Algorithms for Information Retrieval and Intelligence Web

ASSIGNMENT 2

Team Members

Anushka Jalori PES1UG20CS071

Ayushi Soumya PES1UG20CS097

Gaurav Mahajan PES1UG20CS150

Problem statement

Restaurant recommender system using the Zomato's reviews data set.

Introduction

Searching for restaurants on the current Zomato app is purely based on dish and restaurant names. Our goal at the end of this project is to provide a recommendation system not limited by these constraints. We want the user to be able to write down what they are searching for in a restaurant and based on these requirements provide users with the restaurants with their criteria.

To determine the facilities provided by the restaurant at the customer level we will be making use of customer reviews as they best represent what someone can expect at a restaurant. For example, if one types in 'Fun Outings' we want to be able to display all restaurants that may cater to this need and display them to the user.

Data set description

Dataset source: The dataset 'Zomato Bangalore Restaurants' is publicly available on the Kaggle website.

Dataset size: Our dataset contains 5171 row entries and 17 attributes.

Dataset link: https://www.kaggle.com/datasets/himanshupoddar/Zomato-bangalore-restaurants

EDA and Preprocessing

Exploratory data analysis is used by data scientists to analyze dataset and summarize their main characteristics, often employing visualization methods.

DATA CLEANING AND EDA

In [1]:
```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

READING THE CSV FILE

In [2]:
```
df_zomato = pd.read_csv("zomato.csv")
df_zomato.head()
df_zomato.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51717 entries, 0 to 51716
Data columns (total 17 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   url                           51717 non-null  object
 1   address                       51717 non-null  object
 2   name                          51717 non-null  object
 3   online_order                  51717 non-null  object
 4   book_table                    51717 non-null  object
 5   rate                          43942 non-null  object
 6   votes                         51717 non-null  int64
 7   phone                         50509 non-null  object
 8   location                      51696 non-null  object
 9   rest_type                     51490 non-null  object
 10  dish_liked                    23639 non-null  object
 11  cuisines                      51672 non-null  object
```

- Dropping columns not needed and the null values

In [3]:

```
df_zomato.shape
```

Out[3]: (51717, 17)

IT CAN BE OBSERVED MANY ATTRIBUTES HAVE MISSING VALUES

Dish liked has the most null attributes not a good idea to use it for analysis of ALL restaurants

In [4]:

```
df_zomato[["dish_liked"]]
```

Out[4]:

	dish_liked
0	Pasta, Lunch Buffet, Masala Papad, Paneer Laja...
1	Momos, Lunch Buffet, Chocolate Nirvana, Thai G...
2	Churros, Cannelloni, Minestrone Soup, Hot Choc...
3	Masala Dosa
4	Panipuri, Gol Gappe
...	...
51712	NaN
51713	NaN
51714	NaN
51715	Cocktails, Pizza, Buttermilk

In [5]:

```
df_zomato.columns
```

Out[5]:

```
Index(['url', 'address', 'name', 'online_order', 'book_table', 'rate', 'votes',
       'phone', 'location', 'rest_type', 'dish_liked', 'cuisines',
       'approx_cost(for two people)', 'reviews_list', 'menu_item',
       'listed_in(type)', 'listed_in(city)'],
      dtype='object')
```

In [6]:

```
#Dropping Columns not needed at the moment
df_zomato = df_zomato.drop(['url','phone'],axis = 1)
df_zomato.head()
```

Out[6]:

	address	name	online_order	book_table	rate	votes	location	rest_type	dish_liked	cuisines	approx_cost(for two people)	reviews_list	menu_item	listed_in(type)	listed
0	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	Yes	Yes	4.1/5	775	Banashankari	Casual Dining	Pasta, Lunch Buffet, Masala Papad, Paneer Laja...	North Indian, Mughlai, Chinese	800	[('Rated 4.0', 'RATED\n A beautiful place to ...	[]	Buffet	Bana
1	2nd Floor, 80 Feet Road, Near Big Bazaar, 6th ...	Spice Elephant	Yes	No	4.1/5	787	Banashankari	Casual Dining	Momos, Lunch Buffet, Chocolate Nirvana, Thai G...	Chinese, North Indian, Thai	800	[('Rated 4.0', 'RATED\n Had been here for din...	[]	Buffet	Bana

- Removing the duplicate values

In [7]:
```
#Removing all duplicate valuesdf_zomato.drop_duplicates(inplace = True)
```

In [8]:
```
df_zomato.drop_duplicates(inplace = True)
df_zomato.shape
```

Out[8]: (51674, 15)

THIS HOWEVER DOES NOT TAKE CARE OF REMOVAL OF RESTAURANTS WITH SAME NAME

In [9]:
```
example = df_zomato[df_zomato['name'] == 'Jalsa']
example
```

Out[9]:

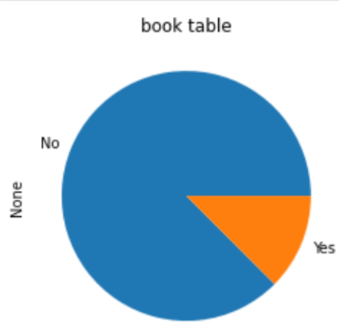
	address	name	online_order	book_table	rate	votes	location	rest_type	dish_liked	cuisines	approx_cost(for two people)	reviews_list	menu_item	listed_in(type
0	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	Yes	Yes	4.1/5	775	Banashankari	Casual Dining	Pasta, Lunch Buffet, Masala Papad, Paneer Laja...	North Indian, Mughlai, Chinese	800	[('Rated 4.0', 'RATED\n A beautiful place to ...	[]	Buffe
456	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	Yes	Yes	4.1/5	775	Banashankari	Casual Dining	Pasta, Lunch Buffet, Masala Papad, Paneer Laja...	North Indian, Mughlai, Chinese	800	[('Rated 4.0', 'RATED\n A beautiful place to ...	[]	Deliver

EDA:

OBSERVING ONLINE ORDER RATIO

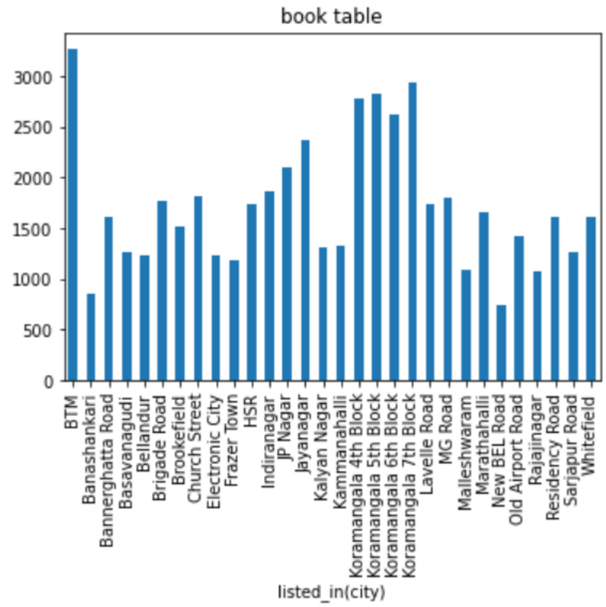
In [10]:
```
k = df_zomato.groupby(['online_order']).size().plot(kind='pie', title = 'online ordering')
```

In [11]:
```
k = df_zomato.groupby(['book_table']).size().plot(kind='pie', title = 'book table')
```



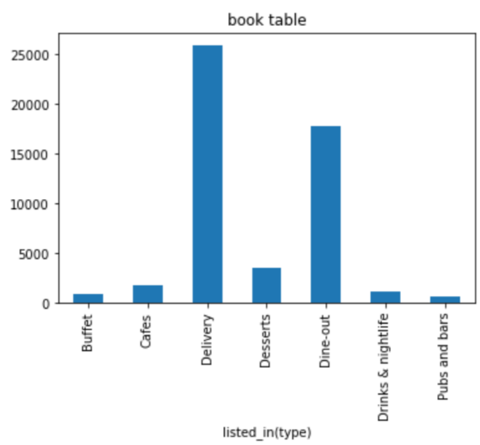
In [12]:

```
k = df_zomato.groupby(['listed_in(city)']).size().plot(kind='bar', title = 'book table')
#btm layout has max number of restaurants
```



In [13]:

```
k = df_zomato.groupby(['listed_in(type)']).size().plot(kind='bar', title = 'book table')
#delivery takes the lead in modern bangalore
```



In [14]:

```
#now i will try to get the number of high rated restaurants in bangalore grouped by area
#first i will clean the rating column
df_zomato['rate'].isnull().sum()
```

Out[14]: 7767

Cleaning the rating column:

- *get the number of high rated restaurants in Bangalore grouped by area*

In [17]:
```
#upon working on rate column pbserved that data has characters like /
df_zomato['rate'].unique()
```

Out[17]:
```
array(['4.1/5', '3.8/5', '3.7/5', '3.6/5', '4.6/5', '4.0/5', '4.2/5',
       '3.9/5', '3.1/5', '3.0/5', '3.2/5', '3.3/5', '2.8/5', '4.4/5',
       '4.3/5', 'NEW', '2.9/5', '3.5/5', nan, '2.6/5', '3.8 /5', '3.4/5',
       '4.5/5', '2.5/5', '2.7/5', '4.7/5', '2.4/5', '2.2/5', '2.3/5',
       '3.4 /5', '-', '3.6 /5', '4.8/5', '3.9 /5', '4.2 /5', '4.0 /5',
       '4.1 /5', '3.7 /5', '3.1 /5', '2.9 /5', '3.3 /5', '2.8 /5',
       '3.5 /5', '2.7 /5', '2.5 /5', '3.2 /5', '2.6 /5', '4.5 /5',
       '4.3 /5', '4.4 /5', '4.9/5', '2.1/5', '2.0/5', '1.8/5', '4.6 /5',
       '4.9 /5', '3.0 /5', '4.8 /5', '2.3 /5', '4.7 /5', '2.4 /5',
       '2.1 /5', '2.2 /5', '2.0 /5', '1.8 /5'], dtype=object)
```

In [18]:
```
def handlerate(value):
    if value=='-' or value == 'NEW':
        return np.nan

    else:
        value = str(value).split('/')
        value = value[0]
        return float(value)

df_zomato['rate'] = df_zomato['rate'].apply(handlerate)
```

In [19]:
```
df_zomato['rate']
```

Out[19]:
```
0        4.1
1        4.1
2        3.8
3        3.7
```

- *analyze the costs of restaurants in Bangalore*

In [21]:
```
#koramangala 7th block has the highest rated restaurants
#now we can analyse the costs of restaurants in bangalore
df_zomato['approx_cost(for two people)'].unique()
```

Out[21]:
```
array(['800', '300', '600', '700', '550', '500', '450', '650', '400',
       '900', '200', '750', '150', '850', '100', '1,200', '350', '250',
       '950', '1,000', '1,500', '1,300', '199', '80', '1,100', '160',
       '1,600', '230', '130', '50', '190', '1,700', nan, '1,400', '180',
       '1,350', '2,200', '2,000', '1,800', '1,900', '330', '2,500',
       '2,100', '3,000', '2,800', '3,400', '40', '1,250', '3,500',
       '4,000', '2,400', '2,600', '120', '1,450', '469', '70', '3,200',
       '60', '560', '240', '360', '6,000', '1,050', '2,300', '4,100',
       '5,000', '3,700', '1,650', '2,700', '4,500', '140'], dtype=object)
```

In [22]:
```
def handlecomma(value):
    value = str(value)
    if ',' in value:
        value = value.replace(',', '')
        return float(value)
    else:
        return float(value)

df_zomato['approx_cost(for two people)'] = df_zomato['approx_cost(for two people)'].apply(handlecomma)
df_zomato['approx_cost(for two people)'].unique()
```

Out[22]:
```
array([ 800.,  300.,  600.,  700.,  550.,  500.,  450.,  650.,  400.,
        900.,  200.,  750.,  150.,  850.,  100., 1200.,  350.,  250.,
        950., 1000., 1500., 1300.,  199.,   80., 1100.,  160., 1600.,
        230.,  130.,   50.,  190., 1700.,   nan, 1400.,  180., 1350.,
       2200., 2000., 1800., 1900.,  330., 2500., 2100., 3000., 2800.,
       3400.,   40., 1250., 3500., 4000., 2400., 2600.,  120., 1450.,
        469.,   70., 3200.,   60.,  560.,  240.,  360., 6000., 1050.,
       2300., 4100., 5000., 3700., 1650., 2700., 4500.,  140.])
```

Min and max costs:

In [23]:
```python
df_zomato['approx_cost(for two people)'].min()
```

Out[23]: 40.0

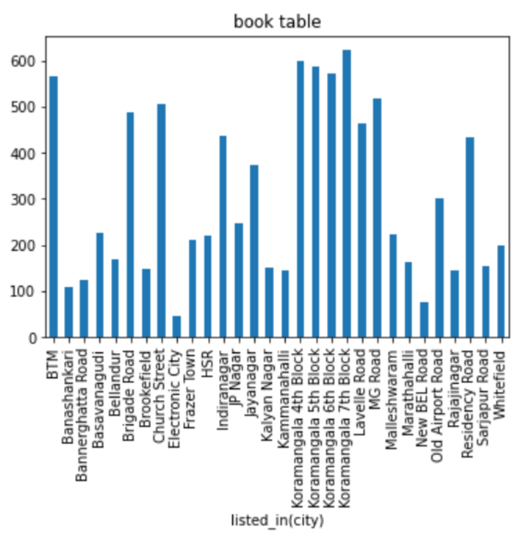
In [24]:
```python
df_zomato['approx_cost(for two people)'].max()
```

Out[24]: 6000.0

SEEING DISTRIBUTION OF HIGH RATED RESTAURANTS

In [25]:
```python
df_high = df_zomato.query('rate > 4')
```

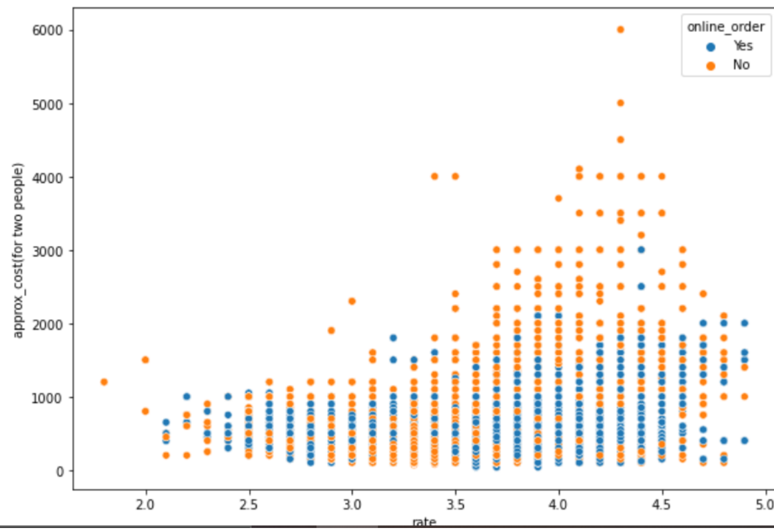
In [26]:
```python
k = df_high.groupby(['listed_in(city)']).size().plot(kind='bar', title = 'book table')
```



Cost VS Rating Scatterplot for predicting correlation

In [27]:
```
cost_dist=df_zomato[['rate','approx_cost(for two people)','online_order']].dropna(axis = 0,how = "any")
```

In [28]:
```
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(10,7))
sns.scatterplot(x="rate",y='approx_cost(for two people)',hue='online_order',data=cost_dist)
plt.show()
```



Explantory variable: Rating

Response variable: Cost

In [29]:
```
xarr=cost_dist['rate'].to_numpy()
yarr=cost_dist['approx_cost(for two people)'].to_numpy()
```

Calculating persons moment correlation coefficient

In [30]:
```
R = np.corrcoef(xarr, yarr)
print(R[0,1])
```

```
0.3850604365893679
```

Since R > 0 we can say that rate and cost are positively correlated YET there are no definitive results

In [31]:
```
df_zomato['rest_type'].value_counts()
```

Out[31]:
```
Quick Bites                   19114
Casual Dining                 10322
Cafe                           3730
Delivery                       2600
Dessert Parlor                 2263
                              ...
Food Court, Beverage Shop         2
Cafe, Food Court                  2
Dessert Parlor, Food Court        2
Sweet Shop, Dessert Parlor        1
```

Methodology

Cleaning cuisines model

In [32]:

```
cuisines = df_zomato['cuisines'].value_counts(ascending  = False)


cuisines_lessthan100 = cuisines[cuisines<100]



def handle_cuisines(value):
    if(value in cuisines_lessthan100):
        return 'others'
    else:
        return value

df_zomato['cuisines'] = df_zomato['cuisines'].apply(handle_cuisines)
df_zomato['cuisines'].value_counts()
```

Out[32]:

```
others                                26440
North Indian                           2912
North Indian, Chinese                  2381
South Indian                           1826
Biryani                                 917
                                      ...  
South Indian, Chinese, North Indian     105
Italian, Pizza                          105
North Indian, Mughlai, Chinese          104
South Indian, Fast Food                 104
North Indian, Chinese, Seafood          102
Name: cuisines, Length: 70, dtype: int64
```

Plotting the Correlation Matrix

In [33]:

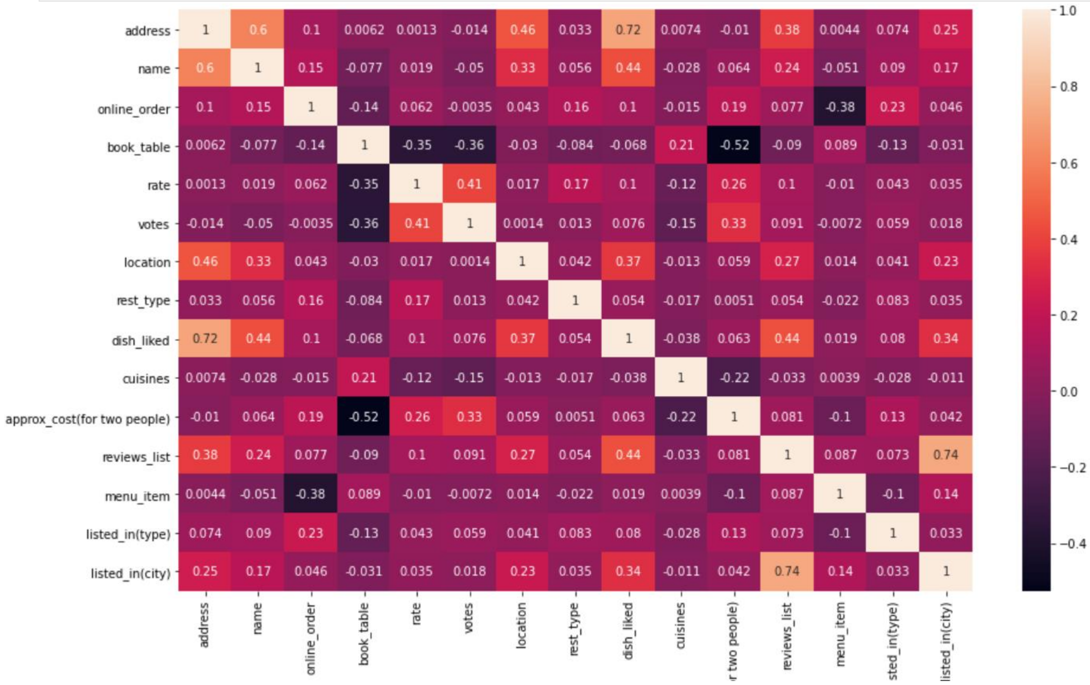
```
df2= df_zomato
df2.dropna(how='any',inplace=True)
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 23248 entries, 0 to 51715
Data columns (total 15 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   address                      23248 non-null  object 
 1   name                         23248 non-null  object 
 2   online_order                 23248 non-null  object 
 3   book_table                   23248 non-null  object 
 4   rate                         23248 non-null  float64
 5   votes                        23248 non-null  int64  
 6   location                     23248 non-null  object 
```

```
In [34]:
#Encode the input Variables
def Encode(zomato):
    for column in zomato.columns[~zomato.columns.isin(['rate', 'approx_cost(for two people)', 'votes'])]:
        zomato[column] = zomato[column].factorize()[0]
    return zomato

zomato_en = Encode(df2.copy())
```

```
In [35]:
corr=zomato_en.corr(method='kendall')
plt.figure(figsize=(15,9))
sns.heatmap(corr,annot=True)
```



Results

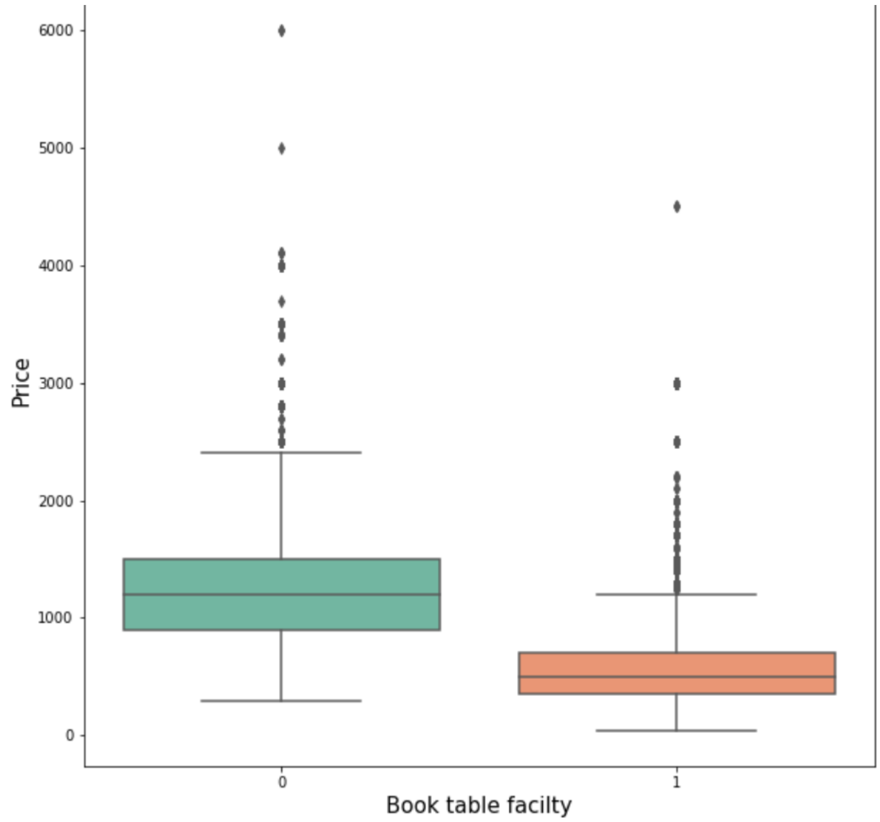
Correlation matrix results:

The highest spurious correlation is between review_list and listed_in(city) = 0.74

book_table and approx_cost (for 2 people) is negatively correlated= -0.52

Restaurants which provide an option of booking table in advance has a high average cost.

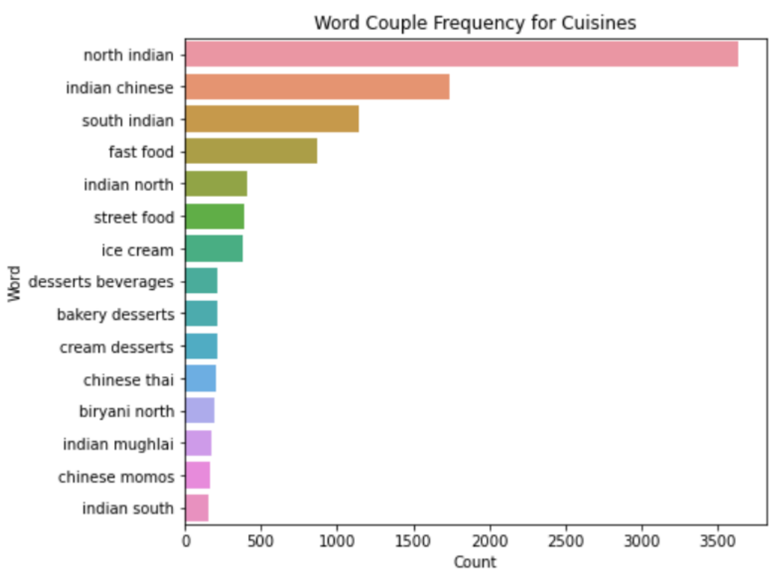
Effect of book table option on restaurant price



Green box plot for restaurants that provide online booking facility

Orange plot for restaurants that do not provide online booking facility

In [38]:
```
# Top 15 two word frequencies for Cuisines
lst = get_top_words(df_zomato['cuisines'], 15, (2,2))
df_words = pd.DataFrame(lst, columns=['Word', 'Count'])
plt.figure(figsize=(7,6))
sns.barplot(data=df_words, x='Count', y='Word')
plt.title('Word Couple Frequency for Cuisines');
```



CONTENT BASED MODEL FOR RECOMMENDATION OF SIMILIAR RESTAURANTS

After cleaning and pre-processing the reviews column:

can do semantic ananlysis of reviews to ensure that the restaurant is liked by the users as well

```
In [46]:
## Lower Casing

df_zomato["reviews_list"] = df_zomato["reviews_list"].str.lower()

## 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))
df_zomato["reviews_list"] = df_zomato["reviews_list"].apply(lambda text: remove_punctuation(text))

# Removal of 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])
df_zomato["reviews_list"] = df_zomato["reviews_list"].apply(lambda text: remove_stopwords(text))

#Cleaning URL
def remove_urls(text):
    url_pattern = re.compile(r'https?://\S+|www\.\S+')
    return url_pattern.sub(r'', text)
df_zomato["reviews_list"] = df_zomato["reviews_list"].apply(lambda text: remove_urls(text))
```

```
In [47]:
df_zomato[df_zomato['address'] == '942, 21st Main Road, 2nd Stage, Banashankari, Bangalore']
```

After removing duplicates, we calculate the cosine similarities

```
In [54]:
tfidf = TfidfVectorizer(analyzer='word', ngram_range=(1, 2), min_df=0, stop_words='english')
tfidf_matrix = tfidf.fit_transform(df_zomato['reviews_list'])
```

```
In [55]:
cosine_similarities = linear_kernel(tfidf_matrix, tfidf_matrix)
```

In [56]:

```python
def recommend(name, cosine_similarities = cosine_similarities):

    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 biggest 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_zomato.index)[each])

    # Creating the new data set to show similar restaurants
    df_new = pd.DataFrame(columns=['cuisines', 'rate', '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_zomato[['cuisines','rate', 'cost']][df_zomato.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','rate', 'cost'], keep=False)
    df_new = df_new.sort_values(by='rate', ascending=False).head(10)

    print('TOP %s RESTAURANTS LIKE %s WITH SIMILAR REVIEWS: ' % (str(len(df_new)), name))

    return df_new
```

Results

In [58]:

```python
df_zomato.loc['Jalsa'][:1]
```

Out[58]:

name	address	online_order	book_table	rate	votes	location	rest_type	dish_liked	cuisines	cost	reviews_list	menu_item	listed_in(type)	listed_i
Jalsa	942, 21st Main Road, 2nd Stage, Banashankari, ...	Yes	Yes	4.1	783	Banashankari	Casual Dining	Pasta, Lunch Buffet, Masala Papad, Paneer Laja...	North Indian, Mughlai, Chinese	800.0	rated 40 ratedn beautiful place dine inthe int...	[]	Dine-out	Jay

In [59]:

```python
recommend('Jalsa')
```

```
TOP 10 RESTAURANTS LIKE Jalsa WITH SIMILAR REVIEWS:
```

Out[59]:

	cuisines	rate	cost
Byg Brewski Brewing Company	others	4.9	1600.0
Biergarten	others	4.8	2100.0
The Black Pearl	others	4.8	1500.0
Truffles	others	4.7	900.0
AB's - Absolute Barbecues	others	4.7	1600.0
Brew and Barbeque - A Microbrewery Pub	others	4.6	1400.0
Big Pitcher	others	4.6	1800.0
Koramangala Social	others	4.6	1500.0

NOW USING THIS CONTENT BASED MODEL TO FIND SEARCH QUERY BASED RECOMMENDATIONS

In [79]:
```
df_zomato = df_zomato.append({"reviews_list": "outdoor family","name":"thisisuser"}, ignore_index=True)
```

In [80]:
```
# Creating tf-idf matrix
tfidf = TfidfVectorizer(analyzer='word', ngram_range=(1, 2), min_df=0, stop_words='english')
tfidf_matrix = tfidf.fit_transform(df_zomato['reviews_list'])
```

In [81]:
```
cosine_similarities = linear_kernel(tfidf_matrix, tfidf_matrix)
```

In [82]:
```
def recommend(cosine_similarities = cosine_similarities):#location,

    recommend_restaurant = []

    # Find the index of the hotel entered
    idx = df_zomato[df_zomato['name'] == "thisisuser"].index[0]

    # Find the restaurants with a similar cosine-sim value and order them from biggest 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_zomato.index)[each])

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

    # Create the top 30 similar restaurants with some of their columns
    for each in recommend_restaurant:
        df_new = df_new.append(pd.DataFrame(df_zomato[['name','cuisines','rate', 'cost','location']][df_zomato.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=['name','cuisines','rate', 'cost','location'], keep=False)
    df_new = df_new.sort_values(by='rate', ascending=False).head(10)
    #df_new = df_new[df_new['location'] == location]

    return df_new
```

In [83]:
```
recommend()
```

RESULTS:

Out[83]:

	name	cuisines	rate	cost	location
1068	Hakuna Matata	others	4.5	1200.0	JP Nagar
4234	Opus Food Stories	others	4.5	1800.0	Sarjapur Road
2337	Ssaffron - Shangri-La Hotel	North Indian	4.4	3000.0	Vasanth Nagar
104	Spice Elephant	others	4.1	800.0	Banashankari
1421	Caffe Pascucci	others	4.1	950.0	HSR
2001	Fresh Pressery Cafe	others	4.1	1200.0	Koramangala 5th Block
4377	Herbs & Spices	others	4.0	1000.0	Whitefield
2371	1947	North Indian, Chinese	4.0	950.0	Malleshwaram
906	Adithya	South Indian, North Indian, Chinese	4.0	450.0	JP Nagar
1748	Bella	others	3.9	1000.0	Jayanagar

Latent Dirichlet Allocation (LDA) model with search query result

Latent Dirichlet allocation (LDA) is a generative probabilistic model of a corpus. The basic idea is that documents are represented as random mixtures over latent(hidden) topics, where each topic is characterized by a distribution over words.

In [86]:

```
df= pd.read_csv("zomato.csv")
```

In [87]:

```
from tqdm import tqdm
all_ratings = []

for name,ratings in tqdm(zip(df['name'],df['reviews_list'])):
    ratings = eval(ratings)
    for score, doc in ratings:
        if score:
            score = score.strip("Rated").strip()
            doc = doc.strip('RATED').strip()
            score = float(score)
            all_ratings.append([name,score, doc])
```

```
51717it [00:30, 1690.61it/s]
```

In [88]:

```
rating_df=pd.DataFrame(all_ratings,columns=['name','rating','review'])
rating_df['review']=rating_df['review'].apply(lambda x : re.sub('[^a-zA-Z0-9\s]',"",x))
```

In [89]:

```
from nltk import word_tokenize, pos_tag
def nouns_adj(text):
    '''Given a string of text, tokenize the text and pull out only     the nouns and adjectives.'''
    is_noun_adj = lambda pos: pos[:2] == 'NN' or pos[:2] == 'JJ'
    tokenized = word_tokenize(text)
    nouns_adj = [word for (word, pos) in pos_tag(tokenized) if   is_noun_adj(pos)]
    return ' '.join(nouns_adj)
```

In [90]:

```
rating_df.drop_duplicates(subset = ['name'], inplace = True)
```

In [91]:
```python
data_nouns_adj = pd.DataFrame(rating_df.review.apply(nouns_adj))
data_nouns_adj['name'] = rating_df['name']
data_nouns_adj['rating'] = rating_df['rating']
```

In [93]:
```python
data_nouns_adj = data_nouns_adj[data_nouns_adj['rating']>3.5]
data_nouns_adj = data_nouns_adj.sort_values(by = ['rating'],ascending = False)
data_nouns_adj
```

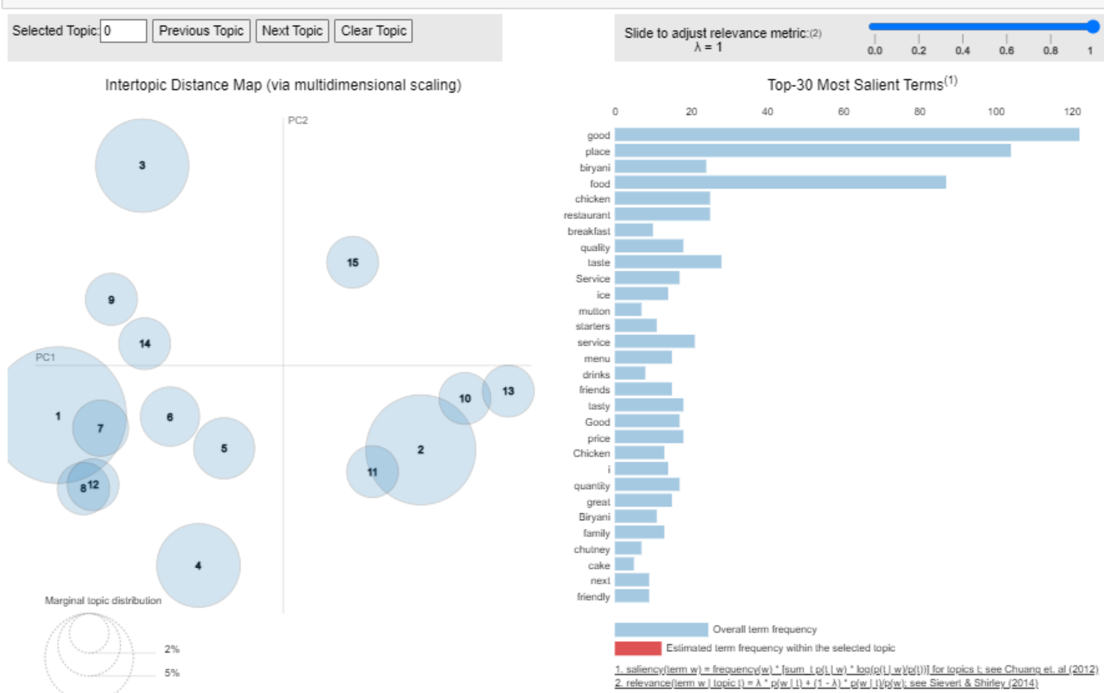
Out[93]:

	review	name	rating
191136	restaurant best north indian cuisines other re...	Khaja Point	5.0
9022	Awesome place taste Banglore No tension vehicl...	Davanagere Benne Dose Hut	5.0
112985	Order Swiggy pop rs99 delicious egg rice chick...	Xian	5.0
112986	quick bite Serves Kulfi Sandwiches cheese omel...	Kulfi Point	5.0
1291088	place amazing clean ambience few Chinese resta...	NISO Chinese Restaurant	5.0
...	...	...	...
33959	Amazing place Saturday nights little food good...	Thirsty Tiger	4.0
33956	cool place lunch dinner Good raspy Arabic roll...	Dhe Chef Cafe	4.0
33905	Veg Crunchy Cheese Sandwich super awesome gene...	Aha Juice Bar	4.0
33794	place many times place quality food nice price...	Spice Taj	4.0
0	beautiful place inThe interiors Mughal era lig...	Jalsa	4.0

4150 rows × 3 columns

In [90]:
```python
pyLDAvis.display(vis)
```

Out[90]:



KNOWLEDGE BASED RECOMMENDER SYSTEM

This will be a simple recommender system that will perform the following tasks. Ask the user for her/his preferences of:

- Locality
- Cuisines
- Budget for restaurant

In [39]:
```
df_zomato = df_zomato.rename(columns={'approx_cost(for two people)':'cost'})
```

In [40]:
```
def find_resturants(df,locality,min_budget,max_budget,cuisine):
    #Define a new rest variable to store the preferred rest. Copy the contents of df to rest
    rest = df.copy()
    percentile=0.8
    #Filter based on the condition
    rest = rest[(rest['location'] == locality) &
                    (rest['cost'] >= min_budget) &
                        (rest['cost'] <= max_budget)]

    rest=rest[rest.cuisines.str.contains(cuisine)]

    if(len(rest)==0):
        print("No restaurants with this combination!")
        return rest
    else:
        #Compute the values of C and m for the filtered rest
        C = rest['rate'].mean()
        m = rest['votes'].quantile(percentile)

        #Only consider restaurants that have higher than m votes. Save this in a new dataframe m_rest
        m_rest = rest.copy().loc[rest['votes'] >= m]

        #Calculate score using the weighted avg formula
        m_rest['score'] = m_rest.apply(lambda x: (x['votes']/(x['votes']+m) * x['rate'])
                                  + (m/(m+x['rate']) * C)
                                  ,axis=1)

        #Sort restaurants in descending order of their scores
        m_rest = m_rest.sort_values('score', ascending=False)

        return m_rest
```

In [41]:
```
find_resturants(df_zomato,'Banashankari',600.0,800.0,'North Indian').head()
```

Out[41]:

	address	name	online_order	book_table	rate	votes	location	rest_type	dish_liked	cuisines	cost	reviews_list	menu_item	listed_in(type)	listed_in(cit
3462	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	Yes	Yes	4.1	804	Banashankari	Casual Dining	Pasta, Lunch Buffet, Paneer Lajawab, Masala Pa...	North Indian, Mughlai, Chinese	800.0	[('Rated 4.0', 'RATED\n Super ambience\nGreat...	[]	Dine-out	Basavanagu
19401	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	Yes	Yes	4.1	783	Banashankari	Casual Dining	Pasta, Lunch Buffet, Masala Papad, Paneer Laja...	North Indian, Mughlai, Chinese	800.0	[('Rated 4.0', 'RATED\n A beautiful place to ...	[]	Buffet	Jayanag
20399	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	Yes	Yes	4.1	783	Banashankari	Casual Dining	Pasta, Lunch Buffet, Masala Papad, Paneer Laja...	North Indian, Mughlai, Chinese	800.0	[('Rated 4.0', 'RATED\n A beautiful place to ...	[]	Delivery	Jayanag
21302	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	Yes	Yes	4.1	783	Banashankari	Casual Dining	Pasta, Lunch Buffet, Masala Papad, Paneer	North Indian, Mughlai, Chinese	800.0	[('Rated 4.0', 'RATED\n A beautiful place to ...	[]	Dine-out	Jayanag

In [42]:
```
find_resturants(df_zomato,'Whitefield',500.0,800.0,'Chinese').head()
```

Out[42]:

	address	name	online_order	book_table	rate	votes	location	rest_type	dish_liked	cuisines	cost	reviews_list	menu_item	listed_in(type)	li
50507	46, Ramagondanahalli, Varthur Main Road, White...	Hyderabad Biryaani House	Yes	No	3.6	378	Whitefield	Casual Dining	Chicken Biryani, Mutton Biryani, Hyderabadi Bi...	Biryani, North Indian, Chinese	700.0	[('Rated 3.0', 'RATED\n I ordered veg biryani...	[]	Delivery	
51125	46, Ramagondanahalli, Varthur Main Road, White...	Hyderabad Biryaani House	Yes	No	3.6	378	Whitefield	Casual Dining	Chicken Biryani, Mutton Biryani, Hyderabadi Bi...	Biryani, North Indian, Chinese	700.0	[('Rated 3.0', 'RATED\n I ordered veg biryani...	[]	Dine-out	
50541	107, Praveen Transport Complex, Near ITPL Gate...	Alpha - House of Biryani & Tandoor	Yes	No	3.5	395	Whitefield	Casual Dining	Raita, Paneer Biryani	Biryani, North Indian, Chinese	800.0	[('Rated 4.0', "RATED\n Went to the place wit...	[]	Delivery	
51136	107, Praveen Transport Complex, Near ITPL Gate...	Alpha - House of Biryani & Tandoor	Yes	No	3.5	395	Whitefield	Casual Dining	Raita, Paneer Biryani	Biryani, North Indian, Chinese	800.0	[('Rated 4.0', "RATED\n Went to the place wit...	[]	Dine-out	

CONCLUSION AND EVALUATION METRICS:

We have thus created a recommender system capable of providing users with a variety of restaurant recommendations based on their search query, their constraints, requirements etc.

This model can be used as a plug in in a variety of online ordering platforms

For evaluation we take in user feedback since there is no data available with the ground truth and we consider the users need as the highest priority.