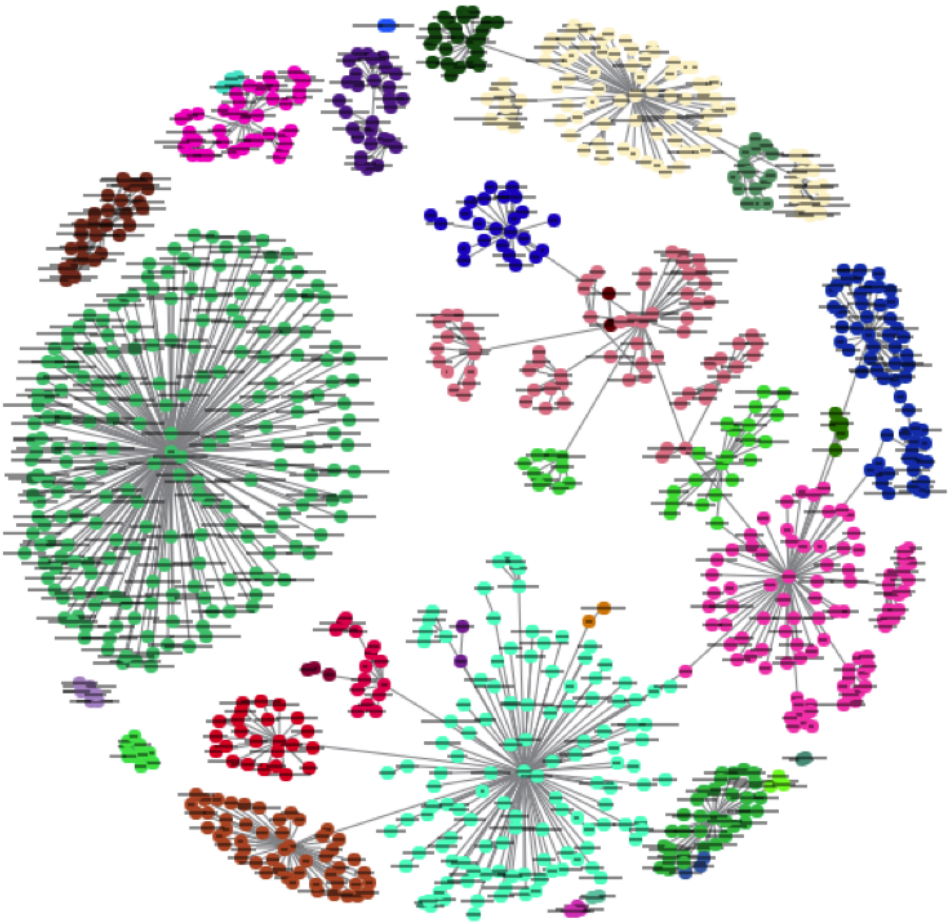


K-nearest neighbors (KNN)

Giáo viên hướng dẫn: Ths Hồ Ngọc Lâm
Nhóm học sinh thực hiện:

1. Huỳnh Chí Tài
2. Nguyễn Quang Anh
3. Văn Công Minh Triết
4. Phạm Hoàng Phúc

Học sinh lớp: 12CTin – Khóa 2021 - 2024

1. Đặt vấn đề 3
2. Cơ sở lý thuyết 3
2.1. Định nghĩa 3
2.2. Thuật toán 3
2.2.1. Ý tưởng 3
2.2.2. Các cách tính khoảng cách đến các điểm dữ liệu 3
2.2.3. Cách chọn giá trị K 4
3. Cách tiếp cận một bài toán sử dụng KNN 4
4. Các bài toán minh họa 5
4.1. Sử dụng thuật toán KNN để dự đoán khách hàng mua (1) hay không mua sản phẩm (0) dựa trên các thông tin được cung cấp 5
4.1.1. Đọc tệp dữ liệu 5
4.1.2. Chọn các giá trị sẽ dùng để dự đoán và chuyển về dạng số hoặc nhị phân 5
4.1.3. Tạo ra bộ dữ liệu để huấn luyện mô hình và bộ dữ liệu để kiểm tra mô hình . 6
4.1.4. Chia tỉ lệ các biến 6
4.1.5. Xét độ chính xác của từng k trong mô hình 6
4.1.6. Huấn luyện mô hình với số k có độ chính xác cao nhất 8
4.2. Sử dụng thuật toán KNN để dự đoán loại kính dựa trên các thông tin được cung cấp 8
4.2.1. Đọc tệp dữ liệu 8
4.2.2. Chọn các giá trị sẽ dùng để dự đoán và chuyển về dạng số hoặc nhị phân 9
4.2.3. Tạo ra bộ dữ liệu để huấn luyện mô hình và bộ dữ liệu để kiểm tra mô hình . 9
4.2.4. Xét độ chính xác của từng k trong mô hình 9
4.2.5. Huấn luyện mô hình với số k có độ chính xác cao nhất 12
5. Tài liệu tham khảo 13
6. Nguồn bài tập tham khảo 13

1. Đặt vấn đề

Trong cuộc sống hiện nay có rất nhiều sự vật cần phân loại để hỗ trợ đời sống con người và giảm chi phí nhân công ví dụ như phân loại email rác và rác thải có thể tái chế. Vì vậy chúng ta cần một mô hình có độ chính xác cao để hỗ trợ chúng ta trong các vấn đề trên và KNN có thể đáp ứng nhu cầu của con người trong một số trường hợp.

2. Cơ sở lý thuyết

2.1. Định nghĩa

K-nearest neighbors (KNN) là một thuật toán thuộc nhóm Supervised Learning được sử dụng trong classification và regression:

- Với classification, đầu ra là giá trị dựa trên KNN trong training data.
- Với regression, đầu ra là trung bình các giá trị của target variable dựa trên KNN trong training data.

2.2. Thuật toán

2.2.1. Ý tưởng

Các điểm dữ liệu được phân loại và xác định loại của một điểm dữ liệu mới, K điểm gần nhất được sử dụng trong quá trình này
Dựa trên giá trị của K mà việc phân loại các điểm dữ liệu có thể được phân loại

2.2.2. Các cách tính khoảng cách đến các điểm dữ liệu

Euclidean distance: Khoảng cách Euclide có thể được hình dung bằng độ dài của đường thẳng nối hai điểm đang xét. Số liệu này giúp chúng tôi tính toán độ dịch chuyển thực được thực hiện giữa hai trạng thái của một vật thể.

$$\mathrm{d(a,b)}^2 = \sum_{i=1}^{n} (a_i - b_i)^2$$

Manhattan distance: Khoảng cách Manhattan thường được sử dụng khi chúng ta quan tâm đến tổng quãng đường mà vật thể đã di chuyển. Số liệu này được tính bằng cách tính tổng chênh lệch tuyệt đối giữa tọa độ của các điểm trong n thứ nguyên.

$$\mathrm{d(a,b)} = \sum_{i=1}^{n} |a_i - b_i|$$

Minkowski distance: Có thể nói rằng khoảng cách Euclide cũng như khoảng cách Manhattan là những trường hợp đặc biệt của khoảng cách Minkowski.

Từ công thức trên, chúng ta có thể nói rằng khi p = 2 thì nó giống như công thức tính khoảng cách Euclide và khi p = 1 thì chúng ta thu được công thức tính khoảng cách Manhattan.
Các số liệu được thảo luận ở trên là phổ biến nhất khi xử lý vấn đề Machine Learning nhưng có các số liệu khoảng cách khác cũng như Khoảng cách Hamming rất hữu ích khi xử lý các vấn đề yêu cầu so sánh chồng chéo giữa hai vectơ có nội dung có thể là boolean cũng như chuỗi các giá trị.

$$\mathrm{d(a,b)} = (\sum_{i=1}^{n} |a_i - b_i|^p)^{\frac{1}{p}}$$

2.2.3. Cách chọn giá trị K

Giá trị của k rất quan trọng trong thuật toán KNN để xác định số lượng lân cận trong thuật toán. Giá trị của k trong thuật toán k-nearest neighbors (KNN) phải được chọn dựa trên dữ liệu đầu vào. Nếu dữ liệu đầu vào có nhiều ngoại lệ hoặc nhiễu hơn thì giá trị k cao hơn sẽ tốt hơn. Nên chọn giá trị lẻ cho k để tránh bị ràng buộc trong phân loại. Các phương pháp xác thực chéo có thể giúp chọn giá trị k tốt nhất cho tập dữ liệu đã cho.
Người ta thường chọn k là $\sqrt{n}$ với n là số mẫu trong bộ dữ liệu vì nếu k quá nhỏ thì tuy xét ít giá trị hơn nhưng mô hình sẽ nhạy cảm với nhiễu, nếu k lớn hơn thì ranh giới mượt hơn, tốt cho viêc khái quát hơn nhưng chỉ khi cục bộ được bảo tồn và k xét quá nhiều giá trị.
Không chọn k = $\frac{n}{2}$ vì dù $\frac{n}{2}$ trông có vẻ ở giữa và cân bằng hơn nhưng nếu xét hàm số thì ta thấy cả 2 đều đồng biến nhưng $\sqrt{n}$ sẽ tăng chậm hơn điều đó đồng nghĩ với việc $\sqrt{n}$ sẽ tốt hơn trong việc kiểm soát dị thường trong dữ liệu

3. Cách tiếp cận một bài toán sử dụng KNN

Bước 1: Đọc tệp dữ liệu
Bước 2: Xử lý các giá trị bị thiếu bằng hai cách:

- Thay các giá trị bị thiếu đó bằng giá trị 0.
- Thay các giá trị bị thiếu đó bằng giá trị trung bình.

Bước 3: Chọn các giá trị sẽ dùng để dự đoán và chuyển về dạng số hoặc nhị phân ví dụ:

- Thay các biến chữ như đồng, kẽm, vàng về khối lượng mol (kiểu dữ liệu số).
- Giới tính chuyển về dạng nhị phân 0 là nữ và 1 là nam hoặc ngược lại.

Bước 4: Tạo ra bộ dữ liệu để huấn luyện mô hình và bộ dữ liệu để kiểm tra mô hình có thể dùng hàm train_test_split được thư viện sklearn cung cấp sẵn.
Bước 5: Chia tỉ lệ các biến để không bị ảnh hưởng nó giúp ngăn chặn các đối tượng có cường độ lớn hơn chi phối các tính toán khoảng cách.
Bước 6: Xét độ chính xác của từng k trong mô hình (xem 2.2.3)
Bước 7: Huấn luyện mô hình với số k có độ chính xác cao nhất.

4. Các bài toán minh họa

4.1. Sử dụng thuật toán KNN để dự đoán khách hàng mua (1) hay không mua sản phẩm (0) dựa trên các thông tin được cung cấp

Nguồn dữ liệu: *https://www.kaggle.com/datasets/rakeshrau/social-network-ads/*

4.1.1. Đọc tệp dữ liệu

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
```

```
# lấy dữ liệu file Social Netword Ads từ link
https://www.kaggle.com/datasets/rakeshrau/social-network-ads/
# vì không có giá trị null nên chúng ta bỏ qua bước 2
data = pd.read_csv("Social_Network_Ads.csv")
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   User ID          400 non-null    int64
 1   Gender           400 non-null    object
 2   Age              400 non-null    int64
 3   EstimatedSalary  400 non-null    int64
 4   Purchased        400 non-null    int64
dtypes: int64(4), object(1)
memory usage: 15.8+ KB
```

4.1.2. Chọn các giá trị sẽ dùng để dự đoán và chuyển về dạng số hoặc nhị phân

```
from sklearn.preprocessing import LabelEncoder
# chuyển giá trị Male và Female về dạng nhị phân
le = LabelEncoder()
data['Gender_E'] = le.fit_transform(data.Gender)
inputs = data.iloc[:,[2,3,5]]
outputs = data['Purchased']
```

4.1.3. Tạo ra bộ dữ liệu để huấn luyện mô hình và bộ dữ liệu để kiểm tra mô hình

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(inputs, outputs,
 test_size=0.30,
random_state=1)
```

4.1.4. Chia tỉ lệ các biến

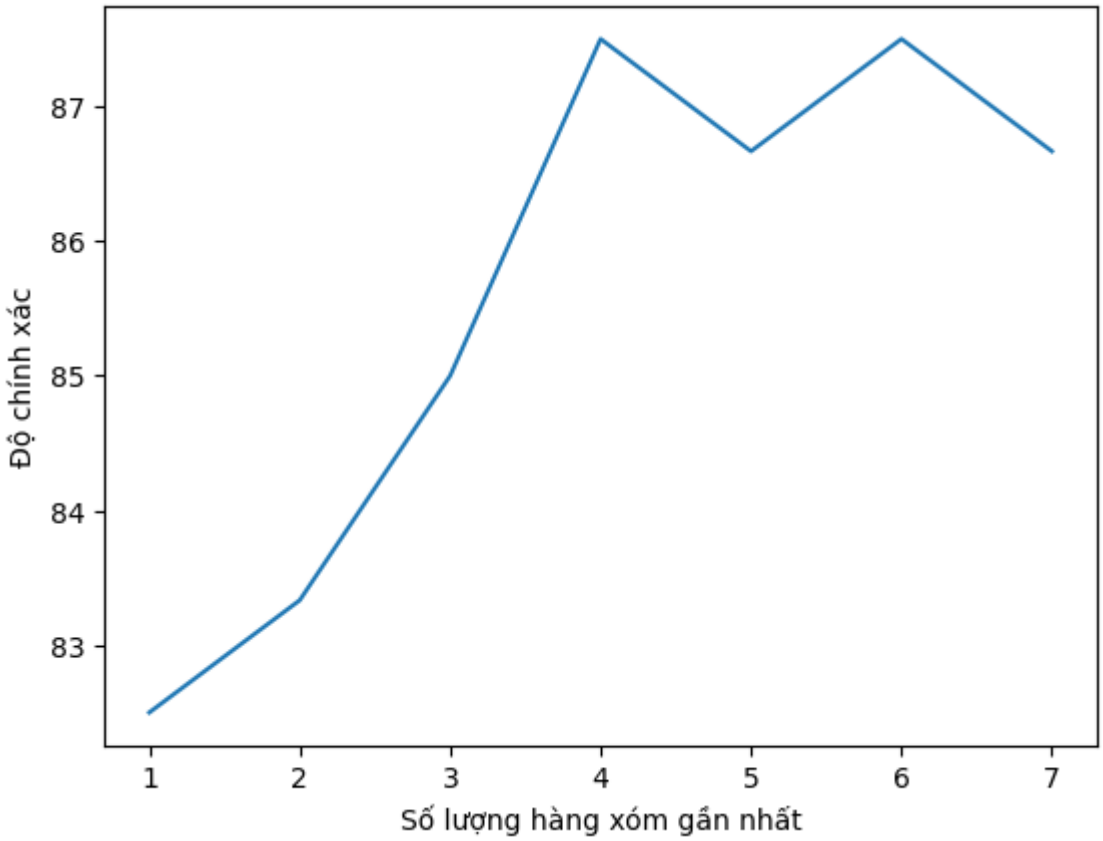
```
from sklearn.preprocessing import StandardScaler
# Cần phải Scale dữ liệu vì Age và EstimatedSalary có thang đo khác nhau
nhiều
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

4.1.5. Xét độ chính xác của từng k trong mô hình

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
list_k = []
list_acc = []
for K_value in range(1,8):
   list_k.append(K_value)
   neigh = KNeighborsClassifier(n_neighbors = K_value)
   neigh.fit(X_train, y_train)
   y_pred = neigh.predict(X_test)
   acc = accuracy_score(y_test,y_pred)*100
   list_acc.append(acc)
   print("với k = ", K_value,": độ chính xác là ",
accuracy_score(y_test,y_pred))
```

```
Số lượng hàng xóm tối ưu là 4 với độ chính xác là 87.5 %
k= 4 : độ chính xác dự đoán trên tập huấn luyện là:  93.57142857142857 %
------ Độ chính xác dự đoán trên tập test là:  87.5 %
k= 6 : độ chính xác dự đoán trên tập huấn luyện là:  93.21428571428572 %
------ Độ chính xác dự đoán trên tập test là:  87.5 %
```

```
plt.plot(list_k, list_acc)
plt.xlabel('Số lượng hàng xóm gần nhất')
plt.ylabel('Độ chính xác')
plt.show()
```




```
from sklearn.neighbors import KNeighborsClassifier
# vì 4 với 6 có độ chính xác bằng nhau trong test mẫu nên ta thử với bộ data
dùng để huấn luyện
for i in (4,6):
   knn = KNeighborsClassifier(n_neighbors=i)
   knn.fit(X_train, y_train)
   print("k=", i, ": độ chính xác dự đoán trên tập huấn luyện là: ",
   knn.score(X_train,y_train)*100,"%")
   print("------ Độ chính xác dự đoán trên tập test là: ",
   knn.score(X_test,y_test)*100,"%")
```

```
k= 4 : độ chính xác dự đoán trên tập huấn luyện là:  93.57142857142857 %
------ Độ chính xác dự đoán trên tập test là:  87.5 %
k= 6 : độ chính xác dự đoán trên tập huấn luyện là:  93.21428571428572 %
------ Độ chính xác dự đoán trên tập test là:  87.5 %
```

4.1.6. Huấn luyện mô hình với số k có độ chính xác cao nhất

```
from sklearn.neighbors import KNeighborsClassifier
```

```
#vì 4 có độ chính xác được được dự đoán trên tập huấn luyện => chúng ta chọn
4
knn = KNeighborsClassifier(n_neighbors=4)
knn.fit(X_train, y_train)
```

4.2. Sử dụng thuật toán KNN để dự đoán loại kính dựa trên các thông tin được cung cấp

Dữ liệu lấy từ: *https://www.kaggle.com/datasets/uciml/glass*
Thông tin thuộc tính file dữ liệu:
1. Mã số: 1 đến 214
2. RI: chỉ số khúc xạ
3. Na: Natri (đơn vị đo: phần trăm trọng lượng trong oxit tương ứng, cũng như các thuộc tính 4-10)
4. Mg: Magiê
5. Al: Nhôm
6. Si: Silicon
7. K: Kali
8. Ca: Canxi
9. Ba: Bari
10. Fe: Sắt
11. Loại kính: (thuộc tính lớp) -- 1 Building_windows_float_processed -- 2 Building_windows_non_float_processed -- 3 xe_windows_float_processed -- 4 Vehicle_windows_non_float_processed (không có trong cơ sở dữ liệu này) -- 5 hộp đựng -- 6 bộ đồ ăn -
- 7 đèn pha

4.2.1. Đọc tệp dữ liệu

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
```

```
# lấy dữ liệu từ https://www.kaggle.com/datasets/uciml/glass
# vì không có dữ liệu null nên bỏ qua bước 2
```

```
# vì các dữ liệu đều dung kiểu dữ liệu số nên bỏ qua bước 3
# vì các dữ liệu dung chung thông số đo lường nên bỏ qua bước 4
data = pd.read_csv("glass.data.txt", sep=",", header=None)
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 214 entries, 0 to 213
Data columns (total 11 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   0       214 non-null    int64
 1   1       214 non-null    float64
 2   2       214 non-null    float64
 3   3       214 non-null    float64
 4   4       214 non-null    float64
 5   5       214 non-null    float64
 6   6       214 non-null    float64
 7   7       214 non-null    float64
 8   8       214 non-null    float64
 9   9       214 non-null    float64
 10  10      214 non-null    int64
dtypes: float64(9), int64(2)
memory usage: 18.5 KB
```

4.2.2. Chọn các giá trị sẽ dùng để dự đoán và chuyển về dạng số hoặc nhị phân

```
# giá trị dùng để huấn luyện mô hình
inputs = data.iloc[:,1:-1]
outputs = data[10]
outputs = np.array(outputs)
```

4.2.3. Tạo ra bộ dữ liệu để huấn luyện mô hình và bộ dữ liệu để kiểm tra mô hình

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(inputs, outputs,
 test_size=0.30,
random_state=1)
```

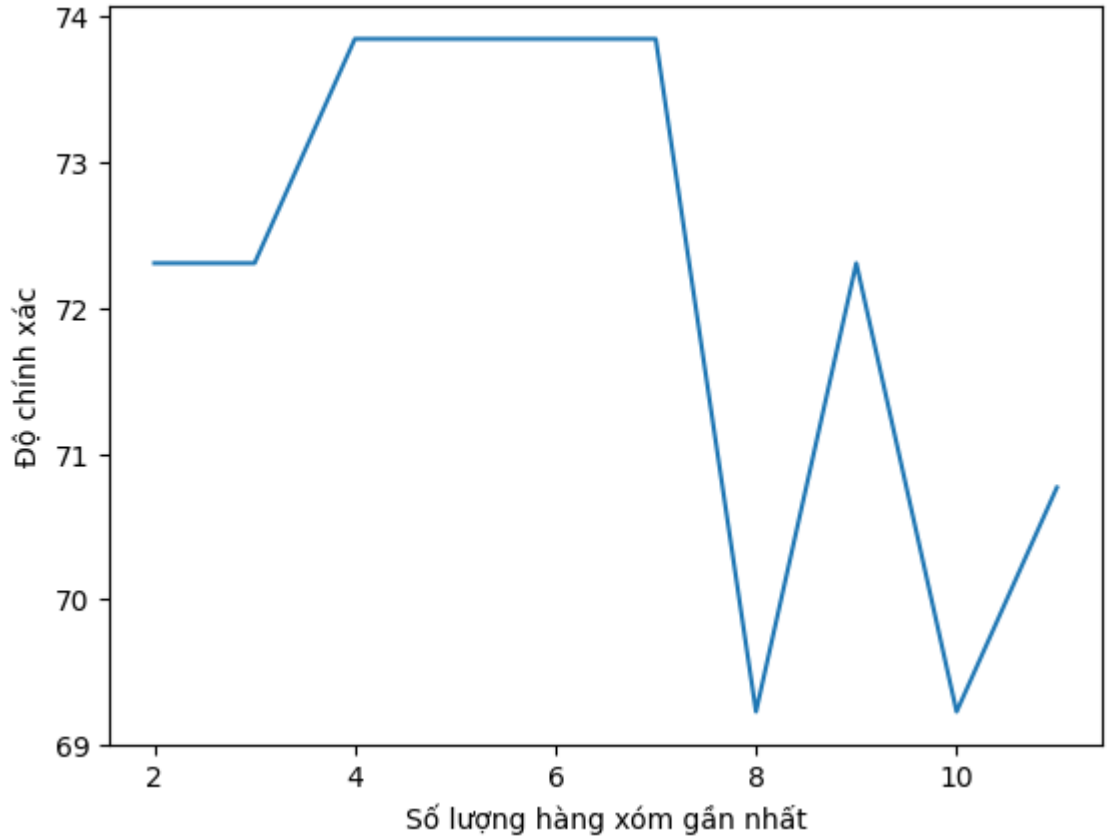
4.2.4. Xét độ chính xác của từng k trong mô hình

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
list_k = []
list_acc = []
```

```
for K_value in range(2,int(y_train.shape[0]**0.5)):
    list_k.append(K_value)
    neigh = KNeighborsClassifier(n_neighbors = K_value)
    neigh.fit(X_train, y_train)
    y_pred = neigh.predict(X_test)
    acc = accuracy_score(y_test,y_pred)*100
    list_acc.append(acc)
    print("k = ", K_value,": độ chính xác là ",
accuracy_score(y_test,y_pred))
```

```
k =  2 : độ chính xác là  0.7230769230769231
k =  3 : độ chính xác là  0.7230769230769231
k =  4 : độ chính xác là  0.7384615384615385
k =  5 : độ chính xác là  0.7384615384615385
k =  6 : độ chính xác là  0.7384615384615385
k =  7 : độ chính xác là  0.7384615384615385
k =  8 : độ chính xác là  0.6923076923076923
k =  9 : độ chính xác là  0.7230769230769231
k =  10 : độ chính xác là  0.6923076923076923
k =  11 : độ chính xác là  0.7076923076923077
```

```
plt.plot(list_k, list_acc)
plt.xlabel('Số lượng hàng xóm gần nhất')
plt.ylabel('Độ chính xác')
plt.show()
```




```
from sklearn.neighbors import KNeighborsClassifier
# vì  4, 5, 6, 7 có độ chính xác bằng nhau trong test mẫu nên ta thử với bộ
data dùng để huấn luyện
for i in (4,5,6,7):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train, y_train)
    print("k=", i, ": độ chính xác dự đoán trên tập huấn luyện là: ",
    knn.score(X_train,y_train)*100,"%")
    print("------ Độ chính xác dự đoán trên tập test là: ",
    knn.score(X_test,y_test)*100,"%")
```

```
k= 4 : độ chính xác dự đoán trên tập huấn luyện là:  76.51006711409396 %
------ Độ chính xác dự đoán trên tập test là:  73.84615384615385 %
k= 5 : độ chính xác dự đoán trên tập huấn luyện là:  72.48322147651007 %
------ Độ chính xác dự đoán trên tập test là:  73.84615384615385 %
k= 6 : độ chính xác dự đoán trên tập huấn luyện là:  71.14093959731544 %
------ Độ chính xác dự đoán trên tập test là:  73.84615384615385 %
```

```
k= 7 : độ chính xác dự đoán trên tập huấn luyện là:  66.44295302013423 %
------ Độ chính xác dự đoán trên tập test là:  73.84615384615385 %
```

4.2.5. Huấn luyện mô hình với số k có độ chính xác cao nhất

```
from sklearn.neighbors import KNeighborsClassifier
```

```
#vì 4 có độ chính xác được được dự đoán trên tập huấn luyện => chúng ta chọn
4
knn = KNeighborsClassifier(n_neighbors=4)
knn.fit(X_train, y_train)
```

5. Tài liệu tham khảo

[1] https://viblo.asia/p/knn-k-nearest-neighbors-1-djeZ14ejKWz, truy cập lần cuối 11h03 ngày 15/10/2023.
[2] https://codelearn.io/sharing/thuat-toan-k-nearest-neighbors-knn, truy cập lần cuối 11h10 ngày 15/10/2023.
[3] https://machinelearningcoban.com/2017/01/08/knn/, truy cập lần cuối 11h18 ngày 15/10/2023.
[4] https://www.ibm.com/topics/knn, truy cập lần cuối 11h27 ngày 15/10/2023.
[5] https://www.geeksforgeeks.org/k-nearest-neighbours/, truy cập lần cuối 22h32 ngày 16/10/2023.

6. Nguồn bài tập tham khảo

[1] Nguồn bài tập 4.1: https://www.kaggle.com/datasets/rakeshrau/social-network-ads, truy cập lần cuối 15h00 ngày 16/10/2023.
[2] Nguồn bài tập 4.2: https://www.kaggle.com/datasets/uciml/glass, truy cập lần cuối 15h30 ngày 16/10/2023.