kB)
Requirement already satisfied: numpy>=1.15 in c:\users\asus\desktop\vaibhav\env\lib\site
-packages (from seaborn) (1.18.5)
Requirement already satisfied: scipy>=1.0 in c:\users\asus\desktop\vaibhav\env\lib\site-
packages (from seaborn) (1.5.4)
Requirement already satisfied: pandas>=0.23 in c:\users\asus\desktop\vaibhav\env\lib\sit
e-packages (from seaborn) (1.1.4)
Collecting matplotlib>=2.2
  Using cached matplotlib-3.3.3-cp37-cp37m-win_amd64.whl (8.5 MB)
Requirement already satisfied: pillow>=6.2.0 in c:\users\asus\desktop\vaibhav\env\lib\si
te-packages (from matplotlib>=2.2->seaborn) (8.0.1)
```

```
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in c:\users\asus
\desktop\vaibhav\env\lib\site-packages (from matplotlib>=2.2->seaborn) (2.4.7)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\asus\desktop\vaibhav\env
\lib\site-packages (from matplotlib>=2.2->seaborn) (2.8.1)
Collecting cycler>=0.10
  Using cached cycler-0.10.0-py2.py3-none-any.whl (6.5 kB)
Requirement already satisfied: six in c:\users\asus\desktop\vaibhav\env\lib\site-package
s (from cycler>=0.10->matplotlib>=2.2->seaborn) (1.15.0)
Collecting kiwisolver>=1.0.1
  Using cached kiwisolver-1.3.1-cp37-cp37m-win_amd64.whl (51 kB)
Requirement already satisfied: pytz>=2017.2 in c:\users\asus\desktop\vaibhav\env\lib\sit
e-packages (from pandas>=0.23->seaborn) (2020.4)
Installing collected packages: kiwisolver, cycler, matplotlib, seaborn
Successfully installed cycler-0.10.0 kiwisolver-1.3.1 matplotlib-3.3.3 seaborn-0.11.1
Requirement already satisfied: matplotlib in c:\users\asus\desktop\vaibhav\env\lib\site-
packages (3.3.3)
Requirement already satisfied: numpy>=1.15 in c:\users\asus\desktop\vaibhav\env\lib\site
-packages (from matplotlib) (1.18.5)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\asus\desktop\vaibhav\env\li
b\site-packages (from matplotlib) (1.3.1)
Requirement already satisfied: pillow>=6.2.0 in c:\users\asus\desktop\vaibhav\env\lib\si
te-packages (from matplotlib) (8.0.1)
Requirement already satisfied: cycler>=0.10 in c:\users\asus\desktop\vaibhav\env\lib\sit
e-packages (from matplotlib) (0.10.0)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\asus\desktop\vaibhav\env
\lib\site-packages (from matplotlib) (2.8.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in c:\users\asus
\desktop\vaibhav\env\lib\site-packages (from matplotlib) (2.4.7)
Requirement already satisfied: six in c:\users\asus\desktop\vaibhav\env\lib\site-package
s (from cycler>=0.10->matplotlib) (1.15.0)
Collecting sklearn
  Downloading sklearn-0.0.tar.gz (1.1 kB)
Collecting scikit-learn
  Downloading scikit_learn-0.24.0-cp37-cp37m-win_amd64.whl (6.8 MB)
Requirement already satisfied: numpy>=1.13.3 in c:\users\asus\desktop\vaibhav\env\lib\si
te-packages (from scikit-learn->sklearn) (1.18.5)
Requirement already satisfied: scipy>=0.19.1 in c:\users\asus\desktop\vaibhav\env\lib\si
te-packages (from scikit-learn->sklearn) (1.5.4)
Collecting joblib>=0.11
  Downloading joblib-1.0.0-py3-none-any.whl (302 kB)
Collecting threadpoolctl>=2.0.0
  Downloading threadpoolctl-2.1.0-py3-none-any.whl (12 kB)
Building wheels for collected packages: sklearn
  Building wheel for sklearn (setup.py): started
  Building wheel for sklearn (setup.py): finished with status 'done'
  Created wheel for sklearn: filename=sklearn-0.0-py2.py3-none-any.whl size=1315 sha256=
28d46a16576ccd4a9420431934dd7da3693a8db5fdb4565c065185516d3a603e
  Stored in directory: c:\users\asus\appdata\local\pip\cache\wheels\46\ef\c3\157e41f5ee1
372d1be90b09f74f82b10e391eaacca8f22d33e
Successfully built sklearn
Installing collected packages: threadpoolctl, joblib, scikit-learn, sklearn
Successfully installed joblib-1.0.0 scikit-learn-0.24.0 sklearn-0.0 threadpoolctl-2.1.0
Collecting nltk
  Downloading nltk-3.5.zip (1.4 MB)
Requirement already satisfied: click in c:\users\asus\desktop\vaibhav\env\lib\site-packa
ges (from nltk) (7.1.2)
Requirement already satisfied: joblib in c:\users\asus\desktop\vaibhav\env\lib\site-pack
ages (from nltk) (1.0.0)
Collecting regex
  Downloading regex-2020.11.13-cp37-cp37m-win_amd64.whl (269 kB)
Collecting tqdm
  Downloading tqdm-4.55.1-py2.py3-none-any.whl (68 kB)
Building wheels for collected packages: nltk
  Building wheel for nltk (setup.py): started
  Building wheel for nltk (setup.py): finished with status 'done'
```

```
  Created wheel for nltk: filename=nltk-3.5-py3-none-any.whl size=1434673 sha256=c2db532
85113df0f267178c8136d01ef488bb922ed50b8d01d3bec583a111515
  Stored in directory: c:\users\asus\appdata\local\pip\cache\wheels\45\6c\46\a1865e7ba70
6b3817f5d1b2ff7ce8996aabdd0d03d47ba0266
Successfully built nltk
Installing collected packages: tqdm, regex, nltk
Successfully installed nltk-3.5 regex-2020.11.13 tqdm-4.55.1
ERROR: Could not find a version that satisfies the requirement re
ERROR: No matching distribution found for re
ERROR: Could not find a version that satisfies the requirement warnings
ERROR: No matching distribution found for warnings
```

Now ,I will start importing the necessary Python Libraries:

In [2]:
```
import numpy as np
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import r2_score
import warnings
warnings.filterwarnings('always')
warnings.filterwarnings('ignore')
import re
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
```

Now, I will load and read the dataset:

In [3]:
```
zomato_real = pd.read_csv("zomato.csv")

# prints the first 5 rows of the dataset
zomato_real.head()
```

Out[3]:

	url	address	name	online_order	book_table	rate	v
0	https://www.zomato.com/bangalore/jalsa-banasha...	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	Yes	Yes	4.1/5	
1	https://www.zomato.com/bangalore/spice-elephan...	2nd Floor, 80 Feet Road, Near Big Bazaar, 6th ...	Spice Elephant	Yes	No	4.1/5	

	url	address	name	online_order	book_table	rate
2	https://www.zomato.com/SanchurroBangalore?cont...	1112, Next to KIMS Medical College, 17th Cross...	San Churro Cafe	Yes	No	3.8/5
3	https://www.zomato.com/bangalore/addhuri-udupi...	1st Floor, Annakuteera, 3rd Stage, Banashankar...	Addhuri Udupi Bhojana	No	No	3.7/5
4	https://www.zomato.com/bangalore/grand-village...	10, 3rd Floor, Lakshmi Associates, Gandhi Baza...	Grand Village	No	No	3.8/5

Now the next step is data cleaning and feature engineering for this step we need to do a lot of stuff with the data

In [4]:
```
#Deleting Unnnecessary Columns
zomato=zomato_real.drop(['url','dish_liked','phone'],axis=1) #Dropping the column "dish
```

In [5]:
```
#Removing the Duplicates
zomato.duplicated().sum()
zomato.drop_duplicates(inplace=True)
```

In [6]:
```
#Remove the NaN values from the dataset
zomato.isnull().sum()
zomato.dropna(how='any',inplace=True)
```

In [7]:
```
#Changing the column names
zomato = zomato.rename(columns={'approx_cost(for two people)':'cost','listed_in(type)':
```

In [8]:
```
#Some Transformations
zomato['cost'] = zomato['cost'].astype(str) #Changing the cost to string
zomato['cost'] = zomato['cost'].apply(lambda x: x.replace(',','.')) #Using lambda funct
zomato['cost'] = zomato['cost'].astype(float)
```

In [9]:
```
#Removing '/5' from Rates
zomato = zomato.loc[zomato.rate !='NEW']
zomato = zomato.loc[zomato.rate !='-'].reset_index(drop=True)
remove_slash = lambda x: x.replace('/5', '') if type(x) == np.str else x
zomato.rate = zomato.rate.apply(remove_slash).str.strip().astype('float')
```

In [10]:
```
# Adjust the column names
zomato.name = zomato.name.apply(lambda x:x.title())
zomato.online_order.replace(('Yes','No'),(True, False),inplace=True)
zomato.book_table.replace(('Yes','No'),(True, False),inplace=True)
```

In [11]:
```
## Computing Mean Rating
restaurants = list(zomato['name'].unique())
zomato['Mean Rating'] = 0
```

```
for i in range(len(restaurants)):
    zomato['Mean Rating'][zomato['name'] == restaurants[i]] = zomato['rate'][zomato['na

from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range = (1,5))
zomato[['Mean Rating']] = scaler.fit_transform(zomato[['Mean Rating']]).round(2)
```

Now the next step is to perform some text preprocessing steps which include:

In [12]:
```
## Lower Casing
zomato["reviews_list"] = zomato["reviews_list"].str.lower()
```

In [13]:
```
## 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["reviews_list"] = zomato["reviews_list"].apply(lambda text: remove_punctuation(t
```

In [15]:
```
## Removal of Stopwords
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
STOPWORDS = set(stopwords.words('english'))
def remove_stopwords(text):
    """custom function to remove the stopwords"""
    return " ".join([word for word in str(text).split() if word not in STOPWORDS])
zomato["reviews_list"] = zomato["reviews_list"].apply(lambda text: remove_stopwords(tex
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\ASUS\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
```

In [16]:
```
## Removal of URLS
def remove_urls(text):
    url_pattern = re.compile(r'https?://\S+|www\.\S+')
    return url_pattern.sub(r'', text)

zomato["reviews_list"] = zomato["reviews_list"].apply(lambda text: remove_urls(text))
zomato[['reviews_list', 'cuisines']].sample(5)
```

Out[16]:

	reviews_list	cuisines
33169	rated 10 ratedn satisfied everything food drin...	North Indian, Chinese, Andhra
18490	rated 40 ratedn small place donã□x83ã□x83ã□x82...	North Indian
29256	rated 10 ratedn went today lunch horrible expe...	North Indian, Bengali
30993	rated 10 ratedn ordered spaghetti aglio pasta ...	Continental, Burger, Cafe, Steak
15601	rated 10 ratedn food prepared dirty oil firstl...	Street Food

In [17]:
```
# RESTAURANT NAMES:
restaurant_names = list(zomato['name'].unique())
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)
```

```
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.items()]
    words_freq =sorted(words_freq, key = lambda x: x[1], reverse=True)
    return words_freq[:top_nu_of_words]

zomato=zomato.drop(['address','rest_type', 'type', 'menu_item', 'votes'],axis=1)
import pandas

# Randomly sample 60% of your dataframe
df_percent = zomato.sample(frac=0.5)
```

TF-IDF Vectorization

TF-IDF (Term Frequency-Inverse Document Frequency) vectors for each document. This will give you a matrix where each column represents a word in the general vocabulary (all words that appear in at least one document) and each column represents a restaurant, as before.

TF-IDF is the statistical method of assessing the meaning of a word in a given document. Now, I will use the TF-IDF vectorization on the dataset:

In [18]:
```
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='engl
tfidf_matrix = tfidf.fit_transform(df_percent['reviews_list'])

cosine_similarities = linear_kernel(tfidf_matrix, tfidf_matrix)
```

Now the last step for creating a Restaurant Recommendation System is to write a function that will recommend restaurants:

In [19]:
```
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 n
    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
```

```
    # 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=Fal
    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
```

In [20]:
```
recommend('Pai Vihar')
```

```
TOP 10 RESTAURANTS LIKE Pai Vihar WITH SIMILAR REVIEWS:
```

Out[20]:

	cuisines	Mean Rating	cost
Burma Burma	Asian, Burmese	4.74	1.5
Cravy Wings	American, Burger, Fast Food	4.11	400.0
Ilyazsab The House Of Chicken	Rolls, Kebab	3.84	250.0
Andhra Ruchulu	Andhra, North Indian	3.72	400.0
Andhra Ruchulu	Andhra, Biryani, North Indian, Chinese	3.72	1.0
Dum Biryani Hub	Biryani	3.71	700.0
Cafe Aladdin	Cafe, Chinese	3.71	500.0
Wow Paratha	North Indian	3.71	400.0
1992 Chats - Space	Street Food	3.45	200.0
Nys Kitchen	North Indian, Chinese	3.39	500.0

In [ ]:
```

```