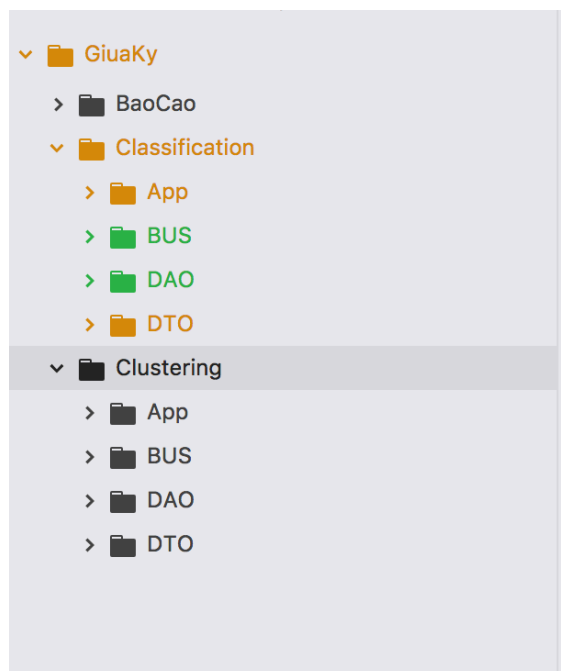


# PHẦN 1: CẤU TRÚC PHÂN TẦNG

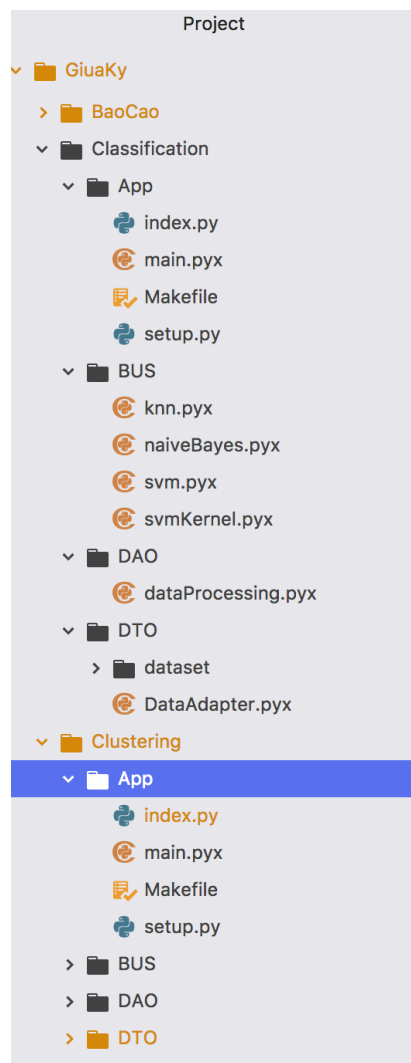
## I. TỔNG QUAN

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.



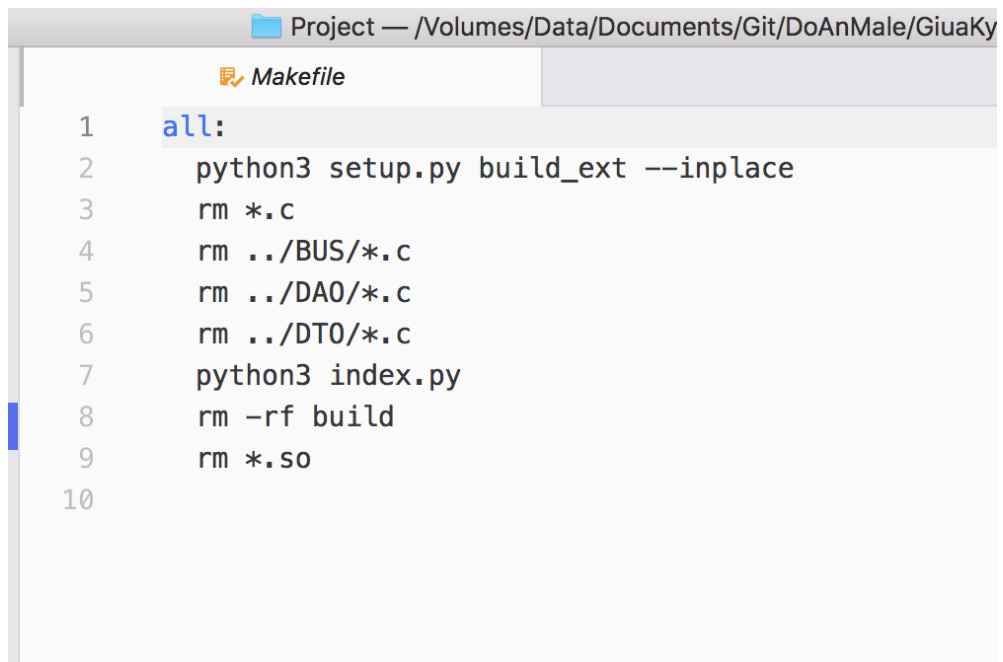
Hình 1: Cấu trúc 1.



Hình 2: Cấu trúc 2.

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

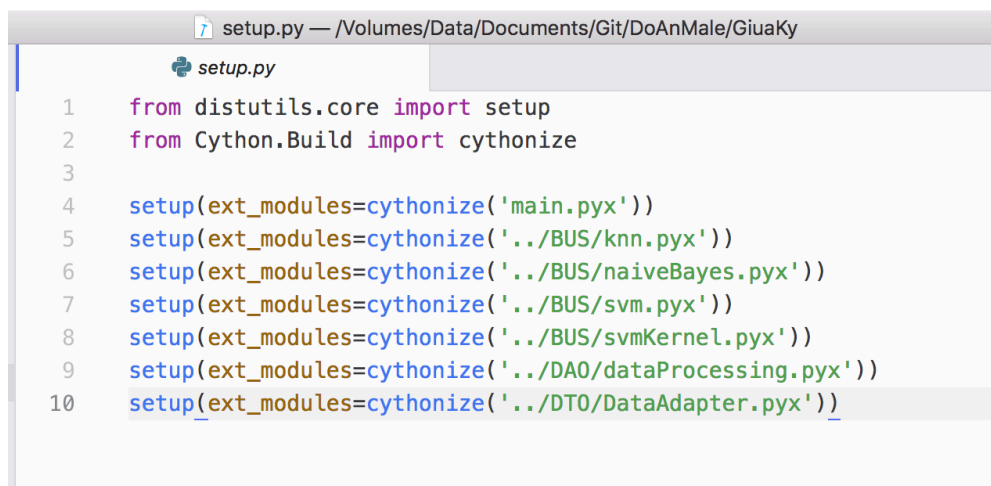
- Là ngôn ngữ nhằm hỗ trợ cú pháp của Python, C và được thiết kế để mang hiệu năng như ngôn ngữ C.
- Cython thực thi sẽ biên dịch ra mã nguồn C và xuất ra các module để chương trình python có thể gọi và thực thi.
- Cơ chế là đầu tiên Cython sẽ biên dịch ra mã nguồn C rồi tiếp đến là file .so và cuối 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:



The screenshot shows a code editor window titled "Project — /Volumes/Data/Documents/Git/DoAnMale/GiuaKy". The active file is "Makefile". The content of the Makefile is as follows:

```
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
```

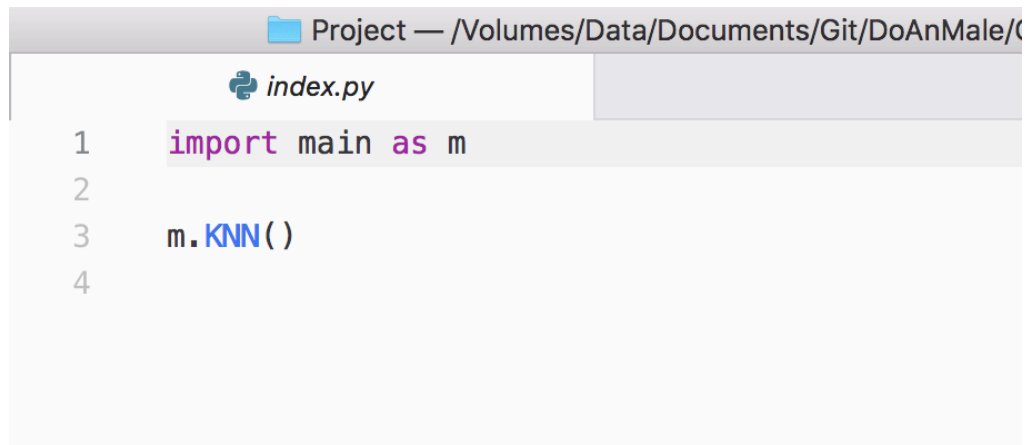
Hình 3: Cấu trúc Makefile.



The screenshot shows a code editor window titled "setup.py — /Volumes/Data/Documents/Git/DoAnMale/GiuaKy". The active file is "setup.py". The content of the setup.py file is as follows:

```
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__(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]
```

## 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 xoá trùng.</p> <p>Output: Mảng đã được xoá phần tử trùng.</p>	Xoá phần tử trùng trong 1 mảng lấy từ 1 cột trong dataset, cung cấp cho hàm ganSoPhanLoai	dataProcessi ng.pyx	Nguyễn Tiến Đạt
2	<p>ganSoPhanLoai</p> <p>Input: xoaTrung.</p> <p>Output: mảng đã được đánh số.</p>	Lấy phần tử đã xoá trùng từ hàm xoaTrung để đánh số tương ứng với mỗi phần tử.	dataProcessi ng.pyx	Nguyễn Tiến Đạt



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: 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
```

**d. 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
```

**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_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

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"> <li>Là tập dữ liệu vẽ nếu 0 tập có lọc, 1 tập không lọc.</li> </ul> <p>mangCacDacTrungVe:</p> <ul style="list-style-type: none"> <li>Là mảng các đặc trưng cần vẽ với None là tất cả.</li> </ul> <p>soLuongDiemVe:</p> <ul style="list-style-type: none"> <li>Là số lượng điểm vẽ.</li> </ul>	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"> <li>Là tập dữ liệu vẽ nếu 0 tập có lọc, 1 tập không lọc.</li> </ul> <p>mangCacDacTrungVe:</p> <ul style="list-style-type: none"> <li>Là mảng các đặc trưng cần vẽ với None là tất cả.</li> </ul> <p>soLuongDiemVe:</p> <ul style="list-style-type: none"> <li>Là số lượng điểm vẽ.</li> </ul>	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_Kh
    ongLoc)

    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:**

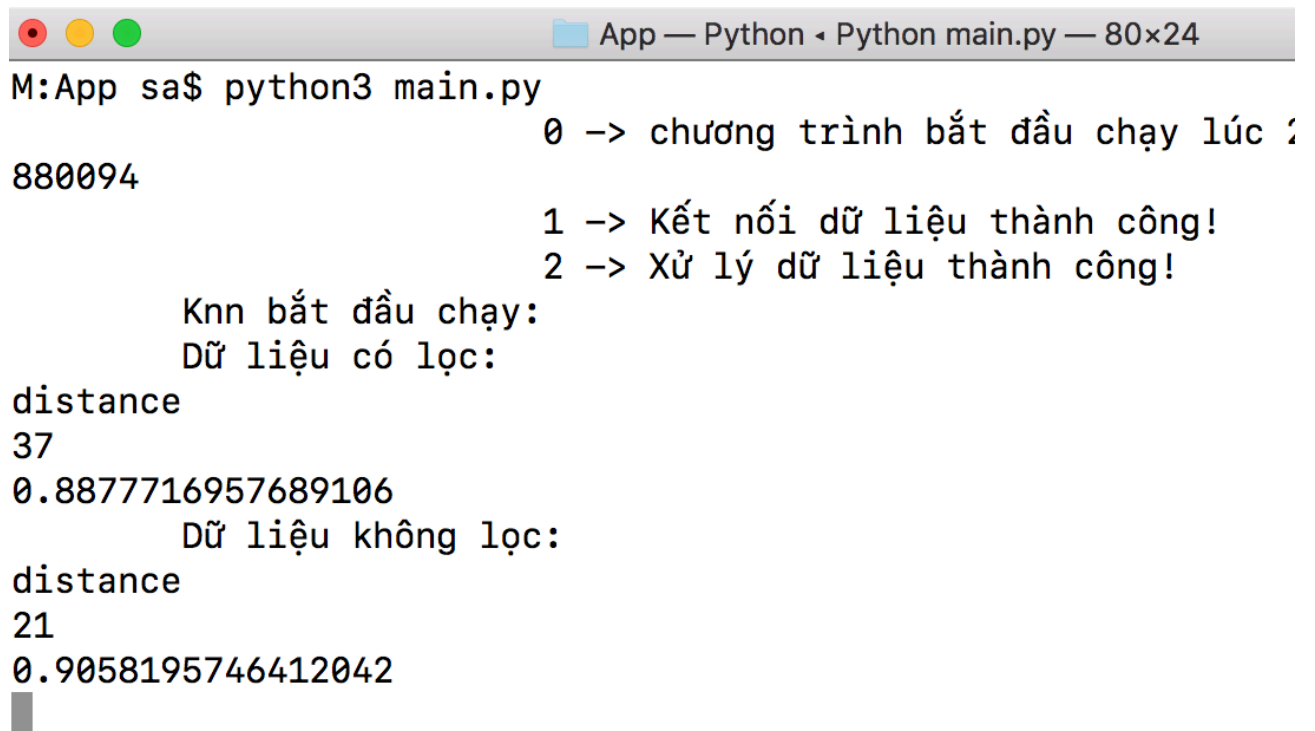
```

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



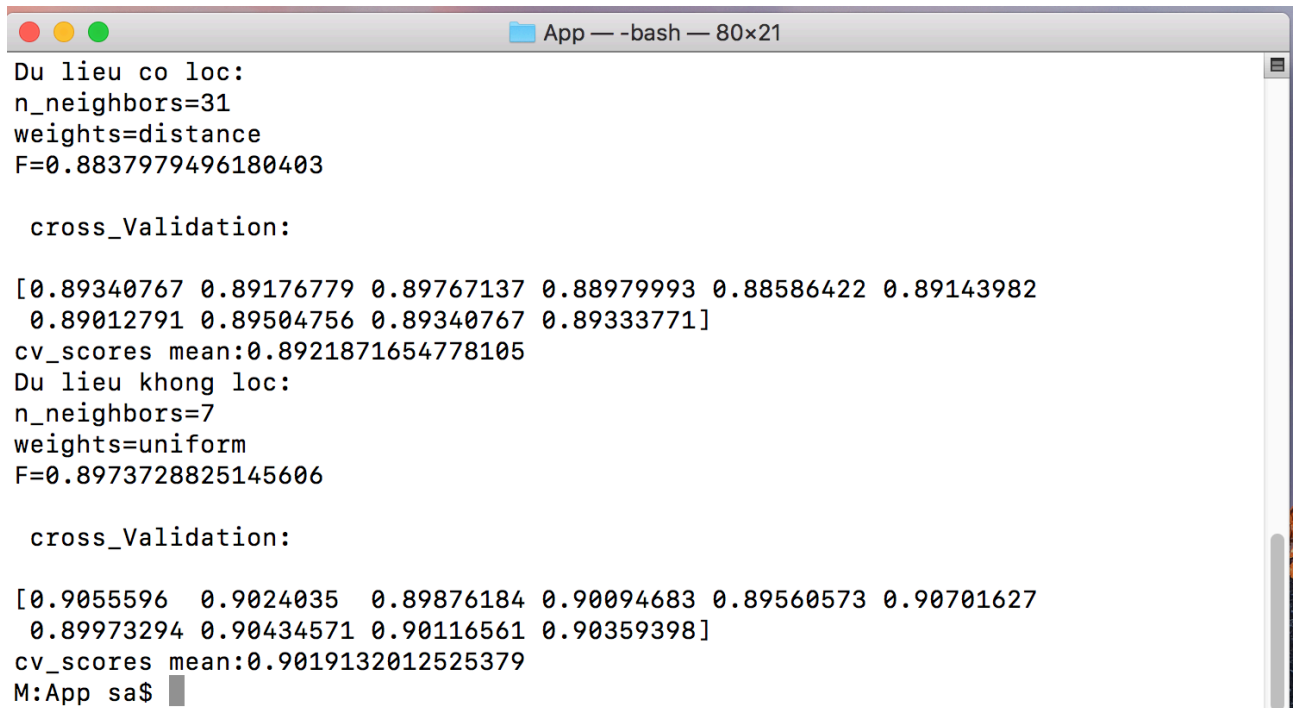
```
App — Python • Python main.py — 80x24
M:App sa$ python3 main.py
0 -> chương trình bắt đầu chạy lúc : 880094
Knn bắt đầu chạy:
Dữ liệu có lọc:
distance
37
0.8877716957689106
Dữ liệu không lọc:
distance
21
0.9058195746412042
```

Hình 6: Kết quả knn1.

Ở hình 6 em dùng 2 tập dữ liệu là có lọc và không lọc unknown 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:

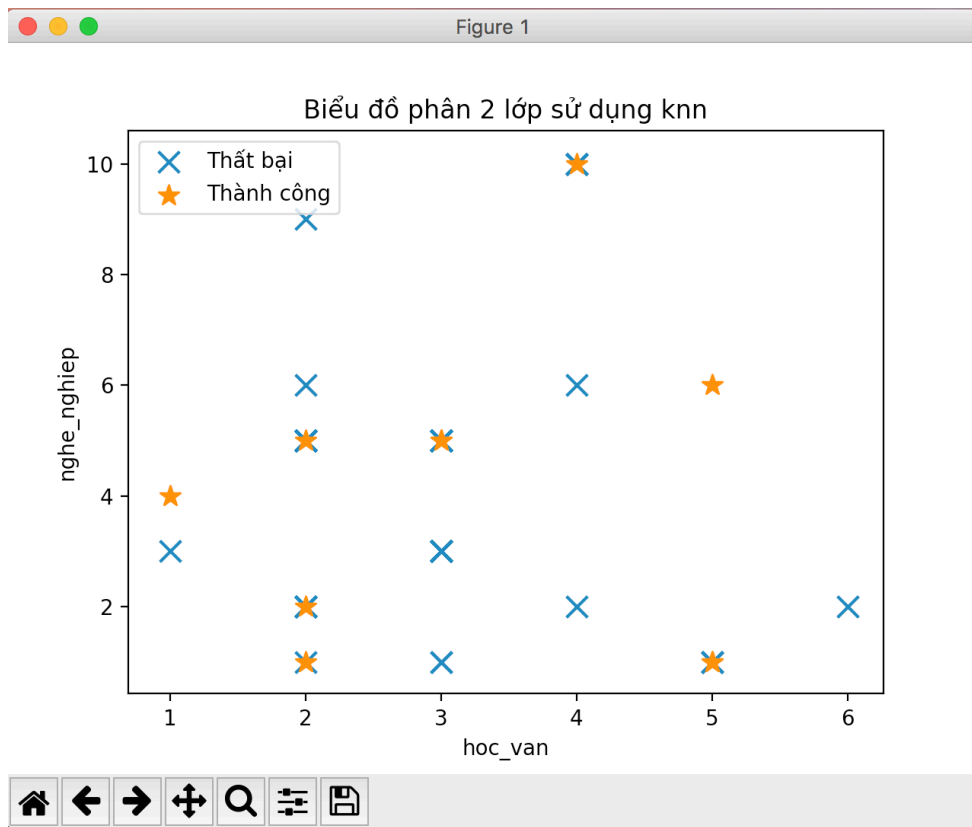
[0.89340767 0.89176779 0.89767137 0.88979993 0.88586422 0.89143982
 0.89012791 0.89504756 0.89340767 0.89333771]
cv_scores mean:0.8921871654778105
Du lieu khong loc:
n_neighbors=7
weights=uniform
F=0.8973728825145606

cross_Validation:

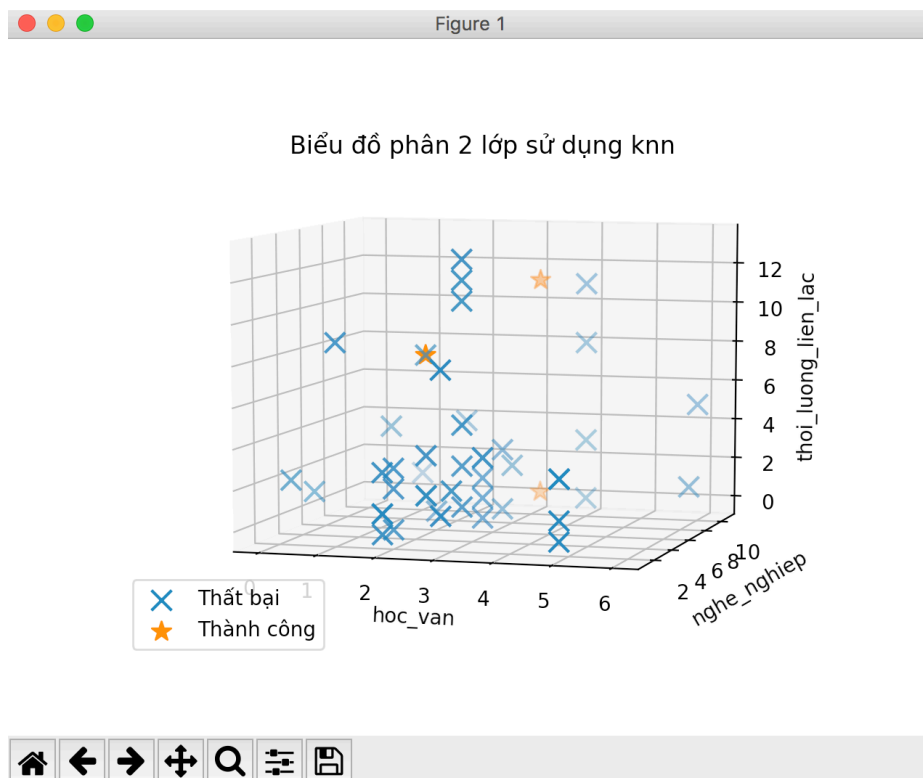
[0.9055596 0.9024035 0.89876184 0.90094683 0.89560573 0.90701627
 0.89973294 0.90434571 0.90116561 0.90359398]
cv_scores mean:0.9019132012525379
M:App sa$
```

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%.

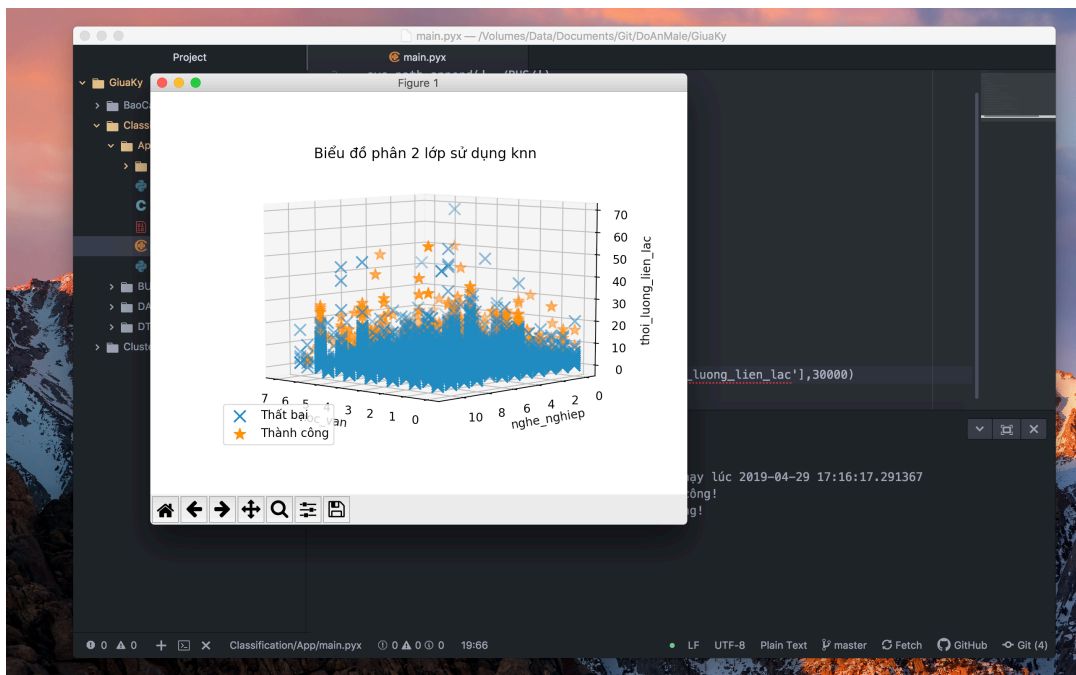


Hình 8: Vẽ 2d KNN.



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.

