



---

BÀI TẬP THỰC HÀNH

---

BỘ MÔN HỆ THỐNG THÔNG TIN



# LẬP TRÌNH PHÂN TÍCH DỮ LIỆU

OCTOBER 5, 2023

TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP TP. HỒ CHÍ MINH

## MỤC LỤC

BÀI TẬP TUẦN 1 .....	2
1) ÔN PYTHON .....	2
2) LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG.....	2
3) CÁC BÀI LAB KHÁC.....	12
BÀI TẬP TUẦN 2 .....	13
1) PANDAS CƠ BẢN.....	13
2) CÁC BÀI LAB KHÁC.....	18
BÀI TẬP TUẦN 3 - SEABORN .....	19
1) CÁC DẠNG BIỂU ĐỒ .....	19
BÀI TẬP TUẦN 4 -5: LÀM SẠCH DỮ LIỆU – KHAI THÁC DỮ LIỆU .....	31
1) CÁC BÀI LAB LÀM SẠCH DỮ LIỆU .....	31
2) KHAI THÁC VÀ XỬ LÝ DỮ LIỆU – TRỰC QUAN HOÁ DỮ LIỆU .....	31
3) MINI PROJECT XỬ LÝ DỮ LIỆU BẰNG ĐIỂM BẰNG PANDAS.....	61
4) KẾT NỐI VỚI SQL SERVER.....	64
BÀI THỰC HÀNH TUẦN 5: THỐNG KÊ SUY DIỄN .....	67
1) CÁC BÀI TOÁN VỀ KIỂM ĐỊNH .....	67
BÀI TẬP TUẦN 6: HỒI QUY TUYẾN TÍNH .....	86
1) LAB 1 HỒI QUY TUYẾN TÍNH.....	99
2) LAB 2 HỒI QUY TUYẾN TÍNH.....	105
BÀI TOÁN PHÂN LỚP – CLASSIFICATION - CLUSTERING.....	111
1) BÀI 1.....	111
2) BÀI 2.....	118
3) BÀI 3.....	126
4) BÀI 4 .....	128
5) BÀI 5 .....	135
6) BÀI 6 .....	135
7) BÀI 7 .....	142

## BÀI TẬP TUẦN 1

### 1) ÔN PYTHON

### 2) LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

#### LAB EMPLOYEES

```
class Employee:
    def __init__(self,code,name,age,salary):
        self.code = code
        self.name = name
        self.age = age
        self.salary = salary
    def income(self):
        result = 0.9*12*self.salary
        return result
    def increaseSalary(self,amount):
        if amount > 0:
            self.salary = self.salary+ amount
    def display(self):
        print(f'code: {self.code}, name: {self.name}, age: {self.age}, salary: {self.salary},
income: {self.income()}\n')
```

#### File 2: LabEmployee.py

```
'''
Khai báo đối tượng Employee có:
Các thuộc tính (fields, states) sau
- mã số (code)
- tên (name)
- tuổi (age)
- lương (salary)
Các phương thức (behaviors, methods)
- Tổng thu nhập hằng năm sau thuế (income) biết rằng tax = 10%
- In ra thông tin của nhân viên (display)
- Tăng lương cho nhân viên (increaseSalary), biết rằng số lương tăng phải lớn hơn 0
- (Sinh viên tự viết) Giảm lương cho nhân viên (decreaseSalary), biết rằng số lương giảm
phải lớn hơn 0 và không vượt quá 20% lương hiện tại
===== YÊU CẦU CHƯƠNG TRÌNH=====
'''
```

Khai báo biến danh sách (list) nhân viên (dsNhanVien) để lưu trữ các nhân viên và viết chương trình menu thực hiện các chức năng bên dưới

- Opt-1: Tải danh sách nhân viên từ file dbemp\_input.db
- Opt-2: Thêm nhân viên vào danh sách
- Opt-3: Hiển thị danh sách nhân viên
- Opt-4: Hiển thị thông tin của một nhân viên khi biết mã nhân viên
- Opt-5: Chỉnh sửa thông tin một nhân viên
- Opt-6: Xóa một nhân viên ra khỏi danh sách
- Opt-7: Tăng lương cho một nhân viên
- Opt-8: Giảm lương cho một nhân viên
- Opt-9: Tính số lượng nhân viên (countEmp) và xuất ra màn hình
- Opt-10: Tính tổng tiền lương của công ty phải trả hàng tháng (sumSalary) và xuất ra màn hình
- Opt-11: Tính trung bình lương của nhân viên (avgSalary) và xuất ra màn hình
- Opt-12: Tính độ tuổi trung bình của nhân viên (avgAge) và xuất ra màn hình
- Opt-13: Tính tuổi lớn nhất của các nhân viên (maxAge) và hiển thị danh sách nhân viên có tuổi lớn nhất
- Opt-14: Sắp xếp danh sách nhân viên tăng dần theo lương
- Opt-15: Vẽ biểu đồ tương quan lương theo độ tuổi
- Opt-16: Vẽ biểu đồ so sánh lương trung bình của các nhóm tuổi: nhỏ hơn 35, từ 35 đến 50, hơn 50 trở lên
- Opt-17: Vẽ biểu đồ thể hiện phần trăm tổng lương trên các nhóm tuổi như Opt-16
- Opt-18: Vẽ biểu đồ thể hiện phần trăm số lượng nhân viên theo các nhóm tuổi như Opt-16
- Opt-19: Lưu danh sách nhân viên xuống file dbemp\_output.db, biết rằng mỗi nhân viên là một dòng và các thông tin nhân viên được phân cách bởi dấu '-'
- Opt-Khác: Thoát chương trình

'''

```
import matplotlib.pyplot as plt
```

```
import Employee as emp
```

```
menu_options = {
    1:'Load data from file',
    2:'Add new employee',
    3:'Display list of employee',
    4:'Show employee details',
```

```

5:'Update employee information',
6:'Delete employee',
7:'Increase salary of employee',
8:'Decrease salary of employee',
9:'Show total employee a month',
10:'Show total salary a month',
11:'Show average of salary a month',
12:'Show average of age',
13:'Show maximum age',
14:'Sort list of employee according to salary by ascending',
15: 'Draw salary according to age',
16:'Draw average of salary chart by age group',
17:'Draw percentage of salary by age group',
18:'Draw percentage of total employee by age group',
19:'Store data to file',
'Others': 'Exit program'
}

def print_menu():
    for key in menu_options.keys():
        print (key, '--', menu_options[key] )

# Khai báo biến lưu trữ những nhân viên
dsNhanVien = []

while(True):
    print_menu()
    userChoice = "
    try:
        userChoice = int(input('Input choice: '))
    except:
        print('Invalid input, try again')
        continue
    #Check what choice was entered and act accordingly
    if userChoice == 1:
        fr = open('dbemp_input.db',mode='r',encoding='utf-8')

```

```

for line in fr:
    stripLine = line.strip('\n')
    ds = stripLine.split(',')
    maso = ds[0]
    ten = ds[1]
    tuoi = int(ds[2])
    luong = float(ds[3])
    nv = emp.Employee(maso,ten,tuoi,luong)
    dsNhanVien.append(nv)
fr.close()
elif userChoice == 2:
    maso = input("Input code: ")
    ten = input("Input name: ")
    tuoi = int(input("Input age: "))
    luong = float(input("Input salary: "))
    nv = emp.Employee(maso,ten,tuoi,luong)
    dsNhanVien.append(nv)
elif userChoice == 3:
    if dsNhanVien.count==0:
        print('Danh sach rong')
    else:
        for item in dsNhanVien:
            item.display()
elif userChoice == 4:
    #4:Show employee details
    if dsNhanVien.count==0:
        print('Danh sach rong')
    else:
        ma =input("Input Code:")
        for item in dsNhanVien:
            if(item.code ==ma):
                item.display()
elif userChoice == 5:
    #5:Update employee information
    if dsNhanVien.count==0:
        print('Danh sach rong')

```

```

else:
    ma =input("Input Code for Update:")
    for item in dsNhanVien:
        if(item.code ==ma):
            item.display()
            menu={
                1:'Update Name',
                2:'Update Age',
                3:'Update luong',
                'Others':'Thoat'
            }
            def Xuat_menu():
                for key in menu.keys():
                    print(key,'--',menu[key])
            while (True):
                Xuat_menu()
                traloi=""
                try:
                    traloi =int(input('Nhap cac tuy chon:'))
                except:
                    print('Nhap sai dinh dang, nhap lai:')
                    continue
                if traloi==1:
                    ten = input("Input name: ")
                    item.name =ten
                    item.display()
                elif traloi ==2:
                    tuoi = int(input("Input age: "))
                    item.age =tuoi
                    item.display()
                elif traloi ==3:
                    luong = int(input("Input salary: "))
                    item.salary =luong
                    item.display()
                else:
                    print('Ket thuc chinh sua')

```

```

        break

elif userChoice == 6:
    #6:Delete employee
    if dsNhanVien.count==0:
        print('Danh sach rong')
    else:
        ma =input("Input Code for Update:")
        for item in dsNhanVien:
            if(item.code ==ma):
                item.display()
                tl = input('Ban co chac chan xoa nhan vien nay khong Y/N?')
                if tl =='Y':
                    #del item
                    dsNhanVien.remove(item)
        for item in dsNhanVien:
            item.display()
elif userChoice == 7:
    #7:Increase salary of employee
    if dsNhanVien.count==0:
        print('Danh sach rong')
    else:
        ma =input("Input Code for Update:")
        for item in dsNhanVien:
            if(item.code ==ma):
                item.display()
                luongtang = int(input('Nhap muc luong tang'))
                item.salary = item.salary + luongtang
                item.display()
elif userChoice == 8:
    #8:Decrease salary of employee
    if dsNhanVien.count==0:
        print('Danh sach rong')
    else:
        ma =input("Input Code for Update:")
        for item in dsNhanVien:

```



```

        if(item.code ==ma):
            item.display()
            luonggiam = int(input('Nhap muc luong giam'))
            item.salary = item.salary -luonggiam
            item.display()
    elif userChoice == 9:
        #9:'Show total employee a month'
        if dsNhanVien.count==0:
            print('Danh sach rong')
        else:
            tongsnv=0
            for item in dsNhanVien:
                tongsnv = tongsnv+1
                item.display()
            print('Tong so nhan vien =',tongsnv)
    elif userChoice == 10:
        sumSalary = 0.0
        for item in dsNhanVien:
            sumSalary = sumSalary + item.salary
        print(f'Total salary: {sumSalary}')
    elif userChoice == 11:
        #11:'Show average of salary a month'
        if dsNhanVien.count==0:
            print('Danh sach rong')
        else:
            tongsnv=0
            tongluong=0
            for item in dsNhanVien:
                tongsnv = tongsnv+1
                tongluong =tongluong +item.salary
                item.display()
            luongtb = tongluong/tongsnv
            print(f'Luong trung binh nhan vien ={luongtb}')
    elif userChoice == 12:
        #12:'Show average of age'
        if dsNhanVien.count==0:

```

```

        print('Danh sach rong')
    else:
        tongtuoi=0
        tongsnv=0
        for item in dsNhanVien:
            tongsnv =tongsnv+1
            tongtuoi = tongtuoi+item.age
            item.display()
        tuoitb = tongtuoi/tongsnv
        print(f'Tuoi trung binh nhan vien ={tuoi tb}')
elif userChoice == 13:
    #13:'Show maximum age'
    if dsNhanVien.count==0:
        print('Danh sach rong')
    else:
        for item in dsNhanVien:
            tuoimax=item.age
            break
        for item in dsNhanVien:
            if(item.age>tuoimax):
                tuoimax = item.age
        print('Tuoi lon nhat =',tuoimax)
elif userChoice == 14:
    #14:'Sort list of employee according to salary by ascending'
    if dsNhanVien.count==0:
        print('Danh sach rong')
    else:
        tongsnv=0
        for item in dsNhanVien:
            tongsnv = tongsnv+1
            item.display()
        print('Tong so nhan vien =',tongsnv)
elif userChoice == 15:
    #Draw salary according to age
    arrTuoi = []
    arrLuong = []

```

```

for item in dsNhanVien:
    arrTuoi.append(item.age)
    arrLuong.append(item.salary)

# Vẽ đồ thị
plt.figure(figsize=(15,5))

plt.title("Age and salary chart")
plt.xlabel("Ox: age")
plt.ylabel("Oy: salary")

plt.plot(arrTuoi,arrLuong, "go")
plt.show()
elif userChoice == 16:
    #16:'Draw average of salary chart by age group',
    arrTuoi = []
    arrLuong = []
    for item in dsNhanVien:
        arrTuoi.append(item.age)
        arrLuong.append(item.salary)

    # Vẽ đồ thị
    plt.figure(figsize=(15,5))

    plt.title("Age and salary chart")
    plt.xlabel("Ox: age")
    plt.ylabel("Oy: salary")

    plt.plot(arrTuoi,arrLuong, "go")
    plt.show()
elif userChoice == 17:
    #17:'Draw percentage of salary by age group'
    arrTuoi = []
    arrLuong = []
    for item in dsNhanVien:
        arrTuoi.append(item.age)

```

```

        arrLuong.append(item.salary)

# Vẽ đồ thị
plt.figure(figsize=(15,5))

plt.title("Age and salary chart")
plt.xlabel("Ox: age")
plt.ylabel("Oy: salary")

plt.plot(arrTuoi,arrLuong, "go")
plt.show()
elif userChoice == 18:
    #18:'Draw percentage of total employee by age group'
    arrTuoi = []
    arrLuong = []
    for item in dsNhanVien:
        arrTuoi.append(item.age)
        arrLuong.append(item.salary)

# Vẽ đồ thị
plt.figure(figsize=(15,5))

plt.title("Age and salary chart")
plt.xlabel("Ox: age")
plt.ylabel("Oy: salary")

plt.plot(arrTuoi,arrLuong, "go")
plt.show()
elif userChoice == 19:
    #19:'Store data to file'
    arrTuoi = []
    arrLuong = []
    for item in dsNhanVien:
        arrTuoi.append(item.age)
        arrLuong.append(item.salary)

```

```
# Vẽ đồ thị
plt.figure(figsize=(15,5))

plt.title("Age and salary chart")
plt.xlabel("Ox: age")
plt.ylabel("Oy: salary")

plt.plot(arrTuoi,arrLuong, "go")
plt.show()
else:
    print('BYE BYE')
    break
```

### 3) CÁC BÀI LAB KHÁC

Lab-01-OnPython.pdf

Lab Day1

Lab Day2

## BÀI TẬP TUẦN 2

### 1) PANDAS CƠ BẢN

```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import random
5
```

Ô chứa mã <undefined>

# %% [code]

```
1 #1. Đọc File với các file trong thư mục dữ liệu. Ứng với mỗi file sinh viên thực hiện các câu sau
2 df =pd.read_csv('original_sales_data_edit.csv')
3 df
```

```
1 #3. Đọc File với các tùy chọn mặc định. Hiển thị 5 dòng dữ liệu đầu tiên
2 df =pd.read_csv('original_sales_data_edit.csv')
3 df.head(5)
```

```
1 #4. Đọc File với các tùy chọn mặc định. Hiển thị 5 dòng dữ liệu cuối cùng
2 df.tail(5)
```

```
1 #5. Chuyển kiểu dữ liệu cho 1 cột nào đó
2 df["YEAR_ID"]=df["YEAR_ID"].astype("int32")
3 df.head(5)
```

```
1 #6. Xem chiều dài của df, tương đương shape[0]
2 print('Len:', len(df))
```

```
1 #7. Xem thông tin dataframe vừa đọc được
2 df.info()
```

```
1 #8. Xem kích thước của dataframe
2 print('Shape:', df.shape)
```

```
1 #9. Hiển thị dữ liệu của cột thứ 10
2 df['PRODUCTLINE']
```

```
1 #10. Hiển thị dữ liệu của cột 1,2,3,5,6
2 df[['ORDERNUMBER', 'QUANTITYORDERED', 'PRICEEACH', 'PRODUCTLINE']]
```

```
1 #11. Hiển thị 5 dòng dữ liệu đầu tiên gồm các cột 1,2,3,5,6
2 df[['ORDERNUMBER', 'QUANTITYORDERED', 'PRICEEACH', 'PRODUCTLINE']].head(5)
```

```
1 #12. Hiển thị 5 dòng dữ liệu đầu tiên theo chỉ số
2 df[0:5]
```

```
1 #13. Hiển thị 5 dòng dữ liệu đầu tiên theo chỉ số gồm các cột 1,2,3,5,6
2 df[['ORDERNUMBER', 'QUANTITYORDERED', 'PRICEEACH', 'PRODUCTLINE']][0:5]
```

```
1 #14. loại bỏ các dòng trùng nhau
2 df.drop_duplicates(inplace=True)
3 df
```

```
1 #15. Loại bỏ các dòng trống có dữ liệu của 1 cột là trống, có 2823 dòng
2 df['ADDRESSLINE2'].isna().sum() #Còn 2521 dòng
```

```
1 #16. Loại bỏ các dòng không biết của 1 cột có giá trị là Unknown, có 2521 dòng
2 df["ADDRESSLINE2"].fillna('Unknown',inplace=True)
3 df['ADDRESSLINE2'].isna().sum()
4 df
```

```
1 #17. Lấy dữ liệu của 1 cột theo dạng chuỗi
2 df["QUANTITYORDERED"]
```

```
1 #18. Lấy dữ liệu của 1 cột về một mảng
2 df["QUANTITYORDERED"].values
3
```

```
1 #20. Lấy dữ liệu từ dòng số 4 đến dòng 9
2 df[4:10]
```

```
1 #20. Lấy dữ liệu từ dòng số 4 đến dòng 9
2 df[4:10]
```

```
1 #21. Đọc dữ liệu từ dòng 4 đến dòng 9
2 df.loc[4:10]
3 df.iloc[4:10]
```

```
1 #22. Lấy thông tin tại dòng có chỉ số là 2
2 df.loc[2]
```

```
1 #23. Lấy thông tin từ dòng 4 đến dòng 10 của một số cột
2 df.loc[4:10,['QUANTITYORDERED','SALES']]
```

```
1 #24. Lấy thông tin dòng 2 đến dòng 9, từ cột 4 đến cột 7
2 df.iloc[2:9,4:7]
```

```
1 #25. Lấy dữ liệu tại chỉ số (index) là 2
2 df.iloc[2]
```

```
1 #26. Lấy dữ liệu từ dòng đầu tiên đến dòng 9 dùng iloc
2 df.iloc[:10]
```

```
1 #27. Lấy dữ liệu từ dòng đầu tiên đến dòng 9 gồm các cột 4 đến cột 7 dùng iloc
2 df.iloc[2:9,4:7]
```

```
1 #28. Sắp xếp dữ liệu theo sales tăng dần
2 df.sort_values(by='SALES',ascending=True)
```

```
1 #29. Sắp xếp dữ liệu theo nhiều tiêu chí
2 df.sort_values(by=['QUANTITYORDERED','PRICEEACH'],ascending=[True,False])
```

```
1 #30. Lọc dữ liệu theo 1 điều kiện
2 df[df['SALES']>5000]
3 print('Cách 2')
4 df.loc[df['SALES']>5000]
```

```
1 #31. Lọc dữ liệu theo nhiều điều kiện
2 df[(df['SALES']>5000) & (df['QUANTITYORDERED']>40)]
```

```
1 #32. Lọc giá trị và gán điều kiện dùng loc
2 df.loc[df['PRICEEACH']>=65, 'FLAG']='EXPENSIVE'
3 df.loc[df['PRICEEACH']<65, 'FLAG']='CHEAP'
4 #df['FLAG']
5 df[['PRICEEACH', 'FLAG']]#[:50]
```

```
1 #33. Viết hàm trả về giá trị có nhiều điều kiện và áp dụng hàm gán trị trả về cho 1 cột
2 def foo(x):
3     if x<10:
4         return 'BAD'
5     elif x>=10 and x<50:
6         return 'GOOD'
7     else:
8         return 'EXCELLENT'
9 df['WORTH']=df[['QUANTITYORDERED']].applymap(foo)
10 df[['QUANTITYORDERED', 'WORTH']]
```

```
1 #34. Ánh xạ giá trị tối 1 cột
2 dict_map1 ={'Qui_1':1, 'Qui_2':2, 'Qui_3':3, 'Qui_4':4}
3 df['QTR_ID']=df['QTR_ID'].map(dict_map1)
4 df['QTR_ID']
```

```
1 #35. Lấy những dòng dữ liệu bằng 1 điều kiện nào đó
2 df[['QUANTITYORDERED', 'PRICEEACH']].loc[df['YEAR_ID']==2003]
```

```
1 #36. Hiển thị các bản ghi có số lượng hơn 25
2 df[df['QUANTITYORDERED']>25]
3 print("Hiển thị 5 dòng")
4 Tuoitre =df[df['QUANTITYORDERED']>25]
5 Tuoitre[:5]
```

```
1 #37. Hiển thị các hoá đơn đã được giao
2 dg=df[df['STATUS'] == 'Shipped']
3 dg[:10]
```

```
1 #37. So sánh chuỗi
2 sosanh =df['STATUS'].str.contains('Shipped')
3 sosanh.head(5)
```

```
1 #38 Lấy giá trị trả về mảng
2 df['STATUS'].values
Kết quả thực thi
0KB
text/plain
array(['Shipped', 'Shipped', 'Shipped', ..., 'Resolved', 'Shipped',
      'On Hold'], dtype=object)
```



```
1 #40. Thêm 1 cột vào file dữ liệu
2 df_len =len(df)
3 ngaylap =[random.randrange(2003,2005,1) for i in range(df_len)]
4 df['ORDERDATE']=ngaylap
5 df.tail(5)
```

```
1 #41. Thêm 1 cột (Dagiao)vào dữ liệu theo tiêu chí nếu điều kiện thoả thì giá trị mặc định là True, ngược lại là False.
2 df['DaGiao']=df['STATUS']=='Shipped'
3 df.head(5)
```

```
1 #42. Tạo 1 cột mới có giá trị rỗng
2 df['TONGTIEN']=None
3 df
```

```
1 #44. Sửa giá trị của cột
2 df['QTR_ID']=None
3 df['QTR_ID']='Quy '
4 df.head(5)
```

```
1 #45. Xóa cột trong dataframe
2 df.drop(['TONGTIEN'],axis=1)
3 df.head(5)
```

```
1 #46. Xóa bản ghi theo chỉ số
2 df.drop([0,1]) #Xoá bản ghi có chỉ số 1 và 2
```

```
1 #47.Sử dụng hàm describe() để thống kê dữ liệu
2 df.describe()
```

```
1 #48. Xem thống kê trên từng cột
2 df['STATUS'].value_counts()
```

```
1 #49. Vẽ đồ thị xem phân bố giá trị của 1 trường trong dataframe
2 df['STATUS'].value_counts().plot(kind='bar')
```

```
1 #50. Tạo mới dataframe từ các python list
2 peoples = {'name': ['Nguyễn Văn Hiếu', 'Hiếu Nguyễn Văn'], 'age': [28, 28], 'website': ['https://blog.luyencode.net', None]}
3 df1 = pd.DataFrame(peoples)
4 print(df1)
```

```
1 #Tạo mới dataframe từ các python list
2 txts = ['chỗ này ăn cũng khá ngon', 'ngon, nhất định sẽ quay lại', 'thái độ phục vụ quá tệ']
3 labels = [1, 1, 0]
4 df1 = pd.DataFrame()
5 df1['txt'] = txts
6 df1['label'] = labels
7 print(df1)
```

```
1 #51. Sắp xếp dataframe
2 # Sắp xếp df tăng dần theo cột nào đó
3 df1 = pd.DataFrame({'name': ['Nam', 'Hiếu', 'Mai', 'Hoa'], 'age': [18,18,17,19]})
4 print('Before sort\n', df1)
5 df1 = df1.sort_values('age', ascending=True)
6 print('After sort\n', df1)
```

```

1 #52. Nối 2 dataframe
2 # Gộp 2 dataframe
3 df_1 = pd.DataFrame({'name': ['Hiếu'], 'age': [18], 'gender': ['male']})
4 df_2 = pd.DataFrame({'name': ['Nam', 'Mai', 'Hoa'], 'age': [15,17,19]})
5 df_3 = df_1.append(df_2, sort=True)
6 print(df_3)

```

```

1 #53. Xáo trộn các bản ghi trong dataframe
2 df1 = pd.DataFrame({'name': ['Hiếu', 'Nam', 'Mai', 'Hoa'], 'age': [18,15,17,19]})
3 print('Before shuffle\n', df)
4 df1 = df.sample(frac=1).reset_index(drop=True)
5 print('After shuffle\n', df1)

```

```

1 #54. Lưu dataframe về file csv
2 df1.to_csv('KHACHHANG.csv')

```

```

1 #55. Tối ưu bộ nhớ khi dùng pandas
2 import numpy as np # linear algebra
3 import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
4
5 def reduce_mem_usage(df):
6     """ iterate through all the columns of a dataframe and modify the data type
7     to reduce memory usage.
8     """
9     start_mem = df.memory_usage().sum() / 1024**2
10    print('Memory usage of dataframe is {:.2f} MB'.format(start_mem))
11
12    for col in df.columns:
13        col_type = df[col].dtype
14
15        if col_type != object and col_type.name != 'category' and 'datetime' not in col_type.name:
16            c_min = df[col].min()
17            c_max = df[col].max()
18            if str(col_type)[:3] == 'int':
19                if c_min > np.iinfo(np.int8).min and c_max < np.iinfo(np.int8).max:
20                    df[col] = df[col].astype(np.int8)
21                elif c_min > np.iinfo(np.int16).min and c_max < np.iinfo(np.int16).max:
22                    df[col] = df[col].astype(np.int16)
23                elif c_min > np.iinfo(np.int32).min and c_max < np.iinfo(np.int32).max:
24                    df[col] = df[col].astype(np.int32)
25                elif c_min > np.iinfo(np.int64).min and c_max < np.iinfo(np.int64).max:
26                    df[col] = df[col].astype(np.int64)
27            else:
28                if c_min > np.finfo(np.float16).min and c_max < np.finfo(np.float16).max:
29                    df[col] = df[col].astype(np.float16)
30                elif c_min > np.finfo(np.float32).min and c_max < np.finfo(np.float32).max:
31                    df[col] = df[col].astype(np.float32)
32                else:
33                    df[col] = df[col].astype(np.float64)
34            elif 'datetime' not in col_type.name:
35                df[col] = df[col].astype('category')

```

```

34         elif 'datetime' not in col_type.name:
35             df[col] = df[col].astype('category')
36
37     end_mem = df.memory_usage().sum() / 1024**2
38     print('Memory usage after optimization is: {:.2f} MB'.format(end_mem))
39     print('Decreased by {:.1f}%'.format(100 * (start_mem - end_mem) / start_mem))
40
41     return df
42 df

```

## 2) CÁC BÀI LAB KHÁC

## BÀI TẬP TUẦN 3 - SEABORN

### 1) CÁC DẠNG BIỂU ĐỒ

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
```

```
#loading dataset
#Iris dataset contains five columns such as Petal Length, Petal Width, Sepal Length,
Sepal Width and Species Type. Iris is a flowering plant, the researchers have measured
various features of the different iris flowers and recorded them digitally.
data =sns.load_dataset("iris")
#draw lineplot
sns.lineplot(x="sepal_length",y="sepal_width",data=data)
```

```
#Seaborn is built on the top of Matplotlib. It means that Seaborn can be used with
Matplotlib.
#Using Seaborn with Matplotlib
#We will be using the above example and will add the title to the plot using the
Matplotlib.
#Draw lineplot
sns.lineplot(x="sepal_length",y="sepal_width",data=data)
#setting the title using Matplotlib
plt.title("Title using Matplotlib Function")
plt.show()
```

```
#Setting the xlim and ylim
# draw lineplot
sns.lineplot(x="sepal_length", y="sepal_width", data=data)
# setting the x limit of the plot vẽ bd với x=5
plt.xlim(5)
plt.show()
```

```
#Customizing Seaborn Plots
```

#set\_style() method is used to set the aesthetic of the plot. It means it affects things like the color of the axes, whether the grid is active or not, or other aesthetic elements. There are five themes available in Seaborn:

#darkgrid, whitegrid, dark, white, ticks

#set\_style(style=None, rc=None)

# draw lineplot

sns.lineplot(x="sepal\_length", y="sepal\_width", data=data)

# changing the theme to dark

sns.set\_style("dark")

plt.show()

#Removal of Spines

#Spines are the lines noting the data boundaries and connecting the axis tick marks. It can be removed using the despine() method.

#sns.despine(left = True)

# draw lineplot

sns.lineplot(x="sepal\_length", y="sepal\_width", data=data)

# Removing the spines

sns.despine(left=True)

plt.show()

#Changing the figure Size

# changing the figure size

plt.figure(figsize = (2, 4))

# draw lineplot

sns.lineplot(x="sepal\_length", y="sepal\_width", data=data)

# Removing the spines

sns.despine()

plt.show()

#Scaling the plots

#It can be done using the set\_context() method. It allows us to override default parameters. This affects things like the size of the labels, lines, and other elements of the plot, but not the overall style. The base context is “notebook”, and the other contexts are “paper”, “talk”, and “poster”. font\_scale sets the font size.

```
#set_context(context=None, font_scale=1, rc=None)
# draw lineplot
sns.lineplot(x="sepal_length", y="sepal_width", data=data)
# Setting the scale of the plot
sns.set_context("poster")
plt.show()
```

```
#Setting the Style Temporarily
#axes_style() method is used to set the style temporarily. It is used along with the with
statement.
#axes_style(style=None, rc=None)
def plot():
    sns.lineplot(x="sepal_length", y="sepal_width", data=data)
with sns.axes_style('darkgrid'):
    # Adding the subplot
    plt.subplot(211)
    plot()
plt.subplot(212)
plot()
```

```
#Color Palette
#color_palette() method is used to give colors to the plot.
#palplot() is used to deal with the color palettes and plots the color palette as a horizontal
array.
# current colot palette
palette = sns.color_palette()
# plots the color palette as a
# horizontal array
sns.palplot(palette)
plt.show()
```

```
#Diverging Color Palette: uses two different colors where each color depicts different
points ranging from a common point in either direction. Consider a range of -10 to 10 so
the value from -10 to 0 takes one color and values from 0 to 10 take another.
# current colot palette
palette = sns.color_palette('PiYG', 11)
```

```
# diverging color palette
sns.palplot(palette)
plt.show()
```

```
#Sequential Color Palette: is used where the distribution ranges from a lower value to a
higher value. To do this add the character 's' to the color passed in the color palette.
# current color palette
palette = sns.color_palette('Greens', 11)
# sequential color palette
sns.palplot(palette)
plt.show()
```

```
#Setting the default Color Palette
#set_palette() method is used to set the default color palette for all the plots. The
arguments for both color_palette() and set_palette() is same. set_palette() changes the
default matplotlib parameters.
def plot():
    sns.lineplot(x="sepal_length", y="sepal_width", data=data)
# setting the default color palette
sns.set_palette('vlag')
plt.subplot(211)
# plotting with the color palette
# as vlag
plot()
# setting another default color palette
sns.set_palette('Accent')
plt.subplot(212)
plot()
plt.show()
```

```
#Multiple plots with Seaborn
#Using Matplotlib: add_axes(), subplot(), and subplot2grid().
def graph():
    sns.lineplot(x="sepal_length", y="sepal_width", data=data)
# Creating a new figure with width = 5 inches
# and height = 4 inches
```

```
fig = plt.figure(figsize=(5, 4))
# Creating first axes for the figure
ax1 = fig.add_axes([0.1, 0.1, 0.8, 0.8])
# plotting the graph
graph()
# Creating second axes for the figure
ax2 = fig.add_axes([0.5, 0.5, 0.3, 0.3])
# plotting the graph
graph()
plt.show()
```

```
#Using subplot() method
def graph():
    sns.lineplot(x="sepal_length", y="sepal_width", data=data)
# Adding the subplot at the specified
# grid position
plt.subplot(121)
graph()
# Adding the subplot at the specified
# grid position
plt.subplot(122)
graph()
plt.show()
```

```
#Using subplot() method
def graph():
    sns.lineplot(x="sepal_length", y="sepal_width", data=data)
# Adding the subplot at the specified
# grid position
plt.subplot(121)
graph()
# Adding the subplot at the specified
# grid position
plt.subplot(122)
graph()
plt.show()
```



```
#Using subplot2grid() method
def graph():
    sns.lineplot(x="sepal_length", y="sepal_width", data=data)
# adding the subplots
axes1 = plt.subplot2grid (
    (7, 1), (0, 0), rowspan = 2, colspan = 1)
graph()
axes2 = plt.subplot2grid (
    (7, 1), (2, 0), rowspan = 2, colspan = 1)
graph()
axes3 = plt.subplot2grid (
    (7, 1), (4, 0), rowspan = 2, colspan = 1)
graph()
#Using Seaborn: Using FacetGrid() method
#FacetGrid class helps in visualizing distribution of one variable as well as the
relationship between multiple variables separately within subsets of your dataset using
multiple panels.
#seaborn.FacetGrid( data, \*\*kwargs)
plot = sns.FacetGrid(data, col="species")
plot.map(plt.plot, "sepal_width")
plt.show()
```

```
#Using PairGrid() method
#Subplot grid for plotting pairwise relationships in a dataset.
#This class maps each variable in a dataset onto a column and row in a grid of multiple
axes. Different axes-level plotting functions can be used to draw bivariate plots in the
upper and lower triangles, and the marginal distribution of each variable can be shown on
the diagonal.
#seaborn.PairGrid( data, \*\*kwargs)
data = sns.load_dataset("flights")
plot = sns.PairGrid(data)
plot.map(plt.plot)
plt.show()
#Creating Different Types of Plots
#Relplot()
```

```
#This function provides us the access to some other different axes-level functions which shows the relationships between two variables with semantic mappings of subsets. It is plotted using the relplot() method
#seaborn.relplot(x=None, y=None, data=None, **kwargs)
# loading dataset
data = sns.load_dataset("iris")
# creating the relplot
sns.relplot(x='sepal_width', y='species', data=data)
plt.show()
```

```
#The scatter plot is a mainstay of statistical visualization. It depicts the joint distribution of two variables using a cloud of points, where each point represents an observation in the dataset. This depiction allows the eye to infer a substantial amount of information about whether there is any meaningful relationship between them. It is plotted using the scatterplot() method
#seaborn.scatterplot(x=None, y=None, data=None, **kwargs)
sns.scatterplot(x='sepal_length', y='sepal_width', data=data)
plt.show()
```

```
#The scatter plot is a mainstay of statistical visualization. It depicts the joint distribution of two variables using a cloud of points, where each point represents an observation in the dataset. This depiction allows the eye to infer a substantial amount of information about whether there is any meaningful relationship between them. It is plotted using the scatterplot() method
#seaborn.scatterplot(x=None, y=None, data=None, **kwargs)
sns.scatterplot(x='sepal_length', y='sepal_width', data=data)
plt.show()
```

```
#Line Plot: For certain datasets, you may want to consider changes as a function of time in one variable, or as a similarly continuous variable. In this case, drawing a line-plot is a better option. It is plotted using the lineplot() method.
#seaborn.lineplot(x=None, y=None, data=None, **kwargs)
sns.lineplot(x='sepal_length', y='species', data=data)
plt.show()
```

#Categorical Plots are used where we have to visualize relationship between two numerical values. A more specialized approach can be used if one of the main variable is categorical which means such variables that take on a fixed and limited number of possible values.

#A barplot is basically used to aggregate the categorical data according to some methods and by default its the mean. It can also be understood as a visualization of the group by action. To use this plot we choose a categorical column for the x axis and a numerical column for the y axis and we see that it creates a plot taking a mean per categorical column. It can be created using the barplot() method.

```
#barplot([x, y, hue, data, order, hue_order, ...])
sns.barplot(x='species', y='sepal_length', data=data)
plt.show()
```

#A countplot basically counts the categories and returns a count of their occurrences. It is one of the most simple plots provided by the seaborn library. It can be created using the countplot() method.

```
#countplot([x, y, hue, data, order, ...])
sns.countplot(x='species', data=data)
plt.show()
```

#A boxplot is sometimes known as the box and whisker plot. It shows the distribution of the quantitative data that represents the comparisons between variables. boxplot shows the quartiles of the dataset while the whiskers extend to show the rest of the distribution i.e. the dots indicating the presence of outliers. It is created using the boxplot() method.

```
#boxplot([x, y, hue, data, order, hue_order, ...])
sns.boxplot(x='species', y='sepal_width', data=data)
plt.show()
```

#Violinplot

#It is similar to the boxplot except that it provides a higher, more advanced visualization and uses the kernel density estimation to give a better description about the data distribution. It is created using the violinplot() method.

```
#violinplot([x, y, hue, data, order, ...])
sns.violinplot(x='species', y='sepal_width', data=data)
plt.show()
```

#Stripplot: It basically creates a scatter plot based on the category. It is created using the stripplot() method.

```
#stripplot([x, y, hue, data, order, ...])  
sns.stripplot(x='species', y='sepal_width', data=data)  
plt.show()
```

#Swarmplot is very similar to the stripplot except the fact that the points are adjusted so that they do not overlap. Some people also like combining the idea of a violin plot and a stripplot to form this plot. One drawback to using swarmplot is that sometimes they don't scale well to really large numbers and takes a lot of computation to arrange them. So in case we want to visualize a swarmplot properly we can plot it on top of a violinplot. It is plotted using the swarmplot() method.

```
#swarmplot([x, y, hue, data, order, ...])  
sns.swarmplot(x='species', y='sepal_width', data=data)  
plt.show()
```

#Factorplot is the most general of all these plots and provides a parameter called kind to choose the kind of plot we want thus saving us from the trouble of writing these plots separately. The kind parameter can be bar, violin, swarm etc. It is plotted using the factorplot() method.

```
#sns.factorplot([x, y, hue, data, row, col, ...])  
# loading dataset  
data = sns.load_dataset("iris")  
sns.factorplot(x='species', y='sepal_width', data=data)  
plt.show()
```

#A histogram is basically used to represent data provided in a form of some groups. It is an accurate method for the graphical representation of numerical data distribution. It can be plotted using the histplot() function.

```
#histplot(data=None, *, x=None, y=None, hue=None, **kwargs)  
sns.histplot(x='species', y='sepal_width', data=data)  
plt.show()
```

#Distplot is used basically for univariate set of observations and visualizes it through a histogram i.e. only one observation and hence we choose one particular column of the dataset. It is plotted using the distplot() method.

```
#distplot(a[, bins, hist, kde, rug, fit, ...])
```

```
import seaborn as sns
import matplotlib.pyplot as plt
# loading dataset
data = sns.load_dataset("iris")
sns.distplot(data['sepal_width'])
plt.show()
```

#Jointplot is used to draw a plot of two variables with bivariate and univariate graphs. It basically combines two different plots. It is plotted using the jointplot() method.

```
#jointplot(x, y[, data, kind, stat_func, ...])
sns.jointplot(x='species', y='sepal_width', data=data)
plt.show()
```

#Pairplot represents pairwise relation across the entire dataframe and supports an additional argument called hue for categorical separation. What it does basically is create a jointplot between every possible numerical column and takes a while if the dataframe is really huge. It is plotted using the pairplot() method.

```
#pairplot(data[, hue, hue_order, palette, ...])
sns.pairplot(data=data, hue='species')
plt.show()
```

#Rugplot plots datapoints in an array as sticks on an axis. Just like a distplot it takes a single column. Instead of drawing a histogram it creates dashes all across the plot. If you compare it with the jointplot you can see that what a jointplot does is that it counts the dashes and shows it as bins. It is plotted using the rugplot() method.

```
#rugplot(a[, height, axis, ax])
sns.rugplot(data=data)
plt.show()
```

#KDE Plot described as Kernel Density Estimate is used for visualizing the Probability Density of a continuous variable. It depicts the probability density at different values in a continuous variable. We can also plot a single graph for multiple samples which helps in more efficient data visualization.

```
#seaborn.kdeplot(x=None, *, y=None, vertical=False, palette=None, **kwargs)
sns.kdeplot(x='sepal_length', y='sepal_width', data=data)
plt.show()
```

```
#The regression plots are primarily intended to add a visual guide that helps to emphasize patterns in a dataset during exploratory data analyses. Regression plots as the name suggests creates a regression line between two parameters and helps to visualize their linear relationships.
```

```
#lmpplot() method can be understood as a function that basically creates a linear model plot. It creates a scatter plot with a linear fit on top of it.
```

```
#seaborn.lmpplot(x, y, data, hue=None, col=None, row=None, **kwargs)
```

```
# loading dataset
```

```
data = sns.load_dataset("tips")
```

```
sns.lmpplot(x='total_bill', y='tip', data=data)
```

```
plt.show()
```

```
#regplot() method is also similar to lmpplot which creates linear regression model.
```

```
#seaborn.regplot( x, y, data=None, x_estimator=None, **kwargs)
```

```
# loading dataset
```

```
data = sns.load_dataset("tips")
```

```
sns.regplot(x='total_bill', y='tip', data=data)
```

```
plt.show()
```

```
#Heatmap is defined as a graphical representation of data using colors to visualize the value of the matrix. In this, to represent more common values or higher activities brighter colors basically reddish colors are used and to represent less common or activity values, darker colors are preferred. it can be plotted using the heatmap() function.
```

```
#seaborn.heatmap(data, *, vmin=None, vmax=None, cmap=None, center=None, annot_kws=None, linewidths=0, linecolor='white', cbar=True, **kwargs)
```

```
# loading dataset
```

```
import seaborn as sns
```

```
import matplotlib.pyplot as plt
```

```
data = sns.load_dataset("tips")
```

```
# correlation between the different parameters
```

```
tc = data.corr()
```

```
sns.heatmap(tc)
```

```
plt.show()
```

#The clustermap() function of seaborn plots the hierarchically-clustered heatmap of the given matrix dataset. Clustering simply means grouping data based on relationship among the variables in the data.

```
#clustermap(data, *, pivot_kws=None, **kwargs)
```

```
import seaborn as sns
```

```
import matplotlib.pyplot as plt
```

```
# loading dataset
```

```
data = sns.load_dataset("tips")
```

```
# correlation between the different parameters
```

```
tc = data.corr()
```

```
sns.clustermap(tc)
```

```
plt.show()
```

```
--loi
```

## BÀI TẬP TUẦN 4 -5: LÀM SẠCH DỮ LIỆU – KHAI THÁC DỮ LIỆU

### 1) CÁC BÀI LAB LÀM SẠCH DỮ LIỆU

- a) Lab4-data-wragling.pdf
- b) Lab4\_LamSachDuLieu\_new.pdf

### 2) KHAI THÁC VÀ XỬ LÝ DỮ LIỆU – TRỰC QUAN HOÁ DỮ LIỆU

#=====QUÁ TRÌNH EXPLANATORY DATA ANALYSIS=====

#Đây là quá trình khai thác thông tin từ dữ liệu thông qua biểu đồ.

#Khi vẽ biểu đồ chúng ta cần đặt ra các câu hỏi

- #1. Có bao nhiêu biến tham gia vào biểu đồ
- #2. Loại (định tính, định lượng) của từng biến số
- #3. Biểu đồ biểu diễn bao nhiêu thông tin
- #4. Tri thức gì được rút trích ra từ biểu đồ
- #5. Tri thức được rút trích có liên quan gì đến nghiệp vụ

#=====

#Bước 1: Xử lý dữ liệu cơ bản theo yêu cầu

#=====

#1.1. Đọc dữ liệu

#1.2. Loại bỏ dòng dữ liệu trống

#1.3. Loại bỏ dòng dữ liệu bị trùng

#1.4. Kiểm tra các dữ liệu thiếu bằng chart

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

import numpy as np

#1.1. Đọc dữ liệu

#df = pd.read\_csv('orginal\_sales\_data\_edit.csv')

df = pd.read\_csv('orginal\_sales\_data\_edit.csv', encoding='utf-8', header=0, delimiter=',')

df

#Kết quả: 2824 dòng

```
1 #1.2. Loại bỏ dòng dữ liệu rỗng
2 df.dropna(how='all', inplace=True)
3 df #2824 dòng
```

#1.3. Loại bỏ dữ liệu trùng, biết rằng dữ liệu trùng là dữ liệu có đồng thời ORDERNUMBER và OrDERDATE như nhau



```
#Kiểm tra lại nghiệp vụ này
df.drop_duplicates(inplace=True)
df #2824 dòng
#1.3. Loại bỏ dữ liệu trùng, biết rằng dữ liệu trùng là dữ liệu có đồng thời
ORDERNUMBER và OrDERDATE như nhau
#Kiểm tra lại nghiệp vụ này
df.drop_duplicates(inplace=True)
df #2824 dòng
```

```
#1.4. Kiểm tra dữ liệu thiếu bằng chart
# Cách 2: Trục quan dữ liệu thiếu với Seaborn Displt
plt.figure(figsize=(10,6))
sns.displot(data=df.isna().melt(value_name="missing"),
y="variable",
hue="missing",
multiple="fill",
aspect=1.25)
plt.savefig("my_missing_value_2.png",dpi=100)
#1.4.1. Điền thiếu dữ liệu với dữ liệu định tính
#1.4.1.1. Với dữ liệu biểu diễn dạng chuỗi thì thay bằng Unknown
#1.4.1.2. Với dữ liệu biểu diễn dạng số thì thay bằng 0
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```
#1.1. Đọc dữ liệu
df =pd.read_csv('orginal_sales_data_edit.csv', encoding='UTF8',header=0, delimiter=',')
```

```
#1.2. Loại bỏ dòng dữ liệu rỗng
print("Dữ liệu chưa loại bỏ")
df.info
df
#Dữ liệu sau khi loại bỏ
print("Dữ liệu sau khi loại bỏ")
df.dropna(how='all',inplace=True)
df
```

### #1.3. Loại bỏ dữ liệu trùng

#2823 dòng

```
df.drop_duplicates(inplace=True)
df
df.drop_duplicates(subset=['ORDERNUMBER','ORDERDATE'],inplace=True)
df[['ORDERNUMBER','ORDERDATE']]#Kết quả là 2733
```

### #1.4. Kiểm tra các dữ liệu thiếu bằng chart

#Trực quan dữ liệu thiếu với Seaborn HeatMap, cột màu đậm đang bị thiếu dữ liệu

```
plt.figure(figsize=(10,6))
sns.heatmap(df.isna().transpose(),cmap="YlGnBu",cbar_kws={'label':'Missing Data'})
plt.savefig("my_missing_value_1.png",dpi=100)
```

### #Cách 2: Trực quan dữ liệu thiếu bằng Seaborn Displot

```
plt.figure(figsize=(10,6))
sns.displot(data=df.isna().melt(value_name="missing"),y="variable",
hue="missing",multiple="fill",aspect=1.25)
plt.savefig("my_missing_value_2.png",dpi=100)
```

#### #1.4.1. Điền thiếu dữ liệu với dữ liệu định tính

#1.4.1.1. Với dữ liệu biểu diễn dạng chuỗi thì thay dữ liệu thiếu bằng Unknown

#1.4.1.2. Với dữ liệu biểu diễn dạng số thì thay bằng 0

```
df['ADDRESSLINE2'].fillna('Unknown', inplace=True)
df['STATE'].fillna('Unknow',inplace=True)
df['TERRITORY'].fillna('Unknown',inplace=True)
df['POSTALCODE'].fillna(0,inplace=True)
print("Xem lại thông tin")
df[['ADDRESSLINE2','STATE','TERRITORY','POSTALCODE']]
```

### #1.5. Tách 1 cột thành 2 cột. Sau đó xoá cột bị tách

```
df[['PAYMENTLASTNAME',
'PAYMENTFIRSTNAME']]=df['PAYMENTFULLNAME'].str.split(' ',expand=True)
df=df.drop('PAYMENTFULLNAME',axis=1)
df[['PAYMENTLASTNAME', 'PAYMENTFIRSTNAME']]
```

### #1.6. Lưu dữ liệu thành file mới

```
df.to_csv('processed_sales_data.csv',sep=',',encoding='utf-8',index=False)
```

#### #1.1.1. Cho biết tổng số lượng sản phẩm bán được của mỗi năm

```
sns.barplot(x='YEAR_ID',
            y='QUANTITYORDERED',data=df,errorbar=None,estimator=sum)
plt.show()
```

#### #1.1.2. Có bao nhiêu đơn hàng theo Dealsize

```
labels=df['DEALSIZE'].value_counts().index
values=df['DEALSIZE'].value_counts().values
colors=sns.color_palette('pastel')
plt.pie(values,labels=labels, colors=colors, autopct='% 1.1f%%', shadow=True)
plt.show()
```

### 1.1.3. HOẶC SỬ DỤNG NĂNG CAO VỚI NHIỀU TUY CHỈNH TRONG HÀM TỌA ĐỘ

#### #1.1.3. Cho biết tỉ lệ giá trị SALES theo DEALSIZE

```
gb=df.groupby(['DEALSIZE'])['ORDERNUMBER'].agg(['count'])
data=list(gb['count'])
labels=gb.index
colors=sns.color_palette('pastel')
plt.pie(values,labels=labels, colors=colors, autopct='% 1.1f%%', shadow=True)
plt.show()
```

### CHO BIẾT GIÁ TRỊ SALES THEO NGÀY

#### #1.1.7. Cho biết tổng giá trị SALES theo ngày

```
sns.lineplot(x="ORDERDATE",y="SALES",data=df, estimator=sum)
```

#### #1.1.8. Cho biết giá trị SALES trung bình theo tháng năm

```
#default estimator=mean
```

```
sns.lineplot(x="MONTH_ID", y="SALES", hue='YEAR_ID',data=df)
plt.show()
```

#### #1.1.8. Cho biết giá trị SALES trung bình theo tháng năm

```
#default estimator=mean
sns.lineplot(x="MONTH_ID", y="SALES", hue='YEAR_ID',data=df)
plt.show()
```

```
#1.1.8.a Cho biết giá trị SALES theo tháng năm
from numpy import count_nonzero
sns.lineplot (
x="MONTH_ID",y="SALES",hue='YEAR_ID',data=df,estimator=count_nonzero)
plt.show()
```

```
#1.1.9. Cho biết số lượng hoá đơn theo tháng, năm
#Dùng biểu đồ Line
from numpy import count_nonzero
sns.lineplot(x='MONTH_ID',y='ORDERNUMBER',hue='YEAR_ID',data=df,estimator=
count_nonzero)
plt.show()
```

```
#Cách 2: Dùng barplot
sns.barplot(x='STATUS', y='ORDERNUMBER',hue='YEAR_ID',data=df,
errorbar=None,estimator=count_nonzero)
plt.show()
```

CHO BIẾT GIÁ TRỊ ĐƠN HÀNG THEO TRẠNG THÁI (STATUS) THEO NHÓM CÁC NĂM(YEAR\_ID)

```
#Dùng biểu đồ barplot
sns.barplot(x='STATUS',y='SALES',hue='YEAR_ID',data=df,errorbar=None)
plt.show()
```

TỔNG GIÁ TRỊ ĐƠN HÀNG THEO TRẠNG THÁI(STATUS) THEO NHÓM CÁC NĂM (YEAR\_ID)

```
#1.2.1. Cho biết tổng giá trị đơn hàng theo từng trạng thái (STATUS) của mỗi năm
sns.barplot(x='STATUS',y='SALES',hue='YEAR_ID',data=df,errorbar=None,estimator=sum)
plt.show()
```

## SO LUONG HOA DON GIUA CAC DEALSIZE THEO YEAR\_ID

```
#1.2.3. Cho biết số lượng hoá đơn giữa các nhóm DEALSIZE theo từng năm YEAR_ID
gb=df.groupby(['DEALSIZE','YEAR_ID'])['ORDERNUMBER'].count().unstack()
gb.plot(kind='bar',stacked=True)
#just add a title and rotate the x-axis labels to the horizontal
plt.title('Number of Orders in DEALSIZE, YEAR_ID')
plt.xticks(rotation=0,ha='center')
plt.show()
```

## TRUNG BINH SALES CAC NHOM STATUS THEO DEALSIZE

```
#1.2.4. Cho biết trung bình SALES theo từng STATUS và DEALSIZE
gb=df.groupby(['STATUS','DEALSIZE'])['SALES'].mean().unstack()
gb.plot(kind='barh',stacked=True)
#just add a title and rotate the x-axis labels to the horizontal
plt.title(' TRUNG BINH SALES CAC NHOM THEO DEALSIZE')
plt.xticks(rotation=0,ha='center')
plt.show()
```

## 3. MÔ TẢ DỮ LIỆU

```
#3.1. Mô tả dữ liệu cột QuantityOrdered
df['QUANTITYORDERED'].describe()
```

## MO TA DU LIEU CUA QUANTITYORDERED, PRICEEACH, SALES

```
#3.2. Mô tả dữ liệu của QUANTITYORDERED, PRICEEACH
df[['QUANTITYORDERED','PRICEEACH','SALES']].describe()
```

## MO TA DU LIEU SALES THEO NHOM DEALSIZE

```
#Mô tả số lượng bán hàng theo từng DEALSIZE
df.groupby('DEALSIZE')['QUANTITYORDERED'].describe()
```

## PHẦN 4: KHAI THÁC SỰ PHÂN PHỐI DỮ LIỆU, CHỈ DÙNG TRÊN BIẾN ĐỊNH LƯỢNG

### #4.1. VẼ BIỂU ĐỒ HISTOGRAM CỦA QUANTITYORDERED

```
df['QUANTITYORDERED'].hist()
plt.show()
```

### #HOAC NANG CAO HON VOI NHIEU TUY CHON

```
# sns.displot(df,x="QUANTITYORDERED",kind="kde")
# plt.show()
```

### #HOAC NANG CAO HON VOI NHIEU TUY CHON

```
sns.displot(df,x="QUANTITYORDERED",kind="kde")
plt.show()
```

## CHO BIẾT XÁC SUẤT XẢY RA TRÊN MỖI ĐOẠN

### #4.2. VẼ BIỂU ĐỒ HIS CỦA QUANTITYORDERED THEO DEALSIZE //PHÂN PHỐI HIS CỦA QUANTITYORDERED THEO NHÓM DEALSIZE

#màu xanh lá cây ổn định nhất

```
sns.displot(df, x="QUANTITYORDERED",hue="DEALSIZE",kind="kde")
plt.show()
```

### #4.3. VẼ BIỂU ĐỒ HIS CỦA QUANTITYORDERED THEO DEALSIZE //PHÂN PHỐI HIS CỦA QUANTITYORDERED THEO NHÓM DEALSIZE

```
sns.displot(df, x="QUANTITYORDERED",hue="DEALSIZE",kind="ecdf")
plt.show()
```

### #4.4. VẼ BIỂU ĐỒ HIS CỦA QUANTITYORDERED THEO DEALSIZE //PHÂN PHỐI HIS CỦA QUANTITYORDERED THEO NHÓM DEALSIZE

```
sns.displot(df, x="QUANTITYORDERED",hue="DEALSIZE",kind="hist")
plt.show()
```

## VẼ BIỂU ĐỒ JOINPLOT CỦA QUANTITYORDERED VÀ PRICEEACH

### #4.5. VẼ BIỂU ĐỒ JOINPLOT CỦA QUANTITYORDERED VÀ PRICEEACH

```
sns.jointplot(data=df,x='PRICEEACH',y='QUANTITYORDERED',kind='kde',color='y')
```

```
#4.6. VẼ BIỂU ĐỒ PAIRPLOT CỦA QUANTITYORDERED, PRICEEACH, SALES
sns.pairplot(df[['QUANTITYORDERED', 'PRICEEACH', 'SALES']], diag_kind='hist', kind='kde')
plt.show()
```

## BIỂU DIỄN TƯƠNG QUAN BIẾN SỐ THEO TỪNG CẶP

```
#4.8. Vẽ trực quan số lượng sản phẩm phân phối theo năm của QuantityOrdered (catplot)
sns.catplot(x='YEAR_ID', y='QUANTITYORDERED', data=df)
plt.show()
# Hỏi câu hỏi:
```

## VE TRUC QUAN SO LUONG SAN PHAM PHAN PHOI THEO NAM CUA QUANTITYORDERED

```
#4.9. Vẽ trực quan số lượng sản phẩm phân phối theo năm của cột QUANTITYORDERED
sns.boxplot(data=df['QUANTITYORDERED'])
plt.show()
```

## BA CÁCH BIỂU DIỄN

```
#4.10.1. BIỂU THỊ CHO NGOẠI BIÊN
#BIỂU DIỄN (BOXPLOT, VIOLIN) CỦA QUANTITYORDERED TRÊN CUNG CHART
fig, ax = plt.subplots()
ax.plot(x, y)
ax.set_title('Simple plot')
#fig, axes = plt.subplots(2, 2, subplot_kw=dict(projection="polar"))
fig, axes = plt.subplots(nrows=1, ncols=2, sharey=True, figsize=(6,4))
sns.boxplot(data=df['QUANTITYORDERED'], ax=axes[0])
sns.violinplot(data=df['QUANTITYORDERED'], ax=axes[1])
plt.show()
-----loi
```

## BIỂU DIỄN BOXPLOT CỦA QUANTITYORDERED THEO NHÓM DEALSIZE

```
sns.boxplot(x='DEALSIZE', y='QUANTITYORDERED', data=df)
```

```
plt.show()
```

BIEU DIENCATPLOT CUA QUANTITYORDERED THEO NHOM DEALSIZE

```
sns.catplot(x='YEAR_ID',y='QUANTITYORDERED',hue='DEALSIZE',data=df)
plt.show()
```

BIEU DIEN BOXPLOT CUA QUANTITYORDERED THEO NAM,NHOM  
DEALSIZE

```
sns.boxplot(x='YEAR_ID',y='QUANTITYORDERED',hue='DEALSIZE',data=df)
plt.show()
```

#BIEU DIEN BOXPLOT CUA QUANTITYORDERED,PRICEEACH

```
sns.boxplot(data=df[['QUANTITYORDERED','PRICEEACH']])
plt.show()
```

#DO XIEN CUA PHAN PHOI (SKEW) CUA QUANTITYORDERED,PRICEEACH

```
df[['QUANTITYORDERED','PRICEEACH']].skew()
```

#DO NHON CUA PHAN PHOI (KURTOSIS) CUA QUANTITYORDERED ,  
#PRICEEACH

```
df[['QUANTITYORDERED','PRICEEACH']].kurtosis()
```

#KIEM TRA TINH CHUAN (NORMAL DISTRIBUTION) CUA  
#QUANTITYORDERED,PRICE,SALES

```
from scipy import stats
stats.probplot(df['QUANTITYORDERED'],plot=sns.mpl.pyplot)
plt.show()
```

THUC HIEN NORAMLIZE DU LIEU QUANTITYORDERED

```
from sklearn import preprocessing
min_max_scaler=preprocessing.MinMaxScaler()
df[['QUANTITYORDERED']]=
min_max_scaler.fit_transform(df[['QUANTITYORDERED']])
sns.displot(df, x="QUANTITYORDERED", kind="kde")
```



```
plt.show()
```

## THUC HIEN STANDARDIZATION DU LIEU QUANTITYORDERED

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df[['QUANTITYORDERED']] = scaler.fit_transform(df[['QUANTITYORDERED']])
sns.displot(df, x="QUANTITYORDERED", kind="kde")
plt.show()
```

```
#c2
from scipy import stats
df['QUANTITYORDERED'] = stats.zscore(df['QUANTITYORDERED'])
stats.zscore(df['QUANTITYORDERED'])
sns.displot(df, x="QUANTITYORDERED", kind="kde")
plt.show()
```

MA TRAN TUONG QUAN TUYEN TINH(PERSON) CUA CAC CAC  
QUANTITYORDERED, PRICEEACH, SALES

```
df[['QUANTITYORDERED', 'PRICEEACH', 'SALES']].corr()
#hoat df[['QUANTITYORDERED', 'PRICEEACH', 'SALES']].cov()
```

VE BIEU DO HEATMAP TUONG QUAN GIUA CAC CAP  
'QUANTITYORDERED', 'PRICEEACH', 'SALES'

```
sns.heatmap(df[['QUANTITYORDERED', 'PRICEEACH', 'SALES']].corr(), vmax=1.0, square=False).xaxis.tick_top()
```

TUONG QUAN CUA BIEN QUANTITYORDERED, PRICEEACH, SALES THEO  
NHOM DEALSIZE

```
df.groupby('DEALSIZE')[['QUANTITYORDERED', 'PRICEEACH', 'SALES']].corr()
```

VE BIEU DO CHO BIAET GIA TRI MIN ,MAX CUA SO LUONG SAN PHAM  
TRONG MOI DON HANG THEO QUY, NAM

```
gb = df.groupby(['QTR_ID', 'YEAR_ID'])['QUANTITYORDERED'].agg(['min', 'max'])
```

```
gb.plot(kind='bar',stacked=False)
plt.show()
```

VE BIEU DO CHO BIET TONG SO LUONG SAN PHAM THEO THANG

```
sns.lineplot(x="MONTH_ID",y="QUANTITYORDERED",hue='YEAR_ID',data=df,estimator=sum)
plt.show()
```

VE BIEU DO CHO BIET QUAN HE GIUA SALES (Oy) VA DON GIA(Ox) // DUNG BD SCATTER

```
sns.lmplot(data=df,x='PRICEEACH',y='SALES',fit_reg=False)
plt.show()
```

VE BIEU DO CHO BIET QUAN HE GIUA SALES (Oy) VA DON GIA(Ox) THEO DEALSIZE

```
sns.lmplot(data=df,x='PRICEEACH',y='SALES',hue='DEALSIZE',fit_reg=False)
plt.show()
```

VE BIEU DO CHO BIET QUAN HE GIUA SALES (Oy) VA DON GIA(Ox) THEO DEALSIZE QUA CAC NAM (YEAR\_ID) THEO COT VA THEO QUY QUA DONG

```
sns.lmplot(data=df,x='PRICEEACH',y='SALES',hue='DEALSIZE',col='YEAR_ID',row='QTR_ID',fit_reg=False)
plt.show()
```

#sắp xếp dữ liệu theo sales tăng dần

```
df.sort_values(by='SALES',ascending=True)
```

SAP XEP DU LIEU TANG DAN NHUNG KHI TRUNG LAP SE XET THEO MUC GIAM DAN O COT KHAC

```
df.sort_values(by=['QUANTITYORDERED','PRICEEACH'],ascending=[True,False])
```

LỌC DATA FARMER

#Hiển thị các dòng có Sales > 5000

```
df[df['SALES']>5000]
```

#Cách 2:

```
df.loc[df['SALES']>5000]
```

ĐIỀU KIỆN GỘP

#Hiển thị các dòng có Sales>5000 và QuantityOrdered>40

```
df[(df['SALES']>5000) & (df['QUANTITYORDERED']>40)]
```

TẠO CỘT MỚI VÀ GÁN THUỘC TÍNH MỚI TRÊN BẢNG

#Lọc ra các dòng có Priceeach<65, nếu có thay giá trị bằng Cheap, ngược lại thay bằng Expensive

```
df.loc[df['PRICEEACH']<65,'FLAG']='CHEAP'
```

```
df.loc[df['PRICEEACH']>=65,'FLAG']='EXPENSIVE'
```

```
df[['PRICEEACH','FLAG']]
```

#Viết hàm foo(x) nếu x<10 trả về Bad, nếu x>=10 và <50 trả về Good, ngược lại trả về Excellent

```
def foo(x):
```

```
    if x < 10 :
```

```
        return 'BAD'
```

```
    elif x >= 10 and x < 50:
```

```
        return 'GODD'
```

```
    else:
```

```
        return 'EXCELLENT'
```

#Áp dụng cho cột month

```
df['MONTH']= df[['QUANTITYORDERED']].applymap(foo)
```

#Hiển thị kết quả

```
df[['QUANTITYORDERED','MONTH']]
```

SO SÁNH CỘT

#Viết hàm so sánh giá trị nếu x<=y trả về giá trị YES, ngược lại là NO

```
def ftrust(x,y):
```

```
    if x<=y:
```

```
        return 'YES'
```

```
    else:
```

```
        return 'NO'
```

#Áp dụng hàm để gán giá trị trả về cho TRUST

```
df['TRUST']=list(map(ftrust, df['PRICEEACH'],df['MSRP']))
```

```
#Hiển thị kết quả sau khi gọi hàm
```

```
df[['PRICEEACH','MSRP','TRUST']]
```

```
#Thay đổi giá trị cho cột QTR_ID là Q1 nếu giá trị là 1, Q2 nếu giá trị là 2,...
```

```
dict_map= {1:'Q1', 2:'Q2', 3:'Q3', 4:'Q4'}
```

```
df['QTR_ID']= df['QTR_ID'].map(dict_map)
```

## XÁC ĐỊNH CÁC HÀM TỔNG HỢP TRÊN CÁC BIẾN ĐỊNH LƯỢNG VÀ ĐỊNH TÍNH

```
#Hiển thị giá trị tổng , giá trị nhỏ nhất cho cột QUANTITYORDERED, giá trị nhỏ nhất và lớn nhất và trung bình cho cột PRICEEACH, giá trị nhỏ nhất và trung bình cho cột SALES
```

```
df.aggregate({'QUANTITYORDERED':['sum','min'],'PRICEEACH':['min', 'max', 'mean'],'SALES':['min','mean']})
```

## GROUP BY: NHÓM

```
#Hiển thị giá trị tổng , giá trị nhỏ nhất cho cột QUANTITYORDERED, giá trị nhỏ nhất và lớn nhất và trung bình cho cột PRICEEACH, giá trị nhỏ nhất và trung bình cho cột SALES
```

```
df.aggregate({'QUANTITYORDERED':['sum','min'],'PRICEEACH':['min', 'max', 'mean'],'SALES':['min','mean']})
```

```
#Thống kê tổng, trung vị, độ lệch chuẩn của SALES theo từng nhóm DEALSIZE
```

```
df.groupby(['DEALSIZE'])['SALES'].agg(['sum','mean','std'])
```

```
#Thống kê tổng, trung vị, độ lệch chuẩn của SALES theo từng nhóm DEALSIZE, cùng DEALSIZE thì nhóm theo QTR_ID
```

```
df.groupby(['DEALSIZE','QTR_ID'])['SALES'].agg(['sum','mean','std'])
```

## XUẤT BẢNG TÍNH THEO THUỘC TÍNH CỦA BẢNG

```
pd.pivot_table(df,values='SALES',columns='YEAR_ID',aggfunc=['sum','mean'])
```

```
#Thống kê theo dạng bảng Pivot 2 chiều: Thống kê tổng và trung bình số lượng QUANTITYORDERED và SALES theo Year_ID
```

```
pd.pivot_table(df,values=['SALES','QUANTITYORDERED'],columns='YEAR_ID',aggfunc=['sum','mean'])
```

## LAB 2 TUYENSINHDAIHOC

```
"""
Phần 1: thao tác cơ bản
Bước 1: Xử lý cơ bản
1. Xác định số lượng yếu tố (biến số) tham gia vào yêu cầu
2. Thu thập dữ liệu (data collection)
3. Tổng quan dữ liệu VD: df.info()...
4. Xử lý cơ bản:
    - Loại bỏ dòng rỗng
    - Loại bỏ dòng trùng
    - Khảo sát dữ liệu thiếu
    - Xử lý dữ liệu thiếu
5. Kiểm tra lại dữ liệu
"""
```

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
#Đọc file
df = pd.read_csv('dulieuxettuyendaihoc.csv',header=0,delimiter=',',
encoding='utf-8')
#Đọc 5 dòng dữ liệu đầu tiên
df.head(5)
```

```
#Xem thông tin tổng quan
df.info()
# Tổng quan dữ liệu
```

Biến	Kiểu dữ liệu	Loại	Thang đo	
Malop	Chuỗi ký tự	Định tính		

```
#Lấy thông tin các cột
df = df[['T5','T6','GT','DT','KV','KT','NGONNGU','TOANLOGICPHANTICH',
'GIAIQUYETVANDE','NGAYTHI','DINHHUONGNGHENGHIEP']]
```

```
# Đổi tên cột
df.rename(columns={'TOANLOGICPHANTICH':'LOGIC','GIAIQUYETVANDE':'UNG
XU',
'DINHHUONGNGHENGHIEP':'HUONGNGHIEP'},inplace=True)
```

```
df.head(5)
```

```
#Xóa bỏ các dòng dữ liệu rỗng
df.dropna(how='all',inplace=True)
```

```
# dùng heatmap để trực quan dữ liệu bị thiếu
plt.figure(figsize=(10,6))
sns.heatmap(df.isna().transpose(),cmap='YlGnBu',
cbar_kws={'label':'Dữ liệu thiếu'})
plt.savefig('missingdata.png',dpi=100)
'''
```

Note: Với dữ liệu bị thiếu:

1. Cần xác định biến số nào bị thiếu
  2. Mức độ thiếu dữ liệu
  3. Có cần phải xử lý không
- '''

```
# Điền giá trị thiếu
df['DT'].fillna('KINH',inplace=True)
# Lưu ý: Với biến định tính ta có thể thay bằng giá trị yếu vị (mode)
# df['DT'].fillna(df['DT'].mode()[0],inplace=True)
```

```
# Điền thiếu giá trị phần NGONNGU bằng 0 (nếu có)
df['NGONNGU'].fillna(0,inplace=True)
# Điền thiếu giá trị phần LOGIC bằng trung bình (nếu có)
df['LOGIC'].fillna(df['LOGIC'].mean(), inplace=True)
```

```
# Điền thiếu giá trị phần UNGXU bằng trung vị (nếu có)
df['UNGXU'].fillna(df['UNGXU'].median(),inplace=True)
# Lưu ý: Với biến định lượng thì ta nên thay bằng trung vị
```

```
df.head(5)
```

```
'''
```

Phần 2: Kỹ thuật Feature Engineering (thường dùng cho Machine Learning)  
Nếu chỉ là xử lý phân tích dữ liệu thì ta gọi là New Attribute

Đây là kỹ thuật tạo thêm hoặc biến đổi số liệu sẵn có thành các biến số mới phù hợp với nghiệp vụ để phân tích

Tạo biến TBTOAN: trung bình toán lớp 12

```
'''
```

```
df['TBTOAN'] = (df['T5']+df['T6'])/2
df.head(5)
```

```
# Tạo biến XEPLOAI: đánh giá môn toán dựa trên df['TBTOAN']
```

```
df.loc[df['TBTOAN'] < 5.0, 'XEPLOAI'] = 'FAIL'
```

```
df.loc[(df['TBTOAN'] >= 5.0) & (df['TBTOAN'] < 7.0), 'XEPLOAI'] = 'FAIR'
```

```
df.loc[(df['TBTOAN'] >= 7.0) & (df['TBTOAN'] < 9.0), 'XEPLOAI'] = 'GOOD'
```

```
df.loc[(df['TBTOAN'] >= 9.0), 'XEPLOAI'] = 'EXCEL'
```

```
df.head(5)
```

```
#Xem thông tin 5 dòng đầu gồm các cột TBTOAN, XEPLOAI
```

```
df[['TBTOAN','XEPLOAI']].head(5)
```

```
'''
```

Tạo biến nhóm khối thi NHOMKT thỏa mãn

A1: G1

C: G3

D1: G3

A: G1

B: G2

```
'''
```

```
dict_map = {'A1':'G1','C':'G3','D1':'G3','A':'G1','B':'G2'}
df['NHOMKT'] = df['KT'].map(dict_map)
```

```
df[['KT','NHOMKT']].head(5)
```

```
"""BỮA SAU ngày 25/9
Tạo biến số điểm cộng: CONG
Nếu khối thi thuộc nhóm G1, G2 và TBTOAN >= 5.0 thì là 1.0
Ngược lại thì là 0.0
"""
def fplus(x,y):
    if(x == 'G1' or x == 'G2') and (y>=5.0):
        return 1.0
    else:
        return 0.0
df['CONG'] = list(map(fplus,df['NHOMKT'],df['TBTOAN']))
```

```
df[['TBTOAN','NHOMKT','CONG']].head(5)
```

```
#Phần 3: Trực quan hóa dữ liệu
"""
Để trực quan số liệu ta cần lưu ý: Mục đích, sự phối hợp giữa các loại biến để chọn lựa
biểu đồ phù hợp đi kèm số liệu
trực quan
Định tính: bar, pie
Định lượng: line , histogram, box-plot, scatter
"""
"""
Hãy trực quan số lượng học sinh theo giới tính
"""
# Biểu đồ bar
sns.countplot(x='GT', data=df)
plt.show()
```

```
# Lưu ý
```



```
# Các biến dùng để phân nhóm, gom nhóm thông thường là biến định tính và nằm ở các
thang đo mức 1,2,3,4
# tương ứng định danh, phân loại, thứ bậc và khoảng
```

```
# Dựa trên biểu đồ DT cho biết tạo sao ta không phân tích theo nhóm DT : vì đa số là dân
tộc kinh
# Tương tự cho các cột KV, DT, KT
#DT
sns.countplot(x='DT', data=df)
plt.show()
```

```
sns.countplot(x='KV', data=df)
plt.show()
```

```
sns.countplot(x='KT', data=df)
plt.show()
```

```
"""
Hãy so sánh số lượng học sinh đăng ký khối thi dựa trên nhóm giới tính
"""
sns.countplot(x='KT',hue='GT',data=df)
plt.show()
```

```
"""
Làm tương tự cho các nhóm biến định tính (KV,KT)
Hãy cho biết khối A có sinh viên khu vực nào đăng ký cao nhất
Trả lời: KV1
"""
sns.countplot(x='KV',hue='KT',data=df)
plt.show()
```

```
"""
Hãy so sánh điểm trung bình NGONNGU theo nhóm giới tính
"""
sns.barplot(x='GT',y='NGONNGU',data=df,errorbar=None)
plt.show()
```

```
# Hãy so sánh điểm LOGIC theo nhóm KT (nhóm khối thi)
sns.barplot(x='NHOMKT',y='LOGIC',data=df,errorbar=None)
plt.show()
```

```
"""
So sánh điểm trung bình của NGONNGU theo nhóm GT dựa trên KT
"""
sns.barplot(x='GT',y='NGONNGU',hue='KT',data=df,errorbar=None)
plt.show()
```

```
# So sánh sai số trên NGONNGU theo nhóm GT theo KT
# Sai số càng cao độ tin cậy càng thấp
sns.barplot(x='GT',y='NGONNGU',hue='KT',data=df)
plt.show()
```

```
"""
So sánh điểm cao nhất của NGONNGU theo nhóm GT theo KT
Lưu ý: không để estimator thì mặc định là mean
estimator: count, min max sum std, mean (default)
"""
sns.barplot(x='GT',y='NGONNGU',hue='KT',data=df,errorbar=None,estimator=max)
plt.show()
```

```
"""
So sánh điểm cao nhất của NGONNGU theo nhóm GT theo KT
Lưu ý: không để estimator thì mặc định là mean
estimator: count, min max sum std, mean (default)
"""
sns.barplot(x='GT',y='NGONNGU',hue='KT',data=df,errorbar=None,estimator=max)
plt.show()
```

```
"""
So sánh điểm cao nhất của NGONNGU theo nhóm GT theo KT
Lưu ý: không để estimator thì mặc định là mean
estimator: count, min max sum std, mean (default)
"""
```

```
'''
sns.barplot(x='GT',y='NGONNGU',hue='KT',data=df,errorbar=None,estimator=max)
plt.show()
```

```
'''
Khi biến định tính dùng làm nhóm tổng hợp có nhiều hơn 2 giá trị thì ta cần dùng hàm
tổng hợp thông qua thư viện numpy
'''
sns.barplot(x='KV',y='NGONNGU',hue='KT',data=df,errorbar=None,estimator=np.max)
plt.show()
```

```
'''
Lưu ý:
- Với biến định tính thì ta chỉ có 1 hàm tổng hợp là hàm COUNT, MODE
- Với định lượng thì ta có thể sử dụng các hàm tổng hợp như: COUNT, MAX, MIN,
MEAN, MEDIAN, MODE, SUM, STD
'''
```

```
'''
Biểu đồ PIE
Mục đích: Trực quan dữ liệu theo nhóm tỉ lệ phần trăm
'''
gb = df.groupby(['KT'])['KT'].agg(['count'])
# group by trên nhóm khối thi trên biến khối thi và dùng hàm count
```

```
'''
Biểu đồ PIE
Mục đích: Trực quan dữ liệu theo nhóm tỉ lệ phần trăm
'''
gb = df.groupby(['KT'])['KT'].agg(['count'])
# group by trên nhóm khối thi trên biến khối thi và dùng hàm count
labels = gb.index
data = list(gb['count'])
colors = sns.color_palette('pastel') # tạo bảng màu
plt.pie(data,labels=labels,colors=colors,autopct='%1.1f%%',shadow=True)
```

```
plt.show()
```

```
"""
Trực quan tỉ lệ % tổng điểm CONG trên từng nhóm khu vực
# coi khu vực nào dc cộng điểm nhiều nhất
"""
gb = df.groupby(['KV'])['CONG'].agg(['sum'])
labels = gb.index
data = list(gb['sum'])
colors = sns.color_palette('pastel') # tạo bảng màu
plt.pie(data,labels=labels,colors=colors,autopct='%1.1f%%',shadow=True)
plt.show()
```

```
"""
KHi trực quan dữ liệu ta cần lưu ý đến loại biến đang tham gia vào trực quan
Thông thường việc chọn lựa biểu đồ sẽ căn cứ dựa trên ý nghĩa nghiệp vụ và sự phối hợp
giữa các loại biến như:
- Định tính kết hợp định tính
- Định tính kết hợp định lượng
- Định lượng kết hợp định lượng
"""
```

```
"""
Biểu đồ line thường dùng để tổng hợp dữ liệu theo trục "Thời gian" hoặc "có thứ tự"
Ví dụ: tổng hợp trung bình điểm cộng theo các năm thi
"""
sns.lineplot(x='NGAYTHI',y='CONG',data=df)
plt.show()
```

```
# Trực quan dữ liệu tổng điểm CONG dựa theo năm bằng biểu đồ line
sns.lineplot(x='NGAYTHI',y='CONG',data=df,estimator=sum)
plt.show()
```

```
# tổng hợp tổng điểm cộng theo các năm thitrên từng nhóm giới tính bằng biểu đồ line
sns.lineplot(x='NGAYTHI',y='CONG',hue='GT',data=df,estimator=sum)
plt.show()
```

```
'''
Buổi 3
Bước 1: mô tả biến định lượng
'''

df['NGONNGU'].describe()
# Giải thích ý nghĩa các đại lượng
# Độ lệch chuẩn (std) bằng căn bậc 2 giá trị phương sai, độ lệch chuẩn và phương sai thể
hiện mức độ biến thiên, biến động
# sự đa dạng của tập dữ liệu số. Độ lệch chuẩn càng cao thì tập dữ liệu biến động mạnh
=> mức độ đa dạng nhiều và ngược lại thì tập dữ liệu sẽ ổn định hơn
# Tứ phân vị
# Q1 : 25% -> Có 25% dữ liệu nhỏ hơn giá trị Q1
# Q2: 50% (median trung vị) -> giá trị này cho biết có 50% sv nhỏ hơn Q2 và lớn hơn Q2
# Q3: 75% -> có 25% số lượng lớn hơn Q3
# Q1 - Q3 là khoảng IQR: khoảng mà các dữ liệu được diễn ra dc coi là thông thường
(50%)
```

```
df[['NGONNGU','LOGIC','UNGXU']].describe()
```

```
'''
CV = std/mean (Coefficient of variant)
So sánh mức độ ổn định của điểm số
'''

cvNN = df['NGONNGU'].std()/df['NGONNGU'].mean()
cvLogic = df['LOGIC'].std()/df['LOGIC'].mean()
cvUngXu = df['UNGXU'].std()/df['UNGXU'].mean()
print('cvNN: ', cvNN)
print('cvLogic: ', cvLogic)
print('cvUngXu: ', cvUngXu)
#
df[['NGONNGU','LOGIC','UNGXU']].std()/df[['NGONNGU','LOGIC','UNGXU']].mean()
()
```

```
df.groupby('GT')[['NGONNGU','LOGIC','UNGXU']].describe()
```

```
'''
Histogram cho biết xác suất xảy ra của biến cố trong khoảng giá trị dữ liệu nào nhiều nhất
'''
df['NGONNGU'].hist(bins=20)
plt.show()
```

```
# Hướng dẫn khi vẽ bins trong histogram
# Lưu ý: khi số lượng bins khác nhau sẽ dẫn đến hình dạng của histogram khác nhau
df['NGONNGU'].hist(bins=14)
plt.show()
```

```
'''
Nâng cao hơn histogram thì ra khám phá dạng phân phối xác suất
Làm mịn với phân phối xác suất
'''
sns.displot(df,x='NGONNGU',kind='kde')
plt.show()
```

```
sns.displot(data= df[['NGONNGU','LOGIC','UNGXU']],kind='kde')
plt.show()
# Hãy cho biết phân phối của biến số nào gần với phân phối chuẩn hơn => logic với ứng xử
```

```
sns.displot(df,x='NGONNGU',hue='GT',kind='kde')
plt.show()
# Câu hỏi: nhóm giới tính nào gần hơn: F gần hơn
```

```
'''
Đây là biểu đồ quan trọng trong việc phân tích dữ liệu định lượng
Biểu đồ này cung cấp các thông tin quan trọng như
1. Q1: Tứ phân vị 25%
2. Q2: Tứ phân vị 50% (median)
3. Q3: Tứ phân vị 75%
4. Độ lớn IQR = |Q3-Q1|
'''
```

5. Lower bound:  $Q1 - 1.5 * IQR$
6. Upper bound =  $Q3 + 1.5 * IQR$
7. Các ngoại biên, bất thường (outlier) cần xử lý trong dữ liệu

Outlier: là điểm dữ liệu khác biệt quá nhiều so với đa số

Hướng dẫn

- + Tính khoảng nghi ngờ chứa outliers
- + Tính khoảng chắc chắn chứa outliers

'''

```
sns.boxplot(data=df['LOGIC'],orient="h")
print('lower bound = ', df['LOGIC'].quantile(0.25) - 1.5*(df['LOGIC'].quantile(0.75) -
df['LOGIC'].quantile(0.25)))
print('upper bound = ', df['LOGIC'].quantile(0.75) + 1.5*(df['LOGIC'].quantile(0.75) -
df['LOGIC'].quantile(0.25)))
IQR = df['LOGIC'].quantile(0.75) - df['LOGIC'].quantile(0.25)
print('IQR',IQR)
print('ngoai bien dưới', 1.625 -1.5*IQR)
print('ngoai bien tren', 6.625 +1.5*IQR)
```

```
# Tính khoảng giá trị nghi ngờ bất thường
# Tính khoảng giá trị được cho là bất thường
# Tính xem có bao nhiêu sinh viên có điểm thi là bất thường
```

```
sns.boxplot(data=df[['NGONNGU','LOGIC',"UNGXU"]],orient='h')
```

```
# Hãy cho biết điểm số môn nào không xảy ra bất thường => NGONNGU
```

```
sns.boxplot(x='NGONNGU',y='KT',data=df,orient='h')
plt.show()
# Câu hỏi: khối thi nào có lower bound trùng với phân vị thứ 1 (Q1) => khối B
```

```
sns.boxplot(x='NGONNGU',y='KV',data=df,orient='h')
plt.show()
```

```
sns.boxplot(x='KT',y='NGONNGU',hue='GT',data=df)
```

```
sns.boxplot(x='KT',y='NGONNGU',hue='GT',data=df)
```

```
plt.show()
# câu hỏi: xác định các biểu đồ bất thường
# Khối A1 không đủ dữ liệu để vẽ
```

```
'''
Skewness = độ xiên, độ lớn (trị tuyệt đối) cho biết mức độ dữ liệu lệch nhiều hay ít so với
đường cong phân phối chuẩn
Cho biết xác suất được phân bố lệch về phía nào nhiều
Trị tuyệt đối giá trị càng lớn thì dữ liệu phân phối nghiêng càng nhiều (lệch)
Diễn giải cho skewness
skewness > 0 tức là mean > median: ta gọi là positive skewness
hay lệch phải, tức là giá trị ngoại biên outliers nhận giá trị lớn sẽ đẩy giá trị trung bình về
phía cuối
skewness < 0 tức là mean < median: ta gọi là negative skewness hay lệch trái, tức là giá trị
outliers nhận giá trị nhỏ sẽ đẩy giá trị trung
bình về phía đầu
skewness = 0 tức là mean = median = mode: phân phối không lệch còn được gọi là phân
phối đối xứng
'''

df['NGONNGU'].skew()
'''
Note: Khi ptich dữ liệu với các phương pháp có liên quan phân phối chuẩn thì cần kiểm
tra skewness
nếu dl quá lệch so với phân phối chuẩn thì ta cần điều chỉnh bằng các hàm transform cho
bớt lệch
đặc biệt là phân tích hồi quy
'''
```

```
df[['NGONNGU','LOGIC','UNGXU']].skew()
```

```
'''
Kurtosis: Độ nhọn, trị tuyệt đối cho biết mức độ nhọn của phân phối
Độ lớn của kurtosis càng gần 3 thì fit
Dưới 3 thì fat
Trên 3 thì thin
'''
```



Thông thường để đánh giá hình dáng độ nhọn ta dùng đại lượng excess kurtosis (theo Pearson) - 3

+ Nếu  $\text{excess} > 0 \rightarrow$  thin

+ Nếu  $\text{excess} = 0 \rightarrow$  fit

+ Nếu  $\text{excess} < 0 \rightarrow$  fat

Trong pandas sử dụng Fisher's Kurtosis tức là đã chuẩn hóa giá trị về excess kurtosis theo  $\text{mean} = 0$

+ Trị tuyệt đối excess kurtosis càng cao thì mức độ thin, fat càng lớn

Lưu ý : với phân phối chuẩn thì  $\text{excess kurtosis} = 0$ ,  $\text{skewness} = 0$

'''

```
df[['NGONNGU']].kurtosis()
```

# Câu hỏi: biến ngôn ngữ có độ nhọn như thế nào

# Trả lời là: fat

```
df[['NGONNGU','LOGIC','UNGXU']].kurtosis()
```

```
sns.displot(data=df[['LOGIC','UNGXU']],kind='kde')
```

```
plt.show()
```

# Nhìn biểu đồ cho biết ý nghĩa kurtosis cho LOGIC UNGXU

'''

Kiểm định phân phối chuẩn

'''

```
from scipy import stats
```

```
stats.probplot(df['NGONNGU'],plot=sns.mpl.pyplot)
```

```
plt.show()
```

# không có phân phối chuẩn

'''

Phân tích sự tác động (ảnh hưởng) qua lại giữa 2 biến định lượng

'''

'''

Buổi 3:

Hiệp phương sai: co-variance

Giá trị co-variance  $> 0$  thì 2 biến có tương quan thuận (đồng biến)

Giá trị co-variance  $< 0$  thì 2 biến có tương quan nghịch (nghịch biến)

Độ lớn (trị tuyệt đối của giá trị) càng lớn thì mức độ quan hệ (tương quan) càng chặt chẽ

Ma trận hiệp phương sai: co-variance matrix

'''

```
df[['T5','T6']].cov()
```

# So sánh mức độ tương quan giữa T5 so với LOGIC và T6 so với LOGIC

```
df[['T5','T6','LOGIC']].cov()
```

'''

Với phương pháp so sánh tương quan bằng co-variance thì ta không đo lường được cường độ

tương quan giữa 2 biến định lượng.

Pearson Correlation: tương quan tuyến tính

r nằm trong khoảng [-1,1]

$r = 0 \Rightarrow$  Không tương quan

$r < 0$ : Tương quan nghịch

$r > 0$ : tương quan thuận

|r| (độ lớn) càng gần 1 thì tương quan càng cao

|r| < 0.5 thì tương quan thấp

[0.5,0.65]: Khá

[0.65,0.75]: Tốt

[0.75,0.9]: Rất tốt

> 0.9: hoàn hảo

Ma trận tương quan: correlation matrix

Lưu ý: được sử dụng khảo sát tương quan tuyến tính nhằm phân tích mối quan hệ tuyến tính giữa 2 biến định lượng

'''

```
df[['T5','T6']].corr()
```

'''

Trực quan hóa tương quan tuyến tính giữa 2 biến định lượng

Khám phá tương quan tuyến tính của 2 biến định lượng

thông qua biểu đồ phân tán (scatter)

'''

```
sns.lmplot(data=df, x='T5',y='T6', fit_reg=True)
```

```
plt.show()
```

```
# Sinh viên tự khám phá độ tương quan giữa biến T6 và UNGXU
df[['T6','UNGXU']].corr()
sns.lmplot(data=df, x='T6',y='UNGXU', fit_reg=True)
plt.show()
```

```
df[['T6','UNGXU']].corr()
```

```
df[['T6','UNGXU','NGONNGU','LOGIC','UNGXU']].corr()
```

```
sns.heatmap(df[['T5','T6','UNGXU','NGONNGU','LOGIC','UNGXU']].corr(),vmax=1.0,
square=False).xaxis.tick_top()
```

```
# Biểu đồ tổng hợp khám phá tổng hợp nhiều biến định lượng
sns.pairplot(df[['T5','T6','NGONNGU','LOGIC','UNGXU']], diag_kind='kde',kind='reg')
plt.show()
```

```
# Trục quan tương quan tuyến tính theo nhóm (định tính) giữa 2 biến định lượng
sns.lmplot(data=df,x='T5',y='T6',hue='GT',fit_reg=True)
plt.show()
```

### BÀI 3

Sinh viên làm các câu hỏi sau theo dữ liệu của mỗi sinh viên

1. Đọc File với các file trong thư mục dữ liệu. Ứng với mỗi file sinh viên thực hiện các câu sau. Hiển thị toàn bộ dữ liệu của file dữ liệu đã đọc
2. Đọc file với chỉ định không có header, dòng header của chúng ta đã biến thành 1 bản ghi dữ liệu. Hiển thị toàn bộ dữ liệu của file dữ liệu đã đọc
3. Đọc File với các tùy chọn mặc định. Hiển thị 5 dòng dữ liệu đầu tiên
4. Đọc File với các tùy chọn mặc định. Hiển thị 5 dòng dữ liệu cuối cùng
5. Chuyển kiểu dữ liệu cho 1 cột nào đó
6. Xem chiều dài của df, tương đương shape[0]  
print('Len:', len(df))
7. Xem thông tin dataframe vừa đọc được  
df.info()

8. Xem kích thước của dataframe  
`print('Shape:', df.shape)`
9. Hiển thị dữ liệu của cột thứ 3  
`df['tên cột']`
10. Hiển thị dữ liệu của cột 1,2,3,5,6  
`df[['Tên cột 1', 'Tên cột 2', 'Tên cột 3']]`
11. Hiển thị 5 dòng dữ liệu đầu tiên gồm các cột 1,2,3,5,6  
`df[['Tên cột 1', 'Tên cột 2', 'Tên cột 3']].head[5]`
12. Hiển thị 5 dòng dữ liệu đầu tiên theo chỉ số  
`df[0:5]`
13. Hiển thị 5 dòng dữ liệu đầu tiên theo chỉ số gồm các cột 1,2,3,5,6  
`df[['Tên cột 1', 'Tên cột 2', 'Tên cột 3']][:5]`
14. Loại bỏ các dòng trùng nhau  
`df.drop_duplicates(inplace=True)`
15. Loại bỏ các dòng trống có dữ liệu của 1 cột là trống  
`df['ADDRESSLINE2'].isna().sum()`
16. Điền dữ liệu cho 1 cột thiếu dữ liệu thành có giá trị là không biết (“unknown”)  
`df["ADDRESSLINE2"].fillna('Unknown',inplace=True)`  
`df['ADDRESSLINE2'].isna().sum()`
17. Lấy dữ liệu của 1 cột theo dạng chuỗi  
`df["QUANTITYORDERED"]`
18. Lấy dữ liệu của 1 cột về một mảng  
`df["QUANTITYORDERED"].values`
19. Lấy dữ liệu của 1 cột về một mảng  
`df["QUANTITYORDERED"].values`
20. Lấy dữ liệu từ dòng số 4 đến dòng 9  
`df[4:10]`
21. Đọc dữ liệu từ dòng 4 đến dòng 9  
`df.loc[4:10]`  
`df.iloc[4:10]`
22. Lấy thông tin tại dòng có chỉ số là 2  
`df.loc[2]`
23. Lấy thông tin từ dòng 4 đến dòng 10 của một số cột  
`df.loc[4:10,['QUANTITYORDERED','SALES']]`
24. Lấy thông tin dòng 2 đến dòng 9, từ cột 4 đến cột 7  
`df.iloc[2:9,4:7]`

25. Lấy dữ liệu tại chỉ số (index) là 2

```
df.iloc[2]
```

26. Lấy dữ liệu từ dòng đầu tiên đến dòng 9 dùng iloc

```
df.iloc[:10]
```

27. Lấy dữ liệu từ dòng đầu tiên đến dòng 9 gồm các cột 4 đến cột 7 dùng iloc

```
df.iloc[2:9,4:7]
```

28. Sắp xếp dữ liệu theo 1 cột tăng dần

```
df.sort_values(by='SALES',ascending=True)
```

29. Sắp xếp dữ liệu theo nhiều tiêu chí

```
df.sort_values(by=['QUANTITYORDERED','PRICEEACH'],ascending=[True,False])
```

30. Lọc dữ liệu theo 1 điều kiện

```
df[df['SALES']>5000]
```

```
df.loc[df['SALES']>5000]
```

31. Lọc dữ liệu theo nhiều điều kiện

```
df[(df['SALES']>5000) & (df['QUANTITYORDERED']>40)]
```

32. Lọc giá trị và gán điều kiện dùng loc

```
df.loc[df['PRICEEACH']>=65,'FLAG']='EXPENSIVE'
```

```
df.loc[df['PRICEEACH']<65,'FLAG']='CHEAP'
```

```
#df['FLAG']
```

```
df[['PRICEEACH','FLAG']]#[:50]
```

33. Viết hàm trả về giá trị có nhiều điều kiện và áp dụng hàm gán trị trả về cho 1 cột

```
def foo(x):
```

```
    if x<10:
```

```
        return 'BAD'
```

```
    elif x>=10 and x<50:
```

```
        return 'GOOD'
```

```
    else:
```

```
        return 'EXCELLENT'
```

```
df['WORTH']=df[['QUANTITYORDERED']].applymap(foo)
```

```
df[['QUANTITYORDERED','WORTH']]
```

34. Ánh xạ giá trị tới 1 cột

```
dict_map = {1:'Qui_1',2:'Qui_2',3:'Qui_3',4:'Qui_4'}
```

```
df['QTR_ID']=df['QTR_ID'].map(dict_map)
```

```
df['QTR_ID']
```

35. Lấy những dòng dữ liệu bằng 1 điều kiện nào đó  
`df[['QUANTITYORDERED','PRICEEACH']].loc[df['YEAR_ID']==2003]`
36. Hiện thị các bản ghi có cột kiểu số hơn 25  
`df[df['age']<25]`  
`Tuoitre =df[df['age']<25]`  
`Tuoitre[:5]`
37. Sinh viên tự nghĩ ra các câu hỏi đọc dữ liệu theo các điều kiện và thực hiện lại các câu hỏi đó. Yêu cầu sinh viên ghi lại các lệnh và thực hiện các câu hỏi đó. Sau đó chụp kết quả dán vào file word
38. Thực hiện 1 ví dụ để lấy giá trị của một cột trả về dưới dạng numpy array trong thư viện pandas python.
39. Sử dụng thư viện random để sinh ngẫu nhiên một list năm sinh và thêm vào dataframe.
40. Thêm 1 cột vào file dữ liệu
41. Thêm 1 cột vào dữ liệu theo tiêu chí nếu điều kiện thoả thì giá trị mặc định là True, ngược lại là False.
42. Tạo 1 cột mới có giá trị rỗng
- 43. Thêm 1 bản ghi trong dataframe**
44. Sửa giá trị của cột
45. Xóa cột trong dataframe
46. Xóa bản ghi theo chỉ số
47. Sử dụng hàm describe() để thống kê dữ liệu
48. Xem thống kê trên từng cột
49. Vẽ đồ thị xem phân bố giá trị của 1 trường trong dataframe
- 50. Tạo mới dataframe từ các python list**
51. Sắp xếp dataframe
52. Nối 2 dataframe (Lỗi)
53. Xáo trộn các bản ghi trong dataframe
54. Lưu dataframe về file csv
55. Tối ưu bộ nhớ khi dùng pandas
56. Tạo 1 file chương trình hiện các menu gồm các mục trên và mục cuối là thoát. Người dùng chọn chức năng nào trong menu thì chương trình sẽ thực hiện chức năng tương ứng.

### **3) MINI PROJECT XỬ LÝ DỮ LIỆU BẢNG ĐIỂM BẰNG PANDAS**

**Yêu cầu dự án:** Cho trước các bộ dữ liệu sau

[dataset.zip](#)

## Giải thích về bộ dữ liệu

- roster.csv: Chứa danh sách sinh viên, gồm các thông tin:
  - ID: ID của sinh viên
  - Name: Họ tên đầy đủ của sinh viên
  - Net ID: Net ID của sinh viên
  - Email Address: Địa chỉ mail của sinh viên
  - Section: Ca học của sinh viên
- hw\_exam\_grades.csv: Chứa thông tin về điểm của các bài tập về nhà và bài kiểm tra
  - First Name: Tên
  - Last Name: Họ
  - SID: SID của sinh viên
  - Homework x: Điểm của sinh viên trong bài tập thứ x
  - Homework x - Max Points: Điểm tối đa của bài tập thứ x
  - Homework x - Submission Time: Thời gian nộp bài tập của sinh viên
  - Exam x: Điểm của bài kiểm tra thứ x
  - Exam x - Max Points: Điểm tối đa của bài kiểm tra thứ x
  - Exam x - Submission Time: Thời gian nộp bài kiểm tra của sinh viên
- quiz\_x\_grades.csv: Chứa thông tin điểm trong quiz thứ x:
  - Last Name: Họ
  - First Name: Tên
  - Email: Địa chỉ mail của sinh viên
  - Grade: Điểm quiz của sinh viên

## Yêu cầu

Hãy xử lý dữ liệu từ dữ liệu thô cho trước để xây dựng bảng điểm cho mỗi ca.

Kết quả xây dựng được như sau:

[result.zip](#)

## Các nhiệm vụ

### Nhiệm vụ 1

Hãy nhận xét về bộ dữ liệu.

### Nhiệm vụ 2

1. Load dữ liệu roster.csv lên:
  - NetID và Email Address cần được chuyển thành định dạng viết thường.
  - Chỉ giữ lại cột Section, Email Address, NetID
  - Cột index là NetID
  - Gợi ý

Tham khảo tại liệu [Read CSV](#)

2. Load dữ liệu hw\_exam\_grades.csv lên:

- SID cần được chuyển thành định dạng viết thường.
- Bỏ đi các cột Submission
- Cột index là SID
- Gợi ý

Tham khảo tại liệu [Read CSV](#)

3. Load dữ liệu quiz\_grades:

- Gộp các bảng trong quiz\_x\_grades.csv lại thành một DataFrame
- Email cần được chuyển thành định dạng viết thường.
- Giữ lại cột Email và Grade
- Cột index là Email
- Đổi tên cột Grade thành Quiz 1, Quiz 2, ...
- Gợi ý

Tham khảo tại liệu [Rename column of DataFrame](#) và [Concat DataFrame](#)

### Nhiệm vụ 3

- Trộn 3 DataFrame có được từ nhiệm vụ 2 thành 1 DataFrame duy nhất
- Dữ liệu NaN được chuyển thành 0
- Gợi ý

Tham khảo tại liệu [DataFrame Merging](#)

### Nhiệm vụ 4

1. Tính điểm bài kiểm tra: Lấy điểm kiểm tra chia cho điểm tối đa

- Gợi ý

Tham khảo tại liệu [Calculating](#)

2. Tính điểm homework:

Có 2 cách tính điểm:

- Theo tổng điểm: Tính tổng điểm thô và điểm tối đa một cách độc lập, sau đó lấy tỷ lệ.
- Theo điểm trung bình: Chia từng điểm thô cho số điểm tối đa tương ứng, sau đó lấy tổng của các tỷ lệ này và chia tổng cho số lượng bài tập.

Điểm sẽ được tính theo 2 cách, điểm sinh viên nhận được là điểm lớn nhất trong 2 điểm này.

- Gợi ý

Tham khảo tại liệu [Filter with Regex](#) và [Max function](#)

3. Tính điểm quiz theo cách tương tự homework

Điểm tối đa của mỗi quiz như sau:



- Quiz 1: 11
  - Quiz 2: 15
  - Quiz 3: 17
  - Quiz 4: 14
  - Quiz 5: 12
4. Tính điểm trung bình (final score)

Trọng số các cột điểm như sau:

- Exam 1: 0.05
- Exam 2: 0.1
- Exam 3: 0.15
- Quiz Score: 0.3
- Homework Score: 0.4

Điểm được làm tròn lên (ceiling)


5. Tính điểm chữ

Điểm chữ được tính như sau:

- Từ 90 điểm trở lên: A
- Từ 80 đến cận 90: B
- Từ 70 đến cận 80: C
- Từ 60 đến cận 70: D
- Dưới 60: F

#### 4) KẾT NỐI VỚI SQL SERVER

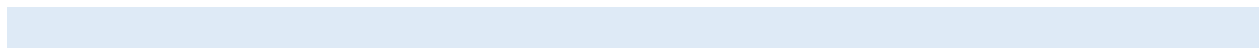
- 1) Sinh viên tạo quyền truy cập MS-SQL Server
  - a. Tài khoản: sa
  - b. Mật khẩu: 123456
- 2) Tạo CSDL mang tên dbPythonTestSQL
- 3) Tạo bảng dữ liệu tblUsers như thiết kế

MRNAM\SQLEXPRESS....st - dbo.tblUsers			
	Column Name	Data Type	Allow Nulls
	uid	varchar(5)	<input type="checkbox"/>
	uname	nvarchar(25)	<input checked="" type="checkbox"/>
	ubirthdate	date	<input checked="" type="checkbox"/>
	usalary	float	<input checked="" type="checkbox"/>
	urank	int	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

- 4) Nạp các dữ liệu mẫu

	uid	uname	ubirthdate	usalary	urank
	u0001	Henry	2001-07-31	125.5	1
	u0002	Peter	1990-08-25	500	1
	u0003	Owen	1995-02-28	275.5	2
	u0004	Jackson	2002-04-25	400	4
	u0005	Elite	2000-05-20	600	3

- 5) Đọc bài này: <https://www.educative.io/answers/how-to-connect-the-sql-server-withpython>
- 6) Tiến hành thực hiện các thao tác sau từ Python
  - a. Lấy thông tin nhân viên có tuổi nhỏ hơn 35
  - b. Lấy thông tin uname và tuổi (age) của tất cả các nhân viên
  - c. Thêm nhân viên mới (u0006,"David","07-22-2000",450,5)
  - d. Cập nhật lương của nhân viên u0002 thành 650
  - e. Xóa nhân viên u0006



## BÀI THỰC HÀNH TUẦN 5: THỐNG KÊ SUY DIỄN

### 1) CÁC BÀI TOÁN VỀ KIỂM ĐỊNH

'''

Phân tích suy diễn (inferential statistics)

Lý do tại sao cần suy diễn: Kết luận dựa trên dữ liệu sample (mẫu) nhưng kết luận thì được hiểu là áp dụng cho tổng thể

Câu hỏi đặt ra: Kết luận trên mẫu đó có phù hợp với tổng thể hay không

Khi phân tích suy diễn cần lưu ý:

1. Xác định giả thuyết  $H_0$
2. Các giả định hay các điều kiện về dữ liệu và môi trường để áp dụng kiểm điểm
  - Lưu ý về giả định phân phối của biến số: normal, student, poisson, chi-square...
3. Thiết lập mức tin cậy và sai lầm (alpha): 90% - 10%, 95% - 5% và 99% - 1%
4. Quy tắc suy diễn:
  - 4.1 Nếu  $p\text{-value} < \alpha \Rightarrow \text{reject } H_0$
  - 4.2 Nếu  $p\text{-value} > \alpha \Rightarrow \text{accept } H_0$
5. Kết luận của suy diễn chỉ cho chúng ta biết là có đủ dữ kiện để kết luận cho tổng thể hay không (còn gọi là ý nghĩa thống kê)

'''

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
#Đọc File dulieutuyensinh
df = pd.read_csv('./EDA/dulieuxettuyendaihoc.csv',header=0,delimiter=',',encoding='utf-8')
```

```
#Đổi tên cột
df = df
[['T5','T6','GT','DT','KV','KT','NGONNGU','TOANLOGICPHANTICH','GIAIQUYETV
ANDE','NGAYTHI','DINH HUONGNGHENGHIEP']]
df.rename(columns={'TOANLOGICPHANTICH':'LOGIC',
```

```
'GIAIQUYETVANDE':'UNGXU',
'DINH HUONG NG HENG HIEP':'HUONG NG HIEP'},inplace=True)
```

```
'''
```

### ONE SAMPLE T-TEST

Mục đích: Kiểm định trung bình của 1 biến số (định lượng) có bằng một giá trị

$n < 30$

```
'''
```

```
# H0: mean = X
```

```
import scipy.stats as stats
```

### Thư viện SciPy trong Python là gì?

Là phần mềm nguồn mở miễn phí của Python cho toán học, khoa học và kỹ thuật. Thư viện SciPy được xây dựng dựa trên thư viện NumPy, cung cấp thao tác mảng N chiều thuận tiện và nhanh chóng. SciPy gồm các gói con (submodule) cho đại số tuyến tính, tối ưu hóa, tích hợp và thống kê. NumPy và SciPy rất dễ sử dụng, mạnh mẽ và được nhiều nhà khoa học và kỹ sư hàng đầu thế giới lựa chọn.

SciPy là thư viện cơ sở. Nó được xây dựng trên phần mở rộng NumPy. Không cần nhập NumPy nếu bạn đã nhập SciPy. SciPy tương thích với đối tượng mảng N chiều của NumPy. SciPy bao gồm mã cho hoạt động của các hàm NumPy. SciPy và NumPy cùng là sự lựa chọn tốt nhất cho các hoạt động khoa học.

### Điều kiện tiên quyết để sử dụng Scipy trong Python

Hai điều kiện tiên quyết cần thiết nhất cho SciPy là Python và Toán học. Vì SciPy được xây dựng trên ngôn ngữ python nên việc học cơ bản về Python là một yêu cầu bắt buộc. Ngoài ra, vì SciPy dùng để thực hiện các phép tính toán học, kiến thức về toán học là cần thiết để xác minh và hiểu đầu ra của SciPy.

### Sub-package của SciPy

SciPy bao gồm nhiều gói khác nhau để thực hiện một loạt các chức năng. SciPy có các gói cho các yêu cầu cụ thể. Nó bao gồm hơn 15 gói để thực hiện các hoạt động.

SciPy có một gói dành riêng cho các hàm thống kê, đại số tuyến tính, phân cụm dữ liệu, xử lý hình ảnh và tín hiệu, cho ma trận, để tích hợp và phân biệt, v.v. Dưới đây là một số ví dụ:

Gói con	Miêu tả
Cluster	Thuật toán phân cụm (Clustering Algorithms)
Constants	Các hằng số toán học và vật lý

Fftpack	Hàm biến đổi Fourier nhanh (Fast Fourier Transform)
Integrate	Giải phương trình vi phân và tích phân
Interpolate	Nội suy và làm mịn spline
Io	Đầu vào và đầu ra
Linalg	Đại số tuyến tính
Ndimimage	Xử lý ảnh N chiều
Odr	Hồi quy khoảng cách trực giao
Optimize	Tối ưu hóa và chương trình root-finding
Signal	Xử lý tín hiệu
Sparse	Ma trận sparse và các đoạn chương trình liên quan
Spatial	Các cấu trúc dữ liệu không gian và thuật toán
Special	Các hàm toán học đặc biệt
Stats	Các hàm và phân phối thống kê

### Cài đặt thư viện

- Thông qua các phân phối: Anaconda, Miniconda, WinPython, Pyzo.
- Sử dụng pip: `python-m pip install --user scipy`  
 Bạn có thể cài đặt cùng lúc nhiều thư viện với pip: `python-m pip install --user numpy scipy matplotlib ipython jupyter pandas sympy nos`
- `pip install scipy`

Lưu ý: Numpy phải được cài đặt trước. Bạn cũng nên cài đặt [Matplotlib](#) khi sử dụng Scipy.

Cài đặt thư viện: `from scipy import special`

```
# câu 1: Hãy kiểm tra xem LOGIC thí sinh khối C có bằng 4.0
dfKhoiC = df.loc[df['KT'] == 'C']
dfKhoiC['LOGIC']
```

```

1      4.00
2      6.75
6      6.75
22     3.50
23     5.25
24     2.25
25     2.00
26     4.50
27     5.00
95     1.50
96     3.75
97     8.00
98     3.50
99     2.50
Name: LOGIC, dtype: float64
# Với one sample T-test thì giả thiết G0:  $\mu = 4.0$ 
# Mặc định mức tin cậy là 95% và mức sai lầm là 5%
stats.ttest_1samp(dfKhoiC['LOGIC'], popmean=4.0)
Kết quả:
TtestResult(statistic=0.44599723713991907, pvalue=0.6629370899710998, df=13)

```

```
# Kết luận:
# Do alpha = 0.05 và p-value = 0.66...
# Suy ra đủ dữ liệu để nói rằng trung bình môn thi LOGIC bằng 4.0
# hay nói cách khác là chấp nhận H0 ở mức sai lầm 5%
```

```
# Điểm trung bình có môn thi UNGXU của khối thi C có bằng 5.0 hay không
stats.ttest_1samp(dfKhoiC['UNGXU'],popmean=5)
Kết quả: TtestResult(statistic=-0.7645471693105148, pvalue=0.4581953822944209,
df=13)
# Kết luận :
# do anpha = 0.05 và p-value =0.4581953822944209
# Suy ra đủ dữ liệu để nói rằng trung bình môn thi UNGXU = 5.0
# hay nói cách khác là chấp nhận G0 ở mức sai lầmm 5%
```

```
# Tự nghiên cứu cách thiết lập mức tin cậy hoặc sai lầm trong đoạn code trên
```

```
# Bài tập tương tự: Hãy kiểm tra xem có phải điểm NGONNGU của thí sinh thi khối C
# là 5.5 hay không với mức sai lầm là 10%
```

```
# TWO SAMPLE T-TEST
# Mục đích: Kiểm tra xem trung bình của 2 biến số (định lượng) có bằng nhau không
#The sample size < 30
#H0 :delta_mean = 0
```

```
# Câu 2: Kiểm tra xem trung bình điểm thi LOGIC và trung bình điểm thi UNGXU của
thí sinh thi khối C có bằng nhau không
# H0: mean_LOGIC - mean_UNGXU = 0
dfKhoiC = df.loc[df['KT'] == 'C']
stats.ttest_ind(dfKhoiC['LOGIC'],dfKhoiC['UNGXU'],equal_var=True)
```

**Kết quả:**

```
TtestResult(statistic=-1.0329196014245297, pvalue=0.3111543826061086, df=26.0)
```



```
# Kết luận: Do alpha = 0.05 và p-value = 0.3111543826061086
# Suy ra đủ dữ liệu để nói rằng trung bình LOGIC bằng trung bình UNGXU hay nói cách khác là chấp nhận H0 ở mức sai lầm 5%
```

```
# Kiểm tra xem trung bình 3 UNGXU có bằng trung bình NGONNGU cho thí sinh thi khối C hay không với mức tin cậy 99%
```

```
# ONE SAMPLE Z-TEST
```

```
# Mục đích: kiểm định trung bình của một biến số (định lượng)
```

```
# H0L mean = x
```

```
# n > 30
```

```
from statsmodels.stats.weightstats import ztest as ztest
```

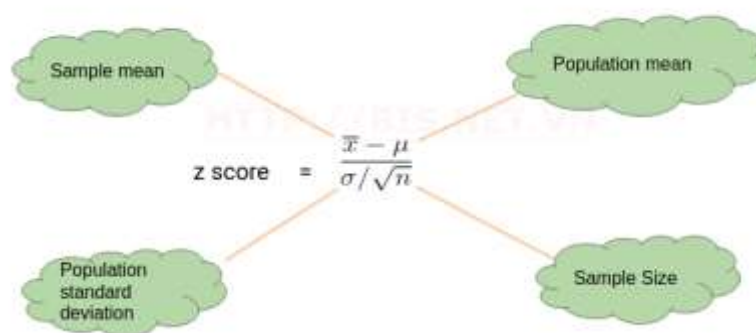
## KIỂM ĐỊNH Z (Z TEST)

Kiểm định Z khi:

- Biết phương sai của tổng thể (population variance), hoặc
- Không biết phương sai tổng thể nhưng kích thước mẫu (sample size)  $\geq 30$

### One-Sample Z test

One-Sample Z test được thực hiện khi muốn so sánh trung bình mẫu (**sample mean**) với trung bình tổng thể (**population mean**).



Ví dụ:

Để kiểm tra liệu điểm thi trung bình của thí sinh nữ có  $> 600$  hay không. Chúng ta có thông tin về độ lệch chuẩn điểm thi của nữ là 100. Chúng ta thu thập dữ liệu điểm thi của 20 sinh viên và thiết lập mức ý nghĩa  $\alpha$  là 5%.



Score
650
730
510
670
480
800
690
530
590
620
710
670
640
780
650
490
800
600
510
700

Trong ví dụ này:

- Mean Score của nữ: 641
- Kích thước mẫu: 20
- Trung bình tổng thể (population mean): 600
- Độ lệch chuẩn của tổng thể (Standard Deviation for Population): 100

$$\begin{aligned}
 \text{z score} &= \frac{\bar{x} - \mu}{\sigma / \sqrt{n}} \\
 &= \frac{641 - 600}{100 / \sqrt{20}} \\
 &= 1.8336
 \end{aligned}$$

$$\text{p value} = .033357$$

$$\text{Critical Value} = 1.645$$

$$\text{Z score} > \text{Critical Value}$$

$$\text{P value} < 0.05$$



$$H_0: \mu \leq 600$$

$$H_1: \mu > 600$$



Vì  $p\text{-value} < 0.05$ , nên ta bác bỏ  $H_0$  (chấp nhận  $H_1$ ) và kết luận rằng điểm trung bình của sinh viên nữ  $> 600$ .

## Two Sample Z Test

Two Sample Z Test được thực hiện khi muốn so sánh giá trị trung bình của 2 mẫu

$$z \text{ score} = \frac{(\bar{x}_1 - \bar{x}_2) - (\mu_1 - \mu_2)}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}$$

Diagram illustrating the components of the Two Sample Z Test formula:

- Difference bw Sample mean:**  $\bar{x}_1 - \bar{x}_2$
- Difference bw population mean:**  $\mu_1 - \mu_2$
- Population standard deviation:**  $\sigma_1, \sigma_2$
- Sample Size:**  $n_1, n_2$

Ví dụ:

Chúng ta muốn biết rằng liệu điểm trung bình của nữ lớn hơn 10 điểm so với điểm trung bình của nam hay không? Chúng ta biết độ lệch chuẩn điểm của nữ là 100 và của nam là 90. Thu thập dữ liệu về điểm của 20 nữ và nam như sau, với mức ý nghĩa  $\alpha$  là 0.05.

Score	Score
650	630
730	720
510	462
670	631
480	440
800	783
690	673
530	519
590	543
620	579
710	677
670	649
640	632
780	768
650	615
490	463
800	781
600	563
510	488
700	650

Trong ví dụ này:

- Điểm trung bình của nữ (Sample Mean): 641
- Điểm trung bình của nữ (Sample Mean): 613.3
- Độ lệch chuẩn tổng thể của nữ: 100
- Độ lệch chuẩn tổng thể của Nam: 90

- Kích thước mẫu (Sample Size): 20 (cho cả Nam và Nữ)
- Sự khác biệt giữa 2 trung bình của tổng thể: 10

$$\begin{aligned}
 z \text{ score} &= \frac{(\bar{x}_1 - \bar{x}_2) - (\mu_1 - \mu_2)}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}} \\
 &= \frac{(641 - 613.3) - (10)}{\sqrt{\frac{100^2}{20} + \frac{90^2}{20}}} \\
 &= 0.588 \\
 P \text{ value} &= 0.278 \\
 \text{Critical Value} &= 1.645 \\
 Z \text{ score} &< \text{Critical Value} \\
 P \text{ value} &> 0.05
 \end{aligned}$$

$H_0: \mu_1 - \mu_2 \leq 10$   
 $H_1: \mu_1 - \mu_2 > 10$

$p\text{-value} > 0.05$  nên ta không có cơ sở để bác bỏ  $H_0$  (Null Hypothesis). Có nghĩa là chúng ta không có đủ bằng chứng để kết luận rằng điểm trung bình của nữ cao hơn điểm trung bình của Nam 10 điểm.

### Kiểm định t (t-Test)

t-test được sử dụng khi:

- Không biết phương sai tổng thể (population variance)
- Kích thước mẫu nhỏ ( $n < 30$ )

#### One-Sample t-test


One-Sample t-test được sử dụng để so sánh trung bình mẫu (sample mean) với trung bình tổng thể (population mean). Khác với Z Test, t-test không cần biết phương sai của tổng thể. Chúng ta sử dụng độ lệch chuẩn của mẫu (sample standard deviation) thay cho độ lệch chuẩn của tổng thể (population standard deviation).

$$t = \frac{\bar{x} - \mu}{s / \sqrt{n}}$$

Ví dụ:

Để xác định liệu điểm thi trung bình của nữ có  $> 600$  trong bài kiểm tra hay không? Chúng ta không có thông tin liên quan đến phương sai (variance) hoặc độ lệch chuẩn điểm thi của

Nữ. Để thực hiện t-test, chúng ta chọn ngẫu nhiên điểm thi của 10 nữ như sau (với mức ý nghĩa  $\alpha$  là 0.05 để kiểm định giả thuyết).



Girls_Score
587
602
627
610
619
622
605
608
596
592

Trong ví dụ này:

- Trung bình điểm thi của Nữ: 606.8
- Kích thước mẫu: 10
- Trung bình tổng thể (population mean): 600
- Độ lệch chuẩn mẫu (Standard Deviation for the sample): 13.14

$$t = \frac{\bar{x} - \mu}{s / \sqrt{n}}$$

$$= \frac{606.8 - 600}{13.14 / \sqrt{10}}$$


$$= 1.64$$

Critical Value = 1.833

t score < Critical Value


P value = 0.0678

P value > 0.05



$H_0: \mu \leq 600$

$H_a: \mu > 600$



P-value > 0.05 vì vậy không có cơ sở để bác bỏ  $H_0$  (null hypothesis). Không có đủ bằng chứng để kết luận rằng điểm trung bình của Nữ lớn hơn 600.

### Two-Sample t-Test

Two-Sample t-test được thực hiện để so sánh trung bình của 2 mẫu.

$$t = \frac{(\bar{x}_1 - \bar{x}_2) - (\mu_1 - \mu_2)}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

Diagram illustrating the components of the Two-Sample t-Test formula:

- Difference bw Sample mean:**  $\bar{x}_1 - \bar{x}_2$
- Difference bw population mean:**  $\mu_1 - \mu_2$
- Sample standard deviation:**  $s_1, s_2$
- Sample Size:**  $n_1, n_2$

### Ví dụ: Two-Sample t-Test

Chúng ta muốn kiểm tra liệu điểm trung bình của nam lớn hơn điểm trung bình của Nữ là 10 điểm hay không. Chúng ta không có thông tin về phương sai (hay độ lệch chuẩn) về điểm thi của cả Nam và Nữ. Thu thập dữ liệu về điểm thi của 10 Nam và 10 Nữ một cách ngẫu nhiên như sau (với mức ý nghĩa  $\alpha$  là 0.05 để kiểm định giả thuyết):

Girls_Score	Boys_Score
587	626
602	643
627	647
610	634
619	630
622	649
605	625
608	623
596	617
592	607

Trong ví dụ này:

- Điểm trung bình của Nam: 630.1
- Điểm trung bình của Nữ: 606.8
- Khác biệt điểm trung bình của Nam và Nữ: 10
- Độ lệch chuẩn điểm thi Nam: 13.42
- Độ lệch chuẩn điểm thi Nữ: 13.14

$$t = \frac{(\bar{x}_1 - \bar{x}_2) - (\mu_1 - \mu_2)}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

$$\frac{(630.1 - 606.8) - (15)}{\sqrt{\frac{(13.42)^2}{10} + \frac{(13.14)^2}{10}}}$$

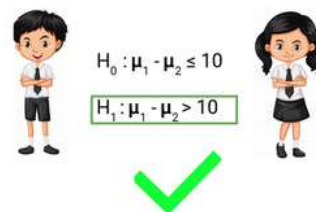
Critical Value = 1.833

t = 2.23

P value = 0.019

Critical Value > t score

P value < 0.05



Vì  $P\text{-value} < 0.05$  nên bác bỏ  $H_0$  (chấp nhận  $H_1$ ) và kết luận rằng điểm trung bình của Nam cao hơn điểm trung bình của Nữ 10 điểm.

Minh họa kiểm định giả thuyết bằng Python

```
import statsmodels.api as sm #for hypothesis testing
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
xls = pd.ExcelFile('D:/Python_Pro/Data_hypothesis_testing.xlsx')
df1 = pd.read_excel(xls, 'Sheet1') #Read sheet1
df2 = pd.read_excel(xls, 'Sheet2')
df3 = pd.read_excel(xls, 'Sheet3')
df4 = pd.read_excel(xls, 'Sheet4')
```

```
df2.sample(10)
```

	Girls_Score	Boys_Score
32	690	485
9	620	579
25	630	659
6	690	673
4	480	440
7	530	519
15	490	463
26	785	637
14	650	615
23	720	579

```
#Z test two-sided
girls = df2['Girls_Score'] #điểm của nữ
boys = df2['Boys_Score'] #điểm của nam
sm.stats.ztest(girls, boys, alternative='two-sided')

(1.18126840231646, 0.23749611550365257)
```

```
== ==
μ1: Điểm trung bình của Nữ
μ2: Điểm trung bình của Nam
Null Hypothesis (H0): μ1=μ2
Alternative Hypothesis (Ha): μ1≠μ2
Kết luận: Vì p-value ≈0.237 (>0.05), nên ta không có cơ sở để bác bỏ H0.
Có nghĩa là chúng ta không có đủ bằng chứng để kết luận rằng
điểm trung bình của nữ khác trung bình của Nam
== ==
```

# Hãy kiểm tra xem trung bình điểm toán học kì 2 lớp 12 có bằng 8.0

`ztest(df['T6'], value=8.0)`

Kết quả: (-7.797828845339864, 6.298135014120743e-15)



```
# Kết luận: điểm trung bình không bằng 8
```

```
# TWO SAMPLE Z-TEST
```

```
# Tương tự  $n > 30$ 
```

```
# H0:  $\mu_1 = \mu_2$  both population means are equal
```

```
# Câu 4: Hãy kiểm tra xem điểm trung bình toán học kì 1 và học kì 2 năm lớp 12 có bằng nhau không
```

```
ztest(df['T5'],df['T6'],value=0)
```

```
Kết quả: (-1.094138573502891, 0.273894207026412)
```

```
# Đủ dữ kiện kết luận trung bình 2 học kì có bằng nhau
```

```
# Tương tự kiểm tra trung bình LOGIC và UNGXU có bằng nhau không
```

```
ztest(df['LOGIC'],df['UNGXU'],value=0)
```

```
# Nhỏ hơn => bác bỏ
```

```
Kết quả: (-4.172765180703833, 3.009250404643791e-05)
```

```
# Tương tự kiểm tra trung bình LOGIC và UNGXU có bằng nhau không với mức tin cậy 95%
```

```
'''
```

```
# Kiểm định tương quan giữa 2 biến định lượng
```

```
# H0:  $r = 0$ 
```

```
'''
```

```
from scipy.stats.stats import pearsonr
```

```
# Câu 5: Kiểm tra xem điểm toán hk1 và hk2 năm lớp 12 có tương quan không
```

```
pearsonr(df['T5'],df['T6'])
```

```
# Bác bỏ H0:  $r = 0$  do  $pvalue < 5\% \Rightarrow r \neq 0 \Rightarrow$  2 biến không tương quan ( $pvalue$  thì tương quan trên tổng thể, và k cho biết tương quan thuận hay nghịch )
```

```
# statistic là gtri tương quan trên mẫu, nhìn lên sẽ biết tương quan mạnh hay sao
```

Kết quả: `PearsonRResult(statistic=0.7786831657869809, pvalue=1.4846407216273482e-21)`

```
# Kết luận: đủ dữ liệu để nói rằng T5 và T6 có tương quan với nhau với mức sai lầm là 5%
'''
Kết luận trên mẫu: tương quan thuận, mức độ tương quan rất cao: r = 0.7786831657869809
Suy diễn trên tổng thể: Đủ dữ liệu để nói rằng T5 và T6 có tương quan với nhau với mức sai lầm là 5%
Với p-value = 1.4846407216273482e-21
'''
'''
Định tính: Fisher (<30)
Chi square (>30)
Cả 2 H0: độc lập
'''
'''
Định lượng:
one sample t-test và one sample z-test => biến định lượng (H0;  $\mu = \alpha$ )
two sample t-test và two sample z-test => kiểm tra 2 biến định lượng có bằng nhau không (H0:  $\mu_1 - \mu_2 = 0$ )
kiểm định Pearson kiểm tra xem 2 biến định lượng có tương quan không (H0:  $\rho = 0$ )
'''
```

```
# Sinh viên làm tương tự cho T5 và LOGIC có tương quan hay không
pearsonr(df['T5'],df['LOGIC'])
# statis: quá thấp, còn pvalue > 0.05 nên chấp nhận H0: r=0 => trên tổng thể không tương quan
PearsonRResult(statistic=0.18464661226012727, pvalue=0.06590059130545516)
```

```
# Fisher Test
# Mục đích: Kiểm tra sự độc lập của 2 biến định tính dạng nhị phân 2x2
```

```
# Ho: Không có sự khác biệt giữa 2 biến định tính
import scipy.stats as stats
```

```
# Hãy kiểm tra xem có sự phụ thuộc nào giữa việc sinh viên có định hướng nghề nghiệp
và giới tính khi thí sinh đăng ký dự thi hay không
crodata =
pd.crosstab(df['GT'],[df['HUONGNGHIEP']],rownames=['GT'],colnames=['HUONGNG
HIEP'])
crodata
```

```
odd_ratio, p_value = stats.fisher_exact(crodata)
print('odd ratio is: ' + str(odd_ratio))
print('p_value is: ' + str(p_value))
```

```
# Kết luận: Chấp nhận Ho vì  $p\_value = 0.22763 > \alpha = 0.05$ 
# Tức là, đủ dữ liệu để nói rằng giới tính và việc định hướng nghề nghiệp là không có
quan hệ gì cả ở mức sai lầm 5%
```

```
# Chi-Square Test
# Mục đích: Kiểm tra sự độc lập của 2 biến định tính
# Ho: Không có sự khác biệt giữa 2 biến định tính
from scipy.stats import chi2_contingency
```

```
# Hãy kiểm tra xem có sự phụ thuộc nào giữa khối thi và khu vực thi đăng ký dự thi hay
không
crodata = pd.crosstab(df['KV'],[df['KT']],rownames=['KV'],colnames=['KT'])
crodata
```

```
stat, p, dof, expected = chi2_contingency(crodata)
```

```
# interpret p-value
alpha = 0.05
print("p value is " + str(p))
if p <= alpha:
    print ('Dependent (Reject H0)')
else:
    print ('Independent (H0 holds true)')
# Có quan hệ giữa khối thi và khu vực thi đăng ký dự thi
# Kết luận: p-value = 0.02 < alpha = 0.05
# Tức là: không đủ dữ liệu để nói rằng KV và KT là độc lập hay có sự quan hệ giữa KT và KV
# GT và KT có mối quan hệ hay không
crodata = pd.crosstab(df['GT'],[df['KT']],rownames=['GT'],colnames=['KT'])
crodata
```

```
stat, p, dof, expected = chi2_contingency(crodata)
# interpret p-value
alpha = 0.05
print("p value is " + str(p))
if p <= alpha:
    print ('Dependent (Reject H0) => có quan hệ với nhau')
else:
    print ('Independent (H0 holds true) => Độc lập với nhau')
```

```
# ONE WAY ANOVA
# Kiểm định ANOVA ONE WAY
# Yêu cầu:
# 1. Biến định lượng trên nhóm định tính
# 2. Các biến định lượng trên từng nhóm theo phân phối chuẩn
# 3. H0: Giá trị trung bình dữ liệu định lượng trên từng nhóm định tính là bằng nhau
```

```
# Điểm toán học kì 2 lớp 12 có phụ thuộc vào giới tính hay không
import statsmodels.api as sm
from statsmodels.formula.api import ols
model = ols('T6 ~ GT', data=df).fit()
aov_table = sm.stats.anova_lm(model, typ = 1)
aov_table
```

```
'''
Trả lời:
H0: mean (nhóm GT) bằng nhau
p-value = 0.363426 > 0.05 => chấp thuận
Không phụ thuộc
'''
```

```
# Điểm LOGIC có phụ thuộc vào KV hay không
model = ols('LOGIC ~ KV', data=df).fit()
aov_table = sm.stats.anova_lm(model, typ = 1)
aov_table
# Ko phụ thuộc
```

```
# Điểm UNGXU có phụ thuộc khối thi hay không
model = ols('UNGXU ~ KT', data=df).fit()
aov_table = sm.stats.anova_lm(model, typ = 1)
aov_table
# p_value = 0.46041 => chấp nhận => không phụ thuộc
```

```
'''
Two way anova
# Kiểm định ANOVA two WAY
# Yêu cầu:
# 1. Biến định lượng trên nhóm định tính
```

```
# 2. Các biến định lượng trên từng nhóm theo phân phối chuẩn
# 3. H0: Trung bình các cột dữ liệu bằng nhau
'''
```

```
# Hãy cho biết điểm LOGIC có phụ thuộc vào loại GT trên từng nhóm KV hay không
# Performing two-way anova
model = ols('LOGIC ~ GT + KV + GT:KV', data=df).fit()
result = sm.stats.anova_lm(model, type = 2)
# Print the result
print(result)
# Chấp nhận Ho=> k phụ thuộc (p-value = 0.052602)
# Bác bỏ H0 => phụ thuộc (pvalue = 0.019173)
# GT:KV bác bỏ -> k phụ thuộc -> tuy nhiên điểm LOGIC trên từng nhóm GT xét theo
từng KV nữa thì nó độc lập với nhau
'''
```

Kết luận khác

```
p-value = 0.052602 -> LOGIC độc lập theo nhóm GT
p-value = 0.019173 -> LOGIC phụ thuộc theo nhóm KV
p-value = 0.576510 -> LOGIC độc lập theo nhóm GT trên từng loại KV
'''
```

```
# Phân tích xem NGONNGU có phụ thuộc theo nhóm KV trên từng nhóm KT hay không
model = ols('NGONNGU ~ KV + KT + KV:KT', data=df).fit()
result = sm.stats.anova_lm(model, type = 2)
print(result)
'''
Kết luận
Tất cả đều phụ thuộc
'''
```

## BÀI TẬP TUẦN 6: HỒI QUY TUYẾN TÍNH

Đọc thêm: Linear Regression in Python using Statsmodels

Linear regression analysis is a statistical technique for predicting the value of one variable(dependent variable) based on the value of another(independent variable). The dependent variable is the variable that we want to predict or forecast. In simple linear regression, there's one independent variable used to predict a single dependent variable. In the case of multilinear regression, there's more than one independent variable. The independent variable is the one you're using to forecast the value of the other variable. The **statsmodels.regression.linear\_model.OLS** method is used to perform linear regression. Linear equations are of the form:

**Syntax:** `statsmodels.regression.linear_model.OLS(endog, exog=None, missing='none', hasconst=None, **kwargs)`

**Parameters:**

- **endog:** array like object.
- **exog:** array like object.
- **missing:** str. None, decrease, and raise are the available alternatives. If the value is 'none,' no nan testing is performed. Any observations with nans are dropped if 'drop' is selected. An error is raised if 'raise' is used. 'none' is the default.
- **hasconst:** None or Bool. Indicates whether a user-supplied constant is included in the RHS. If True, k constant is set to 1 and all outcome statistics are calculated as if a constant is present. If False, k constant is set to 0 and no constant is verified.
- **\*\*kwargs:** When using the formula interface, additional arguments are utilised to set model characteristics.

**Return:** Ordinary least squares are returned.

Installation

```
pip install numpy
```

```
pip install pandas
```

```
pip install statsmodels
```

### Stepwise Implementation

#### Step 1: Import packages.

Importing the required packages is the first step of modeling. The pandas, NumPy, and stats model packages are imported.

```
import numpy as np
```

```
import pandas as pd
```

```
import statsmodels.api as sm
```

## Step 2: Loading data.

To access the CSV file click [here](#). The CSV file is read using `pandas.read_csv()` method. The head or the first five rows of the dataset is returned by using the `head()` method. Head size and Brain weight are the columns.

- Python3

```
df = pd.read_csv('headbrain1.csv')
df.head()
```

The head of the data frame looks like this:

Out[2]:

	Head Size(cm^3)	Brain Weight(grams)
0	4512	1530
1	3738	1297
2	4261	1335
3	3777	1282
4	4177	1590

## Visualizing the data:

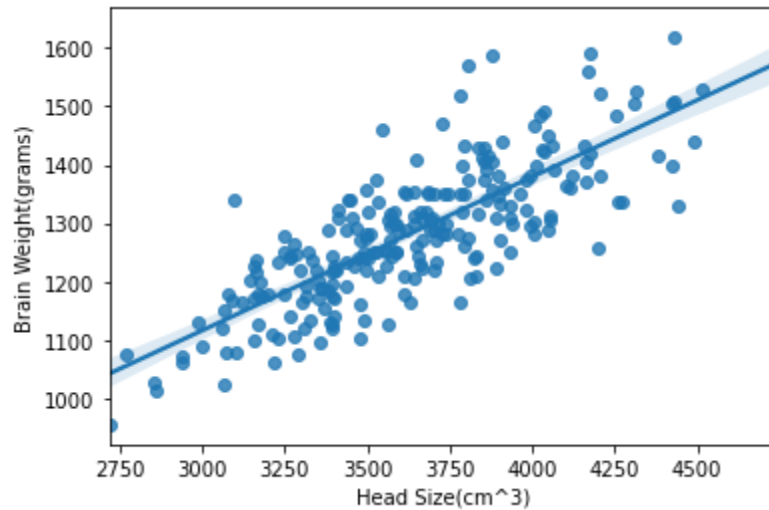
By using the [matplotlib](#) and seaborn packages, we visualize the data. `sns.regplot()` function helps us create a regression plot.

- Python3

```
# import packages
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv('headbrain1.csv')
sns.regplot('Head Size(cm^3)', 'Brain Weight(grams)',
data=df)
plt.show()
```

**Output:**





### Step 3: Setting a hypothesis.

- Null hypothesis (H<sub>0</sub>): There is no relationship between head size and brain weight.
- Alternative hypothesis (H<sub>a</sub>): There is a relationship between head size and brain weight.

### Step 4: Fitting the model

[statsmodels.regression.linear\\_model.OLS\(\)](#) method is used to get ordinary least squares, and `fit()` method is used to fit the data in it. The `ols` method takes in the data and performs linear regression. we provide the dependent and independent columns in this format :

***independent\_columns ~ dependent\_column:***

*left side of the ~ operator contains the independent variables and right side of the operator contains the name of the dependent variable or the predicted column.*

- Python3

```
df.columns = ['Head_size', 'Brain_weight']
model = smf.ols(formula='Head_size ~ Brain_weight',
data=df).fit()
```

### Step 5: Summary of the model.

All the summary statistics of the linear regression model are returned by the `model.summary()` method. The p-value and many other values/statistics are known by this method. Predictions about the data are found by the `model.summary()` method.

```
print(model.summary())
```

### Code Implementation:

```
# import packages
import numpy as np
import pandas as pd
import statsmodels.formula.api as smf
# loading the csv file
df = pd.read_csv('headbrain1.csv')
print(df.head())
# fitting the model
df.columns = ['Head_size', 'Brain_weight']
model = smf.ols(formula='Head_size ~ Brain_weight',
data=df).fit()
# model summary
print(model.summary())
```

# Output:

```

=====
                        OLS Regression Results
=====
Dep. Variable:          Head_size      R-squared:                0.639
Model:                  OLS            Adj. R-squared:         0.638
Method:                 Least Squares   F-statistic:           416.5
Date:                   Sun, 08 May 2022 Prob (F-statistic):     5.96e-54
Time:                   21:40:40        Log-Likelihood:        -1613.4
No. Observations:       237            AIC:                   3231.
Df Residuals:           235            BIC:                   3238.
Df Model:                1
Covariance Type:        nonrobust
=====
                        coef      std err          t      P>|t|      [0.025      0.975]
-----
Intercept      520.6101      153.215      3.398      0.001      218.759      822.461
Brain_weight    2.4269         0.119     20.409      0.000         2.193         2.661
=====
Omnibus:                2.687      Durbin-Watson:           1.726
Prob(Omnibus):          0.261      Jarque-Bera (JB):         2.321
Skew:                   0.207      Prob(JB):                 0.313
Kurtosis:               3.252      Cond. No.                 1.38e+04
=====

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.38e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

```

Description of some of the terms in the table :

- **R- squared value:** R-squared value ranges between 0 and 1. An R-squared of 100 percent indicates that all changes in the dependent variable are completely explained by changes in the independent variable(s). if we get 1 as an r-squared value it means there's a perfect fit. In our example, the r-squared value is 0.638.
- **F- statistic:** The F statistic simply compares the combined effect of all variables. In simplest terms, reject the null hypothesis if your alpha level is greater than your p-value.
- **coef:** the coefficients of the independent variables in the regression equation.

Giá trị bình phương R: Giá trị bình phương R nằm trong khoảng từ 0 đến 1. Bình phương R bằng 100 phần trăm cho thấy rằng tất cả những thay đổi trong biến phụ thuộc hoàn toàn được giải thích bằng những thay đổi trong (các) biến độc lập. nếu chúng ta lấy 1 làm giá trị r bình phương thì có nghĩa là có sự phù hợp hoàn hảo. Trong ví dụ của chúng tôi, giá trị r bình phương là 0,638.

- **Thống kê F:** Thống kê F chỉ đơn giản so sánh tác động tổng hợp của tất cả các biến. Nói một cách đơn giản nhất, hãy bác bỏ giả thuyết không nếu mức alpha của bạn lớn hơn giá trị p của bạn.

- **coef:** hệ số của các biến độc lập trong phương trình hồi quy.

Dự đoán của chúng tôi:

Nếu chúng tôi lấy mức ý nghĩa (alpha) là 0,05 thì chúng tôi bác bỏ giả thuyết không và chấp nhận giả thuyết thay thế là  $p < 0,05$ . vì vậy, chúng ta có thể nói rằng có mối quan hệ giữa kích thước đầu và trọng lượng não.

Our predictions:

If we take our significance level (alpha) to be 0.05, we reject the null hypothesis and accept the alternative hypothesis as  $p < 0.05$ . so, we can say that there is a relationship between head size and brain weight.

## TÓM TẮT HỒI QUY TUYẾN TÍNH LÀ GÌ?

Hồi quy tuyến tính là gì? Hồi quy tuyến tính là một mô hình thống kê phân tích mối quan hệ giữa một biến phản hồi (thường được gọi là y) và một hoặc nhiều biến và tương tác của chúng (thường được gọi là X hoặc các biến giải thích). a statistical model that analyzes the relationship between a response variable (often called y) and one or more variables and their interactions (often called x or explanatory variables).

### Hồi quy tuyến tính trong Python là gì?

Hồi quy tuyến tính là một phương pháp thống kê để mô hình hóa mối quan hệ giữa một biến phụ thuộc với một tập hợp các biến độc lập nhất định. Lưu ý: Trong bài viết này, chúng tôi gọi các biến phụ thuộc là phản hồi và các biến độc lập là các tính năng để đơn giản. a statistical method for modeling relationships between a dependent variable with a given set of independent variables. Note: In this article, we refer to dependent variables as responses and independent variables as features for simplicity.

### Làm thế nào để bạn tìm thấy bản tóm tắt của một mô hình trong Python?

Nếu bạn muốn trích xuất bản tóm tắt mô hình hồi quy trong Python, bạn nên sử dụng gói StatModels. use the statsmodels package.

### Hướng dẫn summary of linear regression in python - tóm tắt hồi quy tuyến tính trong python

Thường thì bạn có thể muốn trích xuất một bản tóm tắt của một mô hình hồi quy được tạo bằng cách sử dụng Scikit-learn trong Python.

Scikit-Learn không cung cấp nhiều chức năng tích hợp để phân tích bản tóm tắt của mô hình hồi quy vì nó thường chỉ được sử dụng cho mục đích dự đoán.

Vì vậy, nếu bạn quan tâm đến việc có được một bản tóm tắt về mô hình hồi quy trong Python, bạn có hai tùy chọn:

- 1. Sử dụng các chức năng hạn chế từ scikit-learn.** Use limited functions from scikit-learn.
- 2. Sử dụng StatModels thay thế.** Use statsmodels instead.

Các ví dụ sau đây cho thấy cách sử dụng từng phương thức trong thực tế với các cấu trúc sau đây:

```
import pandas as pd

#create DataFrame
df = pd.DataFrame({'x1': [1, 2, 2, 4, 2, 1, 5, 4, 2, 4, 4],
                  'x2': [1, 3, 3, 5, 2, 2, 1, 1, 0, 3, 4],
                  'y': [76, 78, 85, 88, 72, 69, 94, 94, 88, 92, 90]})

#view first five rows of DataFrame
df.head()
```

	<b>x1</b>	<b>x2</b>	<b>y</b>
<b>0</b>	<b>1</b>	<b>1</b>	<b>76</b>
<b>1</b>	<b>2</b>	<b>3</b>	<b>78</b>
<b>2</b>	<b>2</b>	<b>3</b>	<b>85</b>
<b>3</b>	<b>4</b>	<b>5</b>	<b>88</b>
<b>4</b>	<b>2</b>	<b>2</b>	<b>72</b>

### **Phương pháp 1: Nhận bản tóm tắt mô hình hồi quy từ Scikit-learn**

Chúng ta có thể sử dụng mã sau để phù hợp với mô hình hồi quy tuyến tính nhiều bằng cách sử dụng Scikit-LEARN:

```
from sklearn.linear_model import LinearRegression
#initiate linear regression model
model = LinearRegression()
#define predictor and response variables
X, y = df[['x1', 'x2']], df.y
#fit regression model
model.fit(X, y)
```

Sau đó, chúng ta có thể sử dụng mã sau để trích xuất các hệ số hồi quy của mô hình cùng với giá trị R-Squared của mô hình:

```
#display regression coefficients and R-squared value of model
```

```
print(model.intercept_, model.coef_, model.score(X, y))
70.4828205704 [ 5.7945 -1.1576] 0.766742556527
```

Sử dụng đầu ra này, chúng ta có thể viết phương trình cho mô hình hồi quy được trang bị:

$$y = 70,48 + 5,79x_1 - 1,16x_2$$

Chúng ta cũng có thể thấy rằng giá trị R2 của mô hình là 76,67.

Điều này có nghĩa là 76,67% biến thể trong biến phản hồi có thể được giải thích bằng hai biến dự đoán trong mô hình. Mặc dù đầu ra này rất hữu ích, nhưng chúng tôi vẫn không biết tổng thể F-thống kê của mô hình, giá trị p của các hệ số hồi quy và các số liệu hữu ích khác có thể giúp chúng tôi hiểu mô hình phù hợp với bộ dữ liệu này như thế nào.

### Phương pháp 2: Nhận tóm tắt mô hình hồi quy từ StatModels

Nếu bạn quan tâm đến việc trích xuất bản tóm tắt mô hình hồi quy trong Python, thì bạn nên sử dụng gói StatModels.**statsmodels** package.

Mã sau đây cho thấy cách sử dụng gói này để phù hợp với cùng một mô hình hồi quy tuyến tính giống như ví dụ trước và trích xuất bản tóm tắt mô hình:

```
import statsmodels.api as sm
#define response variable
y = df['y']
#define predictor variables
x = df[['x1', 'x2']]
#add constant to predictor variables
x = sm.add_constant(x)
#fit linear regression model
model = sm.OLS(y, x).fit()
#view model summary
print(model.summary())
```

### OLS Regression Results

```
=====
=====
Dep. Variable:          y  R-squared:          0.767
Model:                OLS  Adj. R-squared:      0.708
Method:             Least Squares  F-statistic:      13.15
Date:              Fri, 01 Apr 2022  Prob (F-statistic):    0.00296
```

Time: 11:10:16 Log-Likelihood: -31.191  
 No. Observations: 11 AIC: 68.38  
 Df Residuals: 8 BIC: 69.57  
 Df Model: 2  
 Covariance Type: nonrobust

	coef	std err	t	P> t	[0.025	0.975]
const	70.4828	3.749	18.803	0.000	61.839	79.127
x1	5.7945	1.132	5.120	0.001	3.185	8.404
x2	-1.1576	1.065	-1.087	0.309	-3.613	1.298

Omnibus: 0.198 Durbin-Watson: 1.240  
 Prob(Omnibus): 0.906 Jarque-Bera (JB): 0.296  
 Skew: -0.242 Prob(JB): 0.862  
 Kurtosis: 2.359 Cond. No. 10.7

Lưu ý rằng các hệ số hồi quy và giá trị bình phương R phù hợp với các hệ số được tính toán bởi Scikit-learn, nhưng chúng tôi cũng cung cấp một tấn các số liệu hữu ích khác cho mô hình hồi quy.

Ví dụ: chúng ta có thể thấy các giá trị p cho từng biến dự đoán riêng lẻ:

- Giá trị p cho x1 = .001
- Giá trị P cho x2 = 0,309

Chúng ta cũng có thể thấy thống kê F tổng thể của mô hình, giá trị bình phương R được điều chỉnh, giá trị AIC của mô hình và nhiều hơn nữa.

### Hồi quy tuyến tính giải thích với ví dụ là gì?

Hồi quy tuyến tính thường được sử dụng để phân tích và mô hình hóa dự đoán. Ví dụ, nó có thể được sử dụng để định lượng các tác động tương đối của tuổi, giới tính và chế độ ăn uống (các biến dự đoán) về chiều cao (biến kết quả). commonly used for predictive analysis and modeling.

For example, it can be used to quantify the relative impacts of age, gender, and diet (the predictor variables) on height (the outcome variable).

## Hướng dẫn linear regression python residuals - phần dư python hồi quy tuyến tính

Trong hướng dẫn này, bạn sẽ thấy cách thực hiện hồi quy tuyến tính nhiều trong Python bằng cả Sklearn và StatSmodels.

Nội phân chính

- Kiểm tra tuyến tính
- Thực hiện hồi quy tuyến tính nhiều chiều
- (1) Phần đầu tiên hiển thị đầu ra được tạo bởi sklearn:
- (2) Phần thứ hai hiển thị một bảng toàn diện với thông tin thống kê được tạo bởi StatSmodels.

Thực hiện hồi quy tuyến tính nhiều chiều trong Python

```
import pandas as pd
import matplotlib.pyplot as plt
data = {'year':
[2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2016,2016,2016,2016,
2016,2016,2016,2016,2016,2016,2016,2016],
'month': [12,11,10,9,8,7,6,5,4,3,2,1,12,11,10,9,8,7,6,5,4,3,2,1],
'interest_rate':
[2.75,2.5,2.5,2.5,2.5,2.5,2.25,2.25,2.25,2.2,2,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75],
'unemployment_rate':
[5.3,5.3,5.3,5.3,5.4,5.6,5.5,5.5,5.5,5.6,5.7,5.9,6,5.9,5.8,6.1,6.2,6.1,6.1,6.1,5.9,6.2,6.2,6.1],
'index_price':
[1464,1394,1357,1293,1256,1254,1234,1195,1159,1167,1130,1075,1047,965,943,958,971,949,884,866,876,822,704,719]
}

df = pd.DataFrame(data)
plt.scatter(df['interest_rate'], df['index_price'], color='red')
plt.title('Index Price Vs Interest Rate', fontsize=14)
```



```
plt.xlabel('Interest Rate', fontsize=14)
plt.ylabel('Index Price', fontsize=14)
plt.grid(True)
plt.show()

df = pd.DataFrame(data)
plt.scatter(df['unemployment_rate'], df['index_price'], color='green')
plt.title('Index Price Vs Unemployment Rate', fontsize=14)
plt.xlabel('Unemployment Rate', fontsize=14)
plt.ylabel('Index Price', fontsize=14)
plt.grid(True)
plt.show()
```

### Thực hiện hồi quy tuyến tính nhiều chiều

- (1) Phần đầu tiên hiển thị đầu ra được tạo bởi sklearn:
- (2) Phần thứ hai hiển thị một bảng toàn diện với thông tin thống kê được tạo bởi StatModels.

Dưới đây là các chủ đề được đề cập:

```
import pandas as pd
from sklearn import linear_model
import statsmodels.api as sm
data = {'year':
[2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2016,2016,2016,2016,
2016,2016,2016,2016,2016,2016,2016,2016],
      'month': [12,11,10,9,8,7,6,5,4,3,2,1,12,11,10,9,8,7,6,5,4,3,2,1],
      'interest_rate':
[2.75,2.5,2.5,2.5,2.5,2.5,2.5,2.25,2.25,2.25,2,2,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75],
      'unemployment_rate':
[5.3,5.3,5.3,5.3,5.4,5.6,5.5,5.5,5.5,5.6,5.7,5.9,6,5.9,5.8,6.1,6.2,6.1,6.1,6.1,5.9,6.2,6.2,6.1],
      'index_price':
[1464,1394,1357,1293,1256,1254,1234,1195,1159,1167,1130,1075,1047,965,943,958,971,949,884,866,876,822,704,719]}
```

```

    }
df = pd.DataFrame(data)
x = df[['interest_rate','unemployment_rate']]
y = df['index_price']
# with sklearn
regr = linear_model.LinearRegression()
regr.fit(x, y)
print('Intercept: \n', regr.intercept_)
print('Coefficients: \n', regr.coef_)
# with statsmodels
x = sm.add_constant(x) # adding a constant
model = sm.OLS(y, x).fit()
predictions = model.predict(x)
print_model = model.summary()
print(print_model)

```

Xem xét ví dụ sẽ được sử dụng trong hướng dẫn này

**(1) Phần đầu tiên hiển thị đầu ra được tạo bởi sklearn:**

```

Intercept:
1798.4039776258564
Coefficients:
[ 345.54008701 -250.14657137]

```

(2) Phần thứ hai hiển thị một bảng toàn diện với thông tin thống kê được tạo bởi StatModels.

Dưới đây là các chủ đề được đề cập:  $(\text{intercept}) + (\text{interest\_rate coef}) * X1 + (\text{unemployment\_rate coef}) * X2$

Thực hiện hồi quy tuyến tính nhiều biến trong Python:  $(1798.4040) + (345.5401) * X1 + (-250.1466) * X2$

**(2) Phần thứ hai hiển thị một bảng toàn diện với thông tin thống kê được tạo bởi StatModels.**

Dưới đây là các chủ đề được đề cập:

OLS Regression Results

```
=====
Dep. Variable:    index_price  R-squared:    0.898
Model:           OLS  Adj. R-squared:    0.888
Method:          Least Squares  F-statistic:    92.07
Date:            Sat, 30 Jul 2022  Prob (F-statistic):    4.04e-11
Time:            13:47:01  Log-Likelihood:    -134.61
No. Observations:    24  AIC:    275.2
Df Residuals:        21  BIC:    278.8
Df Model:            2
Covariance Type:    nonrobust
=====
```

```
=====
              coef      std err      t      P>|t|    [0.025    0.975]
-----
const          1798.4040      899.248      2.000      0.059     -71.685     3668.493
interest_rate    345.5401     111.367      3.103      0.005     113.940     577.140
unemployment_rate -250.1466     117.950     -2.121      0.046    -495.437     -4.856
=====
```

```
=====
Omnibus:          2.691  Durbin-Watson:          0.530
Prob(Omnibus):    0.260  Jarque-Bera (JB):          1.551
Skew:             -0.612  Prob(JB):          0.461
Kurtosis:         3.226  Cond. No.          394.
=====
```

**Chú ý:** phải xác nhận rằng một số giả định được đáp ứng trước khi bạn áp dụng các mô hình hồi quy tuyến tính. Đáng chú ý nhất, bạn phải đảm bảo rằng mối quan hệ tuyến tính tồn tại giữa biến phụ thuộc và biến/s độc lập (nhiều hơn về phần kiểm tra tuyến tính).

<https://v1study.com/python-tham-khao-hoi-quy-tuyen-tinh-trong-python.html>  
<https://www.geeksforgeeks.org/python-coefficient-of-determination-r2-score/>  
<https://www.statsmodels.org/dev/examples/notebooks/generated/predict.html>  
<https://builtin.com/data-science/train-test-split>

<https://datagy.io/mean-squared-error-python/>

## 1) LAB 1 HỒI QUY TUYẾN TÍNH

### # Dự đoán doanh số

( Hồi quy tuyến tính cơ bản)

#### **\*\*Vấn đề\*\***

Xây dựng một mô hình dự đoán doanh số bán hàng dựa trên số tiền chi cho các nền tảng tiếp thị khác nhau.

#### **\*\*Dữ liệu\*\***

Sử dụng bộ dữ liệu quảng cáo được cung cấp trong ISLR và phân tích mối quan hệ giữa 'quảng cáo truyền hình' và 'doanh số bán hàng' bằng cách sử dụng mô hình hồi quy tuyến tính đơn giản.

### ## 1. Đọc và hiểu dữ liệu

# Import the numpy and pandas package

import numpy as np

import pandas as pd

# Data Visualisation

import matplotlib.pyplot as plt

import seaborn as sns

### ### 1.1. Đọc dữ liệu

Đọc dữ liệu từ file CSV và in ra 5 dòng đầu tiên

```
advertising = pd.DataFrame(pd.read_csv("advertising - advertising.csv"))
```

```
advertising.head()
```

### ### 1.2. In kích thước của dữ liệu

Keyword: Shape

#In kích thước

```
advertising.shape
```

### ### 1.3. In ra thông tin của dữ liệu

Keyword: Info

```
#In ra thông tin của dữ liệu
```

```
advertising.info()
```

### ### 1.4. In ra bảng thống kê mô tả

Keyword: Describe()

```
#In ra bảng thống kê mô tả
```

```
advertising.describe()
```

## ## 2. Làm sạch dữ liệu

### ### 2.1. Kiểm tra null

Keyword: isnull, sum

```
#Kiểm tra giá trị null
```

```
advertising.isnull().sum()
```

```
#Xem thông tin của TV
```

```
advertising[['TV']]
```

### ### 2.2. Kiểm tra outlier

Vẽ boxplot cho dữ liệu TV

```
#Kiểm tra ngoại lệ
```

```
sns.boxplot(advertising['TV'])
```

```
plt.show()
```

## ## 3. Khám phá dữ liệu

Vẽ boxplot của Sale

### ### 3.1. Vẽ boxplot cho Sale

Vẽ boxplot của Sale

```
#Xem Sales
```

```
advertising['Sales']
```

```
#Vẽ boxplot của Sale
```

```
sns.boxplot(advertising['Sales'])
```

```
plt.show()
```

### ### 3.2. Vẽ pairplot

Vẽ pairplot để thấy tương quan giữa Sales và các biến khác.

```
#Vẽ pairplot để thấy tương quan giữa Sales và các biến khác
```

```
sns.pairplot(advertising, x_vars=['TV', 'Newspaper', 'Radio'], y_vars='Sales', height=4, aspect=1, kind='scatter')
```

```
plt.show()
```

### ### 3.3. Vẽ heatmap

Vẽ heatmap để thấy tương quan giữa các biến dữ liệu với nhau

```
#Vẽ heatmap để thấy tương quan giữa các biến khác nhau
```

```
# Let's see the correlation between different variables.
```

```
sns.heatmap(advertising.corr(), cmap="YlGnBu", annot = True)
```

```
plt.show()
```

## ## 4. Xây dựng mô hình

Phương trình hồi quy tuyến tính

$$y = c + m_1x_1 + m_2x_2 + \dots + m_nx_n$$

y

là kết quả

c

là chặn

$m_1$

là hệ số cho biến đầu tiên

$m_n$

là hệ số cho biến thứ  $n$

Trong trường hợp của chúng ta:

$$y = c + m_1 \times TV$$

$m$  là các giá trị được gọi là hệ số mô hình hoặc tham số mô hình.

### ### 4.1. Tạo biến X và Y

X lưu dữ liệu TV, Y lưu dữ liệu Sales

```
X = advertising["TV"]
y = advertising["Sales"]
print(X)
print(y)
```

### ### 4.2. Chia tập train và tập test

Sử dụng sklearn.model\_selection để chia bộ dữ liệu thành 2 phần, 1 phần gọi là tập huấn luyện (train), phần còn lại gọi là tập kiểm thử (test)

Tập train chiếm 70% và tập test chiếm 30%

```
from sklearn.model_selection import train_test_split
#from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.7, test_size = 0.3,
random_state = 100)
print(X_train)
print(X_test)
print(y_train)
print(y_test)
```

### ### 4.3. Xem tập train và test

In ra 5 dòng đầu tiên của tập train

```
X_train.head()
```

```
y_train.head()
```

### ### 4.4. Xây dựng mô hình

Sử dụng statsmodels.api để xây dựng mô hình

```
import statsmodels.api as sm
```

Theo mặc định, thư viện statsmodels phù hợp với một dòng trên tập dữ liệu đi qua điểm gốc. Nhưng để có phần chặn, bạn cần sử dụng thủ công thuộc tính `add_constant` của statsmodels. Và khi bạn đã thêm hằng số vào tập dữ liệu `X_train` của mình, bạn có thể tiếp tục và điều chỉnh đường hồi quy bằng thuộc tính OLS (Bình phương nhỏ nhất thông thường) của mô hình thống kê.

Gọi hàm `add_constant` để thêm chặn (intercept)

```
X_train_sm = sm.add_constant(X_train)
```

Gọi hàm OLS và fit để tìm ra regression line

```
lr = sm.OLS(y_train, X_train_sm).fit()
```

In các tham số, tức là phần chặn và độ dốc của đường hồi quy

```
lr.params
```

Kết quả tìm được sẽ là:  $\text{Sales} = 6.948 + 0.054 \times \text{TV}$

Vẽ biểu đồ scatter và line để thấy rõ kết quả

```
plt.scatter(X_train, y_train)
```

```
plt.plot(X_train, 6.948 + 0.054*X_train, 'r')
```

```
plt.show()
```

```
'''
```

Nếu các điểm dữ liệu phân bố tập trung xung quanh đường tung độ 0 và có xu hướng tạo thành một đường thẳng, giả định liên hệ tuyến tính không bị vi phạm nghĩa là chấp nhận giả thiết.



Cách bố trí của điểm dữ liệu trên đồ thị scatter sẽ tùy thuộc vào bản chất biến phụ thuộc, khi đánh giá, chúng ta cần nhìn tổng quát xu hướng của đám mây điểm dữ liệu.

'''

## ## 5. Đánh giá mô hình

Chúng ta cần kiểm tra xem các số hạng sai số cũng có phân phối chuẩn hay không (thực tế là một trong những giả định chính của hồi quy tuyến tính), chúng ta hãy vẽ biểu đồ của các số hạng sai số và xem nó trông như thế nào.

### ### 5.1. Đánh giá trên tập train

Gọi hàm predict trên tập train, sau đó tìm độ chênh lệch giữa kết quả dự đoán trên tập train và kết quả thực tế.

```
y_train_pred = lr.predict(X_train_sm)
res = (y_train - y_train_pred)
print(res)
```

Vẽ biểu đồ để thấy rõ hơn

```
fig = plt.figure()
sns.displot(res, bins = 15)
fig.suptitle('Error Terms', fontsize = 15)      # Plot heading
plt.xlabel('y_train - y_train_pred', fontsize = 15)  # X-label
plt.show()
'''
```

Đối với biểu đồ Histogram, nếu giá trị trung bình Mean gần bằng 0, độ lệch chuẩn Std. Dev gần bằng 1,

các cột giá trị phần dư phân bố theo dạng hình chuông, ta có thể khẳng định phân phối là xấp xỉ chuẩn, giả định phân phối chuẩn của phần dư không bị vi phạm. Cụ thể trong ảnh trên,

Mean = 5.74E-15 =  $5.74 \times 10^{-15}$  = 0.00000... gần bằng 0, độ lệch chuẩn là 0.991 gần bằng 1.

Như vậy có thể nói, phân phối phần dư xấp xỉ chuẩn, giả định phân phối chuẩn của phần dư không bị vi phạm.

'''

### ### 5.2. Đánh giá trên tập test

Làm tương tự trên tập train

# Add a constant to X\_test

X\_test\_sm = sm.add\_constant(X\_test)

# Predict the y values corresponding to X\_test\_sm

y\_pred = lr.predict(X\_test\_sm)

y\_pred.head()

Khai báo thư viện để dùng mean\_squared\_error và r2\_score

from sklearn.metrics import mean\_squared\_error

from sklearn.metrics import r2\_score

Đánh giá qua mean\_squared\_error

np.sqrt(mean\_squared\_error(y\_test, y\_pred))

Đánh giá qua r2\_score

r\_squared = r2\_score(y\_test, y\_pred)

r\_squared

Vẽ biểu đồ để xem mô hình khớp với dữ liệu thực tế thế nào

plt.scatter(X\_test, y\_test)

plt.plot(X\_test, 6.948 + 0.054 \* X\_test, 'r')

plt.show()

## 2) LAB 2 HỒI QUY TUYẾN TÍNH

Dữ liệu tuyensinhdaihoc

import pandas as pd

```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

```
#Đọc file dulieutuyensin
df = pd.read_csv('../EDA/dulieuxettuyendaihoc.csv',header=0,delimiter=',',encoding='utf-8')
```

```
'''
Phân tích hồi quy tuyến tính
Mục đích: Phân tích tác động hay ảnh hưởng giữa các yếu tố đến mục tiêu (thường dùng
cho các biến (yếu tố) định lượng)
Thường vẽ biểu đồ Scatter để khám phá mối tương quan tuyến tính trước khi khám phá
quan hệ hồi quy tuyến tính
PHƯƠNG PHÁP
1. Xác định biến độc lập (yếu tố) và biến phụ thuộc (mục tiêu)
2. Ghi ra phương trình hồi quy tuyến tính tổng quát  $y = f(x)$ 
3. Chạy dữ liệu mô hình
4. Đọc các giá trị quan trọng và kết luận
5. Dự báo giá trị biến phụ thuộc khi biết trước giá trị biến độc lập
'''
```

```
# Hãy cho biết sự ảnh hưởng của điểm học kì 1 năm lớp 12 đến điểm học kì 2 năm lớp 12
#import statsmodels.api as sm
#pip install statsmodels –cài đặt thư viện này để sử dụng thư viện #statsmodels.api
import statsmodels.api as sm
#linear regression
'''
1. Biến độc lập: học kì 1 (T5)
   Biến phụ thuộc : học kì 2 (T6)
2.  $T6 = f(T5)$ 
    $= A_0 + A_1 * T5 + \text{epsilon}$ 
3. Chạy mô hình
4. Đọc và hiểu kết quả
'''
# adding a constant
```

```
X_with_constant = sm.add_constant(df[["T5"]].values)
y = df[["T6"]].values
# performing the regression
result = sm.OLS(y,X_with_constant).fit()
# Result of statsmodels
print(result.summary())
```

```
'''
Căn cứ vào điểm số T5 sẽ giải thích được 60% sự thay đổi của T6 (Adj. R-squared)
Prob (F-statistic) < 0.05: cho biết mô hình có khả năng phù hợp cho tổng thể [nó là p-
value]
const (2,11) chính là A0
x1 là A1
=>  $T6 = 2.113 + 0.7182 * T5$ 
 $P > |t| = 0.000$  rất nhỏ < 5% => x1 tương ứng với T5 ( $x' = T5$ ) => biến T5 có ý nghĩa
thống kê trong phương trình này hay nói T5 có
ý nghĩa tham gia đánh giá tác động tới biến T6
'''
'''
Bước 5: Giả sử  $T5 = 7.5$ , dự báo  $T6 = 7.499$ 
'''
```

```
# Khám phá sự ảnh hưởng của T6 đến điểm thi LOGIC
# adding a constant
X_with_constant = sm.add_constant(df[["T6"]].values)
y = df[["LOGIC"]].values
# performing the regression
result = sm.OLS(y,X_with_constant).fit()
# Result of statsmodels
print(result.summary())
'''
Adj. R-squared = 8%: quá ít, không thể giải thích cho điểm LOGIC, nói cách khác k dựa
T6 để gthích dc
Prob (F-statistic) = 0.00230 => phù hợp, bé hơn 0.05, có ý nghĩa thống kê
 $LOGIC = 2.6287 + 0.2344 * T6$ 
 $P > |t| = 0.000 < 0.05$ : T6 có ý nghĩa thống kê
'''
```

```
Giả sử T6 = 7.0
=> LOGIC = 4.2695
'''
```

```
# Khám phá sự ảnh hưởng của T6 đến điểm thi LOGIC
# adding a constant
X_with_constant = sm.add_constant(df[["T6"]].values)
y = df[["TOANLOGICPHANTICH"]].values
# performing the regression
result = sm.OLS(y,X_with_constant).fit()
# Result of statsmodels
print(result.summary())
'''

Adj. R-squared = 8%: quá ít, không thể giải thích cho điểm LOGIC, nói cách khác k dựa
T6 để gthích dc
Prob (F-statistic) = 0.00230 => phù hợp, bé hơn 0.05, có ý nghĩa thống kê
LOGIC = 2.6287 + 0.2344 * T6
P>|t| = 0.000 < 0.05: T6 có ý nghĩa thống kê
Giả sử T6 = 7.0
=> LOGIC = 4.2695
'''
```

```
X_with_constant = sm.add_constant(df[["T5","T6"]].values)
y = df[["TOANLOGICPHANTICH"]].values
# performing the regression
result = sm.OLS(y,X_with_constant).fit()
# Result of statsmodels
print(result.summary())
'''

1. Độc lập T5,T6
phụ thuộc Logic
2. Logic f(T5,T6)
Logic = A0 + A1*T5 + A2*T6 * epsilon = 2.7072 - 0.0913*T5 + 0.3115 * T6 + epsilon
'''
```

```
df = df
[['T5','T6','GT','DT','KV','KT','NGONNGU','TOANLOGICPHANTICH','GIAIQUYETVANDE','NGAYTHI','DINHHUONGNGHENGHIEP']]
df.rename(columns={'TOANLOGICPHANTICH':'LOGIC',
                    'GIAIQUYETVANDE':'UNGXU',
                    'DINHHUONGNGHENGHIEP':'HUONGNGHIEP'},inplace=True)
```

# Hãy phân tích sự ảnh hưởng của điểm toán học kì 1,2 năm lớp 12 đến điểm NGONNGU

```
X_with_constant = sm.add_constant(df[["T5","T6"]].values)
```

```
y = df[["NGONNGU"]].values
```

# performing the regression

```
result = sm.OLS(y,X_with_constant).fit()
```

# Result of statsmodels

```
print(result.summary())
```

'''

Adj. R-squared = 1% : quá ít, k gthích dc gì

Prob (F-statistic): 7%

'''

# Đánh giá mức độ tác động giữa các yếu tố đến 1 đối tượng bằng phân tích hồi quy tuyến tính

# Hãy cho biết mức độ tác động của T5, T6 (độc lập) đến điểm LOGIC (phụ thuộc)

# adding a constant

```
X = df[["T5","T6"]].values
```

```
y = df[["LOGIC"]].values
```

# performing the regresssion

```
result = sm.OLS(y,X).fit()
```

# result of statsmodels

```
print(result.summary())
```

'''

$x_2 = |0.5934| \Rightarrow$  T6 tác động mạnh hơn so với T5 ( $x_1 = |0.0063|$ )

vì  $x_2$  dương nên tác động tích cực, còn âm mới tác động tiêu cực (nghịch biến)

'''

# Đánh giá mức độ tác động giữa các yếu tố đến 1 đối tượng bằng phân tích hồi quy tuyến tính

```
# Hãy cho biết mức độ tác động của T5, T6 đến điểm UNGXU
# adding a constant
X = df[["T5", "T6"]].values
y = df[["UNGXU"]].values
# performing the regresssion
result = sm.OLS(y, X).fit()
# result of statsmodels
print(result.summary())
'''
x2 = |0.5318| => T6 tác động mạnh hơn so với T5 (x1 = |0.1519|)
vì x2 dương nên tác động tích cực (đồng biến), còn âm mới tác động tiêu cực (nghịch biến)
'''
```

## BÀI TOÁN PHÂN LỚP – CLASSIFICATION - CLUSTERING

### 1) BÀI 1

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report
```

```
1 import pandas as pd
2 import numpy as np
3 import seaborn as sns
4 import matplotlib.pyplot as plt
5 from sklearn.linear_model import LogisticRegression
6 from sklearn.metrics import confusion_matrix, classification_report
```

```
1 # Dựa vào các điểm thi đánh giá năng lực, hãy phân đoán xem sinh viên có định hướng
2 # nghề nghiệp hay
3 # chưa được định hướng nghề nghiệp
4 # Input: Logic, ngonngu, ungxu
5 # Output: Dinhhuong
6 # Dinhhuong = f(logic, ngonngu, ungxu)
7 ...
8 Ý TƯỞNG TÍNH  $Z = A0 + A1 + LOGIC + A2 + MN + A3 + UX + E$ 
9 Từ đó tính  $f(x) = 1/(1+e^{-z})$  thuộc  $[0,1]$ 
10 Nếu mà  $f(x) > 0.5$  thì YES
11 Nếu mà  $f(x) < 0.5$  thì NO
12 ...
```

```
1 df = pd.read_csv('dulieuxettuyendaihoc.csv', header=0, delimiter=',', encoding='utf-8')
```



```
1 df = df [['NGONNGU','TOANLOGICPHANTICH','GIAIQUYETVANDE','DINHUUONGNGHENGHIEP']]
2 df.rename(columns={'TOANLOGICPHANTICH':'LOGIC',
3                    'GIAIQUYETVANDE':'UNGXU',
4                    'DINHUUONGNGHENGHIEP':'DINHUUONG'},inplace=True)
5 df.head(5)
```

	NGONNGU	LOGIC	UNGXU	DINHUUONG
0	3.25	3.25	4.50	No
1	6.00	4.00	3.50	Yes
2	5.00	6.75	4.00	No
3	4.25	4.25	5.25	No
4	4.25	4.50	5.00	No

```
1 dinhhuong = pd.get_dummies(df['DINHUUONG'],drop_first=True)
2 df.drop(['DINHUUONG'],axis=1,inplace=True)
3 df = pd.concat([df,dinhhuong],axis=1)
4 df.head(5)
5 # Biến đổi biến dinhhuong dưới dạng 0 1
6 # Do biến DINHUUONG là dạng categorical values (định tính) nên cần biến đổi về dạng số
7 # (số hóa) trong bài này biến DINHUUONG
8 # chỉ có 2 giá trị Yes/No nên ta dễ dàng biến đổi tương đương 0/1 nhưng nếu như số
9 # lượng giá trị nhiều hơn 2 thì ta cần dùng
10 # phương pháp One hot encoding để biến đổi về dạng 1 vector tương ứng
```

	NGONNGU	LOGIC	UNGXU	Yes
0	3.25	3.25	4.50	False
1	6.00	4.00	3.50	True
2	5.00	6.75	4.00	False
3	4.25	4.25	5.25	False
4	4.25	4.50	5.00	False

```
1 X = df[['NGONNGU','LOGIC','UNGXU']].values # input
2 y = df[['Yes']].values #output
```

```
1 from sklearn.model_selection import train_test_split
2 X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.30,random_state=1)
```

```

1 from sklearn.model_selection import train_test_split
2 X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.30,random_state=1)

1 from sklearn.linear_model import LogisticRegression
2 classifier = LogisticRegression()
3 classifier.fit(X_train,y_train)

c:\Users\ADMIN\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\utils\validation.py:108: UserWarning: Column 1 of y is not unique.
y = column_or_1d(y, warn=True)

LogisticRegression
LogisticRegression()

1 classifier.coef_
2 # A1 (ngonngu): 0.27976181
3 # A2 (logic): 0.05558809
4 # A3 (ungxu): -0.09715109

array([[ 0.27976181,  0.05558809, -0.09715109]])

```

```

1 classifier.intercept_
2 # A0 = -1.15103264

array([-1.15103264])

1 '''
2 Phương trình phân lớp
3
4  $p(x) = 1/(1+e^{(-z)})$ 
5 Yes = threshold(p(x) so với 0.5)
6 '''

```

```

1 # Với điểm ngôn ngữ là 7.5, logic là 6.5 và ứng xử là 7.0 thì sinh viên có định hướng
2 # hay không
3 # Đáp án: Yes: ?
4 # Z = 0.628 (đoạn này thay số trên ptrình)
5 Z = classifier.intercept_ + classifier.coef_[0][0]*7.5 + classifier.coef_[0][1]*6.5
6 + classifier.coef_[0][2]*7.0
7 print(Z)

```

[0.62844585]

```

1 p =1/(1+np.exp(-Z))
2 print(p)
3
4 if p>0.5:
5     print('Yes')
6 else:
7     print('No')

```

[0.65213698]  
Yes

```

1 # Với điểm ngôn ngữ là 7.5, logic là 6.5 và ứng xử là 5.0 thì sinh viên có định hướng
2 # hay không
3 # Đáp án: Yes: ?
4 # Z = 0.628 (đoạn này thay số trên ptrình)
5 Z = classifier.intercept_ + classifier.coef_[0][0]*7.5 + classifier.coef_[0][1]*6.5 +
6 classifier.coef_[0][2]*5.0
7 print(Z)

```

[0.82274803]

```

1 p =1/(1+np.exp(-Z))
2 print(p)
3
4 if p>0.5:
5     print('Yes')
6 else:
7     print('No')

```

[0.69481936]  
..

```

1 def classifier_cal(a,b,c):
2     Z = classifier.intercept_ + classifier.coef_[0][0]*a + classifier.coef_[0][1]*b +
3       classifier.coef_[0][2]*c
4     return 1/(1+np.exp(-Z))
5 p=classifier_cal(7.6,6.5,5.0)
6 print(p)
7
8 if p>0.5:
9     print('Yes')
10 else:
11     print('No')

```

[0.70071904]  
Yes

```

1 # Defauly threshold is 0.5
2 y_pred = classifier.predict(X_test)
3 y_pred
4

```

array([False, True, False, False, False, False, False, False, False,  
 False, False, False, True, False, False, False, False, False,  
 False, False, True, False, True, False, False, False, False,  
 True, False, False])

```
1 '''
2 hanging threshold and predicting
3 print('Prediction with threshold 0.9: ')
4 y_pred_new_threshold = (classifier.predict_proba(X_test)[:,-1] >= 0.9).astype(int)
5 print(y_pred_new_threshold)
6 '''
7 print('Prediction with threshold 0.9: ')
8 y_pred_new_threshold = (classifier.predict_proba(X_test)[:,-1] >= 0.9).astype(int)
9 print(y_pred_new_threshold)
```

Prediction with threshold 0.9:

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
```

```
1 result = pd.DataFrame({'Actual': y_test.flatten().astype(int),
2 'Predicted': y_pred.astype(int)})
3 result.head(10)
```

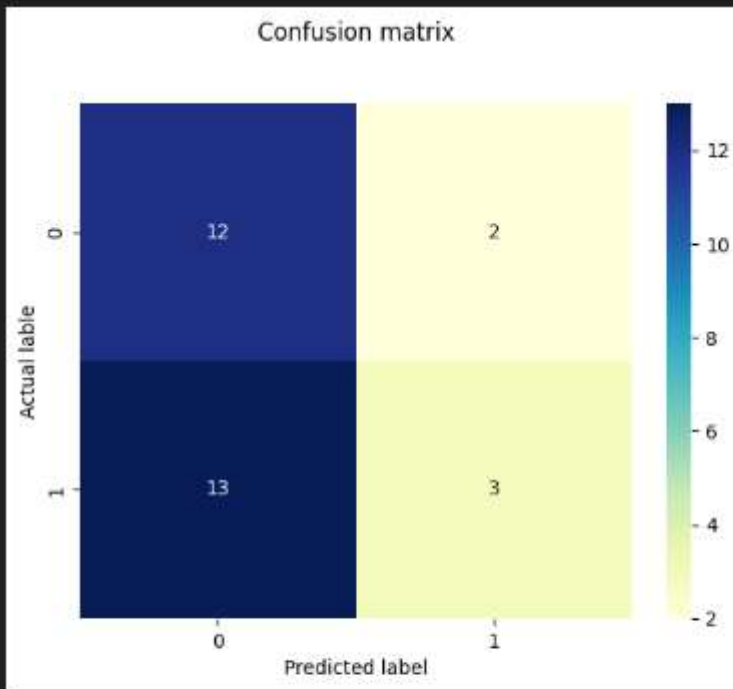
	Actual	Predicted
0	1	0
1	1	1
2	1	0
3	0	0
4	1	0

```
1 from sklearn.metrics import confusion_matrix
2 cf_matrix = confusion_matrix(y_test, y_pred)
3 cf_matrix
4 ...
5 [[TN, FP],
6  | [FN, TP]]
7 ...
```

```
'\n[[TN, FP],\n [FN, TP]]\n'
```

```
1 sns.heatmap(pd.DataFrame(cf_matrix), annot= True, cmap="YlGnBu",fmt='g')
2 plt.title('Confusion matrix', y=1.1)
3 plt.ylabel('Actual lable')
4 plt.xlabel('Predicted label')
```

```
Text(0.5, 23.52222222222222, 'Predicted label')
```



```
1 from sklearn.metrics import accuracy_score
2 accuracy_score(y_test,y_pred)
```

0.5

```
1 from sklearn.metrics import accuracy_score
2 accuracy_score(y_test,y_pred)
```

0.5

```
1 target_names = ['Not oriented', 'Oriented']
2 print(classification_report(y_test, y_pred, target_names=target_names))
```

	precision	recall	f1-score	support
Not oriented	0.48	0.86	0.62	14
Oriented	0.60	0.19	0.29	16
accuracy			0.50	30
macro avg	0.54	0.52	0.45	30
weighted avg	0.54	0.50	0.44	30

## 2) BÀI 2

'''

Dựa vào điểm thi xét tuyển trường IUH cần tuyển những học sinh có điểm thi ưu tiên NGONNGU tốt và LOGIC vừa phải để phù hợp với nhu cầu đào tạo. Hãy đưa ra đề xuất các nhóm học sinh có kết quả phù hợp và căn cứ vào chỉ tiêu (số lượng sinh viên)

Vấn đề đặt ra: Thế nào là tốt? Thế nào là kha khá? -- theo dữ liệu tuyển sinh này

--> Tiêu chuẩn chọn lựa mập mờ => Giải pháp

'''

### # Giải thuật gom cụm (Clustering)

'''

Đây là giải thuật có input đầu vào nhưng không có sẵn output đầu ra trong tập dữ liệu lịch sử

Do đó ta gọi là nhóm kỹ thuật un-supervised (học không giám sát)

Để giải quyết nhóm kỹ thuật này ta cần các yếu tố như sau

1. Dữ liệu đầu vào
2. Phải có metric (độ đo) để xác định các phần tử sẽ thuộc về đâu
3. Cần cung cấp trước số lượng cụm (k) được tạo ra

Hệ quả:

1. Các phần tử thuộc về cùng 1 cụm thì có mức độ gần gũi (theo độ đo đặt ra) hơn so với phần tử ở cụm khác
2. Mỗi cụm sẽ có một đại diện được gọi là trung tâm

--> Trong các giải thuật về dạng toán này người ta thường dùng độ đo Euclide để đo lường mức độ gần nhau của các phần tử

--> Giải thuật điển hình là giải thuật K-Means

'''

```

1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import seaborn as sb

```

```

1 df = pd.read_csv('../EDA/dulieuxettuyendaihoc.csv', header=0, delimiter=',',
2 encoding='utf-8')
3 df.rename(columns={'TOANLOGICPHANTICH': 'LOGIC', 'GIAIQUYETVANDE': 'UNGXU',
4 'DINHUUONGNGHENGHIEP': 'HUONGNGHIEP'}, inplace=True)
5 df

```

	MSSV	T1	T2	T3	T4	T5	T6	GT	DT	KV	NGONNGU	LOGIC	UNGXU	KT	NGAYTHI	HU
0	SV001	7.2	8.4	7.4	7.2	7.4	6.9	F	NaN	2NT	3.25	3.25	4.50	A1	12/7/2018	
1	SV002	5.4	6.3	4.3	4.9	3.0	4.0	M	NaN	1	6.00	4.00	3.50	C	12/7/2018	
2	SV003	5.6	5.0	2.8	6.1	4.8	5.7	M	NaN	1	5.00	6.75	4.00	C	12/7/2018	
3	SV004	6.6	5.1	5.9	4.1	6.1	7.4	M	NaN	1	4.25	4.25	5.25	D1	12/7/2018	
4	SV005	6.0	5.4	7.6	4.4	6.8	8.0	M	NaN	2NT	4.25	4.50	5.00	A	12/7/2018	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
95	SV0096	8.6	8.9	7.7	6.7	7.9	6.6	F	NaN	1	5.25	1.50	6.25	C	7/20/2022	
96	SV0097	3.7	4.1	3.1	3.4	5.5	5.7	F	NaN	1	5.25	3.75	4.75	C	7/20/2022	
97	SV0098	8.8	9.5	9.5	8.5	9.0	8.5	M	NaN	2NT	7.00	8.00	4.00	C	7/20/2022	
98	SV0099	2.7	2.8	6.2	5.2	4.1	4.3	M	NaN	1	5.00	3.50	5.50	C	7/20/2022	



```
1 data = df[['NGONNGU','LOGIC']]
```

```
1 data.head(5)
```

	NGONNGU	LOGIC
0	3.25	3.25
1	6.00	4.00
2	5.00	6.75
3	4.25	4.25
4	4.25	4.50

```
1 # độ đo Euclid
2 d = np.sqrt((3.25-6.0)**2+(3.25-4.0)**2)
3 d
```

```
2.850438562747845
```

```
1 from sklearn.cluster import KMeans
```

```

1 kmeans = KMeans(n_clusters=4).fit(data)

c:\Users\Simming\AppData\Local\Programs\Python\Python311\Lib\site-packages\sk
warnings.warn(

1 print(kmeans.cluster_centers_) # Số tâm
2 print(kmeans.inertia_) # Độ đo đánh giá
3 print(kmeans.n_iter_) # Sẽ lặp bao nhiêu lần để tìm giá trị điển hình t
4 print(kmeans.labels_[:])
5 # 100 sinh viên được chia tối đa là 100 cụm
6 # Tâm là 1 đại lượng được tính toán bằng thuật toán.

[[3.71621622  4.39864865]
 [6.05555556  6.08333333]
 [5.06521739  3.23913043]
 [2.11290323  4.30645161]]
86.70542983376083
5
[0 2 1 0 0 3 1 0 0 0 0 1 3 3 0 2 0 0 0 0 3 0 2 1 2 2 2 1 0 3 2 0 0 3 3 3 2
 3 3 3 2 0 2 3 0 3 2 1 2 0 3 3 0 3 2 0 1 3 3 0 3 0 2 3 0 3 0 0 0 3 3 0 3 3
 2 2 0 0 0 3 2 3 0 3 1 0 3 3 0 3 2 0 0 0 2 2 2 1 2 2]

```

```

1 def euclid(a1,a2,b1,b2):
2     d = np.sqrt((a1-b1)**2+(a2-b2)**2)
3     return d
4 euclid(3.25,3.25,2.11,4.31)

```

1.5566630977832037

```

1 # metric: thuaajt toans

```

```

1 kmeans1 = KMeans(n_clusters=6).fit(data)

```

c:\Users\Simming\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\cluster\\_kmeans.py:100: UserWarning: Number of clusters (6) is less than the number of data points (10).  
warnings.warn(

```

1 print(kmeans1.cluster_centers_) # Số tâm
2 print(kmeans1.inertia_) # Độ đo đánh giá
3 print(kmeans1.n_iter_) # Sẽ lặp bao nhiêu lần để tìm giá trị điển hình tối ưu
4 print(kmeans1.labels_[:])

```

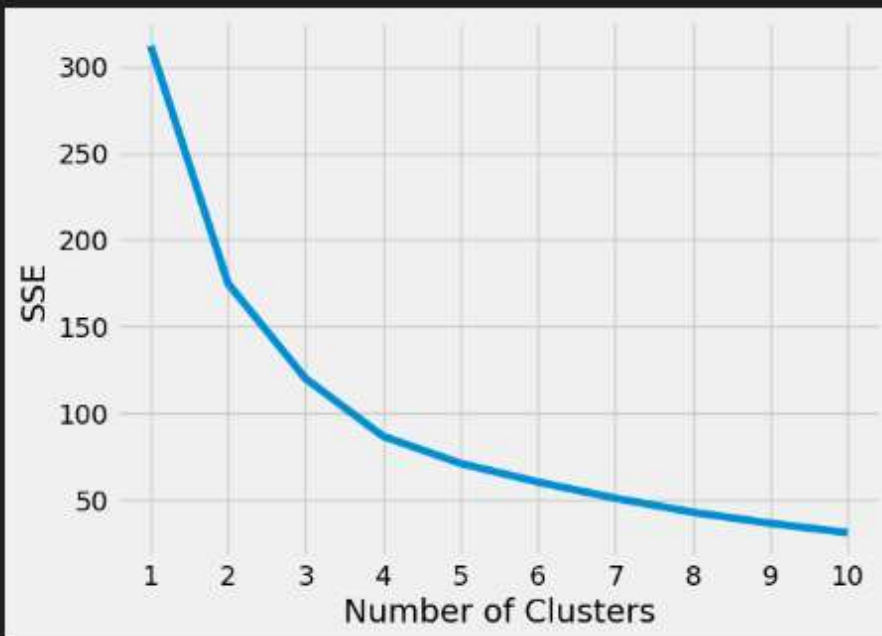
```

[[2.         4.65909091]
 [3.85      4.73       ]
 [6.         4.67857143]

```

```
[6.          4.67857143]
[2.96428571  3.55952381]
[4.975       3.1375      ]
[6.          6.9         ]]
60.16815205627705
5
[3 2 5 1 1 0 5 1 3 1 3 5 0 3 1 3 3 1 1 3 0 3 4 2 4 4 2 2 1 3 4 1 1 0 0 0 4
 0 3 3 4 1 4 0 1 0 4 5 4 1 0 3 3 0 4 1 2 0 0 3 3 3 4 3 4 0 1 1 1 3 0 1 0 0
 4 4 3 3 1 3 4 0 1 0 2 1 0 0 1 0 2 1 1 1 4 4 4 5 4 4]
```

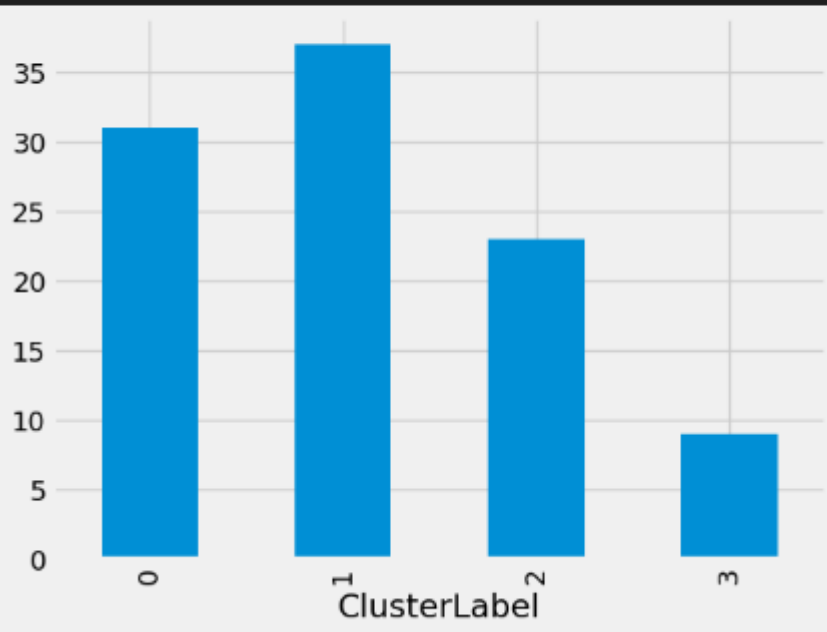
```
1 # Chọn số cụm tốt nhất
2 # url: https://realpython.com/k-means-clustering-python
3 # https://dataofish.com/k-means/clustering-python
4 kmean_kwargs = {"init": "random", "n_init":10,"max_iter":300,"random_state":42}
5 sse = []
6 for i in range(1,11):
7     kmeans = KMeans(n_clusters=i,**kmean_kwargs)
8     kmeans.fit(data)
9     sse.append(kmeans.inertia_)
10 plt.style.use("fivethirtyeight")
11 plt.plot(range(1,11),sse)
12 plt.xticks(range(1,11))
13 plt.xlabel("Number of Clusters")
14 plt.ylabel("SSE")
15 plt.show()
```



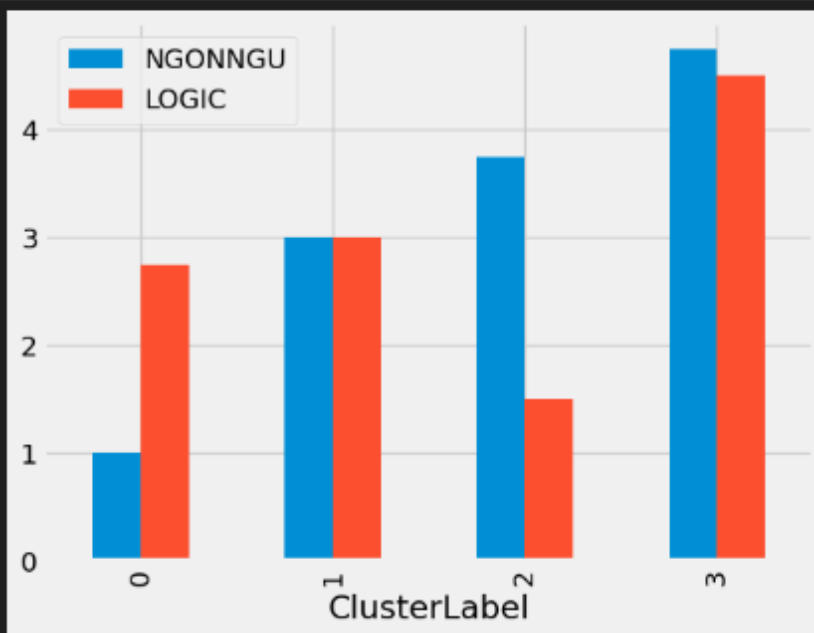
```
1 # Cách 2: Dùng phương pháp độ đo ElBow để tìm số cụm là tốt nhất
2 from kneed import KneeLocator
3 kl = KneeLocator(range(1,11),sse,curve="convex",direction="decreasing")
4 print(kl.elbow)
```

```
1 # Xây dựng ứng dụng hiển thị danh sách sinh viên và nhân (label)
2 # cụm sinh viên thuộc vào
3 kmeans = KMeans(n_clusters=4).fit(data)
4 data['ClusterLabel']= kmeans.labels_[:]
5 print(data[['NGONNGU','LOGIC','Cluster']])
6
```

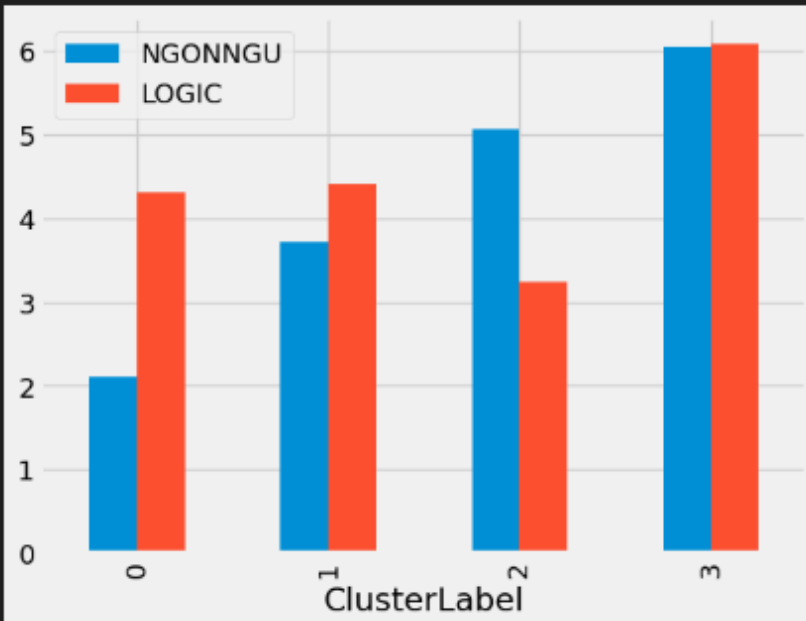
```
1 gr_data = data.groupby(['ClusterLabel']).size()
2 gr_data.plot.bar()
3 plt.show()
```



```
1 gr_data_min = data.groupby(['ClusterLabel'])[['NGONNGU', 'LOGIC']].min()
2 gr_data_min.plot.bar()
3 plt.show()
```



```
1 gr_data_mean = data.groupby(['ClusterLabel'])[['NGONNGU', 'LOGIC']].mean()
2 gr_data_mean.plot(kind='bar')
3 plt.show()
4
```



# Sinh viên khám phá các cụm cho tập điểm số (NGONNGU, LOGIC, UNGXU)

### 3) BÀI 3

'''

#### PCA

1. Principal component analysis là gì?

Phương pháp dựa trên phương pháp tuyến tính

2. Mục đích của PCA

- Khám phá được tương quan tuyến tính của đa biến

- Giảm thiểu số lượng các cột dữ liệu (biến số) nhằm giúp chỉ quan tâm đến các cột có tác động mạnh với nhau.

-> Giảm thiểu số chiều dữ liệu. Hay nói cách khác thay vì phải phân tích trên toàn bộ các cột dữ liệu thì ta sẽ

giảm thiểu nhằm phân tích trên các cột trọng tâm.

3. PCA thuộc nhóm học máy nào?

## Un-supervised

4. Ý nghĩa biểu diễn tập dữ liệu bằng các components (PC)
5. Hiểu được đại lượng proportion of variance: Biểu diễn % sự đa dạng của tập dữ liệu còn giữ được sau khi dùng các component để biểu diễn
6. Các xác định bao nhiêu component để biểu diễn dữ liệu là tốt nhất: Dùng Scree Chart (Elbow test)
7. Đọc và hiểu thông tin dữ liệu trên từng component: Kiểm tra xem mỗi component đang đại diện cho những cột dữ liệu nào trong tập dữ liệu ban đầu

'''

```
1 from sklearn.decomposition import PCA
2 from sklearn.preprocessing import StandardScaler
3 from bioinfokit.analys import get_data
4 import numpy as np
5 import pandas as pd
6
7 df = get_data('gexp').data
8 df.head(2)
```

	A	B	C	D	E	F
0	4.50570	3.26036	-1.24940	8.89807	8.05955	-0.842803
1	3.50856	1.66079	-1.85668	-2.57336	-1.37370	1.196000

```
1 df_st = StandardScaler().fit_transform(df)
2 pd.DataFrame(df_st, columns=df.columns).head(2)
```

	A	B	C	D	E	F
0	0.619654	0.448280	-0.240867	2.457058	2.304732	-0.331489
1	0.342286	-0.041499	-0.428652	-1.214732	-0.877151	0.474930



```

1 pca_out = PCA().fit(df_st)
2 pca_out.explained_variance_ratio_

array([0.2978742 , 0.27481252, 0.23181442, 0.19291638, 0.00144353,
       0.00113895])

1 np.cumsum(pca_out.explained_variance_ratio_)
2

array([0.2978742 , 0.57268672, 0.80450114, 0.99741752, 0.99886105,
       1.          ])

```

#### 4) BÀI 4

```

1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5 %matplotlib inline

```

Python

```

1 from sklearn.datasets import load_breast_cancer
2 # Đọc dữ liệu từ sklearn
3 cancer_set = load_breast_cancer()
4
5 # Chuyển thành DataFrame
6 cancer_data = pd.DataFrame(data=cancer_set['data'], columns=cancer_set['feature_names'])
7
8 cancer_data.head()

```

Python

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fracti dimensio
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.0787
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.0566
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.0599
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.0974
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.0588

```
1 cancer_data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 30 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   mean radius                          569 non-null    float64
1   mean texture                         569 non-null    float64
2   mean perimeter                      569 non-null    float64
3   mean area                          569 non-null    float64
4   mean smoothness                    569 non-null    float64
5   mean compactness                   569 non-null    float64
6   mean concavity                     569 non-null    float64
7   mean concave points                 569 non-null    float64
8   mean symmetry                      569 non-null    float64
9   mean fractal dimension              569 non-null    float64
10  radius error                        569 non-null    float64
11  texture error                      569 non-null    float64
12  perimeter error                    569 non-null    float64
13  area error                        569 non-null    float64
14  smoothness error                   569 non-null    float64
15  compactness error                  569 non-null    float64
16  concavity error                    569 non-null    float64
17  concave points error               569 non-null    float64
18  symmetry error                     569 non-null    float64
19  compactness error                  569 non-null    float64
20  concavity error                    569 non-null    float64
21  concave points error               569 non-null    float64
22  symmetry error                     569 non-null    float64
23  compactness error                  569 non-null    float64
24  concavity error                    569 non-null    float64
25  concave points error               569 non-null    float64
26  symmetry error                     569 non-null    float64
27  compactness error                  569 non-null    float64
28  concavity error                    569 non-null    float64
29  concave points error               569 non-null    float64
30  symmetry error                     569 non-null    float64
```

```

15 compactness error      569 non-null    float64
16 concavity error        569 non-null    float64
17 concave points error   569 non-null    float64
18 symmetry error         569 non-null    float64
19 fractal dimension error 569 non-null    float64
...
28 worst symmetry         569 non-null    float64
29 worst fractal dimension 569 non-null    float64
dtypes: float64(30)
memory usage: 133.5 KB

```

Output is truncated. View as a [scrollable element](#) or open in a [text editor](#). Adjust cell output [settings](#)...

```

1 from sklearn.preprocessing import StandardScaler
2 scaler = StandardScaler()
3
4 # Fit vào dữ liệu
5 scaler.fit(cancer_data)
6
7 # Thực hiện transform scale
8 scale_cancer_data = scaler.transform(cancer_data)

```

1 scale\_cancer\_data

```

array([[ 1.09706398, -2.07333501,  1.26993369, ...,  2.29607613,
         2.75062224,  1.93701461],
       [ 1.82982061, -0.35363241,  1.68595471, ...,  1.0870843 ,
        -0.24388967,  0.28118999],
       [ 1.57988811,  0.45618695,  1.56650313, ...,  1.95500035,
         1.152255  ,  0.20139121],
       ...,
       [ 0.70228425,  2.0455738 ,  0.67267578, ...,  0.41406869,
        -1.10454895, -0.31840916],
       [ 1.83834103,  2.33645719,  1.98252415, ...,  2.28998549,
         1.91908301,  2.21963528],
       [-1.80840125,  1.22179204, -1.81438851, ..., -1.74506282,
        -0.04813821, -0.75120669]])

```

```

1 from sklearn.decomposition import PCA
2
3 # Khởi tạo đối tượng PCA với số comp = 2
4 my_pca = PCA (n_components = 2 )
5
6 # Fit vào data
7 my_pca.fit(scale_cancer_data)
8
9 # Thực hiện transform
10 pca_scale_cancer_data = my_pca.transform(scale_cancer_data)
11

```

```

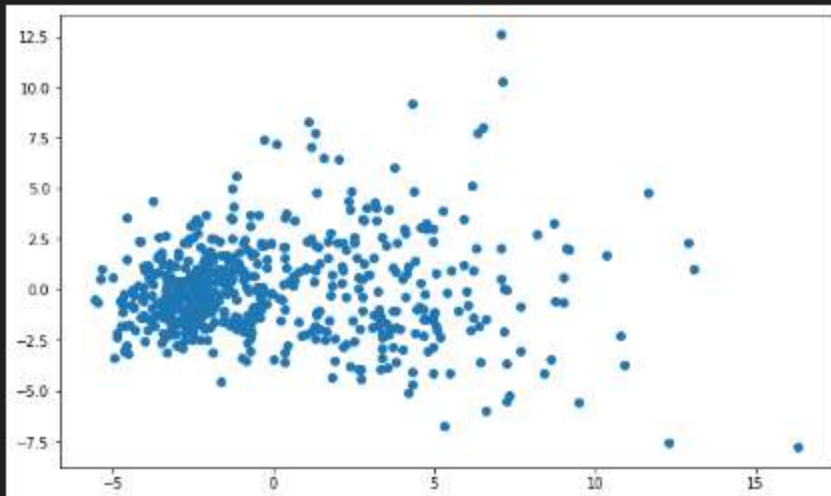
1 print("Dữ liệu gốc: ", scale_cancer_data.shape)
2 # Dữ liệu gốc: (569, 30)
3
4 print("Dữ liệu sau PCA:" , pca_scale_cancer_data.shape)
5 # Dữ liệu sau PCA: (569, 2)

```

Dữ liệu gốc: (569, 30)  
 Dữ liệu sau PCA: (569, 2)

```
1 plt.figure(figsize = (10,6))
2 # Thành phần comp số 1
3 pca_1 = pca_scale_cancer_data[:, 0]
4 # Thành phần comp số 2
5 pca_2 = pca_scale_cancer_data[:, 1]
6
7 # Vẽ đồ thị
8 plt.scatter(x=pca_1, y = pca_2)
```

<matplotlib.collections.PathCollection at 0x7faf41c56a10>

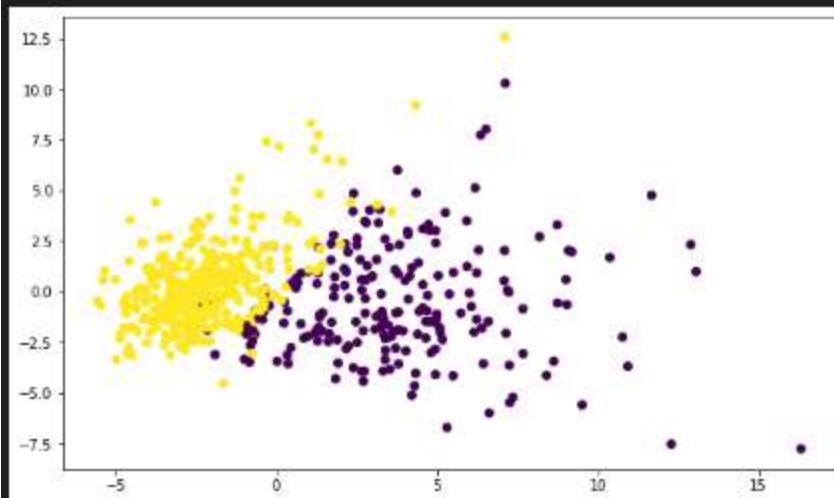


```
1 cancer_set['target']
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0,
       1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0,
       1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0,
       0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
       1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0,
       0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0,
       0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
       1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1,
       1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
```

```
1 plt.figure(figsize = (10,6))
2 # Thành phần comp số 1
3 pca_1 = pca_scale_cancer_data[:, 0]
4 # Thành phần comp số 2
5 pca_2 = pca_scale_cancer_data[:, 1]
6
7 # Vẽ đồ thị
8 plt.scatter(x=pca_1, y = pca_2, c = cancer_set['target'])
```

<matplotlib.collections.PathCollection at 0x7faf41bef150>



```
1 my_pca.components_
```

```
array([[ 0.21890244,  0.10372458,  0.22753729,  0.22099499,  0.14258969,
         0.23928535,  0.25840048,  0.26085376,  0.13816696,  0.06436335,
         0.20597878,  0.01742803,  0.21132592,  0.20286964,  0.01453145,
         0.17039345,  0.15358979,  0.1834174 ,  0.04249842,  0.10256832,
         0.22799663,  0.10446933,  0.23663968,  0.22487053,  0.12795256,
         0.21009588,  0.22876753,  0.25088597,  0.12290456,  0.13178394],
        [-0.23385713, -0.05970609, -0.21518136, -0.23107671,  0.18611302,
         0.15189161,  0.06016536, -0.0347675 ,  0.19034877,  0.36657547,
        -0.10555215,  0.08997968, -0.08945723, -0.15229263,  0.20443045,
         0.2327159 ,  0.19720728,  0.13032156,  0.183848 ,  0.28009203,
        -0.21986638, -0.0454673 , -0.19987843, -0.21935186,  0.17230435,
         0.14359317,  0.09796411, -0.00825724,  0.14188335,  0.27533947]])
```

```
1 pca_comp = pd.DataFrame(data=my_pca.components_, columns=cancer_data.columns)
2 pca_comp.head()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry
0	0.218902	0.103725	0.227537	0.220995	0.142590	0.239285	0.258400	0.260854	0.138167
1	-0.233857	-0.059706	-0.215181	-0.231077	0.186113	0.151892	0.060165	-0.034768	0.190349

## 5) BÀI 5

<https://www.machinelearningnuggets.com/logistic-regression/>

## 6) BÀI 6

### MACHINE LEARNING

```

1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns

```

```

1 df = pd.read_csv('dulieuxettuyendaihoc.csv', header=0, delimiter=',', encoding='utf-8')
2 df.rename(columns={'TOANLOGICPHANTICH': 'LOGIC', 'GIAIQUYETVANDE': 'UNGXU',
3 'DINHQUONGNGHENGHIEP': 'HUONGNGHIEP'}, inplace=True)
4 df

```

	MSSV	T1	T2	T3	T4	T5	T6	GT	DT	KV	NGONNGU	LOGIC	UNGXU	KT	NGAYTHI	HUON
0	SV001	7.2	8.4	7.4	7.2	7.4	6.9	F	NaN	2NT	3.25	3.25	4.50	A1	12/7/2018	
1	SV002	5.4	6.3	4.3	4.9	3.0	4.0	M	NaN	1	6.00	4.00	3.50	C	12/7/2018	
2	SV003	5.6	5.0	2.8	6.1	4.8	5.7	M	NaN	1	5.00	6.75	4.00	C	12/7/2018	
3	SV004	6.6	5.1	5.9	4.1	6.1	7.4	M	NaN	1	4.25	4.25	5.25	D1	12/7/2018	
4	SV005	6.0	5.4	7.6	4.4	6.8	8.0	M	NaN	2NT	4.25	4.50	5.00	A	12/7/2018	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
95	SV0096	8.6	8.9	7.7	6.7	7.9	6.6	F	NaN	1	5.25	1.50	6.25	C	7/20/2022	
96	SV0097	3.7	4.1	3.1	3.4	5.5	5.7	F	NaN	1	5.25	3.75	4.75	C	7/20/2022	
97	SV0098	8.8	9.5	9.5	8.5	9.0	8.5	M	NaN	2NT	7.00	8.00	4.00	C	7/20/2022	
98	SV0099	2.7	2.8	6.2	5.2	4.1	4.3	M	NaN	1	5.00	3.50	5.50	C	7/20/2022	
99	SV0100	4.1	4.4	6.0	4.3	5.6	5.1	M	NaN	2NT	5.25	2.50	4.25	C	7/20/2022	

## # Machine learning

'''

### Dẫn nhập machine learning

- Dựa trên dữ liệu lịch sử (Historical data) để xây dựng mô hình (Model)
- Input đi qua Model (mô hình toán học) sẽ tính toán cho ra Output

### Các bài toán phổ biến



1. Dự báo (Prediction)
2. Phân loại (Classification)
3. Gom cụm (Clustering)
4. Luật kết hợp (Association rules)

Các phương pháp phổ biến

1. supervised
2. un-supervised
3. semi-supervised learning

Quy trình xây dựng ứng dụng Machine Learning

'''

'''

Dẫn nhập hồi quy tuyến tính

Ứng dụng trong bài toán Prediction

Bài toán 1: Hãy dự báo điểm T6 dựa trên T5

input: T5

Output T6

$T6 = f(T5) \rightarrow T6 = A_0 + A_1.T5 + \text{epsilon}$

'''

```
1 df[['T5', 'T6']].head(5)
```

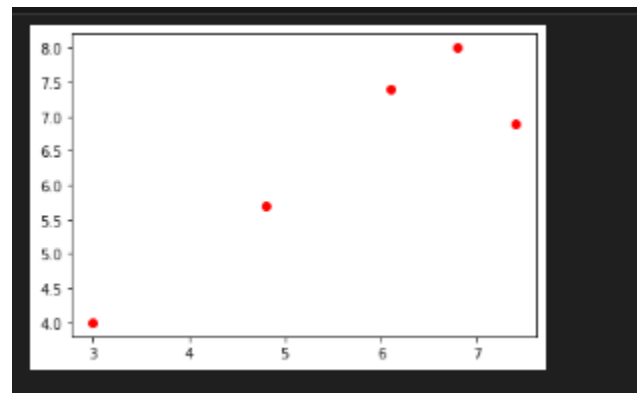
	T5	T6
0	7.4	6.9
1	3.0	4.0
2	4.8	5.7
3	6.1	7.4
4	6.8	8.0

```
1 df[['T5', 'T6']].corr()
```

	T5	T6
T5	1.000000	0.778683
T6	0.778683	1.000000

[+ Code](#) [+ Markdown](#)

```
1 plt.plot(df[['T5']][0:5], df[['T6']][0:5], 'ro')
2 plt.show()
```



'''

Lý thuyết về hồi quy tuyến tính

1. Phương pháp giải
2. Các Phương pháp giải tối ưu chi phí tính toán
3. DLS tối ưu sai số

'''

# Bài toán 2: Hãy dự báo điểm thi đánh giá năng lực phần LOGIC dựa trên điểm trung bình toán học kì 1 và học kì 2 năm lớp 12

'''

Input: T5 T6

Output: LOGIC

$LOGIC = f(T5, T6) = A_0 + A_1.T5 + A_2.T6 + \epsilon$

Xây dựng mô hình trong Machine Learning

Bước 1: Xác định bài toán

B2: Xác định input và output

B3: Xác định mô hình xử lý

B4: Chia tập dữ liệu ra làm 2 phần: training và testing theo tỉ lệ 90% và 10%

B5: Chạy mô hình

B6: Đánh giá mô hình

B7: Triển khai mô hình

'''

```

1 # Tập dữ liệu
2 X = df[['T5','T6']].values #input
3 y = df[['LOGIC']].values #output
4
5 #Chia tách ra training và testing
6 from sklearn.model_selection import train_test_split
7 X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.1,random_state=5)
8
9 #Xây dựng mô hình hồi quy tuyến tính đa biến
10 # LOGIC = A0 + A1*T5 + A2*T6 + epsilon
11 from sklearn import linear_model
12
13 model = linear_model.LinearRegression()
14 model.fit(X_train,y_train)
15
16 #To retrieve the intercept:
17 # intercept: hệ số tự do
18 print(model.intercept_)
19
20 #For retrieving the slope (cofficient):
21 print(model.coef_)
22 # A0, A1, A2 tương ứng với các số dưới

```

```

[2.68763106]
[[-0.11189321  0.3331366 ]]

```

```

1 pd.DataFrame(X_test).to_csv("input_test.csv")

```

```

1 pd.DataFrame(y_test).to_csv("output_test.csv")

```

```

1 # Tính toán giá trị điểm thi LOGIC dựa trên mô hình
2 from sklearn import metrics
3
4 y_test_pred = model.predict(X_test)

```

```

1 pd.DataFrame(y_test_pred).to_csv("output_test_pred.csv")

```

```

1 # sử dụng các hàm python để đánh giá chất lượng mô hình trên tập test
2 '''
3 Tính toán các MAE, MSE, RMSE trên các tập training và test để đánh giá mô hình
4 Sau đó đưa ra kết luận
5 '''
6 import numpy as np
7 print('Mean Absolute Error:', metrics.mean_absolute_error(y_test,y_test_pred))
8 print('Mean Squared Error:', metrics.mean_squared_error(y_test,y_test_pred))
9 print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test,y_test_pred)))
10 print('Scored:', model.score(X_test,y_test))

```

Mean Absolute Error: 0.7739518570284659  
 Mean Squared Error: 0.8285004273228378  
 Root Mean Squared Error: 0.9102199884219406  
 Scored: -0.06473950499320513

'''

Homework: Sử dụng độ đo đánh giá mô hình RMSE trên tập train và test.

Hãy trả lời câu hỏi sau:

- Nếu kết quả rmse trên tập train > rmse trên tập test thì mô hình có ý nghĩa là gì
- Tương tự cho trường hợp bằng nhau và bé hơn

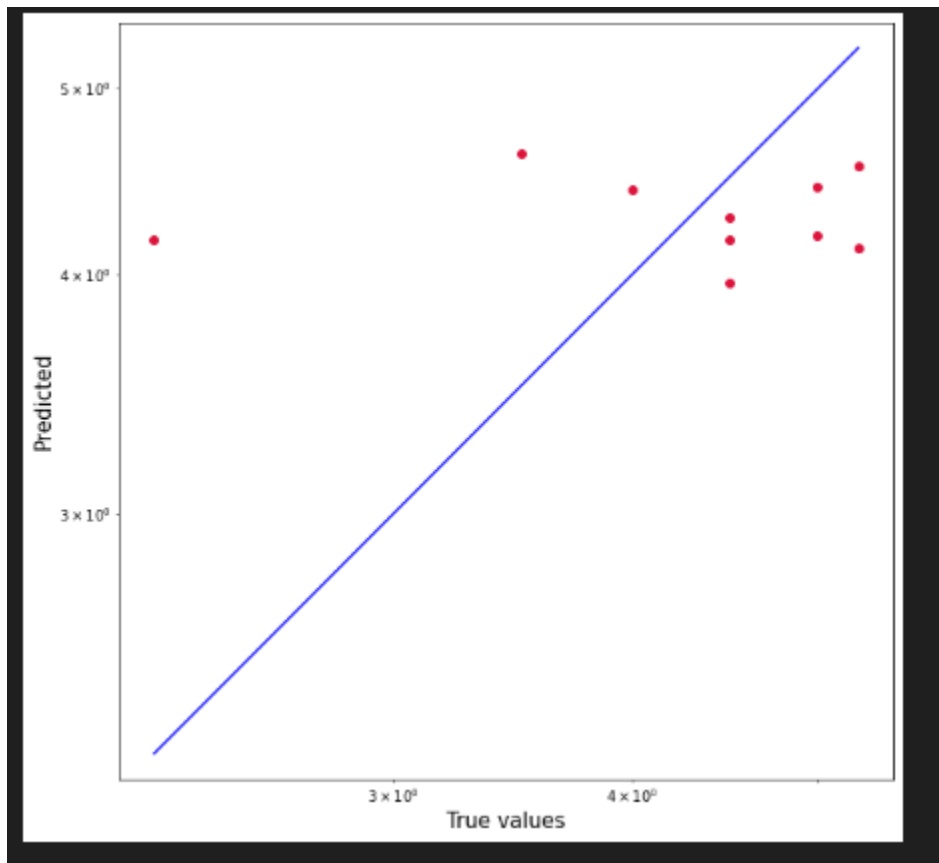
# overfit, underfit

'''

```

1  '''
2  Trực quan hoá dữ liệu điểm thi LOGIC thực (Actual) và dữ liệu LOGIC
3  theo dự báo (Predicted) trên tập test
4  y_test: actual value
5  y_test_pred: predicted value
6  '''
7  plt.figure(figsize=(10,10))
8  plt.scatter(y_test,y_test_pred,c='crimson')
9  plt.yscale('log')
10 plt.xscale('log')
11
12 p1 = max(max(y_test_pred), max(y_test))
13 p2 = min(min(y_test_pred), min(y_test))
14 plt.plot([p1,p2],[p1,p2],'b-')
15 plt.xlabel('True values', fontsize=15)
16 plt.ylabel('Predicted', fontsize=15)
17 plt.axis('equal')
18 plt.show()
19

```



```

1 # Residual Plot
2 # Tự nghiên cứu

1 # Lưu trữ mô hình đã huấn luyện xuống ổ đĩa
2 '''
3 Lưu trữ mô hình hồi quy xuống thiết bị lưu trữ với tên model_linear_regression.sav
4 '''
5 import pickle
6 pickle.dump(model,open('model_linear_regression.sav','wb'))

1 '''
2 Xây dựng chương trình: nhập điểm T5 T6 từ bàn phím, hãy dự báo điểm thi logic
3 '''
4 # Load model từ storage
5 loaded_model = pickle.load(open('model_linear_regression.sav','rb'))
6
7 vT5 = float(input('Điểm học kì 1 năm lớp 12: '))
8 vT6 = float(input('Điểm học kì 2 năm lớp 12: '))
9
10 predicted_vLogic = loaded_model.predict([[vT5,vT6]])
11 print(f'Dự báo điểm thi LOGIC là: {predicted_vLogic}')
12 # MAE = 0.77 thì trừ đi sai số [x,y]

```

Dự báo điểm thi LOGIC là: [[4.29355291]]

## 7) BÀI 7

### LUẬT KẾT HỢP

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 import pandas as pd
4 from apyori import apriori

1 store_data = pd.read_csv('./store_data.csv', header=None)
2 store_data.head()
3

```

	0	1	2	3	4	5	6	7	8	9	10	11
0	shrimp	almonds	avocado	vegetables mix	green grapes	whole weat flour	yams	cottage cheese	energy drink	tomato juice	low fat yogurt	green tea
1	burgers	meatballs	eggs	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	chutney	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	turkey	avocado	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	mineral water	milk	energy bar	whole wheat rice	green tea	NaN	NaN	NaN	NaN	NaN	NaN	NaN

```

1 records = []
2 for i in range(0, 7501):
3     records.append([str(store_data.values[i,j]) for j in range(0, 20)])

1 association_rules = apriori(records, min_support=0.0045, min_confidence=0.2,
2 min_lift=3, min_length=2)
3 association_results = list(association_rules)
4 '''
5 min_support=0.0045, min_confidence=0.2, min_lift=3, min_length=2
6 4 giá trị này có ý nghĩa là gì, khi giá trị tăng giảm thì ảnh hưởng ntn
7 min_length càng dài càng ít luật và ngược lại
8 min_support càng lớn thì càng k khám phá dc gì, tối thiểu là 0 thì nó khám phá hết
9 min_length: Luật càng dài thì càng chi tiết và trừu tượng thấp => ít sự lựa chọn,
10 luật càng ngắn thì trừu tượng cao và ít chi tiết
11 '''

1 print(len(association_results))
2

```



```
1 print(association_results[0])
2
```

RelationRecord(items=frozenset({'chicken', 'light cream'}), support=0.004532728969470737, c

```
1 for item in association_results:
2
3     # first index of the inner list
4     # Contains base item and add item
5     pair = item[0]
6     items = [x for x in pair]
7     print("Rule: " + items[0] + " -> " + items[1])
8
9     #second index of the inner list
10    print("Support: " + str(item[1]))
11
12    #third index of the list located at 0th
13    #of the third index of the inner list
14
15    print("Confidence: " + str(item[2][0][2]))
16    print("Lift: " + str(item[2][0][3]))
17    print("=====")
```

```

Rule: chicken -> light cream
Support: 0.004532728969470737
Confidence: 0.29059829059829057
Lift: 4.84395061728395
=====
Rule: mushroom cream sauce -> escalope
Support: 0.005732568990801226
Confidence: 0.3006993006993007
Lift: 3.790832696715049
=====
Rule: pasta -> escalope
Support: 0.005865884548726837
Confidence: 0.3728813559322034
Lift: 4.700811850163794
=====
Rule: herb & pepper -> ground beef
Support: 0.015997866951073192
Confidence: 0.3234501347708895
Lift: 3.2919938411349285
=====
Rule: tomato sauce -> ground beef
Support: 0.005332622317024397
Confidence: 0.3773584905660377
Lift: 3.840659481324083
=====
...
Support: 0.004532728969470737

```