

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC



ĐỒ ÁN I
HỒI QUY TUYẾN TÍNH TRONG XỬ LÝ
DỮ LIỆU

Chuyên ngành: Toán tin

Giảng viên hướng dẫn: TS. Đoàn Duy Trung

Sinh viên thực hiện: Hoàng Tiến Đạt

MSSV: 20206231

Lớp: Hệ Thống Thông Tin 01 - K65

Hà Nội, 27 tháng 7 năm 2023

NHẬN XÉT CỦA GIẢNG VIÊN

1. Mục tiêu

(a)

(b)

(c)

2. Nội dung

(a)

(b)

(c)

3. Đánh giá kết quả đạt được

(a)

(b)

(c)

Hà Nội, ngày ... tháng ... năm 2023

Giảng viên hướng dẫn

TS. ĐOÀN DUY TRUNG

Lời Cảm Ơn

Báo cáo này được thực hiện và hoàn thành tại Trường Đại học Bách Khoa Hà Nội, nằm trong nội dung học phần Đồ án I của kì học 2022-2.

Em xin được dành lời cảm ơn trân thành tới TS. Đoàn Duy Trung, là giảng viên đã trực tiếp hướng dẫn và gợi ý cho em đề tài rất thú vị này, đồng thời thầy cũng đã giúp đỡ tận tình và có những góp ý, định hướng bổ ích từ đó có thể hoàn thành báo cáo này một cách tốt nhất.

Hà Nội, tháng 07 năm 2023

Tác giả đồ án

Hoàng Tiến Đạt

Mở đầu

Machine learning hay Học máy là một tập con của trí tuệ nhân tạo (AI), nó được ứng dụng cực kỳ nhiều ở thời điểm hiện tại trong hầu hết tất cả các lĩnh vực. Hệ thống tự tag khuôn mặt trong ảnh của Facebook: bộ lọc spam (thư rác) để phân loại email: Hệ thống đề xuất phim của Netflix,... đó là 1 vài ứng dụng nổi bật của machine learning.

Machine learning là mô hình có khả năng học tập tự động 1 lượng lớn dữ liệu để giải quyết những vấn đề cụ thể, có thể là đưa ra những dự đoán, phân loại. Các thuật toán của phương pháp học máy là các chương trình máy tính có khả năng huấn luyện máy tính dựa trên những dữ liệu, sau đó dần dần cải thiện độ chính xác của nó..

Trong học máy, lĩnh vực mô hình dự đoán (predictive modelling) chủ yếu liên quan đến việc đưa ra dự đoán chính xác nhất có thể và giảm thiểu sai số của mô hình. Hồi quy tuyến tính là 1 phương pháp được nghiên cứu nhiều nhất trong lĩnh vực dự đoán. Nó là một mô hình tuyến tính, ví dụ: một mô hình trong đó giả định mối quan hệ tuyến tính giữa các biến độc lập (x) và biến phụ thuộc duy nhất (y). Nói cách khác, y có thể được tính toán từ sự kết hợp tuyến tính của các biến độc lập (x).

Những phát triển thần kỳ của học máy, trí tuệ nhân tạo đã dẫn đến nhu cầu cao về nhân lực những ngành khoa học dữ liệu và các ngành liên quan trên toàn thế giới cũng như ở Việt Nam. Đòi hỏi nhà phân tích dữ liệu phải thành thạo các công cụ, ví dụ như Python, R,... Trong số này, Python được coi là lựa chọn ưa thích của các nhà phân tích dữ liệu trên toàn cầu.

Ở nội dung trong báo cáo này, chúng ta sẽ tìm hiểu về ngôn ngữ lập trình Python và các thư viện hỗ trợ cho khoa học dữ liệu được phát triển bên trong ngôn ngữ lập trình này. Bên cạnh đó chúng ta sẽ tìm hiểu về mô hình hồi quy tuyến tính có dạng như thế nào, áp dụng mô hình hồi quy tuyến tính để giải bài toán toán dự đoán giá nhà.

Mục lục

Lời Cảm Ơn	i
Mở đầu	ii
1 Mô hình lý thuyết	1
1.1 Thuật toán hồi quy tuyến tính	1
1.1.1 Giới thiệu chung	1
1.1.2 Mô hình hồi quy tuyến tính	2
1.1.3 Giải bài toán hồi quy tuyến tính	2
2 Ngôn ngữ Python và một số thư viện	4
2.1 Ngôn ngữ lập trình Python	4
2.1.1 Biến, kiểu dữ liệu	4
2.1.2 Cấu trúc rẽ nhánh	10
2.1.3 Cấu trúc vòng lặp	14
2.1.4 Hàm(thủ tục)	17
2.2 Các thư viện hỗ trợ trong khoa học dữ liệu	18
2.2.1 Numpy	18
2.2.2 Pandas	21
2.2.3 Matplotlib	24
2.2.4 Seaborn	27

2.2.5	Scikit-learn	30
3	Thực nghiệm với dữ liệu	32
3.1	Dữ liệu khảo sát về một số tòa nhà trong 1 thành phố	32
3.2	Bài toán hồi quy tuyến tính	33
3.2.1	Đọc hiểu dữ liệu	33
3.2.2	Trực quan hóa dữ liệu	34
3.2.3	Tiền xử lý dữ liệu	36
3.2.4	Chia tập dữ liệu	37
3.2.5	Huấn luyện mô hình hồi quy tuyến tính	38
3.3	Đánh giá mô hình	38
4	Kết luận	40
4.0.1	Kết luận	40
4.0.2	Hướng phát triển	40
	Tài liệu tham khảo	42

Chương 1

Mô hình lý thuyết

1.1 Thuật toán hồi quy tuyến tính

1.1.1 Giới thiệu chung

Hồi quy tuyến tính là một thuật toán học máy. Nó chủ yếu được sử dụng để dự đoán và tìm ra mối quan hệ giữa các biến khác nhau. Mô hình hồi quy dự đoán một giá trị mục tiêu hay còn gọi là biến phụ thuộc dựa trên các biến độc lập. Biến độc lập là biến đứng một mình, không bị ảnh hưởng bởi biến khác. Khi biến độc lập được điều chỉnh, giá trị của biến phụ thuộc sẽ dao động. Biến phụ thuộc là biến đang được nghiên cứu và đó là những gì mô hình hồi quy cố gắng dự đoán. Trong các tác vụ hồi quy tuyến tính, mọi quan sát bao gồm cả giá trị biến phụ thuộc và giá trị biến độc lập.

Hồi quy tuyến tính được sử dụng trong nhiều lĩnh vực khác nhau, bao gồm tài chính, kinh tế và tâm lý học, để hiểu và dự đoán hành vi của một biến cụ thể. Ví dụ, trong tài chính, hồi quy tuyến tính có thể được sử dụng để hiểu mối quan hệ giữa giá cổ phiếu của công ty và thu nhập của nó hoặc để dự đoán giá trị tương lai của một loại tiền tệ dựa trên hiệu suất trong quá khứ của nó.

1.1.2 Mô hình hồi quy tuyến tính

Mô hình hồi quy tuyến tính dạng tổng quát được biểu diễn qua phương trình:

$$y \approx \hat{y} = \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n + \beta_0 \quad (1.1)$$

Trong đó:

y : giá trị thực tế

\hat{y} : Giá trị dự đoán

$x_i (i = 1, 2, \dots, n)$: Là giá trị các biến độc lập

$\beta_i (i = 0, 1, 2, \dots, n)$: hệ số hồi quy, β_0 còn được gọi là bias

Trường hợp $i=1$, phương trình 1.1 trở thành phương trình cho mô hình hồi quy đơn biến:

$$y = \beta_0 + \beta_1 \times x$$

Trong không gian hai chiều, một hàm số được gọi là tuyến tính nếu đồ thị của nó có dạng một đường thẳng. Trong không gian ba chiều, một hàm số được gọi là tuyến tính nếu đồ thị của nó có dạng một mặt phẳng. Trong không gian nhiều hơn 3 chiều, một hàm số được gọi là tuyến tính nếu đồ thị của nó có dạng siêu mặt phẳng (hyperplane).

1.1.3 Giải bài toán hồi quy tuyến tính

Đối với mô hình hồi quy tuyến tính, ta cần ước lượng giá trị của các hệ số được sử dụng trong mô hình mà chúng ta có. Hai phương pháp được sử dụng nhiều nhất:

- Phương pháp bình phương nhỏ nhất (Ordinary least squares)

Nguyên tắc của phương pháp này là tìm giá trị của hệ số sao cho tổng bình phương các phần dư (residuals) là nhỏ nhất. Cách tiếp cận này coi dữ liệu như một ma trận và sử dụng các phép toán đại số tuyến tính để ước tính các giá trị tối ưu cho các hệ số

- Phương pháp Gradient Descent

Gradient Descent là một phương pháp phổ biến trong học máy (machine learning). Phương pháp này có thể sử dụng với bài toán hồi quy tuyến tính khi có một hoặc nhiều biến đầu vào (x). Ý tưởng là tối ưu hóa giá trị của các hệ số Beta bằng cách thử rất nhiều các giá trị của hệ số trên dữ liệu đào tạo (training data) và chọn ra các giá trị sao cho sai số của mô hình là nhỏ nhất.

Tuy nhiên trong báo cáo này chúng ta sẽ chỉ tìm hiểu về cách sử dụng phần mềm thống kê cụ thể là mô hình hồi quy tuyến tính (model linear regression) của thư viện **scikit-learn** để xây dựng mô hình hồi quy tuyến tính trong phần 2 của báo cáo.

Chương 2

Ngôn ngữ Python và một số thư viện

2.1 Ngôn ngữ lập trình Python

Python là ngôn ngữ lập trình tập trung vào sự đơn giản và dễ sử dụng hướng đối tượng, cấp cao, mạnh mẽ. Ngày nay, Python được sử dụng rộng rãi trong nhiều lĩnh vực và là một trong những ngôn ngữ lập trình được sử dụng nhiều nhất trên thế giới. Kế thừa từ ngôn ngữ lập trình ABC, Python đã được hình thành vào cuối những năm 1980 do Guido van Rossum tạo ra . Python thường được sử dụng trong:

- Xây dựng trang web với các framework như Django và Flask
- Xây dựng game
- Khoa học dữ liệu và học máy(Machine learning)
- Lập trình ứng dụng

2.1.1 Biến, kiểu dữ liệu

Biến

Một biến Python là một khu vực bộ nhớ được dành riêng để lưu trữ các giá trị. Nói cách khác, biến trong python cung cấp dữ liệu cho quá trình xử lý của máy tính. Python cũng giống như một số các ngôn ngữ bậc cao khác, khi ta khai báo biến thì kiểu dữ liệu của nó sẽ tự động được detect. Một biến được tạo ra ngay khi

bạn lần đầu tiên gán một giá trị cho nó.

-Các quy tắc đặt tên biến

- Tên biến phải bắt đầu bằng chữ cái hoặc ký tự gạch dưới(_)
 - Tên biến không thể bắt đầu bằng một số
 - Tên biến chỉ có thể chứa các ký tự chữ và số và dấu gạch dưới (A-z, 0-9 và _)
 - Biến trong Python phải có tên riêng, không trùng lặp với tên của các biến đang tồn tại trên file làm việc của bạn.
 - Tên biến phân biệt chữ hoa chữ thường (name, Name và NAME là ba biến khác nhau)
 - Tên biến không thể là bất kỳ từ khóa Python nào.
- * Chúng ta có thể lấy kiểu dữ liệu của bất kỳ đối tượng nào bằng cách sử dụng hàm: `type()`



The screenshot shows a Python IDE with several tabs open. The active tab is 'KhaiBaobien.py'. The code in the editor is as follows:

```
1 name="Andrew"
2 Name="Lisa"
3 NAME="Messi"
4 _Name7="Ronaldo"
5 print(name)
6 print(Name)
7 print(_Name7)
```

The output console at the bottom shows the results of the print statements:

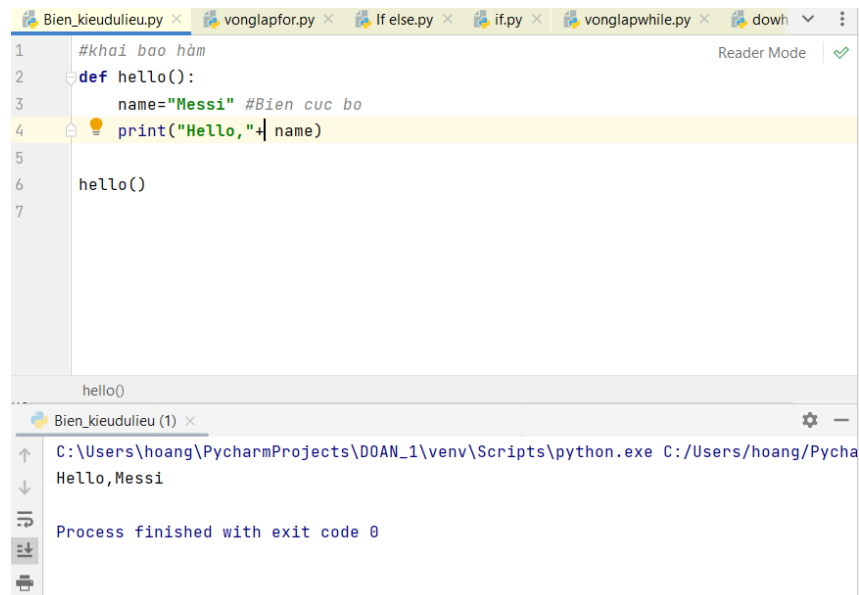
```
Andrew
Lisa
Ronaldo
```

Below the output, it says "Process finished with exit code 0".

Hình 2.1: Ví dụ về biến

-Biến cục bộ và biến toàn cục

+Biến cục bộ(Local variables) là biến được khai báo bên trong 1 hàm

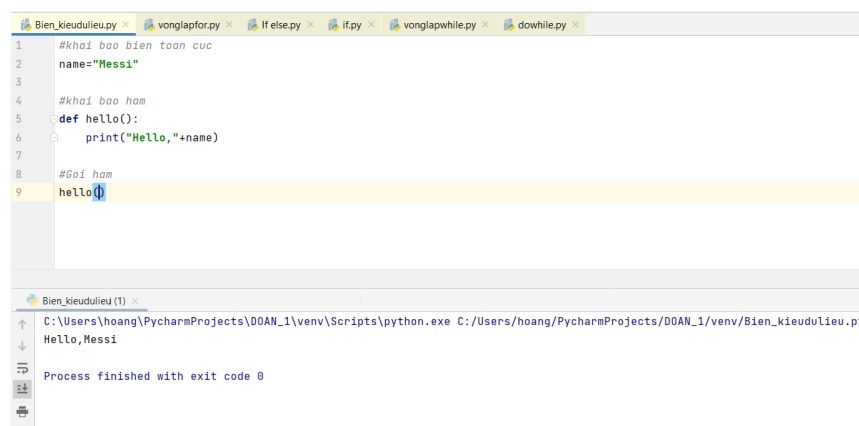


```
1 #khai bao ham
2 def hello():
3     name="Messi" #Bien cuc bo
4     print("Hello,"+ name)
5
6 hello()
7
```

The output console shows the command execution: `C:\Users\hoang\PycharmProjects\DOAN_1\venv\Scripts\python.exe C:/Users/hoang/PycharmProjects/DOAN_1/venv/Bien_kieudulieu.py` and the output `Hello,Messi`. The process finished with exit code 0.

Hình 2.2: Ví dụ biến cục bộ

+Biến toàn cục(global variable) là biến được khai báo bên ngoài hàm.

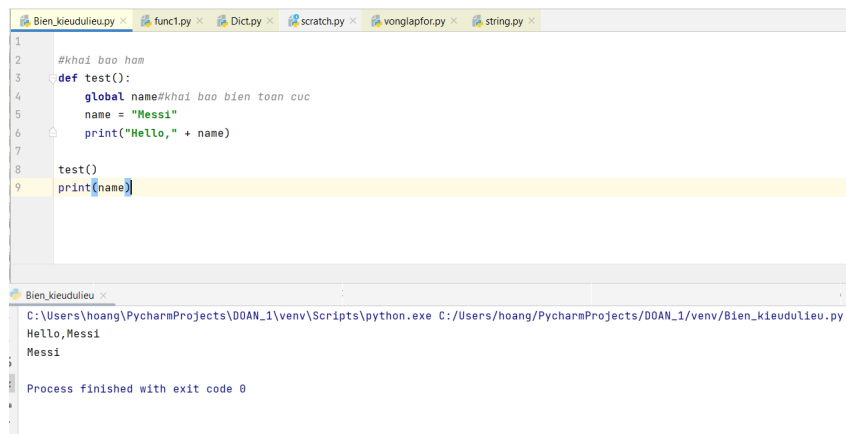


```
1 #khai bao bien toan cuc
2 name="Messi"
3
4 #khai bao ham
5 def hello():
6     print("Hello,"+name)
7
8 #Goi ham
9 hello()
```

The output console shows the command execution: `C:\Users\hoang\PycharmProjects\DOAN_1\venv\Scripts\python.exe C:/Users/hoang/PycharmProjects/DOAN_1/venv/Bien_kieudulieu.py` and the output `Hello,Messi`. The process finished with exit code 0.

Hình 2.3: Ví dụ biến toàn cục

Thông thường, khi chúng ta tạo một biến bên trong một hàm, biến đó là cục bộ, và chỉ có thể được sử dụng bên trong hàm đó. Để tạo một biến toàn cục bên trong một hàm, bạn có thể sử dụng từ khóa **global**



```
1
2 #khai bao ham
3 def test():
4     global name #khai bao bien toan cuc
5     name = "Messi"
6     print("Hello," + name)
7
8 test()
9 print(name)
```

C:\Users\hoang\PycharmProjects\DOAN_1\venv\Scripts\python.exe C:/Users/hoang/PycharmProjects/DOAN_1/venv/Bien_kieudulieu.py

Hello,Messi

Messi

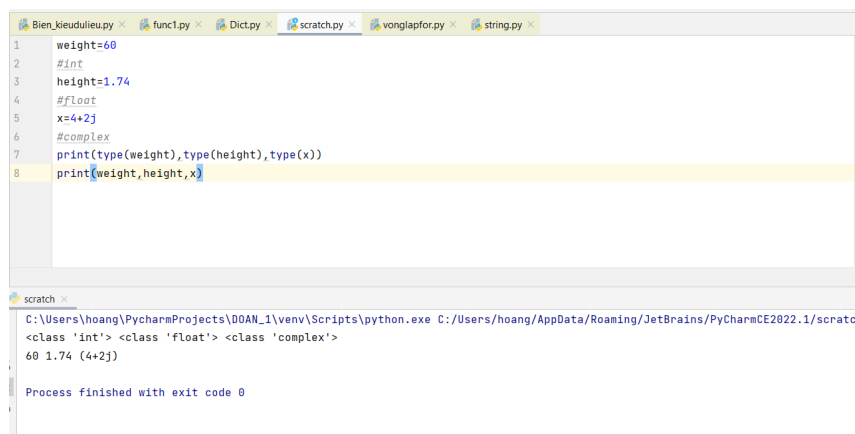
Process finished with exit code 0

Hình 2.4: Ví dụ về từ khóa "Global"

Kiểu dữ liệu

Trong lập trình, kiểu dữ liệu là một khái niệm quan trọng. Dữ liệu được lưu trữ trong bộ nhớ có thể có nhiều kiểu khác nhau, các kiểu khác nhau có thể làm được những thứ khác nhau. Python có các kiểu dữ liệu sau được tích hợp sẵn theo mặc định, trong các danh mục sau:

- Dữ liệu số :Numeric
 - integer(số nguyên):chứa các số nguyên(không có phần thập phân)
 - float(số thực):biểu diễn các số thực với dấu chấm động (có độ chính xác với 15 chữ số ở phần thập phân).
 - complex(số phức):là những số chứa 2 phần,phần thực và phần ảo



```
1 weight=60
2 #int
3 height=1.74
4 #float
5 x=4+2j
6 #complex
7 print(type(weight),type(height),type(x))
8 print(weight,height,x)
```

C:\Users\hoang\PycharmProjects\DOAN_1\venv\Scripts\python.exe C:/Users/hoang/AppData/Roaming/JetBrains/PyCharmCE2022.1/scratc

<class 'int'> <class 'float'> <class 'complex'>

60 1.74 (4+2j)

Process finished with exit code 0

Hình 2.5: Ví dụ kiểu dữ liệuNumeric

- Kiểu boolean: biến chỉ có 2 giá trị là True hoặc False. Trong Python, True được đại diện bởi giá trị 1 và False được đại diện bởi giá trị 0. Thông thường, các giá trị Boolean là True hoặc False không được gán trực tiếp với biến mà thông qua một phép so sánh.

```

1 #khai bao bien
2 x=True
3 y=False
4 print(int(x))
5 print(int(y))
6 # Một phép so sánh có giá trị True/False
7 print(5 < 6) # True
8 print(5 == 6) # False

```

Run: bool

```

C:\Users\hoang\PycharmProjects\DOAN_1\venv\Scripts\python.exe C:/Users/hoang/PycharmProjects/DOAN_1/venv/bool.py
1
0
True
False
Process finished with exit code 0

```

Hình 2.6: Ví dụ kiểu dữ liệu Boolean

- Kiểu String

Một chuỗi ký tự (string) là một chuỗi các ký tự được bao quanh bởi dấu nháy kép "" hoặc dấu nháy đơn '. Chuỗi ký tự (string) trong Python được đại diện bởi lớp str. Trong Python, không có kiểu dữ liệu ký tự (character), mà một ký tự là một chuỗi (string) có độ dài là 1 ký tự.

```

1 #khai bao chuoi
2 chuoi1="Hello,World!"
3 chuoi2='python'
4
5 print(chuoi1)
6 print(chuoi2)

```

Run: string

```

C:\Users\hoang\PycharmProjects\DOAN_1\venv\Scripts\python.exe C:/Users/hoang/PycharmProjects/DOAN_1/venv/string.py
Hello,World!
python
Process finished with exit code 0

```

Hình 2.7: Ví dụ kiểu dữ liệu String

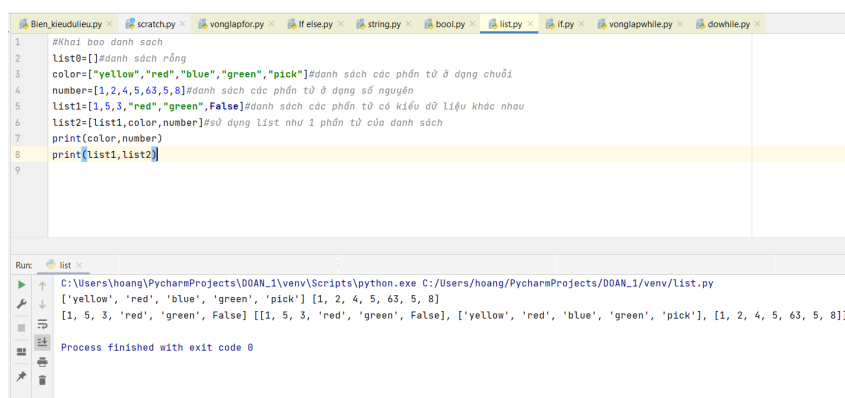
- Kiểu List

-Trong Python list là một dạng dữ liệu cho phép lưu trữ nhiều kiểu dữ liệu

khác nhau và truy xuất các phần tử bên trong nó thông qua vị trí của phần tử đó trong list.

Trong List, các phần tử không nhất thiết phải có kiểu dữ liệu đồng nhất, điều này khiến nó trở thành 1 công cụ mạnh mẽ nhất trong Python. Một list có thể bao gồm các loại Datatypes như Integers, Strings, list,.... cũng như Objects. Lists có thể đổi thay được ngay cả sau khi được tạo.

-Khai báo danh sách: List trong Python có thể được tạo bằng cách chỉ cần đặt chuỗi bên trong dấu ngoặc vuông [].



```
1 #Khai bao danh sach
2 list0=[]#danh sach rỗng
3 color=["yellow","red","blue","green","pick"]#danh sach các phần tử ở dạng chuỗi
4 number=[1,2,4,5,63,5,8]#danh sach các phần tử ở dạng số nguyên
5 list1=[1,5,3,"red","green",False]#danh sach các phần tử có kiểu dữ liệu khác nhau
6 list2=[list1,color,number]#sử dụng List như 1 phần tử của danh sách
7 print(color,number)
8 print(list1,list2)
9
```

Run: list

C:\Users\hoang\PycharmProjects\DOAN_1\venv\Scripts\python.exe C:/Users/hoang/PycharmProjects/DOAN_1/venv/List.py

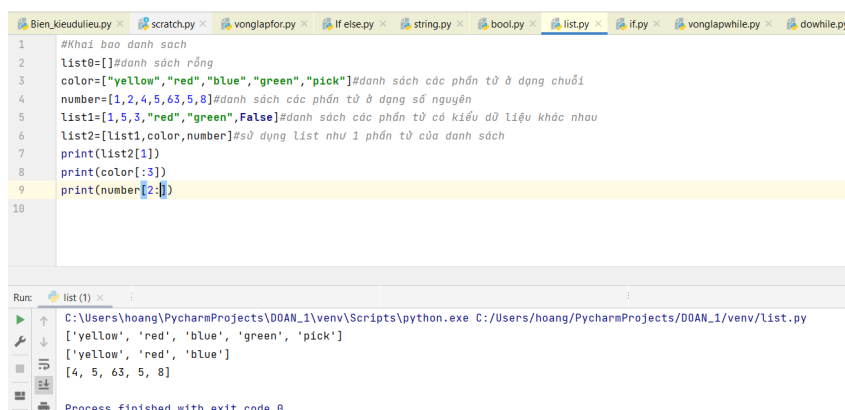
['yellow', 'red', 'blue', 'green', 'pick'] [1, 2, 4, 5, 63, 5, 8]

[1, 5, 3, 'red', 'green', False] [['yellow', 'red', 'blue', 'green', 'pick'], [1, 2, 4, 5, 63, 5, 8]]

Process finished with exit code 0

Hình 2.8: Ví dụ Khai báo List

-List trong Python được sắp xếp theo thứ tự và có số lượng xác định. Các phần tử trong list được lập chỉ mục theo một trình tự xác định và việc lập chỉ mục của danh sách được thực hiện với 0 là chỉ số đầu tiên. Mỗi phần tử đều có vị trí xác định trong list, điều này cho phép sao chép các phần tử trong danh sách, với mỗi phần tử có vị trí và độ tin cậy riêng biệt.



```
1 #Khai bao danh sach
2 list0=[]#danh sach rỗng
3 color=["yellow","red","blue","green","pick"]#danh sach các phần tử ở dạng chuỗi
4 number=[1,2,4,5,63,5,8]#danh sach các phần tử ở dạng số nguyên
5 list1=[1,5,3,"red","green",False]#danh sach các phần tử có kiểu dữ liệu khác nhau
6 list2=[list1,color,number]#sử dụng List như 1 phần tử của danh sách
7 print(list2[1])
8 print(color[:3])
9 print(number[2:4])
10
```

Run: list (1)

C:\Users\hoang\PycharmProjects\DOAN_1\venv\Scripts\python.exe C:/Users/hoang/PycharmProjects/DOAN_1/venv/list.py

['yellow', 'red', 'blue', 'green', 'pick']

['yellow', 'red', 'blue']

[4, 5, 63, 5, 8]

Process finished with exit code 0

Hình 2.9: Ví dụ Kiểu dữ liệu List

- Kiểu Dictionary (từ điển) dictionary là một tập hợp các cặp key-value không có thứ tự, có thể thay đổi và lập chỉ mục (truy cập phần tử theo chỉ mục). Dictionary được khởi tạo với các dấu ngoặc nhọn và chúng có các khóa và giá trị (key-value). Mỗi cặp key-value được xem như là một item. Key mà đã truyền cho item đó phải là duy nhất, trong khi đó value có thể là bất kỳ kiểu giá trị nào. Key phải là một kiểu dữ liệu không thay đổi (immutable) như chuỗi, số hoặc tuple. Key và value được phân biệt riêng rẽ bởi một dấu hai chấm (:). Các item phân biệt nhau bởi một dấu phẩy (.). Các item khác nhau được bao quanh bên trong một cặp dấu ngoặc móc đơn tạo nên một Dictionary trong Python

The screenshot shows a Python IDE with a file named 'Dict.py'. The code in the editor is as follows:

```
1 dict={'1':"yellow",2:"blue",3:"green",4:"red"}
2 print(dict)
```

The output window at the bottom shows the result of running the code:

```
Dict
C:\Users\hoang\PycharmProjects\DOAN_1\venv\Scripts\python.exe C:/Users/hoang/PycharmProjects/DOAN_1/venv/Dict.py
{'1': 'yellow', 2: 'blue', 3: 'green', 4: 'red'}
```

The process finished with exit code 0.

Hình 2.10: Ví dụ kiểu dữ liệu Dictionary

- Kiểu tuple: là một tập hợp các đối tượng Python giống như một danh sách. Một tuple là một bộ sưu tập được sắp xếp thứ tự và không thể thay đổi. Các tuple được viết cùng với ngoặc tròn.

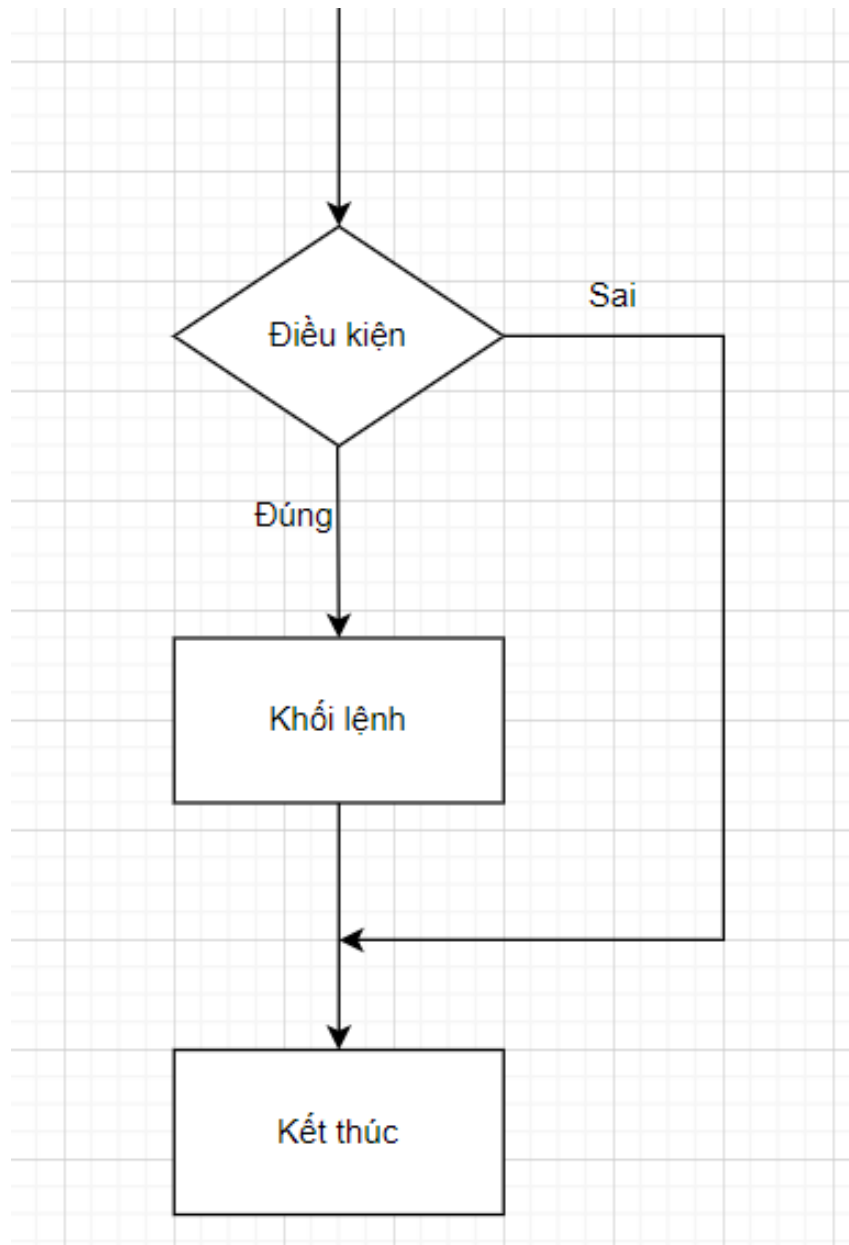
2.1.2 Cấu trúc rẽ nhánh

- Rẽ nhánh dạng thiếu

Cú pháp

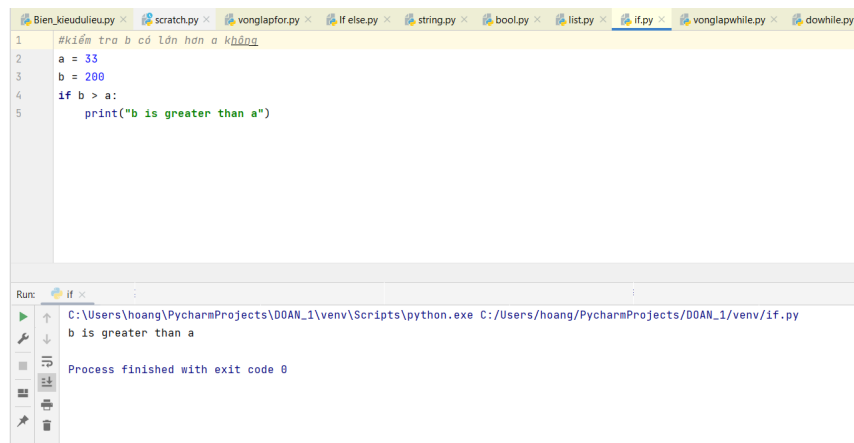
```
1     if (dieu_kien) :
2         cau_lenh
3
```

Sơ đồ khối:



Hình 2.11: Sơ đồ khối rẽ nhánh dạng thiếu

Ví dụ:

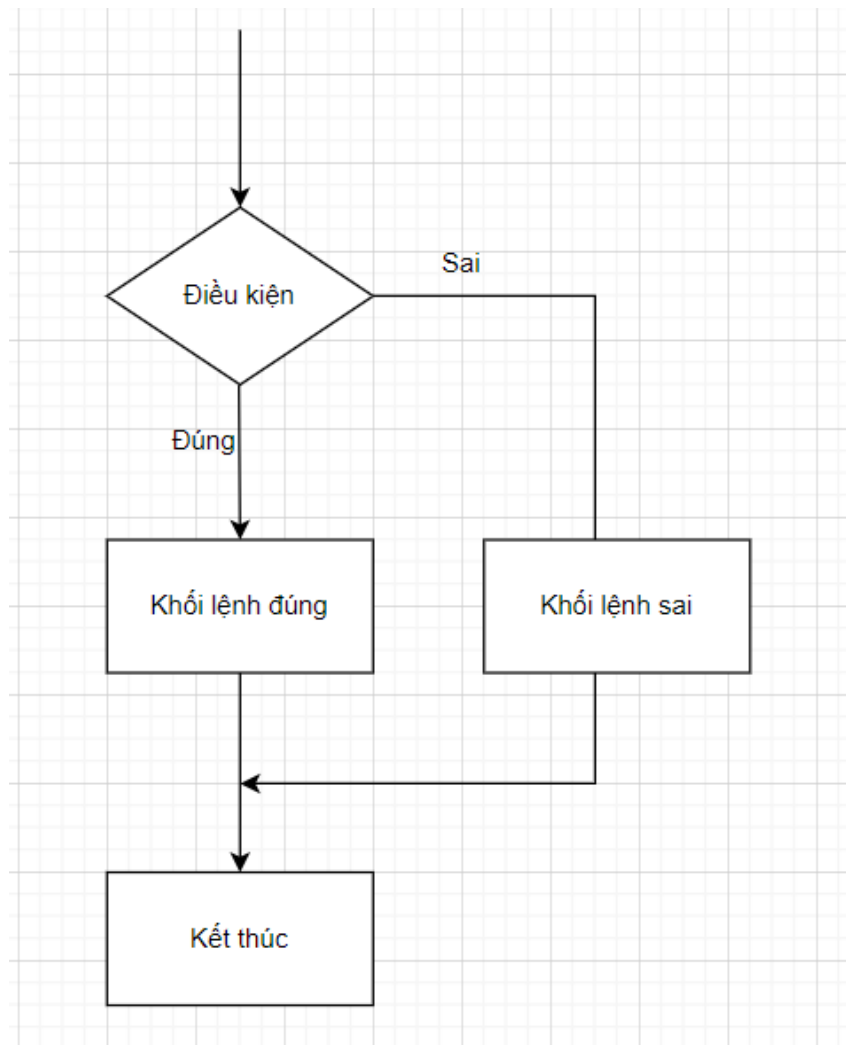


Hình 2.12

- Rẽ nhánh dạng đầy đủ -Cú pháp

```
1         if(dieu_kien):
2             cau_lenh
3         else:
4             cau_lenh
5
```

Sơ đồ khối:



Hình 2.13: Sơ đồ khối rẽ nhánh dạng đầy đủ

Ví dụ:

```

1 #Kiểm tra b có lớn hơn a không
2 a = 200
3 if a%2==0:
4     print(" a chia het cho 2")
5 else:
6     print("a không chia het cho 2")
  
```

Run: if

```

C:\Users\hoang\PycharmProjects\DOAN_1\venv\Scripts\python.exe C:/Users/hoang/PycharmProjects/DOAN_1/venv/if.py
a chia het cho 2
Process finished with exit code 0
  
```

Hình 2.14

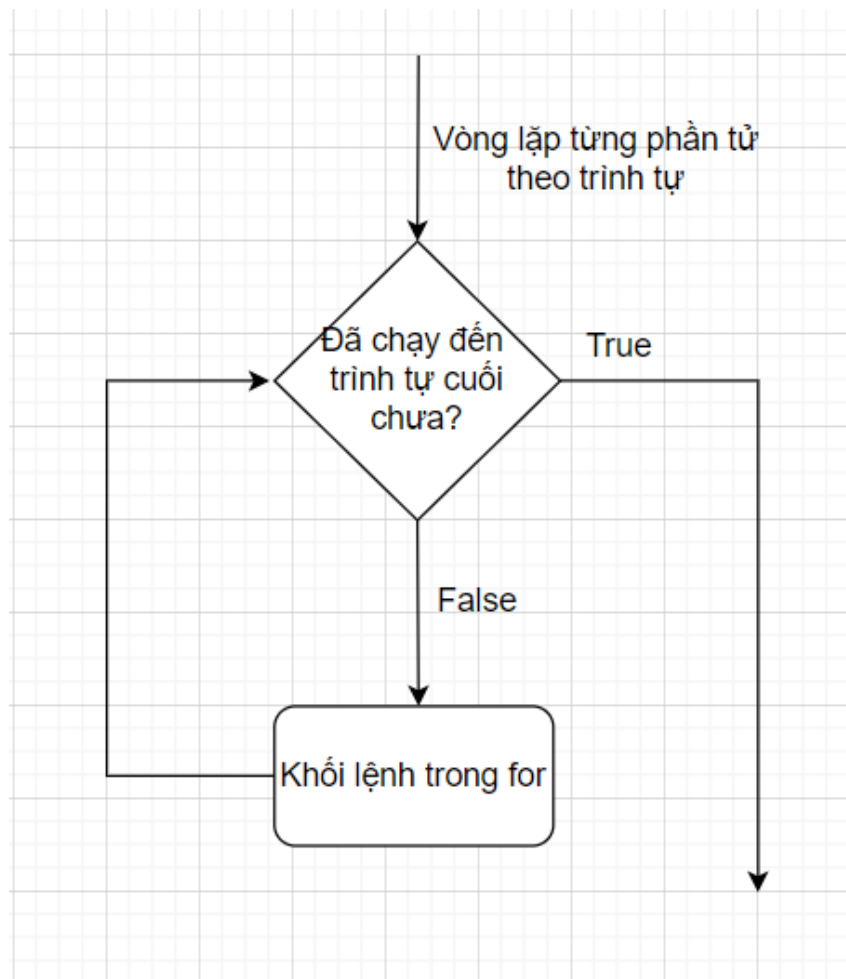
2.1.3 Cấu trúc vòng lặp

- Vòng lặp for

-Cú pháp:

```
1     for bien_lap in chuoi_lap:
2         cau_lenh
3
```

Trong cú pháp trên, `chuoi_lap` là danh sách cần lặp, `bien_lap` là biến nhận giá trị của từng phần tử trong `chuoi_lap` trên mỗi lần lặp. Vòng lặp sẽ tiếp tục cho đến khi nó lặp tới phần tử cuối cùng trong danh sách. Sơ đồ khối:



Hình 2.15: Sơ đồ khối rẽ vòng lặp for

Ví dụ:



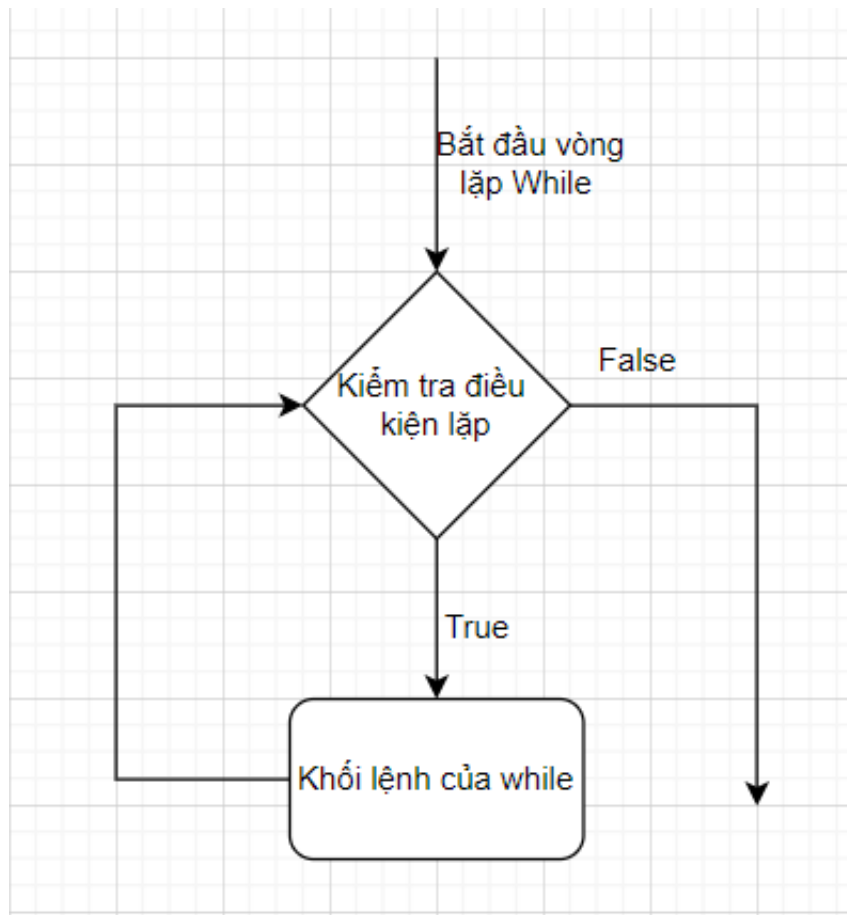
Hình 2.16: Ví dụ vòng lặp for

- Vòng lặp while

-Cú pháp:

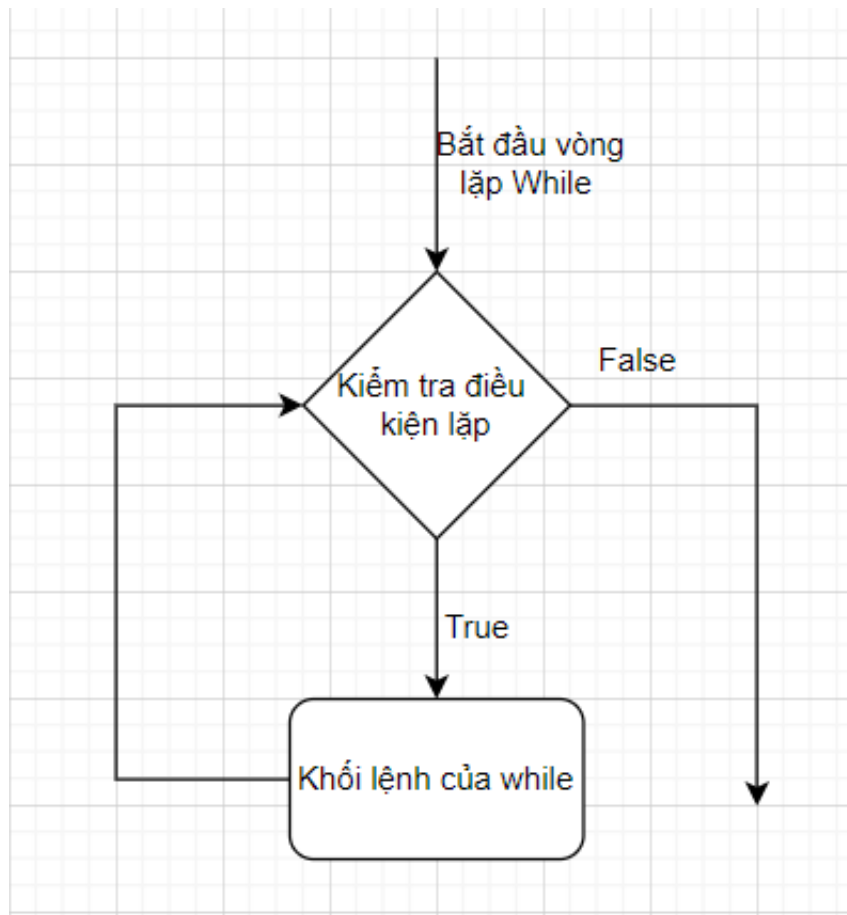
```
1     while dieu_kien:
2         cau_lenh
3
```

Trong vòng lặp while, dieu_kien sẽ được kiểm tra đầu tiên, nếu nó là True, thì khối lệnh của vòng lặp sẽ được thực thi. Sau một lần lặp, dieu_kien sẽ được kiểm tra lại và quá trình lặp này sẽ chỉ dừng cho đến khi điều kiện là False. Sơ đồ khối :



Hình 2.17: Sơ đồ khối vòng lặp while

Ví dụ:



Hình 2.18: Ví dụ vòng lặp whilewhile

2.1.4 Hàm(thủ tục)


Hàm trong Python được sử dụng để tận dụng mã nguồn ở nhiều nơi trong cùng một chương trình. Ý tưởng là đặt một số tác vụ thường được thực hiện hoặc lặp đi lặp lại với nhau và tạo một hàm để thay vì viết cùng một mã lặp đi lặp lại cho các đầu vào khác nhau, chúng ta có thể thực hiện các lệnh gọi hàm để sử dụng lại mã chứa trong nó nhiều lần. Đôi khi hàm còn được gọi là phương thức hoặc thủ tục.

-Các loại hàm trong Python:

- Hàm thư viện tích hợp: Đây là các hàm Tiêu chuẩn trong Python có sẵn để sử dụng.
- Hàm do người dùng xác định: Chúng ta có thể tạo các hàm của riêng mình dựa trên yêu cầu của chúng ta.

Chúng ta có thể tạo một hàm do người dùng xác định trong Python, sử dụng

từ khóa **def**. Chúng ta có thể thêm bất kỳ loại chức năng và thuộc tính nào vào nó. Sau khi tạo một hàm bằng Python, chúng ta có thể gọi nó bằng cách sử dụng tên của hàm theo sau là dấu ngoặc đơn chứa các tham số của hàm cụ thể đó.



```
1 #khai báo biến
2 name1="Messi"
3 name2="Ronaldo"
4 #khởi tạo hàm
5 def hello(name):
6     print("Hello "+name)
7
8 hello(name1)
9 hello(name2)
10
11
```

func1

```
C:\Users\hoang\PycharmProjects\DOAN_1\venv\Scripts\python.exe C:/Users/hoang/PycharmProjects/DOAN_1/venv/func1.py
Hello Messi
Hello Ronaldo
Process finished with exit code 0
```

Hình 2.19: Ví dụ về hàm

2.2 Các thư viện hỗ trợ trong khoa học dữ liệu

2.2.1 Numpy

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.

- Khởi tạo mảng Numpy cho phép chúng ta khởi tạo mảng được cấu hình theo định dạng dữ liệu cụ thể như float, interger, boolean, string. Các phần tử của một mảng trong numpy phải đồng nhất về định dạng dữ liệu.

Cú pháp :np.array().


```
#tạo mảng, mảng sẽ có kiểu dữ liệu ndarray
c=np.array([1,3,5])
print(c)
type(c)
```

```
[1 3 5]
```

```
numpy.ndarray
```

Hình 2.20: Ví dụ khởi tạo mảng

- Khởi tạo mảng ngẫu nhiên

Cú pháp: `np.random.normal(loc, scale, size)`

Khởi tạo mảng (ma trận) mà các phần tử của mảng (ma trận) tuân theo phân phối chuẩn (Gaussian distribution) với trung bình chính là **loc** và phương sai là **scale**, **size** là kích thước của mảng (ma trận).

```
#Tạo 1 mảng gồm các giá trị ngẫu nhiên tuân theo phân phối chuẩn với mean và std đã biết
rand=np.random.normal(1,1,(3,3))
rand
array([[ 0.58339447,  1.23421258,  1.66548752],
       [-0.67262816, -0.10541133,  2.03711906],
       [-0.36949054,  2.46145179,  1.70472716]])
```

Hình 2.21: Ví dụ khởi tạo mảng

- Các phép toán với ma trận -Ma trận khả nghịch

```
A = np.array([
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
])
A_inv=np.linalg.pinv(A)
A_inv
array([[ -6.38888889e-01, -1.66666667e-01,  3.05555556e-01],
       [-5.55555556e-02, -9.89600165e-17,  5.55555556e-02],
       [ 5.27777778e-01,  1.66666667e-01, -1.94444444e-01]])
```

Hình 2.22: Ví dụ ma trận khả nghịch

-Định thức ma trận

```
A = np.array([
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
])
np.linalg.det(A)
```

-9.51619735392994e-16

Hình 2.23: Ví dụ định thức ma trận

- Các phép toán trên mảng

-Min,max,mean,sum:numpy cung cấp một loạt các hàm thống kê thông dụng theo chiều dòng hoặc cột như min, max, mean, sum.

Cú pháp:array.max(axis)

Với axis=1 là theo dòng và axis=0 là theo cột.

```
a=np.array([2,4,6,6,5])
print("Sum =",a.sum())
print("Max =",a.max())
print("Min =",a.min())
print("mean =",a.mean())
```

```
Sum = 23
Max = 6
Min = 2
mean = 4.6
```

Hình 2.24: Ví dụ thống kê cơ bản về mảng

-Sort:Sắp xếp các phần tử trong mảng

Cú pháp:np.sort(array)

```
a=np.array([2,4,6,6,5])  
np.sort(a)  
  
array([2, 4, 5, 6, 6])
```

Hình 2.25: Ví dụ sắp xếp các phần tử trong mảng

2.2.2 Pandas

Thư viện pandas trong python là một thư viện mã nguồn mở, hỗ trợ đắc lực trong thao tác dữ liệu. Đây cũng là bộ công cụ phân tích và xử lý dữ liệu mạnh mẽ của ngôn ngữ lập trình python. Thư viện này được sử dụng rộng rãi trong cả nghiên cứu lẫn phát triển các ứng dụng về khoa học dữ liệu. Thư viện này sử dụng một cấu trúc dữ liệu riêng là Dataframe. Pandas cung cấp rất nhiều chức năng xử lý và làm việc trên cấu trúc dữ liệu này. Chính sự linh hoạt và hiệu quả đã khiến cho pandas được sử dụng rộng rãi.

Đọc File

Cú pháp :pandas.read_csv(filepath,sep,header)

filepath:Đường dẫn file

sep:Dấu phân cách

header:chỉ định file đọc vào có header(tiêu đề của các cột) hay không. Mặc định là infer.

```
import pandas as pd
import numpy as np
df=pd.read_csv("chipotle.tsv",sep="\t")
df
```

	order_id	quantity	item_name	choice_description	item_price
0	1	1	Chips and Fresh Tomato Salsa	NaN	\$2.39
1	1	1	Izze	[Clementine]	\$3.39
2	1	1	Nantucket Nectar	[Apple]	\$3.39
3	1	1	Chips and Tomatillo-Green Chili Salsa	NaN	\$2.39
4	2	2	Chicken Bowl	[Tomatillo-Red Chili Salsa (Hot), [Black Beans...	\$16.98
...
4617	1833	1	Steak Burrito	[Fresh Tomato Salsa, [Rice, Black Beans, Sour ...	\$11.75
4618	1833	1	Steak Burrito	[Fresh Tomato Salsa, [Rice, Sour Cream, Cheese...	\$11.75
4619	1834	1	Chicken Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Pinto...	\$11.25
4620	1834	1	Chicken Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Lettu...	\$8.75
4621	1834	1	Chicken Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Pinto...	\$8.75

4622 rows x 5 columns

Hình 2.26: Ví dụ đọc File

Thao tác với DataFrame

- Mô tả các cột trong DataFrame

Cú pháp: DataFrame.info()

```
df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4622 entries, 0 to 4621
Data columns (total 5 columns):
#   column          Non-Null Count  Dtype
---  -
0   order_id        4622 non-null     int64
1   quantity        4622 non-null     int64
2   item_name       4622 non-null     object
3   choice_description 3376 non-null     object
4   item_price      4622 non-null     object
dtypes: int64(2), object(3)
memory usage: 180.7+ KB
```

Hình 2.27: Ví dụ mô tả các cột trong DataFrame

- Lấy bản ghi theo chỉ số

Cú pháp : DataFrame.iloc[]

```
df.iloc[3:11]
```

	order_id	quantity	item_name	choice_description	item_price
3	1	1	Chips and Tomatillo-Green Chili Salsa	NaN	\$2.39
4	2	2	Chicken Bowl	[Tomatillo-Red Chili Salsa (Hot), [Black Beans...	\$16.98
5	3	1	Chicken Bowl	[Fresh Tomato Salsa (Mild), [Rice, Cheese, Sou...	\$10.98
6	3	1	Side of Chips	NaN	\$1.69
7	4	1	Steak Burrito	[Tomatillo Red Chili Salsa, [Fajita Vegetables...	\$11.75
8	4	1	Steak Soft Tacos	[Tomatillo Green Chili Salsa, [Pinto Beans, Ch...	\$9.25
9	5	1	Steak Burrito	[Fresh Tomato Salsa, [Rice, Black Beans, Pinto...	\$9.25
10	5	1	Chips and Guacamole	NaN	\$4.45

Hình 2.28: Ví dụ về lấy bản ghi theo chỉ số

- Thống kê cơ bản về Dataframe

Cú pháp : DataFrame.describe()

```
df.describe()
```

	order_id	quantity	item_name	choice_description	item_price
count	4622.000000	4622.000000	4622	3376	4622
unique	NaN	NaN	50	1043	78
top	NaN	NaN	Chicken Bowl	[Diet Coke]	\$8.75
freq	NaN	NaN	726	134	730
mean	927.254868	1.075725	NaN	NaN	NaN
std	528.890796	0.410186	NaN	NaN	NaN
min	1.000000	1.000000	NaN	NaN	NaN
25%	477.250000	1.000000	NaN	NaN	NaN
50%	926.000000	1.000000	NaN	NaN	NaN
75%	1393.000000	1.000000	NaN	NaN	NaN
max	1834.000000	15.000000	NaN	NaN	NaN

Hình 2.29: Ví dụ về thống kê cơ bản về Dataframe

- Xóa cột, bản ghi trong Dataframe

Cú pháp :Dataframe.drop(label,axis)

label:tên cột hoặc bản ghi cần xóa

axis:0 là xóa bản ghi,1 là xóa cột.Mặc định là 0.

```
dfdrop=df.drop('quantity',axis=1)
dfdrop
```

	order_id	item_name	choice_description	item_price
0	1	Chips and Fresh Tomato Salsa	NaN	\$2.39
1	1	Izze	[Clementine]	\$3.39
2	1	Nantucket Nectar	[Apple]	\$3.39
3	1	Chips and Tomatillo-Green Chili Salsa	NaN	\$2.39
4	2	Chicken Bowl	[Tomatillo-Red Chili Salsa (Hot), [Black Beans...	\$16.98
...
4617	1833	Steak Burrito	[Fresh Tomato Salsa, [Rice, Black Beans, Sour ...	\$11.75
4618	1833	Steak Burrito	[Fresh Tomato Salsa, [Rice, Sour Cream, Cheese...	\$11.75
4619	1834	Chicken Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Pinto...	\$11.25
4620	1834	Chicken Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Lettu...	\$8.75
4621	1834	Chicken Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Pinto...	\$8.75

4622 rows x 4 columns

Hình 2.30: Ví dụ xóa cột trong Dataframe

```
df.drop([0, 1]) # Xóa bản ghi ở chỉ số 1 và 2
```

	order_id	quantity	item_name	choice_description	item_price
2	1	1	Nantucket Nectar	[Apple]	\$3.39
3	1	1	Chips and Tomatillo-Green Chili Salsa	NaN	\$2.39
4	2	2	Chicken Bowl	[Tomatillo-Red Chili Salsa (Hot), [Black Beans...	\$16.98
5	3	1	Chicken Bowl	[Fresh Tomato Salsa (Mild), [Rice, Cheese, Sou...	\$10.98
6	3	1	Side of Chips	NaN	\$1.69
...
4617	1833	1	Steak Burrito	[Fresh Tomato Salsa, [Rice, Black Beans, Sour ...	\$11.75
4618	1833	1	Steak Burrito	[Fresh Tomato Salsa, [Rice, Sour Cream, Cheese...	\$11.75
4619	1834	1	Chicken Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Pinto...	\$11.25
4620	1834	1	Chicken Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Lettu...	\$8.75
4621	1834	1	Chicken Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Pinto...	\$8.75

4620 rows x 5 columns

Hình 2.31: Ví dụ xóa bản ghi trong Dataframe

- Loại bỏ các bản ghi trùng lặp

Cú pháp :Dataframe.duplicated()

```
df.drop_duplicates(inplace=True)
df
```

	order_id	quantity	item_name	choice_description	item_price
0	1	1	Chips and Fresh Tomato Salsa	NaN	\$2.39
1	1	1	lzze	[Clementine]	\$3.39
2	1	1	Nantucket Nectar	[Apple]	\$3.39
3	1	1	Chips and Tomatillo-Green Chili Salsa	NaN	\$2.39
4	2	2	Chicken Bowl	[Tomatillo-Red Chili Salsa (Hot), [Black Beans ...	\$16.98
...
4617	1833	1	Steak Burrito	[Fresh Tomato Salsa, [Rice, Black Beans, Sour ...	\$11.75
4618	1833	1	Steak Burrito	[Fresh Tomato Salsa, [Rice, Sour Cream, Cheese...	\$11.75
4619	1834	1	Chicken Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Pinto...	\$11.25
4620	1834	1	Chicken Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Lettu...	\$8.75
4621	1834	1	Chicken Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Pinto...	\$8.75

4563 rows x 5 columns

Hình 2.32: Loại bỏ các bản ghi trùng lặp

- Kiểm tra dữ liệu bị thiếu

Cú pháp :`Dataframe.isnull().sum()`

```
df.isnull().sum()
order_id      0
quantity      0
item_name      0
choice_description  1228
item_price     0
dtype: int64
```

Hình 2.33: Ví dụ kiểm tra dữ liệu bị thiếu

2.2.3 Matplotlib

Matplotlib (cụ thể trong đề án này ta quan tâm tới gói `Matplotlib.pyplot` hay được viết gọn là `plt`) là một gói Python được sử dụng cho đồ họa 2D.

Matplotlib có 2 objects chính:

Figure: chứa nhiều axes
 Axes: nơi vẽ đồ thị, thực tế chính là cái đồ thị
 Nói 1 cách dễ hiểu, Figure giống khung ảnh và 1 khung ảnh có thể không chứa hoặc chứa 1 đến nhiều bức ảnh, và Axes tương ứng với những bức ảnh trong cái khung ảnh.

Các thành phần khác:

Title: tên của đồ thị

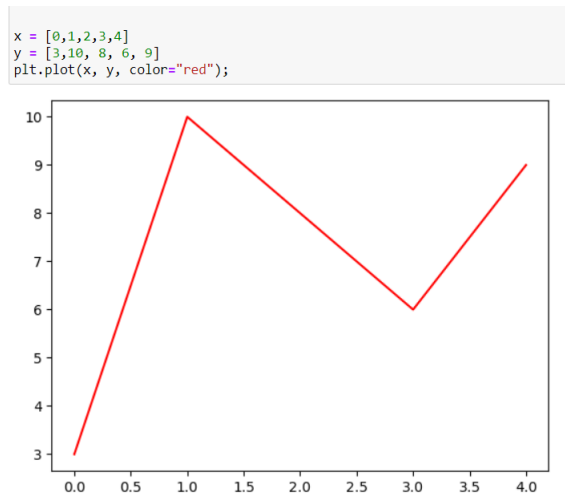
X axis label, Y axis label: 2 trục của đồ thị

Legend: Chú thích

- Line Graph

Biểu đồ đường được sử dụng để biểu diễn mối quan hệ giữa hai dữ liệu X và Y trên một trục khác nhau

Cú pháp: `pyplot.plot(x, y)`



Hình 2.34: Ví dụ biểu đồ Line Graph

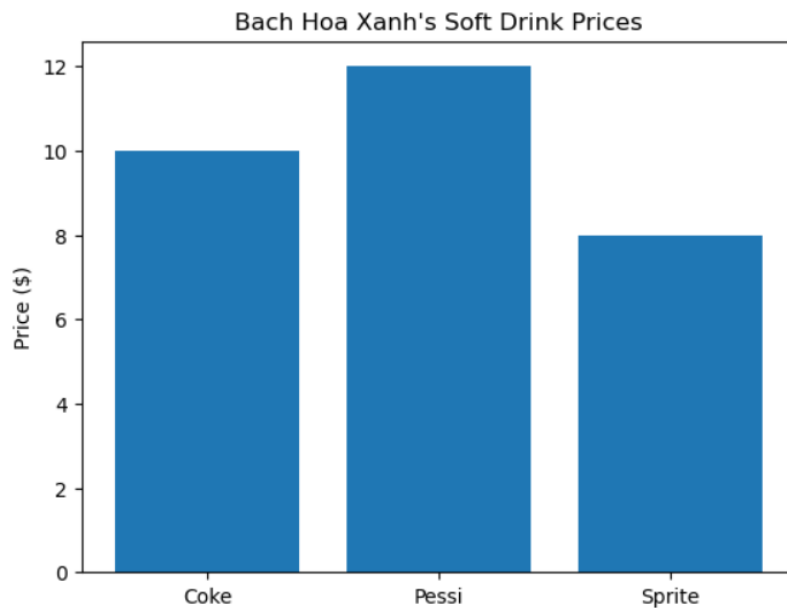
- Bar chart

Biểu đồ Bar chart là biểu đồ đại diện cho danh mục dữ liệu với các thanh hình chữ nhật có độ dài và chiều cao tỷ lệ thuận với các giá trị mà chúng đại diện. Các ô thanh có thể được vẽ theo chiều ngang hoặc chiều dọc. Biểu đồ thanh mô tả sự so sánh giữa các danh mục phân loại.

Cú pháp: `pyplot.bar(keys, values)`

```
#Bar
soft_drink_prices = {"Coke": 10,
                    "Pessi": 12,
                    "Sprite": 8}

plt.bar(soft_drink_prices.keys(), soft_drink_prices.values());
plt.title("Bach Hoa Xanh's Soft Drink Prices"),plt. ylabel("Price ($)");
```

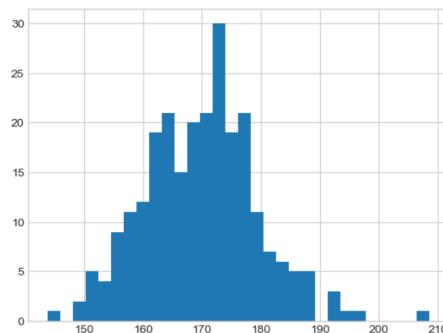


Hình 2.35: Ví dụ biểu đồ Bar chart

- Histogram là biểu đồ thể hiện sự phân bố tần suất của một tập dữ liệu. Histogram là dạng biểu đồ chỉ vẽ một biến dựa trên số lần xuất hiện trong danh mục biến.

Cú pháp: `pyplot.hist(data,bins)`

```
#Histogram
#Tạo 1 mảng gồm các giá trị ngẫu nhiên tuân theo phân phối chuẩn với mean và std đã biết
student_height = np.random.normal(170, 10, 250)
plt.hist(student_height, bins=30);
```

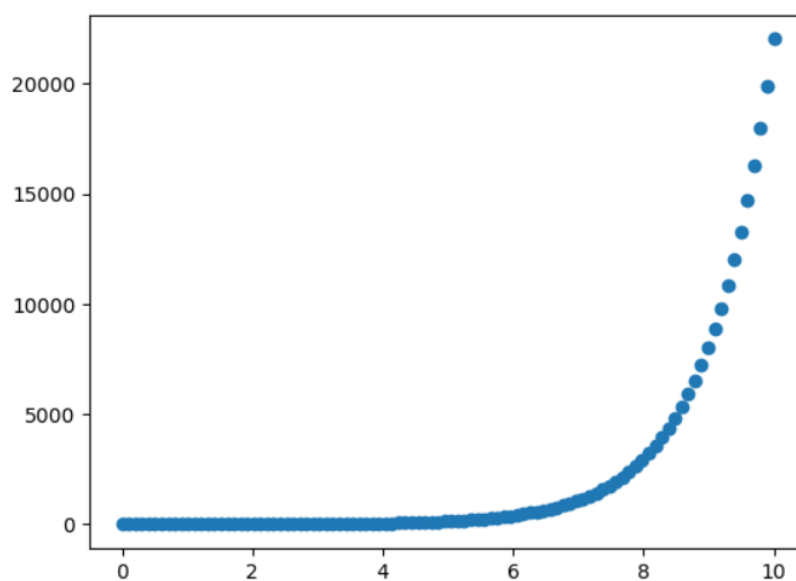


Hình 2.36: Biểu đồ histogram

- Scatter plot Biểu đồ phân tán được sử dụng để quan sát mối quan hệ giữa các biến và sử dụng dấu chấm để thể hiện mối quan hệ giữa chúng.

Cú pháp: `pyplot.(x,y)`

```
#scatter  
x = np.linspace(0,10, 100)  
plt.scatter(x, np.exp(x)); # $y=e^x$ 
```



Hình 2.37: Ví dụ biểu đồ Scater plot

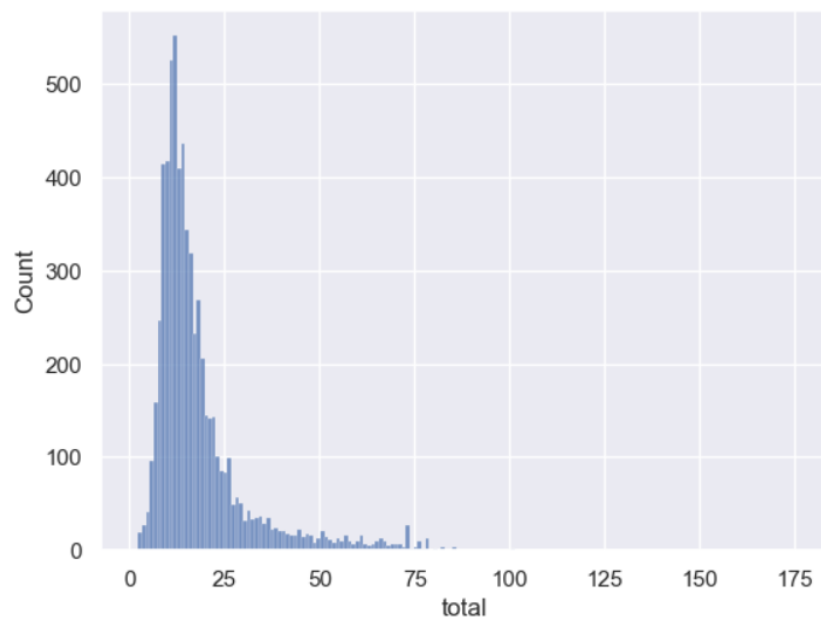
2.2.4 Seaborn

- Distribution plot(Đồ thị phân phối)

Distribution plot đếm theo bins. Số bins càng lớn, chia khoảng càng nhỏ thì các chính xác!

Cú pháp: `sns.distplot(data,bins)`

```
sns.histplot(data=df["total"]);  
# Phân phối của column "total"
```

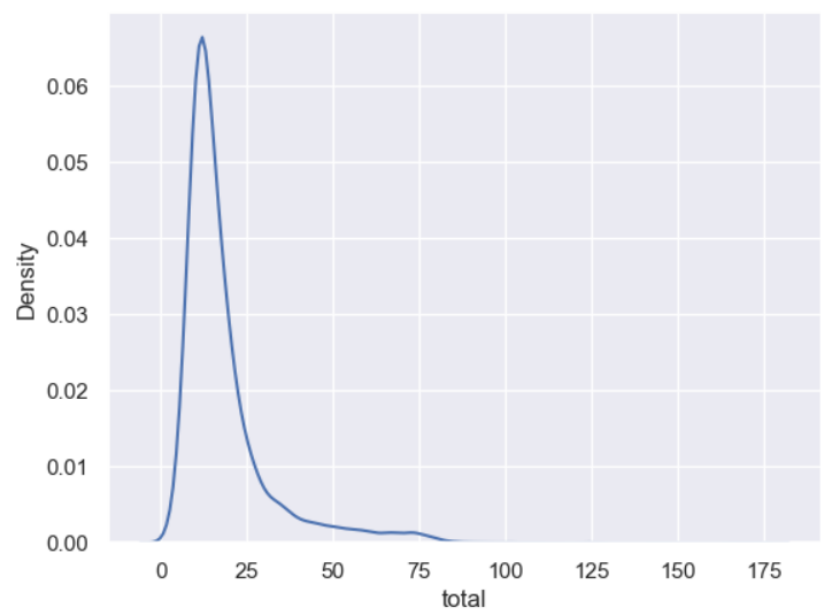


Hình 2.38: Ví dụ biểu đồ Distribution

- KDE plot(Biểu đồ ước tính mật độ hạt nhân)

Cú pháp:sns.kdeplot(data_column)

```
sns.kdeplot(data=df['total']);
```

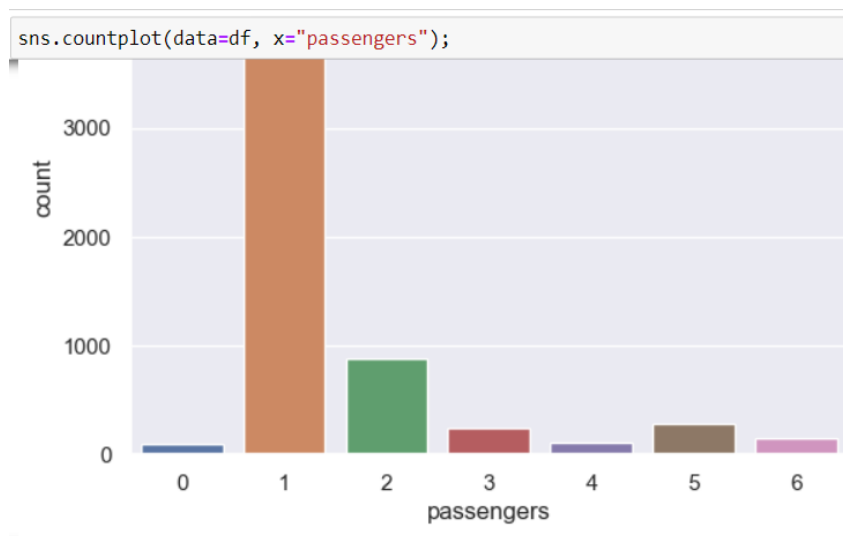


Hình 2.39: Ví dụ biểu đồ KDE

- Count plot

Biểu đồ Countplot sẽ trả về số lượng của từng category dưới dạng cột.

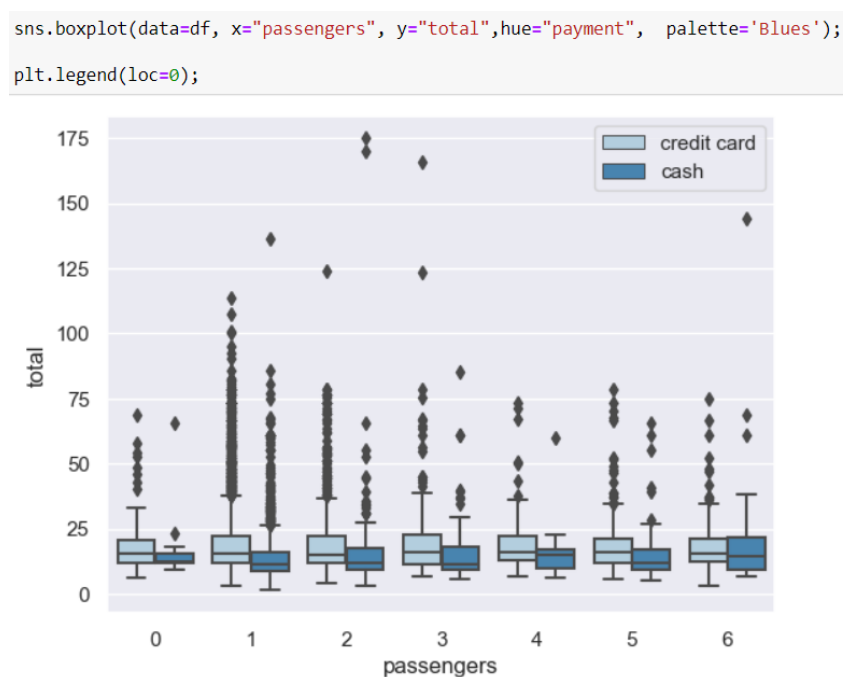
Cú pháp: `sns.countplot(data_column)`



Hình 2.40: Ví dụ biểu đồ Count plot

- Box plot

Đồ thị Box plot là đồ thị biểu đồ diễn tả 5 vị trí phân bố của dữ liệu, đó là: giá trị nhỏ nhất (min), tứ phân vị thứ nhất (Q1), trung vị (median), tứ phân vị thứ 3 (Q3) và giá trị lớn nhất (max).



Hình 2.41: Ví dụ biểu đồ Box plot

2.2.5 Scikit-learn

Trong phần này chúng ta sẽ chỉ tìm hiểu về các hàm để sử dụng cho việc huấn luyện mô hình hồi quy tuyến tính.

- `train_test_split()` Là hàm chia tập dữ liệu thành các tập con :
 - Tập huấn luyện
 - Tập kiểm tra

Cú pháp: `x_train, x_test, y_train, y_test = train_test_split(x, y, test_size, random_state)`

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 0)
print(x_train.shape)
print(x_test.shape)

(3697, 4)
(925, 4)
```

Hình 2.42: Ví dụ `train_test_split()`

- `train_test_split()`
 - Các hệ số đánh giá mô hình Với `y_pred` là giá trị dự đoán với dữ liệu của tập kiểm tra và `y_test` là giá trị thực tế trong tập kiểm tra. Ta có các hệ số đánh giá mô hình như sau:
 - MSE(Mean Square Error):Trung bình bình phương sai số
- Cú pháp:`mean_absolute_error(ypred,y_test)`
- `R2_Score`:Hệ số xác định
- Cú pháp:`r2_score(ypred,y_test)`
- MAE (Mean Absolute Error):Trung bình của giá trị tuyệt đối sai số
- Cú pháp :`mean_squared_error(ypred,y_test)`

```

from sklearn import metrics
print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))

```

```

MAE: 1001405.4100995407
MSE: 1814945930172.843
RMSE: 1347199.2911862903

```

Hình 2.43: Ví dụ các hệ số đánh giá mô hình

- Các hàm trong mô hình LinearRegression()

-LinearRegression.fit(x_train,y_train):Tính toán các hệ số

Với x_train,y_train là giá trị thực của x và y trong tập huấn luyện

-LinearRegression.coef_:Độ chệch(bias)

-LinearRegression.intercept_:Các hệ số hồi quy

```

linear = LinearRegression()
linear.fit(x_train, y_train)
w0= linear.intercept_
w1= linear.coef_
print("intercept:",w0)
print("coef:",w1)

intercept: -402734.409333718
coef: [2.70711520e+02 1.38866393e+05 9.01166423e+05 4.17256461e+05
 4.78319543e+05 2.563908937e+05 4.28826063e+05 9.12357723e+05
 8.77395942e+05 3.14777790e+05 6.15405166e+05 2.34649553e+05]

```

Hình 2.44: Ví dụ các hệ số hồi quy

Chương 3

Thực nghiệm với dữ liệu

3.1 Dữ liệu khảo sát về một số tòa nhà trong 1 thành phố

Bộ dữ liệu chứa các một số thông số yếu tố của một số tòa nhà trong 1 thành phố
Một số thông tin cơ bản của bộ dữ liệu:

- Nguồn dữ liệu: Bộ dữ liệu được tác giả thu thập trên Kaggle, link dữ liệu:
<https://www.kaggle.com/datasets/yasserh/housing-prices-dataset>
- Tên file: Housing.csv
- Số mẫu: 545 mẫu
- Tổng số trường: 13 trường
- Các trường chi tiết:
 - 1 price:Giá nhà
 - 2 area:Diện tích
 - 3 bedrooms:Số lượng phòng ngủ
 - 4 bathrooms:Số lượng phòng tắm
 - 5 stories:Số lượng tầng hầm
 - 6 mainroad:Gần đường chính

- 7 guestroom:Phòng khách
- 8 basement:Tầng hầm
- 9 hotwaterheating:Bình nóng lạnh
- 10 airconditioning:Máy điều hòa
- 11 parking:Chỗ đỗ xe
- 12 prefarea:Vị trí tốt hơn
- 13 furnishingstatus:tình trạng nội thất

3.2 Bài toán hồi quy tuyến tính

3.2.1 Đọc hiểu dữ liệu

- Xem thông tin cơ bản về các trường của dữ liệu

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 545 entries, 0 to 544
Data columns (total 13 columns):
#   Column                Non-Null Count  Dtype
---  -
0   price                 545 non-null   int64
1   area                  545 non-null   int64
2   bedrooms              545 non-null   int64
3   bathrooms             545 non-null   int64
4   stories               545 non-null   int64
5   mainroad              545 non-null   object
6   guestroom             545 non-null   object
7   basement              545 non-null   object
8   hotwaterheating       545 non-null   object
9   airconditioning       545 non-null   object
10  parking               545 non-null   int64
11  prefarea              545 non-null   object
12  furnishingstatus      545 non-null   object
dtypes: int64(6), object(7)
memory usage: 55.5+ KB
```

Hình 3.1: thông tin cơ bản về các trường

Ta có thể thấy dữ liệu gồm 6 trường dạng số và 7 trường dạng phân loại

- Kiểm tra độ tương quan giữa các trường dạng số

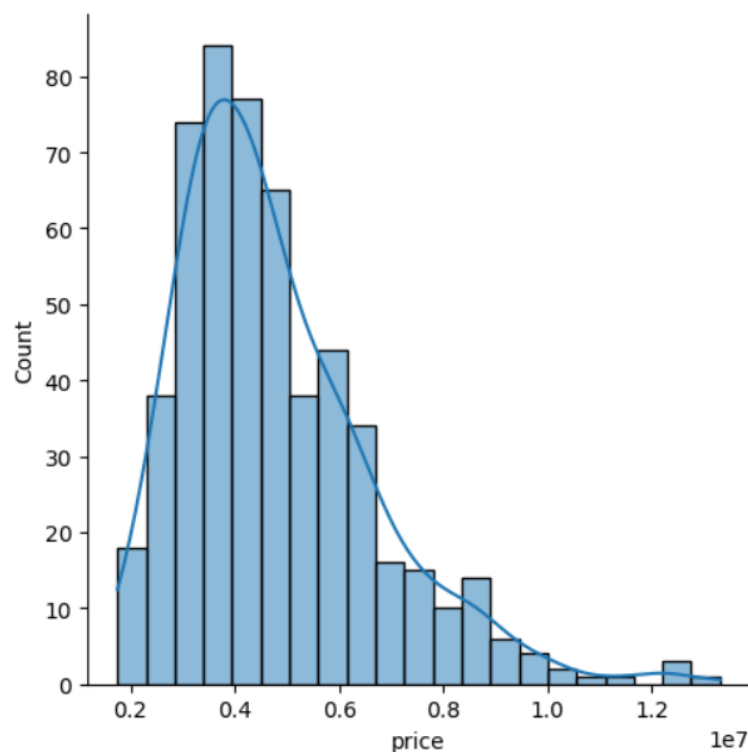
	price	area	bedrooms	bathrooms	stories	parking
price	1.000000	0.535997	0.366494	0.517545	0.420712	0.384394
area	0.535997	1.000000	0.151858	0.193820	0.083996	0.352980
bedrooms	0.366494	0.151858	1.000000	0.373930	0.408564	0.139270
bathrooms	0.517545	0.193820	0.373930	1.000000	0.326165	0.177496
stories	0.420712	0.083996	0.408564	0.326165	1.000000	0.045547
parking	0.384394	0.352980	0.139270	0.177496	0.045547	1.000000

Hình 3.2: độ tương quan giữa các trường dạng số

Từ ma trận tương quan ta thấy các không có 2 trường nào có tương quan cao với nhau, nên ở đây không có hiện tượng đa cộng tuyến .

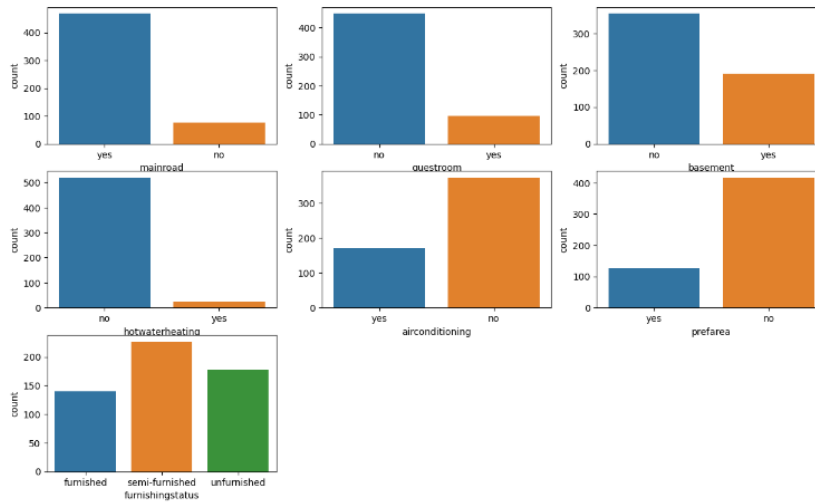
3.2.2 Trực quan hóa dữ liệu

- Trực quan hóa dữ liệu trường "price":



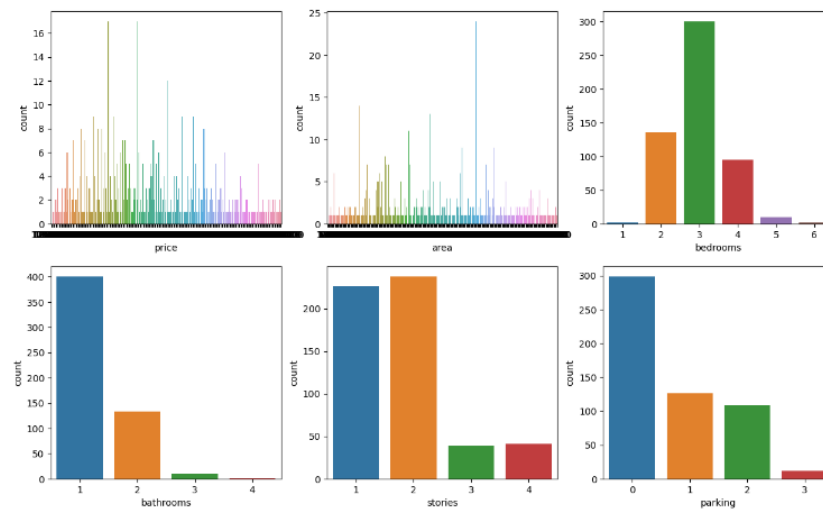
Hình 3.3: Biểu đồ histogram trường "price"

- Trực quan hóa dữ liệu các trường dạng phân loại



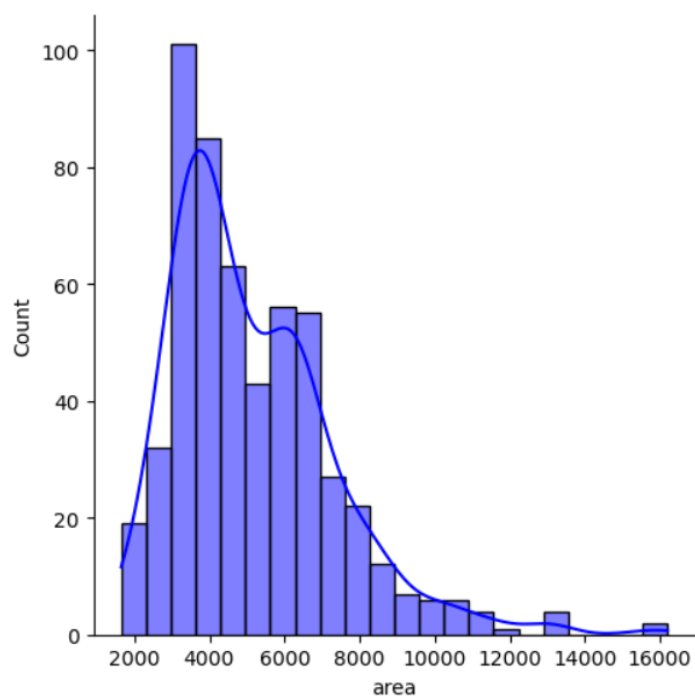
Hình 3.4: Biểu đồ Countplot các trường dạng phân loại

- Trực quan hóa dữ liệu các trường dạng số



Hình 3.5: Biểu đồ Countplot các trường dạng số

- Trực quan hóa dữ liệu trường "area"



Hình 3.6: Biểu đồ histogram trường "area"

3.2.3 Tiền xử lý dữ liệu

- Kiểm tra các dữ liệu bị thiếu

```
df.isnull().sum()
```

```
price          0
area           0
bedrooms       0
bathrooms      0
stories        0
mainroad       0
guestroom      0
basement       0
hotwaterheating 0
airconditioning 0
parking        0
prefarea       0
furnishingstatus 0
dtype: int64
```

Hình 3.7: Kiểm tra các dữ liệu bị thiếu

- Loại bỏ các bản ghi trùng lặp nếu có

```
df.drop_duplicates(inplace=True)
df.shape

(533, 13)
```

Hình 3.8: Loại bỏ các bản ghi trùng lặp nếu có

- Mã hóa các trường dạng phân loại sang dữ liệu dạng số

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	prefarea	furnishingstatus
0	13300000	7420	4	2	3	1	0	0	0	1	2	1	2
1	12250000	8960	4	4	4	1	0	0	0	1	3	0	2
2	12250000	9960	3	2	2	1	0	1	0	0	2	1	1
3	12215000	7500	4	2	2	1	0	1	0	1	3	1	2
4	11410000	7420	4	1	2	1	1	1	0	1	2	0	2
...
540	1820000	3000	2	1	1	1	0	1	0	0	2	0	0
541	1767150	2400	3	1	1	0	0	0	0	0	0	0	1
542	1750000	3620	2	1	1	1	0	0	0	0	0	0	0
543	1750000	2910	3	1	1	0	0	0	0	0	0	0	2
544	1750000	3850	3	1	2	1	0	0	0	0	0	0	0

545 rows x 13 columns

Hình 3.9: Mã hóa các trường dạng phân loại sang dữ liệu dạng số

- Loại bỏ các giá trị ngoại lai trường "area"

```
print("Dữ liệu ban đầu có kích thước:",df.shape)
Q1 = df['area'].quantile(0.25)
Q3 = df['area'].quantile(0.75)
IQR = Q3 - Q1
df = df[df['area'] <= (Q3+(1.5*IQR))]
df = df[df['area'] >= (Q1-(1.5*IQR))]
df = df.reset_index(drop=True)
print("Dữ liệu sau khi loại bỏ dữ liệu ngoại lai có kích thước:",df.shape)
```

Dữ liệu ban đầu có kích thước: (545, 13)

Dữ liệu sau khi loại bỏ dữ liệu ngoại lai có kích thước: (533, 13)

Hình 3.10: Loại bỏ các giá trị ngoại lai trường "area"

Nhận thấy rằng trường "area" có 12 giá trị ngoại lai

3.2.4 Chia tập dữ liệu

Chia tập dữ liệu thành tập huấn luyện và tập kiểm thử

```
#Define target value and independent variables
x = df.drop(columns = 'price')
y = df['price']
#Split data
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 0)
print(x_train.shape)
print(x_test.shape)

(426, 12)
(107, 12)
```

Hình 3.11: Chia tập dữ liệu thành tập huấn luyện và tập kiểm tra

3.2.5 Huấn luyện mô hình hồi quy tuyến tính

Sau khi chạy mô hình hồi quy tuyến tính ta tìm được các hệ số hồi quy(coef) và bias(intercept)

```
linear = LinearRegression()
linear.fit(x_train, y_train)
w0= linear.intercept_
w1= linear.coef_
print("intercept:",w0)
print("coef:",w1)

intercept: -402734.409333718
coef: [2.70711520e+02 1.38866393e+05 9.01166423e+05 4.17256461e+05
4.78319543e+05 2.56398937e+05 4.28826863e+05 9.12357723e+05
8.77395942e+05 3.14777790e+05 6.15405166e+05 2.34649553e+05]
```

Hình 3.12: Các hệ số hồi quy

3.3 Đánh giá mô hình

Sử dụng dữ liệu của tập kiểm tra ta tính được cái hệ số đánh giá mô hình như sau:

- Mean square error: 1180430381270.3052
- R2 score: 0.49236204710671794
- mean absolute error is: 846364.8952933942

-Trực quan hóa phần dư:

$$\text{residual} = Y_{\text{train}} - Y_{\text{train_predict}}$$

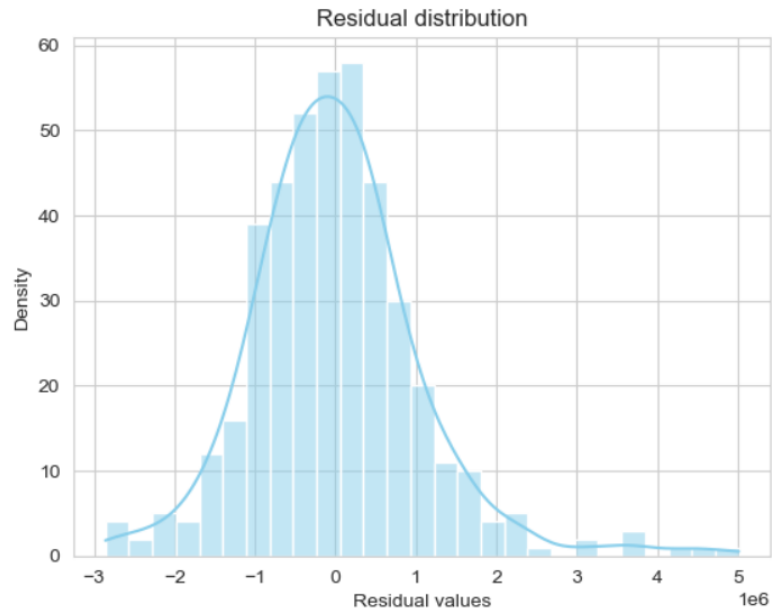
Trong đó:

residual :Phần dư

Y_{train} :giá trị thực đã có trong tập huấn luyện

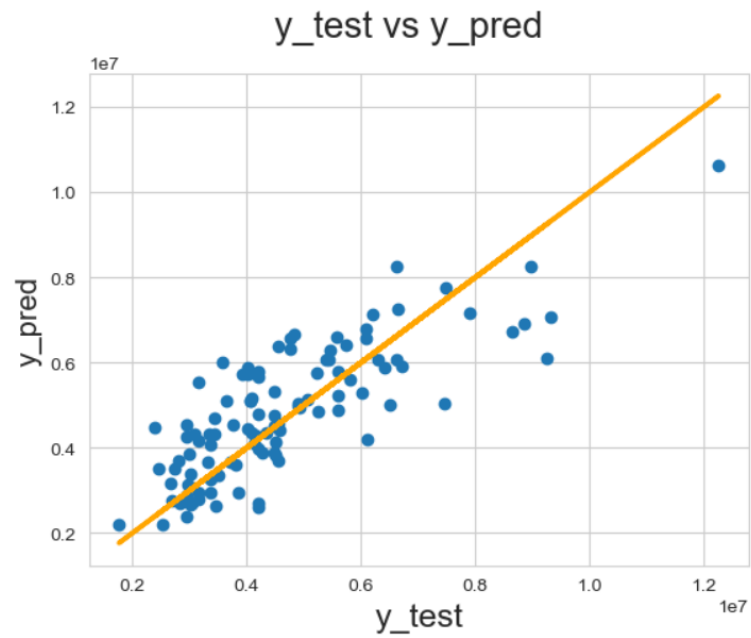
$Y_{\text{train_predict}}$:giá trị dự đoán

Sai số trung bình: $9.182053552546971 \times 10^{-11}$



Hình 3.13: Biểu đồ Distribution của phần dư

Ta có thể thấy rằng sai số trung bình gần bằng 0. Tiếp theo ta vẽ biểu đồ thể hiện quan hệ giữa y và \hat{y} trong tập kiểm tra



Hình 3.14: Biểu đồ thể hiện quan hệ y và \hat{y}

Chương 4

Kết luận

4.0.1 Kết luận

Các mô hình hồi quy tuyến tính tương đối đơn giản và cung cấp một công thức toán học dễ giải thích để đưa ra các dự đoán. Hồi quy tuyến tính là một kỹ thuật thống kê được sử dụng từ lâu và áp dụng dễ dàng cho phần mềm và tính toán. Các nhà khoa học trong nhiều lĩnh vực sử dụng hồi quy tuyến tính để tiến hành phân tích dữ liệu sơ bộ và dự đoán các xu hướng tương lai. Nhiều phương pháp khoa học dữ liệu, chẳng hạn như máy học và trí tuệ nhân tạo, sử dụng hồi quy tuyến tính để giải quyết các bài toán phức tạp. Việc sử dụng các hàm có sẵn trong Python để xây dựng mô hình hồi quy tuyến tính sẽ thuận tiện hơn nhiều so với các phương pháp bình phương nhỏ nhất và Gradient Descent mà kết quả thu được vẫn tối ưu. Khi xây dựng mô hình hồi quy tuyến tính chúng ta cần tiền xử lý (pre-processing) dữ liệu thì mô hình thu được sẽ phù hợp, nếu chúng ta có nhiều sai lầm trong việc tiền xử lý thì mô hình có thể không còn tốt.

4.0.2 Hướng phát triển

Trong quá trình làm đề tài, em nhận thấy có một số hướng có thể phát triển để mô hình hồi quy tuyến tính được tốt hơn:

- Loại bỏ giá trị ngoại lai: Hồi quy tuyến tính rất nhạy cảm với nhiễu. Vì vậy, trước khi xây dựng mô hình hồi quy tuyến tính, các giá trị ngoại lai (outlier)

cần phải được loại bỏ. Bước này được gọi là tiền xử lý (pre-processing)

- Loại bỏ tương quan giữa các biến : Mô hình hồi quy tuyến tính sẽ không chính xác khi chúng ta có các biến đầu vào tương quan cao với nhau, hiện tượng này được gọi là đa cộng tuyến. Do vậy cần tính toán các hệ số tương quan theo cặp cho dữ liệu đầu vào của bạn và loại bỏ bớt biến đầu vào nếu nó có tương quan cao với biến khác.

Tài liệu tham khảo

Tiếng Việt

- [1] Vũ Hữu Tiệp "*Machine Learning cơ bản*"

Tiếng Anh

- [2] Zed A.Shaw "*Learn Python the Hard Way* "
- [3] Andreas C.Muller, Sarah Guido "*Introduction to Machine Learning with Python* "