TRƯỜNG ĐẠI HỌC

**SƯ PHẠM KỸ THUẬT THÀNH PHỐ HỒ CHÍ MINH**



**KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**

NGÀNH CÔNG NGHỆ THÔNG TIN

**BÁO CÁO ĐỒ ÁN 3**

**TÌM HIỂU THUẬT TOÁN RECOMMENDATION ĐỂ GỢI Ý DANH SÁCH PHÒNG TRỌ CHO KHÁCH HÀNG**

**SVTH: CHU MINH HOÀNG 17110139**

**PHẠM HUỲNH THANH LÂM 17110168**

**Khóa: 2017 – 2021**

 **Ngành: CÔNG NGHỆ THÔNG TIN**

 **` GVHD: TS. Huỳnh Xuân Phụng**

TP. Hồ Chí Minh, tháng 12 năm 2020

**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  from sklearn.metrics.pairwise import cosine\_similarity  from sklearn.feature\_extraction.text import CountVectorizer  from sklearn.feature\_extraction.text import TfidfVectorizer |

Thêm các thư viện để thực hiện bài toán dễ dàng hơn:

1. **import pandas as pd**

Pandas là một thư viện Python cung cấp các cấu trúc dữ liệu nhanh, mạnh mẽ, linh hoạt và mang hàm ý. Tên thư viện được bắt nguồn từ panel data (bảng dữ liệu). Pandas được thiết kế để làm việc dễ dàng và trực quan với dữ liệu có cấu trúc (dạng bảng, đa chiều, có tiềm năng không đồng nhất) và dữ liệu chuỗi thời gian.

1. **import numpy as np**

Numpy (Numeric Python): là một thư viện toán học phổ biến và mạnh mẽ của Python. Cho phép làm việc hiệu quả với ma trận và mảng, đặc biệt là dữ liệu ma trận và mảng lớn với tốc độ xử lý nhanh hơn nhiều lần khi chỉ sử dụng “core Python” đơn thuần.

1. **import matplotlib.pyplot as plt**

Matplotlib là một thư viện vẽ đồ thị rất mạnh mẽ hữu ích cho những người làm việc với Python và NumPy. Module được sử dụng nhiều nhất của Matplotib là Pyplot cung cấp giao diện như MATLAB nhưng thay vào đó, nó sử dụng Python và matplotlib là nguồn mở.

1. **import re**

Biểu thức chính quy (Regular Expressions) hay Regex trong Python có thể được định nghĩa là chuỗi các ký tự được sử dụng để tìm kiếm một mẫu trong chuỗi

1. **import nltk**

Bộ công cụ ngôn ngữ tự nhiên (NLTK) là thư viện phổ biến nhất để xử lý ngôn ngữ tự nhiên (NLP). Nó được viết bằng Python và có một cộng đồng lớn đằng sau Python.

1. **import seaborn as sns**

Trực quan hóa dữ liệu thống kê là một thư viện Python phổ biến để thực hiện EDA. Phân tích dữ liệu thăm dò ( EDA ) là một cách tiếp cận để phân tích các tập dữ liệu để tóm tắt các đặc điểm chính của chúng, thường bằng các phương pháp trực quan.

1. **from sklearn.metrics.pairwise import cosine\_similarity**

Độ tương tự cosine là một phép đo độ giống nhau giữa hai vectơ khác không của một không gian tích bên trong . Nó được định nghĩa là bằng cosin của góc giữa chúng, cũng giống như tích trong của các vectơ giống nhau được chuẩn hóa để cả hai đều có độ dài 1. Tính cosin của 0 ° là 1 và nó nhỏ hơn 1 đối với bất kỳ góc nào trong khoảng (0, π] radian. Do đó, nó là một phán đoán về định hướng chứ không phải độ lớn: hai vectơ cùng hướng có độ tương tự cosin là 1, hai vectơ hướng 90 ° so với nhau có độ giống nhau là 0, và hai vectơ đối nhau theo đường kính có độ giống nhau là – 1

1. **from sklearn.feature\_extraction.text import CountVectorizer**

Chuyển đổi bộ sưu tập tài liệu văn bản thành ma trận số lượng mã thông báo

1. **from sklearn.feature\_extraction.text import TfidfVectorizer**

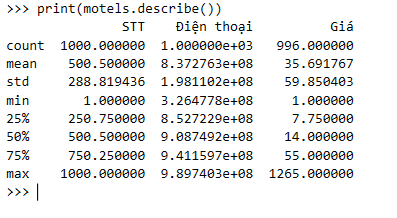
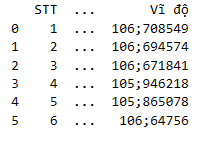
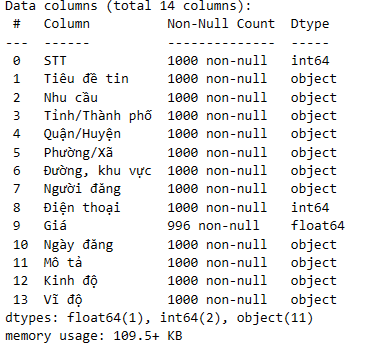
Là 1 kĩ thuật sử dụng trong khai phá dữ liệu văn bản. Trọng số này được sử dụng để đánh giá tầm quan trọng của một từ trong một văn bản.

# In[1]: Xem nhanh dữ liệu

Sau khi thực hiện khai báo các thư viện cần sử dụng, tiếp tục đến bước xem nhanh dữ liệu.

|  |
| --- |
| motels = pd.read\_csv(r"C:\Users\hoang\Desktop\DOAN\_3\DULIEU\dulieu.csv")  moteltest = pd.read\_csv(r"C:\Users\hoang\Desktop\DOAN\_3\DULIEU\dulieutest.csv")  print('\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Dataset info  print(motels.info())  print('\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Some first data examples  print(motels.head(6))  print('\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Statistics of numeric features  print(motels.describe()) |

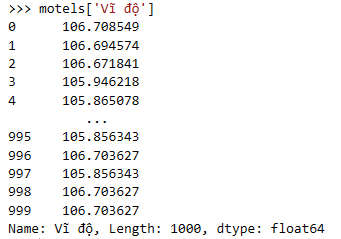
Đọc dữ liệu từ file csv dữ liệu đã chuẩn bị sẵn. Dữ liệu(998x14) gồm 14 cột (bao gồm các mục:STT, Đường, khu vực, Người đăng, Điện thoại, Giá, Ngày đăng, Mô tả, Kinh độ, Vĩ độ) và 998 dòng.



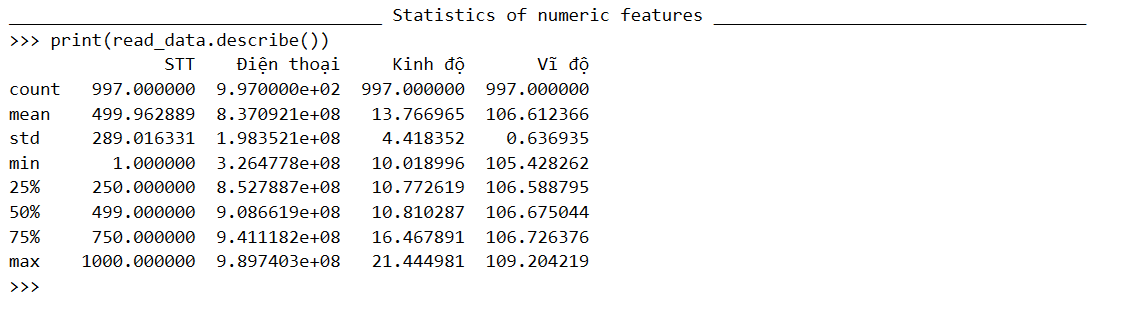
Vấn đề khi load dữ liệu từ cột “Kinh độ” và “Vĩ độ” ở dạng dấu chấm phẩy sẽ chuyển thành dấu chấm do khi sử lý dữ liệu từ excel sang csv dấu chấm sẽ bị lỗi (106.708549 sẽ chuyển thành 106.708.549). Để sử lý dữ liệu ta sử dụng hàm Clean để chuyển dữ liệu chuyển dấu ; (dữ liệu ban đầu: 10;731739)thành 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

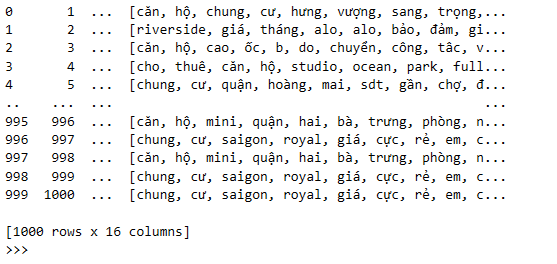
Khi sử lý dữ liệu ở phần “Mô tả”, chứa các thông tin về nhà trọ và được viết như một văn bản, từ văn bản này để cho máy tính hiểu được thì ta sẽ chuyển văn bản về dạng mảng để cho máy tính đọc hiểu. Để giải quyết vấn đề đó thì ta sẽ khởi tạo một hàm clean\_text(text): ở dạng chuỗi là một chuỗi dài gồm các ký tự có dấu nên để sử lý nhóm em sử dụng thư viện “re” để tách chuỗi bao gồm các chữ có dấu.

|  |
| --- |
| 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 |

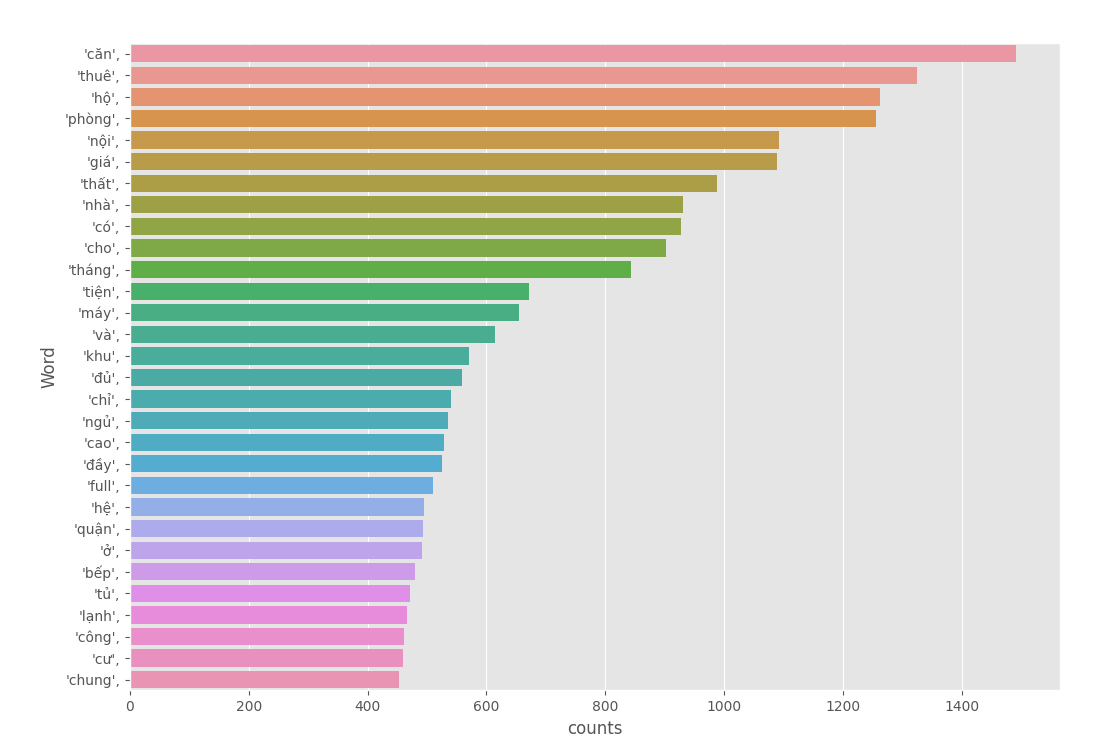
Hàm clean\_text(text) lấy ý tưởng: khi gặp khoảng trắng hoặc ký tự đặc biệt sẽ tách ký tự ra thành một keyword và một đoạn văn sẽ được lưu thành một mảng và các mảng sẽ được lưu vào một cột có tên là “keyword”

|  |
| --- |
| motels['merger table'] = motels['Tiêu đề tin']+' '+motels['Mô tả']  motels['keyword'] = motels['merger table'].apply(lambda x: clean\_text(x)) |

Trong dữ liệu nhóm em tham khảo có 2 yếu tố ảnh hưởng đến bài toán gợi ý là “Tiêu đề tin” và “Mô tả”, “Tiêu để tin” chứa tiêu đề sẽ được gợi ý ra và “ Mô tả” sẽ là phần ghi chú thông tin của “Tiêu đề tin” đó. Bước tiếp theo để thực hiện đưa các chuỗi về mảng, ở bước này sẽ gộp hai cột lại với nhau tạo ra cột “merger table” và tiếp sau đó là thực hiện tách các chữ trong các hàng của cột “merger table”.



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() |

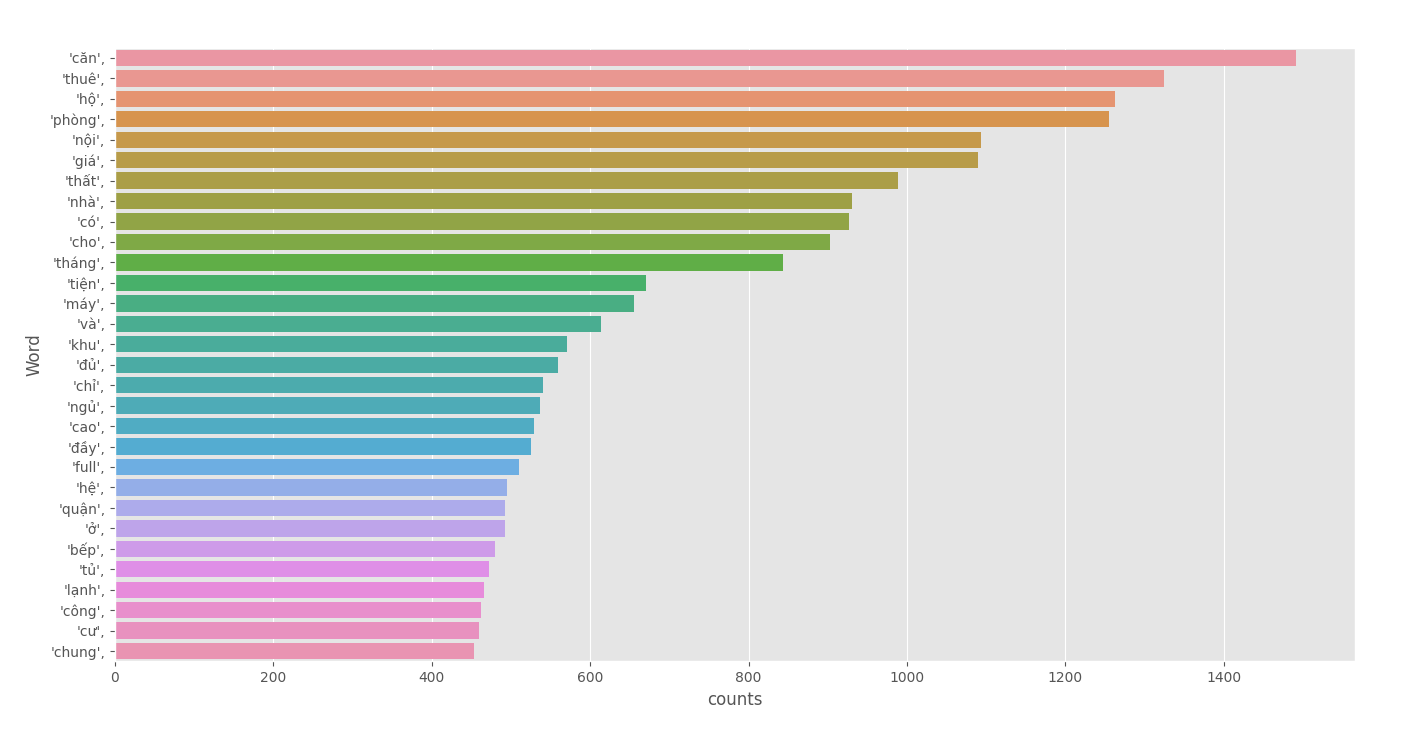
Tiếp theo sau khi thực hiện tìm ra được keyword, tiếp tục thực hiện xóa các từ không liên quan( ở đây các từ không liên quan được gọi là stopword những từ này sẽ làm cho bài toán bị nhiễu). Nhóm em tiếp tục xóa các từ dừng(stopword). Để sử lý vấn đề này thì nltk đã cung cấp một thư viện để thực hiện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

|  |
| --- |
| # Đánh số label  # Training  train\_data = count\_matrix.toarray()  label = np.random.randint(5, size=(1000))  # Call MultinomialNB  clf = BernoulliNB()  clf.fit(train\_data, label)  array = clf.predict(train\_data)  motels['labelrank'] = pd.DataFrame(array, columns=['labelrank']) |

Thực hiện đánh số lable cho từng giá trị train, sau đó ta dùng BernoulliNB()(bộ phân loại này phù hợp với dữ liệu rời rạc. Sự khác biệt là trong khi MultinomialNB hoạt động với số lần xuất hiện, BernoulliNB được thiết kế cho các tính năng nhị phân / boolean.) dữ liệu truyền vào là một countdown vector train\_data và lable của dữ liệu. Kết quả trả về là một mảng gồm các lable của từng tiêu đề tin

# In[3]: Similarity between motels

|  |
| --- |
| from scipy import spatial  def Similarity(motelid1, motelid2):  wordsA = count\_matrix.toarray()[motelid1]  wordsB = count\_matrix.toarray()[motelid2]  wordsDistance = spatial.distance.cosine(wordsA, wordsB)  return wordsDistance |

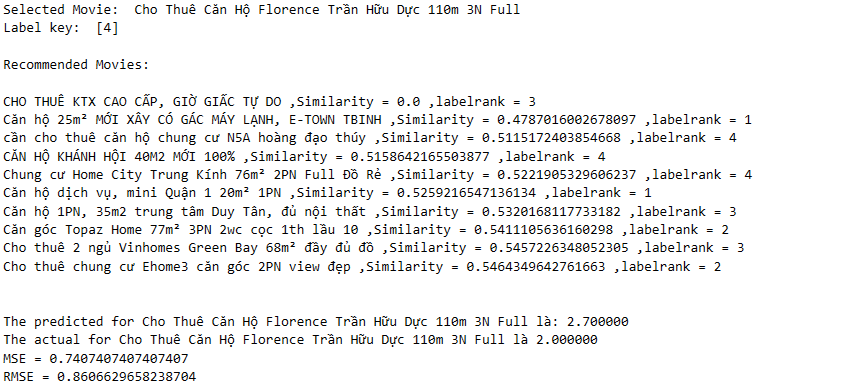
Hàm tính độ giống nhau giữa hai mảng.

# In45]: Score Predictor

|  |
| --- |
| def predict\_score(title, id):  #name = input('Nhập title của nhà trọ: ')  #title = "THIẾT KẾ MỚI - NỘI THẤT ĐẸP - 2PN 65m2 - SAFIRA Q9"  name = title;  try:  new\_motel = motels[motels['Tiêu đề tin'].str.contains(name)].iloc[0].to\_frame().T    original\_title = 'Tiêu đề tin';  print('Selected Movie: ',new\_motel['Tiêu đề tin'].values[0])  array = np.array([count\_matrix\_test.toarray()[id]])  print('Label key: ',clf.predict(array))  def getNeighbors(baseMovie, K):  distances = []    for index, motel in motels.iterrows():  if motel['STT'] != baseMovie['STT'].values[0]:  dist = Similarity(baseMovie['STT'].values[0], motel['STT'] - 1)  distances.append((motel['STT'], dist))    distances.sort(key=operator.itemgetter(1))  neighbors = []    for x in range(K):  neighbors.append(distances[x])  return neighbors  # Lấy 10 motel  K = 10  labelkey = 0  neighbors = getNeighbors(new\_motel, K)  print('\nRecommended Movies: \n')  for neighbor in neighbors:  labelkey = labelkey + motels.iloc[neighbor[0]][16]  print("{} ,Similarity = {} ,labelrank = {}".format(motels.iloc[neighbor[0]][1], neighbor[1], motels.iloc[neighbor[0]][16]))    print('\n')  labelkey = labelkey/K  print('The predicted for %s là: %f' %(new\_motel['Tiêu đề tin'].values[0],labelkey))  print('The actual for %s là %f' %(new\_motel['Tiêu đề tin'].values[0],new\_motel['labelrank']))  mse = float(new\_motel['labelrank'].values[0])/labelkey  rmse = np.sqrt(mse)  print('MSE = {}'.format(mse))  print('RMSE = {}'.format(rmse))  except:  print("Không tìm thể tìm thấy") |

Ở trong hàm predict\_score sẽ thực hiện tìm ra được giá tiền mà căn nhà bạn muốn gợi ý và giá tiền thật của căn nhà đó là bao nhiêu:

getNeighbors là tìm kiếm những tiêu đề có độ giống nhau(similar) cao hoặc là gần nhau



Kết quả thu được sau khi chạy .