

MỤC LỤC

MỤC LỤC	5
DANH MỤC HÌNH ẢNH	11
PHẦN 1: CẤU TRÚC PHÂN TẦNG	12
I. TỔNG QUAN	12
II. CÀI ĐẶT	15
PHẦN 2: CẤU TRÚC CHO PHÂN LỚP	16
I. TẦNG DTO	16
1. CHỨC NĂNG	16
2. BẢNG HÀM VÀ LỚP	16
3. MÃ NGUỒN	17
a. Kết nối dataset	17
b. 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
c. Hàm multiprocessing: Nhiệm vụ là xử lý song song 4 cột cùng 1 lúc để tạo ra đối tượng data cho class dataset	17
d. Lớp dataset: Nhiệm vụ lấy các data của hàm multiprocessing() để tạo ra 1 đối tượng dataset cho tầng DAO sử dụng	18
II. TẦNG DAO	19
1. CHỨC NĂNG	19
2. BẢNG HÀM VÀ LỚP	19
3. MÃ NGUỒN	22
a. Các thư viện sử dụng và thiết đặc liên kết với tầng DTO	22
b. Lấy đối tượng dataset từ tầng DTO	22
c. Hàm xoaTrung: Xóa phần tử trùng trong 1 mảng lấy từ 1 cột trong dataset, cung cấp cho hàm ganSoPhanLoai	23
d. Hàm ganSoPhanLoai: Lấy phần tử đã xóa trùng từ hàm xoaTrung để đánh số tương ứng với mỗi phần tử	23
e. Hàm chuyen_dac_truong_sang_so: Gán số từ hàm ganSoPhanLoai ánh xạ vào cột dataset tương 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

- f. Hàm `traSoThuTu`: Hàm cung cấp chỉ mục vị trí của các cột trong dataset nhằm cung cấp chọn cột dataset 24
- g. 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
- h. Chuyển các đặc trưng sang số có cho phép null 27
- i. Bắt đầu lọc dữ liệu xóa null 29
- j. 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

III. TẦNG BUS 31

- 1. KNN 31
 - a. Cơ sở lý thuyết: 31
 - b. Bảng hàm và lớp: 31
 - c. Mã nguồn 35
 - ❖ Các thư viện và trở 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 giá 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

d. Đánh giá 49

IV. TẦNG APP 53

1. CHỨC NĂNG 53

2. BẢNG HÀM 53

3. MÃ NGUỒN 53

a. 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
b. File main.pyx	54
❖ Chức năng	54
c. 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

I. TẦNG DTO 55

1. CHỨC NĂNG 55

2. BẢNG HÀM VÀ LỚP 55

3. MÃ NGUỒN 56

a. Các thư viện sử dụng:	56
b. Kết nối dataset	56
c. Hàm getFeature: Lấy dataset theo từng cột của đối tượng DauVao và cung cấp cho hàm multiprocessing	56
d. Hàm multiprocessing: Nhiệm vụ là xử lý song song 4 cột cùng 1 lúc để tạo ra đối tượng data cho class dataset	57
e. Lớp dataset: Trả về đối tượng dữ liệu dataset cho tầng DAO sử dụng	57
f. Hàm xuấtFile: Ghi kết quả phân cụm của KM và HC vào file mới là ketqua.csv	58

II. TẦNG DAO 58

1. CHỨC NĂNG 58

2. BẢNG HÀM 58

3. MÃ NGUỒN 59

a. Các thư viện sử dụng và chỉ liên kết tới tầng DTO:	59
b. Hàm traSoThuTu: Hàm cung cấp chỉ mục vị trí của các cột trong dataset nhằm cung cấp chọn cột dataset	60
c. Hàm chuyenData: Mục đích cung cấp cho hàm dataset	60
d. Hàm dataset: Cung cấp đối tượng dataset cho tầng BUS có 2 tùy chọn là None: trả về toàn bộ các đặc trưng và array chỉ trả về đặc trưng trong mảng array	61

e. Hàm xuấtFile: Cung cấp hàm ghi file tương tác với hàm xuấtFile của tầng DTO	61
III. TẦNG BUS	62
1. KMeans	62
a. Cơ sở lý thuyết:	62
b. Bảng hàm và lớp:	62
c. Mã nguồn:	64
❖ Các thư viện và trở 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
d. Đánh giá	70
2. Hierarchical clustering(HC)	72
a. Bảng hàm :	72
b. Mã nguồn:	73
❖ Các thư viện và trở 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
c. Đánh giá	75
IV. TẦNG APP	77
1. CHỨC NĂNG	77
2. BẢNG HÀM	77

3. MÃ NGUỒN	78
a. 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
b. File main.pyx	79
❖ Chức năng	79
c. 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

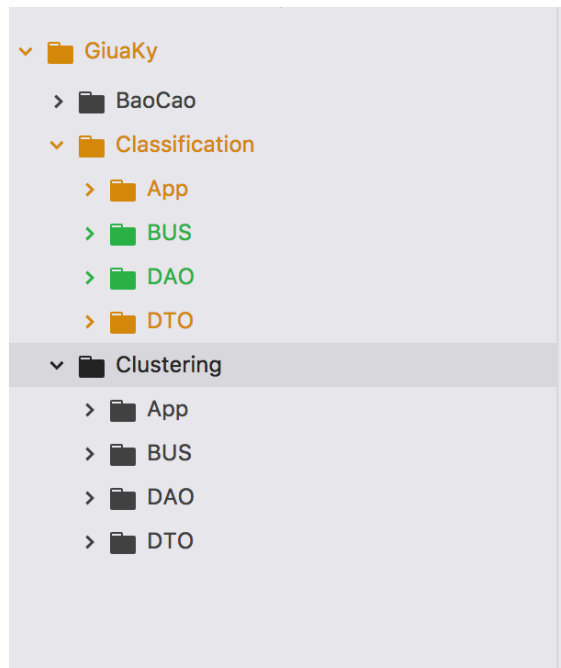
DANH MỤC HÌNH ẢNH

PHẦN 1: CẤU TRÚC PHÂN TẦNG

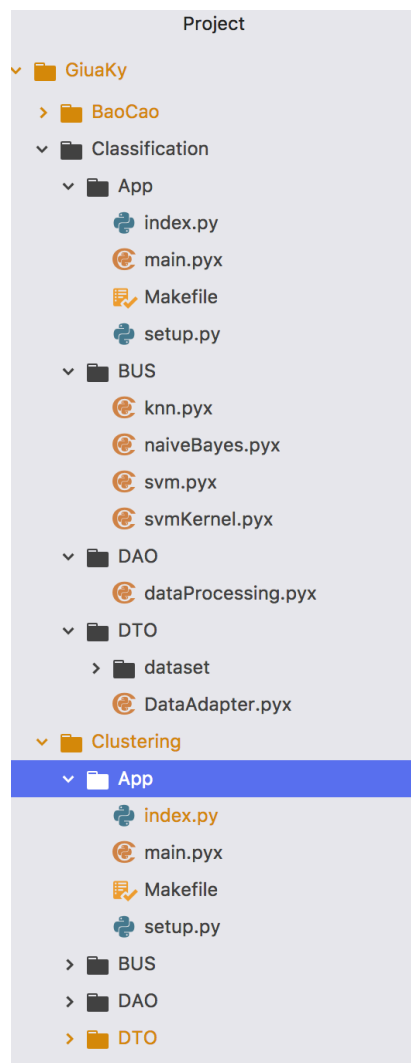
I. TỔNG QUAN

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 \longleftrightarrow BUS \longleftrightarrow DAO \longleftrightarrow 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.



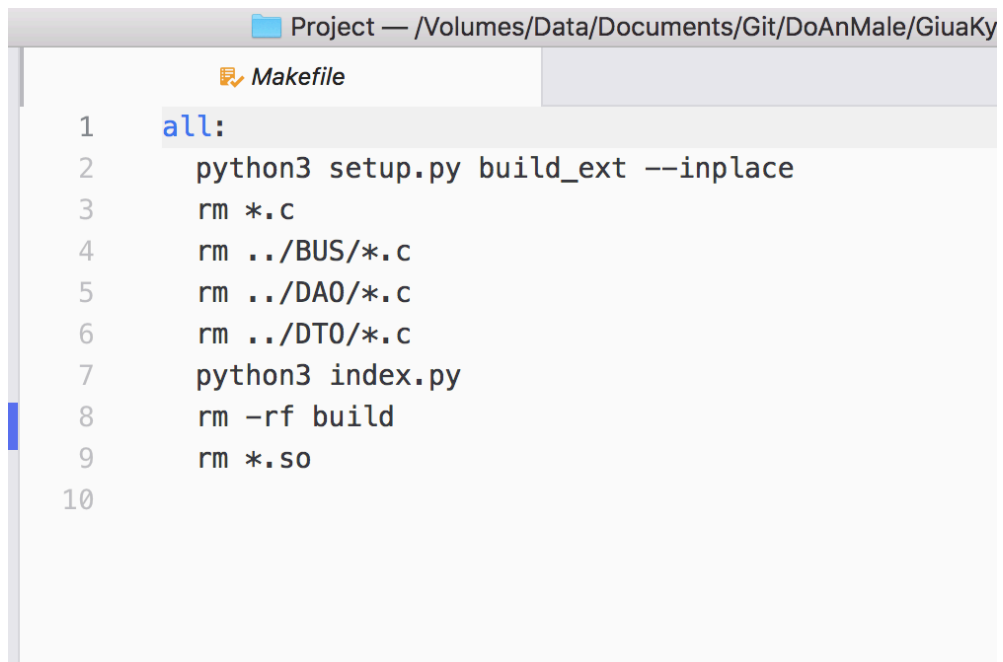
Hình 1: Cấu trúc 1.



Hình 2: Cấu trúc 2.

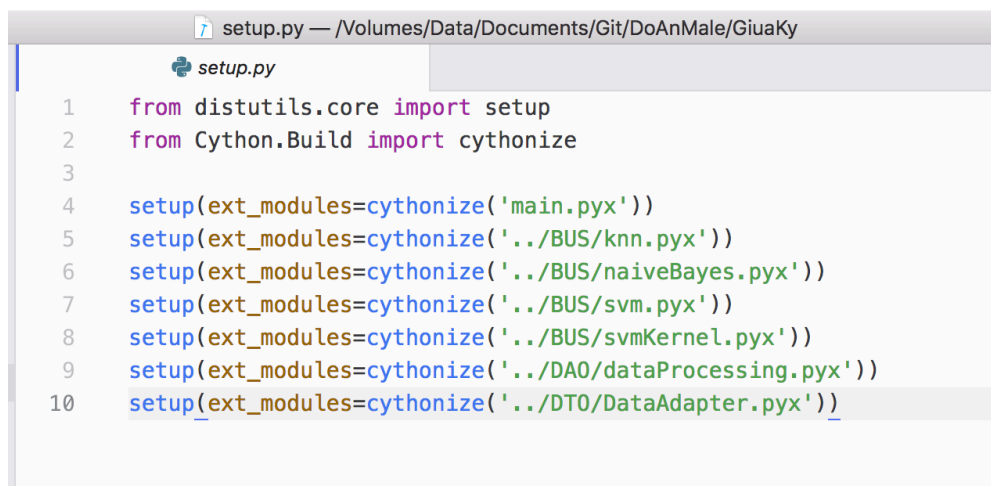
Toàn bộ chương trình để tăng tốc độ thực thi nên chúng em có sử dụng Cython:

- 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 thỏa đầu vào biên dịch. Để biên dịch toàn bộ chương trình sử dụng Cython vì có nhiều tầng nên chúng em sử dụng file setup.py nằm tại thư mục App để biên dịch cho tất cả các tầng và được thực thi thông qua Makefile:



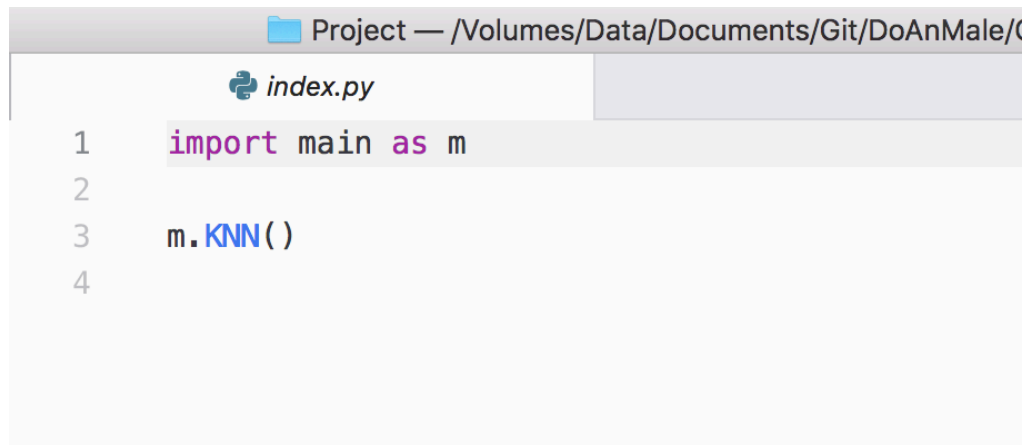
```
Project — /Volumes/Data/Documents/Git/DoAnMale/GiuaKy
Makefile
1 all:
2   python3 setup.py build_ext --inplace
3   rm *.c
4   rm ../BUS/*.c
5   rm ../DAO/*.c
6   rm ../DT0/*.c
7   python3 index.py
8   rm -rf build
9   rm *.so
10
```

Hình 3: Cấu trúc Makefile.



```
setup.py — /Volumes/Data/Documents/Git/DoAnMale/GiuaKy
setup.py
1 from distutils.core import setup
2 from Cython.Build import cythonize
3
4 setup(ext_modules=cythonize('main.pyx'))
5 setup(ext_modules=cythonize('../BUS/knn.pyx'))
6 setup(ext_modules=cythonize('../BUS/naiveBayes.pyx'))
7 setup(ext_modules=cythonize('../BUS/svm.pyx'))
8 setup(ext_modules=cythonize('../BUS/svmKernel.pyx'))
9 setup(ext_modules=cythonize('../DAO/dataProcessing.pyx'))
10 setup(ext_modules=cythonize('../DT0/DataAdapter.pyx'))
```

Hình 4: setup.py.



```
Project — /Volumes/Data/Documents/Git/DoAnMale/
index.py
1  import main as m
2
3  m.KNN()
4
```

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.

II. CÀI ĐẶT

Yêu cầu chương trình hoặc thư viện 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

I. TẦNG DTO

1. 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.

2. BẢNG HÀM VÀ LỚP

Bảng 1: Hàm

STT	Tên hàm	Mục đích của hàm	File lưu trữ	Tên sinh viên phụ trách
1	<p>getFeature</p> <p>Input: Tên cột trong dataset</p> <p>Output: Trả về list của cột đó.</p>	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	<p>multiprocessing</p> <p>Input: Không</p> <p>Output: Đối tượng dữ liệu data.</p>	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.

3. MÃ NGUỒN

a. Kết nối dataset

```
DauVao=pd.read_csv('../DTO/dataset/
dataset_for_classification.csv',encoding='utf-8')
```

b. 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
```

c. 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

```

d. 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__(self):
    data=multiprocessing()
    self.tuoi=data[0]
    self.nghe_nghiep=data[1]
    self.hon_nhan=data[2]
    self.hoc_van=data[3]
    self.co_the_tin_dung=data[4]
    self.co_nha_o=data[5]
    self.vay_ca_nhan=data[6]
    self.kenh_lien_lac=data[7]
    self.thang_lien_lac=data[8]
    self.ngay_lien_lac=data[9]
    self.thoi_luong_lien_lac= data[10]
    self.so_luong_lien_lac=data[11]
    self.ngay=data[12]
    self.so_luong_lien_lac_truoc_day=data[13]
    self.ket_qua_lan_truoc=data[14]
    self.ti_le_thay_doi_viec_lam=data[15]
    self.CPI=data[16]
    self.CCI=data[17]
    self.lai_suat_3thang=data[18]
    self.so_luong_nhan_vien=data[19]

```

seft.label=data[20]

II. TẦNG DAO

1. 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.

2. BẢNG HÀM VÀ LỚP

Bảng 1: Hàm

STT	Tên hàm	Mục đích của hàm	File lưu trữ	Tên sinh viên phụ trách
1	<p>xoaTrung</p> <p>Input: 1 mảng cần xóa trùng.</p> <p>Output: Mảng đã được xóa phần tử trùng.</p>	<p>Xoá phần tử trùng trong 1 mảng lấy từ 1 cột trong dataset, cung cấp cho hàm ganSoPhanLoai</p>	<p>dataProcessi ng.pyx</p>	<p>Nguyễn Tiến Đạt</p>
2	<p>ganSoPhanLoai</p> <p>Input: xoaTrung.</p> <p>Output: mảng đã được đánh số.</p>	<p>Lấy phần tử đã xóa trùng từ hàm xoaTrung để đánh số tương ứng với mỗi phần tử.</p>	<p>dataProcessi ng.pyx</p>	<p>Nguyễn Tiến Đạt</p>

STT	Tên hàm	Mục đích của hàm	File lưu trữ	Tên sinh viên phụ trách
3	<p>chuyen_dac_truong_sang_so</p> <p>Input: cột đặc trưng cũ, ganSoPhanLoai.</p> <p>Output: có 2 trường hợp 1 mảng được gán số cho dữ liệu unknown và 1 mảng không có dữ liệu unknown.</p>	<p>Gán số từ hàm ganSoPhanLoai ánh xạ vào cột dataset tương 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.</p>	dataProcessing.pyx	Nguyễn Tiến Đạt
4	<p>traSoThuTu</p> <p>Input: tên cột trong dataset.</p> <p>Output: index của cột đó.</p>	<p>Hàm cung cấp chỉ mục vị trí của các cột trong dataset nhằm cung cấp chọn cột dataset.</p>	dataProcessing.pyx	Nguyễn Tiến Đạt

STT	Tên hàm	Mục đích của hàm	File lưu trữ	Tên sinh viên phụ trách
5	<p>data_khongLoc</p> <p>Input: k, array</p> <p>Output: tập dataset mới hoặc label của nó.</p>	<p>Trả về tập dataset mà không lọc unknown, đầu vào là giá trị k và mảng danh sách các đặc trưng array.</p> <p>Nếu k=0 trả về dataset và k=1 trả về label của nó.</p> <p>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</p>	dataProcessing.pyx	Nguyễn Tiến Đạt

STT	Tên hàm	Mục đích của hàm	File lưu trữ	Tên sinh viên phụ trách
6	<p>data_CoLoc</p> <p>Input: k, array</p> <p>Output: tập dataset mới hoặc label của nó.</p>	<p>Trả về tập dataset mà có lọc unknown, đầu vào là giá trị k và mảng danh sách các đặc trưng array.</p> <p>Nếu k=0 trả về dataset và k=1 trả về label của nó.</p> <p>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</p>	dataProcessing.pyx	Nguyễn Tiến Đạt

3. MÃ NGUỒN

a. Các thư viện sử dụng và thiết lập 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
```

b. Lấy đối tượng dataset từ tầng DTO

```
dataset=da.dataset()
```

c. Hàm xoaTrung: Xóa 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
```

d. Hàm ganSoPhanLoai: Lấy phần tử đã xóa 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
```

e. Hàm chuyen_dac_truong_sang_so: Gán số từ hàm ganSoPhanLoai ánh xạ vào cột dataset tương 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;

```

f. Hàm traSoThuTu: Hàm cung cấp chỉ mục vị trí của các cột trong dataset nhằm 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;

```

g. 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;

```

h. 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_oi1[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]
])

```

i. Bắt đầu lọc dữ liệu xóa 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

```

j. 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;
```

III. TẦNG BUS

1. KNN

a. 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*).

b. 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 tìmNhungDacTrungTotNhat.	knn.pyx	Nguyễn Tiến Đạt

STT	Tên hàm	Mục đích của hàm	File lưu trữ	Tên sinh viên phụ trách
3	<p>cross_Validation</p> <p>Input: n_neighbors,X,Y n_neighbors: số hàng xóm. X: là tập dữ liệu. Y: là tập label của X.</p>	Cung cấp đánh giá 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	<p>n_neighbors</p> <p>Input: k Output: list các phần tử ứng với k</p>	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	<p>timF_CoLoc</p> <p>Input: n, good_KNN_CoLoc Output: good_KNN_CoLoc</p>	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

STT	Tên hàm	Mục đích của hàm	File lưu trữ	Tên sinh viên phụ trách
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_Colloc Input: Không. Output: good_KNN_Colloc	Mục đích là gọi hàm timF_Colloc, 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

STT	Tên hàm	Mục đích của hàm	File lưu trữ	Tên sinh viên phụ trách
10	<p>traSoThuTu</p> <p>Input: tên đặc trưng Output: index của đặc trưng đó trong tập dữ liệu.</p>	Mục đích là trả về chỉ mục vị trí trong dataset.	knn.pyx	Nguyễn Tiến Đạt
11	<p>Ve2D</p> <p>Input: chonBoDuLieu:</p> <ul style="list-style-type: none"> Là tập dữ liệu vẽ nếu 0 tập có lọc, 1 tập không lọc. <p>mangCacDacTrungVe:</p> <ul style="list-style-type: none"> Là mảng các đặc trưng cần vẽ với None là tất cả. <p>soLuongDiemVe:</p> <ul style="list-style-type: none"> Là số lượng điểm vẽ. 	Cung cấp hàm vẽ 2 chiều cho các đặc trưng.	knn.pyx	Nguyễn Tiến Đạt
12	<p>Ve3D</p> <p>Input: chonBoDuLieu:</p> <ul style="list-style-type: none"> Là tập dữ liệu vẽ nếu 0 tập có lọc, 1 tập không lọc. <p>mangCacDacTrungVe:</p> <ul style="list-style-type: none"> Là mảng các đặc trưng cần vẽ với None là tất cả. <p>soLuongDiemVe:</p> <ul style="list-style-type: none"> Là số lượng điểm vẽ. 	Cung cấp hàm vẽ 3 chiều cho các đặc trưng.	knn.pyx	Nguyễn Tiến Đạt

STT	Tên hàm	Mục đích của hàm	File lưu trữ	Tên sinh viên phụ trách
13	help Input: Không. Output: Không	Cung cấp cú pháp để dễ sử dụng.	knn.pyx	Nguyễn Tiến Đạt
14	timNhưngDacTrungTotNhat Input: Không. Output: đối tượng nhưngDacTrungTotNhat.	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	nhưngDacTrungTotNhat	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.

c. Mã nguồn

❖ Các thư viện và trở 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 hungDacTrungTotNhat:
    def __init__(self,array,F):
        self.array=array
        self.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 tìmNhưngDacTrungTotNhat:** 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 tìmNhưngDacTrungTotNhat():
    NDTTN=nhưngDacTrungTotNhat(0,0)
    for i in range(3,7):
        tmp=NhưngDacTrungTotNhat(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 giá 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 lưu param tốt nhất

class good_KNN:

    def __init__(self, weights, n_neighbors, F):

        self.weights = weights

        self.n_neighbors = n_neighbors

        self.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**

```
# tìm 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`

```

# tìm 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_Khong
    Loc)

    good_KNN_KhongLoc=timF_KhongLoc(n_neighbors(1),good_KNN_Khong
    Loc)

    good_KNN_KhongLoc=timF_KhongLoc(n_neighbors(2),good_KNN_Khong
    Loc)

    good_KNN_KhongLoc=timF_KhongLoc(n_neighbors(3),good_KNN_Khong
    Loc)

    good_KNN_KhongLoc=timF_KhongLoc(n_neighbors(4),good_KNN_Khong
    Loc)

    good_KNN_KhongLoc=timF_KhongLoc(n_neighbors(5),good_KNN_Khong
    Loc)

    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)

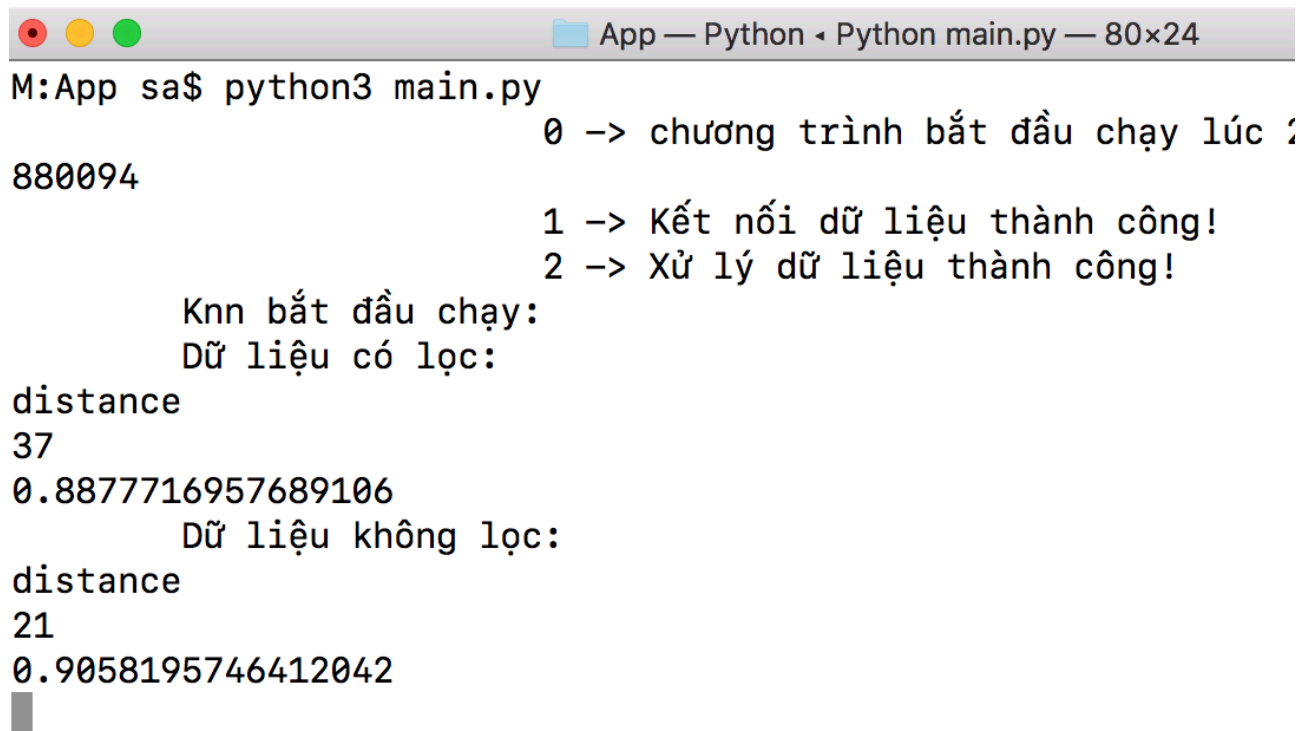
```

```
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:
```


d. Đánh giá



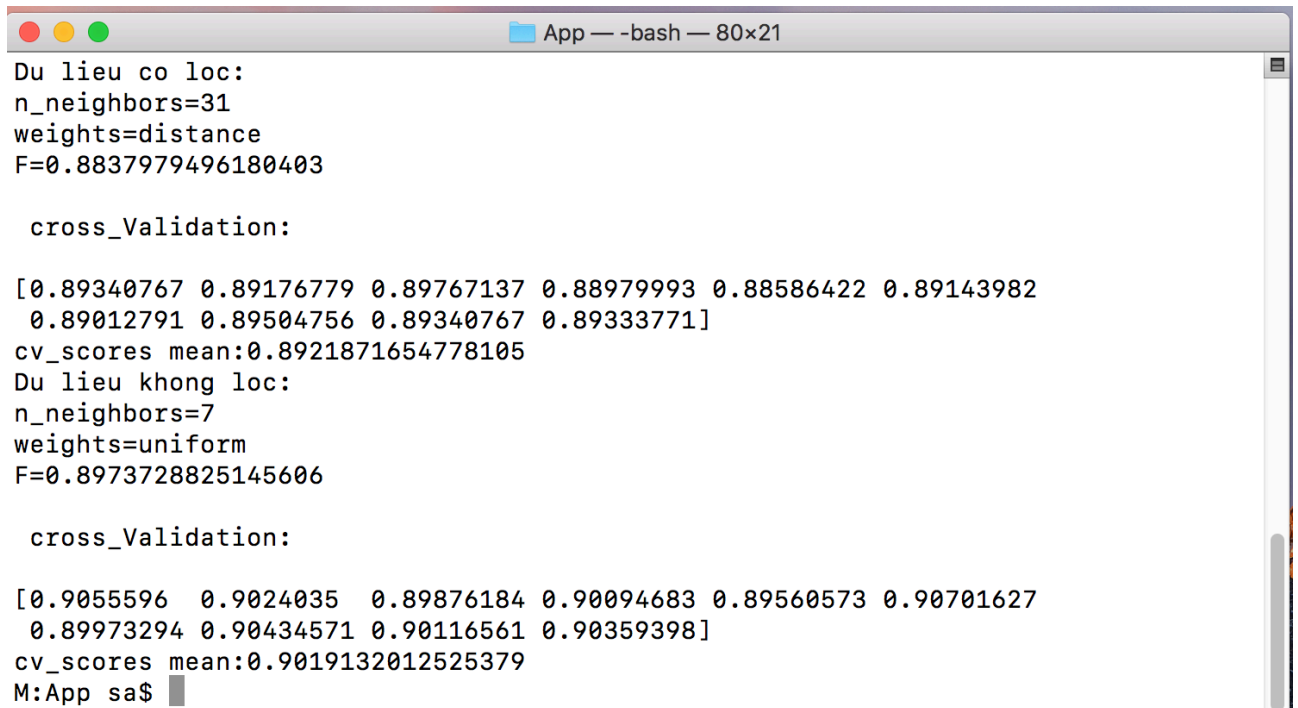
```
M:App sa$ python3 main.py
0 -> chương trình bắt đầu chạy lúc :
880094
1 -> Kết nối dữ liệu thành công!
2 -> Xử lý dữ liệu thành công!
Knn bắt đầu chạy:
Dữ liệu có lọc:
distance
37
0.8877716957689106
Dữ liệu không lọc:
distance
21
0.9058195746412042
```

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 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.



```
App — -bash — 80x21
Du lieu co loc:
n_neighbors=31
weights=distance
F=0.8837979496180403

cross_Validation:

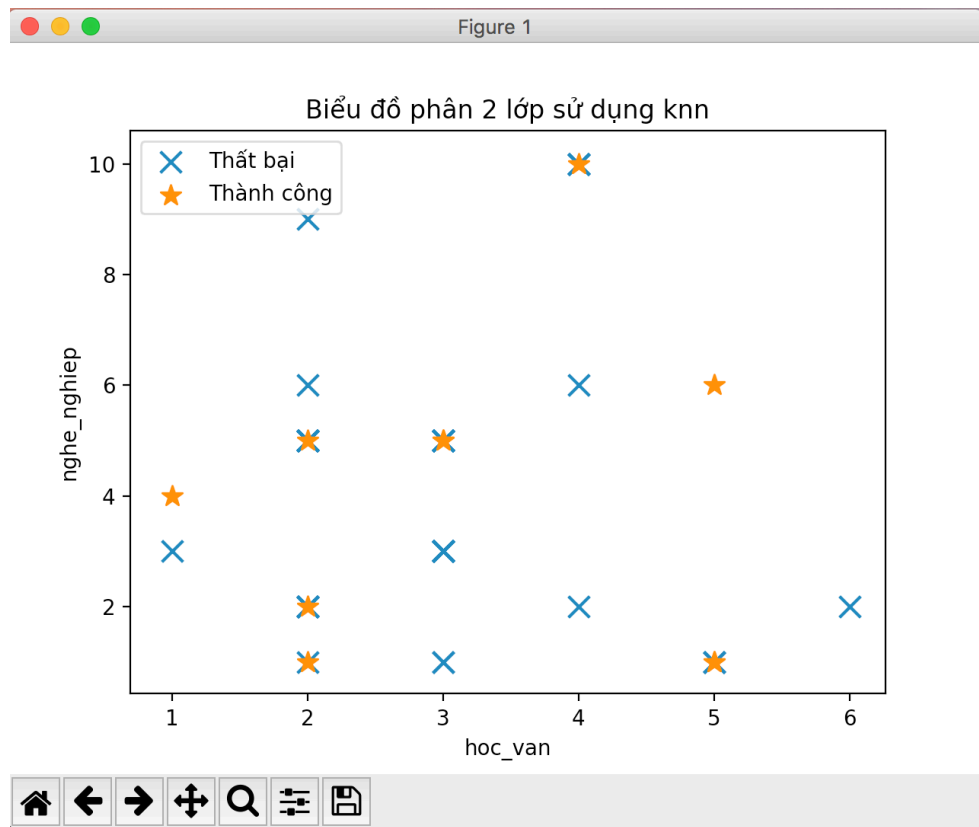
[0.89340767 0.89176779 0.89767137 0.88979993 0.88586422 0.89143982
 0.89012791 0.89504756 0.89340767 0.89333771]
cv_scores mean:0.8921871654778105
Du lieu khong loc:
n_neighbors=7
weights=uniform
F=0.8973728825145606

cross_Validation:

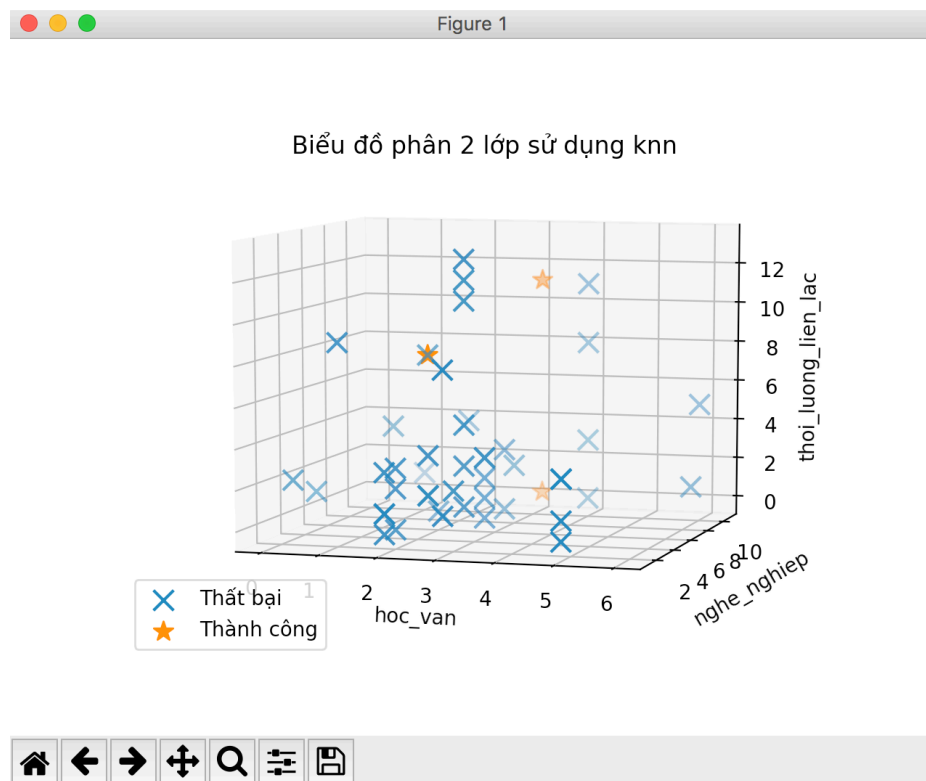
[0.9055596 0.9024035 0.89876184 0.90094683 0.89560573 0.90701627
 0.89973294 0.90434571 0.90116561 0.90359398]
cv_scores mean:0.9019132012525379
M:App sa$
```

Hình 7: Cross validation.

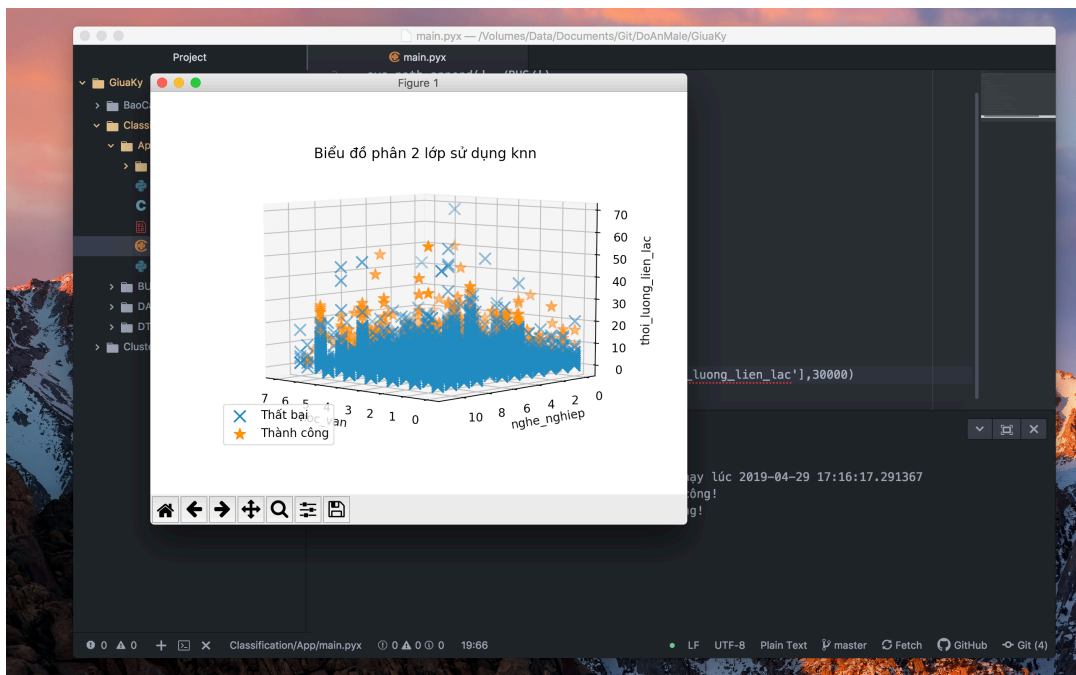
Ở hình 7 em có sử dụng cross validation với $cv=10$ cho cả 2 tập dữ liệu thì thấy mức đánh giá của tập không lọc unknown là tầm 90% và của tập có lọc unknown là 89%.



Hình 8: Vẽ 2d KNN.



Hình 9: Vẽ 3d KNN1.



Hình 10: Vẽ 3d KNN2.

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

IV. TẦNG APP

1. 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.

2. 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
			aa	ssss
			s	s
			aaa	

3. MÃ NGUỒN

a. 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'))
```

b. 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.

c. 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 thãi 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

I. TẦNG DTO

1. 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.

2. 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.py x	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.py x	Nguyễn Tiến Đạt

STT	Tên hàm	Mục đích của hàm	File lưu trữ	Tên sinh viên phụ trách
3	xuấtFile 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.py x	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.

3. MÃ NGUỒN

a. Các thư viện sử dụng:

```
# coding=utf-8
import pandas as pd
import datetime
from multiprocessing.dummy import Pool as ThreadPool
```

b. Kết nối dataset

```
DauVao=pd.read_csv('../DTO/dataset/dataset_for_clustering.csv',encoding='utf-8')
```

c. 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()
```

d. 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
```

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

```
class dataset:
```

```
    def __init__(self):  
        data=multiprocessing()  
        self.ten_xe=data[0]  
        self.luong_hao_xang=data[1]  
        self.so_luong_xi_lanh=data[2]  
        self.the_tich_dong_co=data[3]  
        self.ma_luc=data[4]  
        self.ty_le_truc_sau=data[5]  
        self.khoi_luong_xe=data[6]  
        self.gia_toc_xe=data[7]  
        self.loai_xy_lanh_dong_co=data[8]  
        self.loai_truyen_dong=data[9]  
        self.so_luong_banh_rang=data[10]  
        self.so_luong_bo_che_hoa_khi=data[11]
```

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

```
def xuấtFile(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)
```

II. TẦNG DAO

1. 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.

2. 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 nhằm 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 tùy 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 xuấtFile của tầng DTO.	dataProcessing.pyx	Nguyễn Tiến Đạt

3. MÃ NGUỒN

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

```
import sys
sys.path.append('../DTO/')
import DataAdapter
```

b. Hàm traSoThuTu: Hàm cung cấp chỉ mục vị trí của các cột trong dataset nhằm 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;
```

c. 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

```

- d. Hàm dataset: Cung cấp đối tượng dataset cho tầng BUS có 2 tùy 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

```

- e. Hàm xuấtFile: Cung cấp hàm ghi file tương tác với hàm xuấtFile của tầng DTO**

```

#xuấtFile

def xuấtFile(km,hc):
    DataAdapter.xuấtFile(km.labels_,hc.labels_)

```

III. TẦNG BUS

1. KMeans

a. 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.

b. 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: <ul style="list-style-type: none">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

STT	Tên hàm	Mục đích của hàm	File lưu trữ	Tên sinh viên phụ trách
4	<p>veDacTrung2D</p> <p>Input:</p> <p>mangCacDacTrung:</p> <ul style="list-style-type: none"> 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. <p>Output: Không.</p>	<p>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.</p>	KMeans.pyx	Nguyễn Tiến Đạt
5	<p>veDacTrung3D</p> <p>Input:</p> <p>mangCacDacTrung:</p> <ul style="list-style-type: none"> 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. <p>Output: Không.</p>	<p>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.</p>	KMeans.pyx	Nguyễn Tiến Đạt

STT	Tên hàm	Mục đích của hàm	File lưu trữ	Tên sinh viên phụ trách
6	veTimSoCluster Input: <ul style="list-style-type: none"> 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

c. Mã nguồn:

❖ Các thư viện và trở 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()

```

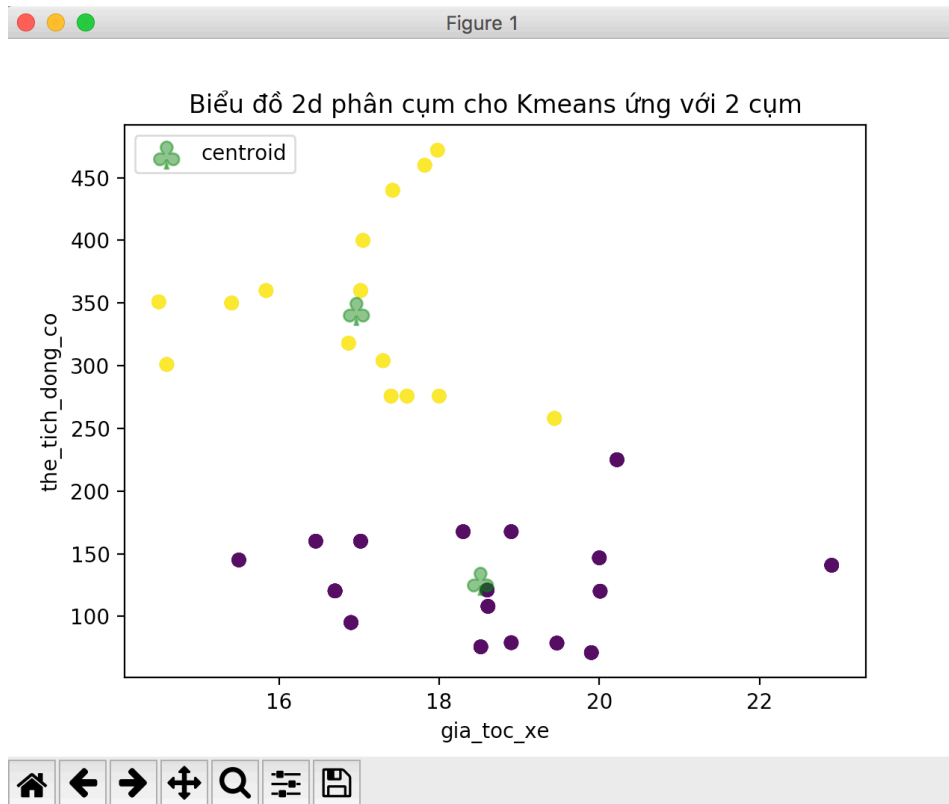
❖ **Hàm help: Hàm cung cấp cú pháp để dễ sử dụng**

```

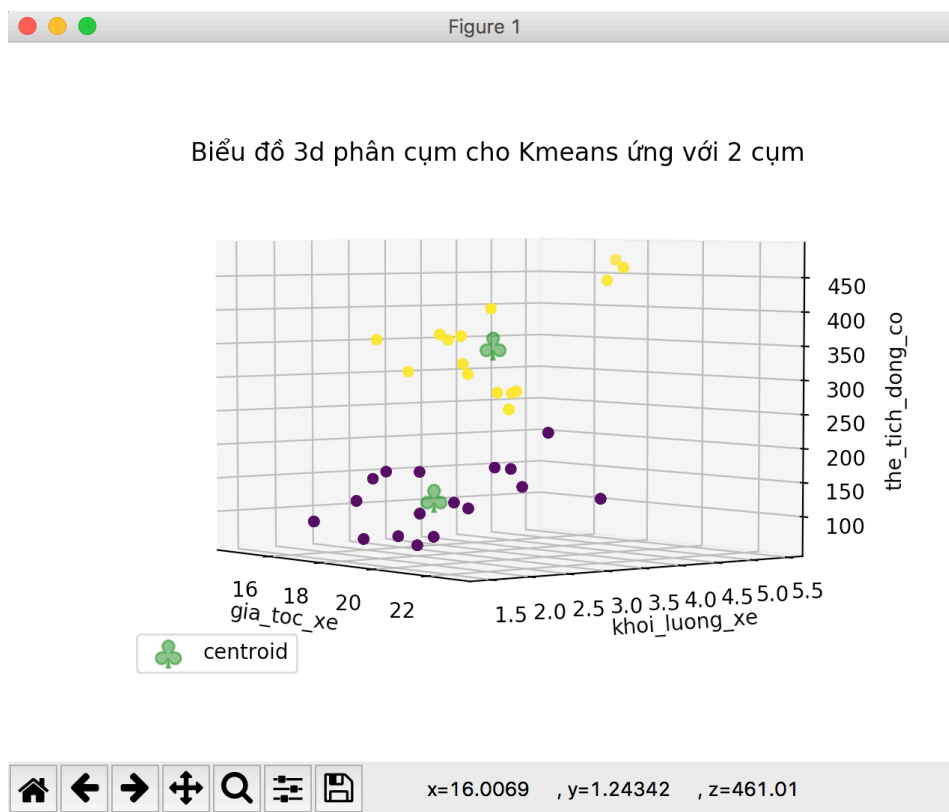
def help():
    print("\nhelp:
\t\t.veTimSoCluster(mangCacDacTrung,diemBatDau,diemKetThuc)\n")
    print("\t\t.KMean(mangCacDacTrung,n_clusters) \n")

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

```

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.

2. Hierarchical clustering(HC)

a. 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: <ul style="list-style-type: none"> 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

STT	Tên hàm	Mục đích của hàm	File lưu trữ	Tên sinh viên phụ trách
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

b. Mã nguồn:

❖ Các thư viện và trở 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()

```

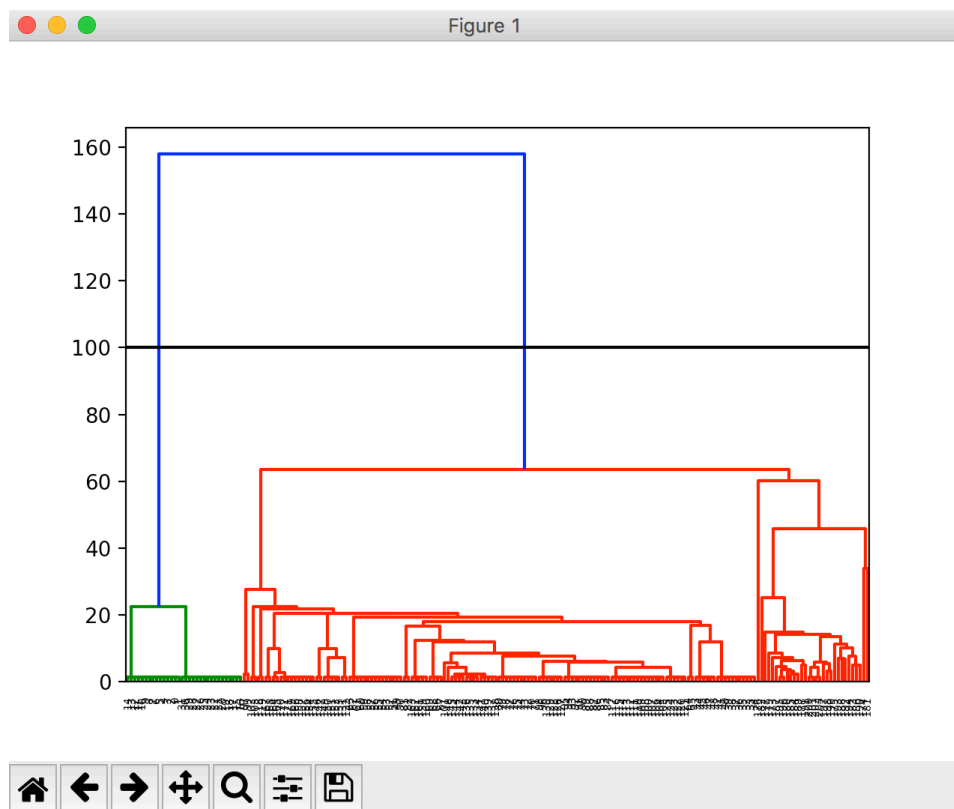
❖ **Hàm help: Hàm cung cấp cú pháp để dễ sử dụng**

```

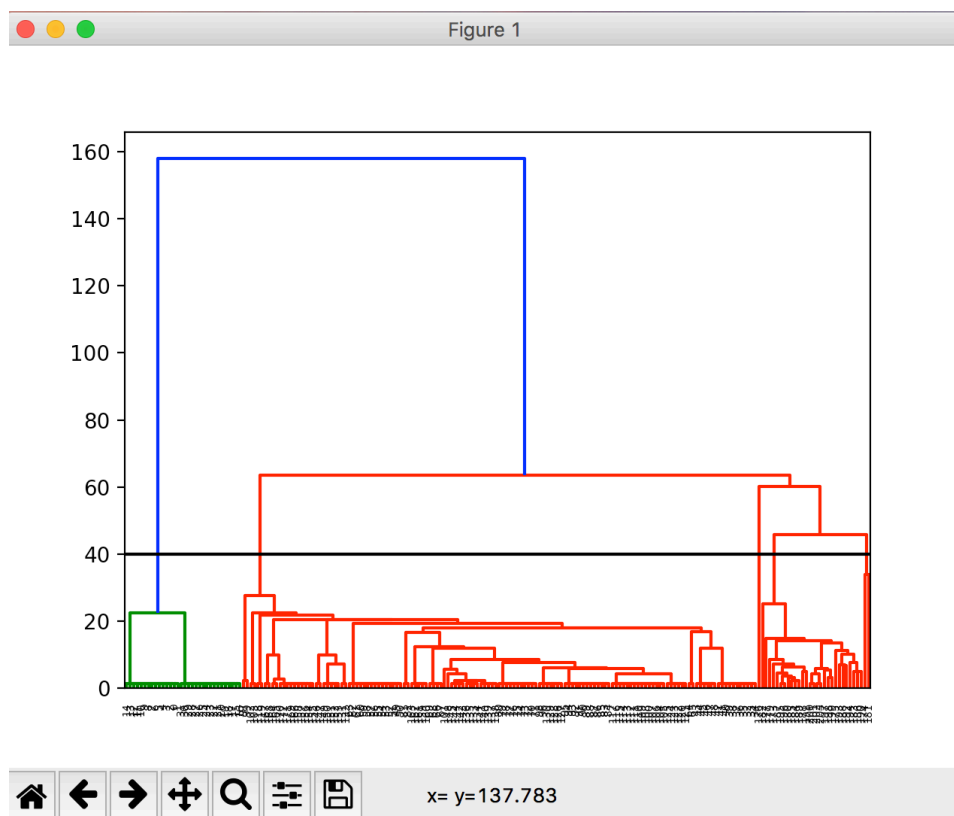
def help():
    print("\nhelp:\t\t HC(mangCacDacTrung,n_clusters) tra ve HC\n")
    print("\t\t ve_HC(mangCacDacTrung,n_clusters)\n")

```

c. 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

IV. TẦNG APP

1. 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.

2. 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

STT	Tên hàm	Mục đích của hàm	File lưu trữ	Tên sinh viên phụ trách
3	xuấtFile Input: <ul style="list-style-type: none"> 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

3. MÃ NGUỒN

a. 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'))
```

b. 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.

c. 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 thãi 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	<ul style="list-style-type: none"> Tiền xử lý dữ liệu(DAO, DTO, APP). Xử lý thuật toán KNN. 	<ul style="list-style-type: none"> Tiền xử lý dữ liệu(DAO, DTO, APP). Xử lý thuật toán KMeans. Xử lý thuật toán HC.
			Không
			Không