TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN TP.HCM

KHOA: KHOA HỌC MÁY TÍNH

BÀI TẬP NHÓM

CASE STUDY 1: CLASSIFICATION – TITANIC DATASET

SUBJECT: MACHINE LEARNING

TEACHER: HUỲNH THỊ THANH PHƯƠNG

CLASS: CS114.J22.KHCL

❖ ***Thành viên tham gia:***

1. Huỳnh Minh Tuấn – 17521212
2. Nguyễn Thanh Tú – 17521201
3. Hoàng Ngọc Quân - 17520934
4. Ngô Anh Vũ - 17521272

MỤC LỤC

CHƯƠNG 1: BÀI TOÁN PHÂN LOẠI VÀ ĐÁNH GIÁ SỰ SỐNG SÓT CỦA CÁC HÀNH KHÁCH TRÊN TÀU TITANIC

1.1 Giới thiệu.

+ Hiện nay nhu cầu phân loại và kiểm soát thông tin của hành khách, khách hàng hiện trong các lĩnh vực là rất cần thiết. Song vì số lượng khách hang ngày một tăng lên khiến cho việc quản lý thủ công trở nên khó khăn và mất rất nhiều thời gian cho rất nhiều cửa hàng và doanh nghiệp. Các hệ thống quản lý thông tin khách hàng cần được cải thiện giúp cho việc phân loại và quản lý trở nên nhanh chóng, dễ dàng, hiệu quả hơn. Vì vậy mà nhu cầu phân loại và kiểm soát tự động là rất cần thiết.

+ Xuất phát từ vấn đề đó chúng ta cần phải xây dựng môt mô hình tổ chức quản lý thông tin khách hàng hợp lý hiệu quả hơn. Có rất phương pháp để phân loại khách hàng như là: Support Vector Machine, K-Nearest Neighbor, Naïve Bayes, Decision Tree…Điểm chung của các phương pháp này điều dựa vào xác suất thống kê . Bài toán phân loại và đánh giá sự sống sốt của các hành khách trên tàu TITANIC là một ví dụ cụ thể. Trong bài toán phân loại và đánh giá sự sống sót của các hành khách trên tàu TITANIC này, ta sử dụng 3 phương pháp phổ biến nhất để làm bài toán này: 1KNN-KNN, Naives Bayes và Decision Tree.

1.2 Phát biểu bài toán.

Bài toán phân loại hành khách có thể được phát biểu như sau: Cho trước một tập hành khách D = { d1, d2,…, dn} và tập các kết quả có thể xảy ra C = { c1, c2,...,cn}. Nhiệm vụ của bài toán là gán lớp Di thuộc về Cj đã cho ttrước. Hay nói cách khác là ta đi tìm hàm f:

f: D × C $\longrightarrow$ Boolean

$f(d,c) = \begin{cases} true \\ false \end{cases}$

$f(d,c)$ = true nếu d thuộc về lớp c

$f(d,c)$ = false nếu d không thuộc về lớp c

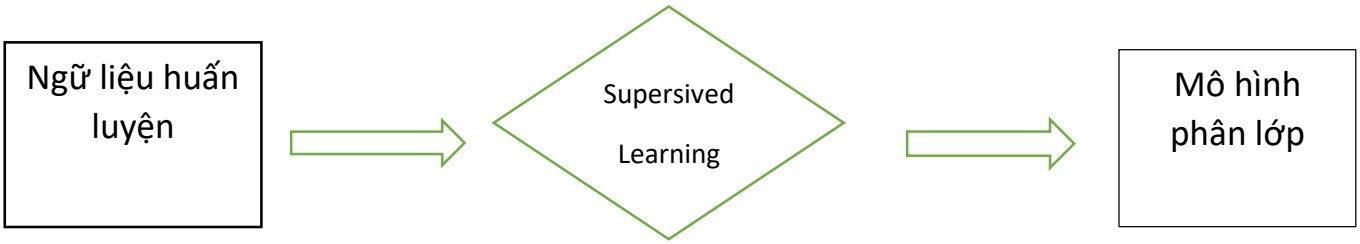
1.3 Mô hình bài toán.

- Có rất nhiều hướng tiếp cận bài toán phân loại hành khách đã được nghiên cứu như: tiếp cận bài toán phân loại dựa trên lí thuyết đồ thị, tiếp cận sử dụng lí thuyết tập thô, cách tiếp cận thống kê,…Tuy nhiên tất cả phương pháp trên điều dựa vào phương pháp chung là máy học, đó là: supervised learning, Unspervised learning, Semi – Supervised learning, Reinforcement learning.
- Vấn đề phân loại hành khách theo phương pháp thống kê dựa trên kiểu học có giám sát được mô tả theo 2 giai đoạn: giai đoạn huấn luyện và giai đoạn phân lớp.
- Giai đoạn huấn luyện:

  + Chúng ta có một tập huấn luyện, mỗi phần tử trong huấn luyện được gán vào một hoặc nhiều lớp mà chúng ta sẽ thể hiện chúng bằng một mô hình mã hóa. Thông thường, mỗi phần tử

trong tập huấn luyện được thể hiện theo dạng $(\vec{x}, c)$. Trong đó, $\vec{x}$ là vector biểu diễn cho các dữ liệu đầu vào trong tập huấn luyện.

+ Sau đó chúng ta định nghĩa một lớp mô hình và một thủ tục huấn luyện. Lớp mô hình là họ các tham số của bộ phân loại, thủ tục huấn luyện là một giải thuật (hay thuật toán) để chọn ra một họ các tham số tối ưu cho bộ phân loại.

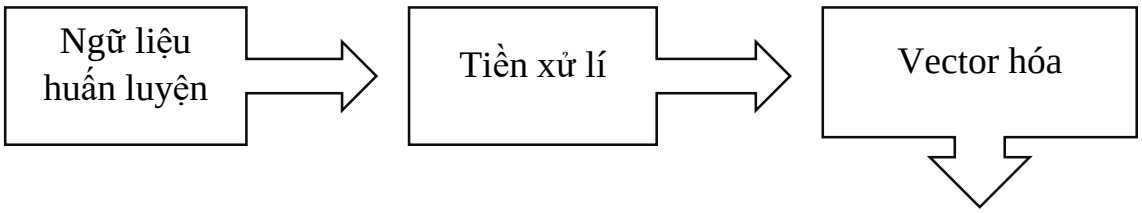


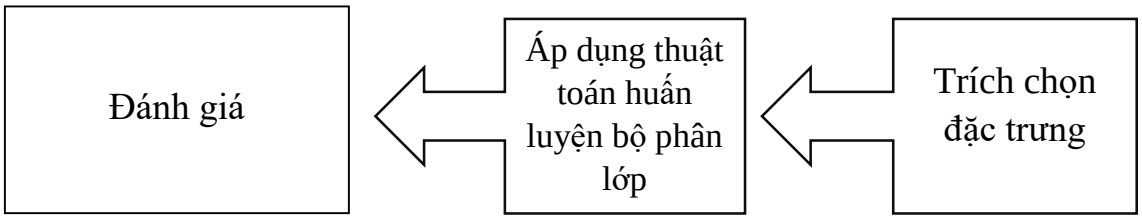
Hình 1.1. Mô hình giai đoạn huấn luyện.

➢ Mô hình giai đoạn huấn luyện.

- Đầu vào: dữ liệu huấn luyện và thuật toán huấn luyện.
- Đầu ra: mô hình phân lớp (bộ phân lớp – classifer).

Các bước trong giai đoạn huấn luyện:





Hình 1.2. Các bước trong giai đoạn huấn luyện.

➢ Các bước trong giai đoạn huấn luyện.

- Trong đó:

+ Ngữ liệu huấn luyện: kho ngữ liệu thu nhập từ nhiều nguồn khác nhau.

+ Tiền xử lý: chuyển đổi dữ liệu trong kho dữ liệu thành một hình thức phù hợp để phân loại.

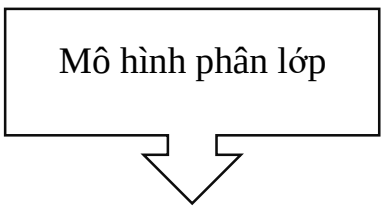
+ Vector hóa: mã hóa các thuộc tính bởi một mô hình trọng số.

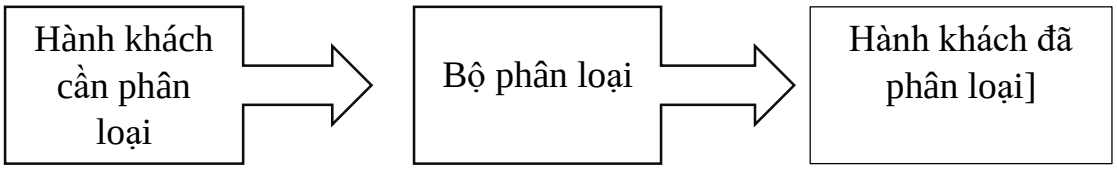
+ Trích chọn đặc trưng: loại bỏ những đặc trưng không mang thông tin khỏi tài liệu nhằm nâng cao hiệu suất phân loại và giảm độ phức tạp của thuật toán huấn luyện.

+ Thuật toán huấn luyện: thuật toán huấn luyện bộ phân lớp để tìm ra họ các tham số tối ưu.

+ Đánh giá: các bước đánh giá hiệu suất của bộ phân lớp.

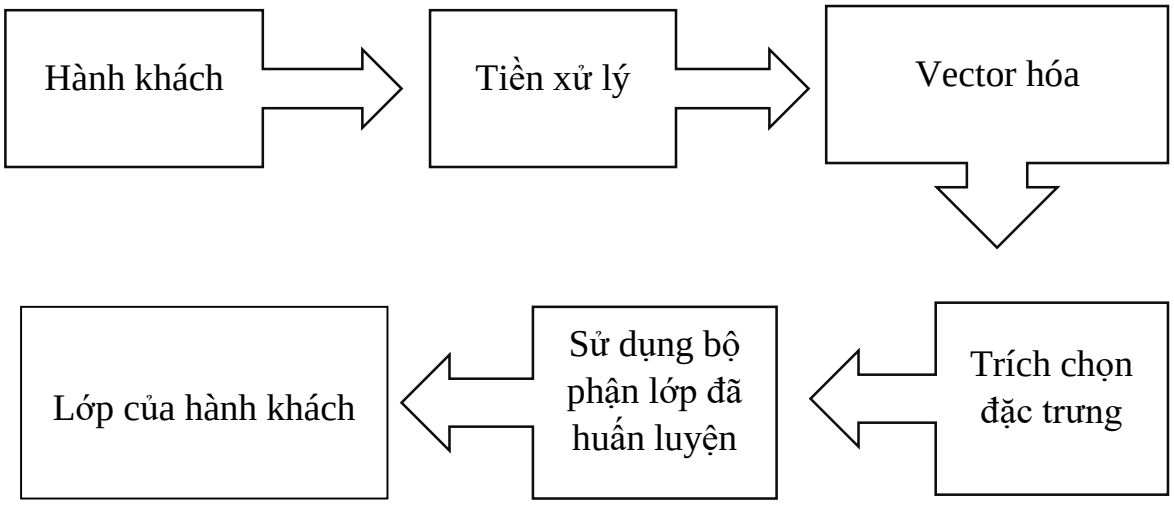
- Giai đoạn phân lớp: Sau khi hoàn thành giai đoạn huấn luyện, mô hình phân lớp sẽ áp dụng cho các hành khách mới cần phân loại.





Hình 1.3. Mô hình giai đoạn phân lớp.

Các bước trong giai đoạn phân lớp:



Hình 1.4. Các bước trong giai đoạn phân lớp.

CHƯƠNG 2: XÂY DỰNG MÔ HÌNH TỪ TẬP HUẤN LUYỆN

2.1. Mã giả thuật toán k-NN và Naïve Bayes:

1. Mã giả thuật toán k-NN:

- Tính toán khoảng cách điểm cần
- dự đoán đến bộ dữ liệu trong tập train
- lấy chỉ số dữ liệu gần x nhất
- Đếm những điểm dữ liệu mà nhãn là 0 trong tập nearest

```
dem=0
for i
if i == 0
dem ++
```

- Quyết định nhãn mới của dữ liệu cần dự đoán

```
If dem0>dem1
Return 0
else
Return 1
```

- Dự đoán trên tập testing và đếm số lượng điểm dự đoán

```
dem = 0
for i, t trong tất cả các dòng thuộc tính của testing
if dự đoán với t,k == nhãn testing thứ i
dem++
return dem // in ra tỷ lệ dự đoán
```

- Kiểm tra k = 1 – 100:

list_acc = [] // danh sách tỷ lệ của mỗi k

for k in range(1,101): //biểu diễn các giá trị tìm được trên đồ thị

tính tỷ lệ mỗi k và cho vào danh sách list_acc

vẽ đồ thị dựa trên danh sách list_acc

2. Mã giả thuật toán Naïve Bayes:

- Đọc dữ liệu từ bộ dataset

- Tính giá trị trung bình và độ lệch chuẩn của các biến dự đoán trong mỗi lớp

- Tính xác suất fi sử dụng phương trình mật độ gauss trong mỗi lớp

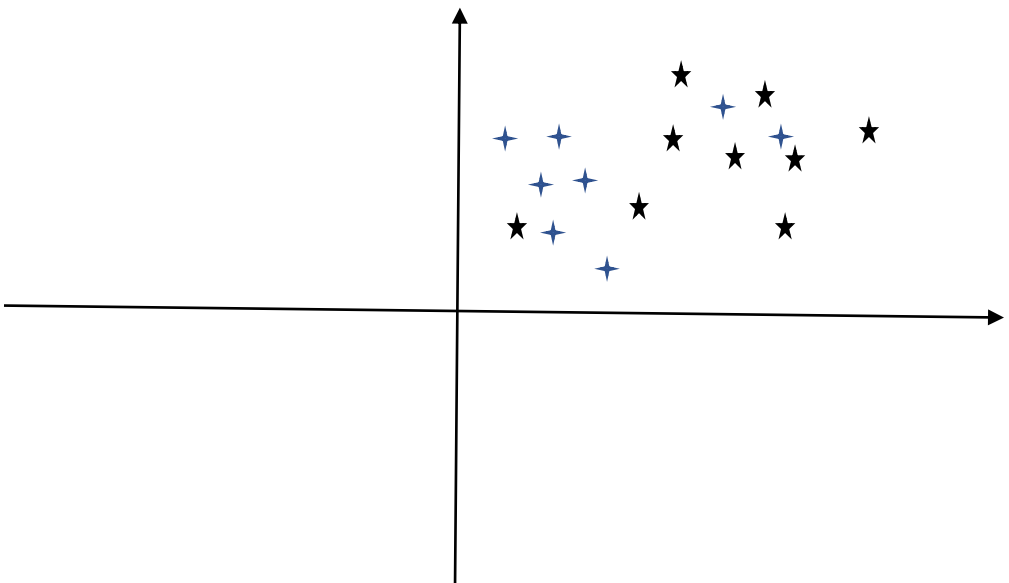
- Lặp lại cho đến khi xác suất của tất cả các biến dự đoán từ f1 đến fn được tính toán hết

- Tính toán khả năng cho mỗi lớp

- Lấy xác suất có khả năng lớn nhất

2.2. Minh họa thuật toán k-NN và Naïve Bayes:

1. Minh họa thuật toán k-NN:



Với ★ Là người còn sống

✦ Là người đã chết

Áp dụng công thức tính khoảng các Manhattan

⇨ Được một điểm nào đó trong đồ thị sẽ có xác suất sống hoặc chết

2. Minh họa thuật toán Naïve Bayes:

khách Loại				Độ tuổi		Giới tính		Yes/No	
1st	2nd	3rd	Crew	Adult	Child	Male	Female	Yes	No
Y:12	Y:7	Y:17	Y:10	Y:42	Y:4	Y:35	Y:11	46	54
N:6	N:18	N:6	N:24	N:45	N:9	N:37	N:17	46/100	54/100
Y:12/46	Y:7/46	Y:17/46	Y:10/46	Y:42/46	Y:4/46	Y:35/46	Y:11/46		
No:6/54	N:18/54	N:6/54	N:24/54	No:45/54	N:9/54	N:37/54	N:17/54		

1 person name: Kha

x(L,Đ,G) = (Y,N,Y)

$P(X) = \frac{46}{100} \times \frac{12}{46} \times \frac{9}{54} \times \frac{11}{46} + \frac{54}{100} \times \frac{12}{54} \times \frac{4}{46} \times \frac{17}{54} \approx 0.008$

$$P(Y|Kha) = \frac{P(Y).P(X|Y)}{P(X)} = \frac{\frac{46}{100} \times \frac{12}{46} \times \frac{9}{54} \times \frac{11}{46}}{0.008} \approx 0.5978$$

$$P(N|Kha) = \frac{P(N).P(X|N)}{P(X)} = \frac{\frac{54}{100} \times \frac{12}{54} \times \frac{4}{46} \times \frac{17}{54}}{0.008} \approx 0.4106$$

⇨ $Kha\ sống$

CHƯƠNG 3: THỰC NGHIỆM ĐÁNH GIÁ

3.1. Xử lý dữ liệu từ bộ dataset

1. Trước tiên, gọi các thư viện cần thiết, bao gồm thư viện numpy để xử lý số và ma trận, thư viện re (regex expression) để xử lý chuỗi.

```
In [1]: import numpy as np
        import re
```

2. Mở dataset và đọc dữ liệu, sau đó visualize dữ liệu

```
In [2]: f = open("./Dataset.data")
        data = f.read()
        data
```

```
yes\n1st  adult female yes\n1st  adult female
adult female yes\n1st  adult female yes\n1st
le yes\n1st  adult female yes\n1st  adult fema
t  adult female yes\n1st  adult female yes\n1s
emale yes\n1st  adult female no\n1st  adult fe
  child male   yes\n1st  child male   yes\n1st
le   yes\n2nd  adult male   yes\n2nd  adult ma
2nd  adult male   yes\n2nd  adult male   yes\n
male   yes\n2nd  adult male   yes\n2nd  adult
nd  adult male   no\n2nd  adult male   no\n2nd
```

3. Vì các điểm dữ liệu tách nhau bằng dấu "\n" nên ta dùng hàm split để tách các điểm dữ liệu ra. Sau đó bỏ đi điểm dữ liệu cuối vì đó là một chuỗi rỗng (sai sót của dataset)

```
In [3]: data = data.split("\n")
        data = data[:-1]
        data
```

```
Out[3]: ['1st  adult male   yes',
         '1st  adult male   yes',
         '1st  adult male   yes',
         '1st  adult male   yes',
         '1st  adult male   yes',
         '1st  adult male   yes',
         '1st  adult male   yes',
         '1st  adult male   yes',
         '1st  adult male   yes'
```

4. Hiện tại, mỗi điểm dữ liệu đại diện bởi một chuỗi, ta cần phải tách các thuộc tính của điểm dữ liệu bằng hàm split và thư viện re

```
In [4]: def data_split(data):
            data = re.sub("\s+"," ",data) #xóa các khoảng trắng dư thừa
            data = data.split()           #tách chuỗi bằng các khoảng trắng
            data = np.array([data])
            return data
        data_split(data[0])
```

```
Out[4]: array([['1st', 'adult', 'male', 'yes']], dtype='<U5')
```

5. Thực hiện tách thuộc tính của tất cả các điểm dữ liệu, sau đó ghép tất cả các điểm dữ liệu cùng thuộc tính thành một mảng 2 chiều với kích thước N x d, với N là số lượng điểm dữ liệu, d là số chiều (số thuộc tính của mỗi điểm dữ liệu)

In [5]:
```
data_np = data_split(data[0])
for i, d in enumerate(data):
    data_np = np.concatenate((data_np,data_split(d)), axis = 0)
data_np
```

Out[5]:
```
array([['1st', 'adult', 'male', 'yes'],
       ['1st', 'adult', 'male', 'yes'],
       ['1st', 'adult', 'male', 'yes'],
       ...,
       ['crew', 'adult', 'female', 'no'],
       ['crew', 'adult', 'female', 'no'],
       ['crew', 'adult', 'female', 'no']], dtype='<U6')
```

In [6]:
```
data_np.shape
```

Out[6]:
```
(2202, 4)
```

6. Thực hiện số hóa tất cả các thuộc tính

In [8]:
```
data_np[:, 0][np.where(data_np[:, 0] == '1st')] = 0
data_np[:, 0][np.where(data_np[:, 0] == '2nd')] = 1
data_np[:, 0][np.where(data_np[:, 0] == '3rd')] = 2
data_np[:, 0][np.where(data_np[:, 0] == 'crew')] = 3
```

In [9]:
```
data_np[:, 1][np.where(data_np[:, 1] == 'adult')] = 0
data_np[:, 1][np.where(data_np[:, 1] == 'child')] = 1
```

In [10]:
```
data_np[:, 2][np.where(data_np[:, 2] == 'male')] = 0
```

In [11]:
```
data_np[:, 2][np.where(data_np[:, 2] == 'female')] = 1
```

In [12]:
```
data_np[:, 3][np.where(data_np[:, 3] == 'yes')] = 0
data_np[:, 3][np.where(data_np[:, 3] == 'no')] = 1
```

```
In [13]: data_np

Out[13]: array([['0', '0', '0', '0'],
                ['0', '0', '0', '0'],
                ['0', '0', '0', '0'],
                ...,
                ['3', '0', '1', '1'],
                ['3', '0', '1', '1'],
                ['3', '0', '1', '1']], dtype='<U6')
```

7. Vì dữ liệu hiện tại vẫn đang ở dạng chuỗi, ta cần ép kiểu chúng về dạng int

```
In [14]: data_np = np.array(data_np,dtype=np.int)
         data_np

Out[14]: array([[0, 0, 0, 0],
                [0, 0, 0, 0],
                [0, 0, 0, 0],
                ...,
                [3, 0, 1, 1],
                [3, 0, 1, 1],
                [3, 0, 1, 1]])
```

8. Chia toàn bộ dữ liệu hiện tại thành các tập dữ liệu cần thiết bao gồm phân ra thuộc tính, nhãn, tập training, tập testing

```
In [15]: ntraining = int(0.8 * data_np.shape[0])
         np.random.shuffle(data_np)            # trộn bộ dữ liệu trước khi phân chia
         training_att = data_np[0:ntraining, 0:3]
         testing_att = data_np[ntraining:,0:3]
         training_label = data_np[0:ntraining,3]
         testing_label = data_np[ntraining:,3]
```

Đã kết thúc công việc xử lý dữ liệu

3.2. Thuật giải k-NN

a. Viết thuật toán dữ liệu với k-NN

In [16]:
```
def predict(x,k):
    """
    dự đoán nhãn mới của 1 điểm dữ liệu x bằng thuật toán k-NN
    """

    distance = (x-training_att)
    distance = np.linalg.norm(distance,axis = 1) # tính toán khoảng cách từ điểm cần
                                                 # dự đoán đến toàn bộ dữ liệu trong tập train

    indices = np.argsort(distance)
    nearest = indices[0:k]          # lấy chỉ số của những điểm dữ liệu gần với x nhất

    # đếm những điểm dữ liệu mà nhãn là 0 trong tập nearest
    dem0 = 0
    for i in nearest:
        if training_label[i] == 0:
            dem0 += 1
    dem1 = k - dem0

    # quyết định nhãn mới của điểm dữ liệu cần dự đoán
    if dem0 > dem1:
        return 0
    else:
        return 1
predict(testing_att[6],k=4)
```

Out[16]: 1

Kiểm tra mô hình với k = 3

In [38]:
```
# thực hiện dự đoán trên tập testing và đếm số lượng điểm dự đoán chính xác
def testAll(k):
    dem =0
    for i, t in enumerate(testing_att):
        if predict(t,k)==testing_label[i]:
            dem += 1

    # in ra tỷ lệ điểm dự đoán chính xác
    return dem / testing_label.shape[0] * 100

testAll(k = 3)
```

Out[38]: 78.00453514739229

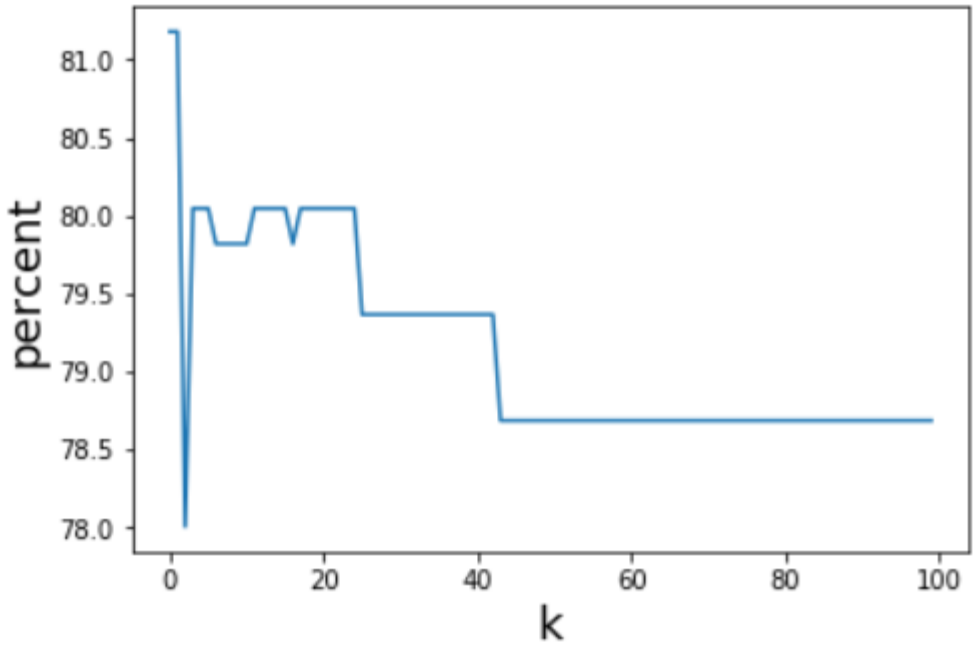
Kiểm tra mô hình lần lượt với k = 1 →100, ta được đồ thị sau:

```
In [22]: import matplotlib.pyplot as plt
```

```
In [25]: list_acc = []
         for k in range(1,101):
             list_acc.append(testAll(k))

         plt.plot(list_acc)
         plt.xlabel('k',fontsize=18)
         plt.ylabel('percent',fontsize=18)
```

```
Out[25]: Text(0, 0.5, 'percent')
```




```
In [28]: testAll(3)
```

```
Out[28]: 78.00453514739229
```

Nhận xét:

- Mô hình có độ chính xác tốt nhất khoảng 79% rơi vào khi k là 1 số nhỏ, độ chính xác sẽ giảm một ít nhưng sẽ ổn định khi k lớn.

- Mô hình có độ chính xác thấp nhất khi k = 3.
- Lưu ý: kết quả mỗi lần chạy sẽ khác do bước trộn dữ liệu trước khi phân chia (tại bước cuối cùng trong quá trình xử lý dữ liệu).

b. Tạo mô hình k-NN bằng thư viện sklearn của python

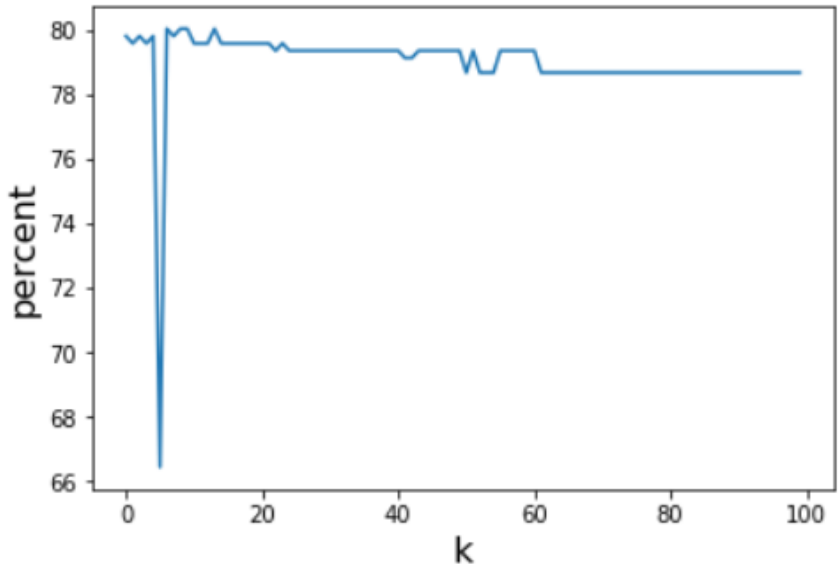
```
In [42]: from sklearn.neighbors import KNeighborsClassifier
```

```
In [43]: knn = KNeighborsClassifier(n_neighbors=3)
         knn.fit(training_att,training_label)
         knn.score(testing_att,testing_label)*100
Out[43]: 79.81859410430839
```

Kiểm tra mô hình lần lượt với k = 1 →100, ta được đồ thị sau:

```
In [44]: list_acc = []
for k in range(1,101):
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(training_att,training_label)
    list_acc.append(knn.score(testing_att,testing_label)*100)
plt.plot(list_acc)
plt.xlabel('k',fontsize=16)
plt.ylabel('percent',fontsize=16)
```

```
Out[44]: Text(0, 0.5, 'percent')
```




```
In [51]: list_acc[5]
```

```
Out[51]: 66.43990929705215
```

Nhận xét:

- Mô hình có độ chính xác tốt nhất khoảng 82% rơi vào khi k = 6, khi k < 6 thì tỉ lệ chính xác rất thấp, khi k > 6, tỉ lệ tốt hơn
- Lưu ý: kết quả mỗi lần chạy sẽ khác do bước trộn dữ liệu trước khi phân chia (tại bước cuối cùng trong quá trình xử lý dữ liệu).

- Nhược điểm lớn nhất của k-NN là kết quả bài toán phụ thuộc rất lớn vào việc chọn tham số k.

3.3. Thuật giải Naïve Bayes

- Tạo mô hình Naïve Bayes với thư viện sklearn của Python

```
In [26]: #Naive bayes với sklearn
         from sklearn.naive_bayes import GaussianNB
         from sklearn import metrics as sk_metrics

         gnb = GaussianNB()
         gnb.fit(training_att,training_label)
         gnb_predictions = gnb.predict(testing_att)
         sk_metrics.accuracy_score(testing_label, gnb_predictions)

Out[26]: 0.7687074829931972
```

Kiểm tra mô hình, ta được kết quả như trên.

- Nhận xét :

  - Mô hình có độ chính xác tốt nhất khoảng 76%. Tuy nhiên, kết quả mỗi lần chạy sẽ khác do bước trộn dữ liệu trước khi phân chia (tại bước cuối cùng trong quá trình xử lý dữ liệu).

  - Độ chính xác của Naïve Bayes nếu so với các thuật toán khác thì không cao.

  - Trong thế giới thực, hầu như bất khả thi khi các feature của dữ liệu test là độc lập với nhau

CHƯƠNG 4: LẬP TRÌNH CÀI ĐẶT

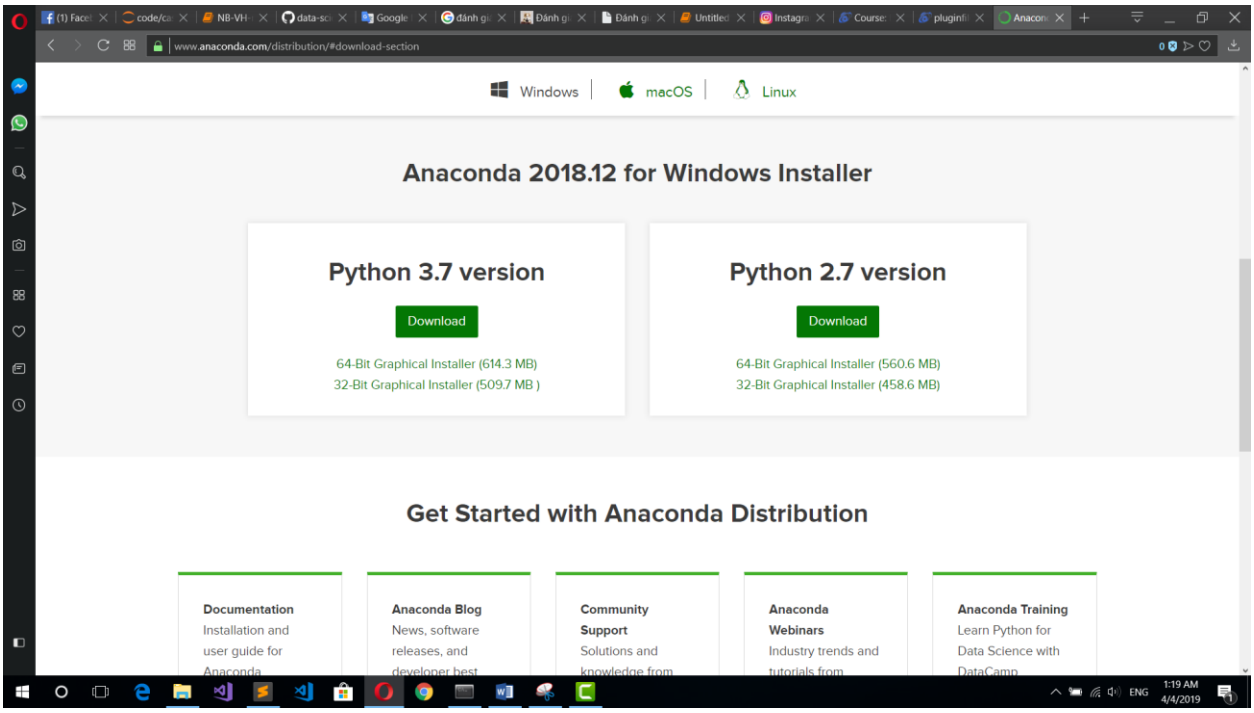
4.1. Tool, thư viện đã sử dụng

- Tool: Anaconda(Jupyter notebook)
- Thư viện: sklearn, numpy, re, sklearn.neigbors, sklearn.naive_bayes
- Ngôn ngữ lập trình: Python

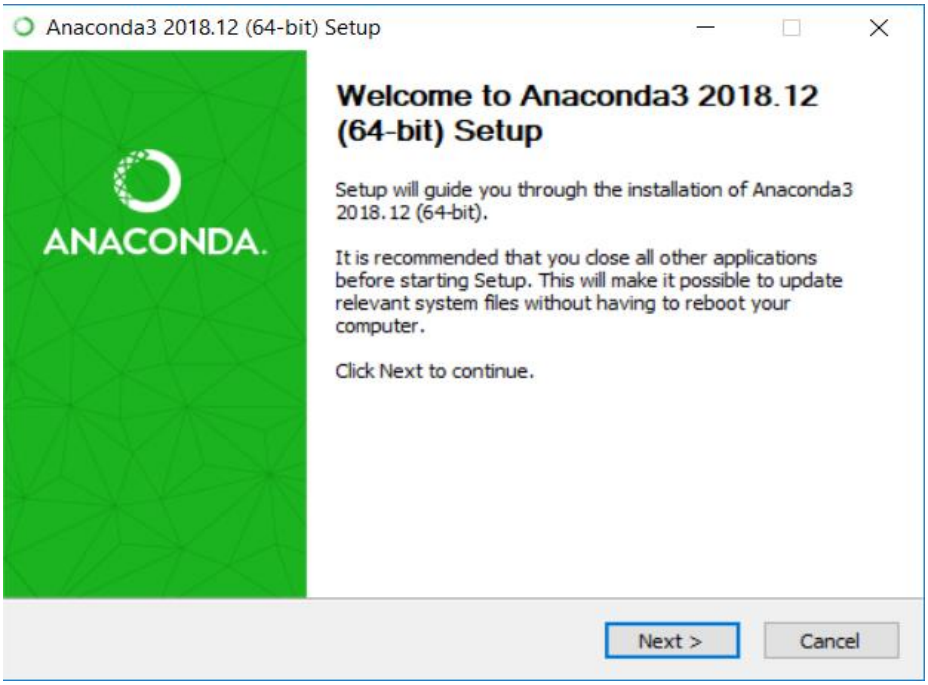
4.2. Hướng dẫn cài đặt Anaconda và các thư viện:

- Download Anaconda tại địa chỉ:

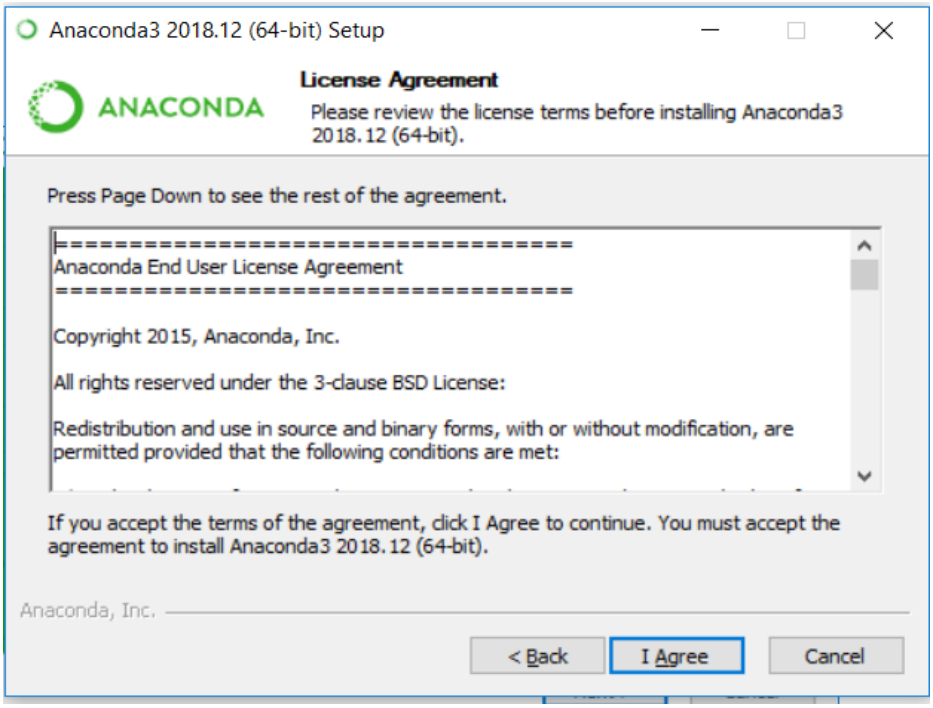
https://www.anaconda.com/distribution/



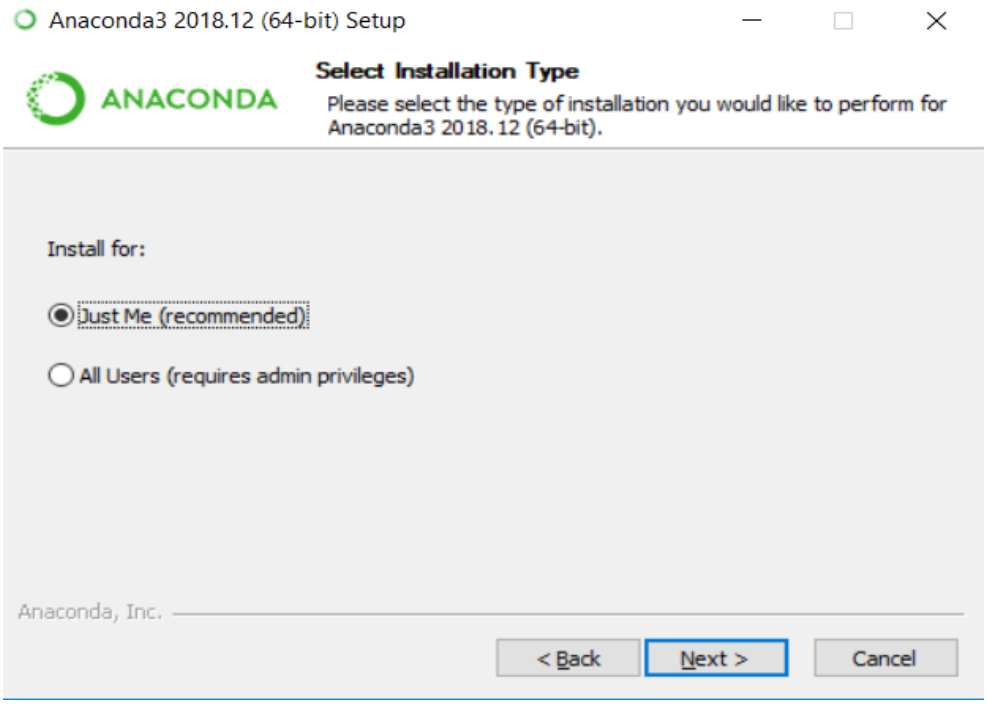
- Chọn phiên bản 2.7 hoặc 3.7, bản 32-bit hoặc 64-bit để tải về.
- Sau khi tải về, tiến hành cài đặt Anaconda.



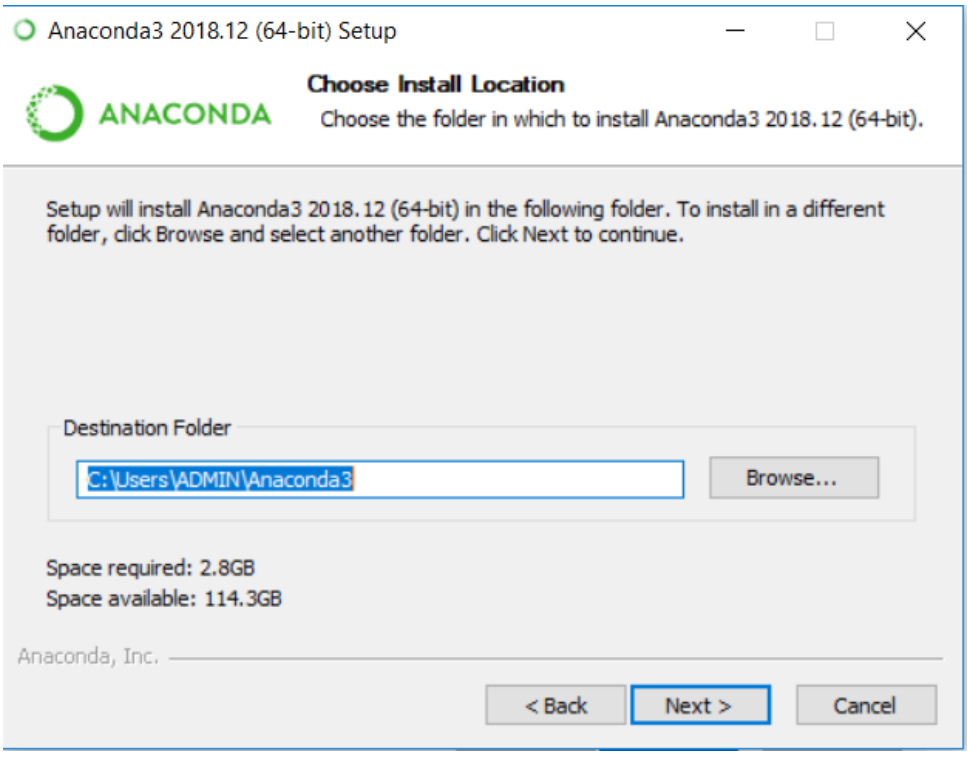
- Tiếp tục bấm Next, sau đó bấm I Agree:



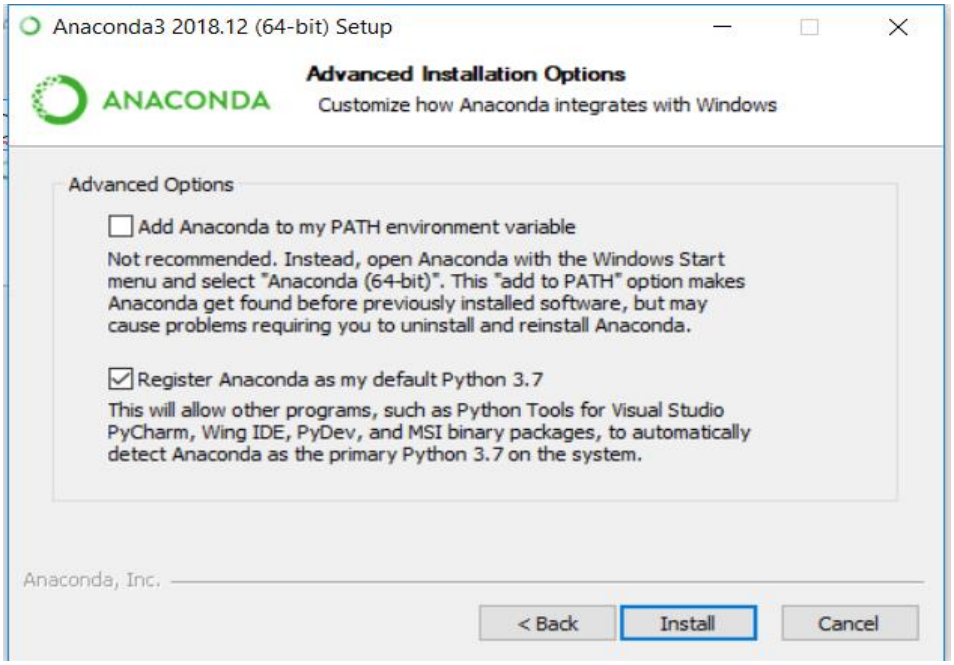
- Tiếp đến chọn Just Me (Nếu chỉ có 1 user trên máy tính), hoặc All users (Tuy nhiên phải cấp quyền cho các admin nếu chọn vào mục này)



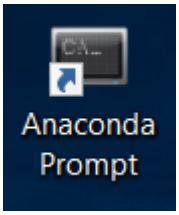
- Chọn đường dẫn cài đặt



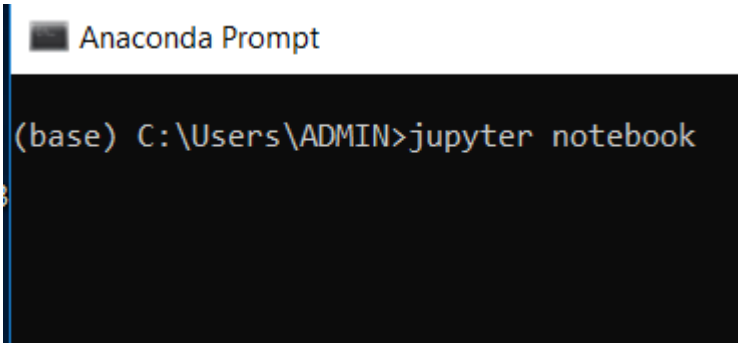
- Chọn Anaconda là môi trường Python hay không. Ở đây khuyến khích chọn như mặc định để tránh bị xung đột với python cài đặt riêng với Anaconda

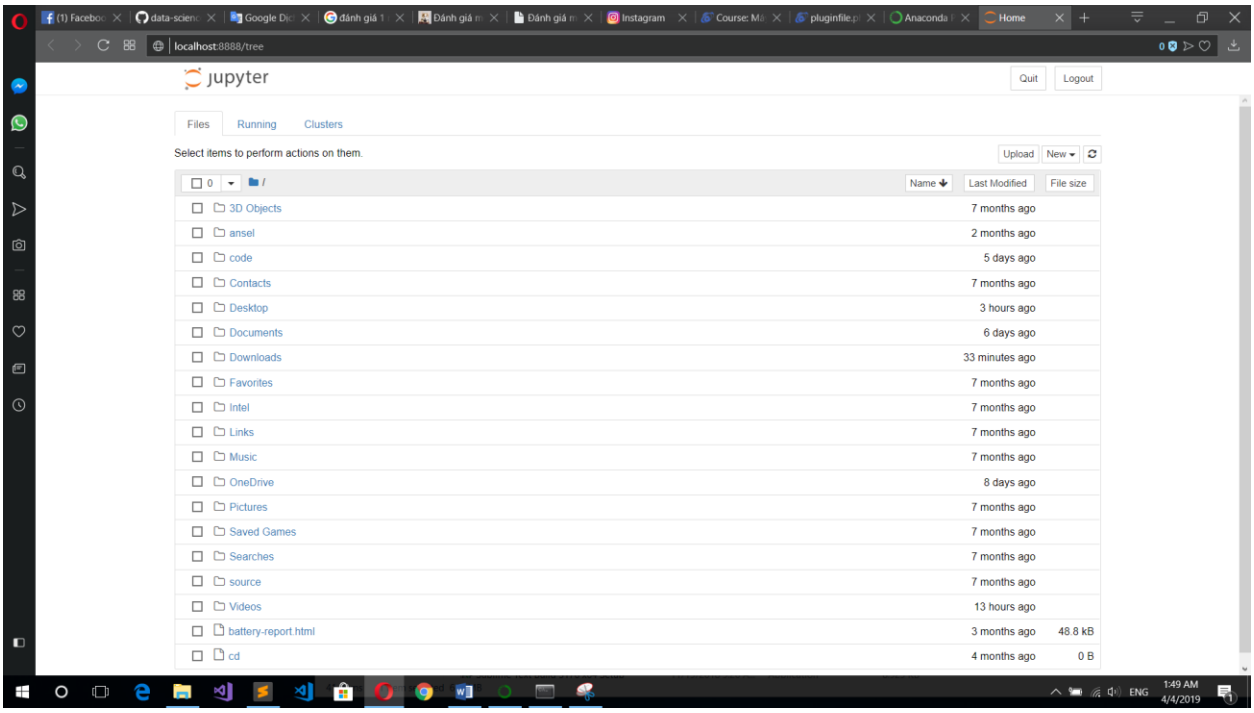


- Tiến hành cài đặt và khởi động Anaconda.
- Để code python trên Anaconda, ta sử dụng Anaconda Prompt

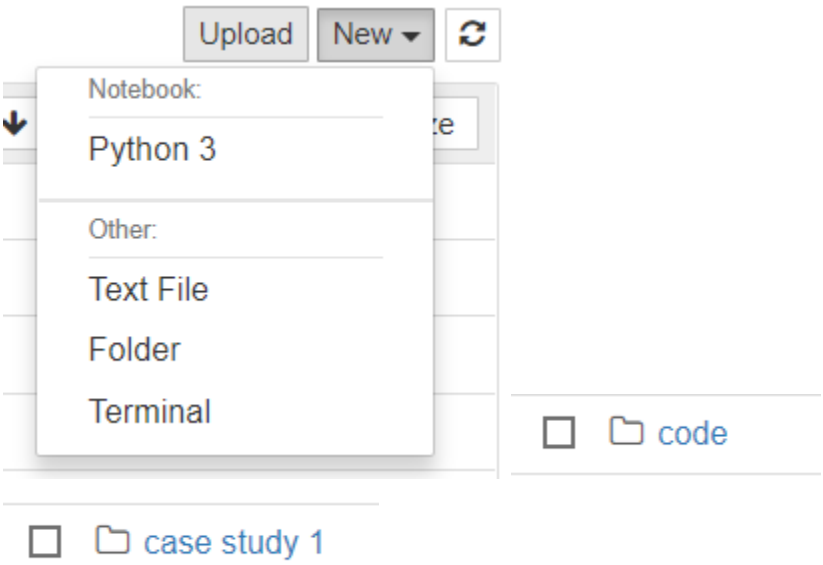


- Mở Anaconda Prompt, gõ jupyter notebook, ngay lập tức sẽ chọn trình duyệt internet mặc định để khởi động jupyter notebook.

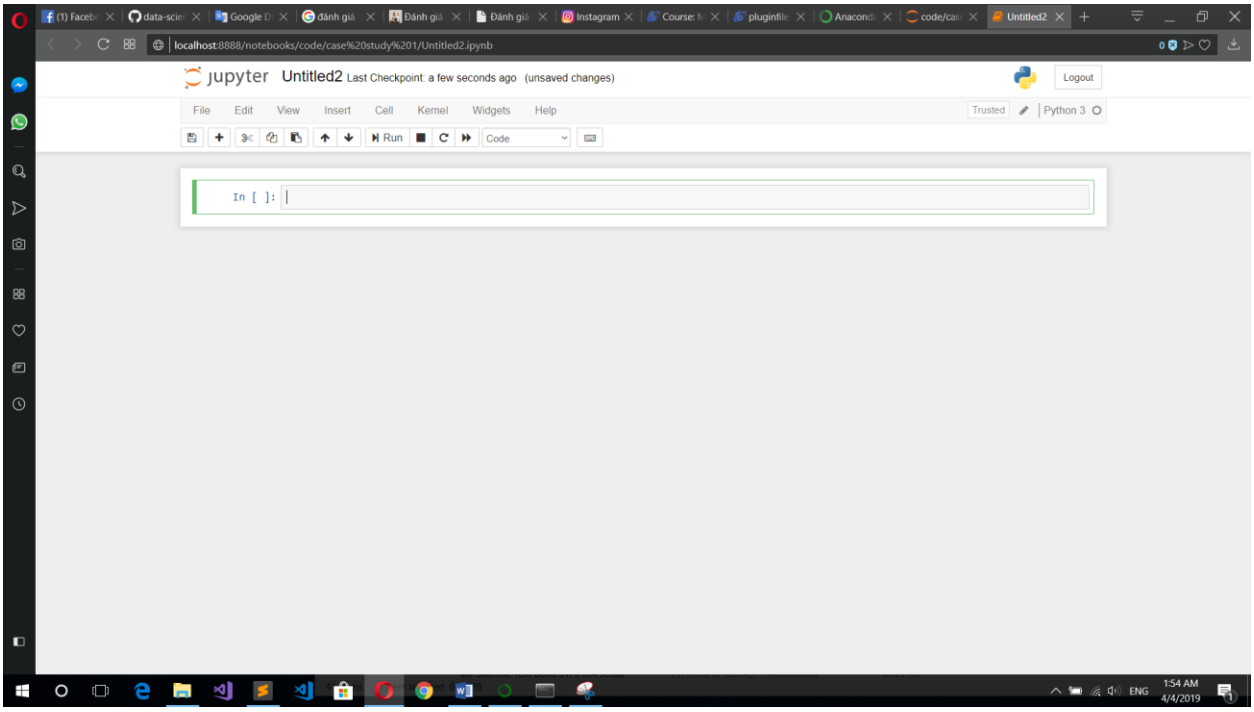




- Tạo thư mục để lưu file làm việc, ở đây nhóm tạo thư mục tên là “code” và thư mục “case study 1” trong thư mục ”code”



- Mở thư mục “case study 1” tạo file python với new -> python 3
- Hệ thống sẽ mở 1 tab mới và tạo nơi để làm việc



- Cách chạy code theo từng block:

  gõ dòng lệnh và nhấn enter để tiếp tục nhập trong 1 block, nhấn shift + enter để thực thi các dòng lệnh trong block

```
In [35]: #Naive bayes với sklearn
         from sklearn.naive_bayes import GaussianNB
         from sklearn import metrics as sk_metrics

         gnb = GaussianNB()
         gnb.fit(training_att,training_label)
         gnb_predictions = gnb.predict(testing_att)
         sk_metrics.accuracy_score(testing_label, gnb_predictions)

Out[35]: 0.7845804988662132
```

Nhấn shift + enter để cho ra kết quả ở dòng out[35]

4.3. Source code báo cáo:

```
import numpy as np #gọi thư viện xử lý số học của python
import re # dùng để tách dữ liệu trong bài này
f = open("./Dataset.data") #mở bộ dữ liệu
data = f.read() # đọc dữ liệu
data
data = data.split("\n") # xóa các dòng \n
data = data[:-1]  # xóa dòng cuối vì dòng cuối rỗng
data
def data_split(data):
    data = re.sub("\s+"," ",data) # xóa các khoảng trắng dư thừa
    data = data.split()          # tách chuỗi bằng các khoảng trắng
    data = np.array([data])
    return data
data_split(data[0])
```

```
data_np = data_split(data[0])
for i, d in enumerate(data):
    data_np = np.concatenate((data_np,data_split(d)), axis = 0)
data_np
data_np.shape #xuất ra tổng số dòng và số cột của bộ dữ liệu
# gán các thuộc tính thành các con số phục vụ cho việc học
data_np[:, 0][np.where(data_np[:, 0] == '1st')] = 0
data_np[:, 0][np.where(data_np[:, 0] == '2nd')] = 1
data_np[:, 0][np.where(data_np[:, 0] == '3rd')] = 2
data_np[:, 0][np.where(data_np[:, 0] == 'crew')] = 3
data_np[:, 1][np.where(data_np[:, 1] == 'adult')] = 0
data_np[:, 1][np.where(data_np[:, 1] == 'child')] = 1
data_np[:, 2][np.where(data_np[:, 2] == 'male')] = 0
data_np[:, 2][np.where(data_np[:, 2] == 'female')] = 1
data_np[:, 3][np.where(data_np[:, 3] == 'yes')] = 0
data_np[:, 3][np.where(data_np[:, 3] == 'no')] = 1
data_np
data_np = np.array(data_np,dtype=np.int) # xử lý các số dạng chuỗi thành int
data_np
# Chia dataset thành training set và testing set
ntraining = int(0.8 * data_np.shape[0]) # lấy 80% dữ liệu cho training, 20% cho testing
```

```
np.random.shuffle(data_np)            # trộn bộ dữ liệu trước khi
phân chia
training_att = data_np[0:ntraining, 0:3]
testing_att = data_np[ntraining:,0:3]
training_label = data_np[0:ntraining,3]
testing_label = data_np[ntraining:,3]
training_att.shape # in ra số dòng và cột các thuộc tính training
# k-NN
# Cách 1: hàm dự đoán nhãn mới của 1 điểm dữ liệu x
def predict(x,k):
    """
    dự đoán nhãn mới của 1 điểm dữ liệu x bằng thuật toán k-NN
    """

    distance = (x-training_att)
    distance = np.linalg.norm(distance,axis = 1) # tính toán khoảng
cách từ điểm cần
                                    # dự đoán đến toàn bộ dữ liệu trong
tập train

    indices = np.argsort(distance)
    nearest = indices[0:k]        # lấy chỉ số của những điểm dữ liệu
gần với x nhất
```

```
    # đếm những điểm dữ liệu mà nhãn là 0 trong tập nearest
    dem0 = 0
    for i in nearest:
        if training_label[i] == 0:
            dem0 += 1
    dem1 = k - dem0

    # quyết định nhãn mới của điểm dữ liệu cần dự đoán
    if dem0 > dem1:
        return 0
    else:
        return 1
predict(testing_att[6],k=3) # test với thuộc tính thứ 6 với k =3
testing_label # dùng để test hàm predict
# thực hiện dự đoán trên tập testing và đếm số lượng điểm dự đoán chính xác
def testAll(k):
    dem =0
    for i, t in enumerate(testing_att):
        if predict(t,k)==testing_label[i]:
            dem += 1
```

```
    # in ra tỷ lệ điểm dự đoán chính xác
    return dem / testing_label.shape[0] * 100

testAll(k = 3)
import matplotlib.pyplot as plt # gọi thư viện đồ thị
# thực hiện đưa tất cả các thuộc tính test lên đồ thị
list_acc = []
for k in range(1,101):
    list_acc.append(testAll(k))

plt.plot(list_acc)
plt.xlabel('k',fontsize=18)
plt.ylabel('percent',fontsize=18)
# Cách 2: Tạo mô hình Naïve Bayes với thư viện sklearn của
Python
from sklearn.neighbors import KNeighborsClassifier
# sử dụng thư viện KNeighborsClassifier từ sklearn.neighbors
list_acc = [] # tương tự như cách 1
for k in range(1,101):
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(training_att,training_label)
    list_acc.append(knn.score(testing_att,testing_label)*100)
plt.plot(list_acc)
```

```
plt.xlabel('k',fontsize=16)
plt.ylabel('percent',fontsize=16)
list_acc[3] # test tỷ lệ sống sót với k = 3
#Naive bayes với sklearn
#sử dụng GaussianNaiveBayes để giải quyết bài toán
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics as sk_metrics

gnb = GaussianNB()
gnb.fit(training_att,training_label)
gnb_predictions = gnb.predict(testing_att)
#Xuất tỷ lệ sống sót trên toàn bộ dữ liệu test
sk_metrics.accuracy_score(testing_label, gnb_predictions)
```

PHỤ LỤC

Tài liệu tham khảo:

1. *Hàm Naïve Bayes bằng thư viện sklearn:* https://github.com/davidasboth/data-science-learning-club/blob/master/activity-5-naive-bayes/Using%20the%20GaussianNaiveBayes%20class.ipynb?fbclid=IwAR1hVr8M6JDEH5w_462LNMk8pY0R6Z3oY5CIx8UBOYiRg5oRd9C7g2wLn7M
2. *Naïve Bayes:* https://machinelearningcoban.com/2017/08/08/nbc/
3. *K-Nearest Neighbors:* https://machinelearningcoban.com/2017/01/08/knn/

Link video quá trình chạy code(Đăng nhập mail UIT để xem):

https://drive.google.com/open?id=1_8oe8mdqbc5brahCpwMQWAXmyqN1NPAS