# Project - Building a Restaurant Recommendation System

**Link for the submission -** [https://drive.google.com/drive/folders/1yG_sHXO9HluqhL6LU4PV8A-JLWJOIbPh?usp=sharing](https://drive.google.com/drive/folders/1yG_sHXO9HluqhL6LU4PV8A-JLWJOIbPh?usp=sharing)

## Team - Flipping - a - Coin

### 1. Devansh Shrestha

### 2. Lakshay Chawla

### 3. Rommel Jalasutram

### 4. Varnika Vatsyayan

### Aim - To build a restaurant recommendation based on yelp dataset using various techniques.

# Introduction -

**We got our datasetr from yelp. The dataset contains 5 tables namely**

***1.Business***

***2.User***

***3.Review***

***4.Checkin***

***5.Tip***

**Reading the dataset from a tar file , and then converting it to CSV files.**

**We performed EDA on the dataset and then came up with a plan of action.**

**Firstly, we used item-item collaborative based filtering using cosine similarity.**

**Second, we used embeddings of latent factors using stoichastic gradient descent.**

**Last but not the least, we implemented a hybrid of the above two.**

## Libraries to be used

In [6]:

```
import tarfile
import json
import pandas as pd
import os
import pandas as pd
```

```python
import numpy as np
import os
import seaborn as sns
import matplotlib.pyplot as plt
from prettytable import PrettyTable
from pywaffle import Waffle
from scipy import spatial
import tensorflow as tf
from sklearn.model_selection import train_test_split
from tensorflow.keras.layers import Input, Embedding, Add, Dot, Flatten,Reshape,Dense,Con
catenate
from tensorflow.keras import backend as K
from tensorflow.keras.layers import Activation
from tensorflow.keras import Model
from tensorflow.keras.regularizers import l2
from tensorflow.keras.optimizers import Adam
from tensorflow.keras import utils
from sklearn import  preprocessing
from IPython.display import Image
```

## Extracting the files

In [8]:

```python
%pwd
```

Out[8]:

```
'C:\\Users\\laksh\\Downloads\\CODES\\Course\\DS-1\\Project'
```

In [ ]:

```python
# open file
file = tarfile.open('yelp_photos.tar')

# print file names
print(file.getnames())

# extract files
file.extractall('yelp_photos')

# close file
file.close()
```

In [ ]:

```python
for i, walk_items in enumerate(os.walk(dataset_path)):
    for j in walk_items[2]:
        if j.split('.')[-1] == 'json':
            temp = [json.loads(line) for line in open(f'{j}','rb')]
            df = pd.DataFrame.from_dict(temp)
            df.to_csv(f"{j.split('.')[-2].split('_')[-1]}.csv")
    break
```

In [2]:

```python
cd C:\\Users\\laksh\\Downloads\\CODES\\Course\\DS-1\\Project\\csv
```

```
C:\Users\laksh\Downloads\CODES\Course\DS-1\Project\csv
```

In [16]:

```python
## Setting the dataset path.
dataset_path = %pwd
```

## EDA and Pre-processing

In [18]:

```
## Reading all the csv files
for i, walk_items in enumerate(os.walk(dataset_path)):
    for j in walk_items[2]:
        globals()[j.split('.')[0]] = pd.read_csv(f'{j}', index_col=0)
```

In [19]:

```
dataframes = [business, checkin, review, tip, user]
names = ['Business', 'Checkin', 'Review', 'Tip', 'User']
```

In [20]:

```
pt = PrettyTable()
pt.field_names = ["Table Name","Shape"]
for i in range(len(names)):
    pt.add_row([names[i],dataframes[i].shape])
print(pt)
```

```
+------------+---------------+
| Table Name |     Shape     |
+------------+---------------+
|  Business  |  (150346, 14) |
|  Checkin   |  (131930, 2)  |
|   Review   |  (6990280, 9) |
|    Tip     |  (908915, 5)  |
|    User    | (1987897, 22) |
+------------+---------------+
```

In [21]:

```
for j in range(len(names)):
    print(f'Null Values for table {names[j]}')
    pt = PrettyTable()
    pt.field_names = ["Column Name","Percentage of null values"]
    for i in range(len(dataframes[j].columns)):
        pt.add_row([(dataframes[j].isnull().astype(int).sum()/len(dataframes[j])).index[
i],round(((dataframes[j].isnull().astype(int).sum()/len(dataframes[j]))[i]*100),4)])
    print(pt)
    print(f'Datatypes of table {names[j]}')
    pt = PrettyTable()
    pt.field_names = ["Column Name","Data Type"]
    for i in range(len(dataframes[j].columns)):
        pt.add_row([dataframes[j].dtypes.index[i],dataframes[j].dtypes[i]])
    print(pt)
    print(f'Unique datatypes in table {names[j]}')
    temp = {}
    for i in dataframes[j].dtypes:
        if i in temp.keys():
            temp[i] = temp[i] + 1
        else:
            temp[i] = 1
    pt = PrettyTable()
    pt.field_names = ["Data Type","Number of Occurences"]
    for i,j in temp.items():
        pt.add_row([i,j])
    print(pt)
    print('--------------------------------------------------------------------------------'
)
```

```
Null Values for table Business
+--------------+---------------------------+
| Column Name  | Percentage of null values |
+--------------+---------------------------+
| business_id  |            0.0            |
|     name     |            0.0            |
|   address    |           3.4101          |
|     city     |            0.0            |
|    state     |            0.0            |
| postal_code  |           0.0486          |
|   latitude   |            0.0            |
|  longitude   |            0.0            |
```

```
|    stars     |           0.0            |
| review_count |           0.0            |
|   is_open    |           0.0            |
|  attributes  |          9.1416          |
|  categories  |          0.0685          |
|    hours     |         15.4464          |
+--------------+--------------------------+
Datatypes of table Business
+--------------+-----------+
| Column Name  | Data Type |
+--------------+-----------+
| business_id  |   object  |
|     name     |   object  |
|   address    |   object  |
|     city     |   object  |
|    state     |   object  |
| postal_code  |   object  |
|   latitude   |  float64  |
|  longitude   |  float64  |
|    stars     |  float64  |
| review_count |   int64   |
|   is_open    |   int64   |
|  attributes  |   object  |
|  categories  |   object  |
|    hours     |   object  |
+--------------+-----------+
Unique datatypes in table Business
+-----------+----------------------+
| Data Type | Number of Occurences |
+-----------+----------------------+
|   object  |          9           |
|  float64  |          3           |
|   int64   |          2           |
+-----------+----------------------+
----------------------------------------------------------------------------
Null Values for table Checkin
+-------------+---------------------------+
| Column Name | Percentage of null values |
+-------------+---------------------------+
| business_id |            0.0            |
|     date    |            0.0            |
+-------------+---------------------------+
Datatypes of table Checkin
+-------------+-----------+
| Column Name | Data Type |
+-------------+-----------+
| business_id |   object  |
|     date    |   object  |
+-------------+-----------+
Unique datatypes in table Checkin
+-----------+----------------------+
| Data Type | Number of Occurences |
+-----------+----------------------+
|   object  |          2           |
+-----------+----------------------+
----------------------------------------------------------------------------
Null Values for table Review
+-------------+---------------------------+
| Column Name | Percentage of null values |
+-------------+---------------------------+
|  review_id  |            0.0            |
|   user_id   |            0.0            |
| business_id |            0.0            |
|    stars    |            0.0            |
|    useful   |            0.0            |
|    funny    |            0.0            |
|     cool    |            0.0            |
|     text    |            0.0            |
|     date    |            0.0            |
+-------------+---------------------------+
Datatypes of table Review
+-------------+-----------+
```

```
| Column Name | Data Type |
+-------------+-----------+
|  review_id  |   object  |
|   user_id   |   object  |
| business_id |   object  |
|    stars    |  float64  |
|    useful   |   int64   |
|    funny    |   int64   |
|     cool    |   int64   |
|     text    |   object  |
|     date    |   object  |
+-------------+-----------+
Unique datatypes in table Review
+-----------+----------------------+
| Data Type | Number of Occurences |
+-----------+----------------------+
|   object  |          5           |
|  float64  |          1           |
|   int64   |          3           |
+-----------+----------------------+
----------------------------------------------------------------------------
Null Values for table Tip
+------------------+--------------------------+
|   Column Name    | Percentage of null values |
+------------------+--------------------------+
|     user_id      |           0.0            |
|   business_id    |           0.0            |
|       text       |          0.0006          |
|       date       |           0.0            |
| compliment_count |           0.0            |
+------------------+--------------------------+
Datatypes of table Tip
+------------------+-----------+
|   Column Name    | Data Type |
+------------------+-----------+
|     user_id      |   object  |
|   business_id    |   object  |
|       text       |   object  |
|       date       |   object  |
| compliment_count |   int64   |
+------------------+-----------+
Unique datatypes in table Tip
+-----------+----------------------+
| Data Type | Number of Occurences |
+-----------+----------------------+
|   object  |          4           |
|   int64   |          1           |
+-----------+----------------------+
----------------------------------------------------------------------------
Null Values for table User
+--------------------+--------------------------+
|    Column Name     | Percentage of null values |
+--------------------+--------------------------+
|      user_id       |           0.0            |
|        name        |          0.0004          |
|    review_count    |           0.0            |
|   yelping_since    |           0.0            |
|       useful       |           0.0            |
|       funny        |           0.0            |
|        cool        |           0.0            |
|       elite        |         95.4123          |
|      friends       |           0.0            |
|        fans        |           0.0            |
|   average_stars    |           0.0            |
|   compliment_hot   |           0.0            |
|  compliment_more   |           0.0            |
| compliment_profile |           0.0            |
|  compliment_cute   |           0.0            |
|  compliment_list   |           0.0            |
|  compliment_note   |           0.0            |
|  compliment_plain  |           0.0            |
|  compliment_cool   |           0.0            |
```

```
|  compliment_funny  |            0.0            |
| compliment_writer  |            0.0            |
| compliment_photos  |            0.0            |
+--------------------+---------------------------+
Datatypes of table User
+--------------------+-----------+
|    Column Name     | Data Type |
+--------------------+-----------+
|      user_id       |   object  |
|        name        |   object  |
|    review_count    |   int64   |
|   yelping_since    |   object  |
|       useful       |   int64   |
|       funny        |   int64   |
|        cool        |   int64   |
|       elite        |   object  |
|      friends       |   object  |
|        fans        |   int64   |
|   average_stars    |  float64  |
|   compliment_hot   |   int64   |
|  compliment_more   |   int64   |
| compliment_profile |   int64   |
|  compliment_cute   |   int64   |
|  compliment_list   |   int64   |
|  compliment_note   |   int64   |
|  compliment_plain  |   int64   |
|  compliment_cool   |   int64   |
|  compliment_funny  |   int64   |
| compliment_writer  |   int64   |
| compliment_photos  |   int64   |
+--------------------+-----------+
Unique datatypes in table User
+-----------+----------------------+
| Data Type | Number of Occurences |
+-----------+----------------------+
|   object  |          5           |
|   int64   |          16          |
|  float64  |          1           |
+-----------+----------------------+
----------------------------------------------------------------------------
```

### Table Business

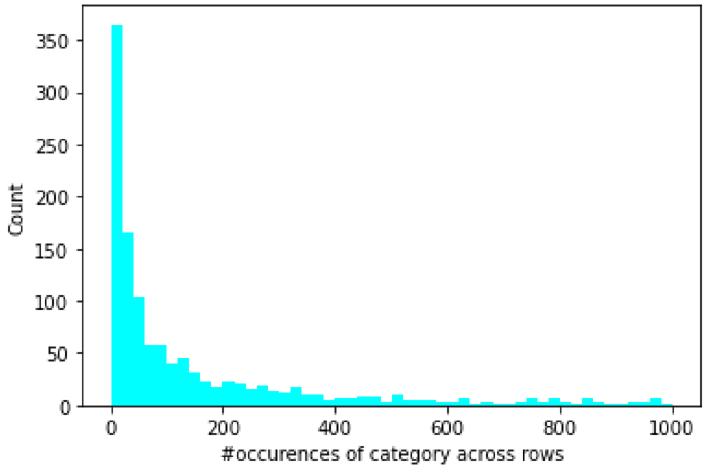
In [49]:

```
n_items = 100
```

In [28]:

```
# one hot encoding categories column
categories_columns={}
def split_categories(cat):
  row = str(cat).split(',')
  ret_row=[]
  for i in row:
    if i[0]==" ": # remove the blank space at begining of some strings
      i = i[1:]
    elif i[-1]==' ':
      i=i[:-1]
    ret_row.append(i)
    if i not in categories_columns.keys():
      categories_columns[i]=1
    else:
      categories_columns[i]+=1
  return ret_row

df_columns=business.categories.apply(split_categories)

# sort based on number of occurence
categories_columns=sorted(categories_columns.items(), key =lambda kv:[kv[1], kv[0]],reve
```

```
rse=True)

dt=np.dtype('object,int')
cat_items=np.array(categories_columns,dtype=dt)
plt.hist(cat_items['f1'],range=(0,1000),bins=50,color="cyan")
plt.ylabel('Count')
plt.xlabel('#occurences of category across rows')
plt.show()
print('A lot of categories occurs in few rows')
print('Considering top 100 categories')
categories_columns = cat_items['f0'][:100]
print('value of hundredth category "',cat_items['f0'][99],'":',cat_items['f1'][99])# one hot encoding categories column
categories_columns={}
```




```
A lot of categories occurs in few rows
Considering top 100 categories
value of hundredth category " Eyelash Service ": 1278
```

In [23]:

```
no_cols = len(categories_columns)
category_ohe = {}
for idx,val in df_columns.items():
  row = np.zeros(no_cols)
  row=row.astype('int')
  for i,v in  enumerate(categories_columns):
    if v in val:
      row[i]=int(1)
  category_ohe[idx]=row
```

In [24]:

```
cols_ohe=pd.DataFrame(category_ohe.values(),columns=categories_columns)
cols_ohe.head()
```

Out[24]:

| | Restaurants | Food | Shopping | Home Services | Beauty & Spas | Nightlife | Health & Medical | Local Services | Bars | Automotive | ... | Financial Services | Trainers | Sty |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | ... | 0 | 0 | |
| **1** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | ... | 0 | 0 | |
| **2** | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | |
| **3** | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | |
| **4** | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | |

**5 rows × 100 columns**

In [25]:

```python
# One hot encode State column
print("Unique states in df:",len(np.unique(business.state.values)))
df_state_ohe =pd.get_dummies(business[['state']])
```

```
Unique states in df: 27
```

In [26]:

```python
def get_attributes(s,cat_columns,index=None):
  field_dict={}
  zero_text = ['None','False','none','false','no']
  pars_table={ord('\''):None,ord('"'):None,ord('{'):ord(' '),ord('}'):None}
  # return None if already none
  if type(s) == float or type(s)== np.nan:
    return None

  # clean the  entry
  s=s.replace('"u\'','\'').replace(' u','').replace('{u','{')
  s=s.translate(pars_table)
  s=s.replace(' BusinessParking: ','').replace('  ','').replace('Ambience:','' )
  s=s.replace('GoodForMeal:','').replace('Music:','').replace('BestNights:','').replace(
'DietaryRestrictions:','')

  att=s.split(',') # split each attribute

  # get attribute values
  for field in att:
    if field =='None'or field=='':
      continue
    f_split=field.split(':')
    attri = f_split[0]
    val = f_split[1]
    if val in zero_text:
      val = 0
    elif val in ['True']:
      val = 1
    elif attri in cat_columns:
      val = str(val)
      field_dict[attri]=val
      continue

    try:
      val=int(val)
    except:
      print('Unknown type')
      print(val,index)
    # update dict

    field_dict[attri]=val

  return field_dict
```

In [27]:

```python
unique_attributes=[]
row_attr =[]
cat_columns=['RestaurantsAttire','Alcohol','NoiseLevel','HairSpecializesIn','WiFi','BYOBC
orkage','Smoking','AgesAllowed']
for index, row in business[['attributes']].iterrows():
  attr_dict=get_attributes(row['attributes'],cat_columns=cat_columns,index=index)
  if attr_dict == None:
    row_attr.append({})
    continue
  for k in list(attr_dict.keys()):
    if k not in unique_attributes:
      unique_attributes.append(k)
  row_attr.append(attr_dict)
```

In [29]:

```python
# create a dataframe from row attributes
```

```
df_attributes=pd.DataFrame(row_attr,columns=unique_attributes)
df_attributes=df_attributes.fillna(0)
```

In [30]:

```
# one hot encode categorical columns
df_attr_cat=pd.get_dummies(df_attributes[cat_columns])
df_attributes.drop(cat_columns,axis=1,inplace=True)
df_attributes=df_attributes.astype('int32')
#numerical_cols=df_attributes.columns
```

In [31]:

```
df_attributes=pd.concat([df_attributes,df_attr_cat],axis=1)
df_attributes.head()
```

Out[31]:

| | ByAppointmentOnly | BusinessAcceptsCreditCards | BikeParking | RestaurantsPriceRange2 | CoatCheck | RestaurantsTakeOut | I |
|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 0 | 0 | 0 | 0 | |
| **1** | 0 | 1 | 0 | 0 | 0 | 0 | |
| **2** | 0 | 1 | 1 | 2 | 0 | 0 | |
| **3** | 0 | 0 | 1 | 1 | 0 | 1 | |
| **4** | 0 | 1 | 1 | 0 | 0 | 1 | |

**5 rows × 104 columns**

In [32]:

```
business.drop(['attributes','categories','state'],axis=1,inplace=True)
```

In [34]:

```
df_final_buisness = pd.concat([business,df_attributes,cols_ohe,df_state_ohe],axis=1)
```

In [35]:

```
df_final_buisness.to_csv('df_pp_buisness.csv',index=False)
```

## Table User

In [ ]:

```
df_users=df_users[df_users['review_count']>2]
```

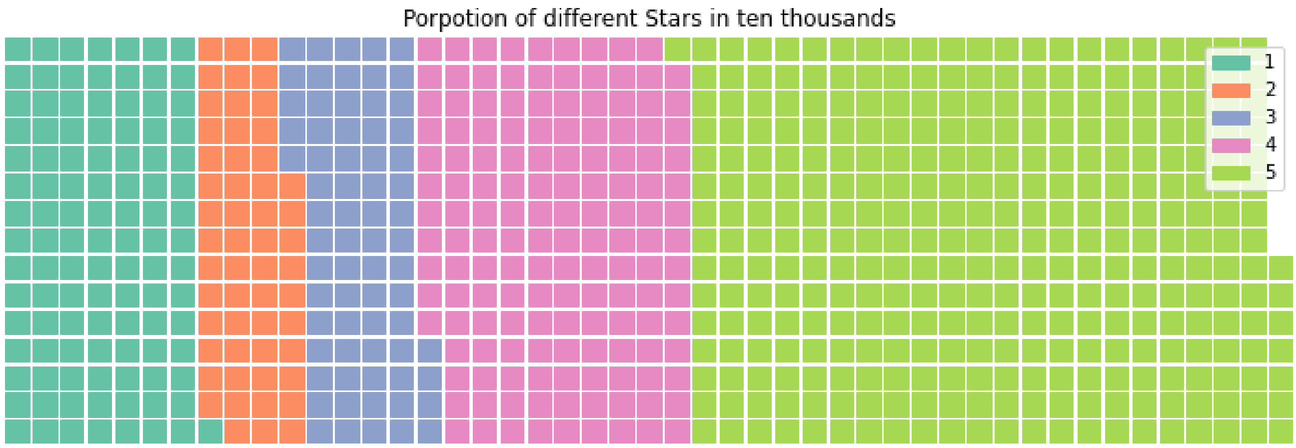
## Table Review

In [38]:

```
df_review['stars']=df_review['stars'].astype('int32')
```

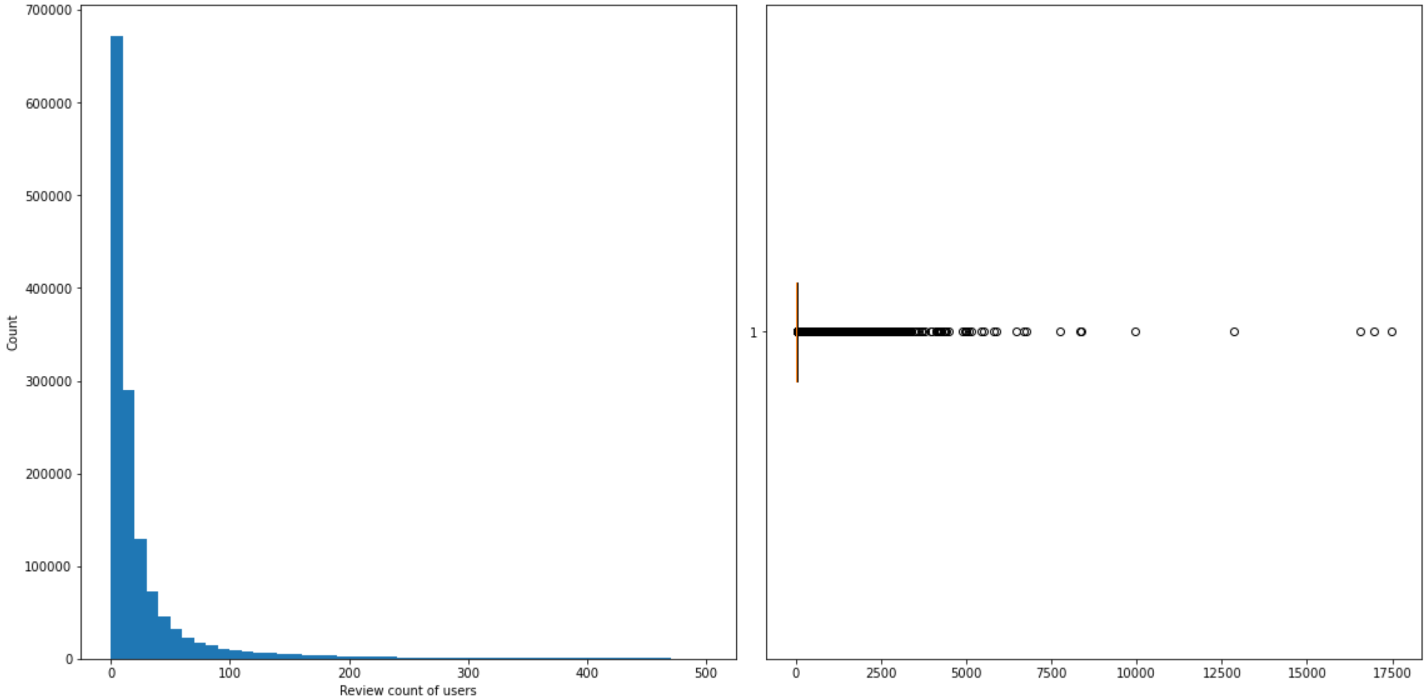
In [41]:

```
plt.rcParams["figure.figsize"] = [10, 10]
plt.rcParams["figure.autolayout"] = True

stars=df_review.groupby('stars')['stars'].count()
star_count=[(i//10000) for i in stars.values]
label=[str(i) for i in list(stars.index)]

plt.figure(FigureClass=Waffle,rows=15,values=star_count,labels=label)
plt.title('Porpotion of different Stars in ten thousands')
plt.show()
```




In [47]:

```
fig,ax=plt.subplots(1,2,figsize=(16,8))
ax[0].hist(df_users.review_count,bins=50,range=(0,500))
ax[0].set_ylabel('Count')
ax[0].set_xlabel('Review count of users')
ax[1].boxplot(df_users.review_count,vert=False)
plt.show()
```




In [51]:

```
most_rated_items=df_review.groupby(['business_id'])['business_id'].count().sort_values(a
scending=False)
print(f'considering top {n_items} most rated items')
```

```
considering top 100 most rated items
```

In [52]:

```
most_rated_items=most_rated_items.iloc[0:n_items].index
```

In [54]:

```
# create a df which contains top n items and respective user that reviewed along with rev
iew
user_item=df_review[df_review['business_id'].isin(most_rated_items)][['user_id','business
_id','stars']]
user_item.reset_index(drop=True,inplace=True)
stars_top_rated_items = user_item.groupby('business_id').mean('stars')
```

In [55]:

In [55]:

```
unique_items=user_item['business_id'].unique()
print('No of unique items:',len(unique_items))
```

```
No of unique items: 100
```

In [56]:

```
unique_user = user_item['user_id'].unique()
null_ui_mat={}
for i in unique_user:
  r = list(np.zeros(len(unique_items)))
  null_ui_mat[i]=r
```

In [57]:

```
UI_matrix = pd.DataFrame.from_dict(null_ui_mat,columns=unique_items,orient='index').asty
pe('int32')
```

In [58]:

```
UI_matrix.head()
```

Out[58]:

| | PY9GRfzr4nTZeINf346QOw | W4ZEKkva9HpAdZG88juwyQ | SZU9c8V2GuREDN5KgyHFJw | YvyVOK0k5 |
|---|---|---|---|---|
| **qEEk0PuoH1dVa619t8fgpw** | 0 | 0 | 0 | |
| **EBa-0-6AKoy6jziNexDJtg** | 0 | 0 | 0 | |
| **JYYYKt6TdVA4ng9lLcXt_g** | 0 | 0 | 0 | |
| **7P9w2PrP4ZcJyDFwch51lg** | 0 | 0 | 0 | |
| **pitYOVSsF8R1gWG1G0qxsA** | 0 | 0 | 0 | |

**5 rows × 100 columns**

In [59]:

```
UI_matrix.to_csv(f'UI_matrix_top {n_items}_items.csv')
```

## Restaurant

#### Considering attributes 'Food' and 'Restaurant' in buisness data frame

#### Get buisness id if either of attributes 'food' or 'restaurant' is 1

In [63]:

```
df_food=df_buisness[['Restaurants','Food','business_id']]
restaurants = []
for i in range(len(df_food.index)):
  if df_food['Restaurants'].iloc[i] == 1 or df_food['Food'].iloc[i] ==1:
    restaurants.append(df_food['business_id'].iloc[i])
```

#### Consider only those business from reviews that are in list restaurants

In [64]:

```
df_restaurant=df_review[df_review.business_id.isin(restaurants)]
```

In [68]:

```
df_restaurant = df_restaurant[['user_id','business_id','stars']]
```

In [69]:

```
df_restaurant.to_csv('df_restaurant.csv',index=False)
```

#### Item-Item Collaborative Based Filtering - Using similarity matrix

In [92]:

```
df_matrix = pd.read_csv('UI_matrix_top 100_items.csv',index_col=[0])
```

In [80]:

```
df_matrix.head()
```

Out[80]:

| | PY9GRfzr4nTZeINf346QOw | W4ZEKkva9HpAdZG88juwyQ | SZU9c8V2GuREDN5KgyHFJw | YvyVOK0k5 |
|---|---|---|---|---|
| **qEEk0PuoH1dVa619t8fgpw** | 4 | 0 | 0 | |
| **EBa-0-6AKoy6jziNexDJtg** | 0 | 3 | 0 | |
| **JYYYKt6TdVA4ng9ILcXt_g** | 0 | 0 | 5 | |
| **7P9w2PrP4ZcJyDFwch51lg** | 0 | 0 | 0 | |
| **pitYOVSsF8R1gWG1G0qxsA** | 0 | 0 | 0 | |

5 rows × 100 columns

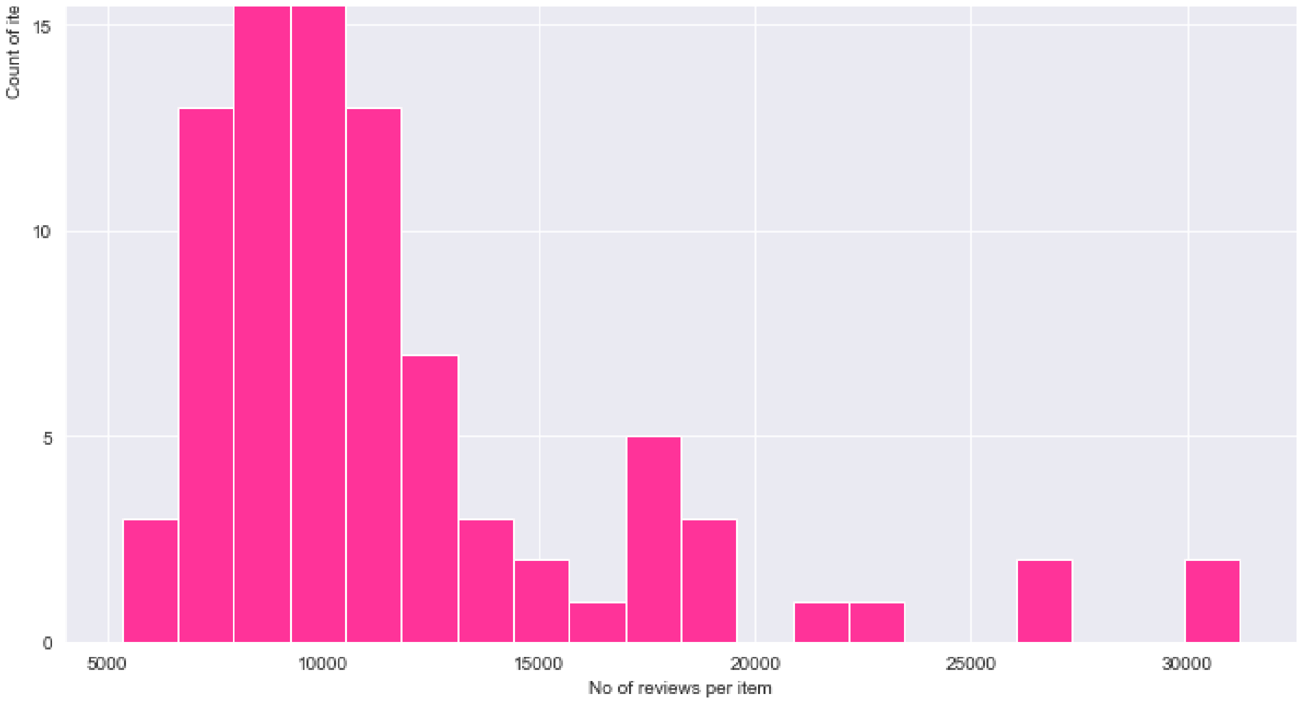
In [81]:

```
print(f'No of users:{df_matrix.shape[0]}, Along rows')
print(f'No of items:{df_matrix.shape[1]}, Along columns')
```

```
No of users:192665, Along rows
No of items:100, Along columns
```

In [82]:

```
sns.set_style("darkgrid")
review_per_item = df_matrix.sum(axis=0)
plt.hist(review_per_item,bins=20,color='#ff3399')
plt.xlabel('No of reviews per item')
plt.ylabel('Count of items')
plt.title('Distribution number of reviews for items')
plt.show()
```




**Remove users with less than 14 reviews to reduce no of users as the algorithm takes a lot of time to run.**

In [104]:

```
items_to_drop=[]
for col,val in df_matrix.sum(axis=1).items():
  if val < 14:
    items_to_drop.append(col)
```

In [105]:

```
print(f'No of items to drop:{len(items_to_drop)}')
```

```
No of items to drop:181362
```

In [106]:

```
df_matrix.drop(items_to_drop,axis=0,inplace=True)
```

In [107]:

```
df_matrix.shape
```

Out[107]:

```
(11303, 100)
```

## Centered data

### Normalizing ratings by subtracting row mean

In [108]:

```
pd.set_option('display.float_format', lambda x: '%.5f' % x)
# mean of the rows
users_mean_rating = df_matrix.mean(axis=1)
# subtract all values by mean of the row
df_mat_norm=df_matrix.sub(users_mean_rating,axis='rows')
```

In [109]:

```
df_mat_norm=df_mat_norm.astype('float16')
```

In [110]:

```
#Convert to matrix
ui_mat_norm = df_mat_norm.values
ui_mat = df_matrix.values
```

## Cosine similarity

In [111]:

```
n_items=ui_mat_norm.shape[1]
sim_matrix={}
# Loop through the columns
for i in range(n_items):
  row_sim=[]
  col = df_matrix.columns[i]
  # Loop through the columns for each column
  for j in range(n_items):
    #sim=cal_cosine_sim(ui_mat_norm[:,i],ui_mat_norm[:,j],ui_mat[:,i],ui_mat[:,j])
    sim=1-spatial.distance.cosine(ui_mat_norm[:,i],ui_mat_norm[:,j])
    row_sim.append(sim)
  sim_matrix[col]=row_sim
  if i%25==0:
    print('item:',i)
```

```
item: 0
item: 25
item: 50
item: 75
```

In [112]:

```
# create df of similarity for better visualization
similarity_df=pd.DataFrame.from_dict(sim_matrix,columns=df_matrix.columns,orient='index'
)
similarity_df.head(10)
```

Out[112]:

| | PY9GRfzr4nTZeINf346QOw | W4ZEKkva9HpAdZG88juwyQ | SZU9c8V2GuREDN5KgyHFJw | YvyVOK0 |
|---|---|---|---|---|
| PY9GRfzr4nTZeINf346QOw | 1.00000 | -0.00919 | 0.17407 | |
| W4ZEKkva9HpAdZG88juwyQ | -0.00919 | 1.00000 | -0.01185 | |
| SZU9c8V2GuREDN5KgyHFJw | 0.17407 | -0.01185 | 1.00000 | |
| Zi-F-YvyVOK0k5QD7IrLOg | 0.00645 | 0.03372 | -0.00419 | |
| GBTPC53ZrG1ZBY3DT8Mbcw | -0.12122 | 0.04486 | -0.06775 | |
| pSmOH4a3HNNpYM82J5ycLA | 0.05823 | -0.03427 | 0.03720 | |
| 8uF-bhJFgT4Tn6DTb27viA | -0.02663 | -0.01408 | -0.02966 | |
| UCMSWPqzXjd7QHq7v8PJjQ | 0.30835 | -0.02058 | 0.20520 | |
| vN6v8m4DO45Z4pp8yxxF_w | -0.01654 | -0.00156 | -0.02403 | |
| g04aAvgol7IW8buqSbT4xA | 0.05533 | 0.00692 | 0.03046 | |

10 rows × 100 columns

**Get item score for each userGet item score for each user**

In [113]:

```
def cal_score(similarities,history,avgRating):
  nume=np.sum((history-avgRating)*similarities)
  deno=np.sum(similarities)

  return nume/deno
```

```python
def get_user_score(sim_mat,df,user,user_items,n=10):
  ratings = user_items.values
  mean_user_rate = np.mean(ratings)
  score_vector=[]
  # iterrate over all items
  for item,rate in user_items.items():
    # for item 'i'
    # check if the item 'i' is already rated
    if rate > 0:
      # if rated store -1 so its not recommended
      score = -1
    else:
      # get 10 most similar items to the item 'i'
      # first most similar item is item 'i', hence 1 to n+1
      topN = sim_mat[item].sort_values(ascending=False).iloc[1:n]
      rate_history=df[topN.index].loc[user]
      # get mean item rating
      mean_user_rate=df[topN.index].mean(axis=0)
      score=cal_score(topN.values,rate_history.values,mean_user_rate.values)
    score_vector.append(score)
  return score_vector
```

In [114]:

```python
user_score_dict={}
for i,user in enumerate(df_matrix.index):
  if i%1000==0:
    print('user:',i)
  user_items = df_matrix.loc[user]
  user_score=get_user_score(similarity_df,df_matrix,user_items.name,user_items)
  user_score_dict[user] = user_score
```

```
user: 0
user: 1000
user: 2000
user: 3000
user: 4000
user: 5000
user: 6000
user: 7000
user: 8000
user: 9000
user: 10000
user: 11000
```

In [115]:

```python
# create a data frame of user item score
user_item_score=pd.DataFrame.from_dict(user_score_dict,orient='index',columns=df_matrix.
columns)
user_item_score.head()
```

Out[115]:

| | PY9GRfzr4nTZeINf346QOw | W4ZEKkva9HpAdZG88juwyQ | SZU9c8V2GuREDN5KgyHFJw | YvyVOK0k5 |
|---|---|---|---|---|
| **EBa-0-6AKoy6jziNexDJtg** | -0.02740 | -1.00000 | -0.03901 | |
| **JYYYKt6TdVA4ng9lLcXt_g** | -0.02740 | -0.32274 | -1.00000 | |
| **pitYOVSsF8R1gWG1G0qxsA** | -0.02740 | 0.05062 | -0.03901 | |
| **1xS8Jj23zHx8axlVopG3wA** | -0.02740 | -1.00000 | -0.03901 | |
| **ftRgzVFzv6-TOCBXEOdWeQ** | -0.02740 | 0.67574 | -0.03901 | |

5 rows × 100 columns

In [116]:

```
user_item_recomend = pd.DataFrame(columns=['1','2','3','4','5'])
```

In [117]:

```
# get top 5 recommendation for every user
for idx in user_item_score.index:
  user_pref=user_item_score.loc[idx]
  user_pref=user_pref.sort_values(ascending=False)[0:5]
  user_item_recomend.loc[idx] = list(user_pref.index)
```

In [118]:

```
user_item_recomend.head()
```

Out[118]:

| | 1 | 2 | 3 | |
|---|---|---|---|---|
| **EBa-0-6AKoy6jziNexDJtg** | iwmW2mgcn2YdirXUHCsgXQ | Zi-F-YvyVOK0k5QD7lrLOg | c-iKAO2GBzSKjm7y1Oljcw | ww3YJ |
| **JYYYKt6TdVA4ng9lLcXt_g** | U3grYFleu6RgAAQgdriHww | ww3YJXu5c18aGZXWmm00qg | skY6r8WAkYqpV7_TxNm23w | yPS |
| **pitYOVSsF8R1gWG1G0qxsA** | iRIHK8-EwpeffwvoO4nzIA | 2BMk_drsikKWslJCXmQtjQ | UFCN0bYdHroPKu6KV5CJqg | g04 |
| **1xS8Jj23zHx8axlVopG3wA** | Y2Pfil51rNvTd_lFHwzb_g | ChlcxTEoWBQJXJ2Xb2vm5g | iSRTaT9WngzB8JJ2YKJUig | VaO-V |
| **ftRgzVFzv6-TOCBXEOdWeQ** | Vz2RN55rTJBGn43K1v84nA | V9VLhHdSFpFi4yXFqVcVEA | hfbZ97Te3T4jeWN6GgsGrQ | oBN |

## Collaborative filtering using Stochastic Gradient

### Create continous values for unique buisness_id and user_id which are of the form string

In [10]:

```
df_restaurant = pd.read_csv('csv/df_restaurant.csv')
```

In [4]:

```
items=df_restaurant.business_id.unique()
users=df_restaurant.user_id.unique()
```

In [5]:

```
userid2idx = {o:i for i,o in enumerate(users)}
items2idx = {o:i for i,o in enumerate(items)}
```

In [6]:

```
df_restaurant['business_id']=df_restaurant['business_id'].apply(lambda x: items2idx[x])
df_restaurant['user_id']=df_restaurant['user_id'].apply(lambda x: userid2idx[x])
```

In [7]:

```
df_restaurant.head()
```

Out[7]:

| | user_id | business_id | stars |
|---|---|---|---|
| **0** | 0 | 0 | 3 |
| **1** | 1 | 1 | 3 |
| **2** | 2 | 2 | 5 |
| **3** | 3 | 3 | 4 |
| **4** | 4 | 4 | 1 |

In [8]:

```
# unique users and items
nitems=df_restaurant.business_id.nunique()
nusers=df_restaurant.user_id.nunique()
```

In [9]:

```
nusers,nitems
```

Out[9]:

```
(1504895, 64577)
```

## Train test split

In [10]:

```
train_idx, val_idx = train_test_split(range(df_restaurant.shape[0]), train_size=0.7,rando
m_state=2971)
df_train= df_restaurant.iloc[train_idx]
df_validate = df_restaurant.iloc[val_idx]
df_train.shape, df_validate.shape
```

Out[10]:

```
((4611978, 3), (512443, 3))
```

In [11]:

```
def create_bias(name, inp, n_in, reg):
    x = Embedding(n_in, 1, input_length=1, name=name)(inp)
    return Flatten(name=name+'_flattened')(x)
```

In [12]:

```
def embedding_input(name, n_in, n_out, reg):
    inp = Input(shape=(1,), dtype='int64', name=name)
    return inp, Embedding(n_in, n_out, input_length=1, name=name.split('_')[0]+'_factor'
, embeddings_regularizer=l2(reg))(inp)
```

In [13]:

```
L = 45
REG=8e-4
```

In [14]:

```
# Create embeddings
user_input, mat_user_lf = embedding_input('user_input', nusers, L, REG)
restraunts_input, mat_restau_lf = embedding_input('restraunts_input', nitems, L, REG)
```

In [15]:

```
# Create bias
user_bias = create_bias('user_bias', user_input, nusers, REG)
restraunts_bias = create_bias('movie_bias', restraunts_input, nitems, REG)
```

In [16]:

```
# create residuals matrix
residual = Dot(axes=2, name="residual")([mat_user_lf, mat_restau_lf])
# Flatten the layer
residual_flatten = Flatten(name="residual_flat")(residual)
```

In [17]:

```
# regression layer
```

```
regression = Add(name="regression")([user_bias, restraunts_bias, residual_flatten])
```

In [18]:

```
# Create a tailor made sigmoid to keep output values within range 0-5
def sigmoid_maker(low, high):
    def custom_sigmoid(x):
        return K.sigmoid(x)*(high - low) + low
    return custom_sigmoid
cs = sigmoid_maker(0, 5.5)
```

In [19]:

```
output = Activation(cs, name="Sigmoid")(regression)
```

In [20]:

```
model_lf1 = Model([user_input, restraunts_input], output)
model_lf1.compile(Adam(0.01), loss='mse')
```

In [21]:

```
model_lf1.summary()
```

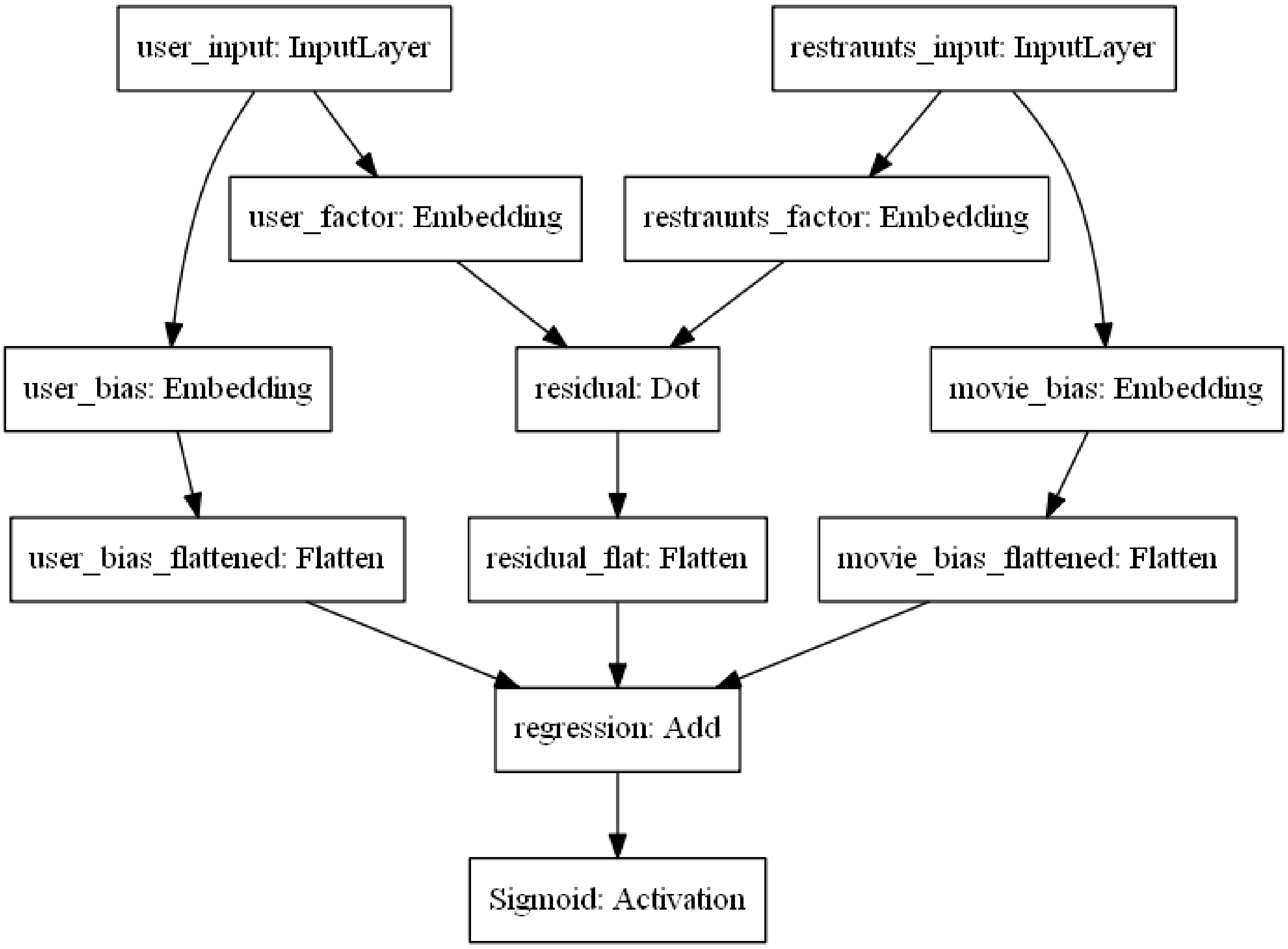
```
Model: "model"
__________________________________________________________________________________________
________
Layer (type)                    Output Shape         Param #     Connected to

==========================================================================================
========
user_input (InputLayer)         [(None, 1)]          0


__________________________________________________________________________________________
________
restraunts_input (InputLayer)   [(None, 1)]          0


__________________________________________________________________________________________
________
user_factor (Embedding)         (None, 1, 45)        67720275    user_input[0][0]


__________________________________________________________________________________________
________
restraunts_factor (Embedding)   (None, 1, 45)        2905965     restraunts_input[0][0]


__________________________________________________________________________________________
________
user_bias (Embedding)           (None, 1, 1)         1504895     user_input[0][0]


__________________________________________________________________________________________
________
movie_bias (Embedding)          (None, 1, 1)         64577       restraunts_input[0][0]


__________________________________________________________________________________________
________
residual (Dot)                  (None, 1, 1)         0           user_factor[0][0]

                                                                 restraunts_factor[0][0
]
__________________________________________________________________________________________
________
user_bias_flattened (Flatten)   (None, 1)            0           user_bias[0][0]


__________________________________________________________________________________________
________
movie_bias_flattened (Flatten)  (None, 1)            0           movie_bias[0][0]


__________________________________________________________________________________________
________
residual_flat (Flatten)         (None, 1)            0           residual[0][0]
```

```
________________________________________________________________________________________________
regression (Add)                (None, 1)            0           user_bias_flattened[0][
0]
                                                                 movie_bias_flattened[0
][0]
                                                                 residual_flat[0][0]

________________________________________________________________________________________________
Sigmoid (Activation)            (None, 1)            0           regression[0][0]

================================================================================================
Total params: 72,195,712
Trainable params: 72,195,712
Non-trainable params: 0
________________________________________________________________________________________________
```

In [22]:

```
tf.keras.utils.plot_model(model_lf1)
```

Out[22]:




In [ ]:

```
history_lf1=model_lf1.fit([df_train.user_id, df_train.business_id], df_train.stars, batch_size=1024, epochs=3, validation_data=([df_validate.user_id, df_validate.business_id], df_validate.stars))
```

In [ ]:

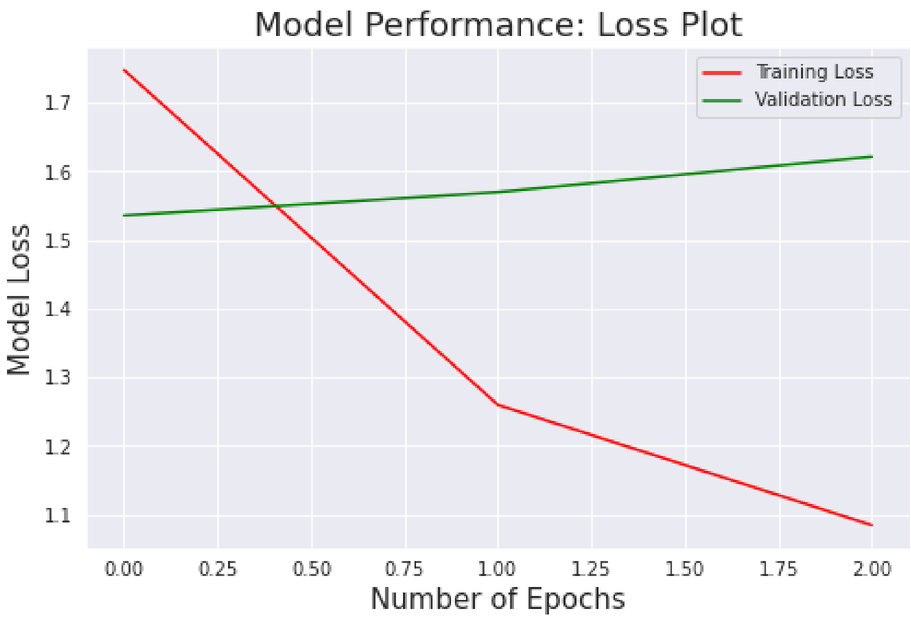
```
# plotting history
sns.set_style('darkgrid')
plt.rcParams['figure.figsize']=(8,5)
plt.plot(history_lf1.history['loss'],label='Training Loss',color='red')
```

```
plt.plot(history_lf1.history['val_loss'],label='Validation Loss',color='green')
plt.xlabel('Number of Epochs',fontsize=15)
plt.ylabel('Model Loss',fontsize=15)
plt.title('Model Performance: Loss Plot',fontsize=18)
plt.legend()
```

In [3]:

```
Image(filename='lossplot_sgd.png')
```

Out[3]:




## Hybrid Approach

In [11]:

```
users_review_count=df_restaurant.groupby('user_id')['stars'].count()
freq_users=[]
for i,val in users_review_count.items():
  if val>2:
    freq_users.append(i)
df_restaurant=df_restaurant[df_restaurant['user_id'].isin(freq_users)]
```

In [12]:

```
most_rated_items=df_restaurant.groupby(['business_id'])['business_id'].count()
popular_rest=[]
for i,val in most_rated_items.items():
  if val>1:
    popular_rest.append(i)
df_restaurant=df_restaurant[df_restaurant['user_id'].isin(popular_rest)]
df_restaurant.reset_index(drop=True,inplace=True)
```

## Users-Products matrix

In [13]:

```
items=df_restaurant.business_id.unique()
users=df_restaurant.user_id.unique()
```

In [14]:

```
userid2idx = {o:i for i,o in enumerate(users)}
items2idx = {o:i for i,o in enumerate(items)}
```

In [15]:

```
df_restaurant.head()
```

Out[15]:

| | user_id | business_id | stars |
|---|---|---|---|
| 0 | DBYhpb5hrAYgQjQaMhNYyQ | oJ4ik-4PZe6gexxW-tSmsw | 4 |
| 1 | XTWdXSOoUJnIMiVSA-1gDg | _RwIMTw9uFeOkfX9Ctf1HA | 1 |
| 2 | f6B7YotlkKfXr9xN-TbpwA | Bt7NBqA31uOI4H_hvasLLg | 5 |
| 3 | Fp0SeuMpAzcwPITfyF95hA | DuPRwh_pNsp4LkbICuF3lg | 4 |
| 4 | DBYhpb5hrAYgQjQaMhNYyQ | JJNCJWaH2KV44r9aeEBlqA | 4 |

In [16]:

```
df_restaurant['business_id']=df_restaurant['business_id'].apply(lambda x: items2idx[x])
df_restaurant['user_id']=df_restaurant['user_id'].apply(lambda x: userid2idx[x])
```

In [17]:

```
def make_user_item_matrix(user_item,df):
  len_UI = user_item.shape[0]
  for i in range(len_UI):
    index = user_item['user_id'].iloc[i]
    col = user_item['business_id'].iloc[i]
    rate = user_item['stars'].iloc[i]
    df[col].loc[index]=rate

  return df
```

In [18]:

```
unique_items = df_restaurant.business_id.unique()
unique_user = df_restaurant['user_id'].unique()
null_ui_mat={}
for i in unique_user:
  r = list(np.zeros(len(unique_items)))
  null_ui_mat[i]=r

UI_matrix = pd.DataFrame.from_dict(null_ui_mat,columns=unique_items,orient='index').asty
pe('int32')
```

In [19]:

```
UI_matrix=make_user_item_matrix(df_restaurant,UI_matrix)
```

In [20]:

```
UI_matrix.head()
```

Out[20]:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... | 2703 | 2704 | 2705 | 2706 | 2707 | 2708 | 2709 | 2710 | 2711 | 2712 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 4 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

5 rows × 2713 columns

In [21]:

```
UI_matrix = pd.DataFrame(preprocessing.MinMaxScaler(feature_range=(0.5,1)).fit_transform(
UI_matrix.values),
```

```
columns=UI_matrix.columns, index=UI_matrix.index)
```

In [22]:

```
split = int(0.7*UI_matrix.shape[1])
df_train = UI_matrix.loc[:, :split-1]
df_val = UI_matrix.loc[:, split:]
```

In [23]:

```
train = df_train.stack(dropna=True).reset_index().rename(columns={'level_0':'user','leve
l_1':'items',0:"y"})
train.head()
```

Out[23]:

| | user | items | y |
|---|---|---|---|
| **0** | 0 | 0 | 1.0 |
| **1** | 0 | 1 | 0.5 |
| **2** | 0 | 2 | 0.5 |
| **3** | 0 | 3 | 0.5 |
| **4** | 0 | 4 | 1.0 |

In [24]:

```
val = df_val.stack(dropna=True).reset_index().rename(columns={'level_0':'user','level_1'
:'items',0:"y"})
```

In [25]:

```
embeddings_shape = 50
user,items = UI_matrix.shape[0], UI_matrix.shape[1]
```

In [26]:

```
nusers_in = Input(name="nusers_in", shape=(1,))
nproducts_in = Input(name="nproducts_in", shape=(1,))
```

## Matrix factorization

In [27]:

```
mf_nusers_emb = Embedding(name="cf_nusers_emb", input_dim=user, output_dim=embeddings_sha
pe)(nusers_in)
mf_nusers = Reshape(name='cf_nusers', target_shape=(embeddings_shape,))(mf_nusers_emb)
```

In [28]:

```
mf_nproducts_emb = Embedding(name="cf_nproducts_emb", input_dim=items, output_dim=embeddi
ngs_shape)(nproducts_in)
mf_nproducts = Reshape(name='cf_nproducts', target_shape=(embeddings_shape,))(mf_nproduc
ts_emb)
```

In [29]:

```
mf_product = Dot(name='cf_product', normalize=True, axes=1)([mf_nusers, mf_nproducts])
```

## Neural Network

In [30]:

```
nn_nusers_emb = Embedding(name="nn_nusers_emb", input_dim=user, output_dim=embeddings_sha
pe)(nusers_in)
```

```
nn_nusers = Reshape(name='nn_nusers', target_shape=(embeddings_shape,))(nn_nusers_emb)
```

In [31]:

```
nn_nproducts_emb = Embedding(name="nn_nproducts_emb", input_dim=items, output_dim=embeddi
ngs_shape)(nproducts_in)
nn_nproducts = Reshape(name='nn_nproducts', target_shape=(embeddings_shape,))(nn_nproduc
ts_emb)
```

In [32]:

```
nn_product = Concatenate()([nn_nusers, nn_nproducts])
nn_product = Dense(name="nn_product", units=int(embeddings_shape/2), activation='relu')(
nn_product)
```

### Merging both

In [33]:

```
y_out = Concatenate()([mf_product, nn_product])
y_out = Dense(name="y_out", units=1, activation='linear')(y_out)
```

In [34]:

```
model_h = Model(inputs=[nusers_in,nproducts_in], outputs=y_out, name="Hybrid_filtering")
```

In [35]:

```
model_h.summary()
```

```
Model: "Hybrid_filtering"
__________________________________________________________________________________________
__________
Layer (type)                    Output Shape         Param #     Connected to

==========================================================================================
=========
nusers_in (InputLayer)          [(None, 1)]          0


__________________________________________________________________________________________
__________
nproducts_in (InputLayer)       [(None, 1)]          0


__________________________________________________________________________________________
__________
nn_nusers_emb (Embedding)       (None, 1, 50)        14300       nusers_in[0][0]


__________________________________________________________________________________________
__________
nn_nproducts_emb (Embedding)    (None, 1, 50)        135650      nproducts_in[0][0]


__________________________________________________________________________________________
__________
cf_nusers_emb (Embedding)       (None, 1, 50)        14300       nusers_in[0][0]


__________________________________________________________________________________________
__________
cf_nproducts_emb (Embedding)    (None, 1, 50)        135650      nproducts_in[0][0]


__________________________________________________________________________________________
__________
nn_nusers (Reshape)             (None, 50)           0           nn_nusers_emb[0][0]


__________________________________________________________________________________________
__________
nn_nproducts (Reshape)          (None, 50)           0           nn_nproducts_emb[0][0]


__________________________________________________________________________________________
__________
cf_nusers (Reshape)             (None, 50)           0           cf_nusers_emb[0][0]
```

```
________________________________________________________________________________________
__________
cf_nproducts (Reshape)          (None, 50)          0           cf_nproducts_emb[0][0]

________________________________________________________________________________________
__________
concatenate (Concatenate)       (None, 100)         0           nn_nusers[0][0]

                                                                nn_nproducts[0][0]

________________________________________________________________________________________
__________
cf_product (Dot)                (None, 1)           0           cf_nusers[0][0]

                                                                cf_nproducts[0][0]

________________________________________________________________________________________
__________
nn_product (Dense)              (None, 25)          2525        concatenate[0][0]

________________________________________________________________________________________
__________
concatenate_1 (Concatenate)     (None, 26)          0           cf_product[0][0]

                                                                nn_product[0][0]

________________________________________________________________________________________
__________
y_out (Dense)                   (None, 1)           27          concatenate_1[0][0]

========================================================================================
==========
Total params: 302,452
Trainable params: 302,452
Non-trainable params: 0
________________________________________________________________________________________
__________
```

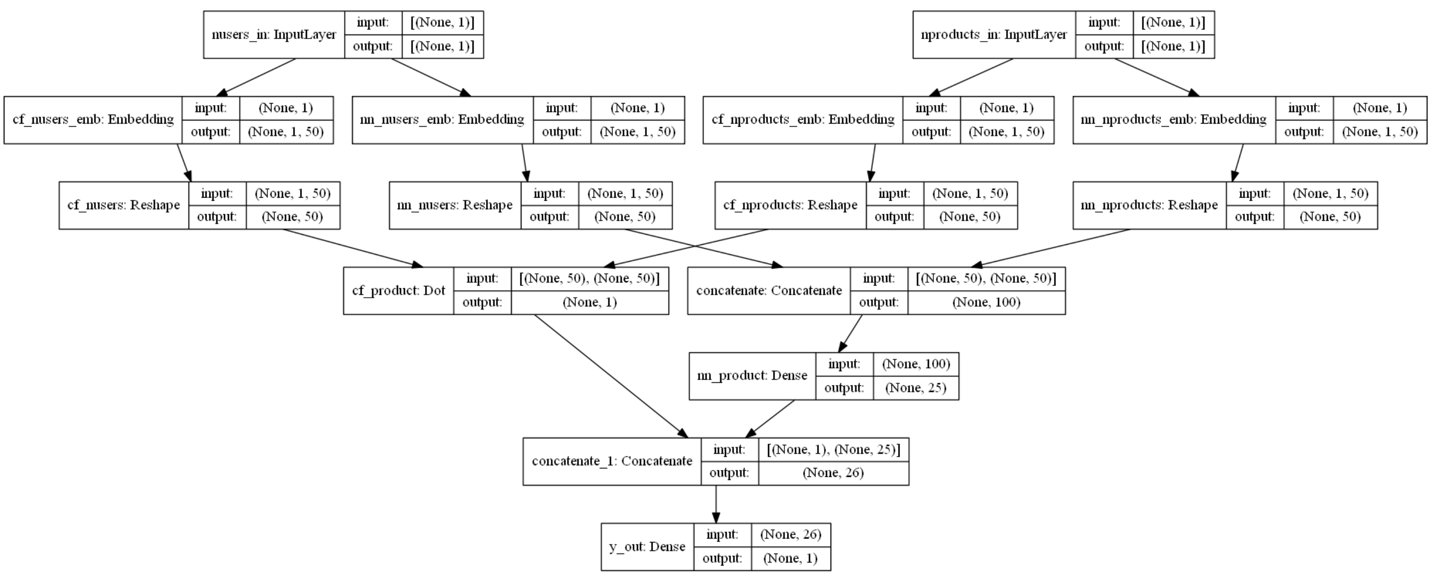
In [36]:

```
model_h.compile(optimizer='adam', loss='mean_absolute_error', metrics=['mean_absolute_pe
rcentage_error'])
```

In [37]:

```
utils.plot_model(model_h, show_shapes=True, show_layer_names=True)
```

Out[37]:




In [38]:

```
history_lf1 = model_h.fit(x=[train["user"], train["items"]], y=train["y"], epochs=10, ba
```

```
history_lf1 = model_n.fit(x=[train['user'], train['items']], y=train['y'], epochs=10, ba
tch_size=64, shuffle=True, validation_split=0.3)
```

```
Epoch 1/10
5941/5941 [==============================] - 25s 4ms/step - loss: 0.0074 - mean_absolute_
percentage_error: 1.2396 - val_loss: 0.0025 - val_mean_absolute_percentage_error: 0.4274
Epoch 2/10
5941/5941 [==============================] - 25s 4ms/step - loss: 0.0030 - mean_absolute_
percentage_error: 0.3632 - val_loss: 0.0018 - val_mean_absolute_percentage_error: 0.2937
Epoch 3/10
5941/5941 [==============================] - 25s 4ms/step - loss: 0.0031 - mean_absolute_
percentage_error: 0.3731 - val_loss: 0.0014 - val_mean_absolute_percentage_error: 0.2130
Epoch 4/10
5941/5941 [==============================] - 24s 4ms/step - loss: 0.0030 - mean_absolute_
percentage_error: 0.3668 - val_loss: 0.0014 - val_mean_absolute_percentage_error: 0.2118
Epoch 5/10
5941/5941 [==============================] - 24s 4ms/step - loss: 0.0030 - mean_absolute_
percentage_error: 0.3608 - val_loss: 0.0016 - val_mean_absolute_percentage_error: 0.2461
Epoch 6/10
5941/5941 [==============================] - 24s 4ms/step - loss: 0.0030 - mean_absolute_
percentage_error: 0.3568 - val_loss: 0.0013 - val_mean_absolute_percentage_error: 0.2028
Epoch 7/10
5941/5941 [==============================] - 24s 4ms/step - loss: 0.0030 - mean_absolute_
percentage_error: 0.3524 - val_loss: 0.0014 - val_mean_absolute_percentage_error: 0.2064
Epoch 8/10
5941/5941 [==============================] - 23s 4ms/step - loss: 0.0030 - mean_absolute_
percentage_error: 0.3499 - val_loss: 0.0014 - val_mean_absolute_percentage_error: 0.2174
Epoch 9/10
5941/5941 [==============================] - 24s 4ms/step - loss: 0.0029 - mean_absolute_
percentage_error: 0.3462 - val_loss: 0.0015 - val_mean_absolute_percentage_error: 0.2302
Epoch 10/10
5941/5941 [==============================] - 24s 4ms/step - loss: 0.0029 - mean_absolute_
percentage_error: 0.3420 - val_loss: 0.0012 - val_mean_absolute_percentage_error: 0.1782
```

In [39]:

```
sns.set_style('darkgrid')
plt.rcParams['figure.figsize']=(8,5)
plt.plot(history_lf1.history['mean_absolute_percentage_error'],label='Training Loss',colo
r='red')
plt.plot(history_lf1.history['val_mean_absolute_percentage_error'],label='Validation Loss
',color='green')
plt.xlabel('Number of Epochs',fontsize=15)
plt.ylabel('Model Loss',fontsize=15)
plt.title('Model Performance: Loss Plot',fontsize=18)
plt.legend()
plt.show()
```


## Conclusion

**We tried different types of techniques to build a restaurant recommendation system.**
**EDA helped us gather insights about the data which helped in model building. The first cosine similarity based collaborative model was the starting point.**
**Moving on to the neural netowrk based models, the first one overfit the training data in just 3 epochs, so we decided to land on a hybrid model which performed far more better than the previous one.**