**Phân tích thuật toán Recommendation**

Thuật toán được cài đặt qua 4 bước chính:

# In[0]: thêm thư viện

Thêm các thư viện để sử dụng các lệnh từ thư viện đã khai báo

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import re

import nltk

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.metrics.pairwise import cosine\_similarity

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.feature\_extraction.text import TfidfVectorizer

# In[1]: Xem nhanh dữ liệu

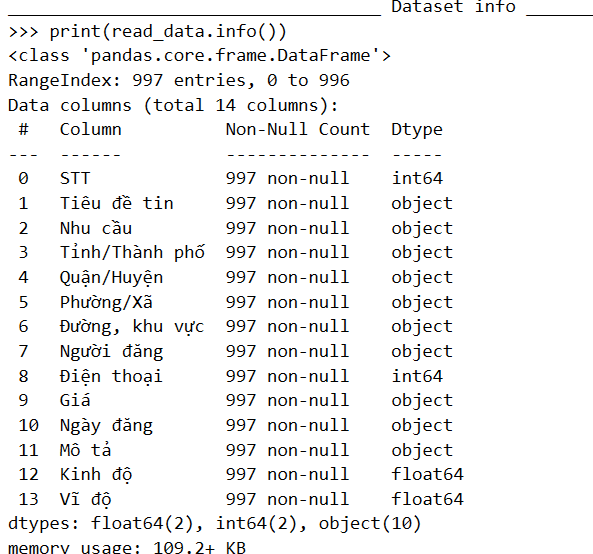
|  |
| --- |
| read\_data=pd.read\_csv(r"C:\CNTT\_Namw4\_kỳ1\DOAN3\dulieucsv")  print("Done.") |

Đọc dữ liệu từ file csv dữ liệu đã chuẩn bị sẵn. Dữ liệu(998x14) gồm 14 cột (bao gồm các mục:STT, Đường, khu vực, Người đăng, Điện thoại, Giá, Ngày đăng, Mô tả, Kinh độ, Vĩ độ) và 998 dòng.

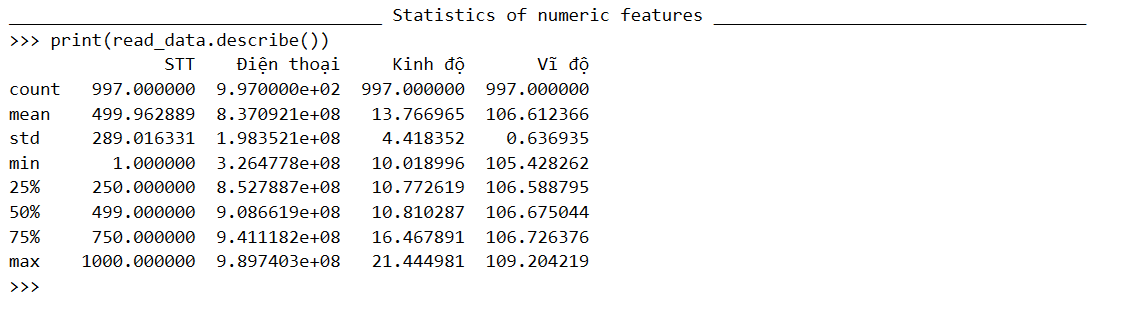
Vấn đề khi load dữ liệu từ cột “Kinh độ” và “Vĩ độ” ở dạng dấu chấm phẩy sẽ chuyển thành dấu chấm do khi sử lý dữ liệu từ excel sang csv dấu chấm sẽ bị lỗi (106.708549 sẽ chuyển thành 106.708.549). Để sử lý dữ liệu ta sử dụng hàm Clean để chuyển dữ liệu chuyển dấu ; thành dấu chấm khi đó dữ liệu được định dạng lại đúng là kiểu float64.

|  |
| --- |
| def clean(text):  text = text.replace(";", ".")  return float(text)  read\_data['Vĩ độ'] = read\_data['Vĩ độ'].apply(lambda x: clean(x))  read\_data['Kinh độ'] = read\_data['Kinh độ'].apply(lambda x: clean(x)) |

Dữ liệu sau khi được chuẩn hóa



Đánh giá dữ liệu: Số lượng các cột(count), mean là trung bình cộng của tất cả các giá trị, min là giá trị nhỏ nhất, 25%-50%-75% là phần trăm giá trị bé hơn trong 100% giá trị, max là giá trị lớn nhất.

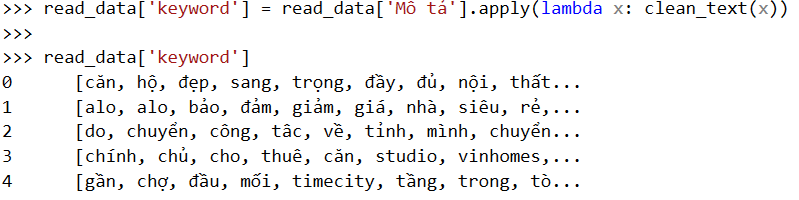


# In[2]: Tìm key word

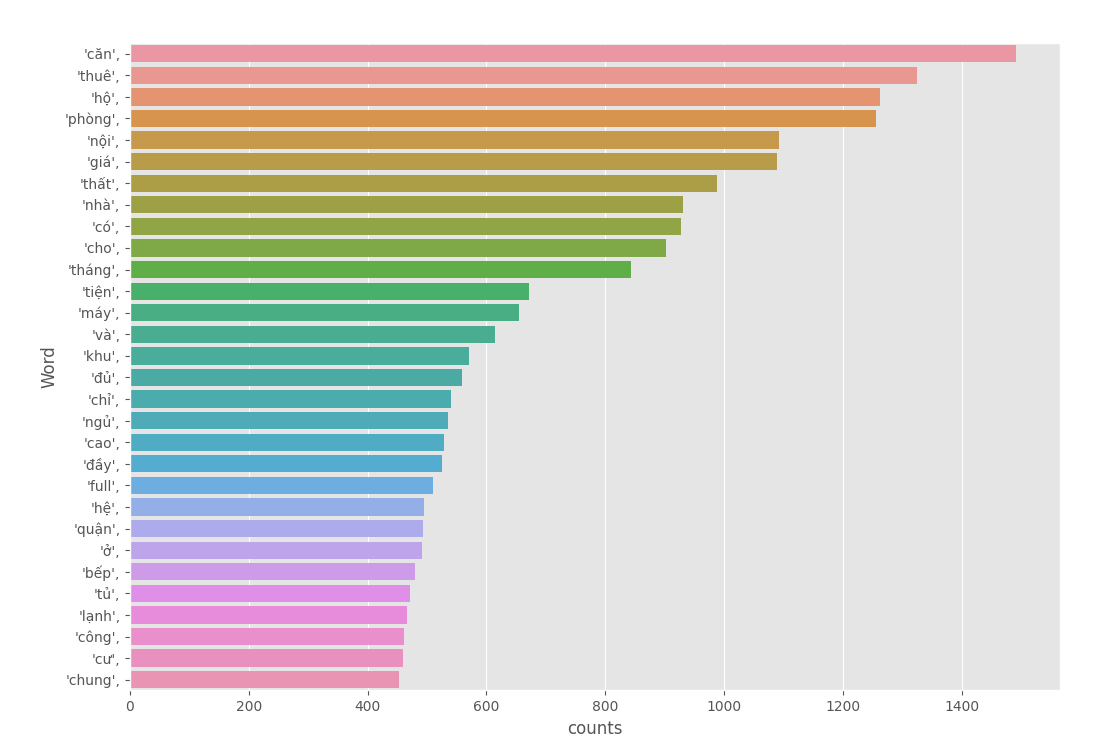
Khi sử lý dữ liệu ở phần “Mô tả”, chứa các thông tin về nhà trọ và được viết như một văn bản, từ văn bản này để cho máy tính hiểu được thì ta sẽ chuyển văn bản về dạng mảng để cho máy tính đọc hiểu. Để giải quyết vấn đề đó thì ta sẽ khởi tạo một hàm clean\_text(text):

|  |
| --- |
| def clean\_text(text):  # remove backslash-apostrophe  text = re.sub("\'", "", text)  # remove whitespaces  text = ' '.join(text.split())  # remove everything except alphabets  # text = re.sub("[^a-zA-Z]"," ",text)  # convert text to lowercase  text = text.lower()  text = re.findall(r'(?i)\b[a-záàảãạăắằẳẵặâấầẩẫậéèẻẽẹêếềểễệóòỏõọôốồổỗộơớờởỡợíìỉĩịúùủũụưứừửữựýỳỷỹỵđ]+\b', text)  return text |

Hàm clean\_text(text) lấy ý tưởng: khi gặp khoảng trắng hoặc ký tự đặc biệt sẽ tách ký tự ra thành một keyword và một đoạn văn sẽ được lưu thành một mảng và các mảng sẽ được lưu vào một cột có tên là “keyword”



Sau khi đã có dữ liệu sẽ tiến hành lọc ra xem keyword nào được sử dụng nhiều nhất



Ở bước này chạy khởi tạo một hàm freq\_words(x, terms = 30) để thực hiện đếm dữ liệu và vẽ dữ liệu ra trên hình

|  |
| --- |
| def freq\_words(x, terms = 30):  #all\_words = ' '.join([text for text in x])  all\_words = ' '.join(str(text) for text in x)  all\_words = all\_words.split()  fdist = nltk.FreqDist(all\_words)  words\_df = pd.DataFrame({'word':list(fdist.keys()), 'counts':list(fdist.values())})    # selecting top 20 most frequent words  d = words\_df.nlargest(columns="counts", n = terms)    # visualize words and frequencies  plt.figure(figsize=(12,15))  ax = sns.barplot(data=d, x= "counts", y = "word")  ax.set(ylabel = 'Word')  plt.show() |

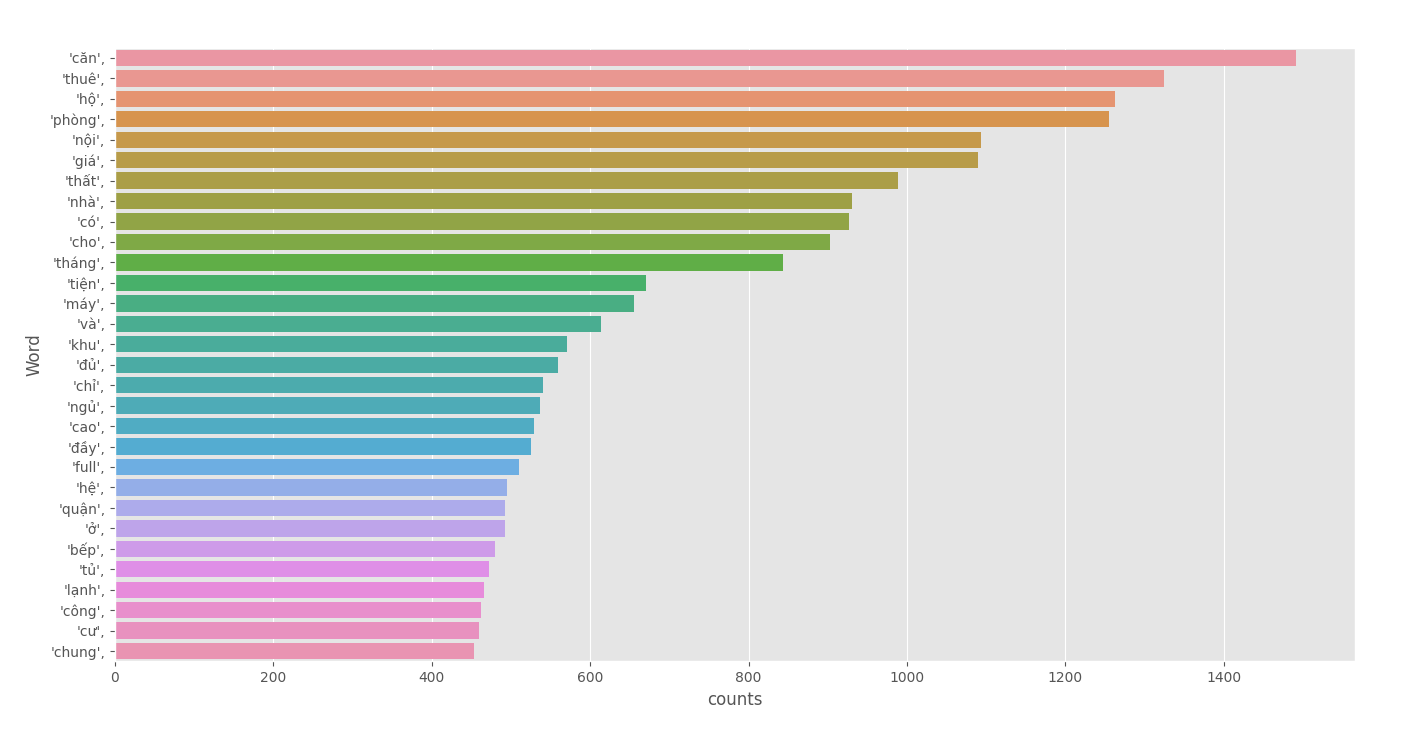
Tiếp theo sau khi thực hiện tìm ra được keyword, tiếp tục thực hiện xóa các từ không liên quan( ở đây các từ không liên quan được gọi là stopword những từ này sẽ làm cho bài toán bị nhiễu). Nhóm em tiếp tục xóa các từ dừng(stopword). Để sử lý vấn đề này thì nltk đã cung cấp một thư viện để thực hiện xóa các từ dừng này.

|  |
| --- |
| nltk.download('stopwords')  from nltk.corpus import stopwords |

Sau đó thực hiện khởi tạo một hàm remove\_stopwords(text):

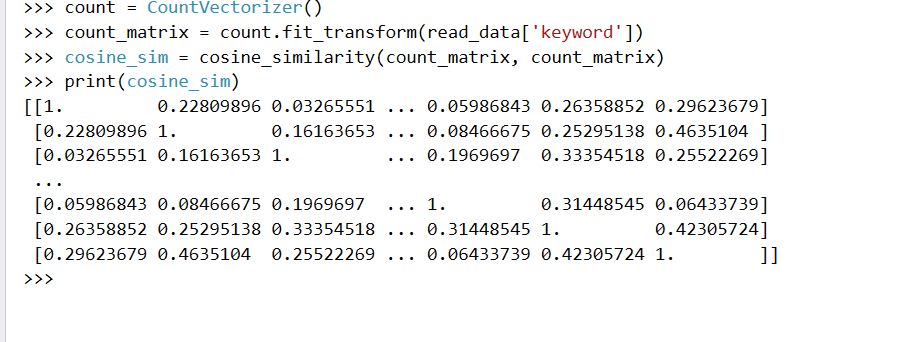
|  |
| --- |
| def remove\_stopwords(text):  no\_stopword\_text = [w for w in str(text).split() if not w in stop\_words]  return ' '.join(no\_stopword\_text) |

Sau khi xóa từ dừng ta sẽ đưuọc dữ liệu mới và sẽ in ra 30 dữ liệu đầu tiên



Để học được keyword, ở đây đã có sẵn hàm để hỗ trợ trong sklearn là cosine\_similarity, CountVectorizer, TfidfVectorizer:

* CountVectorizer thực hiện: Chuyển đổi bộ sưu tập tài liệu văn bản thành ma trận số lượng mã thông báo
* TfidfVectorizer thực hiện :Chuyển đổi một bộ sưu tập các tài liệu thô sang một ma trận các tính năng TF-IDF
* Cosine\_similarity thực hiện: tính tích số chấm chuẩn hóa L2 của vectơ. Đó là, nếu x và y là các vectơ hàng, tính tương tự cosin của chúng



Số càng cao thì dữ liệu ảnh hưởng đến kết quả càng cao.

# In[3]: Tìm key location

Đến bước 3: Nhóm em thực hiện gợi ý theo khoảng cách. Bước đầu tiên sử lý dữ liệu nhằm hạn chế nhập sai dữ liệu thì nhóm em thực hiện rào dữ liệu lại để tránh sai xót trong quá trình train dữ liệu.

|  |
| --- |
| lon\_min, lat\_min, lon\_max, lat\_max = 10.01899606, 105.4282622 , 21.44498123, 109.2042187  nyc\_events = read\_data[(read\_data['Kinh độ']>lon\_min) &  (read\_data['Kinh độ']<lon\_max) &  (read\_data['Vĩ độ']>lat\_min) &  (read\_data['Vĩ độ']<lat\_max)]  nyc\_events.head()  nyc\_events.shape |

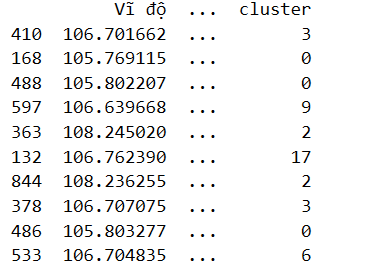
lon\_min là giá trị kinh độ bé nhất, lat\_min là giá trị vĩ độ bé nhất, lon\_max là kinh độ lớn nhất, lat\_max là vĩ độ bé nhất.

Sau đó thực hiện phân cụm cho dữ liệu bằng thư viện đã có sẵn của sklearn là KMeans, silhouette\_score

* silhouette\_score :Hệ số Silhouette được tính bằng cách sử dụng khoảng cách trung bình trong cụm ( a) và khoảng cách trung bình trong cụm gần nhất ( b) cho mỗi mẫu. Hệ số Silhouette cho một mẫu là . Để làm rõ, là khoảng cách giữa mẫu và cụm gần nhất mà mẫu đó không phải là một phần của nó.
* KMeans: Trong thuật toán K-means clustering, chúng ta không biết nhãn (label) của từng điểm dữ liệu. Mục đích là làm thể nào để phân dữ liệu thành các cụm (cluster) khác nhau sao cho dữ liệu trong cùng một cụm có tính chất giống nhau.

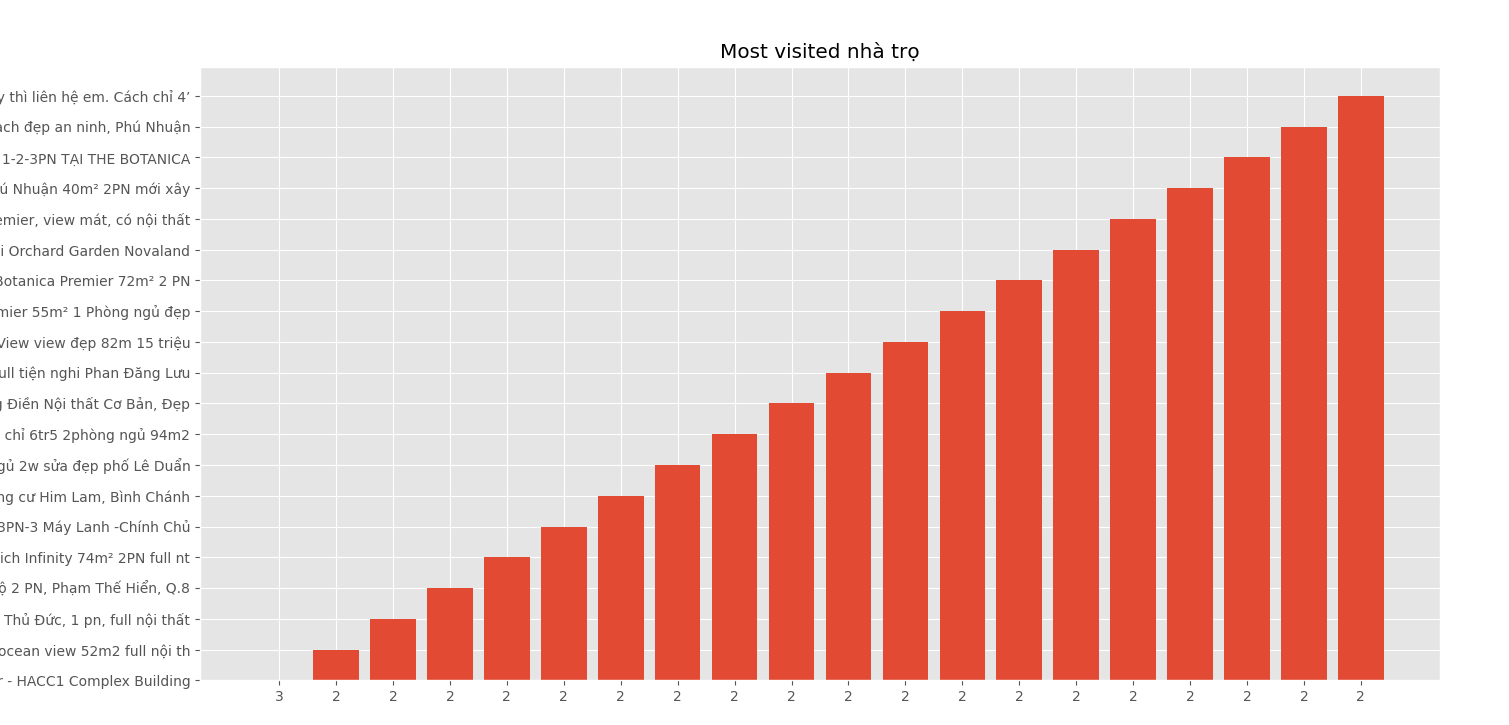
|  |
| --- |
| KMeans(n\_clusters=20)  k = 20 silhouette\_score 0.568102318846464 |

K là số scluster được chia thành 20 cụm, và có khoảng cách trung bình giữa các cụm là 0.568102318846464.



Ở bước tiếp theo nhóm em in ra màn hình 10 tiêu đề thuộc nhiều cluster nhất.

|  |
| --- |
| nyc\_events['cluster'] = kmeans.predict(nyc\_events[['Kinh độ','Vĩ độ']])  nyc\_events[['Vĩ độ','Kinh độ','Tiêu đề tin','cluster']].sample(10)  gdf = nyc\_events.groupby(['cluster', 'Tiêu đề tin']).size().reset\_index()  gdf.columns = ['cluster', 'Tiêu đề tin', 'count']  idx = gdf.groupby(['cluster'])['count'].transform(max) == gdf['count']  topvenues\_df = gdf[idx].sort\_values(by='count', ascending=False)  #top 10 out of 200 clusters by events count  topvenues = topvenues\_df[:10]  plt.style.use('ggplot')  fig = plt.figure()  fig.set\_size\_inches(21,5)  plt.bar(range(len(topvenues)),topvenues['Tiêu đề tin'], align='center')  plt.xticks(range(len(topvenues)), topvenues['count'])  plt.title('Most visited nhà trọ')  plt.show() |



# In[4]: Tìm key word

Sau khi sử lý hoàn tất dữ liệu nhóm em thực hiện train dữ liệu để lấy được ra gợi ý. Bắt đầu ta khởi tạo hàm recommend\_venues(df, longitude, latitude):

* df là mảng giá trị khi tính ra được predict dựa vào kmeans
* longitude là kinh độ của điểm cần dự đoán
* latitude là vĩ độ của điểm cần dự đoán

|  |
| --- |
| def recommend\_venues(df, longitude, latitude):  array = []  predicted\_cluster = kmeans.predict(np.array([longitude,latitude]).reshape(1,-1))[0]  # Fetch the venue name of the top most record in the topvenues dataframe for the predicted cluster  #venue\_name = df[df['cluster']==predicted\_cluster].iloc[0]['Tiêu đề tin']  print(predicted\_cluster)  for i in range(0, len(df[df['cluster']==predicted\_cluster])-1, 1):  array.append(df[df['cluster']==predicted\_cluster].iloc[i]['Tiêu đề tin'])  #msg = 'What about visiting the ' + venue\_name + '?'  #return msg  return array |

Và cuối cùng là thực hiện đưa ra dự đoán

|  |
| --- |
| indices = pd.Series(read\_data['Tiêu đề tin'])  def recommend(title, cosine\_sim = cosine\_sim):  recommended\_nhatrokeyword = []  idx = indices[indices == title].index[0]  score\_series = pd.Series(cosine\_sim[idx]).sort\_values(ascending = False)  top\_10\_indices = list(score\_series.iloc[1:11].index)    for i in top\_10\_indices:  recommended\_nhatrokeyword.append(list(read\_data['Tiêu đề tin'])[i])    return recommended\_nhatrokeyword |

Dựa vào keyword để đưa ra dự đoán(dự đoán 10 dự đoán )

|  |
| --- |
| for i in recommend\_venues(topvenues\_df, 10.809097, 106.672583):  print(i)   * In ra gợi ý về các nhà trọ có khoảng cách giữa điểm đang đứng(điểm nhập liệu) đến các nhà trọ gần địa điểm đang đứng.   for i in recommend('CH Orchard Park View 95m² 3PN FUll NT cao cấp'):  print(i)   * In ra gợi ý tên các nhà trọ (tiêu đề trong dữ liệu) |

Kết quả khi xuất ra cuối cùng:

|  |
| --- |
| for i in recommend\_venues(topvenues\_df, 10.809097, 106.672583):  print(i)   * Kết quả: |

|  |
| --- |
| for i in recommend('CH Orchard Park View 95m² 3PN FUll NT cao cấp'):  print(i)   * Kết quả: |