# MỤC LỤC

**MỤC LỤC 5**

**DANH MỤC HÌNH ẢNH 11**

**PHẦN 1: CẤU TRÚC PHÂN TẦNG 12**

1. **TỔNG QUAN 12**
2. **CÀI ĐẶT 15**

**PHẦN 2: CẤU TRÚC CHO PHÂN LỚP 16**

1. **TẦNG DTO 16**
2. **CHỨC NĂNG 16**
3. **BẢNG HÀM VÀ LỚP 16**
4. **MÃ NGUỒN 17**
5. **Kết nối dataset 17**
6. **Hàm getFeature: Lấy dataset theo từng cột của đối tượng DauVao và sẽ xử lý cột ‘thoi\_luong\_lien\_lac’ và ‘so\_luong\_nhan\_vien’ cung cấp cho hàm multiprocessing 17**
7. **Hàm multiprocessing: Nhiệm vụ là xử lý song song 4 cột cùng 1 lúc để tạo ra đối tượng data cho class dataset 17**
8. **Lớp dataset: Nhiệm vụ lấy các data của hàm multiprocessing() để tạo ra 1 đối tượng dataset cho tầng DAO sử dụng 18**
9. **TẦNG DAO 19**
10. **CHỨC NĂNG 19**
11. **BẢNG HÀM VÀ LỚP 19**
12. **MÃ NGUỒN 22**
13. **Các thư viện sử dụng và thiết đặc liên kết với tầng DTO 22**
14. **Lấy đối tượng dataset từ tầng DTO 22**
15. **Hàm xoaTrung: Xoá phần tử trùng trong 1 mảng lấy từ 1 cột trong dataset, cung cấp cho hàm ganSoPhanLoai 23**
16. **Hàm ganSoPhanLoai: Lấy phần tử đã xoá trùng từ hàm xoaTrung để đánh số tương ứng với mỗi phần tử 23**
17. **Hàm chuyen\_dac\_truong\_sang\_so: Gán số từ hàm ganSoPhanLoai ánh xạ vào cột dataset tưong ứng để chuyển đặc trưng sang dạng số, sẽ có 2 tập là 1 tập dữ liệu unknown vẫn được đánh số và tập biến unknown thành null 23**
18. **Hàm traSoThuTu: Hàm cung cấp chỉ mục vị trí của các cột trong dataset nhầm cung cấp chọn cột dataset 24**
19. **Hàm data\_khongLoc: Trả về tập dataset mà không lọc unknown, đầu vào là giá trị k và mảng danh sách các đặc trưng array. Nếu k=0 trả về dataset và k=1 trả về label của nó. Nếu array=None trả về toàn bộ các đặc trưng còn lại chỉ trả về các đặc trưng trong array, hàm này cung cấp tập dữ liệu không lọc unknown lên tầng BUS 24**
20. **Chuyển các đặc trưng sang số có cho phép null 27**
21. **Bắt đầu lọc dữ liệu xoá null 29**
22. **Hàm data\_CoLoc: Trả về tập dataset mà có lọc unknown, đầu vào là giá trị k và mảng danh sách các đặc trưng array. Nếu k=0 trả về dataset và k=1 trả về label của nó. Nếu array=None trả về toàn bộ các đặc trưng còn lại chỉ trả về các đặc trưng trong array, hàm này cung cấp tập dữ liệu có lọc unknown lên tầng BUS 30**
23. **TẦNG BUS 31**
24. **KNN 31**
25. **Cơ sở lý thuyết: 31**
26. **Bảng hàm và lớp: 31**
27. **Mã nguồn 35**

* **Các thư viện và trỏ liên kết tới DAO 35**
* **Hàm sinhToHop: Sinh tổ hợp chập k của 19(đặc trưng), mục đích cung cấp cho hàm NhungDacTrungTotNhat 36**
* **Lớp nhungDacTrungTotNhat: Lưu lại những đặc trưng tốt cho thuật toán sau khi sinh tổ hợp và tính toán 37**
* **Hàm NhungDacTrungTotNhat: Mục đích là tìm những đặc trưng tốt nhất trong khoảng nhỏ vì thực thi hết là không thể nên chưa bao gồm tất cả các đặc trưng, hàm cung cấp cho hàm timNhungDacTrungTotNhat 37**
* **Hàm timNhungDacTrungTotNhat: Mục đích là tìm những đặc trưng tốt nhất trong khoảng nhỏ vì thực thi hết là không thể và so sánh với tất cả các đặc trưng 38**
* **Tập dữ liệu sử dụng sinh tổ hợp 39**
* **Tập dữ liệu không sử dụng sinh tổ hợp 39**
* **Hàm cross\_Validation: Cung cấp đánh gía về độ chính xác của KNN trên tập dữ liệu X ở đây em cho cv=10 chia thành 10 tập 39**
* **Lớp class good\_KNN: Lưu lại những parameters tốt nhất sau khi chạy Grid Search 40**
* **Hàm n\_neighbors: Cung cấp list các n\_neighbors cho hàm xuLy\_knn\_CoLoc và xuLy\_knn\_KhongLoc ở đây em chia thành 5 giá trị k từ 0 đến 74 vì mục đích chính chia nhỏ khoảng lại thì Grid Search trong các hàm đó sẽ chính xác hơn 40**
* **Hàm timF\_CoLoc: Mục đích là dùng Grid Search để tìm ra những parameters tốt nhất cho KNN ở đây em cho chạy tìm 2 parameters là n\_neighbors, weights vì thời gian tính sẽ hạn chế hơn và những parameters phụ thuộc nhau như p, metric đều không ảnh hưởng lớn đến kết quả, hàm này chỉ dành cho tập dữ liệu có lọc của DAO cung cấp, kết quả trả ra là đối tượng good\_KNN\_CoLoc lưu lại những parameters tốt nhất, hàm này cung cấp cho hàm xuLy\_knn\_CoLoc 41**
* **Hàm timF\_KhongLoc: Mục đích là dùng Grid Search để tìm ra những parameters tốt nhất cho KNN ở đây em cho chạy tìm 2 parameters là n\_neighbors, weights vì thời gian tính sẽ hạn chế hơn và những parameters phụ thuộc nhau như p, metric đều không ảnh hưởng lớn đến kết quả, hàm này chỉ dành cho tập dữ liệu không lọc của DAO cung cấp, kết quả trả ra là đối tượng good\_KNN\_KhongLoc lưu lại những parameters tốt nhất, hàm này cung cấp cho hàm xuLy\_knn\_KhongLoc 42**
* **Hàm xuLy\_knn\_CoLoc: Mục đích là gọi hàm timF\_CoLoc, n\_neighbors để truyền list n\_neighbors mục đích là Grid Search ở những khoảng nhỏ sẽ ra độ chính xác tốt hơn, chỉ dùng cho tập dữ liệu có lọc, hàm này cung cấp cho hàm ketQua 43**
* **Hàm xuLy\_knn\_KhongLoc: Mục đích là gọi hàm timF\_KhongLoc, n\_neighbors để truyền list n\_neighbors mục đích là Grid Search ở những khoảng nhỏ sẽ ra độ chính xác tốt hơn, chỉ dùng cho tập dữ liệu không lọc, hàm này cung cấp cho hàm ketQua 43**
* **Hàm ketQua: Hàm này là hàm hiện kết quả cuối cùng như cross\_Validation hay kết quả tính F, n\_neighbors, weights sau khi đã Grid Search. Ở đây có k đầu vào là nếu k=0 cho tập dữ liệu có lọc và k=1 cho tập dữ liệu không lọc 44**
* **Hàm traSoThuTu: Mục đích là trả về chỉ mục vị trí trong dataset 45**
* **Hàm Ve2D: Cung cấp hàm vẽ 2 chiều cho các đặc trưng 45**
* **Hàm Ve3D: Cung cấp hàm vẽ 3 chiều cho các đặc trưng 47**
* **Hàm help: Cung cấp cú pháp để dễ sử dụng 48**

1. **Đánh giá 49**
2. **TẦNG APP 53**
3. **CHỨC NĂNG 53**
4. **BẢNG HÀM 53**
5. **MÃ NGUỒN 53**
6. **File setup.py 53**

* **Chức năng 53**
* **Khai báo thư viện Cython sử dụng: 54**
* **Trỏ đường dẫn đến các file .pyx của các tầng để biên dịch ra các module(file .o) cung cấp cho file index.py sử dụng: 54**

1. **File main.pyx 54**

* **Chức năng 54**

1. **File index.py 54**

* **Chức năng 54**
* **Liên kết tới main.o khi mới tạo ra 54**
* **Sử dụng các hàm trong main.o 55**

**PHẦN 3: CẤU TRÚC CHO PHÂN CỤM 55**

1. **TẦNG DTO 55**
2. **CHỨC NĂNG 55**
3. **BẢNG HÀM VÀ LỚP 55**
4. **MÃ NGUỒN 56**
5. **Các thư viện sử dụng: 56**
6. **Kết nối dataset 56**
7. **Hàm getFeature: Lấy dataset theo từng cột của đối tượng DauVao và cung cấp cho hàm multiprocessing 56**
8. **Hàm multiprocessing: Nhiệm vụ là xử lý song song 4 cột cùng 1 lúc để tạo ra đối tượng data cho class dataset 57**
9. **Lớp dataset: Trả về đối tượng dữ liệu dataset cho tầng DAO sử dụng 57**
10. **Hàm xuatFile: Ghi kết quả phân cụm của KM và HC vào file mới là ketqua.csv 58**
11. **TẦNG DAO 58**
12. **CHỨC NĂNG 58**
13. **BẢNG HÀM 58**
14. **MÃ NGUỒN 59**
15. **Các thư viện sử dụng và chỉ liên kết tới tầng DTO: 59**
16. **Hàm traSoThuTu: Hàm cung cấp chỉ mục vị trí của các cột trong dataset nhầm cung cấp chọn cột dataset 60**
17. **Hàm chuyenData: Mục đích cung cấp cho hàm dataset 60**
18. **Hàm dataset: Cung cấp đối tượng dataset cho tầng BUS có 2 tuỳ chọn là None: trả về toàn bộ các đặc trưng và array chỉ trả về đặc trưng trong mảng array 61**
19. **Hàm xuatFile: Cung cấp hàm ghi file tương tác với hàm xuatFile của tầng DTO 61**
20. **TẦNG BUS 62**
21. **KMeans 62**
22. **Cơ sở lý thuyết: 62**
23. **Bảng hàm và lớp: 62**
24. **Mã nguồn: 64**

* **Các thư viện và trỏ liên kết tới DAO 64**
* **Hàm lay\_data: Lấy tập dataset từ tầng DAO lên và trả về 1 đối tượng dataset 65**
* **Hàm KMean: Là hàm xử lý thuật toán KMeans 65**
* **Hàm veTimSoCluster: Cung cấp hàm vẽ để tìm ra số cụm thích hợp 65**
* **Hàm traSoThuTu: Mục đích là trả về chỉ mục vị trí trong dataset 66**
* **Hàm veDacTrung2D: Cung cấp hàm vẽ 2 chiều cho KM với mangCacDacTrung là None thì ta lấy mangCacDacTrungVe training, soLuongDiemVe là None thì ta vẽ tất cả điểm 66**
* **Hàm veDacTrung3D: Cung cấp hàm vẽ 3 chiều cho KM với mangCacDacTrung là None thì ta lấy mangCacDacTrungVe training, soLuongDiemVe là None thì ta vẽ tất cả điểm 68**
* **Hàm help: Hàm cung cấp cú pháp để dễ sử dụng 69**

1. **Đánh giá 70**
2. **Hierarchical clustering(HC) 72**
3. **Bảng hàm : 72**
4. **Mã nguồn: 73**

* **Các thư viện và trỏ liên kết tới DAO 73**
* **Hàm lay\_data: Lấy tập dataset từ tầng DAO lên và trả về 1 đối tượng dataset 73**
* **Hàm HC: Là hàm xử lý thuật toán HC 74**
* **Hàm vitriCat: Là hàm trả về vị trí cắt để phân cụm 74**
* **Hàm ve\_HC: Vẽ biểu đồ cột phân cụm để phân chia cụm 75**
* **Hàm help: Hàm cung cấp cú pháp để dễ sử dụng 75**

1. **Đánh giá 75**
2. **TẦNG APP 77**
3. **CHỨC NĂNG 77**
4. **BẢNG HÀM 77**
5. **MÃ NGUỒN 78**
6. **File setup.py 78**

* **Chức năng 78**
* **Khai báo thư viện Cython sử dụng: 78**
* **Trỏ đường dẫn đến các file .pyx của các tầng để biên dịch ra các module(file .o) cung cấp cho file index.py sử dụng: 78**

1. **File main.pyx 79**

* **Chức năng 79**

1. **File index.py 79**

* **Chức năng 79**
* **Liên kết tới main.o khi mới tạo ra 79**
* **Sử dụng các hàm trong main.o 79**

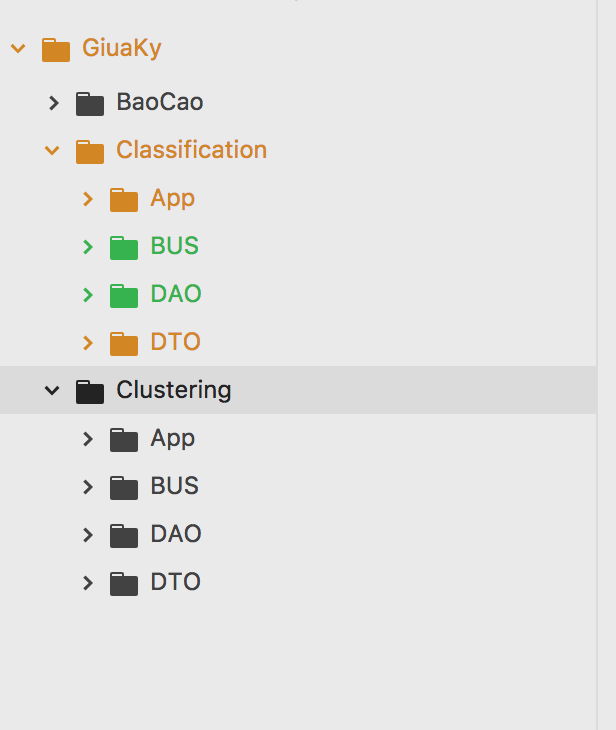
**PHẦN 4: PHÂN CÔNG 80**

# DANH MỤC HÌNH ẢNH

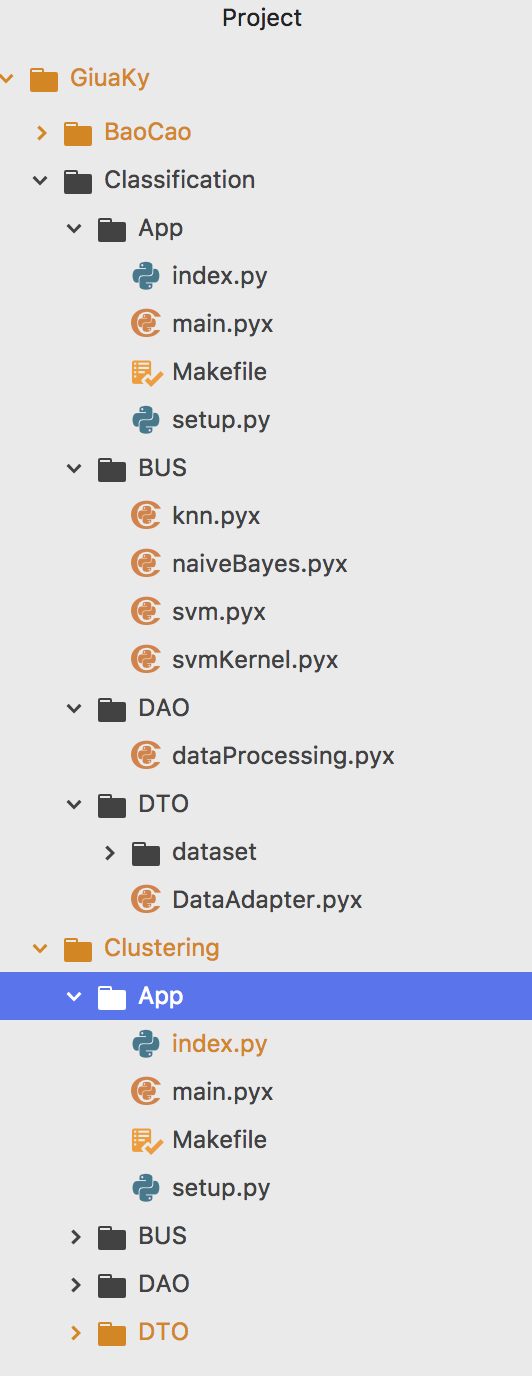
# PHẦN 1: CẤU TRÚC PHÂN TẦNG

## TỔNG QUAN

Các thuật toán trong phân lớp và phân cụm được cài đặt chia thành 4 tầng khác nhau việc chia các tầng nhằm đảm bảo tính linh động cũng như dễ kiểm soát lỗi, nâng cấp và bảo trì. Ở các tầng chỉ có liên kết theo thứ tự như sau App <—> BUS <—> DAO <—> DTO :

* DTO: Nhiệm vụ là tầng chính kết nối đọc và ghi xuống dataset và trả lên đối tượng dữ liệu cho DAO.
* DAO: Nhiệm vụ là biến đổi xử lý dữ liệu và xử lý dữ liệu cho phù hợp với thuật toán và cung cấp đối tượng dữ liệu lên tầng BUS.
* BUS: Là tầng chính xử lý các thuật toán, cung cấp các hàm vẽ và kết quả xử lý các thuật toán lên tầng App.
* App: Là tầng giao tiếp trực tiếp với người dùng cuối và cấu hình thông qua Makefile.

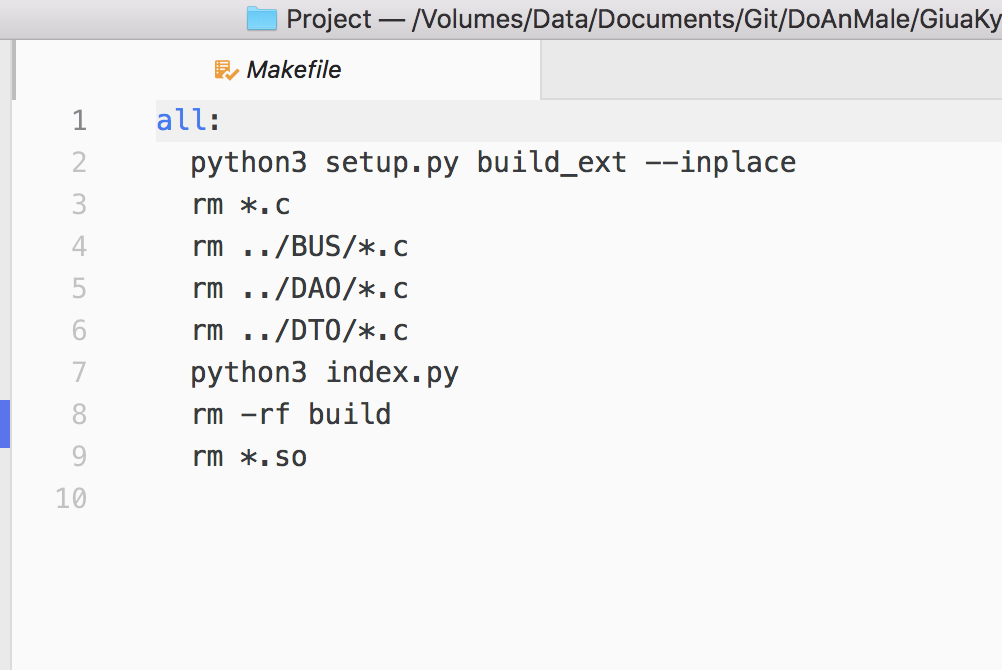
Hình 1: Cấu trúc 1.



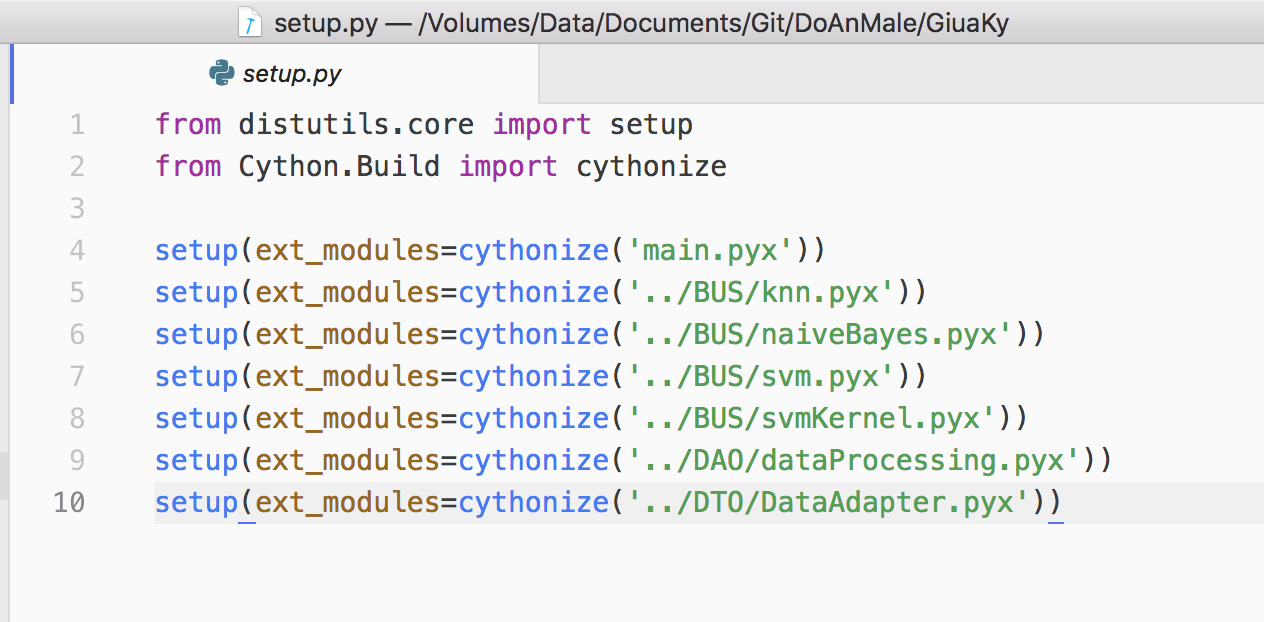
Hình 2: Cấu trúc 2.

Toàn bộ chương trình để tăng tốc độ thực thi nên chúng em có sử dụng Cython:

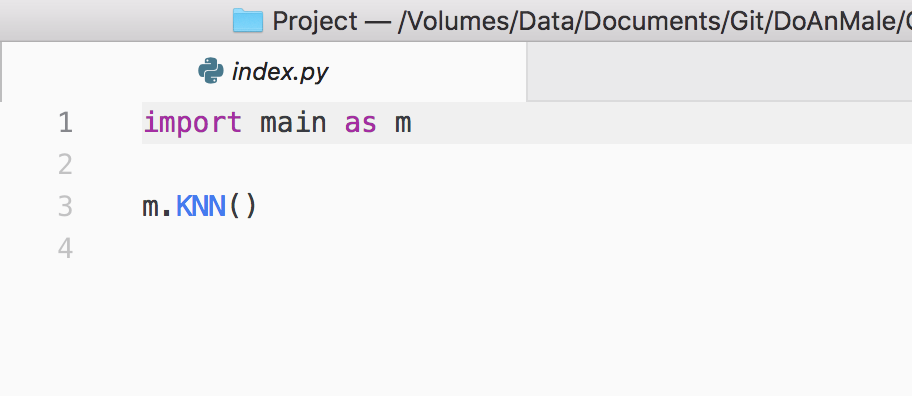
* Là ngôn ngữ nhằm hỗ trợ cú pháp của Python, C và được thiết kế để mang hiệu năng như ngôn ngữ C.
* Cython thực thi sẽ biên dịch ra mã nguồn C và xuất ra các module để chương trình python có thể gọi và thực thi.
* Cơ chế là đầu tiên Cython sẽ biên dịch ra mã nguồn C rồi tiếp đến là file .so và cuối cùng là file object(.o).
* Cython có đuôi mở rộng là .pyx nên mọi tệp Python (.py) cần được đổi sang .pyx mới thoả đầu vào biên dịch. Để biên dịch toàn bộ chương trình sử dụng Cython vì có nhiều tầng nên chúng em sử dụng f ile setup.py nằm tại thư mục App để biên dịch cho tất cả các tầng và được thực thi thông qua Makefile:



Hình 3: Cấu trúc Makefile.



Hình 4: setup.py.



Hình 5: index.py của phân lớp.

Từ ảnh 3, 4 và 5 cho ta biết lệnh biên dịch là “python3 setup.py build\_ext —inplace” và lệnh này sẽ thực thi file setup.py để tạo ra các file object(.o) tương ứng và lệnh “python3 index.py” sẽ gọi file index.py để gọi các object file tương ứng.

## CÀI ĐẶT

Yêu cầu chương trình hoặc thư viên cần có:

* Python3
* Matplotlib
* Sklearn
* Pandas
* Cython
* Cmake/Gcc

Để chạy chương trình ta cần chạy Makefile trong thư mục App với lệnh “make” thì chương trình sẽ tự động biên dịch và thực thi nếu cần cấu hình thì chỉ việc cấu hình trong file index.py và file main.py.

# PHẦN 2: CẤU TRÚC CHO PHÂN LỚP

## TẦNG DTO

### CHỨC NĂNG

Trong phân lớp ở tầng này nhiệm vụ là đọc dataset và chuyển nó thành đối tượng dữ liệu và nó được thực thi trong file DataAdapter.pyx.

### BẢNG HÀM VÀ LỚP

Bảng 1: Hàm

| STT | Tên hàm | Mục đích của hàm | File lưu trữ | Tên sinh viên phụ trách |
| --- | --- | --- | --- | --- |
| 1 | getFeature  Input: Tên cột trong dataset  Output: Trả về list của cột đó. | Lấy dataset theo từng cột của đối tượng DauVao và sẽ xử lý cột ‘thoi\_luong\_lien\_lac’ và ‘so\_luong\_nhan\_vien’ cung cấp cho hàm multiprocessing | DataAdapter.pyx | Nguyễn Tiến Đạt |
| 2 | multiprocessing  Input: Không  Output: Đối tượng dữ liệu data. | Nhiệm vụ là xử lý song song 4 cột cùng 1 lúc để tạo ra đối tượng data cho class dataset | DataAdapter.pyx | Nguyễn Tiến Đạt |

Bảng 2: Lớp

| STT | Tên lớp | Tên sinh viên phụ trách | Mục đích của lớp |
| --- | --- | --- | --- |
| 1 | dataset | Nguyễn Tiến Đạt | Trả về đối tượng dữ liệu dataset cho tầng DAO sử dụng. |

### MÃ NGUỒN

#### Kết nối dataset

DauVao=pd.read\_csv('../DTO/dataset/dataset\_for\_classification.csv',encoding='utf-8')

#### Hàm getFeature: Lấy dataset theo từng cột của đối tượng DauVao và sẽ xử lý cột ‘thoi\_luong\_lien\_lac’ và ‘so\_luong\_nhan\_vien’ cung cấp cho hàm multiprocessing

def getFeature(name):

if name=='thoi\_luong\_lien\_lac' or name=='so\_luong\_nhan\_vien':

o=[]

for i in DauVao[name].values:

o.append(int((i\*13.999-13)/999))

else:

o=(DauVao[name].values).tolist()

return o

#### Hàm multiprocessing: Nhiệm vụ là xử lý song song 4 cột cùng 1 lúc để tạo ra đối tượng data cho class dataset

def multiprocessing():

t=['tuoi','nghe\_nghiep','hon\_nhan','hoc\_van','co\_the\_tin\_dung',

'co\_nha\_o','vay\_ca\_nhan','kenh\_lien\_lac','thang\_lien\_lac',

'ngay\_lien\_lac','thoi\_luong\_lien\_lac','so\_luong\_lien\_lac',

'ngay','so\_luong\_lien\_lac\_truoc\_day','ket\_qua\_lan\_truoc',

'ti\_le\_thay\_doi\_viec\_lam','CPI','CCI','lai\_suat\_3thang',

'so\_luong\_nhan\_vien','label']

pool = ThreadPool(4)

data=pool.map(getFeature,t)

return data

#### Lớp dataset: Nhiệm vụ lấy các data của hàm multiprocessing() để tạo ra 1 đối tượng dataset cho tầng DAO sử dụng

class dataset:

def \_\_init\_\_(seft):

data=multiprocessing()

seft.tuoi=data[0]

seft.nghe\_nghiep=data[1]

seft.hon\_nhan=data[2]

seft.hoc\_van=data[3]

seft.co\_the\_tin\_dung=data[4]

seft.co\_nha\_o=data[5]

seft.vay\_ca\_nhan=data[6]

seft.kenh\_lien\_lac=data[7]

seft.thang\_lien\_lac=data[8]

seft.ngay\_lien\_lac=data[9]

seft.thoi\_luong\_lien\_lac= data[10]

seft.so\_luong\_lien\_lac=data[11]

seft.ngay=data[12]

seft.so\_luong\_lien\_lac\_truoc\_day=data[13]

seft.ket\_qua\_lan\_truoc=data[14]

seft.ti\_le\_thay\_doi\_viec\_lam=data[15]

seft.CPI=data[16]

seft.CCI=data[17]

seft.lai\_suat\_3thang=data[18]

seft.so\_luong\_nhan\_vien=data[19]

seft.label=data[20]

## TẦNG DAO

### CHỨC NĂNG

Mục đích chính ở tầng DAO là biến đổi dữ liệu ban đầu thành các tập dữ liệu phù hợp với thuật toán sử dụng.

### BẢNG HÀM VÀ LỚP

Bảng 1: Hàm

| STT | Tên hàm | Mục đích của hàm | File lưu trữ | Tên sinh viên phụ trách |
| --- | --- | --- | --- | --- |
| 1 | xoaTrung  Input: 1 mảng cần xoá trùng.  Output: Mảng đã được xoá phần tủ trùng. | Xoá phần tử trùng trong 1 mảng lấy từ 1 cột trong dataset, cung cấp cho hàm ganSoPhanLoai | dataProcessing.pyx | Nguyễn Tiến Đạt |
| 2 | ganSoPhanLoai  Input: xoaTrung.  Output: mảng đã được đánh số. | Lấy phần tử đã xoá trùng từ hàm xoaTrung để đánh số tương ứng với mỗi phần tử. | dataProcessing.pyx | Nguyễn Tiến Đạt |
| 3 | chuyen\_dac\_truong\_sang\_so  Input: cột đặc trưng cũ, ganSoPhanLoai.  Output: có 2 trường hơp 1 mảng được gắn số cho dữ liệu unknown và 1 mảng không có dữ liệu unknown. | Gán số từ hàm ganSoPhanLoai ánh xạ vào cột dataset tưong ứng để chuyển đặc trưng sang dạng số, sẽ có 2 tập là 1 tập dữ liệu unknown vẫn được đánh số và tập biến unknown thành null. | dataProcessing.pyx | Nguyễn Tiến Đạt |
| 4 | traSoThuTu  Input: tên cột trong dataset.  Output: index của cột đó. | Hàm cung cấp chỉ mục vị trí của các cột trong dataset nhầm cung cấp chọn cột dataset. | dataProcessing.pyx | Nguyễn Tiến Đạt |
| 5 | data\_khongLoc  Input: k, array  Output: tập dataset mới hoặc label của nó. | Trả về tập dataset mà không lọc unknown, đầu vào là giá trị k và mảng danh sách các đặc trưng array.  Nếu k=0 trả về dataset và k=1 trả về label của nó.  Nếu array=None trả về toàn bộ các đặc trưng còn lại chỉ trả về các đặc trưng trong array, hàm này cung cấp tập dữ liệu không lọc unknown lên tầng BUS | dataProcessing.pyx | Nguyễn Tiến Đạt |
| 6 | data\_CoLoc  Input: k, array  Output: tập dataset mới hoặc label của nó. | Trả về tập dataset mà có lọc unknown, đầu vào là giá trị k và mảng danh sách các đặc trưng array.  Nếu k=0 trả về dataset và k=1 trả về label của nó.  Nếu array=None trả về toàn bộ các đặc trưng còn lại chỉ trả về các đặc trưng trong array, hàm này cung cấp tập dữ liệu có lọc unknown lên tầng BUS | dataProcessing.pyx | Nguyễn Tiến Đạt |

### MÃ NGUỒN

#### Các thư viện sử dụng và thiết đặc liên kết với tầng DTO

# coding=utf-8

import sys

sys.path.append('../DTO/')

import pandas as pd

import DataAdapter as da

from multiprocessing.dummy import Pool as ThreadPool

#### Lấy đối tượng dataset từ tầng DTO

dataset=da.dataset()

#### Hàm xoaTrung: Xoá phần tử trùng trong 1 mảng lấy từ 1 cột trong dataset, cung cấp cho hàm ganSoPhanLoai

# xoa phan tu trung lap va cho gia tri unknown anh xa --> 0

def xoaTrung(a):

b=[]

for i in a:

if i not in b:

b.append(i)

for i in range(len(b)):

if b[i]=='unknown':

b[0],b[i]=b[i],b[0]

return b

#### Hàm ganSoPhanLoai: Lấy phần tử đã xoá trùng từ hàm xoaTrung để đánh số tương ứng với mỗi phần tử

# ham chuyen doi qua so cho cac gia tri da xoa trung

def ganSoPhanLoai(mangXoaTrung):

b=[]

for i in range(len(mangXoaTrung)):

b.append([mangXoaTrung[i],i])

return b

#### Hàm chuyen\_dac\_truong\_sang\_so: Gán số từ hàm ganSoPhanLoai ánh xạ vào cột dataset tưong ứng để chuyển đặc trưng sang dạng số, sẽ có 2 tập là 1 tập dữ liệu unknown vẫn được đánh số và tập biến unknown thành null

# ham chuyen cac dac trung ban dau thanh dang so

def chuyen\_dac\_truong\_sang\_so(dac\_trung\_cu,mang\_ganSoPhanLoai,f):

a=[]

for i in dac\_trung\_cu:

if f==0:

for j in mang\_ganSoPhanLoai:

if i==j[0]:

a.append(j[1])

# ham chuyen cac dac trung ban dau thanh dang so neu gia tri unknown thi chuyen thanh null

elif f==1:

for j in mang\_ganSoPhanLoai:

if i==j[0] and i!= 'unknown':

a.append(j[1])

elif i== 'unknown' and i==j[0]:

a.append(None)

return a;

#### Hàm traSoThuTu: Hàm cung cấp chỉ mục vị trí của các cột trong dataset nhầm cung cấp chọn cột dataset

def traSoThuTu(ten):

t=['tuoi','nghe\_nghiep','hon\_nhan','hoc\_van','co\_the\_tin\_dung',

'co\_nha\_o','vay\_ca\_nhan','kenh\_lien\_lac','thang\_lien\_lac',

'ngay\_lien\_lac','thoi\_luong\_lien\_lac','so\_luong\_lien\_lac',

'ngay','so\_luong\_lien\_lac\_truoc\_day','ket\_qua\_lan\_truoc',

'ti\_le\_thay\_doi\_viec\_lam','CPI','CCI','lai\_suat\_3thang',

'so\_luong\_nhan\_vien']

for i in range(len(t)):

if t[i]==ten:

return i;

return -1;

#### Hàm data\_khongLoc: Trả về tập dataset mà không lọc unknown, đầu vào là giá trị k và mảng danh sách các đặc trưng array. Nếu k=0 trả về dataset và k=1 trả về label của nó. Nếu array=None trả về toàn bộ các đặc trưng còn lại chỉ trả về các đặc trưng trong array, hàm này cung cấp tập dữ liệu không lọc unknown lên tầng BUS

# bat dau chuyen cac dac trung sang so khong cho phep null

def data\_khongLoc(\_k,array):

k=[]

data=[]

nghe\_nghiep=chuyen\_dac\_truong\_sang\_so(dataset.nghe\_nghiep, ganSoPhanLoai(xoaTrung(dataset.nghe\_nghiep)),0)

hon\_nhan=chuyen\_dac\_truong\_sang\_so(dataset.hon\_nhan, ganSoPhanLoai(xoaTrung(dataset.hon\_nhan)),0)

hoc\_van=chuyen\_dac\_truong\_sang\_so(dataset.hoc\_van, ganSoPhanLoai(xoaTrung(dataset.hoc\_van)),0)

co\_the\_tin\_dung=chuyen\_dac\_truong\_sang\_so(dataset.co\_the\_tin\_dung, ganSoPhanLoai(xoaTrung(dataset.co\_the\_tin\_dung)),0)

co\_nha\_o=chuyen\_dac\_truong\_sang\_so(dataset.co\_nha\_o, ganSoPhanLoai(xoaTrung(dataset.co\_nha\_o)),0)

vay\_ca\_nhan=chuyen\_dac\_truong\_sang\_so(dataset.vay\_ca\_nhan, ganSoPhanLoai(xoaTrung(dataset.vay\_ca\_nhan)),0)

kenh\_lien\_lac=chuyen\_dac\_truong\_sang\_so(dataset.kenh\_lien\_lac, ganSoPhanLoai(xoaTrung(dataset.kenh\_lien\_lac)),0)

thang\_lien\_lac=chuyen\_dac\_truong\_sang\_so(dataset.thang\_lien\_lac, ganSoPhanLoai(xoaTrung(dataset.thang\_lien\_lac)),0)

ngay\_lien\_lac=chuyen\_dac\_truong\_sang\_so(dataset.ngay\_lien\_lac, ganSoPhanLoai(xoaTrung(dataset.ngay\_lien\_lac)),0)

ngay=chuyen\_dac\_truong\_sang\_so(dataset.ngay, ganSoPhanLoai(xoaTrung(dataset.ngay)),0)

ket\_qua\_lan\_truoc=chuyen\_dac\_truong\_sang\_so(dataset.ket\_qua\_lan\_truoc, ganSoPhanLoai(xoaTrung(dataset.ket\_qua\_lan\_truoc)),0)

tuoi=dataset.tuoi

thoi\_luong\_lien\_lac=dataset.thoi\_luong\_lien\_lac

so\_luong\_lien\_lac=dataset.so\_luong\_lien\_lac

so\_luong\_lien\_lac\_truoc\_day=dataset.so\_luong\_lien\_lac\_truoc\_day

ti\_le\_thay\_doi\_viec\_lam=dataset.ti\_le\_thay\_doi\_viec\_lam

CPI=dataset.CPI

CCI=dataset.CCI

lai\_suat\_3thang=dataset.lai\_suat\_3thang

so\_luong\_nhan\_vien=dataset.so\_luong\_nhan\_vien

x\_training\_khongLoc=[]

# tao tap du lieu x\_training khong loc unknown

for i in range(len(tuoi)):

x\_training\_khongLoc.append([

tuoi[i]

,nghe\_nghiep[i]

,hon\_nhan[i]

,hoc\_van[i]

,co\_the\_tin\_dung[i]

,co\_nha\_o[i]

,vay\_ca\_nhan[i]

,kenh\_lien\_lac[i]

,thang\_lien\_lac[i]

,ngay\_lien\_lac[i]

,thoi\_luong\_lien\_lac[i]

,so\_luong\_lien\_lac[i]

,ngay[i]

,so\_luong\_lien\_lac\_truoc\_day[i]

,ket\_qua\_lan\_truoc[i]

,ti\_le\_thay\_doi\_viec\_lam[i]

,CPI[i]

,CCI[i]

,lai\_suat\_3thang[i]

,so\_luong\_nhan\_vien[i]

])

if \_k==0 and array is None:

return x\_training\_khongLoc

elif \_k==1 and array is None:

return da.dataset().label

elif array is not None:

if len(array)>19:

return None;

else:

for i in array:

if traSoThuTu(i)!=-1:

k.append(traSoThuTu(i))

for i in range(len(x\_training\_khongLoc)):

k1=[]

for j in k:

k1.append(x\_training\_khongLoc[i][j])

data.append(k1)

return data;

#### Chuyển các đặc trưng sang số có cho phép null

# bat dau chuyen cac dac trung sang vector so cho phep null

nghe\_nghiep1=chuyen\_dac\_truong\_sang\_so(dataset.nghe\_nghiep, ganSoPhanLoai(xoaTrung(dataset.nghe\_nghiep)),1)

hon\_nhan1=chuyen\_dac\_truong\_sang\_so(dataset.hon\_nhan, ganSoPhanLoai(xoaTrung(dataset.hon\_nhan)),1)

hoc\_van1=chuyen\_dac\_truong\_sang\_so(dataset.hoc\_van, ganSoPhanLoai(xoaTrung(dataset.hoc\_van)),1)

co\_the\_tin\_dung1=chuyen\_dac\_truong\_sang\_so(dataset.co\_the\_tin\_dung, ganSoPhanLoai(xoaTrung(dataset.co\_the\_tin\_dung)),1)

co\_nha\_o1=chuyen\_dac\_truong\_sang\_so(dataset.co\_nha\_o, ganSoPhanLoai(xoaTrung(dataset.co\_nha\_o)),1)

vay\_ca\_nhan1=chuyen\_dac\_truong\_sang\_so(dataset.vay\_ca\_nhan, ganSoPhanLoai(xoaTrung(dataset.vay\_ca\_nhan)),1)

kenh\_lien\_lac1=chuyen\_dac\_truong\_sang\_so(dataset.kenh\_lien\_lac, ganSoPhanLoai(xoaTrung(dataset.kenh\_lien\_lac)),1)

thang\_lien\_lac1=chuyen\_dac\_truong\_sang\_so(dataset.thang\_lien\_lac, ganSoPhanLoai(xoaTrung(dataset.thang\_lien\_lac)),1)

ngay\_lien\_lac1=chuyen\_dac\_truong\_sang\_so(dataset.ngay\_lien\_lac, ganSoPhanLoai(xoaTrung(dataset.ngay\_lien\_lac)),1)

ngay1=chuyen\_dac\_truong\_sang\_so(dataset.ngay, ganSoPhanLoai(xoaTrung(dataset.ngay)),1)

ket\_qua\_lan\_truoc1=chuyen\_dac\_truong\_sang\_so(dataset.ket\_qua\_lan\_truoc, ganSoPhanLoai(xoaTrung(dataset.ket\_qua\_lan\_truoc)),1)

tuoi1=dataset.tuoi

thoi\_luong\_lien\_lac1=dataset.thoi\_luong\_lien\_lac

so\_luong\_lien\_lac1=dataset.so\_luong\_lien\_lac

so\_luong\_lien\_lac\_truoc\_day1=dataset.so\_luong\_lien\_lac\_truoc\_day

ti\_le\_thay\_doi\_viec\_lam1=dataset.ti\_le\_thay\_doi\_viec\_lam

CPI1=dataset.CPI

CCI1=dataset.CCI

lai\_suat\_3thang1=dataset.lai\_suat\_3thang

so\_luong\_nhan\_vien1=dataset.so\_luong\_nhan\_vien

# tao tap du lieu x\_training co loc du lieu unknown

x\_training\_CoLoc=[]

for i in range(len(tuoi1)):

x\_training\_CoLoc.append([

tuoi1[i]

,nghe\_nghiep1[i]

,hon\_nhan1[i]

,hoc\_van1[i]

,co\_the\_tin\_dung1[i]

,co\_nha\_o1[i]

,vay\_ca\_nhan1[i]

,kenh\_lien\_lac1[i]

,thang\_lien\_lac1[i]

,ngay\_lien\_lac1[i]

,thoi\_luong\_lien\_lac1[i]

,so\_luong\_lien\_lac1[i]

,ngay1[i]

,so\_luong\_lien\_lac\_truoc\_day1[i]

,ket\_qua\_lan\_truoc1[i]

,ti\_le\_thay\_doi\_viec\_lam1[i]

,CPI1[i]

,CCI1[i]

,lai\_suat\_3thang1[i]

,so\_luong\_nhan\_vien1[i]

])

#### Bắt đầu lọc dữ liệu xoá null

# bat dau loc du lieu x\_training\_coLoc chua unknown

y\_training\_CL=da.dataset().label

i=len(x\_training\_CoLoc)-1

while i>=0:

if i<len(x\_training\_CoLoc) :

for j in x\_training\_CoLoc[i]:

if j is None:

x\_training\_CoLoc.pop(i)

y\_training\_CL.pop(i)

break

i-=1

#### Hàm data\_CoLoc: Trả về tập dataset mà có lọc unknown, đầu vào là giá trị k và mảng danh sách các đặc trưng array. Nếu k=0 trả về dataset và k=1 trả về label của nó. Nếu array=None trả về toàn bộ các đặc trưng còn lại chỉ trả về các đặc trưng trong array, hàm này cung cấp tập dữ liệu có lọc unknown lên tầng BUS

def data\_CoLoc(\_k,array):

k=[]

data=[]

if \_k==0 and array is None:

return x\_training\_CoLoc

elif \_k==1 and array is None:

return y\_training\_CL

elif array is not None:

if len(array)>19:

return None;

else:

for i in array:

if traSoThuTu(i)!=-1:

k.append(traSoThuTu(i))

for i in range(len(x\_training\_CoLoc)):

k1=[]

for j in k:

k1.append(x\_training\_CoLoc[i][j])

data.append(k1)

return data;

## TẦNG BUS

### KNN

#### Cơ sở lý thuyết:

Thuật toán KNN được cho là thuật toán đơn giản nhất trong Machine learning. Mô hình được xây dựng chỉ bao gồm việc lưu trữ dữ liệu tập huấn *(training dataset)*. Để dự đoán được một điểm dữ liệu mới, thuật toán sẽ tìm ra những *láng giềng* trong dữ liệu tập huấn *(training dataset).*

#### Bảng hàm và lớp:

Bảng 1: Hàm

| STT | Tên hàm | Mục đích của hàm | File lưu trữ | Tên sinh viên phụ trách |
| --- | --- | --- | --- | --- |
| 1 | sinhToHop  Input: k  Output: sinh tổ hợp chập k của 19 đặc trưng | Sinh tổ hợp chập k của 19(đặc trưng), mục đích cung cấp cho hàm NhungDacTrungTotNhat | knn.pyx | Nguyễn Tiến Đạt |
| 2 | NhungDacTrungTotNhat  Input: k  Output: Những đặc trưng tốt nhất chưa bao gồm tất cả các đặc trưng | Mục đích là tìm những đặc trưng tốt nhất trong khoảng nhỏ vì thực thi hết là không thể nên chưa bao gồm tất cả các đặc trưng, hàm cung cấp cho hàm timNhungDacTrungTotNhat. | knn.pyx | Nguyễn Tiến Đạt |
| 3 | cross\_Validation  Input: n\_neighbors,X,Y  n\_neighbors: số hàng xóm.  X: là tập dữ liệu.  Y: là tập label của X. | Cung cấp đánh gía về độ chính xác của KNN trên tập dữ liệu X ở đây em cho cv=10 chia thành 10 tập. | knn.pyx | Nguyễn Tiến Đạt |
| 4 | n\_neighbors  Input: k  Output: list các phần tử ứng với k | Cung cấp list các n\_neighbors cho hàm xuLy\_knn\_CoLoc và xuLy\_knn\_KhongLoc ở đây em chia thành 5 giá trị k từ 0 đến 74 vì mục đích chính chia nhỏ khoảng lại thì Grid Search trong các hàm đó sẽ chính xác hơn. | knn.pyx | Nguyễn Tiến Đạt |
| 5 | timF\_CoLoc  Input: n, good\_KNN\_CoLoc  Output: good\_KNN\_CoLoc | Mục đích là dùng Grid Search để tìm ra những parameters tốt nhất cho KNN ở đây em cho chạy tìm 2 parameters là n\_neighbors, weights vì thời gian tính sẽ hạn chế hơn và những parameters phụ thuộc nhau như p, metric đều không ảnh hưởng lớn đến kết quả, hàm này chỉ dành cho tập dữ liệu có lọc của DAO cung cấp, kết quả trả ra là đối tượng good\_KNN\_CoLoc lưu lại những parameters tốt nhất, hàm này cung cấp cho hàm xuLy\_knn\_CoLoc. | knn.pyx | Nguyễn Tiến Đạt |
| 6 | timF\_KhongLoc  Input: n,good\_KNN\_KhongLoc  Output: good\_KNN\_KhongLoc | Mục đích là dùng Grid Search để tìm ra những parameters tốt nhất cho KNN ở đây em cho chạy tìm 2 parameters là n\_neighbors, weights vì thời gian tính sẽ hạn chế hơn và những parameters phụ thuộc nhau như p, metric đều không ảnh hưởng lớn đến kết quả, hàm này chỉ dành cho tập dữ liệu không lọc của DAO cung cấp, kết quả trả ra là đối tượng good\_KNN\_KhongLoc lưu lại những parameters tốt nhất, hàm này cung cấp cho hàm xuLy\_knn\_KhongLoc. | knn.pyx | Nguyễn Tiến Đạt |
| 7 | xuLy\_knn\_CoLoc  Input: Không.  Output: good\_KNN\_CoLoc | Mục đích là gọi hàm timF\_CoLoc, n\_neighbors để truyền list n\_neighbors mục đích là Grid Search ở những khoảng nhỏ sẽ ra độ chính xác tốt hơn, chỉ dùng cho tập dữ liệu có lọc, hàm này cung cấp cho hàm ketQua. | knn.pyx | Nguyễn Tiến Đạt |
| 8 | xuLy\_knn\_KhongLoc  Input: Không.  Output: good\_KNN\_KhongLoc. | Mục đích là gọi hàm timF\_KhongLoc, n\_neighbors để truyền list n\_neighbors mục đích là Grid Search ở những khoảng nhỏ sẽ ra độ chính xác tốt hơn, chỉ dùng cho tập dữ liệu không lọc, hàm này cung cấp cho hàm ketQua. | knn.pyx | Nguyễn Tiến Đạt |
| 9 | ketQua  Input: k.  Output: Không. | Hàm này là hàm hiện kết quả cuối cùng như cross\_Validation hay kết quả tính F, n\_neighbors, weights sau khi đã Grid Search. Ở đây có k đầu vào là nếu k=0 cho tập dữ liệu có lọc và k=1 cho tập dữ liệu không lọc. | knn.pyx | Nguyễn Tiến Đạt |
| 10 | traSoThuTu  Input: tên đặc trưng  Output: index của đặc trưng đó trong tập dữ liệu. | Mục đích là trả về chỉ mục vị trí trong dataset. | knn.pyx | Nguyễn Tiến Đạt |
| 11 | Ve2D  Input: chonBoDuLieu:   * Là tập dữ liệu vẽ nếu 0 tập có lọc,1 tập không lọc.   mangCacDacTrungVe:   * Là mảng các đặc trưng cần vẽ với None là tất cả.   soLuongDiemVe:   * Là số lượng điểm vẽ. | Cung cấp hàm vẽ 2 chiều cho các đặc trưng. | knn.pyx | Nguyễn Tiến Đạt |
| 12 | Ve3D  Input: chonBoDuLieu:   * Là tập dữ liệu vẽ nếu 0 tập có lọc,1 tập không lọc.   mangCacDacTrungVe:   * Là mảng các đặc trưng cần vẽ với None là tất cả.   soLuongDiemVe:   * Là số lượng điểm vẽ. | Cung cấp hàm vẽ 3 chiều cho các đặc trưng. | knn.pyx | Nguyễn Tiến Đạt |
| 13 | help  Input: Không.  Output: Không | Cung cấp cú pháp để dễ sử dụng. | knn.pyx | Nguyễn Tiến Đạt |
| 14 | timNhungDacTrungTotNhat  Input: Không.  Output: đối tượng nhungDacTrungTotNhat. | Mục đích là tìm những đặc trưng tốt nhất trong khoảng nhỏ vì thực thi hết là không thể và so sánh với tất cả các đặc trưng. | knn.pyx | Nguyễn Tiến Đạt |

Bảng 2: Lớp

| STT | Tên lớp | Tên sinh viên phụ trách | Mục đích của lớp |
| --- | --- | --- | --- |
| 1 | nhungDacTrungTotNhat | Nguyễn Tiến Đạt | Lưu lại những đặc trưng tốt cho thuật toán sau khi sinh tổ hợp và tính toán. |
| 2 | class good\_KNN | Nguyễn Tiến Đạt | Lưu lại những parameters tốt nhất sau khi chạy Grid Search. |

#### Mã nguồn

##### Các thư viện và trỏ liên kết tới DAO

# coding=utf-8

import sys

sys.path.append('../DAO/')

# tap du lieu su dung

import dataProcessing as dp

# thu vien ve cua python

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D

# thu vien sklearn cho ho tro knn

from sklearn.neighbors import KNeighborsClassifier

#Đánh giá

from sklearn.metrics import recall\_score

from sklearn.metrics import precision\_score

#xu ly matrix

import numpy as np

#tap du lieu training va testing

# chia tap du lieu ban dau thanh 2 tap la training va testing

from sklearn.model\_selection import train\_test\_split

from sklearn.model\_selection import learning\_curve, GridSearchCV

#sinh tổ hợp

from itertools import permutations

#cross-validation

from sklearn.model\_selection import cross\_val\_score

##### Hàm sinhToHop: Sinh tổ hợp chập k của 19(đặc trưng), mục đích cung cấp cho hàm NhungDacTrungTotNhat

#sinh to hop

def sinhToHop(k):

perm = permutations(['tuoi','nghe\_nghiep','hon\_nhan','hoc\_van','co\_the\_tin\_dung',

'co\_nha\_o','vay\_ca\_nhan','kenh\_lien\_lac','thang\_lien\_lac',

'ngay\_lien\_lac','thoi\_luong\_lien\_lac','so\_luong\_lien\_lac',

'ngay','so\_luong\_lien\_lac\_truoc\_day','ket\_qua\_lan\_truoc',

'ti\_le\_thay\_doi\_viec\_lam','CPI','CCI','lai\_suat\_3thang',

'so\_luong\_nhan\_vien'],k)

array=[]

for i in list(perm):

array.append(i)

return array

##### Lớp nhungDacTrungTotNhat: Lưu lại những đặc trưng tốt cho thuật toán sau khi sinh tổ hợp và tính toán

class nhungDacTrungTotNhat:

def \_\_init\_\_(seft,array,F):

seft.array=array

seft.F=F

##### Hàm NhungDacTrungTotNhat: Mục đích là tìm những đặc trưng tốt nhất trong khoảng nhỏ vì thực thi hết là không thể nên chưa bao gồm tất cả các đặc trưng, hàm cung cấp cho hàm timNhungDacTrungTotNhat

def NhungDacTrungTotNhat(k):

NDTTN=nhungDacTrungTotNhat(0,0)

for i in range(100):

x\_train\_KhongLoc, x\_test\_KhongLoc, y\_train\_KhongLoc, y\_test\_KhongLoc=train\_test\_split(

dp.data\_khongLoc(0,sinhToHop(k)[i]),dp.data\_khongLoc(1,None),test\_size=0.2)

clf=KNeighborsClassifier(n\_neighbors=13).fit(x\_train\_KhongLoc,y\_train\_KhongLoc)

precision= precision\_score(y\_test\_KhongLoc,clf.predict(x\_test\_KhongLoc), average='weighted')

recall= recall\_score(y\_test\_KhongLoc,clf.predict(x\_test\_KhongLoc), average='weighted')

F\_KhongLoc=(2\*precision\*recall)/(precision+recall)

if F\_KhongLoc>NDTTN.F:

NDTTN.array=sinhToHop(k)[i]

NDTTN.F=F\_KhongLoc

return NDTTN

##### Hàm timNhungDacTrungTotNhat: Mục đích là tìm những đặc trưng tốt nhất trong khoảng nhỏ vì thực thi hết là không thể và so sánh với tất cả các đặc trưng

def timNhungDacTrungTotNhat():

NDTTN=nhungDacTrungTotNhat(0,0)

for i in range(3,7):

tmp=NhungDacTrungTotNhat(i)

if tmp.F>NDTTN.F:

NDTTN.array=tmp.array

NDTTN.F=tmp.F

x\_train\_KhongLoc, x\_test\_KhongLoc, y\_train\_KhongLoc, y\_test\_KhongLoc=train\_test\_split(

dp.data\_khongLoc(0,None),dp.data\_khongLoc(1,None),test\_size=0.2)

clf=KNeighborsClassifier(n\_neighbors=13).fit(x\_train\_KhongLoc,y\_train\_KhongLoc)

precision= precision\_score(y\_test\_KhongLoc,clf.predict(x\_test\_KhongLoc), average='weighted')

recall= recall\_score(y\_test\_KhongLoc,clf.predict(x\_test\_KhongLoc), average='weighted')

F\_KhongLoc=(2\*precision\*recall)/(precision+recall)

if F\_KhongLoc>NDTTN.F:

NDTTN.array=None

NDTTN.F=F\_KhongLoc

return NDTTN

##### Tập dữ liệu sử dụng sinh tổ hợp

#tap du lieu

# x\_train\_CoLoc, x\_test\_CoLoc, y\_train\_CoLoc, y\_test\_CoLoc=train\_test\_split(

# dp.data\_CoLoc(0,timNhungDacTrungTotNhat().array),

# dp.data\_CoLoc(1,None),

# test\_size=0.2)

#

# x\_train\_KhongLoc, x\_test\_KhongLoc, y\_train\_KhongLoc, y\_test\_KhongLoc=train\_test\_split(

# dp.data\_khongLoc(0,timNhungDacTrungTotNhat().array),

# dp.data\_khongLoc(1,None),test\_size=0.2)

##### Tập dữ liệu không sử dụng sinh tổ hợp

x\_train\_CoLoc, x\_test\_CoLoc, y\_train\_CoLoc, y\_test\_CoLoc=train\_test\_split(

dp.data\_CoLoc(0,None),

dp.data\_CoLoc(1,None),

test\_size=0.2,random\_state=1)

x\_train\_KhongLoc, x\_test\_KhongLoc, y\_train\_KhongLoc, y\_test\_KhongLoc=train\_test\_split(

dp.data\_khongLoc(0,None),

dp.data\_khongLoc(1,None),test\_size=0.2,random\_state=1)

##### Hàm cross\_Validation: Cung cấp đánh gía về độ chính xác của KNN trên tập dữ liệu X ở đây em cho cv=10 chia thành 10 tập

#cross-validation

def cross\_Validation(n\_neighbors,X,Y):

knn\_cv = KNeighborsClassifier(n\_neighbors=n\_neighbors)

cv\_scores = cross\_val\_score(knn\_cv, X,Y, cv=10)

print("\n cross\_Validation:\n")

print(cv\_scores)

print("cv\_scores mean:{}".format(np.mean(cv\_scores)))

##### Lớp class good\_KNN: Lưu lại những parameters tốt nhất sau khi chạy Grid Search

# object luu param tot nhat

class good\_KNN:

def \_\_init\_\_(seft,weights,n\_neighbors,F):

seft.weights=weights

seft.n\_neighbors=n\_neighbors

seft.F=F

##### Hàm n\_neighbors: Cung cấp list các n\_neighbors cho hàm xuLy\_knn\_CoLoc và xuLy\_knn\_KhongLoc ở đây em chia thành 5 giá trị k từ 0 đến 74 vì mục đích chính chia nhỏ khoảng lại thì Grid Search trong các hàm đó sẽ chính xác hơn

def n\_neighbors(k):

n\_neighbors=[]

if k==0:

for i in range(15):

if i%2 !=0:

n\_neighbors.append(i)

elif k==1:

for i in range(15,30):

if i%2 !=0:

n\_neighbors.append(i)

elif k==2:

for i in range(30,42):

if i%2 !=0:

n\_neighbors.append(i)

elif k==3:

for i in range(42,50):

if i%2 !=0:

n\_neighbors.append(i)

elif k==4:

for i in range(52,56):

if i%2 !=0:

n\_neighbors.append(i)

elif k==5:

for i in range(56,74):

if i%2 !=0:

n\_neighbors.append(i)

return n\_neighbors

##### Hàm timF\_CoLoc: Mục đích là dùng Grid Search để tìm ra những parameters tốt nhất cho KNN ở đây em cho chạy tìm 2 parameters là n\_neighbors, weights vì thời gian tính sẽ hạn chế hơn và những parameters phụ thuộc nhau như p, metric đều không ảnh hưởng lớn đến kết quả, hàm này chỉ dành cho tập dữ liệu có lọc của DAO cung cấp, kết quả trả ra là đối tượng good\_KNN\_CoLoc lưu lại những parameters tốt nhất, hàm này cung cấp cho hàm xuLy\_knn\_CoLoc

# tim F cho tap du lieu co loc

def timF\_CoLoc(n,good\_KNN\_CoLoc):

weights=['uniform','distance']

knn = KNeighborsClassifier()

param\_grid = dict(n\_neighbors=n, weights=weights)

grid = GridSearchCV(knn, param\_grid, cv=10, scoring='accuracy')

grid.fit(x\_test\_CoLoc, y\_test\_CoLoc)

# du lieu co loc

clf=KNeighborsClassifier(n\_neighbors=grid.best\_estimator\_.n\_neighbors,weights=grid.best\_estimator\_.weights).fit(x\_train\_CoLoc,y\_train\_CoLoc)

precision= precision\_score(y\_test\_CoLoc,clf.predict(x\_test\_CoLoc), average='weighted')

recall= recall\_score(y\_test\_CoLoc,clf.predict(x\_test\_CoLoc), average='weighted')

F\_CoLoc=(2\*precision\*recall)/(precision+recall)

#so sanh

if F\_CoLoc>good\_KNN\_CoLoc.F:

good\_KNN\_CoLoc=good\_KNN(grid.best\_estimator\_.weights,

grid.best\_estimator\_.n\_neighbors,F\_CoLoc)

return good\_KNN\_CoLoc

##### Hàm timF\_KhongLoc: Mục đích là dùng Grid Search để tìm ra những parameters tốt nhất cho KNN ở đây em cho chạy tìm 2 parameters là n\_neighbors, weights vì thời gian tính sẽ hạn chế hơn và những parameters phụ thuộc nhau như p, metric đều không ảnh hưởng lớn đến kết quả, hàm này chỉ dành cho tập dữ liệu không lọc của DAO cung cấp, kết quả trả ra là đối tượng good\_KNN\_KhongLoc lưu lại những parameters tốt nhất, hàm này cung cấp cho hàm xuLy\_knn\_KhongLoc

# tim F cho tap du lieu khong loc

def timF\_KhongLoc(n,good\_KNN\_KhongLoc):

weights=['uniform','distance']

knn = KNeighborsClassifier()

param\_grid = dict(n\_neighbors=n, weights=weights)

grid = GridSearchCV(knn, param\_grid, cv=10, scoring='accuracy')

grid.fit(x\_test\_KhongLoc, y\_test\_KhongLoc)

#du lieu khong loc

clf=KNeighborsClassifier(n\_neighbors=grid.best\_estimator\_.n\_neighbors,weights=grid.best\_estimator\_.weights).fit(x\_train\_KhongLoc,y\_train\_KhongLoc)

precision= precision\_score(y\_test\_KhongLoc,clf.predict(x\_test\_KhongLoc), average='weighted')

recall= recall\_score(y\_test\_KhongLoc,clf.predict(x\_test\_KhongLoc), average='weighted')

F\_KhongLoc=(2\*precision\*recall)/(precision+recall)

#so sanh

if F\_KhongLoc>good\_KNN\_KhongLoc.F:

good\_KNN\_KhongLoc=good\_KNN(grid.best\_estimator\_.weights,

grid.best\_estimator\_.n\_neighbors,F\_KhongLoc)

return good\_KNN\_KhongLoc

##### Hàm xuLy\_knn\_CoLoc: Mục đích là gọi hàm timF\_CoLoc, n\_neighbors để truyền list n\_neighbors mục đích là Grid Search ở những khoảng nhỏ sẽ ra độ chính xác tốt hơn, chỉ dùng cho tập dữ liệu có lọc, hàm này cung cấp cho hàm ketQua

#Xu ly tinh toan cho tap du lieu co loc

def xuLy\_knn\_CoLoc():

good\_KNN\_CoLoc=good\_KNN(0,0,0)

good\_KNN\_CoLoc=timF\_CoLoc(n\_neighbors(0),good\_KNN\_CoLoc)

good\_KNN\_CoLoc=timF\_CoLoc(n\_neighbors(1),good\_KNN\_CoLoc)

good\_KNN\_CoLoc=timF\_CoLoc(n\_neighbors(2),good\_KNN\_CoLoc)

good\_KNN\_CoLoc=timF\_CoLoc(n\_neighbors(3),good\_KNN\_CoLoc)

good\_KNN\_CoLoc=timF\_CoLoc(n\_neighbors(4),good\_KNN\_CoLoc)

good\_KNN\_CoLoc=timF\_CoLoc(n\_neighbors(5),good\_KNN\_CoLoc)

return good\_KNN\_CoLoc

##### Hàm xuLy\_knn\_KhongLoc: Mục đích là gọi hàm timF\_KhongLoc, n\_neighbors để truyền list n\_neighbors mục đích là Grid Search ở những khoảng nhỏ sẽ ra độ chính xác tốt hơn, chỉ dùng cho tập dữ liệu không lọc, hàm này cung cấp cho hàm ketQua

#Xu ly tinh toan cho tap du lieu khong loc

def xuLy\_knn\_KhongLoc():

good\_KNN\_KhongLoc=good\_KNN(0,0,0)

good\_KNN\_KhongLoc=timF\_KhongLoc(n\_neighbors(0),good\_KNN\_KhongLoc)

good\_KNN\_KhongLoc=timF\_KhongLoc(n\_neighbors(1),good\_KNN\_KhongLoc)

good\_KNN\_KhongLoc=timF\_KhongLoc(n\_neighbors(2),good\_KNN\_KhongLoc)

good\_KNN\_KhongLoc=timF\_KhongLoc(n\_neighbors(3),good\_KNN\_KhongLoc)

good\_KNN\_KhongLoc=timF\_KhongLoc(n\_neighbors(4),good\_KNN\_KhongLoc)

good\_KNN\_KhongLoc=timF\_KhongLoc(n\_neighbors(5),good\_KNN\_KhongLoc)

return good\_KNN\_KhongLoc

##### Hàm ketQua: Hàm này là hàm hiện kết quả cuối cùng như cross\_Validation hay kết quả tính F, n\_neighbors, weights sau khi đã Grid Search. Ở đây có k đầu vào là nếu k=0 cho tập dữ liệu có lọc và k=1 cho tập dữ liệu không lọc

def ketQua(k):

if k==0:

cross\_Validation(13,dp.data\_CoLoc(0,None),dp.data\_CoLoc(1,None))

x= xuLy\_knn\_CoLoc()

print("Du lieu co loc: ")

print("n\_neighbors=%s"%x.n\_neighbors)

print("weights=%s"%x.weights)

print("F=%s"%x.F)

elif k==1:

cross\_Validation(13,dp.data\_khongLoc(0,None),dp.data\_khongLoc(1,None))

x= xuLy\_knn\_KhongLoc()

print("Du lieu khong loc: ")

print("n\_neighbors=%s"%x.n\_neighbors)

print("weights=%s"%x.weights)

print("F=%s"%x.F)

##### Hàm traSoThuTu: Mục đích là trả về chỉ mục vị trí trong dataset

#tra ve vi tri cac dac trung cung cap cho ham ve

def traSoThuTu(ten):

t=['tuoi','nghe\_nghiep','hon\_nhan','hoc\_van','co\_the\_tin\_dung',

'co\_nha\_o','vay\_ca\_nhan','kenh\_lien\_lac','thang\_lien\_lac',

'ngay\_lien\_lac','thoi\_luong\_lien\_lac','so\_luong\_lien\_lac',

'ngay','so\_luong\_lien\_lac\_truoc\_day','ket\_qua\_lan\_truoc',

'ti\_le\_thay\_doi\_viec\_lam','CPI','CCI','lai\_suat\_3thang',

'so\_luong\_nhan\_vien']

for i in range(len(t)):

if t[i]==ten:

return i;

return -1;

##### Hàm Ve2D: Cung cấp hàm vẽ 2 chiều cho các đặc trưng

#ve 2 dac trung trong cac dac trung

def Ve2D(chonBoDuLieu,mangCacDacTrungVe,soLuongDiemVe):

if soLuongDiemVe>len(x\_train\_CoLoc) and chonBoDuLieu==0:

return None

if soLuongDiemVe>len(x\_train\_KhongLoc) and chonBoDuLieu==1:

return None

if len(mangCacDacTrungVe)!=2:

return False

m=[]

for i in mangCacDacTrungVe:

if traSoThuTu(i)!=-1:

m.append(traSoThuTu(i))

mangVe0=[]

mangVe1=[]

if chonBoDuLieu==0:

for i in range(soLuongDiemVe):

if y\_train\_CoLoc[i] ==0:

mangVe0.append([x\_train\_CoLoc[i][m[0]],x\_train\_CoLoc[i][m[1]],y\_train\_CoLoc[i]])

elif y\_train\_CoLoc[i]==1:

mangVe1.append([x\_train\_CoLoc[i][m[0]],x\_train\_CoLoc[i][m[1]],y\_train\_CoLoc[i]])

elif chonBoDuLieu==1:

for i in range(soLuongDiemVe):

if y\_train\_KhongLoc[i]==0:

mangVe0.append([x\_train\_KhongLoc[i][m[0]],x\_train\_KhongLoc[i][m[1]],y\_train\_KhongLoc[i]])

elif y\_train\_KhongLoc[i]==1:

mangVe1.append([x\_train\_KhongLoc[i][m[0]],x\_train\_KhongLoc[i][m[1]],y\_train\_KhongLoc[i]])

mangVe0=np.array(mangVe0)

mangVe1=np.array(mangVe1)

plt.scatter(mangVe0[:,0],mangVe0[:,1],marker="x",label="Thất bại",s=100)

plt.scatter(mangVe1[:,0],mangVe1[:,1],marker="\*",label="Thành công",s=100)

plt.xlabel(mangCacDacTrungVe[0])

plt.ylabel(mangCacDacTrungVe[1])

plt.title("Biểu đồ phân 2 lớp sử dụng knn")

plt.legend(loc='upper left')

plt.show()

##### Hàm Ve3D: Cung cấp hàm vẽ 3 chiều cho các đặc trưng

def Ve3D(chonBoDuLieu,mangCacDacTrungVe,soLuongDiemVe):

if soLuongDiemVe>len(x\_train\_CoLoc) and chonBoDuLieu==0:

return None

if soLuongDiemVe>len(x\_train\_KhongLoc) and chonBoDuLieu==1:

return None

if len(mangCacDacTrungVe)!=3:

return False

m=[]

for i in mangCacDacTrungVe:

if traSoThuTu(i)!=-1:

m.append(traSoThuTu(i))

mangVe0=[]

mangVe1=[]

if chonBoDuLieu==0:

for i in range(soLuongDiemVe):

if y\_train\_CoLoc[i] ==0:

mangVe0.append([x\_train\_CoLoc[i][m[0]],x\_train\_CoLoc[i][m[1]],x\_train\_CoLoc[i][m[2]],y\_train\_CoLoc[i]])

elif y\_train\_CoLoc[i]==1:

mangVe1.append([x\_train\_CoLoc[i][m[0]],x\_train\_CoLoc[i][m[1]],x\_train\_CoLoc[i][m[2]],y\_train\_CoLoc[i]])

elif chonBoDuLieu==1:

for i in range(soLuongDiemVe):

if y\_train\_KhongLoc[i]==0:

mangVe0.append([x\_train\_KhongLoc[i][m[0]],x\_train\_KhongLoc[i][m[1]],x\_train\_KhongLoc[i][m[2]],y\_train\_KhongLoc[i]])

elif y\_train\_KhongLoc[i]==1:

mangVe1.append([x\_train\_KhongLoc[i][m[0]],x\_train\_KhongLoc[i][m[1]],x\_train\_KhongLoc[i][m[2]],y\_train\_KhongLoc[i]])

mangVe0=np.array(mangVe0)

mangVe1=np.array(mangVe1)

fig = plt.figure()

ax = fig.add\_subplot(111, projection='3d')

ax.scatter(mangVe0[:,0],mangVe0[:,1],mangVe0[:,2],marker="x",label="Thất bại",s=100)

ax.scatter(mangVe1[:,0],mangVe1[:,1],mangVe1[:,2],marker="\*",label="Thành công",s=100)

ax.set\_xlabel(mangCacDacTrungVe[0])

ax.set\_ylabel(mangCacDacTrungVe[1])

ax.set\_zlabel(mangCacDacTrungVe[2])

plt.title("Biểu đồ phân 2 lớp sử dụng knn")

plt.legend(loc='lower left')

plt.show()

##### Hàm help: Cung cấp cú pháp để dễ sử dụng

def help():

print("\nhelp:\t\t.Ve2D(chonBoDuLieu,mangCacDacTrungVe,soLuongDiemVe)\n")

print("\t\t.Ve3D(chonBoDuLieu,mangCacDacTrungVe,soLuongDiemVe)\n")

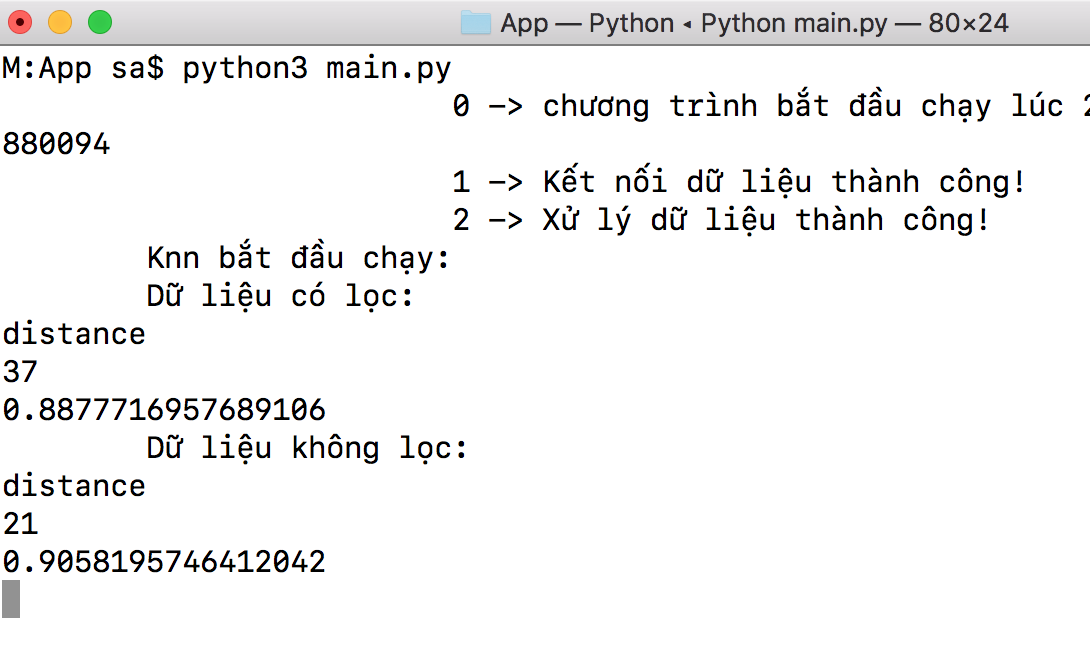
print("\t\t\tchonBoDuLieu: 0 -> du lieu co Loc\n")

print("\t\t\t 1 -> du lieu khong Loc\n")

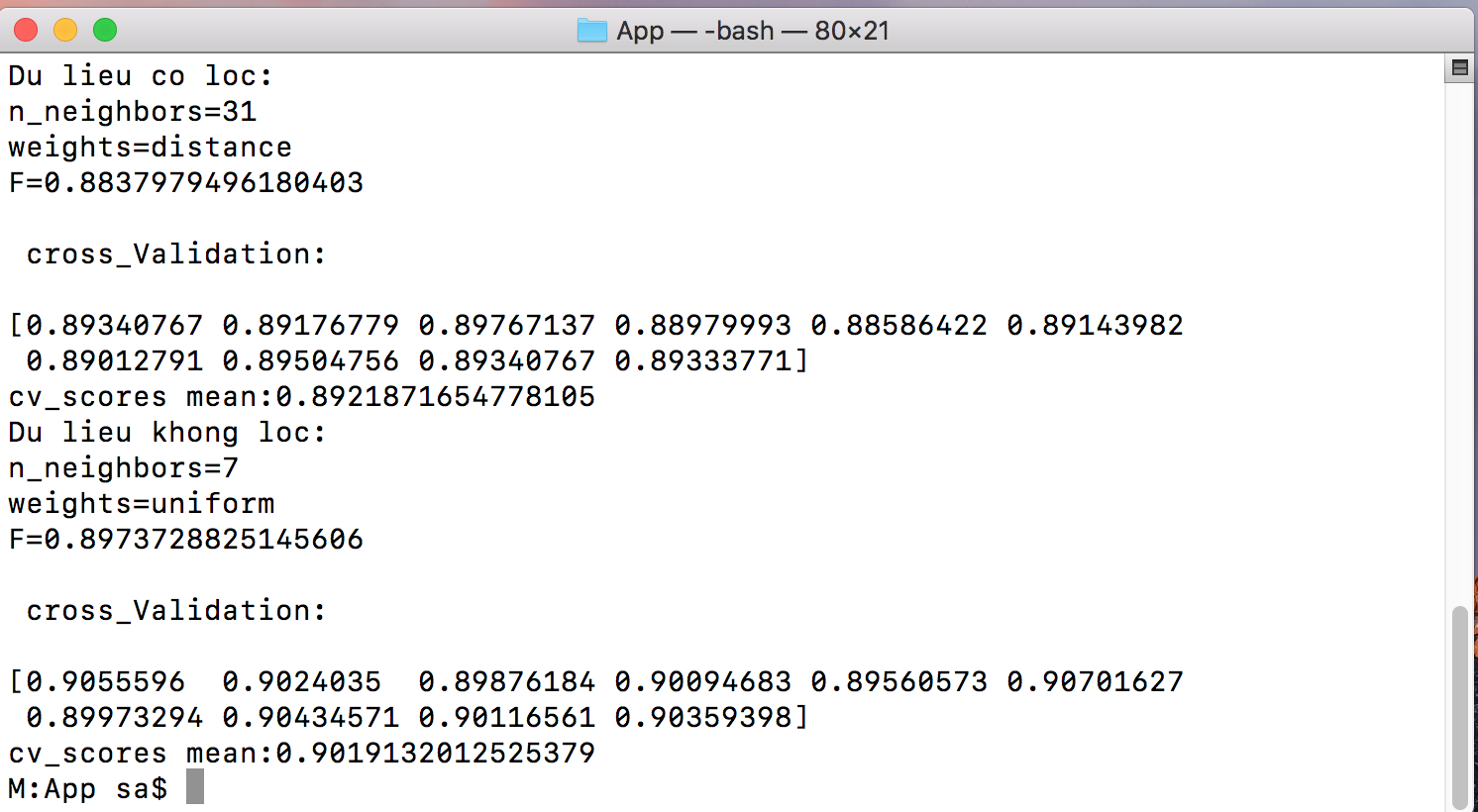
print("\t\t.ketQua(k) tra ve gia tri F voi k: 0 -> du lieu co Loc\n")

print("\t\t k: 1 -> du lieu khong Loc\n")

#### Đánh giá

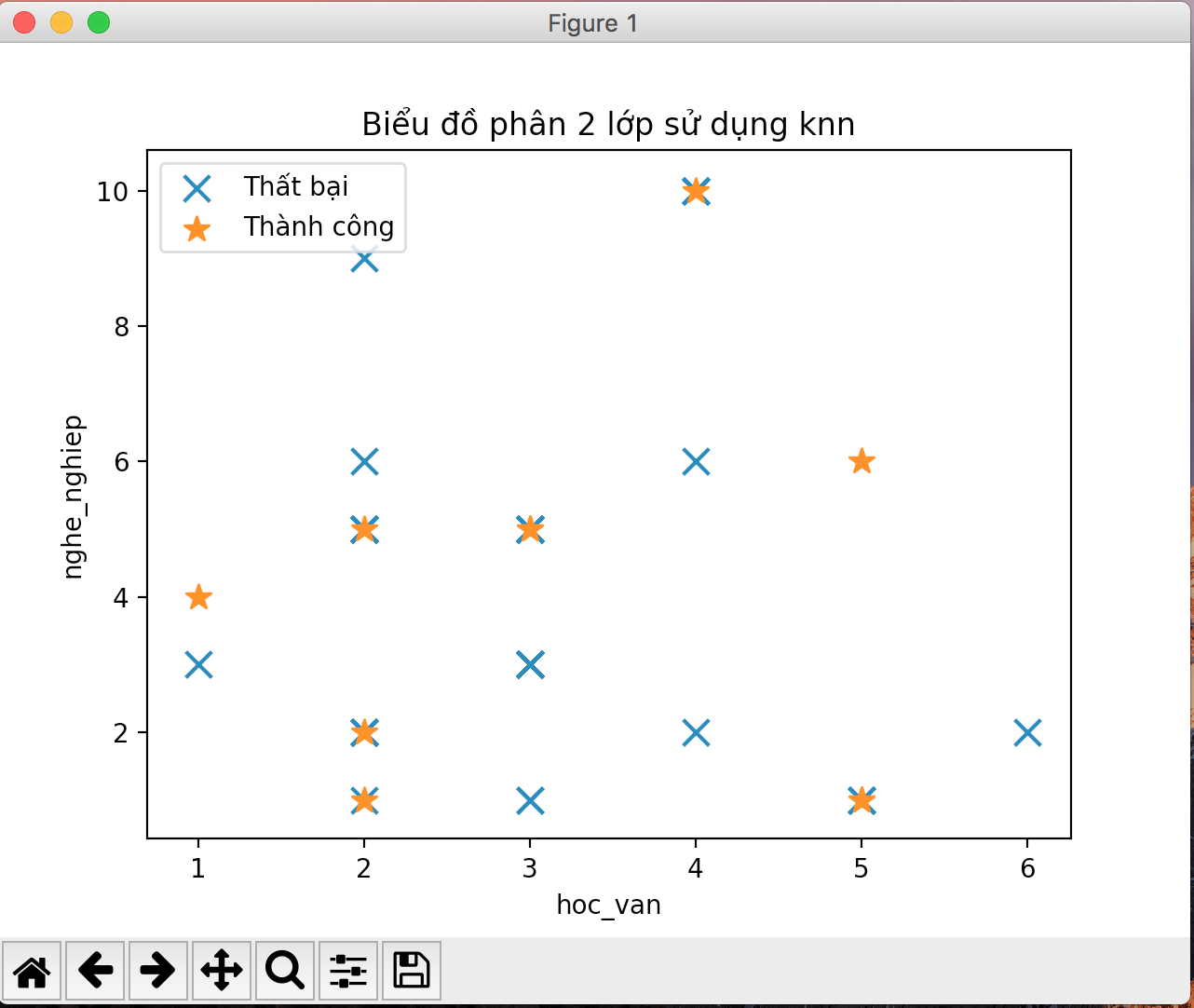
Hình 6: Kết quả knn1.

Ở hình 6 em dùng 2 tập dữ liệu là có lọc và không lọc unknown có sử dụng Grid Search cho 2 parameters là n\_neighbors, weights với cv=10 thì với tập không lọc nó lại cho kết quả cao hơn tới 90.5 % với 2 tập này em lấy hết đặc trưng đem đi training nhưng kết quả như hình 6 ở đây không cao lắm em nghĩ có có 3 lý do sau:

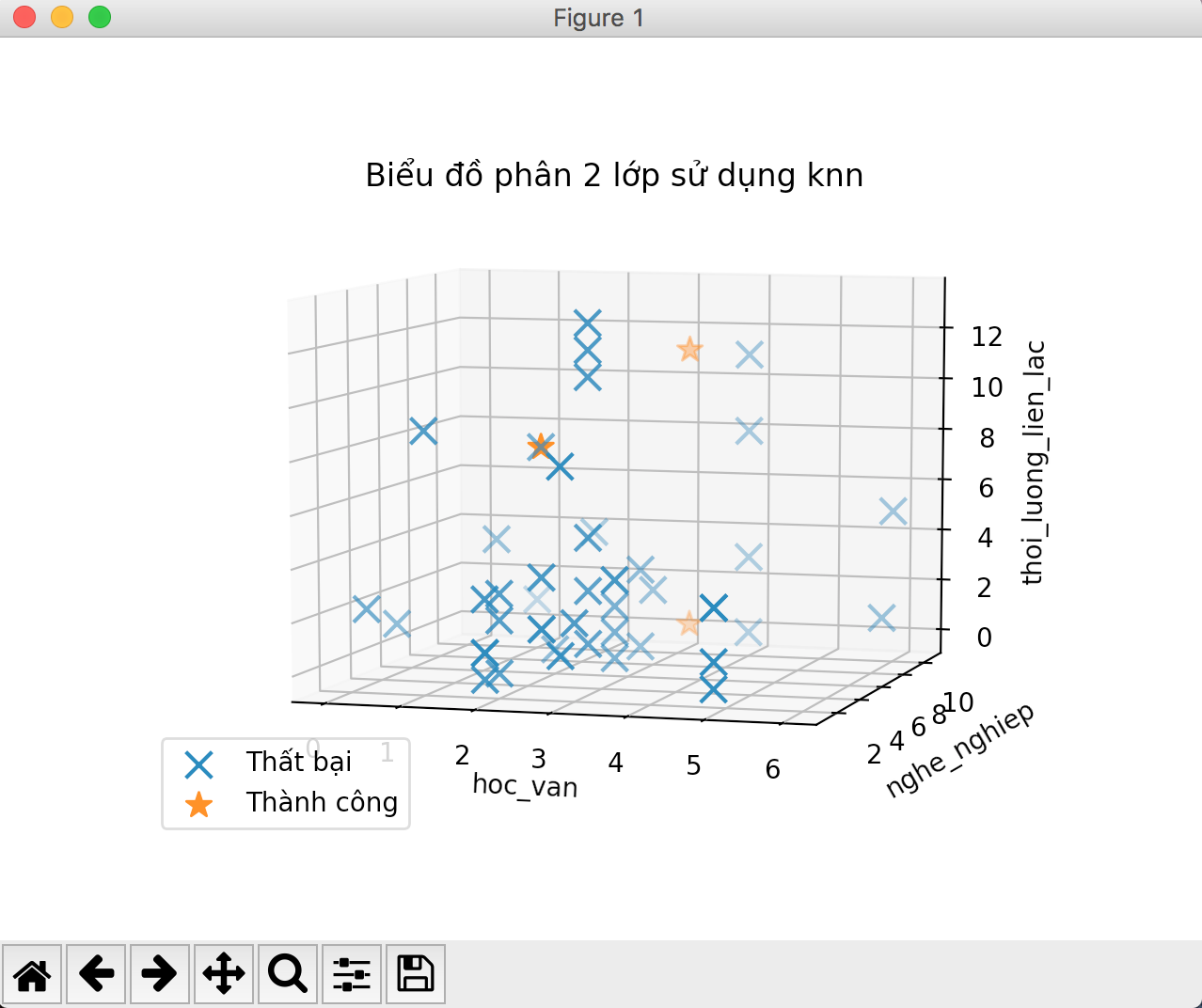
* Thứ 1: Em chưa dùng phương pháp sinh tổ hợp để tìm đặc trưng tốt, em có xây dựng nó trong chương trình chỉ có dùng mức nhỏ nhưng chưa đạt được kết quả tốt vì chỉ chạy ở mức dưới tổ hợp chập 5 của 19 khi đi so với lấy đủ 19 đặc trưng sẽ thấp hơn nhưng nếu nâng mức tổ hợp lên thì máy tính của em hiện không có khả năng xử lý.
* Thứ 2: Là trước đó em có để nguyên dữ liệu số cụ thể là 2 cột so\_luong\_nhan\_vien và thoi\_luong\_lien\_lac chỉ biến đổi dữ liệu chữ thì kết quả lại trên > 92% nhưng mất rất nhiều giờ em không có lưu lại. Còn khi biến đổi dữ liệu cho 2 cột này em dùng 1 phương trình hàm y=(x\*13.999-13)/999 nên có lẽ dữ liệu đã bị nhiễu phần nào.
* Thứ 3: Theo em nghĩ là bộ dữ liệu này không phù hợp cho KNN nó chỉ ở mức này thôi.

Hình 7: Cross validation.

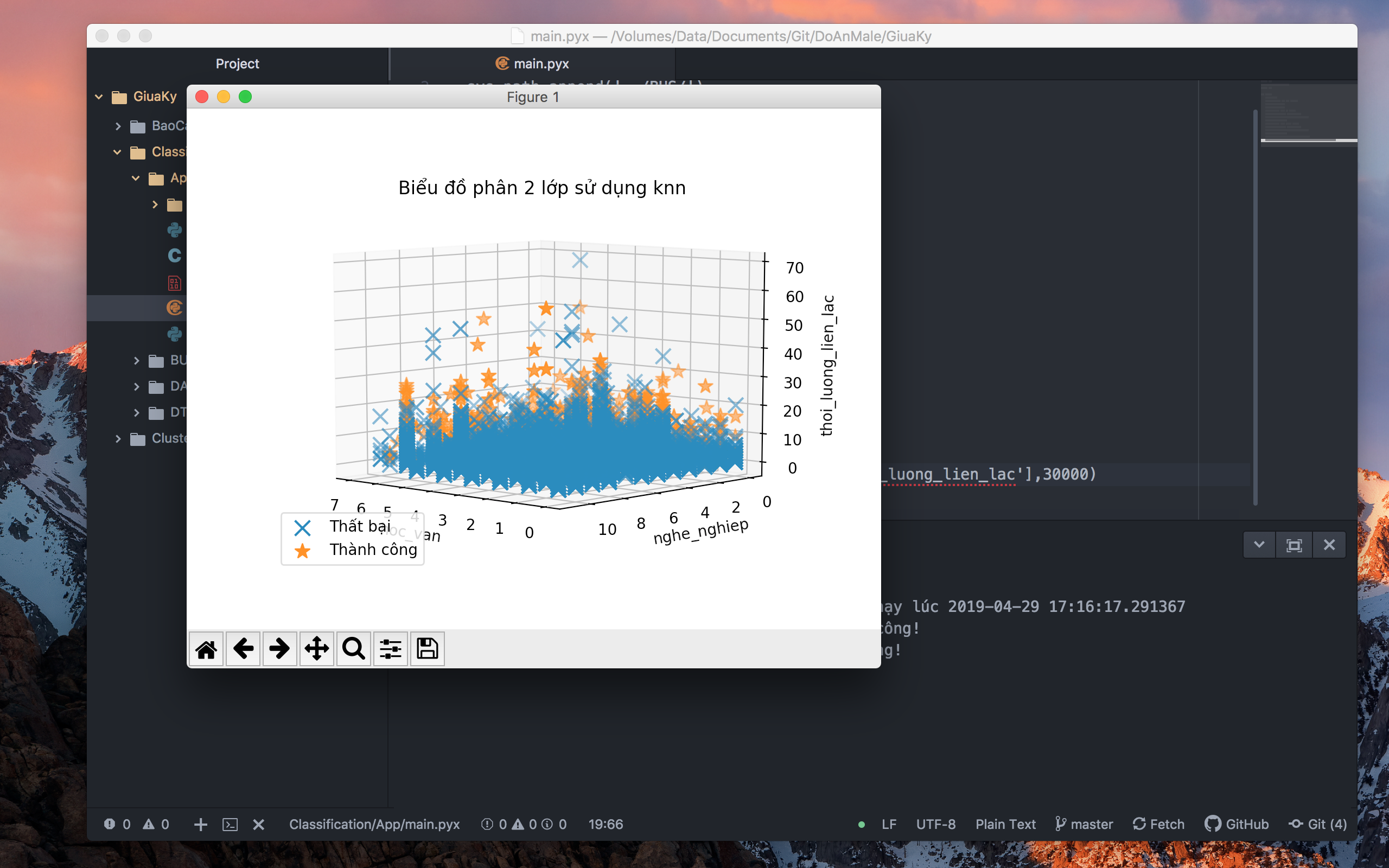
Ở hình 7 em có sử dụng cross validation với cv=10 cho cả 2 tập dữ liệu thì thấy mức đánh giá của tập không lọc unknown là tầm 90% và của tập có lọc unknown là 89%.



Hình 8: Vẽ 2d KNN.



Hình 9: Vẽ 3d KNN1.



Hình 10: Vẽ 3d KNN2.

Hình 8, 9, 10 là những hình về hàm vẽ 2d và 3d cho các đặc trưng xây dựng cho KNN.

## TẦNG APP

### CHỨC NĂNG

Là tầng cấu hình, cài đặt, biên dịch và gọi nghiệp vụ xử lý của tầng BUS hiển thị kết quả ra ngoài như giao diện trực quan hay command line interface.

### BẢNG HÀM

Bảng 1: Hàm

| STT | Tên hàm | Mục đích của hàm | File lưu trữ | Tên sinh viên phụ trách |
| --- | --- | --- | --- | --- |
| 1 | KNN  Input: Không.  Output: Không. | Gọi các nghiệp vụ của Knn dưới tầng BUS như ketQua ứng với tập dữ liệu có lọc, không lọc unknown và các hàm vẽ ve2D, ve3D | main.pyx | Nguyễn Tiến Đạt |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

### MÃ NGUỒN

#### File setup.py

##### Chức năng

Mục đích để biên dịch tất cả file .pyx ra thành các module để cho file index.py có thể sử dụng.

##### Khai báo thư viện Cython sử dụng:

from distutils.core import setup

from Cython.Build import cythonize

##### Trỏ đường dẫn đến các file .pyx của các tầng để biên dịch ra các module(file .o) cung cấp cho file index.py sử dụng:

setup(ext\_modules=cythonize('main.pyx'))

setup(ext\_modules=cythonize('../BUS/knn.pyx'))

setup(ext\_modules=cythonize('../BUS/naiveBayes.pyx'))

setup(ext\_modules=cythonize('../BUS/svm.pyx'))

setup(ext\_modules=cythonize('../BUS/svmKernel.pyx'))

setup(ext\_modules=cythonize('../DAO/dataProcessing.pyx'))

setup(ext\_modules=cythonize('../DTO/DataAdapter.pyx'))

#### File main.pyx

##### Chức năng

Mục đích để sử dụng các nghiệp vụ từ tầng BUS và triển khai thành các hàm cung cấp cho file index.py thực thi.

#### File index.py

##### Chức năng

Là file tương tác trực tiếp với người dùng cuối thông qua gọi các hàm từ file main.pyx. Các file main.pyx và index.py không phải là các file thừa thải vì để biên dịch theo cấu trúc Cython chúng ta cần 1 file main.py để kết nối và file index.py để sử dụng các module đã được tạo ra.

##### Liên kết tới main.o khi mới tạo ra

import main as m

##### Sử dụng các hàm trong main.o

m.KNN()

# PHẦN 3: CẤU TRÚC CHO PHÂN CỤM

## TẦNG DTO

### CHỨC NĂNG

Trong phân cụm ở tầng này nhiệm vụ là đọc dataset, ghi ra file mới là ketqua.csv với đọc thì chuyển nó thành đối tượng dữ liệu và đọc, ghi được thực thi trong file DataAdapter.pyx.

### BẢNG HÀM VÀ LỚP

Bảng 1: Hàm

| STT | Tên hàm | Mục đích của hàm | File lưu trữ | Tên sinh viên phụ trách |
| --- | --- | --- | --- | --- |
| 1 | getFeature  Input: Tên cột trong dataset  Output: Trả về list của cột đó. | Lấy dataset theo từng cột của đối tượng DauVao và cung cấp cho hàm multiprocessing. | DataAdapter.pyx | Nguyễn Tiến Đạt |
| 2 | multiprocessing  Input: Không  Output: Đối tượng dữ liệu data. | Nhiệm vụ là xử lý song song 4 cột cùng 1 lúc để tạo ra đối tượng data cho class dataset. | DataAdapter.pyx | Nguyễn Tiến Đạt |
| 3 | xuatFile  Input: KMeans\_labels\_: Kết quả phân cụm của KM.  HC\_labels\_: Kết quả phân cụm của HC. | Ghi kết quả phân cụm của KM và HC vào file mới là ketqua.csv. | DataAdapter.pyx | Nguyễn Tiến Đạt |

Bảng 2: Lớp

| STT | Tên lớp | Tên sinh viên phụ trách | Mục đích của lớp |
| --- | --- | --- | --- |
| **1** | dataset | Nguyễn Tiến Đạt | Trả về đối tượng dữ liệu dataset cho tầng DAO sử dụng. |

### MÃ NGUỒN

#### Các thư viện sử dụng:

# coding=utf-8

import pandas as pd

import datetime

from multiprocessing.dummy import Pool as ThreadPool

#### Kết nối dataset

DauVao=pd.read\_csv('../DTO/dataset/dataset\_for\_clustering.csv',encoding='utf-8')

#### Hàm getFeature: Lấy dataset theo từng cột của đối tượng DauVao và cung cấp cho hàm multiprocessing

def getFeature(name):

return (DauVao[name].values).tolist()

#### Hàm multiprocessing: Nhiệm vụ là xử lý song song 4 cột cùng 1 lúc để tạo ra đối tượng data cho class dataset

def multiprocessing():

t=['ten\_xe','luong\_hao\_xang','so\_luong\_xi\_lanh',

'the\_tich\_dong\_co','ma\_luc','ty\_le\_truc\_sau','khoi\_luong\_xe',

'gia\_toc\_xe','loai\_xy\_lanh\_dong\_co',

'loai\_truyen\_dong','so\_luong\_banh\_rang',

'so\_luong\_bo\_che\_hoa\_khi']

pool = ThreadPool(4)

data=pool.map(getFeature,t)

return data

#### Lớp dataset: Trả về đối tượng dữ liệu dataset cho tầng DAO sử dụng

class dataset:

def \_\_init\_\_(seft):

data=multiprocessing()

seft.ten\_xe=data[0]

seft.luong\_hao\_xang=data[1]

seft.so\_luong\_xi\_lanh=data[2]

seft.the\_tich\_dong\_co=data[3]

seft.ma\_luc=data[4]

seft.ty\_le\_truc\_sau=data[5]

seft.khoi\_luong\_xe=data[6]

seft.gia\_toc\_xe=data[7]

seft.loai\_xy\_lanh\_dong\_co=data[8]

seft.loai\_truyen\_dong=data[9]

seft.so\_luong\_banh\_rang=data[10]

seft.so\_luong\_bo\_che\_hoa\_khi=data[11]

#### Hàm xuatFile: Ghi kết quả phân cụm của KM và HC vào file mới là ketqua.csv

def xuatFile(KMeans\_labels\_,HC\_labels\_):

data=dataset()

df = pd.DataFrame({

'ten\_xe': data.ten\_xe,

'luong\_hao\_xang': data.luong\_hao\_xang,

'so\_luong\_xi\_lanh':data.so\_luong\_xi\_lanh,

'the\_tich\_dong\_co':data.the\_tich\_dong\_co,

'ma\_luc':data.ma\_luc,

'ty\_le\_truc\_sau':data.ty\_le\_truc\_sau,

'khoi\_luong\_xe':data.khoi\_luong\_xe,

'gia\_toc\_xe':data.gia\_toc\_xe,

'loai\_xy\_lanh\_dong\_co':data.loai\_xy\_lanh\_dong\_co,

'loai\_truyen\_dong':data.loai\_truyen\_dong,

'so\_luong\_banh\_rang':data.so\_luong\_banh\_rang,

'so\_luong\_bo\_che\_hoa\_khi':data.so\_luong\_bo\_che\_hoa\_khi,

'KMeans\_labels\_':KMeans\_labels\_,

'HC\_labels\_':HC\_labels\_

})

df.to\_csv(‘../DTO/dataset/ketqua.csv',encoding='utf-8',index=False)

## TẦNG DAO

### CHỨC NĂNG

Mục đích chính ở tầng DAO là biến đổi dữ liệu ban đầu thành các tập dữ liệu phù hợp với thuật toán sử dụng.

### BẢNG HÀM

Bảng 1: Hàm

| STT | Tên hàm | Mục đích của hàm | File lưu trữ | Tên sinh viên phụ trách |
| --- | --- | --- | --- | --- |
| 1 | traSoThuTu  Input: Tên đặc trưng  Output: Index của cột tương ứng trong dataset. | Hàm cung cấp chỉ mục vị trí của các cột trong dataset nhầm cung cấp chọn cột dataset. | dataProcessing.pyx | Nguyễn Tiến Đạt |
| 2 | chuyenData  Input: Không.  Output: Đối tượng dataset. | Mục đích cung cấp cho hàm dataset | dataProcessing.pyx | Nguyễn Tiến Đạt |
| 3 | dataset  Input: Không.  Output: Đối tượng dataset. | Cung cấp đối tượng dataset cho tầng BUS có 2 tuỳ chọn là None: trả về toàn bộ các đặc trưng và array chỉ trả về đặc trưng trong mảng array. | dataProcessing.pyx | Nguyễn Tiến Đạt |
| 4 | xuatFile | Cung cấp hàm ghi file tương tác với hàm xuatFile của tầng DTO. | dataProcessing.pyx | Nguyễn Tiến Đạt |

### MÃ NGUỒN

#### Các thư viện sử dụng và chỉ liên kết tới tầng DTO:

import sys

sys.path.append('../DTO/')

import DataAdapter

#### Hàm traSoThuTu: Hàm cung cấp chỉ mục vị trí của các cột trong dataset nhầm cung cấp chọn cột dataset

def traSoThuTu(ten):

t=['ten\_xe','luong\_hao\_xang',

'so\_luong\_xi\_lanh','the\_tich\_dong\_co',

'ma\_luc','ty\_le\_truc\_sau','khoi\_luong\_xe',

'gia\_toc\_xe','loai\_xy\_lanh\_dong\_co',

'loai\_truyen\_dong','so\_luong\_banh\_rang',

'so\_luong\_bo\_che\_hoa\_khi']

for i in range(len(t)):

if t[i]==ten:

return i;

return -1;

#### Hàm chuyenData: Mục đích cung cấp cho hàm dataset

def chuyenData():

data=DataAdapter.dataset()

ten\_xe=data.ten\_xe

luong\_hao\_xang=data.luong\_hao\_xang

so\_luong\_xi\_lanh=data.so\_luong\_xi\_lanh

the\_tich\_dong\_co=data.the\_tich\_dong\_co

ma\_luc=data.ma\_luc

ty\_le\_truc\_sau=data.ty\_le\_truc\_sau

khoi\_luong\_xe=data.khoi\_luong\_xe

gia\_toc\_xe=data.gia\_toc\_xe

loai\_xy\_lanh\_dong\_co=data.loai\_xy\_lanh\_dong\_co

loai\_truyen\_dong=data.loai\_truyen\_dong

so\_luong\_banh\_rang=data.so\_luong\_banh\_rang

so\_luong\_bo\_che\_hoa\_khi=data.so\_luong\_bo\_che\_hoa\_khi

dataset=[]

for i in range(len(ten\_xe)):

dataset.append([ten\_xe[i],luong\_hao\_xang[i],

so\_luong\_xi\_lanh[i],the\_tich\_dong\_co[i],ma\_luc[i],ty\_le\_truc\_sau[i],

khoi\_luong\_xe[i],gia\_toc\_xe[i],loai\_xy\_lanh\_dong\_co[i],

loai\_truyen\_dong[i],so\_luong\_banh\_rang[i],

so\_luong\_bo\_che\_hoa\_khi[i]])

return dataset

#### Hàm dataset: Cung cấp đối tượng dataset cho tầng BUS có 2 tuỳ chọn là None: trả về toàn bộ các đặc trưng và array chỉ trả về đặc trưng trong mảng array

def dataset(array):

dt=chuyenData()

if array is None:

return dt

else:

k=[]

data=[]

for i in array :

if traSoThuTu(i)!=-1:

k.append(traSoThuTu(i))

for i in range(len(dt)):

k1=[]

for j in k:

k1.append(dt[i][j])

data.append(k1)

return data

#### Hàm xuatFile: Cung cấp hàm ghi file tương tác với hàm xuatFile của tầng DTO

#xuatFile

def xuatFile(km,hc):

DataAdapter.xuatFile(km.labels\_,hc.labels\_)

## TẦNG BUS

### KMeans

#### Cơ sở lý thuyết:

K-means Clustering là một thuật toán dùng trong các bài toán phân loại/nhóm n đối tượng thành k nhóm dựa trên đặc tính/thuộc tính của đối tượng.

#### Bảng hàm và lớp:

Bảng 1: Hàm

| STT | Tên hàm | Mục đích của hàm | File lưu trữ | Tên sinh viên phụ trách |
| --- | --- | --- | --- | --- |
| 1 | lay\_data  Input: Không.  Output: đối tượng dataset. | Lấy tập dataset từ tầng DAO lên và trả về 1 đối tượng dataset. | KMeans.pyx | Nguyễn Tiến Đạt |
| 2 | KMean  Input:   * mangCacDacTrung:mảng các đặc trưng. * n\_clusters: số cụm   Output: trả về kết quả của KMeans | Là hàm xử lý thuật toán KMeans. | KMeans.pyx | Nguyễn Tiến Đạt |
| 3 | traSoThuTu  Input: tên đặc trưng  Output: index của đặc trưng đó trong tập dữ liệu. | Mục đích là trả về chỉ mục vị trí trong dataset. | KMeans.pyx | Nguyễn Tiến Đạt |
| 4 | veDacTrung2D  Input: mangCacDacTrung:   * Là mảng các đặc trưng training cho KM. * mangCacDacTrungVe: Là mảng các đặc trưng cần vẽ. * soLuongDiemVe: Là số lượng điểm vẽ. * n\_clusters: là số cụm.   Output: Không. | Cung cấp hàm vẽ 2 chiều cho KM với mangCacDacTrung là None thì ta lấy mangCacDacTrungVe training, soLuongDiemVe là None thì ta vẽ tất cả điểm. | KMeans.pyx | Nguyễn Tiến Đạt |
| 5 | veDacTrung3D  Input: mangCacDacTrung:   * Là mảng các đặc trưng training cho KM. * mangCacDacTrungVe: Là mảng các đặc trưng cần vẽ. * soLuongDiemVe: Là số lượng điểm vẽ. * n\_clusters: là số cụm.   Output: Không. | Cung cấp hàm vẽ 3 chiều cho KM với mangCacDacTrung là None thì ta lấy mangCacDacTrungVe training, soLuongDiemVe là None thì ta vẽ tất cả điểm. | KMeans.pyx | Nguyễn Tiến Đạt |
| 6 | veTimSoCluster  Input:   * mangCacDacTrung: Là mảng các đặc trưng training cho KM. * diemBatDau: là điểm bắt đầu của số cụm. * diemKetThuc: là điểm kết thúc của số cụm. | Cung cấp hàm vẽ để tìm ra số cụm thích hợp. | KMeans.pyx | Nguyễn Tiến Đạt |
| 7 | help  Input: Không.  Output: Không. | Hàm cung cấp cú pháp để dễ sử dụng. | KMeans.pyx | Nguyễn Tiến Đạt |

#### Mã nguồn:

##### Các thư viện và trỏ liên kết tới DAO

import sys

sys.path.append('../DAO/')

import dataProcessing as dp

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt

import numpy as np

from mpl\_toolkits.mplot3d import Axes3D

##### Hàm lay\_data: Lấy tập dataset từ tầng DAO lên và trả về 1 đối tượng dataset

def lay\_data():

data=dp.dataset(['luong\_hao\_xang',

'so\_luong\_xi\_lanh','the\_tich\_dong\_co',

'ma\_luc','ty\_le\_truc\_sau','khoi\_luong\_xe',

'gia\_toc\_xe','loai\_xy\_lanh\_dong\_co',

'loai\_truyen\_dong','so\_luong\_banh\_rang',

'so\_luong\_bo\_che\_hoa\_khi'])

return data

##### Hàm KMean: Là hàm xử lý thuật toán KMeans

def KMean(mangCacDacTrung,n\_clusters):

data=[]

if mangCacDacTrung is None:

data=dp.dataset(['luong\_hao\_xang',

'so\_luong\_xi\_lanh','the\_tich\_dong\_co',

'ma\_luc','ty\_le\_truc\_sau','khoi\_luong\_xe',

'gia\_toc\_xe','loai\_xy\_lanh\_dong\_co',

'loai\_truyen\_dong','so\_luong\_banh\_rang',

'so\_luong\_bo\_che\_hoa\_khi'])

else:

data=dp.dataset(mangCacDacTrung)

kmeans = KMeans(init='k-means++',n\_clusters=n\_clusters, random\_state=0).fit(data)

return kmeans

##### Hàm veTimSoCluster: Cung cấp hàm vẽ để tìm ra số cụm thích hợp

def veTimSoCluster(mangCacDacTrung,diemBatDau,diemKetThuc):

clusters=[]

for i in range(diemBatDau,diemKetThuc):

kmeans = KMean(mangCacDacTrung,i)

clusters.append([i,kmeans.inertia\_])

clusters=np.array(clusters)

plt.plot(clusters[:,0],clusters[:,1],'-o',c='g',marker="+")

plt.xlabel("K")

plt.ylabel("Inertia")

plt.title("Biểu đồ phân tích Kmeans ")

plt.show()

##### Hàm traSoThuTu: Mục đích là trả về chỉ mục vị trí trong dataset

def traSoThuTu(ten):

t=['luong\_hao\_xang',

'so\_luong\_xi\_lanh','the\_tich\_dong\_co',

'ma\_luc','ty\_le\_truc\_sau','khoi\_luong\_xe',

'gia\_toc\_xe','loai\_xy\_lanh\_dong\_co',

'loai\_truyen\_dong','so\_luong\_banh\_rang',

'so\_luong\_bo\_che\_hoa\_khi']

for i in range(len(t)):

if t[i]==ten:

return i;

return -1;

##### Hàm veDacTrung2D: Cung cấp hàm vẽ 2 chiều cho KM với mangCacDacTrung là None thì ta lấy mangCacDacTrungVe training, soLuongDiemVe là None thì ta vẽ tất cả điểm

def veDacTrung2D(mangCacDacTrung,mangCacDacTrungVe,soLuongDiemVe,n\_clusters):

data=lay\_data()

#kiem tra dieu kien

if soLuongDiemVe is not None :

if soLuongDiemVe > len(data):

return None

if len(mangCacDacTrungVe)!=2:

return None

m=[]

for i in mangCacDacTrungVe:

if traSoThuTu(i)!=-1:

m.append(traSoThuTu(i))

mangVe=[]

# xac dinh dac trung ve va dac trung tinh toan

if mangCacDacTrung is None:

kmeans=KMean(mangCacDacTrungVe,n\_clusters)

else:

kmeans=KMean(mangCacDacTrung,n\_clusters)

# <-- -->

#tien hanh ve

if soLuongDiemVe is None:

for i in range(len(data)):

mangVe.append([data[i][m[0]],data[i][m[1]],kmeans.labels\_[i]])

else:

for i in range(soLuongDiemVe):

mangVe.append([data[i][m[0]],data[i][m[1]],kmeans.labels\_[i]])

centroids=kmeans.cluster\_centers\_

mangVe=np.array(mangVe)

centroids=np.array(centroids)

plt.scatter(mangVe[:,0],mangVe[:,1],c=mangVe[:,2])

plt.scatter(centroids[:,0],centroids[:,1],alpha=0.5,marker=r'$\clubsuit$',c='g',s=200,label="centroid")

plt.xlabel(mangCacDacTrungVe[0])

plt.ylabel(mangCacDacTrungVe[1])

plt.title("Biểu đồ 2d phân cụm cho Kmeans ứng với %s cụm" %n\_clusters)

plt.legend(loc='upper left')

plt.show()

##### Hàm veDacTrung3D: Cung cấp hàm vẽ 3 chiều cho KM với mangCacDacTrung là None thì ta lấy mangCacDacTrungVe training, soLuongDiemVe là None thì ta vẽ tất cả điểm

def veDacTrung3D(mangCacDacTrung,mangCacDacTrungVe,soLuongDiemVe,n\_clusters):

data=lay\_data()

#kiem tra dieu kien

if soLuongDiemVe is not None :

if soLuongDiemVe > len(data):

return None

if len(mangCacDacTrungVe)!=3:

return None

m=[]

for i in mangCacDacTrungVe:

if traSoThuTu(i)!=-1:

m.append(traSoThuTu(i))

mangVe=[]

# xac dinh dac trung ve va dac trung tinh toan

if mangCacDacTrung is None:

kmeans=KMean(mangCacDacTrungVe,n\_clusters)

else:

kmeans=KMean(mangCacDacTrung,n\_clusters)

# <-- -->

#tien hanh ve

if soLuongDiemVe is None:

for i in range(len(data)):

mangVe.append([data[i][m[0]],data[i][m[1]],data[i][m[2]],kmeans.labels\_[i]])

else:

for i in range(soLuongDiemVe):

mangVe.append([data[i][m[0]],data[i][m[1]],data[i][m[2]],kmeans.labels\_[i]])

centroids=kmeans.cluster\_centers\_

mangVe=np.array(mangVe)

centroids=np.array(centroids)

fig = plt.figure()

ax = fig.add\_subplot(111, projection='3d')

ax.scatter(mangVe[:,0], mangVe[:,1], mangVe[:,2], marker='o',c=mangVe[:,3])

ax.scatter(centroids[:,0],centroids[:,1],centroids[:,2],alpha=0.5,marker=r'$\clubsuit$',c='g',s=200,label="centroid")

ax.set\_xlabel(mangCacDacTrungVe[0])

ax.set\_ylabel(mangCacDacTrungVe[1])

ax.set\_zlabel(mangCacDacTrungVe[2])

plt.title("Biểu đồ 3d phân cụm cho Kmeans ứng với %s cụm" %n\_clusters)

ax.legend(loc='lower left')

plt.show()

##### Hàm help: Hàm cung cấp cú pháp để dễ sử dụng

def help():

print("\nhelp:\t\t.veTimSoCluster(mangCacDacTrung,diemBatDau,diemKetThuc)\n")

print("\t\t.KMean(mangCacDacTrung,n\_clusters) \n")

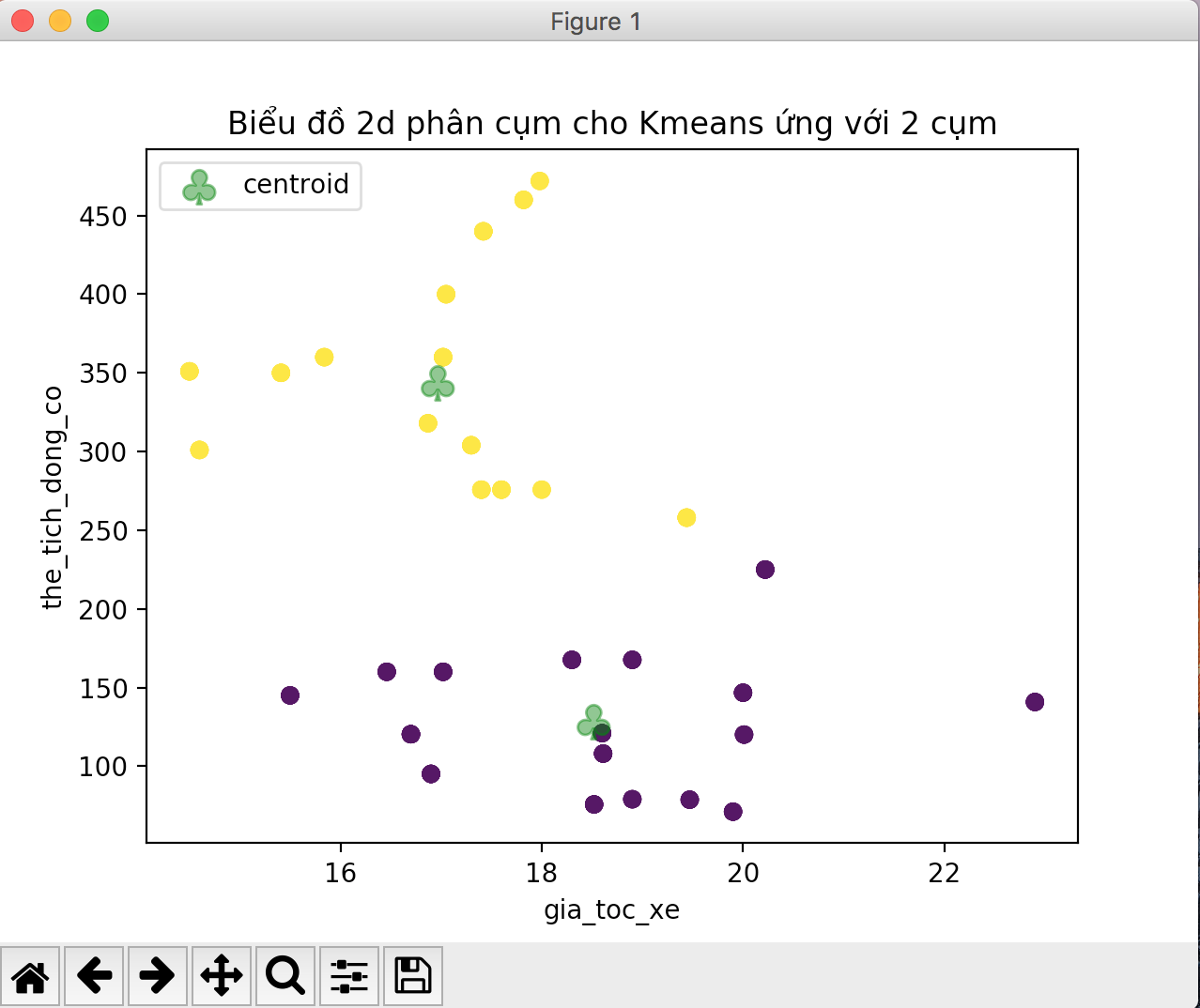
print("\t\t.veDacTrung2D(mangCacDacTrung,mangCacDacTrungVe,soLuongDiemVe,so cum)\n")

print("\t\t.veDacTrung3D(mangCacDacTrung,mangCacDacTrungVe,soLuongDiemVe,so cum)\n")

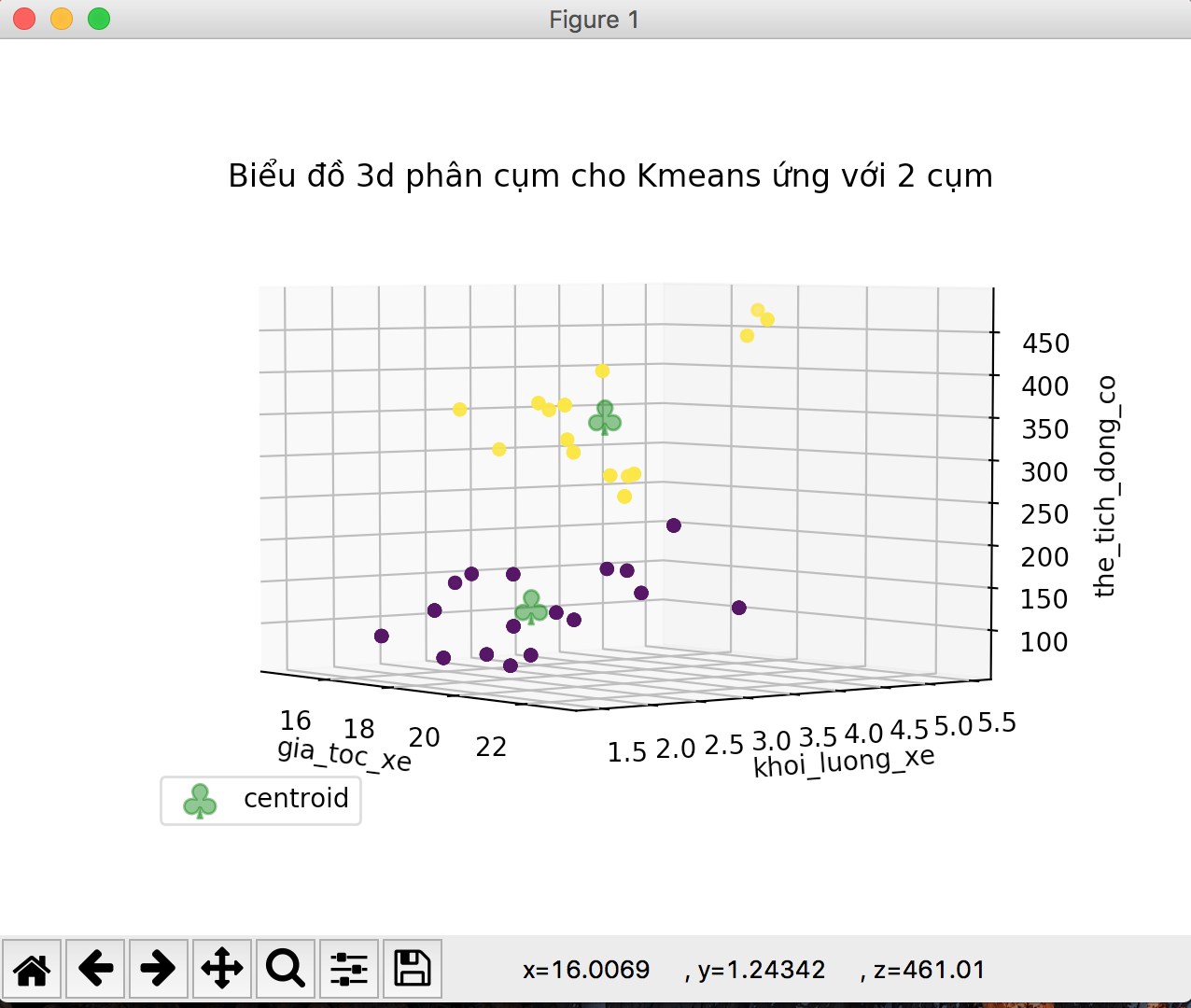
#### Đánh giá

Hình 11: Phân tích KM

Nhìn vào hình 11 em chọn ở đây 2 cụm và có ý nghĩa là



Hình 12: Vẽ 2D KM



Hình 13: Vẽ 3D KM

Ở hình 12, 13 là hình biểu thị hàm vẽ 2 chiều và 3 chiều cung cấp trong cho thuật toán KM xây dựng trong KMeans.pyx.

### Hierarchical clustering(HC)

#### Bảng hàm :

Bảng 1: Hàm

| STT | Tên hàm | Mục đích của hàm | File lưu trữ | Tên sinh viên phụ trách |
| --- | --- | --- | --- | --- |
| 1 | lay\_data  Input: Không.  Output: đối tượng dataset. | Lấy tập dataset từ tầng DAO lên và trả về 1 đối tượng dataset. | HC.pyx | Nguyễn Tiến Đạt |
| 2 | HC  Input:   * mangCacDacTrung:mảng các đặc trưng. * n\_clusters: số cụm   Output: trả về kết quả của HC | Là hàm xử lý thuật toán HC. | HC.pyx | Nguyễn Tiến Đạt |
| 3 | vitriCat  Input: n\_clusters  Output: trả ra 1 số nguyên dương. | Là hàm trả về vị trí cắt để phân cụm. | HC.pyx | Nguyễn Tiến Đạt |
| 4 | ve\_HC  Input: mangCacDacTrung:   * Là mảng các đặc trưng training cho HC. * n\_clusters: là số cụm.   Output: Không. | Vẽ biểu đồ cột phân cụm để phân chia cụm. | HC.pyx | Nguyễn Tiến Đạt |
| 5 | help  Input: Không.  Output: Không. | Hàm cung cấp cú pháp để dễ sử dụng. | HC.pyx | Nguyễn Tiến Đạt |

#### Mã nguồn:

##### Các thư viện và trỏ liên kết tới DAO

import sys

sys.path.append('../DAO/')

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import AgglomerativeClustering

import dataProcessing as dp

from scipy.cluster.hierarchy import dendrogram, linkage

##### Hàm lay\_data: Lấy tập dataset từ tầng DAO lên và trả về 1 đối tượng dataset

def lay\_data():

data=dp.dataset(['luong\_hao\_xang',

'so\_luong\_xi\_lanh','the\_tich\_dong\_co',

'ma\_luc','ty\_le\_truc\_sau','khoi\_luong\_xe',

'gia\_toc\_xe','loai\_xy\_lanh\_dong\_co',

'loai\_truyen\_dong','so\_luong\_banh\_rang',

'so\_luong\_bo\_che\_hoa\_khi'])

return data

##### Hàm HC: Là hàm xử lý thuật toán HC

def HC(mangCacDacTrung,n\_clusters):

data=[]

if mangCacDacTrung is None:

data=dp.dataset(['luong\_hao\_xang',

'so\_luong\_xi\_lanh','the\_tich\_dong\_co',

'ma\_luc','ty\_le\_truc\_sau','khoi\_luong\_xe',

'gia\_toc\_xe','loai\_xy\_lanh\_dong\_co',

'loai\_truyen\_dong','so\_luong\_banh\_rang',

'so\_luong\_bo\_che\_hoa\_khi'])

else:

data=dp.dataset(mangCacDacTrung)

hc = AgglomerativeClustering(n\_clusters=n\_clusters, linkage='ward').fit(data)

return hc

##### Hàm vitriCat: Là hàm trả về vị trí cắt để phân cụm

def vitriCat(n\_clusters):

switcher = {

2: 100,

3: 62,

4:50,

5:40,

6:30,

7:27,

8:24

}

return switcher.get(n\_clusters, 0)

##### Hàm ve\_HC: Vẽ biểu đồ cột phân cụm để phân chia cụm

def ve\_HC(mangCacDacTrung,n\_clusters):

data = HC(mangCacDacTrung,n\_clusters).children\_

Z = linkage(data)

dendrogram(Z=Z)

if mangCacDacTrung is None:

y=vitriCat(n\_clusters)

plt.axhline(y=y,c="black")

plt.show()

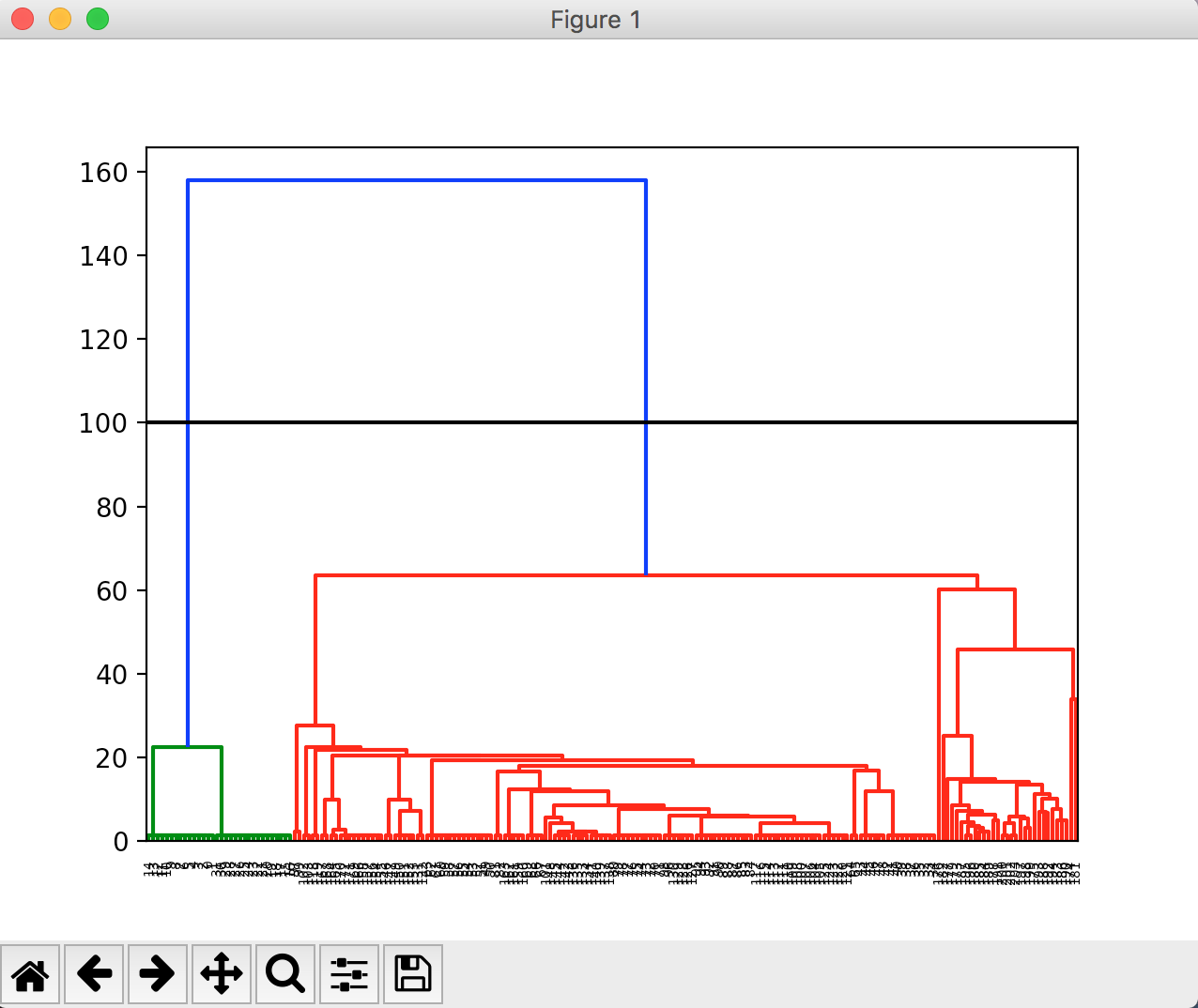
##### Hàm help: Hàm cung cấp cú pháp để dễ sử dụng

def help():

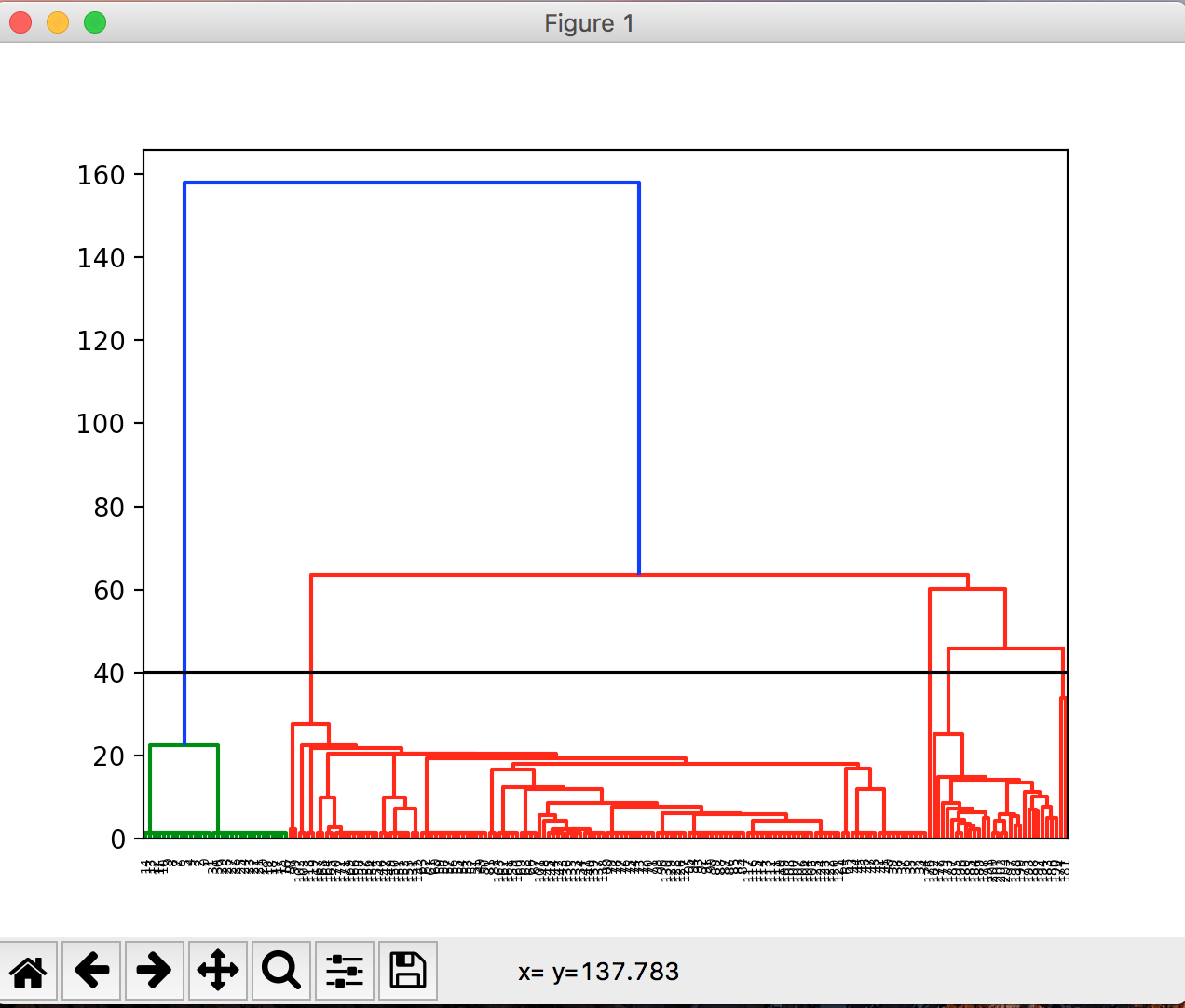
print("\nhelp:\t\t.HC(mangCacDacTrung,n\_clusters) tra ve HC\n")

print("\t\t.ve\_HC(mangCacDacTrung,n\_clusters)\n")

#### Đánh giá



Hình 14: Biểu đồ HC1



Hình 15: Biểu đồ HC1

Nhìn vào hình 14, 15 là biểu đồ từ hàm vẽ của HC trong file Hc.pyx. Em chọn ở đây 2 cụm và có ý nghĩa là

## TẦNG APP

### CHỨC NĂNG

Là tầng cấu hình, cài đặt, biên dịch và gọi nghiệp vụ xử lý của tầng BUS hiển thị kết quả ra ngoài như giao diện trực quan hay command line interface.

### BẢNG HÀM

Bảng 1: Hàm

| STT | Tên hàm | Mục đích của hàm | File lưu trữ | Tên sinh viên phụ trách |
| --- | --- | --- | --- | --- |
| 1 | km  Input: Không.  Output: đối tượng km. | Là hàm sử dụng các nghiệp vụ BUS cho KM để vẽ 2 chiều, 3 chiều và kết quả của thuật toán phân cụm. | main.pyx | Nguyễn Tiến Đạt |
| 2 | hc  Input: Không.  Output: đối tượng hc. | Là hàm sử dụng các nghiệp vụ BUS cho HC để vẽ HC và kết quả của thuật toán phân cụm. | main.pyx | Nguyễn Tiến Đạt |
| 3 | xuatFile  Input:   * km : đối tượng KM. * hc: đối tượng HC. | Là hàm ghi file truyền xuống kết quả phân cụm của HC và KM. | main.pyx | Nguyễn Tiến Đạt |

### MÃ NGUỒN

#### File setup.py

##### Chức năng

Mục đích để biên dịch tất cả file .pyx ra thành các module để cho file index.py có thể sử dụng.

##### Khai báo thư viện Cython sử dụng:

from distutils.core import setup

from Cython.Build import cythonize

##### Trỏ đường dẫn đến các file .pyx của các tầng để biên dịch ra các module(file .o) cung cấp cho file index.py sử dụng:

setup(ext\_modules=cythonize('main.pyx'))

setup(ext\_modules=cythonize('../BUS/Hc.pyx'))

setup(ext\_modules=cythonize('../BUS/KMeans.pyx'))

setup(ext\_modules=cythonize('../DAO/dataProcessing.pyx'))

setup(ext\_modules=cythonize('../DTO/DataAdapter.pyx'))

#### File main.pyx

##### Chức năng

Mục đích để sử dụng các nghiệp vụ từ tầng BUS và triển khai thành các hàm cung cấp cho file index.py thực thi.

#### File index.py

##### Chức năng

Là file tương tác trực tiếp với người dùng cuối thông qua gọi các hàm từ file main.pyx. Các file main.pyx và index.py không phải là các file thừa thải vì để biên dịch theo cấu trúc Cython chúng ta cần 1 file main.py để kết nối và file index.py để sử dụng các module đã được tạo ra.

##### Liên kết tới main.o khi mới tạo ra

import main as m

##### Sử dụng các hàm trong main.o

km=m.km()

hc=m.hc()

m.xuatFile(km,hc)

# PHẦN 4: PHÂN CÔNG

| MÃ SỐ SINH VIÊN | HỌ VÀ TÊN | CÔNG VIỆC TRONG PHÂN LỚP | CÔNG VIỆC TRONG PHÂN CỤM |
| --- | --- | --- | --- |
| 16110048 | Nguyễn Tiến Đạt | * Tiền xử lý dữ liệu(DAO, DTO, APP). * Xử lý thuật toán KNN. | * Tiền xử lý dữ liệu(DAO, DTO, APP). * Xử lý thuật toán KMeans. * Xử lý thuật toán HC. |
|  |  |  | Không |
|  |  |  | Không |