KONERU LAKSHMAIAH EDUCATION FOUNDATION
(Deemed to be university estd, u/s, 3 of the UGC Act, 1956)
(NAAC Accredited "A" Grade University)

PROJECT BASED LAB REPORT

On

RESTAURANT RECOMMENDATION SYSTEM

Submitted in partial fulfilment of the

Requirements for the award of the Degree of

Bachelor of Technology

In

Computer science and Engineering

Under the esteemed guidance of

Dr. S. Sagar Imambi

By

ID.No	Student Name
170030472	J.Lakshmi Sai Pushkala
170030561	Kushwanth Kapa
170031125	R.Naga Swetha Anogini

(DST-FIST Sponsored Department)

K L EDUCATION FOUNDATION

Green Fields, Vaddeswaram, Guntur District-522 502

2019-2020

K L EDUCATION FOUNDATION

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(DST-FIST Sponsored Department)

CERTIFICATE

This is to certify that this project based lab report entitled **"RESTAURANT RECOMMENDATION SYSTEM"** is a bonafide work done by J.Lakshmi Sai Pushkala(170030472), Kushwanth Kapa(170030561), R.Naga Swetha Anogini(170031125) in the course **Deep Learning**, **17CS3074** in partial fulfilment of the requirements for the award of Degree in Bachelor of Technology in **COMPUTER SCIENCEAND ENGNEERING** during the Even Semester of Academic year 2019-2020.

Faculty in Charge	**Head of the Department**
Dr. S. Sagar Imambi	Mr. V. Hari Kiran

K L EDUCATION FOUNDATION

DEPT OF COMPUTER SCINCE AND ENGINEERING

(DST-FIST Sponsored Department)

DECLARATION

We hereby declare that this project based lab report entitled **"RESTAURANT RECOMMENDATION SYSTEM"** has been prepared by us in the course **Deep Learning**, **17CS3074** in partial fulfilment of the requirement for the award of degree bachelor of technology in **COMPUTER SCIENCE AND ENGINEERING** during the Even Semester of the academic year 2019-2020.We also declare that this project-based lab report is of our own effort and it has not been submitted to any other university for the award of any degree.

Date: 20/04/2020

Place: Vaddeswarm

ID.NO	Student Name
170030472	J.Lakshmi Sai Pushkala
170030561	Kushwanth Kapa
170031125	R.Naga Swetha Anogini

ACKNOWLEDGEMENT

Our sincere thanks to ***Dr. S. Sagar Imambi*** in the Lab for her outstanding support throughout the project for the successful completion of the work.

We express our gratitude to ***Dr. S. Sagar Imambi*** the instructor and **Dr Pragnyaban Mishra** Course Co-ordinator for the course **Deep Learning**, **17CS3074** in the Computer Science and Engineering Department for providing us with adequate planning and support and means by which we can complete this project-based Lab.

We express our gratitude to **Mr. V. HARIKIRAN,** Head of the Department for computer science and Engineering for providing us with adequate facilities, ways and means by which we can complete this project-based Lab.

We would like to place on record the deep sense of gratitude to the honourable Vice Chancellor, K L University for providing the necessary facilities to carry the project-based Lab.

Last but not the least, we thank all Teaching and Non-Teaching Staff of our department and especially our classmates and our friends for their support in the completion of our project-based Lab.

ID.NO	**Student Name**
170030472	J.Lakshmi Sai Pushkala
170030561	Kushwanth Kapa
170031125	R.Naga Swetha Anogini

TABLE OF CONTENTS

CHAPTERS	PAGE NO
ABSTRACT	
CHAPTER 1: INTRODUCTION	1
1.1 INTRODUCTION	1
1.2 PROBLEM DEFINITION	1
1.3 SCOPE	2
1.4 PURPOSE	2
1.5 PROBLEM AND EXISTINGTECHNOLOGY	2
1.6 PROPOSED SYSTEM	3
CHAPTER 2: REQIUREMENTS & ANALYSIS	4
2.1 PLATFORM REQUIREMENTS	4
2.2 MODULE DESCRIPTION	4
CHAPTER 3: DESIGN & IMPLEMENTATION	7
3.1 ALGORITHMS	7
3.2 PSEUDO CODE	7
CHAPTER 4: SCREENSHOTS	10
CHAPTER 5: CONCLUSION	17
CHAPTER 6: REFERENCES	18

CHAPTER 1 : Introduction

1.1 Introduction

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). We can break down the large matrix of ratings from users and items into two smaller matrixes of user-feature and item-feature.

Recommender Systems or Recommendation Systems are simple algorithms that aim to provide the most relevant and accurate items (products, movies, events, articles, food, restaurants) to the user (customers, visitors, app users, readers) by filtering useful stuff from a huge pool of information base. Recommendation engines discover data patterns in the data set by learning consumers' choices and produces the outcomes that co-relates to their needs and interests. Our project is to develop similar recommendation system related to restaurants.

1.2 Problem Definition

Restaurant Recommendation System is a recommender system for restaurants based on user preferences. When user or customer requests for a specific type of restaurant, the system automatically recommends the restaurants that meet their requirements.

Our aim is to build a restaurant recommendation system that provides personalized restaurant recommendations to users. Since different people have different food preferences and dietary restrictions, we perform careful feature selection to take advantage of the information reflected in a user's reviews.

Furthermore, since dining is frequently a communal activity, we generated recommendations not just for an individual user, but also for a group of users. This system needs to consider the information of all individuals in a group and recommend restaurants that satisfies the group of users according to some criterion.

1.3 Scope

Restaurant Recommendation Systems has a wide scope in the future. At present, there are few food delivery applications which take help of the restaurant recommendation systems. There is a necessary scope in increasing the efficiency of the system.

The scope of the application is as that: The restaurants and hotels within the valley will be listed in the application.

1.4 Purpose

The main important purpose of the restaurant recommendation system is to help the users to visit and enjoy their favorite food in the best restaurants. For example, In the case of picking a restaurant to have lunch or dinner when you are traveling to other countries or cities, commonly you will ask your friend that lives in that country or city what is the best restaurant in town. The problem is you don’t have any friends that lived in that town. Your personal recommendations can be generated by an artificial friend: the recommender system.

This is the underlying purpose of the system to help the users in picking the best restaurants.

1.5 Problem And Existing Technology

While many existing recommender systems mainly target individuals, there is a remarkable increase of recommender systems which generate suggestions for groups. Some early systems were developed in a variety of domains, such as, group web page recommendation (Lieberman et al. 1999), *tour packages* for groups of tourists (Ardissono et al. 2003), *music tracks* and *playlists* for large groups of many listeners (Crossen et al. 2002), *movies* and *TV programs* for friends and family (O’Connor et al. 2001; Yu et al. 2006). Group scenarios are especially popular in the *food domain* in which a group of family members, friends or colleagues wants to make a party or simply have a meal together. However, the complexity significantly increases when food recommender systems need to take into account the preferences of all group members and strategies for achieving the consensus within group members.

1.6 Proposed System

In this system, we develop a restaurant recommendation system using the Latent Factor Collaborative Filtering Optimization. This system recommends restaurants for users or group of users based on their preferences such as beautiful ambience, good food, tasteful desserts and so.Our system provides personalized restaurant recommendations to users.

CHAPTER 2 : REQIUREMENTS & ANALYSIS

2.1 Platform Requirements

Hardware/Software	Hardware/Software element	Specification Version
Hardware	Processor	Intel core to duo
	RAM	1 GB
	Hard Disk	100 GB
Software	OS	Windows 10
	Python and Jupyter Notebook	

2.2 Module Description

Recommender Systems or Recommendation Systems are simple algorithms that aim to provide the most relevant and accurate items (products, movies, events, articles) to the user (customers, visitors, app users, readers) by filtering useful stuff from a huge pool of information base.

Recommendation engines discover data patterns in the data set by learning consumers' choices and produces the outcomes that co-relates to their needs and interests.

Unlike offline stores, online stores have no sales people, they have huge number of products on their websites and users on other hand have limited time and patience to navigate to the items that they are looking for. Recommendation systems solve these kind of problems by exploiting the user preferences and prioritize the items based on all other users past behaviour.

There are two types of Recommendation Systems

1. **Content-Based Filtering**
   Content-based filtering refers to such methods that provide recommendations by comparing representations of content describing an item to representations of content that interest the user pairs. This systems recommends based on comparison between the descriptors of the items and a user profile.

2. **Collaborative Filtering**
   Unlike content-based filtering, this system doesn't require description of the data hence it recommends without knowing anything about the products.

   Collaborative filtering is the type of recommendation algorithm that bases its predictions and recommendations on the rating or behavior of other users in the system. The fundamental idea of collaborative filtering is to find other users in the community that share opinions.

   There are two popular approaches of collaborative filtering:

   A. **User-based approach**

   Food Recommendation System uses the user ratings of other users with similar preferences to recommend a food item to a certain user. User-based recommendation algorithms firstly identify the k most similar users to the active user in which each user is treated as a vector and the similarities between the active user and other users are computed.

   B. **Item-based approach**

   Though user- based approach is useful, it suffers from the scalability problem as the user base grows. Searching from the neighbors of a user becomes time-consuming. To extend collaborative filtering to the large user base, a more scalable version of collaborative filtering, the i.e. item based approach was introduced.

   The overall structure of this approach seems to be similar to that of content based approach to recommendation and personalization, but item similarity is deduced from user preference patterns rather than extracted from the item data.

Matrix Factorization or Latent Factor Collaborative Filtering

Matrix factorization is a class of collaborative filtering algorithms used in recommender systems. Matrix factorization algorithms work by decomposing the user-item interaction matrix into the product of two lower dimensionality rectangular matrices.

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). We can break down the large matrix of ratings from users and items into two smaller matrixes of user-feature and item-feature.

We have two matrixes (user-features, business-features) that we can multiply to predict the ratings that a user gives to a restaurant. Later, we need to update the values in the features of our two matrixes according to the Error.

To optimize the predictions we need to calculate the error using the function below.

$$\min_{P,Q} \sum_{(i,x)\in R} (r_{xi} - q_i \cdot p_x^T)^2$$

Given P is the users-features matrix and Q is the business-features matrix. If we subtract the real ratings (r) with the predicted ratings (P.Q) and we square it, we get the LSE(Least Square Error).

To avoid overfitting we have to add regularization to our LSE formula and it will become the formula written below.

$$\min_{P,Q} \underbrace{\sum_{training} (r_{xi} - q_i p_x^T)^2}_{\text{"error"}} + \lambda \underbrace{\left[\sum_x \|p_x\|^2 + \sum_i \|q_i\|^2 \right]}_{\text{"length"}}$$

We apply the equation to minimize the error using Gradient Decent to update the values of each feature in matrix P and matrix Q.

CHAPTER 3 : DESIGN & IMPLEMENTATION

3.1 Algorithms

Matrix factorization algorithms work by decomposing the original matrix into two matrices one is the upper triangle (U), and the other is the lower triangle (L).

Algorithm:

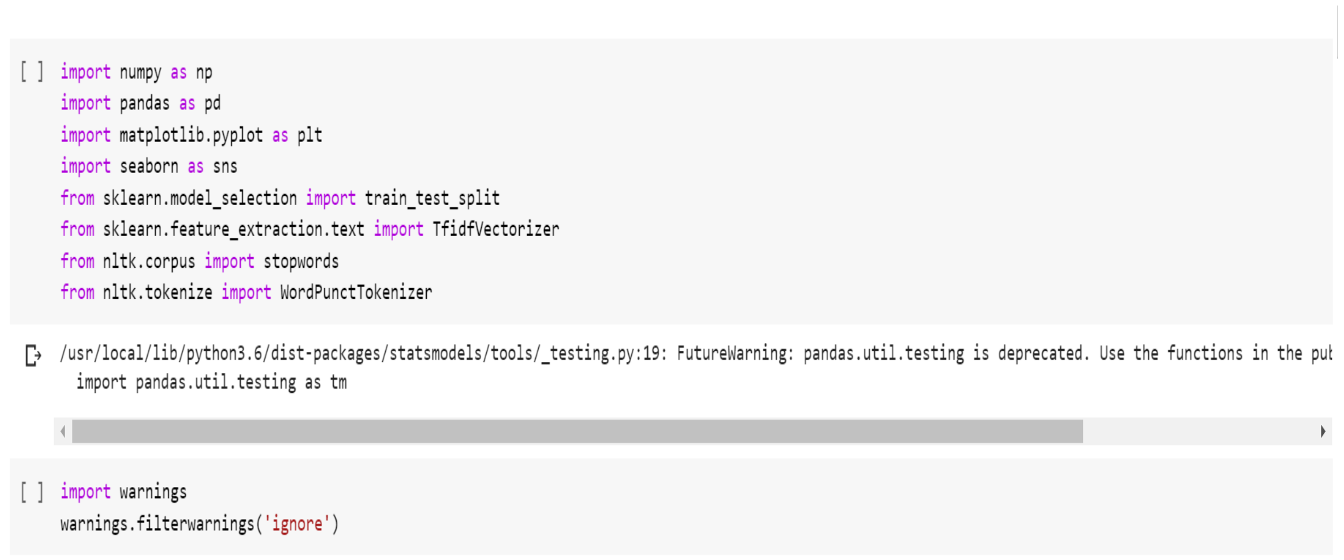
```
Step 1 : Take the input matrix as U and an empty diagonal matrix as L

Step 2: n be length of matrix U

Step 3: for col in range(1,n):

            a.   for row in range(2,n):

                     a-1. if row > col

                              mul = U[[row,col]]/U[[col,col]]

                              L[row,col] = mul

                              U[row,] = U[row,] – mul * U[col,]

            return U
```

3.2 Pseudo Code

```
def text_process(mess):
    """
    Takes in a string of text, then performs the following:
    1. Remove all punctuation
    2. Remove all stopwords
    3. Returns a list of the cleaned text
    """
    # Check characters to see if they are in punctuation
    nopunc = [char for char in mess if char not in string.punctuation]

    # Join the characters again to form the string.
    nopunc = ''.join(nopunc)
    # Now just remove any stopwords
    return " ".join([word for word in nopunc.split() if word.lower() not in stop])
yelp_data['text'] = yelp_data['text'].apply(text_process)


#userid vectorizer
userid_vectorizer = TfidfVectorizer(tokenizer = WordPunctTokenizer().tokenize, max_features=
5000)
userid_vectors = userid_vectorizer.fit_transform(userid_df['text'])
userid_vectors.shape


#Business id vectorizer
businessid_vectorizer = TfidfVectorizer(tokenizer = WordPunctTokenizer().tokenize, max_featu
res=5000)
businessid_vectors = businessid_vectorizer.fit_transform(business_df['text'])
businessid_vectors.shape


#Matrix Factorization
def matrix_factorization(R, P, Q, steps=25, gamma=0.001,lamda=0.02):
    for step in range(steps):
        for i in R.index:
            for j in R.columns:
                if R.loc[i,j]>0:
                    eij=R.loc[i,j]-np.dot(P.loc[i],Q.loc[j])
                    P.loc[i]=P.loc[i]+gamma*(eij*Q.loc[j]-lamda*P.loc[i])
                    Q.loc[j]=Q.loc[j]+gamma*(eij*P.loc[i]-lamda*Q.loc[j])
        e=0
        for i in R.index:
            for j in R.columns:
                if R.loc[i,j]>0:
                    e= e + pow(R.loc[i,j]-
np.dot(P.loc[i],Q.loc[j]),2)+lamda*(pow(np.linalg.norm(P.loc[i]),2)+pow(np.linalg.norm(Q.loc[j
]),2))
        if e<0.001:
            break

    return P,Q


#static input
words = "i want to have dinner with excellent dessert"
test_df= pd.DataFrame([words], columns=['text'])
```

```
test_df['text'] = test_df['text'].apply(text_process)
test_vectors = userid_vectorizer.transform(test_df['text'])
test_v_df = pd.DataFrame(test_vectors.toarray(), index=test_df.index, columns=userid_vectorize
r.get_feature_names())

predictItemRating=pd.DataFrame(np.dot(test_v_df.loc[0],Q.T),index=Q.index,columns=['Rating
'])
topRecommendations=pd.DataFrame.sort_values(predictItemRating,['Rating'],ascending=[0])[:7
]
for i in topRecommendations.index:
    print(df_business[df_business['business_id']==i]['name'].iloc[0])
    print(df_business[df_business['business_id']==i]['categories'].iloc[0])
    print(str(df_business[df_business['business_id']==i]['stars'].iloc[0])+ ' '+str(df_business[df_bus
iness['business_id']==i]['review_count'].iloc[0]))
    print('')


#dynamic input
words = input()
test_df= pd.DataFrame([words], columns=['text'])
test_df['text'] = test_df['text'].apply(text_process)
test_vectors = userid_vectorizer.transform(test_df['text'])
test_v_df = pd.DataFrame(test_vectors.toarray(), index=test_df.index, columns=userid_vectorize
r.get_feature_names())
predictItemRating=pd.DataFrame(np.dot(test_v_df.loc[0],Q.T),index=Q.index,columns=['Rating
'])
topRecommendations=pd.DataFrame.sort_values(predictItemRating,['Rating'],ascending=[0])[:7
]
for i in topRecommendations.index:
    print(df_business[df_business['business_id']==i]['name'].iloc[0])
    print(df_business[df_business['business_id']==i]['categories'].iloc[0])
    print(str(df_business[df_business['business_id']==i]['stars'].iloc[0])+ ' '+str(df_business[df_bus
iness['business_id']==i]['review_count'].iloc[0]))
    print('')
```

CHAPTER 4 : SCREENSHOTS

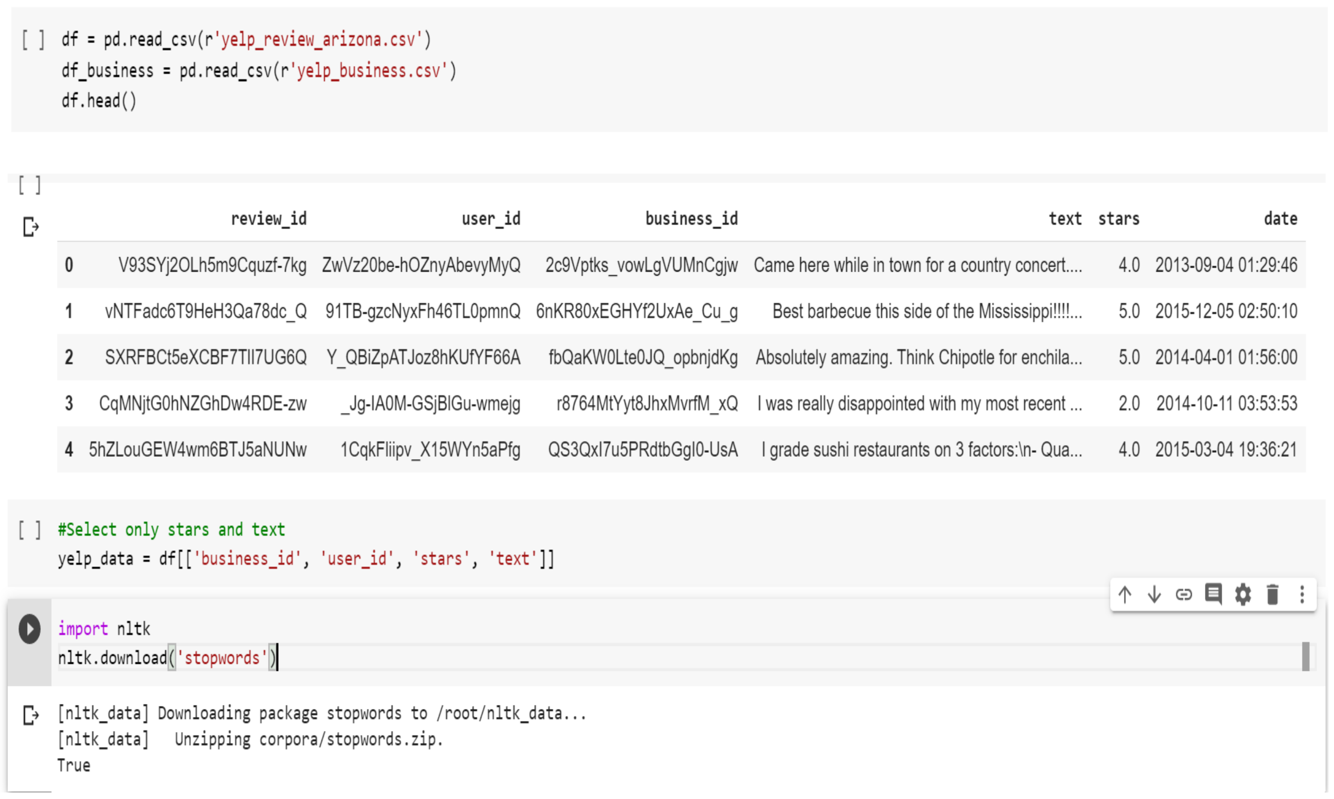


```
[ ] import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    import seaborn as sns
    from sklearn.model_selection import train_test_split
    from sklearn.feature_extraction.text import TfidfVectorizer
    from nltk.corpus import stopwords
    from nltk.tokenize import WordPunctTokenizer
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing is deprecated. Use the functions in the pu
  import pandas.util.testing as tm
```

```
[ ] import warnings
    warnings.filterwarnings('ignore')
```

Import Data



```
[ ] df = pd.read_csv(r'yelp_review_arizona.csv')
    df_business = pd.read_csv(r'yelp_business.csv')
    df.head()
```

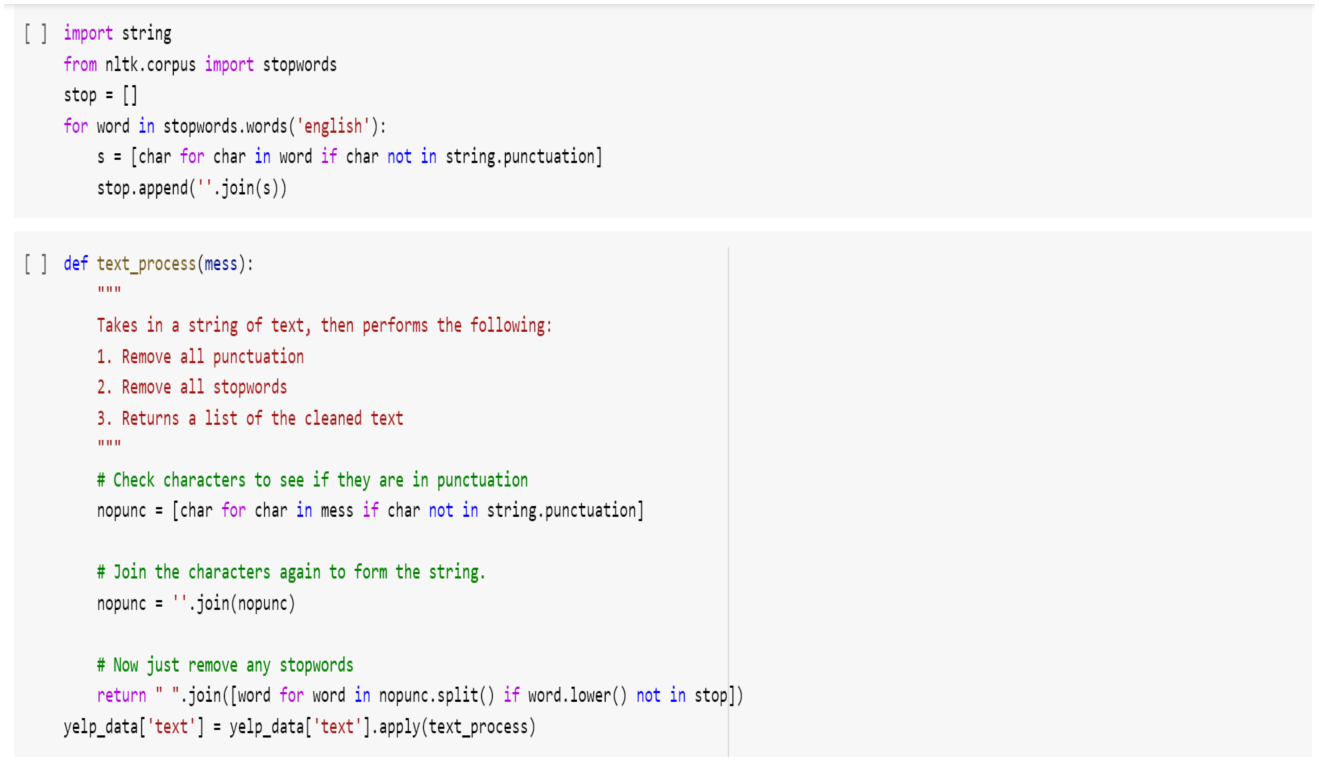
[ ]

	review_id	user_id	business_id	text	stars	date
0	V93SYj2OLh5m9Cquzf-7kg	ZwVz20be-hOZnyAbevyMyQ	2c9Vptks_vowLgVUMnCgjw	Came here while in town for a country concert....	4.0	2013-09-04 01:29:46
1	vNTFadc6T9HeH3Qa78dc_Q	91TB-gzcNyxFh46TL0pmnQ	6nKR80xEGHYf2UxAe_Cu_g	Best barbecue this side of the Mississippi!!!!...	5.0	2015-12-05 02:50:10
2	SXRFBCt5eXCBF7TlI7UG6Q	Y_QBiZpATJoz8hKUfYF66A	fbQaKW0Lte0JQ_opbnjdKg	Absolutely amazing. Think Chipotle for enchila...	5.0	2014-04-01 01:56:00
3	CqMNjtG0hNZGhDw4RDE-zw	_Jg-lA0M-GSjBlGu-wmejg	r8764MtYyt8JhxMvrfM_xQ	I was really disappointed with my most recent ...	2.0	2014-10-11 03:53:53
4	5hZLouGEW4wm6BTJ5aNUNw	1CqkFliipv_X15WYn5aPfg	QS3QxI7u5PRdtbGgI0-UsA	I grade sushi restaurants on 3 factors:\n- Qua...	4.0	2015-03-04 19:36:21

```
[ ] #Select only stars and text
    yelp_data = df[['business_id', 'user_id', 'stars', 'text']]
```

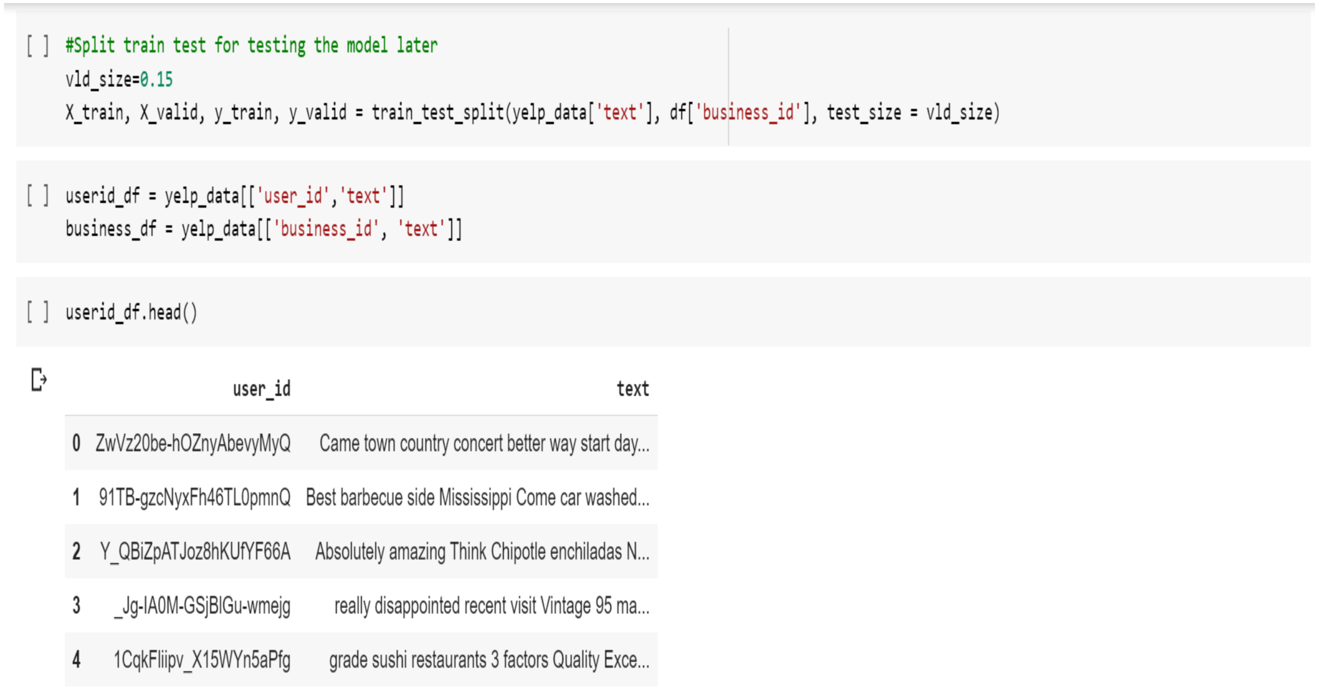
```
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```



```python
import string
from nltk.corpus import stopwords
stop = []
for word in stopwords.words('english'):
    s = [char for char in word if char not in string.punctuation]
    stop.append(''.join(s))
```

```python
def text_process(mess):
    """
    Takes in a string of text, then performs the following:
    1. Remove all punctuation
    2. Remove all stopwords
    3. Returns a list of the cleaned text
    """
    # Check characters to see if they are in punctuation
    nopunc = [char for char in mess if char not in string.punctuation]

    # Join the characters again to form the string.
    nopunc = ''.join(nopunc)

    # Now just remove any stopwords
    return " ".join([word for word in nopunc.split() if word.lower() not in stop])
yelp_data['text'] = yelp_data['text'].apply(text_process)
```

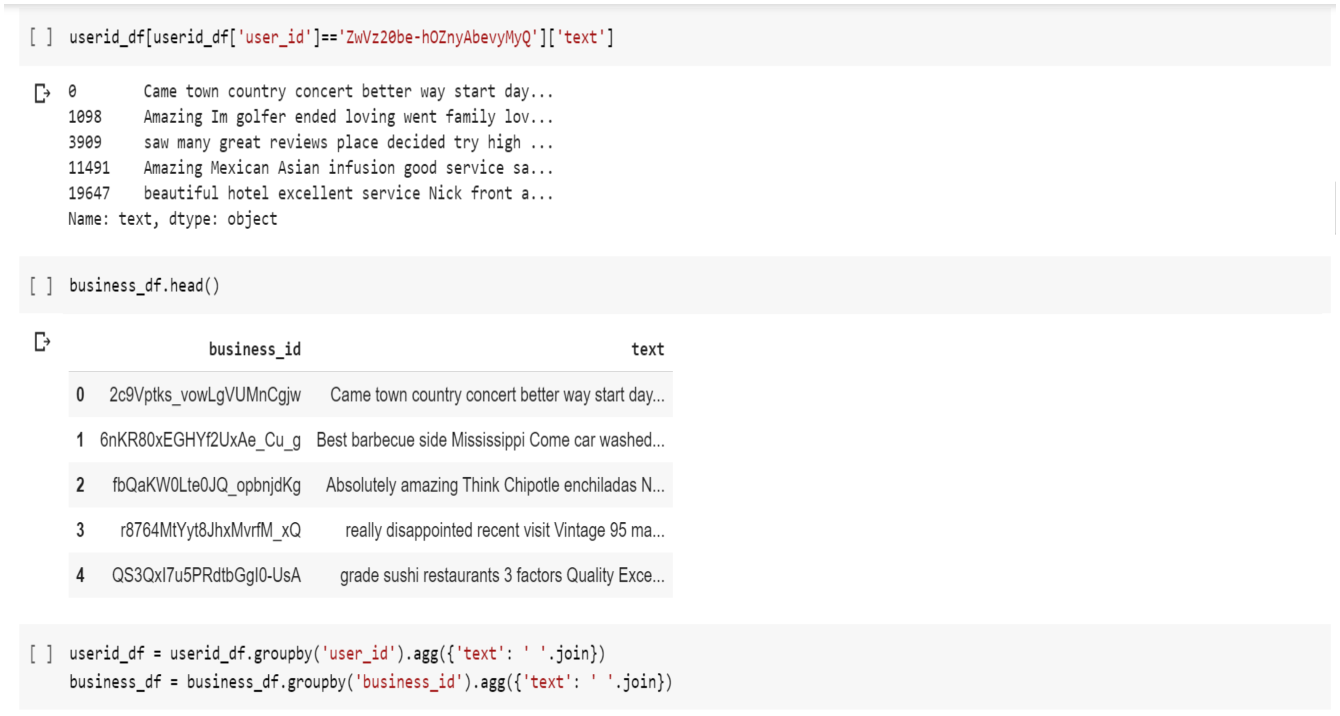


```python
#Split train test for testing the model later
vld_size=0.15
X_train, X_valid, y_train, y_valid = train_test_split(yelp_data['text'], df['business_id'], test_size = vld_size)
```

```python
userid_df = yelp_data[['user_id','text']]
business_df = yelp_data[['business_id', 'text']]
```

```python
userid_df.head()
```

	user_id	text
0	ZwVz20be-hOZnyAbevyMyQ	Came town country concert better way start day...
1	91TB-gzcNyxFh46TL0pmnQ	Best barbecue side Mississippi Come car washed...
2	Y_QBiZpATJoz8hKUfYF66A	Absolutely amazing Think Chipotle enchiladas N...
3	_Jg-lA0M-GSjBlGu-wmejg	really disappointed recent visit Vintage 95 ma...
4	1CqkFliipv_X15WYn5aPfg	grade sushi restaurants 3 factors Quality Exce...



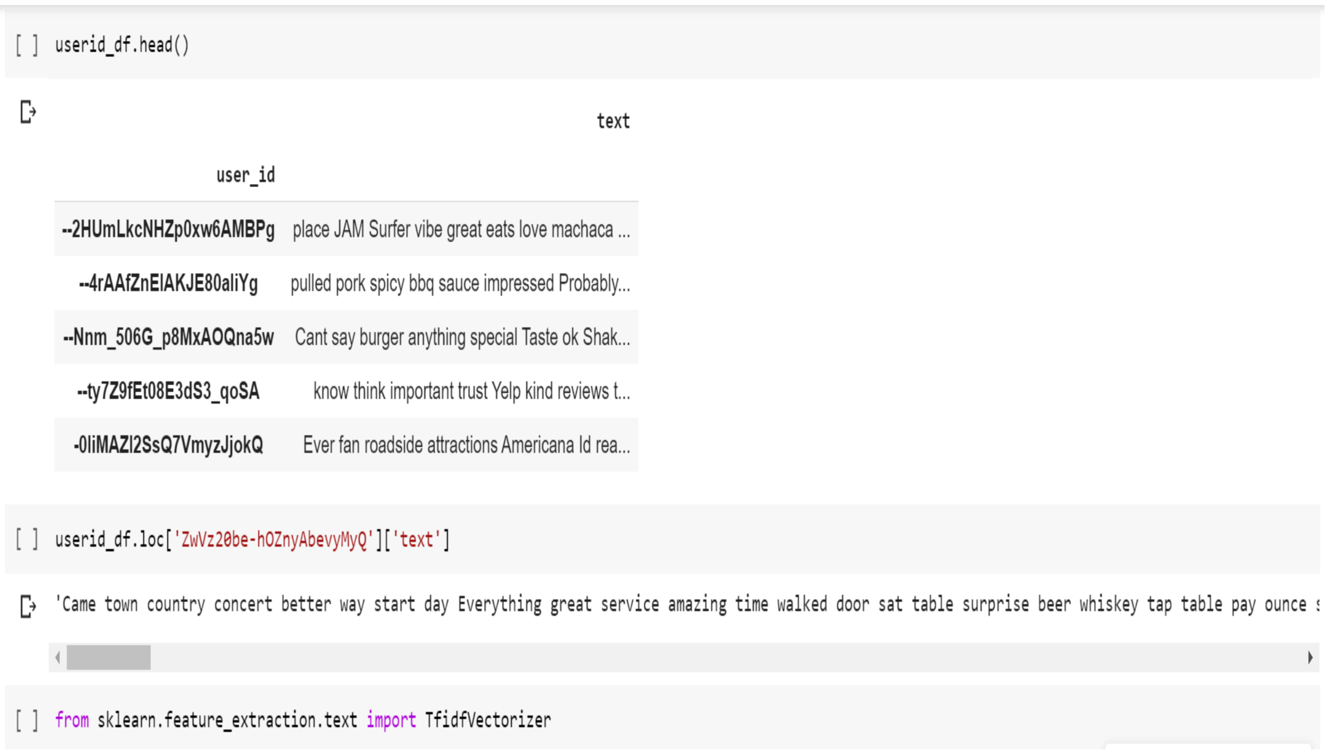
```
[ ] userid_df[userid_df['user_id']=='ZwVz20be-hOZnyAbevyMyQ']['text']
```

```
0        Came town country concert better way start day...
1098     Amazing Im golfer ended loving went family lov...
3909     saw many great reviews place decided try high ...
11491    Amazing Mexican Asian infusion good service sa...
19647    beautiful hotel excellent service Nick front a...
Name: text, dtype: object
```

```
[ ] business_df.head()
```

	business_id	text
0	2c9Vptks_vowLgVUMnCgjw	Came town country concert better way start day...
1	6nKR80xEGHYf2UxAe_Cu_g	Best barbecue side Mississippi Come car washed...
2	fbQaKW0Lte0JQ_opbnjdKg	Absolutely amazing Think Chipotle enchiladas N...
3	r8764MtYyt8JhxMvrfM_xQ	really disappointed recent visit Vintage 95 ma...
4	QS3QxI7u5PRdtbGgl0-UsA	grade sushi restaurants 3 factors Quality Exce...

```
[ ] userid_df = userid_df.groupby('user_id').agg({'text': ' '.join})
    business_df = business_df.groupby('business_id').agg({'text': ' '.join})
```



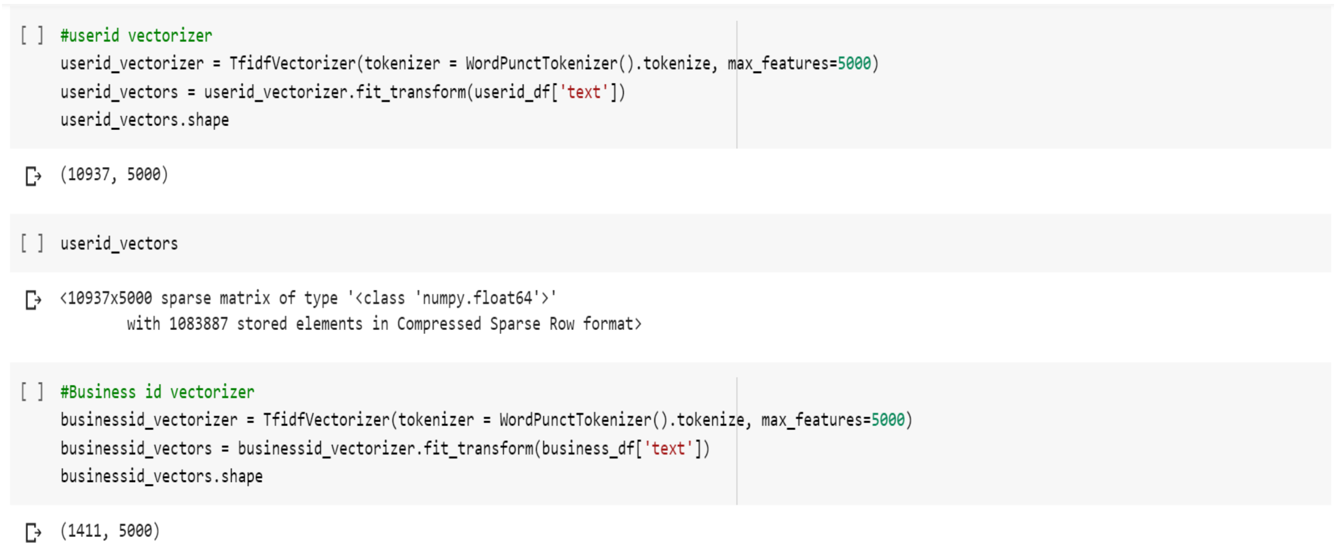
```
[ ] userid_df.head()
```

	text
user_id	
--2HUmLkcNHZp0xw6AMBPg	place JAM Surfer vibe great eats love machaca ...
--4rAAfZnEIAKJE80aliYg	pulled pork spicy bbq sauce impressed Probably...
--Nnm_506G_p8MxAOQna5w	Cant say burger anything special Taste ok Shak...
--ty7Z9fEt08E3dS3_qoSA	know think important trust Yelp kind reviews t...
-0liMAZI2SsQ7VmyzJjokQ	Ever fan roadside attractions Americana Id rea...

```
[ ] userid_df.loc['ZwVz20be-hOZnyAbevyMyQ']['text']
```

```
'Came town country concert better way start day Everything great service amazing time walked door sat table surprise beer whiskey tap table pay ounce
```

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



```
[ ] #userid vectorizer
    userid_vectorizer = TfidfVectorizer(tokenizer = WordPunctTokenizer().tokenize, max_features=5000)
    userid_vectors = userid_vectorizer.fit_transform(userid_df['text'])
    userid_vectors.shape
```

```
(10937, 5000)
```

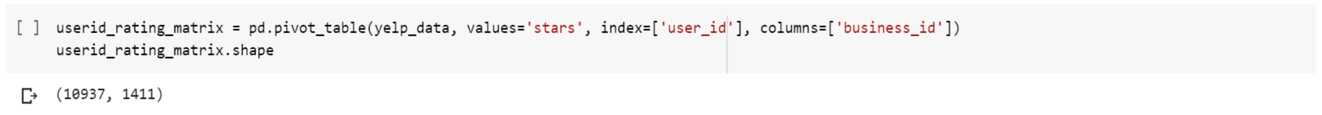
```
[ ] userid_vectors
```

```
<10937x5000 sparse matrix of type '<class 'numpy.float64'>'
        with 1083887 stored elements in Compressed Sparse Row format>
```

```
[ ] #Business id vectorizer
    businessid_vectorizer = TfidfVectorizer(tokenizer = WordPunctTokenizer().tokenize, max_features=5000)
    businessid_vectors = businessid_vectorizer.fit_transform(business_df['text'])
    businessid_vectors.shape
```

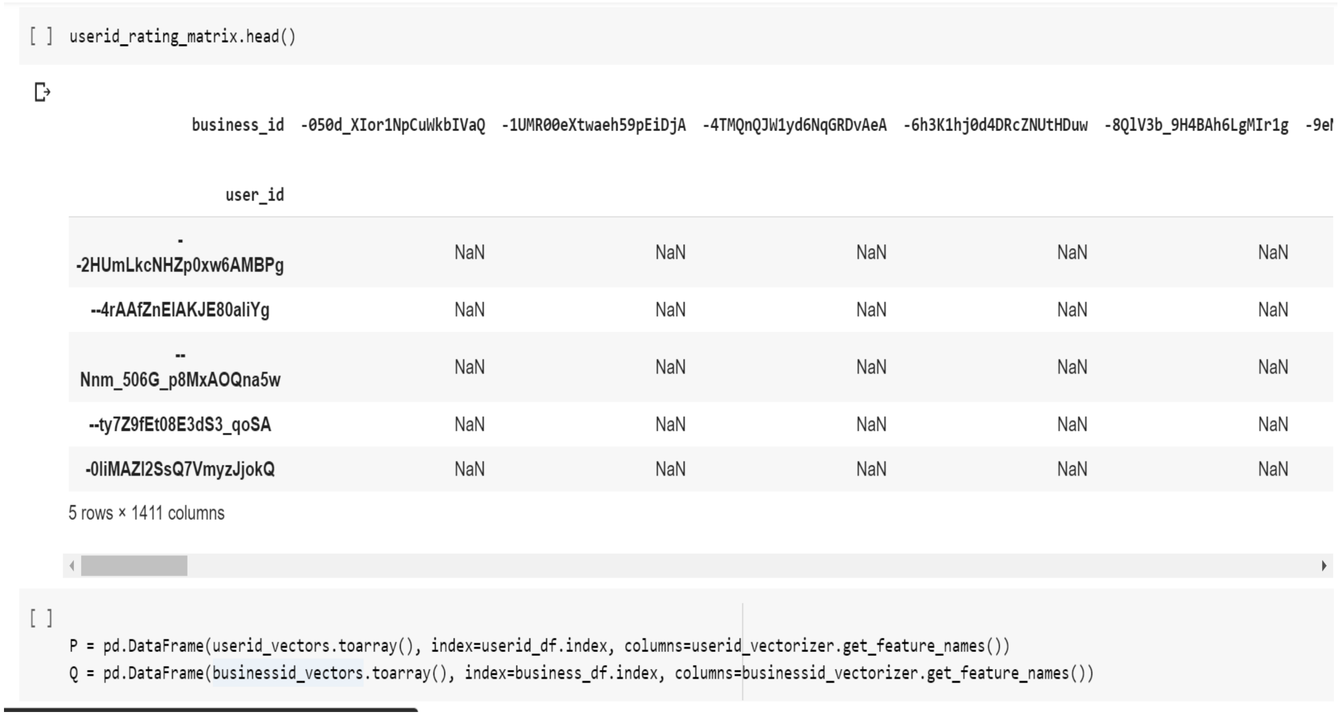
```
(1411, 5000)
```

Matrix Factorization



```
[ ] userid_rating_matrix = pd.pivot_table(yelp_data, values='stars', index=['user_id'], columns=['business_id'])
    userid_rating_matrix.shape
```

```
(10937, 1411)
```

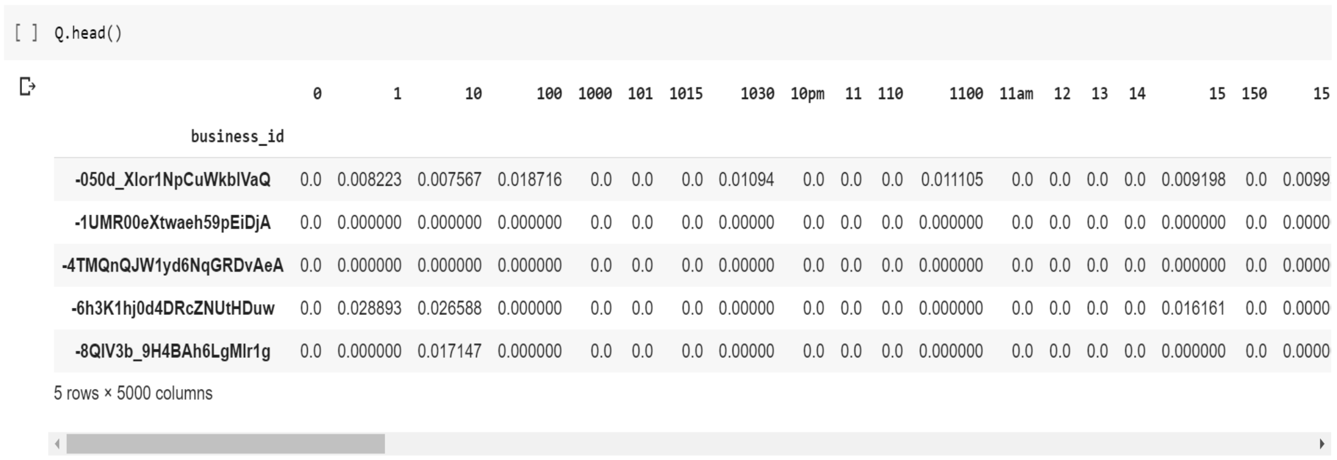


```
[ ] userid_rating_matrix.head()
```

business_id	-050d_XIor1NpCuWkbIVaQ	-1UMR00eXtwaeh59pEiDjA	-4TMQnQJW1yd6NqGRDvAeA	-6h3K1hj0d4DRcZNUtHDuw	-8QlV3b_9H4BAh6LgMIr1g	-9e
user_id						
--2HUmLkcNHZp0xw6AMBPg	NaN	NaN	NaN	NaN	NaN	
--4rAAfZnEIAKJE80aliYg	NaN	NaN	NaN	NaN	NaN	
--Nnm_506G_p8MxAOQna5w	NaN	NaN	NaN	NaN	NaN	
--ty7Z9fEt08E3dS3_qoSA	NaN	NaN	NaN	NaN	NaN	
-0liMAZl2SsQ7VmyzJjokQ	NaN	NaN	NaN	NaN	NaN	

5 rows × 1411 columns

```
[ ]
    P = pd.DataFrame(userid_vectors.toarray(), index=userid_df.index, columns=userid_vectorizer.get_feature_names())
    Q = pd.DataFrame(businessid_vectors.toarray(), index=business_df.index, columns=businessid_vectorizer.get_feature_names())
```

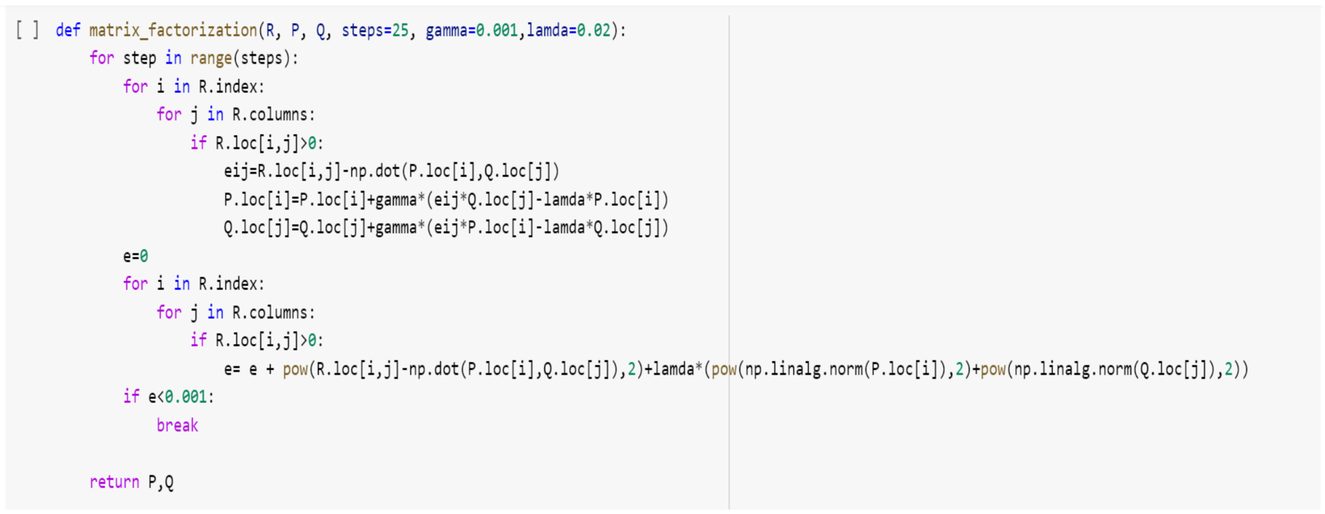


```
[ ] Q.head()
```

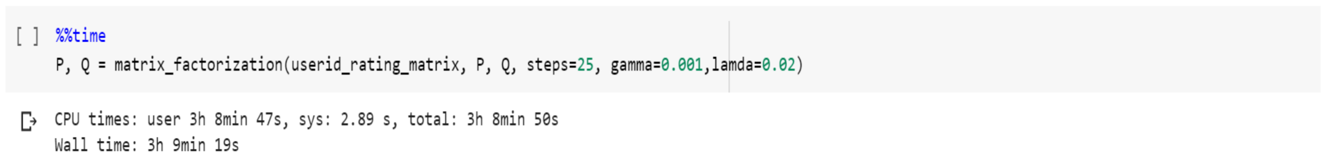
	0	1	10	100	1000	101	1015	1030	10pm	11	110	1100	11am	12	13	14	15	150	15
business_id																			
-050d_Xlor1NpCuWkblVaQ	0.0	0.008223	0.007567	0.018716	0.0	0.0	0.0	0.01094	0.0	0.0	0.0	0.011105	0.0	0.0	0.0	0.0	0.009198	0.0	0.0099
-1UMR00eXtwaeh59pEiDjA	0.0	0.000000	0.000000	0.000000	0.0	0.0	0.0	0.00000	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	0.0000
-4TMQnQJW1yd6NqGRDvAeA	0.0	0.000000	0.000000	0.000000	0.0	0.0	0.0	0.00000	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	0.0000
-6h3K1hj0d4DRcZNUtHDuw	0.0	0.028893	0.026588	0.000000	0.0	0.0	0.0	0.00000	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.016161	0.0	0.0000
-8QIV3b_9H4BAh6LgMIr1g	0.0	0.000000	0.017147	0.000000	0.0	0.0	0.0	0.00000	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	0.0000

5 rows × 5000 columns

Gradient Decent Optimization

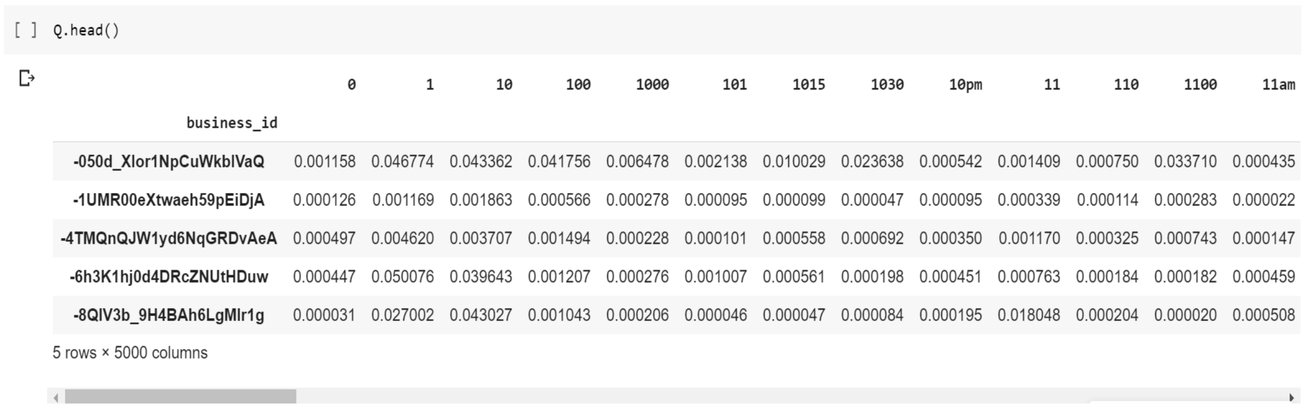


```
[ ] def matrix_factorization(R, P, Q, steps=25, gamma=0.001,lamda=0.02):
        for step in range(steps):
            for i in R.index:
                for j in R.columns:
                    if R.loc[i,j]>0:
                        eij=R.loc[i,j]-np.dot(P.loc[i],Q.loc[j])
                        P.loc[i]=P.loc[i]+gamma*(eij*Q.loc[j]-lamda*P.loc[i])
                        Q.loc[j]=Q.loc[j]+gamma*(eij*P.loc[i]-lamda*Q.loc[j])
            e=0
            for i in R.index:
                for j in R.columns:
                    if R.loc[i,j]>0:
                        e= e + pow(R.loc[i,j]-np.dot(P.loc[i],Q.loc[j]),2)+lamda*(pow(np.linalg.norm(P.loc[i]),2)+pow(np.linalg.norm(Q.loc[j]),2))
            if e<0.001:
                break

        return P,Q
```



```
[ ] %%time
    P, Q = matrix_factorization(userid_rating_matrix, P, Q, steps=25, gamma=0.001,lamda=0.02)
```

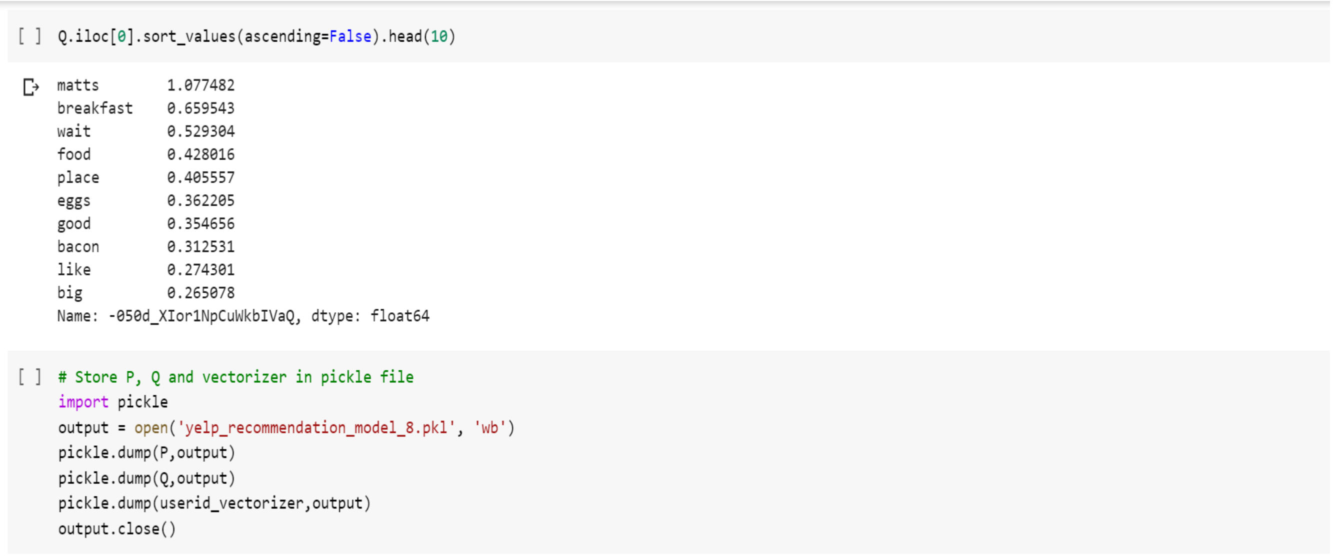
```
CPU times: user 3h 8min 47s, sys: 2.89 s, total: 3h 8min 50s
Wall time: 3h 9min 19s
```



```
[ ] Q.head()
```

	0	1	10	100	1000	101	1015	1030	10pm	11	110	1100	11am
business_id													
-050d_Xlor1NpCuWkblVaQ	0.001158	0.046774	0.043362	0.041756	0.006478	0.002138	0.010029	0.023638	0.000542	0.001409	0.000750	0.033710	0.000435
-1UMR00eXtwaeh59pEiDjA	0.000126	0.001169	0.001863	0.000566	0.000278	0.000095	0.000099	0.000047	0.000095	0.000339	0.000114	0.000283	0.000022
-4TMQnQJW1yd6NqGRDvAeA	0.000497	0.004620	0.003707	0.001494	0.000228	0.000101	0.000558	0.000692	0.000350	0.001170	0.000325	0.000743	0.000147
-6h3K1hj0d4DRcZNUtHDuw	0.000447	0.050076	0.039643	0.001207	0.000276	0.001007	0.000561	0.000198	0.000451	0.000763	0.000184	0.000182	0.000459
-8QlV3b_9H4BAh6LgMlr1g	0.000031	0.027002	0.043027	0.001043	0.000206	0.000046	0.000047	0.000084	0.000195	0.018048	0.000204	0.000020	0.000508

5 rows × 5000 columns

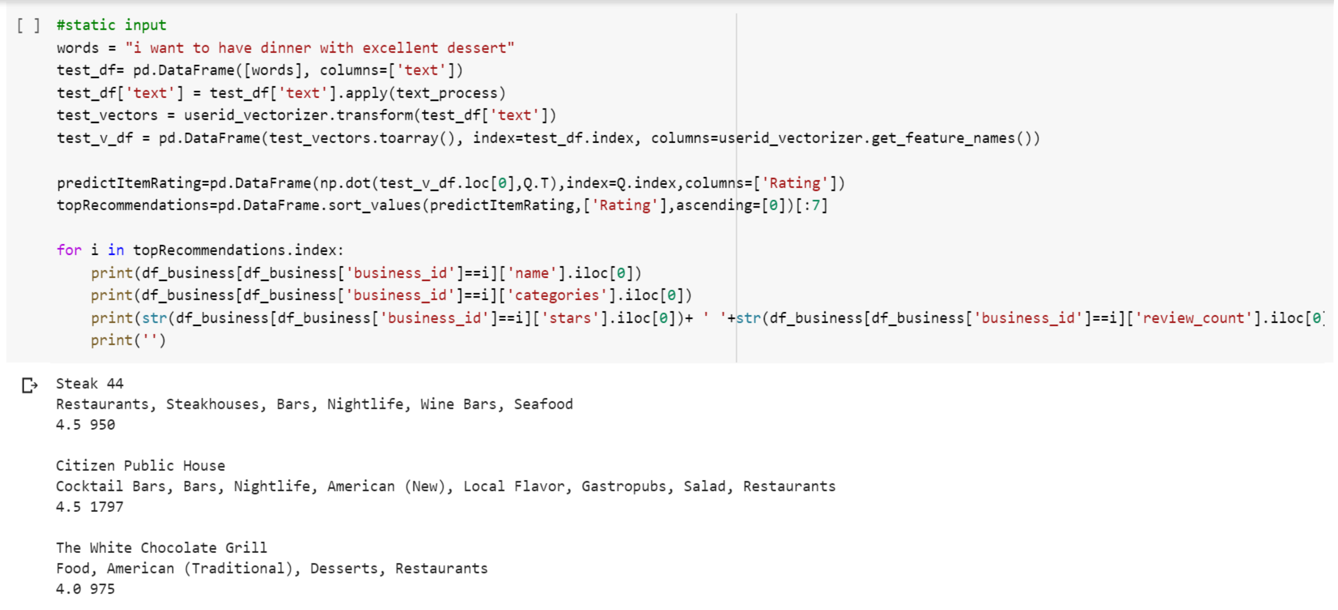


```
[ ] Q.iloc[0].sort_values(ascending=False).head(10)
```

```
matts        1.077482
breakfast    0.659543
wait         0.529304
food         0.428016
place        0.405557
eggs         0.362205
good         0.354656
bacon        0.312531
like         0.274301
big          0.265078
Name: -050d_XIor1NpCuWkbIVaQ, dtype: float64
```

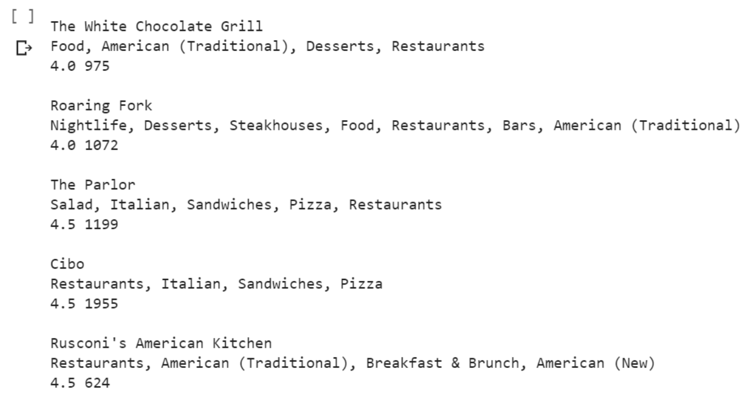
```
[ ] # Store P, Q and vectorizer in pickle file
    import pickle
    output = open('yelp_recommendation_model_8.pkl', 'wb')
    pickle.dump(P,output)
    pickle.dump(Q,output)
    pickle.dump(userid_vectorizer,output)
    output.close()
```

Prediction for input text

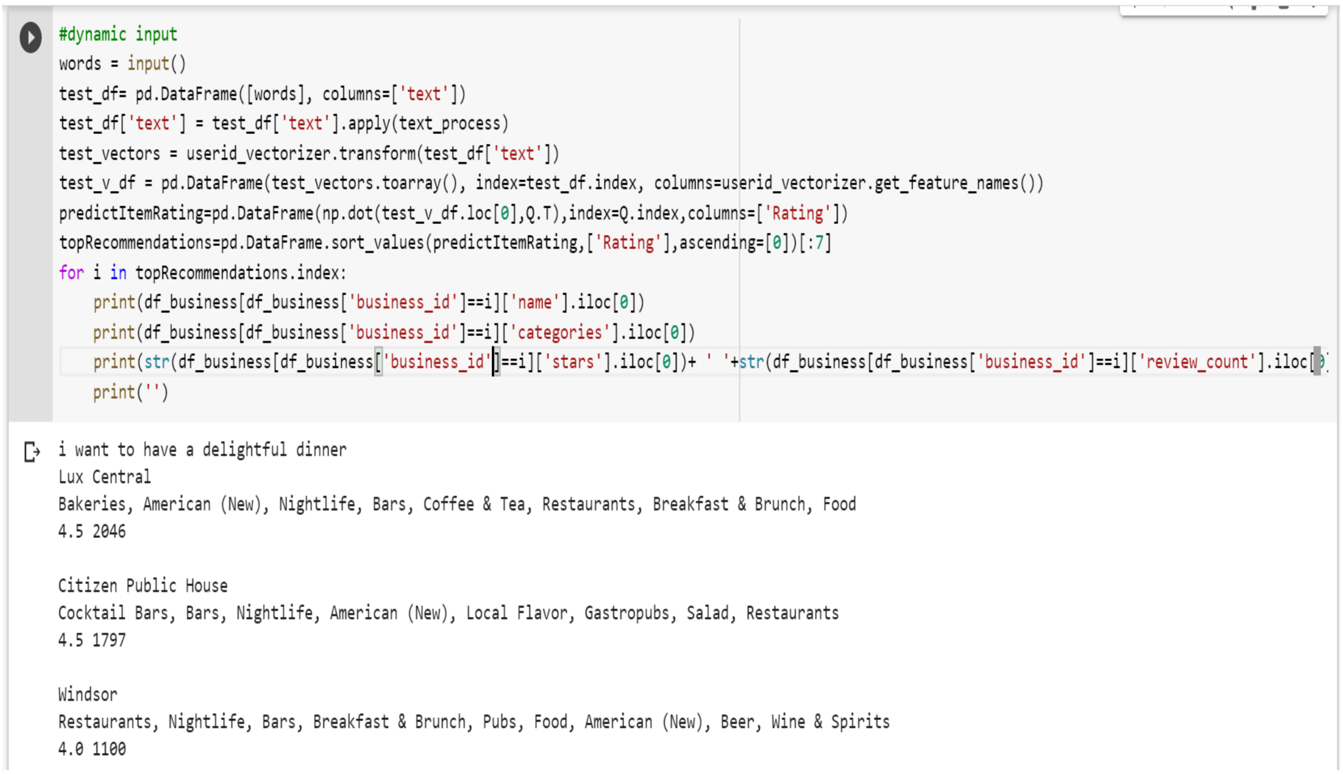


```
[ ] #static input
    words = "i want to have dinner with excellent dessert"
    test_df= pd.DataFrame([words], columns=['text'])
    test_df['text'] = test_df['text'].apply(text_process)
    test_vectors = userid_vectorizer.transform(test_df['text'])
    test_v_df = pd.DataFrame(test_vectors.toarray(), index=test_df.index, columns=userid_vectorizer.get_feature_names())

    predictItemRating=pd.DataFrame(np.dot(test_v_df.loc[0],Q.T),index=Q.index,columns=['Rating'])
    topRecommendations=pd.DataFrame.sort_values(predictItemRating,['Rating'],ascending=[0])[:7]

    for i in topRecommendations.index:
        print(df_business[df_business['business_id']==i]['name'].iloc[0])
        print(df_business[df_business['business_id']==i]['categories'].iloc[0])
        print(str(df_business[df_business['business_id']==i]['stars'].iloc[0])+ ' '+str(df_business[df_business['business_id']==i]['review_count'].iloc[0
        print('')
```

```
Steak 44
Restaurants, Steakhouses, Bars, Nightlife, Wine Bars, Seafood
4.5 950

Citizen Public House
Cocktail Bars, Bars, Nightlife, American (New), Local Flavor, Gastropubs, Salad, Restaurants
4.5 1797

The White Chocolate Grill
Food, American (Traditional), Desserts, Restaurants
4.0 975
```

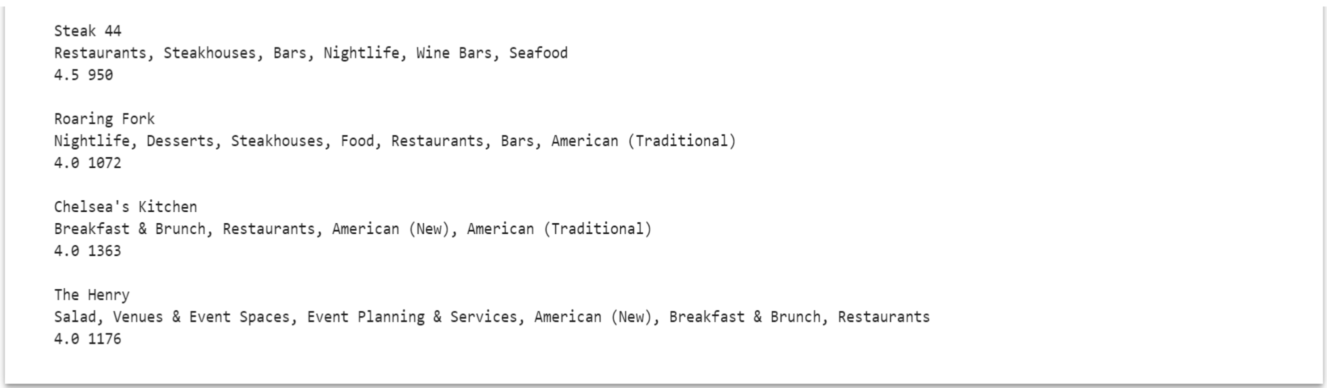


```
The White Chocolate Grill
Food, American (Traditional), Desserts, Restaurants
4.0 975

Roaring Fork
Nightlife, Desserts, Steakhouses, Food, Restaurants, Bars, American (Traditional)
4.0 1072

The Parlor
Salad, Italian, Sandwiches, Pizza, Restaurants
4.5 1199

Cibo
Restaurants, Italian, Sandwiches, Pizza
4.5 1955

Rusconi's American Kitchen
Restaurants, American (Traditional), Breakfast & Brunch, American (New)
4.5 624
```



```
#dynamic input
words = input()
test_df= pd.DataFrame([words], columns=['text'])
test_df['text'] = test_df['text'].apply(text_process)
test_vectors = userid_vectorizer.transform(test_df['text'])
test_v_df = pd.DataFrame(test_vectors.toarray(), index=test_df.index, columns=userid_vectorizer.get_feature_names())
predictItemRating=pd.DataFrame(np.dot(test_v_df.loc[0],Q.T),index=Q.index,columns=['Rating'])
topRecommendations=pd.DataFrame.sort_values(predictItemRating,['Rating'],ascending=[0])[:7]
for i in topRecommendations.index:
    print(df_business[df_business['business_id']==i]['name'].iloc[0])
    print(df_business[df_business['business_id']==i]['categories'].iloc[0])
    print(str(df_business[df_business['business_id']==i]['stars'].iloc[0])+ ' '+str(df_business[df_business['business_id']==i]['review_count'].iloc[0
    print('')
```

```
i want to have a delightful dinner
Lux Central
Bakeries, American (New), Nightlife, Bars, Coffee & Tea, Restaurants, Breakfast & Brunch, Food
4.5 2046

Citizen Public House
Cocktail Bars, Bars, Nightlife, American (New), Local Flavor, Gastropubs, Salad, Restaurants
4.5 1797

Windsor
Restaurants, Nightlife, Bars, Breakfast & Brunch, Pubs, Food, American (New), Beer, Wine & Spirits
4.0 1100
```



```
Steak 44
Restaurants, Steakhouses, Bars, Nightlife, Wine Bars, Seafood
4.5 950

Roaring Fork
Nightlife, Desserts, Steakhouses, Food, Restaurants, Bars, American (Traditional)
4.0 1072

Chelsea's Kitchen
Breakfast & Brunch, Restaurants, American (New), American (Traditional)
4.0 1363

The Henry
Salad, Venues & Event Spaces, Event Planning & Services, American (New), Breakfast & Brunch, Restaurants
4.0 1176
```

CHAPTER 5 : CONCLUSION

The project Restaurant Recommendation System was successfully completed by using latent factor collaborative filtering or matrix factorization.

In this project, we developed a model that could recommend the restaurant based on the choice of your interest.

CHAPTER 6 : REFERENCES

- https://towardsdatascience.com/various-implementations-of-collaborative-filtering-100385c6dfe0
- https://medium.com/@sumith.gannarapu/restaurant-recommendation-system-b52911d1ed0b
- https://towardsdatascience.com/recommender-systems-in-practice-cef9033bb23a