RESTAURANT DATA ANALYSIS

MINING OF MASSIVE DATASET

TEAM

1	CB.EN.U4CSE19013	CH. Rohit Krishna Sai
2	CB.EN.U4CSE19015	Sahitya Edarada
3	CB.EN.U4CSE19054	Srikar L
4	CB.EN.U4CSE19110	D. Monish

INTRODUCTION

Restaurants are playing a greater role in boosting consumer health by offering menu items that are both nutritious and delicious. So we are analyzing the dataset which was taken by zomato to identify various types of restaurants, cuisines, foods, and some other data.

ALGORITHMS

1

TF - IDF

2

APRIORI ALGORITHM

3

JACCARD SIMILARITY

DATASET

We have taken the **Restaurant Recommendation** Dataset from kaggle

The Data used in our dataset is from the restaurants in Bangalore from zomato.

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

PARAMETERS IN DATASET

- reviews_list - list of tuples containing reviews for the restaurant, each tuple consists of two values, rating and review by the customer
- dish_liked - dishes people liked in the restaurant
- online_order - whether online ordering is available in the restaurant
- not book_table - table book option available or not
- Restaurant details like location, name, phone, and address.

PREPROCESSING

```
[21] #Remove the NaN values from the dataset
     df1.isnull().sum()
     df1.dropna(how='any',inplace=True)
```

```
[20] #Removing the Duplicates
     df1.duplicated().sum()
     df1.drop_duplicates(inplace=True)
```

```
[29] df.shape
```

```
(51717, 17)
```

```
df.describe
```

```
<bound method NDFrame.describe of
0      https://www.zomato.com/bangalore/jalsa-banasha...
1      https://www.zomato.com/bangalore/spice-elephan...
2      https://www.zomato.com/SanchurroBangalore?cont...
3      https://www.zomato.com/bangalore/addhuri-udupi...
4      https://www.zomato.com/bangalore/grand-village...
...                                                  ...
51712  https://www.zomato.com/bangalore/best-brews-fo...
51713  https://www.zomato.com/bangalore/vinod-bar-and...
51714  https://www.zomato.com/bangalore/plunge-sherat...
51715  https://www.zomato.com/bangalore/chime-sherato...
51716  https://www.zomato.com/bangalore/the-nest-the-...

                                                address  \
```

PREPROCESSING

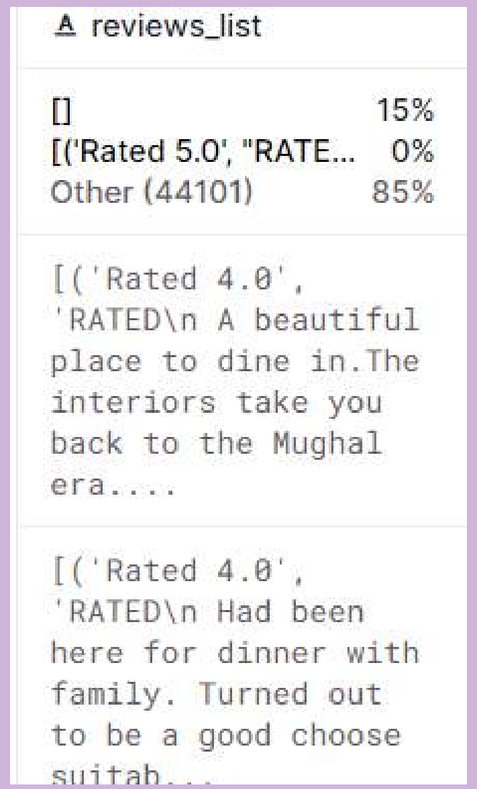
```
## Removal of Stopwords
from nltk.corpus import stopwords
STOPWORDS = set(stop)
def remove_stopwords(text):
    """custom function to remove the stopwords"""
    return " ".join([word for word in str(text).split() if word not in STOPWORDS])

zomato["reviews_list"] = zomato["reviews_list"].apply(lambda text: remove_stopwords(text))
```

```
## 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)
```

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

TF - IDF

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



TF - IDF

```
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)
    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
recommend('Pai Vihar')
```

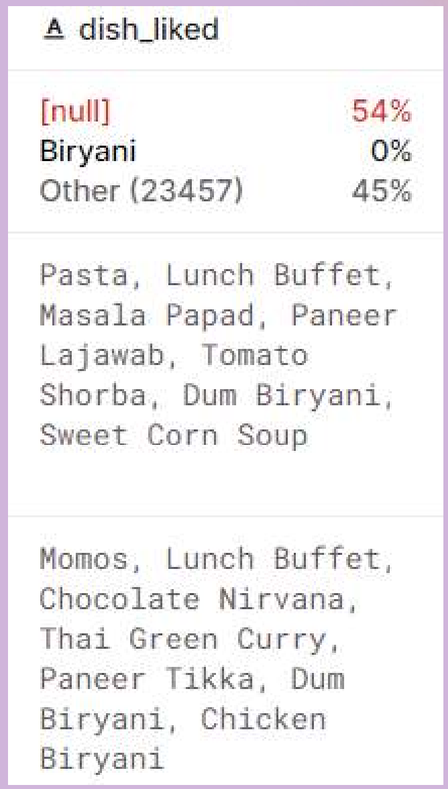
OUTPUT

TOP 10 RESTAURANTS LIKE Pai Vihar WITH SIMILAR REVIEWS:

	cuisines	Mean Rating	cost
Myu Bar At Gilly'S Redefined	American, North Indian, Pizza, Finger Food, Co...	4.41	1.2
Sixth Avenue Cafe And Patisserie	Cafe, Italian, Desserts	4.36	600.0
Zee5 Loft	Cafe, North Indian, Pizza, Sandwich, Salad	4.35	500.0
The Only Place	Italian, American, Steak, Continental, BBQ, Salad	4.10	1.0
The Yogisthaan Cafe	Cafe, Healthy Food, Salad, Beverages	4.01	600.0
World Of Asia	Asian	3.84	300.0
Tangerine - Davanam Sarovar Portico Suites	Continental, Italian, North Indian, Asian, Bak...	3.84	1.1

APRIORI ALGORITHM

dish_liked column will help us to identify the most frequent food pairs in the dataset. This column contains a like of items that are ordered.



APRIORI ALGORITHM

```python
from efficient_apriori import apriori

# our min support is 7, but it has to be expressed as a percentage for efficient-apriori
min_support = 7/len(cuisine)
# min confidence allows you to delete rules with low confidence.
# For now set min_confidence = 0 to obtain all the rules
min_confidence = 0
itemsets, rules = apriori(cuisine,min_support=min_support, min_confidence=min_confidence)
```

```python
for rule in rules:
  print(rule)
```

```
{Chinese} -> {Andhra} (conf: 0.056, supp: 0.018, lift: 1.867, conv: 1.028)
{Andhra} -> {Chinese} (conf: 0.583, supp: 0.018, lift: 1.867, conv: 1.650)
{North Indian} -> {Andhra} (conf: 0.072, supp: 0.025, lift: 2.398, conv: 1.045)
{Andhra} -> {North Indian} (conf: 0.833, supp: 0.025, lift: 2.398, conv: 3.915)
{Desserts} -> {Bakery} (conf: 0.216, supp: 0.020, lift: 5.405, conv: 1.225)
{Bakery} -> {Desserts} (conf: 0.500, supp: 0.020, lift: 5.405, conv: 1.815)
{Cafe} -> {Beverages} (conf: 0.180, supp: 0.022, lift: 1.565, conv: 1.079)
{Beverages} -> {Cafe} (conf: 0.196, supp: 0.022, lift: 1.565, conv: 1.088)
{Desserts} -> {Beverages} (conf: 0.243, supp: 0.022, lift: 2.115, conv: 1.169)
{Beverages} -> {Desserts} (conf: 0.196, supp: 0.022, lift: 2.115, conv: 1.128)
{Fast Food} -> {Beverages} (conf: 0.161, supp: 0.037, lift: 1.403, conv: 1.055)
{Beverages} -> {Fast Food} (conf: 0.326, supp: 0.037, lift: 1.403, conv: 1.139)
{Ice Cream} -> {Beverages} (conf: 0.474, supp: 0.022, lift: 4.119, conv: 1.681)
{Beverages} -> {Ice Cream} (conf: 0.196, supp: 0.022, lift: 4.119, conv: 1.184)
{North Indian} -> {Beverages} (conf: 0.050, supp: 0.018, lift: 0.438, conv: 0.932)
```

JACCARD SIMILARITY

Jaccard similarity can be applied to find the similarity between any restaurant.

- The similarity between each restaurant can be calculated like cuisines, rest_type.

rest_type		cuisines	
Quick Bites	37%	North Indian	6%
Casual Dining	20%	North Indian, Chinese	5%
Other (22255)	43%	Other (46419)	90%
Casual Dining		North Indian, Mughlai, Chinese	
Casual Dining		Chinese, North Indian, Thai	

JACCARD SIMILARITY

```
def jaccard(list1, list2):
  if(types[0]==types[1]):
    intersection = len(list(set(list1).intersection(list2)))
    union = (len(list1) + len(list2)) - intersection
    return float(intersection) / union
#find Jaccard Similarity between the two restaurants
```

```
print(jaccard(cuisine[n1],cuisine[n2]))
```

```
0.25
```

THANK YOU!