

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**  
***TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM***  
**KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**



**BÁO CÁO CUỐI KỲ MACHINE LEARNING**

***Sinh viên thực hiện:***

Nguyễn Tiến Đạt - 16110048.

Đinh Công Minh - 16110153.

Đỗ Quốc Hùng - 16110097.

***Giảng viên phụ trách:***

Cô Trần Lê Minh Sang.

TP.HCM, tháng..... năm 2019

## This image shows a full page of white paper with horizontal dotted lines. The lines are evenly spaced and run across the entire width of the page, providing a guide for handwriting practice. There are no margins, text, or other markings on the page.

(ký ghi rõ họ tên)

# **CAM KẾT KHÔNG ĐẠO VĂN**

Nhóm xin cam đoan đề án này do chính nhóm thực hiện, không sao chép, sử dụng bất kỳ tài liệu, mã nguồn của người khác mà không ghi rõ nguồn gốc. Nhóm xin chịu hoàn toàn trách nhiệm nếu vi phạm.

# MỤC LỤC

<b>PHẦN 1: CẤU TRÚC PHÂN TẦNG</b>	<b>6</b>
TỔNG QUAN	6
GIAO DIỆN NGƯỜI DÙNG	10
CÀI ĐẶT	13
<b>PHẦN 2: CẤU TRÚC CHO PHÂN LỚP</b>	<b>13</b>
TẦNG DTO	13
CHỨC NĂNG	13
BẢNG HÀM VÀ LỚP	14
MÃ NGUỒN	15
Kết nối dataset	15
Lấy dataset theo từng cột của từng loại dataset	16
Xử lý song song 4 cột cùng lúc tương ứng với từng loại dataset	17
Lớp dataset: Nhiệm vụ lấy các data của hàm multiprocessing() để tạo ra 1 đối tượng dataset tương ứng cho tầng DAO sử dụng	18
TẦNG DAO	21
CHỨC NĂNG	21
BẢNG HÀM	21
MÃ NGUỒN	22
Các thư viện sử dụng và thiết đặc liên kết với tầng DTO	22
Hàm locKyTu lọc kí tự không cần thiết	22
Hàm chuyenDoiDuLieu biến dữ liệu chữ sang số	22
Hàm data cung cấp dataset đã biến đổi cho BUS	23
Hàm label_ cung cấp label tương ứng cho BUS	26
TẦNG BUS	27
Multivariable Regression	27
Cơ sở lý thuyết	27
Bảng hàm	27
Mã nguồn	29
PCA	34
Cơ sở lý thuyết	34
Bảng hàm	34
Mã nguồn	36
LDA	42
Cơ sở lý thuyết	42

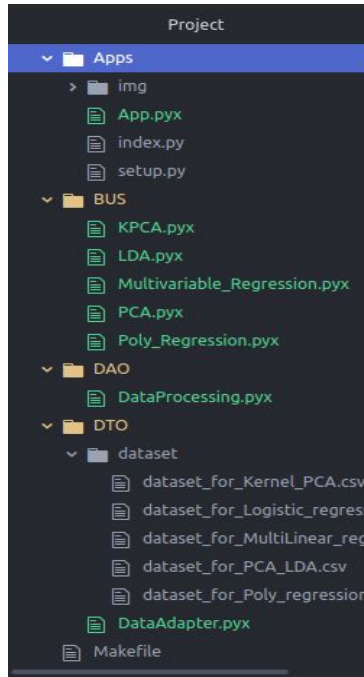
Bảng hàm	42
Mã nguồn	44
Kernel PCA	49
Polynomial Regression	49
Logistic Regression	49
TÀNG APP	49
CHỨC NĂNG	49
BẢNG HÀM VÀ LỚP	50
Mã nguồn	54
File setup.py	54
File index.py	55
File App.py	55
<b>PHẦN 3: ĐÁNH GIÁ</b>	<b>99</b>
MR	99
PCA	101
LDA	106
Logistic regression	127
KPCA	128
Polynomial Regression	132
<b>PHẦN 4: TÀI LIỆU THAM KHẢO</b>	<b>134</b>

# 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.

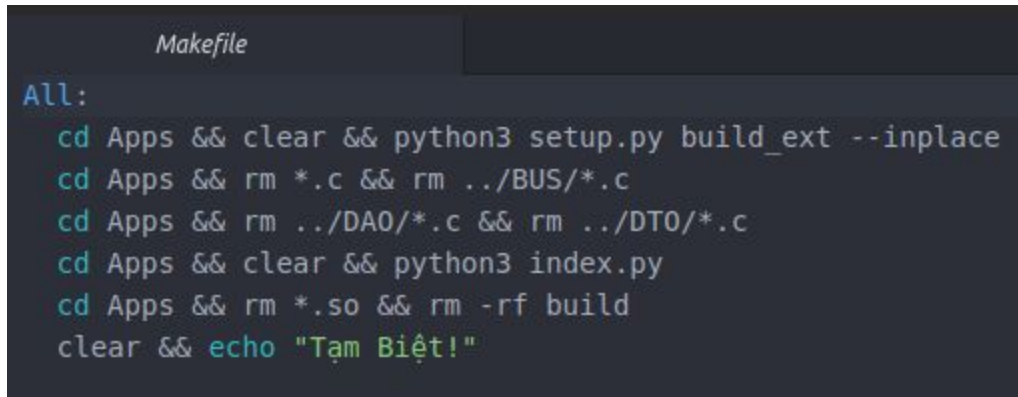


Hình 1: Cấu trúc

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

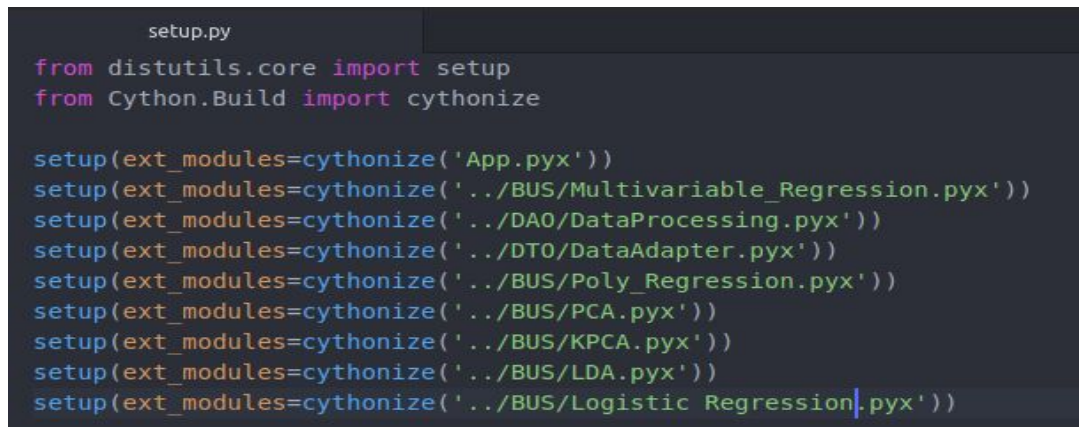
- Là ngôn ngữ nhằm hỗ trợ cú pháp của Python, C và được thiết kế để mang hiệu năng như ngôn ngữ C.
- Cython thực thi sẽ biên dịch ra mã nguồn C và xuất ra các module để chương trình python có thể gọi và thực thi.
- Cơ chế là đầu tiên Cython sẽ biên dịch ra mã nguồn C rồi tiếp đến là file .so và cuối cùng là file object(.o).
- Cython có đuôi mở rộng là .pyx nên mọi tệp Python (.py) cần được đổi sang .pyx mới thoả đầu vào biên dịch. Để biên dịch toàn bộ chương trình sử dụng Cython vì

có nhiều tầng nên chúng em sử dụng file setup.py nằm tại thư mục App để biên dịch cho tất cả các tầng và được thực thi thông qua Makefile:

A screenshot of a code editor showing a Makefile. The file has a tab labeled 'Makefile'. The content is as follows:

```
All:
    cd Apps && clear && python3 setup.py build_ext --inplace
    cd Apps && rm *.c && rm ../BUS/*.c
    cd Apps && rm ../DAO/*.c && rm ../DTO/*.c
    cd Apps && clear && python3 index.py
    cd Apps && rm *.so && rm -rf build
    clear && echo "Tạm Biệt!"
```

Hình 2: Cấu trúc Makefile.

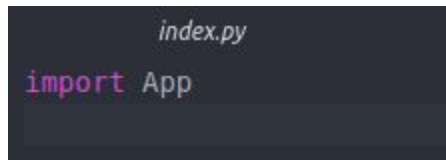
A screenshot of a code editor showing a setup.py file. The file has a tab labeled 'setup.py'. The content is as follows:

```
from distutils.core import setup
from Cython.Build import cythonize

setup(ext_modules=cythonize('App.pyx'))
setup(ext_modules=cythonize('../BUS/Multivariable_Regression.pyx'))
setup(ext_modules=cythonize('../DAO/DataProcessing.pyx'))
setup(ext_modules=cythonize('../DTO/DataAdapter.pyx'))
setup(ext_modules=cythonize('../BUS/Poly_Regression.pyx'))
setup(ext_modules=cythonize('../BUS/PCA.pyx'))
setup(ext_modules=cythonize('../BUS/KPCA.pyx'))
setup(ext_modules=cythonize('../BUS/LDA.pyx'))
setup(ext_modules=cythonize('../BUS/Logistic Regression.pyx'))
```

Hình 3: setup.py.





```
index.py
import App
```

Hình 4: index.py.

Từ ảnh 2, 3 và 4 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. GIAO DIỆN NGƯỜI DÙNG***

Do nhu cầu chọn đặc trưng tính toán hay vẽ và trực quan hóa nên giao diện là cần thiết giúp cho người sử dụng cảm thấy dễ dàng hơn nên chúng em có tích hợp giao diện trong thuật toán của mình là dùng bộ giao diện tkinter tuy không được đẹp mắt chuyên nghiệp so với những bộ khác như Qt, Kivy... nhưng cũng đủ cho mục đích học tập, nó phần nào góp phần tạo giao diện thân thiện. Mặc khác tkinter không hỗ trợ datagridview để tải dữ liệu hay entry(textbox) vẫn còn thô sơ nên giao diện chúng em có sử dụng thêm terminal để hiển thị dữ liệu nhưng vẫn chỉ là 1 nút bấm. Lý do chính ở đây tụi em chọn tkinter vì nó tích hợp sẵn trong python nên sẽ chạy được ở nhiều hệ điều hành có python. Ở bộ giao diện này còn nhiều thô sơ cần người lập trình phải làm việc nhiều mà thời gian có hạn nên tụi em chỉ thể hiện được 2 thuật toán lên giao diện là PCA và Multivariable Regression.

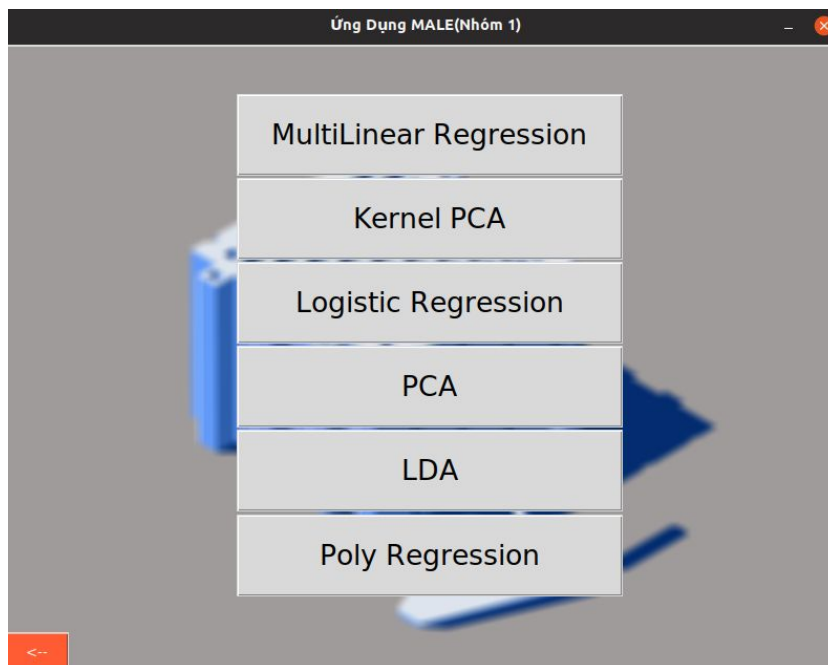
Giao diện sẽ có 3 phần:

- Đăng nhập: Là màn hình đầu tiên cần phải đăng nhập mới sử dụng được ở đây tài khoản là 'Nhom1' mật mã là '123456'.



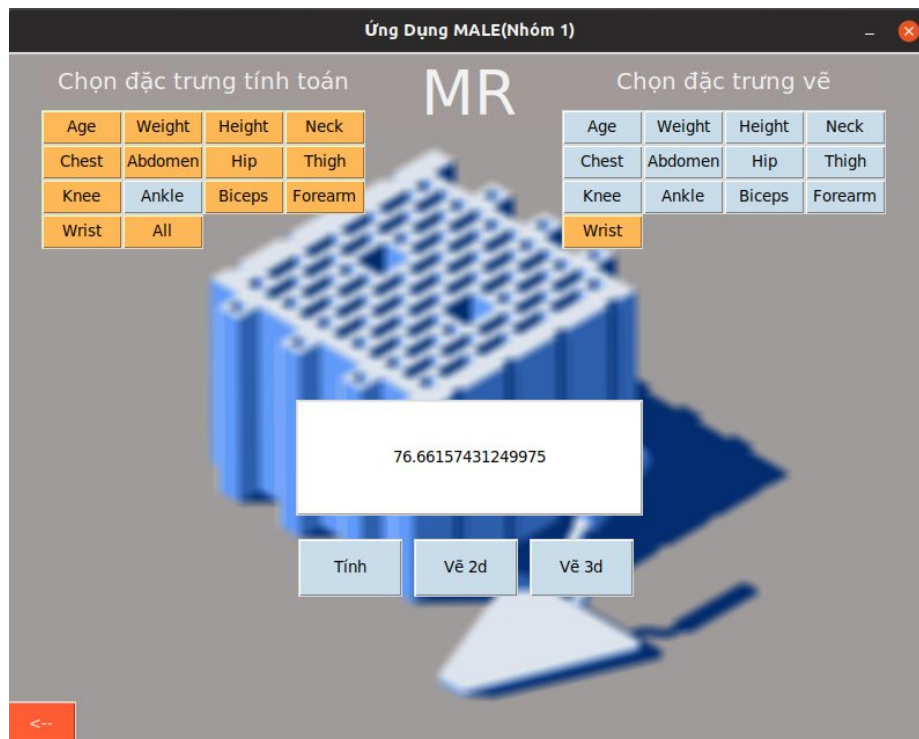
Hình 5: Đăng nhập.

- Màn hình chính: Là nơi chọn giải thuật muốn tính toán.

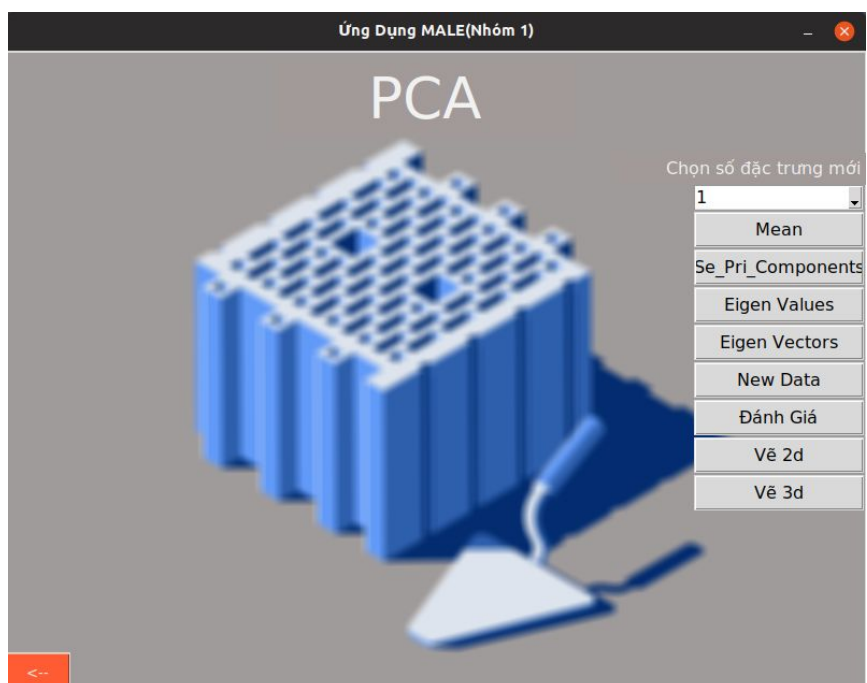


Hình 6: Màn hình chính.

- Màn hình giải thuật: Nơi thao tác các phép tính cho thuật toán hay vẽ.



Hình 7: Màn hình tính cho MR.



Hình 8: Màn hình tính cho PCA.

### ***III. CÀI ĐẶT***

Yêu cầu chương trình hoặc thư viện cần có trên:

- Python3
- Matplotlib
- Sklearn
- Pandas
- Cython
- Cmake/Gcc(Unix), mingw/Windows Subsystem for Linux(Windows)
- OpenCV
- tkinter
- Pillow
- colorama  $\geq 0.3.9$

## 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
1	get_dataset_for_Kernel_PCA	Lấy dataset cho Kernel PCA từ file lên và cung cấp method lấy từng cột tương ứng của dataset đó.	DataAdapter.pyx
2	get_dataset_for_Logistic_regression	Lấy dataset cho Logistic regression từ file lên và cung cấp method lấy từng cột tương ứng của dataset đó.	DataAdapter.pyx
3	get_dataset_for_MultiLinear_regression	Lấy dataset cho MultiLinear regression từ file lên và cung cấp method lấy từng cột tương ứng của dataset đó.	DataAdapter.pyx

4	get_dataset_for_PCA_LDA	Lấy dataset cho PCA/LDA từ file lên và cung cấp method lấy từng cột tương ứng của dataset đó.	DataAdapter.pyx
5	get_dataset_for_Poly_regression	Lấy dataset cho Poly Regression từ file lên và cung cấp method lấy từng cột tương ứng của dataset đó.	DataAdapter.pyx
6	multiprocessing	Nhiệm vụ là xử lý song song 4 cột cùng 1 lúc để tạo ra đối tượng data cho đối tượng dữ liệu class dataset.	DataAdapter.pyx

Bảng 2: Lớp

STT	Tên lớp	Mục đích của lớp
1	dataset	Trả về đối tượng dữ liệu dataset tương ứng với từng thuật toán cho tầng DAO sử dụng.

### 3. MÃ NGUỒN

#### *A. Kết nối dataset*

```
# coding=utf-8

import pandas as pd

from multiprocessing.dummy import Pool as ThreadPool

dataset_for_Kernel_PCA=pd.read_csv('../DTO/dataset/dataset_for_Kernel_PCA.csv',en
coding='utf-8')

dataset_for_Logistic_regression=pd.read_csv('../DTO/dataset/dataset_for_Logistic_regr
ession.csv',encoding='utf-8')

dataset_for_MultiLinear_regression=pd.read_csv('../DTO/dataset/dataset_for_MultiLine
ar_regression.csv',encoding='utf-8')

dataset_for_PCA_LDA=pd.read_csv('../DTO/dataset/dataset_for_PCA_LDA.csv',encod
ing='utf-8')

dataset_for_Poly_regression=pd.read_csv('../DTO/dataset/dataset_for_Poly_regression.c
sv',encoding='utf-8')
```

#### *B. Lấy dataset theo từng cột của từng loại dataset*

- **Cho Kernel\_PCA**

```
def get_dataset_for_Kernel_PCA(name):
    return (dataset_for_Kernel_PCA[name].values).tolist()
```

- **Cho Logistic\_regression**

```
def get_dataset_for_Logistic_regression(name):
```

```
return (dataset_for_Logistic_regression[name].values).tolist()
```

- **Cho MultiLinear\_regression**

```
def get_dataset_for_MultiLinear_regression(name):  
    return (dataset_for_MultiLinear_regression[name].values).tolist()
```

- **Cho PCA/LDA**

```
def get_dataset_for_PCA_LDA(name):  
    return (dataset_for_PCA_LDA[name].values).tolist()
```

- **Cho Poly\_regression**

```
def get_dataset_for_Poly_regression(name):  
    return (dataset_for_Poly_regression[name].values).tolist()
```

### ***C. Xử lý song song 4 cột cùng lúc tương ứng với từng loại dataset***

```
def multiprocessing(k):  
    if k==0:#dataset_for_Kernel_PCA  
        t=['User ID','Gender','Age','EstimatedSalary','Purchased']  
        pool = ThreadPool(4)  
        data=pool.map(get_dataset_for_Kernel_PCA,t)  
    elif k==1:#dataset_for_Logistic_regression  
        t=['Sex','AgeRange','Class_','SiblingSpouse','ParentChild','Embarked_','Survived']  
        pool = ThreadPool(4)  
        data=pool.map(get_dataset_for_Logistic_regression,t)  
    elif k== 2: #dataset_for_MultiLinear_regression:  
        t=['Age','Weight','Height','Neck','Chest','Abdomen','Hip','Thigh','Knee','Ankle','Biceps','Forearm','Wrist','FAT_PER']#dataset_for_MultiLinear_regression  
        pool = ThreadPool(4)  
        data=pool.map(get_dataset_for_MultiLinear_regression,t)  
    elif k==3:#dataset_for_PCA_LDA
```



```

t=['LOC_BLANK','BRANCH_COUNT',
'CALL_PAIRS','LOC_CODE_AND_COMMENT','LOC_COMMENTS','
CONDITION_COUNT','CYCLOMATIC_COMPLEXITY','CYCLOMAT
IC_DENSITY','DECISION_COUNT','DECISION_DENSITY','DESIGN_
COMPLEXITY','DESIGN_DENSITY','EDGE_COUNT','ESSENTIAL_C
OMPLEXITY','ESSENTIAL_DENSITY','LOC_EXECUTABLE','PARA
METER_COUNT','GLOBAL_DATA_COMPLEXITY','GLOBAL_DAT
A_DENSITY','HALSTEAD_CONTENT','HALSTEAD_DIFFICULTY','
HALSTEAD_EFFORT','HALSTEAD_ERROR_EST',
'HALSTEAD_LENGTH','HALSTEAD_LEVEL','HALSTEAD_PROG_TI
ME','HALSTEAD_VOLUME','MAINTENANCE_SEVERITY','MODIFI
ED_CONDITION_COUNT','MULTIPLE_CONDITION_COUNT','FAIL
DE_COUNT','FAILRMALIZED_CYLOMATIC_COMPLEXITY','NUM
_OPERANDS','NUM_OPERATORS','NUM_UNIQUE_OPERANDS','N
UM_UNIQUE_OPERATORS','NUMBER_OF_LINES','PERCENT_CO
MMMENTS','LOC_TOTAL','TEST_RESULT']#dataset_for_PCA_LDA
pool = ThreadPool(4)
data=pool.map(get_dataset_for_PCA_LDA,t)
elif k==4:#dataset_for_Poly_regression
t=['Rating', 'Reviews', 'Size', 'Installs']
pool = ThreadPool(4)
data=pool.map(get_dataset_for_Poly_regression,t)
return data

```

***D. Lớp dataset: Nhiệm vụ lấy các data của hàm multiprocessing() để tạo ra 1 đối tượng dataset tương ứng cho từng DAO sử dụng***

```

class dataset:
    def __init__(self,k):
        data=multiprocessing(k)

```

```
if k==0:
seft.User_ID=data[0]
seft.Gender=data[1]
seft.Age=data[2]
seft.EstimatedSalary=data[3]
seft.Purchased =data[4]
elif k==1:
seft.Sex=data[0]
seft.AgeRange=data[1]
seft.Class_=data[2]
seft.SiblingSpouse=data[3]
seft.ParentChild=data[4]
seft.Embarked_=data[5]
seft.Survived=data[6]
elif k==2:
seft.Age=data[0]
seft.Weight=data[1]
seft.Height=data[2]
seft.Neck=data[3]
seft.Chest=data[4]
seft.Abdomen=data[5]
seft.Hip=data[6]
seft.Thigh=data[7]
seft.Knee=data[8]
seft.Ankle=data[9]
seft.Biceps=data[10]
seft.Forearm=data[11]
seft.Wrist=data[12]
seft.FAT_PER=data[13]
```

```
elif k==3:
seft.LOC_BLANK=data[0]
seft.BRANCH_COUNT=data[1]
seft.CALL_PAIRS=data[2]
seft.LOC_CODE_AND_COMMENT=data[3]
seft.LOC_COMMENTS=data[4]
seft.CONDITION_COUNT=data[5]
seft.CYCLOMATIC_COMPLEXITY=data[6]
seft.CYCLOMATIC_DENSITY=data[7]
seft.DECISION_COUNT=data[8]
seft.DECISION_DENSITY=data[9]
seft.DESIGN_COMPLEXITY=data[10]
seft.DESIGN_DENSITY=data[11]
seft.EDGE_COUNT=data[12]
seft.ESSENTIAL_COMPLEXITY=data[13]
seft.ESSENTIAL_DENSITY=data[14]
seft.LOC_EXECUTABLE=data[15]
seft.PARAMETER_COUNT=data[16]
seft.GLOBAL_DATA_COMPLEXITY=data[17]
seft.GLOBAL_DATA_DENSITY=data[18]
seft.HALSTEAD_CONTENT=data[19]
seft.HALSTEAD_DIFFICULTY=data[20]
seft.HALSTEAD_EFFORT=data[21]
seft.HALSTEAD_ERROR_EST=data[22]
seft.HALSTEAD_LENGTH=data[23]
seft.HALSTEAD_LEVEL=data[24]
seft.HALSTEAD_PROG_TIME=data[25]
seft.HALSTEAD_VOLUME=data[26]
```

```

seft.MAINTENANCE_SEVERITY=data[27]
seft.MODIFIED_CONDITION_COUNT=data[28]
seft.MULTIPLE_CONDITION_COUNT=data[29]
seft.FAILDE_COUNT=data[30]
seft.FAILRMALIZED_CYLOMATIC_COMPLEXITY=data[31]
seft.NUM_OPERANDS=data[32]
seft.NUM_OPERATORS=data[33]
seft.NUM_UNIQUE_OPERANDS=data[34]
seft.NUM_UNIQUE_OPERATORS=data[35]
seft.NUMBER_OF_LINES=data[36]
seft.PERCENT_COMMENTS=data[37]
seft.LOC_TOTAL=data[38]
seft.TEST_RESULT=data[39]
elif k==4:
seft.Rating=data[0]
seft.Reviews=data[1]
seft.Size=data[2]
seft.Installs=data[3]

```

## ***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
1	locKyTu Input: Tập dữ liệu thứ k chưa đọc lọc. Output: Tập dữ liệu thứ k đã được lọc.	Dùng lọc kí tự đặc biệt trong dataset như '+', ','	DataProcessing.pyx
2	chuyenDoiDuLieu Input: Tập dữ liệu thứ k chưa được chuyển đổi. Output: Tập dữ liệu thứ k đã được chuyển đổi.	Dùng để chuyển đổi chuỗi kí hiệu cho số như 'M', 'k' để chuyển sang dạng số hoàn toàn.	DataProcessing.pyx
3	data Input: Tập dữ liệu thứ k chưa được biến đổi. Output: Tập dữ liệu thứ k đã được biến đổi.	Tạo ra tập dữ liệu mới đã xử lý kí tự đặc biệt hay kí tự biểu diễn ý nghĩa số tương ứng của 6 loại dataset và cung cấp lên tầng BUS. Để sử dụng chỉ cần truyền vị trí k tương ứng dataset thứ k.	DataProcessing.pyx
4	label_ Input: Tập dữ liệu thứ k. Output: label tương ứng.	Cung cấp label tương ứng với tập dữ liệu có label lên BUS để sử dụng.	DataProcessing.pyx

### 3. MÃ NGUỒN

#### A. Các thư viện sử dụng và thiết lập liên kết với tầng DTO

```
# coding=utf-8
# thu vien xu ly duong dan
import sys
sys.path.append('../DTO/')
import DataAdapter as da
```

#### B. Hàm locKyTu lọc kí tự không cần thiết

```
def locKyTu(k):
```

```

s=""
for i in k:
    if i!=',' and i!='+':
        s+=i
return int(s)

```

### ***C. Hàm chuyenDoiDuLieu biến dữ liệu chữ sang số***

```

def chuyenDoiDuLieu(k):
    s=""
    for i in k:
        if i!='M' and i!='k':
            s+=i
    if k[len(k)-1]=='k':
        return float(s)*1000
    elif k[len(k)-1]=='M':
        return float(s)*1000000

```

### ***D. Hàm data cung cấp dataset đã biến đổi cho BUS***

```

def data(k):
    dt=[]
    tmp=da.dataset(k)
    if k==0:# dataset_for_Kernel_PCA
        dem=0
        for i in range(len(tmp.User_ID)):
            if tmp.Gender[i]=='Male':
                dt.append([dem,0,tmp.Age[i],
                           tmp.EstimatedSalary[i]/1000])
            elif tmp.Gender[i]=='Female':
                dt.append([dem,1,tmp.Age[i],

```

```

        tmp.EstimatedSalary[i]/1000))
    dem+=1
elif k==1:# dataset_for_Logistic_regression
    Sex=0
    AgeRange=0
    Class_=0
    Embarked_=0
    for i in range(len(tmp.Sex)):
        if tmp.Sex[i]=='Male':
            Sex=0
        elif tmp.Sex[i]=='Female':
            Sex=1
        if tmp.AgeRange[i]=='Age_0_9':
            AgeRange=0
        elif tmp.AgeRange[i]=='Age_10_19':
            AgeRange=1
        elif tmp.AgeRange[i]=='Age_20_29':
            AgeRange=2
        elif tmp.AgeRange[i]=='Age_30_39':
            AgeRange=3
        elif tmp.AgeRange[i]=='Age_40_49':
            AgeRange=4
        elif tmp.AgeRange[i]=='Age_50_59':
            AgeRange=5
        elif tmp.AgeRange[i]=='Age_60_69':
            AgeRange=6
        elif tmp.AgeRange[i]=='Age_70_plus':
            AgeRange=7
        if tmp.Class_[i]=='Class1':

```

```

        Class_=0
    elif tmp.Class_[i]=='Class2':
        Class_=1
    elif tmp.Class_[i]=='Class3':
        Class_=2
    if tmp.Embarked_[i]=='Southampton':
        Embarked_=0
    elif tmp.Embarked_[i]=='Cherbourg':
        Embarked_=1
    elif tmp.Embarked_[i]=='Queenstown':
        Embarked_=2
    elif tmp.Embarked_[i]=='Southampton':
        Embarked_=3

    dt.append([Sex,AgeRange,Class_,tmp.SiblingSpouse[i],tmp.ParentChild[i],
    Embarked_])
elif k==2:# dataset_for_MultiLinear_regression
    for i in range(len(tmp.Age)):
        dt.append([tmp.Age[i],tmp.Weight[i],tmp.Height[i],
        tmp.Neck[i],tmp.Chest[i],tmp.Abdomen[i],tmp.Hip[i],
        tmp.Thigh[i],tmp.Knee[i],tmp.Ankle[i],tmp.Biceps[i],
        tmp.Forearm[i],tmp.Wrist[i]])
elif k==3:# dataset_for_PCA_LDA
    for i in range(len(tmp.TEST_RESULT)):
        dt.append([
            tmp.LOC_BLANK[i],
            tmp.BRANCH_COUNT[i],tmp.CALL_PAIRS[i],tmp.LOC_CODE
            _AND_COMMENT[i],
            tmp.LOC_COMMENTS[i],tmp.CONDITION_COUNT[i],tmp.CY

```



```

        CLOMATIC_COMPLEXITY[i],
        tmp.CYCLOMATIC_DENSITY[i],tmp.DECISION_COUNT[i],tm
        p.DECISION_DENSITY[i],
        tmp.DESIGN_COMPLEXITY[i],tmp.DESIGN_DENSITY[i],tmp.
        EDGE_COUNT[i],tmp.ESSENTIAL_COMPLEXITY[i]
        ,tmp.ESSENTIAL_DENSITY[i],tmp.LOC_EXECUTABLE[i]

        ,tmp.PARAMETER_COUNT[i],tmp.GLOBAL_DATA_COMPLE
        XITY[i],tmp.GLOBAL_DATA_DENSITY[i]
        ,tmp.HALSTEAD_CONTENT[i],tmp.HALSTEAD_DIFFICULTY
        [i],tmp.HALSTEAD_EFFORT[i],
        tmp.HALSTEAD_ERROR_EST[i],tmp.HALSTEAD_LENGTH[i],
        tmp.HALSTEAD_LEVEL[i],
        tmp.HALSTEAD_PROG_TIME[i],tmp.HALSTEAD_VOLUME[i]
        ,tmp.MAINTENANCE_SEVERITY[i],
        tmp.MODIFIED_CONDITION_COUNT[i],tmp.MULTIPLE_CON
        DITION_COUNT[i],tmp.FAILDE_COUNT[i],
        tmp.FAILRMALIZED_CYLOMATIC_COMPLEXITY[i],tmp.NU
        M_OPERANDS[i],tmp.NUM_OPERATORS[i],
        tmp.NUM_UNIQUE_OPERANDS[i],tmp.NUM_UNIQUE_OPER
        ATORS[i],tmp.NUMBER_OF_LINES[i],
            tmp.PERCENT_COMMENTS[i],tmp.LOC_TOTAL[i]
        ])
    elif k==4:# dataset_for_Poly_regression
        for i in range(len(tmp.Rating)):

            dt.append([tmp.Rating[i],tmp.Reviews[i],chuyenDoiDuLieu(tmp.Si
            ze[i]))])

    return dt

```

### ***E. Hàm label\_ cung cấp label tương ứng cho BUS***

```
def label_(k):
    tmp=da.dataset(k)
    dt=[]
    if k==0:
        for i in range(len(tmp.User_ID)):
            dt.append(tmp.Purchased[i])
    elif k==1:
        for i in range(len(tmp.Sex)):
            dt.append(tmp.Survived[i])
    elif k==2:
        for i in range(len(tmp.Age)):
            dt.append(tmp.FAT_PER[i])
    elif k==3:
        TEST_RESULT=0
        for i in range(len(tmp.TEST_RESULT)):
            if tmp.TEST_RESULT[i]=='FAIL':
                TEST_RESULT=0
            elif tmp.TEST_RESULT[i]=='PASS':
                TEST_RESULT=1
            dt.append(TEST_RESULT)
    elif k == 4:
        for i in range(len(tmp.Installs)):
            dt.append(locKyTu(tmp.Installs[i]))
    return dt
```

### III. TẦNG BUS

#### 1. Multivariable Regression

##### A. Cơ sở lý thuyết

Phân tích hồi quy tuyến tính là một phương pháp phân tích quan hệ giữa biến phụ thuộc Y với một hay nhiều biến độc lập X. Mô hình hóa sử dụng hàm tuyến tính (bậc 1). Các tham số của mô hình (hay hàm số) được ước lượng từ dữ liệu.

##### B. Bảng hàm

STT	Tên hàm	Mục đích của hàm	File lưu trữ
1	MR Input: Tập dữ liệu đã chia training và testing. Output: Kết quả qua thuật toán MR.	Là hàm tính của thuật toán MR.	Multivariable_Regr ession.py
2	vitriDacTrung Input: Tên đặc trưng. Output: Vị trí đặc trưng trong dataset.	Là hàm trả về vị trí đặc trưng trong dataset cung cấp cho hàm layviTriDacTrung	Multivariable_Regr ession.py
3	tenDacTrung Input: Vị trí đặc trưng. Output: Tên đặc trưng.	Là hàm trả về tên đặc trưng.	Multivariable_Regr ession.py

4	layviTriDacTrung Input: Mảng các đặc trưng. Output: Mảng vị trí đặc trưng tương ứng.	Hàm trả về mảng vị trí đặc trưng từ mảng tên đặc trưng.	Multivariable_Regr ession.py
5	layDacTrung Input: Mảng các đặc trưng. Output: Mảng các cột đặc trưng tương ứng trong dataset	Trả về 1 DataFrame tương ứng với các đặc trưng.	Multivariable_Regr ession.py
6	ve2D Input: Mảng các đặc trưng và đặc trưng vẽ. Output: vẽ 2d.	Biểu diễn $Y^{\wedge}=a+bX1$ trên không gian 2 chiều.	Multivariable_Regr ession.py
7	ve3D Input: Mảng các đặc trưng và đặc trưng vẽ. Output: vẽ 3d.	Biểu diễn $Y^{\wedge}=a+bX1+cX2$ trên không gian 3 chiều.	Multivariable_Regr ession.py
8	tinhToan Input: Mảng các đặc trưng. Output: kết quả tính MR.	Hàm tính toán kết quả sử dụng hàm MR.	Multivariable_Regr ession.py

### ***C. Mã nguồn***

- Các thư viện và trở liên kết tới DAO

```
import sys
sys.path.append('../DAO/')
# chia tap du lieu ban dau thanh 2 tap la training va testing
from sklearn.model_selection import train_test_split
import DataProcessing as dp
# thu vien thuan toan LinearRegression
from sklearn.linear_model import LinearRegression
# danh gia
from sklearn.metrics import r2_score
#chon dac trung dung cay quyet dinh
from sklearn import tree
#chuyen sang anh
import pydotplus
#xu ly matrix
import numpy as np
# thu vien ve cua python
import matplotlib.pyplot as plt
import pandas as pd
from mpl_toolkits.mplot3d import Axes3D
data=dp.data(2)
label_=dp.label_(2)
```

- Hàm MR trả về kết quả tính toán

```
def MR(x_train,y_train,x_test,y_test):
    regressor = LinearRegression()
    regressor.fit(x_train, y_train)
    y_pred = regressor.predict(x_test)
```

```
score=r2_score(y_test,y_pred)
k=[]
k.append([score,y_pred])
return np.array(k)
```

- Hàm vitriDacTrung trả về vị trí của đặc trưng

```
def vitriDacTrung(dacTrung):
    if dacTrung=='Age':
        return 0
    elif dacTrung=='Weight':
        return 1
    elif dacTrung=='Height':
        return 2
    elif dacTrung=='Neck':
        return 3
    elif dacTrung=='Chest':
        return 4
    elif dacTrung=='Abdomen':
        return 5
    elif dacTrung=='Hip':
        return 6
    elif dacTrung=='Thigh':
        return 7
    elif dacTrung=='Knee':
        return 8
    elif dacTrung=='Ankle':
        return 9
    elif dacTrung=='Biceps':
        return 10
    elif dacTrung=='Forearm':
```

```

        return 11
    elif dacTrung=='Wrist':
        return 12

```

- Hàm tenDacTrung trả về tên của đặc trưng

```

def tenDacTrung(viTri):
    switcher={
        0:'Age',
        1:'Weight',
        2:'Height',
        3:'Neck',
        4:'Chest',
        5:'Abdomen',
        6:'Hip',
        7:'Thigh',
        8:'Knee',
        9:'Ankle',
        10:'Biceps',
        11:'Forearm',
        12:'Wrist'
    }
    return switcher.get(viTri)

```

- Hàm layviTriDacTrung trả về mảng các đặc trưng

```

def layviTriDacTrung(mangCacDacTrung):
    m=[]
    for i in mangCacDacTrung:
        m.append(viTriDacTrung(i))
    return m

```

- Hàm layDacTrung trả về 1 DataFrame tương ứng với mảng đặc trưng

```

def layDacTrung(mangCacDacTrung):

```

```

return pd.DataFrame(data)

[layviTriDacTrung
(mangCacDacTrung)]

```

- Hàm ve2D biểu diễn phương trình tương ứng với 1 đặc trưng  $Y^=a+bx$

```
def ve2D(mangCacDacTrung,dacTrungVe):
```

```

    x_train, x_test, y_train,
    y_test=train_test_split(layDacTrung(mangCacDacTrung),label_,tes
    t_size=0.2)
    m1=MR(x_train[layviTriDacTrung(dacTrungVe)],y_train,x_test[la
    yviTriDacTrung(dacTrungVe)],y_test)
    y_pred=[]
    for i in range(len((m1[:,1])[0])):
        y_pred.append((m1[:,1])[0][i])

```

```

plt.scatter(x_test[layviTriDacTrung(dacTrungVe)],y_test,s=10,c="#
3366cc")
plt.plot(x_test[layviTriDacTrung(dacTrungVe)],y_pred,
color='#cc0099', linewidth=3)
plt.xlabel("%s(Xj)"%dacTrungVe)
plt.ylabel('Y')
plt.show()

```

- Hàm ve3D biểu diễn phương trình tương ứng với 2 đặc trưng

$Y^=a+bx_1+cx_2$

```
def ve3D(mangCacDacTrung,dacTrungVe):
```

```

    x_train, x_test, y_train,
    y_test=train_test_split(layDacTrung(mangCacDacTrung),label_,tes
    t_size=0.2)
    m0=MR(x_train,y_train,x_test,y_test)
    y_pred0=[]

```



```

for i in range(len((m0[:,1])[0])):
    y_pred0.append((m0[:,1])[0][i])
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(x_test[layviTriDacTrung([dacTrungVe[0]])],x_test[layviTriDacTrung([dacTrungVe[1]])],y_test)
ax.plot(x_test[layviTriDacTrung([dacTrungVe[0]])],x_test[layviTriDacTrung([dacTrungVe[1]])], y_pred0)
plt.show()

```

- Hàm tinhToan là hàm tính toán chính cho thuật toán MR sử dụng lại hàm MR để tính giá trị quy đổi đơn vị %

```

def tinhToan(mangCacDacTrung):
    x_train, x_test, y_train,
    y_test=train_test_split(layDacTrung(mangCacDacTrung),label_,test_size=0.2)
    return float(MR(x_train,y_train,x_test,y_test)[:,0])*100

```

## 2. PCA

### A. Cơ sở lý thuyết

Phép phân tích thành phần chính (Principal Components Analysis - PCA) là một thuật toán thống kê sử dụng phép biến đổi trực giao để biến đổi một tập hợp dữ liệu từ một không gian nhiều chiều sang một không gian mới ít chiều hơn (2 hoặc 3 chiều) nhằm tối ưu hóa việc thể hiện sự biến thiên của dữ liệu.

### B. Bảng hàm

STT	Tên hàm	Mục đích của hàm	File lưu trữ
1	data	Cung cấp tập dữ	PCA.pyx

	<p>Input: k nếu k=1 là dataset cho pca, k=2 là tập iris</p> <p>Output: Cung cấp tập dữ liệu.</p>	liệu cho tính toán có 2 tập là tập dataset ban đầu từ DAO đưa lên và tập hoa iris.	
2	<p>label_</p> <p>Input: k nếu k=1 là label cho pca, k=2 là label tập iris</p> <p>Output: Cung cấp label.</p>	Cung cấp label cho tính toán có 2 tập là tập label ban đầu từ DAO đưa lên và tập hoa label iris.	PCA.pyx
3	<p>Standardizing</p> <p>Input: k nếu k=1 là dataset cho pca, k=2 là tập iris</p> <p>Output: Cung cấp tập dữ liệu đã chuẩn hóa .</p>	Chuẩn hóa dữ liệu trước khi tính toán.	PCA.pyx
4	<p>Mean</p> <p>Input: tập dữ liệu.</p> <p>Output: Kỳ vọng trung bình.</p>	Tính kỳ vọng trung bình ứng với tập dữ liệu tương ứng.	PCA.pyx
5	<p>CovarianceMatrix</p> <p>Input: Mean, tập dữ liệu</p> <p>Output: CovarianceMatrix</p>	Tính Covariance Matrix tương ứng.	PCA.pyx
6	<p>Eigen_Values</p> <p>Input: k tập dữ liệu tương ứng.</p> <p>Output: Eigen_Values</p>	Tính Eigen Values tương ứng.	PCA.pyx
7	<p>Eigen_Vectors</p> <p>Input: k tập dữ liệu tương ứng.</p>	Tính Eigen_Vectors tương ứng.	PCA.pyx

	Output: Eigen_Vectors		
8	Selecting_Pri_Components Input: k tập dữ liệu tương ứng. Output: Không.	Vẽ số PC mới và tỉ lệ dữ liệu mà PC giữ lại sau biến để chọn số PC phù hợp.	PCA.pyx
9	new_Data Input: n_pc(số pc), k(k tập) Output: Tập dữ liệu sau biến đổi PCA.	Cung cấp tập dữ liệu mới sau khi biến đổi PCA.	PCA.pyx
10	KNN Input: n_pc(số pc), k(k tập) Output: kết quả đánh giá tập dữ liệu mới.	Đánh giá tập dữ liệu mới và cũ qua thuật toán KNN.	PCA.pyx
11	Draw_2d Input: k(tập dữ liệu), title Output: Không.	Cung cấp hàm vẽ 2 chiều cho PCA.	PCA.pyx
12	Draw_3d Input: k(tập dữ liệu), title Output: Không.	Cung cấp hàm vẽ 3 chiều cho PCA.	PCA.pyx

### C. Mã nguồn

- Các thư viện và trở liên kết tới DAO

```
import sys
```

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

```
# chia tập dữ liệu ban đầu thành 2 tập là training và testing
```

```
from sklearn.model_selection import train_test_split
```

```

import DataProcessing as dp
import plotly.plotly as py
from sklearn.preprocessing import StandardScaler
import numpy as np
# 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
# chia tap du lieu ban dau thanh 2 tap la training va testing
from sklearn.model_selection import train_test_split
from sklearn import datasets
iris = datasets.load_iris()

```

- Hàm data cung cấp dataset với k=1 là dataset gốc k=2 tập dataset iris

```

def data(k):
    if k==1:
        return dp.data(3)
    elif k==2:
        return iris.data

```

- Hàm label\_ cung cấp label với k=1 là label dataset gốc k=2 tập label iris

```

def label_(k):
    if k==1:
        return dp.label_(3)
    elif k==2:
        return iris.target

```

- Hàm Standardizing chuẩn hóa dữ liệu

```
def Standardizing(k):
```

```
    #chuan hoa du lieu
```

```
    dt = StandardScaler().fit_transform(data(k))
```

```
    return dt
```

- Hàm Mean tính kì vọng trung bình

```
def Mean(dt):
```

```
    #tinh ki vong trung binh
```

```
    print(dt[0])
```

```
    mean_vec = np.mean(dt, axis=0)
```

```
    print(mean_vec[0])
```

```
    return mean_vec
```

- Hàm CovarianceMatrix tính Covariance Matrix

```
def CovarianceMatrix(dt,mean_vec):
```

```
    #tinh matrix hiep phuong sai
```

```
    # $\sigma_{jk} = \frac{1}{n-1} \sum_{i=1}^n (x_{ij} - \bar{x}_j)(x_{ik} - \bar{x}_k)$ 
```

```
    cov_mat = (dt - mean_vec).T.dot((dt - mean_vec)) /
```

```
    (dt.shape[0]-1)
```

```
    return cov_mat
```

- Hàm Eigen\_Values tính Eigen Values

```
def Eigen_Values(k):
```

```
    dt=Standardizing(k)
```

```
    mean_vec=Mean(dt)
```

```
    cov_mat=CovarianceMatrix(dt,mean_vec)
```

```
    eig_vals, eig_vecs = np.linalg.eig(cov_mat)
```

```
    return eig_vals
```

- Hàm Eigen\_Vectors tính Eigen Vectors

```
def Eigen_Vectors(k):
```

```
    dt=Standardizing(k)
```

```
    mean_vec=Mean(dt)
```

```

cov_mat=CovarianceMatrix(dt,mean_vec)
eig_vals, eig_vecs = np.linalg.eig(cov_mat)
return eig_vecs

```

- Hàm `Selecting_Pri_Components` để chọn số PC mới

```

def Selecting_Pri_Components(k):
    if k==2:
        return None
    eig_vals=Eigen_Values(k)
    eig_vecs=Eigen_Vectors(k)
    eig = [(np.abs(eig_vals[i]), eig_vecs[:,i]) for i in
range(len(eig_vals))]
    # xap xep eigenvalue, eigenvector tuples tu cao den thap
    eig.sort()
    eig.reverse()
    ##danh gia du lieu bi mat
    s = sum(eig_vals)
    var = [(i / s)*100 for i in sorted(eig_vals, reverse=True)]
    csvar = np.cumsum(var)
    ##ve
    y=[(i+50,csvar[i])for i in range(11)]
    y=np.array(y)
    plt.scatter(y[:,0],y[:,1],marker="*",c="red")
    for i in range(11):
        plt.text(float(y[i,0])+0.3,float(y[i,1])-0.3,"PC%s"%i,fontsize=15)
    plt.plot(y[:,0],y[:,1])
    plt.ylabel("Độ chính xác(%)")
    plt.title("Biểu đồ thể hiện lượng thông tin của các PC")
    plt.show()

```

- Hàm `new_Data` trả về bộ dữ liệu mới đã qua PCA

```

def new_Data(n_pc,k):
    dt=Standardizing(k)
    eig_vals=Eigen_Values(k)
    eig_vecs=Eigen_Vectors(k)
    eig = [(np.abs(eig_vals[i]), eig_vecs[:,i]) for i in
range(len(eig_vals))]
    # xap xep eigenvalue, eigenvector tuples tu cao den thap
    eig.sort()
    eig.reverse()
    ##danh gia du lieu bi mat
    s = sum(eig_vals)
    var = [(i / s)*100 for i in sorted(eig_vals, reverse=True)]
    csvar = np.cumsum(var)
    tmp=[]
    for i in range(n_pc):
        tmp.append(eig[i][1])
    tmp=np.array(tmp)
    #Y=X*W
    Y = dt.dot(tmp.T)
    return Y

```

- Hàm KNN so sánh giữa tập dữ liệu cũ và mới qua thuật toán KNN

```

def KNN(n_pc,k):
    x_train, x_test, y_train, y_test=train_test_split(
data(k),label_(k),test_size=0.2)
    dt=new_Data(n_pc,k)
    if k==2:
        return None
    x_train_PCA, x_test_PCA, y_train_PCA,
y_test_PCA=train_test_split(

```

```

dt,label_(k),test_size=0.2)
clf=KNeighborsClassifier(n_neighbors=13).fit(x_train,y_train)
precision= precision_score(y_test,clf.predict(x_test))
recall= recall_score(y_test,clf.predict(x_test))
F=(2*precision*recall)/(precision+recall)

```

```

clf1=KNeighborsClassifier(n_neighbors=13).fit(x_train_PCA,y_train_PCA)

```

```

precision1= precision_score(y_test_PCA,clf1.predict(x_test_PCA))
recall1= recall_score(y_test_PCA,clf1.predict(x_test_PCA))
F1=(2*precision1*recall1)/(precision1+recall1)
return [F,F1]

```

- Hàm Draw\_2d biểu diễn PCA lên không gian 2 chiều

```

def Draw_2d(k,title):
    dt=new_Data(2,k)
    dt=np.array(dt)
    plt.scatter(dt[:,0],dt[:,1],c=label_(k))
    plt.ylabel("Y")
    plt.xlabel("x")
    plt.title(title)
    plt.show()

```

- Hàm Draw\_3d biểu diễn PCA lên không gian 3 chiều

```

def Draw_3d(k,title):
    dt=new_Data(3,k)
    fig = plt.figure()
    dt=np.array(dt)
    ax = fig.add_subplot(111, projection='3d')
    ax.scatter(dt[:,0],dt[:,1],dt[:,2],c=label_(k),s=100)
    ax.set_xlabel("X")

```



```

ax.set_ylabel("Y")
ax.set_zlabel("Z")
plt.title(title)
plt.show()

```

### 3. LDA

#### A. Cơ sở lý thuyết

Là phép phân tích sự khác biệt tuyến tính được sử dụng cho bài toán phân loại phân lớp trên tập dữ liệu k chiều.

#### B. Bảng hàm

STT	Tên hàm	Mục đích của hàm	File lưu trữ
1	data Input: k nếu k=0 là dataset cho class 0, k=1 là dataset cho class 1, k=2 là dataset tổng , Output: Cung cấp tập dữ liệu.	Cung cấp dataset tương ứng từng class.	LDA.pyx
2	Standardizing Input: k Output: tập dữ liệu thứ k được chuẩn hóa.	Chuẩn hóa dữ liệu thứ k tương ứng.	LDA.pyx
3	Mean Input: Tập dữ liệu dt. Output: mean vector	Tính Mean tương ứng với mỗi class.	LDA.pyx

4	Sw Input: Không. Output: Sw.	Tính Sw tương ứng.	LDA.pyx
5	Sb Input: Không. Output: Sb.	Tính Sb tương ứng.	LDA.pyx
6	LDA_matrix Input: Không. Output: Hàm mục tiêu.	Tính hàm mục tiêu tương ứng.	LDA.pyx
7	Eigen_Values Input: Không. Output: Eigen_Values	Tính Eigen Values.	LDA.pyx
8	Eigen_Vectors Input: Không. Output: Tính Eigen_Vectors	Tính Eigen Vectors	LDA.pyx
9	new_Data Input: n_pc Output: Dữ liệu mới qua LDA	Dữ liệu mới qua LDA.	LDA.pyx
10	Selecting_Pri_Components Input: Không. Output: Không.	Chọn số LD phù hợp.	LDA.pyx
11	KNN Input: n_pc Output: so sánh dữ liệu mới và cũ.	So sánh dữ liệu mới và cũ qua KNN.	LDA.pyx
12	Draw_2d Input: title Output: Không.	Biểu diễn LDA 2 chiều.	LDA.pyx
13	Draw_3d	Biểu diễn LDA 3	LDA.pyx

	Input: title Output: Không.	chiều.	
--	--------------------------------	--------	--

### C. Mã nguồn

- Các thư viện và trở liên kết tới DAO

```
import sys
sys.path.append('../DAO/')
# chia tap du lieu ban dau thanh 2 tap la training va testing
from sklearn.model_selection import train_test_split
import DataProcessing as dp
import plotly.plotly as py
from sklearn.preprocessing import StandardScaler
import numpy as np
# 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
# chia tap du lieu ban dau thanh 2 tap la training va testing
from sklearn.model_selection import train_test_split
from sklearn import datasets
iris = datasets.load_iris()
# k y nghĩa là chọn dataset cho tập iris hoặc dataset gốc
dataset = dp.data(3)
label_=dp.label_(3)
```

- Hàm data cung cấp dataset tương ứng từng class

```
def data(k):
    tmp=[]
    if k==0:
        for i in range(len(dataset)):
            if label_[i]==0:
                tmp.append(dataset[i])
    elif k==1:
        for i in range(len(dataset)):
            if label_[i]==1:
                tmp.append(dataset[i])
    elif k==2:
        tmp=dataset
    return tmp
```

- Hàm Standardizing chuẩn hóa dữ liệu tương ứng tập thứ k

```
def Standardizing(k):
    #chuan hoa du lieu
    dt = StandardScaler().fit_transform(data(k))
    return dt
```

- Hàm Mean tính mean cho từng class

```
def Mean(dt):
    #tinh ki vong trung binh tuong ung voi tung class
    mean_vec = np.mean(dt, axis=0)
    return mean_vec
```

- Hàm Sw tính Sw

```
def Sw():
    
$$\# \sum_{i=1}^N \frac{1}{N_i - 1} \sum_{xx \in D_i} (xx - m_i)(xx - m_i)^T.$$

    dt0=Standardizing(0)
    dt1=Standardizing(1)
```

```

mean_vec0=Mean(dt0)
mean_vec1=Mean(dt1)
a = (dt0 - mean_vec0).T.dot((dt0 - mean_vec0))
b= (dt1 - mean_vec1).T.dot((dt1 - mean_vec1))
return a+b

```

- Hàm Sb tính Sb

```

def Sb():
    dt=Standardizing(2)
    mean=Mean(dt)
    dt0=Standardizing(0)
    dt1=Standardizing(1)
    mean_vec0=Mean(dt0)
    mean_vec1=Mean(dt1)
    a = (mean_vec0-mean).dot((mean_vec0-mean).T)*(dt0.shape[0])
    b = (mean_vec1-mean).dot((mean_vec1-mean).T)*(dt1.shape[0])
    return a+b

```

- Hàm LDA\_matrix tính hàm mục tiêu

```

def LDA_matrix():
    m=np.linalg.inv(Sw()).dot(Sb())
    return m

```

- Hàm Eigen\_Values tính Eigen Values

```

def Eigen_Values():
    LDA_mat=LDA_matrix()
    eig_vals, eig_vecs = np.linalg.eig(LDA_mat)
    return eig_vals

```

- Hàm Eigen\_Vectors tính Eigen Vectors

```

def Eigen_Vectors():
    LDA_mat=LDA_matrix()
    eig_vals, eig_vecs =np.linalg.eig(LDA_mat)

```

```
return eig_vecs
```

- Hàm new\_Data trả về dữ liệu mới qua biến đổi LDA

```
def new_Data(n_pc):  
    eig_vals=Eigen_Values()  
    eig_vecs=Eigen_Vectors()  
    eig = [(np.abs(eig_vals[i]), eig_vecs[:,i]) for i in  
range(len(eig_vals))]  
    # xap xep eigenvalue, eigenvector tuples tu cao den thap  
    eig.sort()  
    eig.reverse()  
    tmp=[]  
    for i in range(n_pc):  
        tmp.append(eig[i][1])  
    tmp=np.array(tmp)  
    Y = Standardizing(2).dot(tmp.T)  
    return Y
```

- Hàm Selecting\_Pri\_Components so sánh tỷ lệ biến đổi dữ liệu mới và cũ

```
def Selecting_Pri_Components():  
    eig_vals=Eigen_Values()  
    eig_vecs=Eigen_Vectors()  
    eig = [(np.abs(eig_vals[i]), eig_vecs[:,i]) for i in  
range(len(eig_vals))]  
    # xap xep eigenvalue, eigenvector tuples tu cao den thap  
    eig.sort()  
    eig.reverse()  
    ##danh gia du lieu bi mat  
    s = sum(eig_vals)  
    var = [(i / s)*100 for i in sorted(eig_vals, reverse=True)]  
    csvar = np.cumsum(var)
```

```

##ve
y=[(i+50,csvar[i])for i in range(11)]
y=np.array(y)
plt.scatter(y[:,0],y[:,1],marker="o",c="#ff0066")
plt.plot(y[:,0],y[:,1],c="#3366cc")
plt.ylabel("Độ chính xác(%)")
plt.title("Biểu đồ thể hiện lượng thông tin của các LD")
plt.show()

```

- Hàm KNN đánh giá dữ liệu mới, cũ qua KNN

```

def KNN(n_pc):
    x_train, x_test, y_train, y_test=train_test_split(
        Standardizing(2),label_,test_size=0.2)
    dt=new_Data(n_pc)
    x_train_LDA, x_test_LDA, y_train_LDA,
y_test_LDA=train_test_split(
    dt,label_,test_size=0.2)
    clf=KNeighborsClassifier(n_neighbors=13).fit(x_train,y_train)
    precision= precision_score(y_test,clf.predict(x_test))
    recall= recall_score(y_test,clf.predict(x_test))
    F=(2*precision*recall)/(precision+recall)

    clf1=KNeighborsClassifier(n_neighbors=13).fit(x_train_LDA,y_train_LD
A)
    precision1=
precision_score(y_test_LDA,clf1.predict(x_test_LDA))
    recall1= recall_score(y_test_LDA,clf1.predict(x_test_LDA))
    F1=(2*precision1*recall1)/(precision1+recall1)
    return [F,F1]

```

- Hàm Draw\_2d biểu diễn LDA 2 chiều

```
def Draw_2d(title):
    dt=new_Data(2)
    dt=np.array(dt)
    plt.scatter(dt[:,0],dt[:,1],c=label_)
    plt.ylabel("Y")
    plt.xlabel("x")
    plt.title(title)
    plt.show()
```

- Hàm Draw\_3d biểu diễn LDA 3 chiều

```
def Draw_3d(title):
    dt=new_Data(3)
    fig = plt.figure()
    dt=np.array(dt)
    ax = fig.add_subplot(111, projection='3d')
    ax.scatter(dt[:,0],dt[:,1],dt[:,2],c=label_,s=100)
    ax.set_xlabel("X")
    ax.set_ylabel("Y")
    ax.set_zlabel("Z")
    plt.title(title)
    plt.show()
```

## 4. Kernel PCA

### A. Cơ sở lý thuyết

Là phần mở rộng của PCA dùng các kỹ thuật của phương pháp Kernel

### B. Bảng hàm

STT	Tên hàm	Mục đích của hàm	File lưu trữ
-----	---------	------------------	--------------



1	<p>data</p> <p>Input: k nếu k=1 là dataset cho pca, k=2 là tập iris</p> <p>Output: Cung cấp tập dữ liệu.</p>	Cung cấp tập dữ liệu cho tính toán có 2 tập là tập dataset ban đầu từ DAO đưa lên và tập hoa iris.	KPCA.pyx
2	<p>stepwise_kpca</p> <p>Input: gamma=0.2</p> <p>,</p> <p>n_components=2 và tập dataset gốc</p>	để tính toán từng bước thuật toán	KPCA.pyx
3	<p>label_</p> <p>Input: k nếu k=1 là label cho pca, k=2 là label tập iris</p> <p>Output: Cung cấp label.</p>	Cung cấp label cho tính toán có 2 tập là tập label ban đầu từ DAO đưa lên và tập hoa label iris.	KPCA.pyx
4	<p>Standardizing</p> <p>Input: k nếu k=1 là dataset cho pca, k=2 là tập iris</p> <p>Output: Cung cấp tập dữ liệu đã chuẩn hóa .</p>	Chuẩn hóa dữ liệu trước khi tính toán.	KPCA.pyx
5	<p>Eigen_Vectors</p> <p>Input: tập dữ liệu tương ứng, gamma=0.2, n_components=2.</p> <p>Output: Eigen_Vectors</p>	Tính Eigen_Vectors tương ứng.	KPCA.pyx
6	<p>Eigen_Values</p> <p>Input: tập dữ liệu tương</p>	Tính Eigen_Values tương ứng.	KPCA.pyx

	ứng, gamma=0.2, n_components=2. Output: Eigen_Vectors		
7	Selecting_Pri_Co mponents Input: k tập dữ liệu tương ứng. Output: Không.	Vẽ số PC mới và tỉ lệ dữ liệu mà PC giữ lại sau biến để chọn số PC phù hợp.	KPCA.pyx
8	KNN Input: n_pc(số pc)=2, gamma=0.2, k(k tập) Output: kết quả đánh giá tập dữ liệu mới.	Đánh giá tập dữ liệu mới và cũ qua thuật toán KNN.	KPCA.pyx
9	Draw_2d Input: k(tập dữ liệu), title Output: Không.	Cung cấp hàm vẽ 2 chiều cho PCA.	KPCA.pyx

### C. Mã nguồn

- Thư viện

```
#dataset(3) -> dataset_for_PCA_LDA
import sys
sys.path.append('../DAO/')
# chia tập dữ liệu ban đầu thành 2 tập là training và testing
from sklearn.model_selection import train_test_split
import DataProcessing as dp
import plotly.plotly as py
from sklearn.preprocessing import StandardScaler
import numpy as np
# thư viện vẽ của 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
# chia tap du lieu ban dau thanh 2 tap la training va testing
from sklearn.model_selection import train_test_split
from sklearn import datasets
iris = datasets.load_iris()
from scipy.spatial.distance import pdist, squareform
from scipy import exp
from scipy.linalg import eigh
import numpy as np
```

- Chọn dataset

```
def data(k):
    if k==1:
        return dp.data(0)
    elif k==2:
        return iris.data
def label_(k):
    if k==1:
        return dp.label_(0)
    elif k==2:
        return iris.target
def Standardizing(k):
    #chuan hoa du lieu
    dt = StandardScaler().fit_transform(data(k))
    return dt
```

- **Chạy thuật toán RBF kernel PCA**

```
def stepwise_kpca(X, gamma, n_components):
    """
```

*Implementation of a RBF kernel PCA.*

*Arguments:*

*X: A MxN dataset as NumPy array where the samples are stored as rows (M),*

*and the attributes defined as columns (N).*

*gamma: A free parameter (coefficient) for the RBF kernel.*

*n\_components: The number of components to be returned.*

```
    """
```

```

# Calculating the squared Euclidean distances for every pair of
points
# in the MxN dimensional dataset.
sq_dists = pdist(Standardizing(X), 'sqeuclidean')

# Converting the pairwise distances into a symmetric MxM matrix.
mat_sq_dists = squareform(sq_dists)

# Computing the MxM kernel matrix.
K = exp(-gamma * mat_sq_dists)

# Centering the symmetric NxN kernel matrix.
N = K.shape[0]
one_n = np.ones((N,N)) / N
K = K - one_n.dot(K) - K.dot(one_n) + one_n.dot(K).dot(one_n)

# Obtaining eigenvalues in descending order with corresponding
# eigenvectors from the symmetric matrix.
eigvals, eigvecs = eigh(K)

# Obtaining the i eigenvectors that corresponds to the i highest
eigenvalues.
X_pc = np.column_stack((eigvecs[:, -i] for i in
range(1, n_components+1)))
return X_pc

```

- **Hàm giá trị**

```

def Eigen_Values(X, gamma):
    sq_dists = pdist(Standardizing(X), 'sqeuclidean')

    # Converting the pairwise distances into a symmetric MxM matrix.
    mat_sq_dists = squareform(sq_dists)

    # Computing the MxM kernel matrix.
    K = exp(-gamma * mat_sq_dists)

    # Centering the symmetric NxN kernel matrix.
    N = K.shape[0]
    one_n = np.ones((N, N)) / N
    K = K - one_n.dot(K) - K.dot(one_n) + one_n.dot(K).dot(one_n)

    # Obtaining eigenvalues in descending order with corresponding
    # eigenvectors from the symmetric matrix.

```

```

    eigvals, eigvecs = eigh(K)
    return eigvals

def Eigen_Vectors(X, gamma):
    sq_dists = pdist(Standardizing(X), 'sqeuclidean')

    # Converting the pairwise distances into a symmetric MxM matrix.
    mat_sq_dists = squareform(sq_dists)

    # Computing the MxM kernel matrix.
    K = exp(-gamma * mat_sq_dists)

    # Centering the symmetric NxN kernel matrix.
    N = K.shape[0]
    one_n = np.ones((N, N)) / N
    K = K - one_n.dot(K) - K.dot(one_n) + one_n.dot(K).dot(one_n)

    # Obtaining eigenvalues in descending order with corresponding
    # eigenvectors from the symmetric matrix.
    eigvals, eigvecs = eigh(K)
    return eigvecs

def Selecting_Pri_Components(k):
    if k==2:
        return None
    eig_vals=Eigen_Values(k,0.1)
    eig_vecs=Eigen_Vectors(k,0.1)
    eig = [(np.abs(eig_vals[i]), eig_vecs[:,i]) for i in
range(len(eig_vals))]
    # xap xep eigenvalue, eigenvector tuples tu cao den thap
    eig.sort()
    eig.reverse()
    ##danh gia du lieu bi mat
    s = sum(eig_vals)
    var = [(i / s)*100 for i in sorted(eig_vals, reverse=True)]
    csvar = np.cumsum(var)
    ##ve
    y=[(i+50,csvar[i])for i in range(11)]
    y=np.array(y)
    plt.scatter(y[:,0],y[:,1],marker="*",c="red")
    for i in range(11):
        plt.text(float(y[i,0])+0.3,float(y[i,1])-0.3,"PC%s"%i,fontsize=15)
    plt.plot(y[:,0],y[:,1])
    plt.ylabel("Độ chính xác(%)")

```

```
plt.title("Biểu đồ thể hiện lượng thông tin của các PC")
plt.show()
```

- **Hàm so sánh độ chính xác giữa 2 tập dataset bằng thuật toán KNN**

```
def KNN(gamma,k,n_comp):
    x_train, x_test, y_train, y_test=train_test_split(
        data(k),label_(k),test_size=0.2)
    dt=stepwise_kpca(k,gamma,n_comp)
    if k==2:
        return None
    x_train_PCA, x_test_PCA, y_train_PCA,
    y_test_PCA=train_test_split(
        dt,label_(k),test_size=0.2)
    clf=KNeighborsClassifier(n_neighbors=13).fit(x_train,y_train)
    precision= precision_score(y_test,clf.predict(x_test))
    recall= recall_score(y_test,clf.predict(x_test))
    F=(2*precision*recall)/(precision+recall)
    clf1=KNeighborsClassifier(n_neighbors=13).fit(x_train_PCA,y_train_PCA)
    precision1= precision_score(y_test_PCA,clf1.predict(x_test_PCA))
    recall1= recall_score(y_test_PCA,clf1.predict(x_test_PCA))
    F1=(2*precision1*recall1)/(precision1+recall1)
    return [F,F1]
```

- **Hàm vẽ**

```
def Draw_2d(k,title,gamma):
    X_pc = stepwise_kpca(k, gamma=gamma, n_components=2)

    #convert string
    x=str(gamma)

    #Plot drawing
    plt.figure(figsize=(8, 6))
    plt.scatter(X_pc[:, 0], X_pc[:, 1], c=label_(k))
    y = label_(k)
    plt.ylabel("PC2")
    plt.xlabel("PC1")
    plt.title(title+' gamma='+x )
    plt.show()
```

- **Chạy**

```
print('Eigen_Values:', Eigen_Values(1,0.2))
print('Eigen_Vectors:', Eigen_Vectors(1,0.2))
```

```

Selecting_Pri_Components(1)
print(KNN(0.2,1,2))
Draw_2d(1,'Gaussian RBF kernel PCA',0.2)

```

## 5. Polynomial Regression

### a. Cơ sở lý thuyết

Polynomial là một dạng của hồi quy tuyến tính dự đoán biến phụ thuộc vào Y dựa trên biến độc lập X xây dựng model với n bậc tuyến tính trong biến độc lập X

### b. Bảng hàm

STT	Tên hàm	Mục đích của hàm	File lưu trữ
1	PLR Input: tập được chia thành training và testing để chạy thuật toán  Output: Kết quả chạy thuật toán.	training model	Polynomial_Regression.pyx
2	vitriDacTrung Input: Tên đặc trưng. Output: Vị trí đặc trưng trong dataset.	Là hàm trả về vị trí đặc trưng trong dataset cung cấp cho hàm layviTriDacTrung	Polynomial_Regression.pyx

3	<p>tenDacTrung</p> <p>Input: Vị trí đặc trung.</p> <p>Output: Tên đặc trung.</p>	Là hàm trả về tên đặc trung.	Polynomial_Regression.pyx
4	<p>layviTriDacTrung</p> <p>Input: Mảng các đặc trung.</p> <p>Output: Mảng vị trí đặc trung tương ứng.</p>	Hàm trả về mảng vị trí đặc trung từ mảng tên đặc trung.	Polynomial_Regression.pyx
5	<p>layDacTrung</p> <p>Input: Mảng các đặc trung.</p> <p>Output: Mảng các cột đặc trung tương ứng trong dataset</p>	Trả về 1 DataFrame tương ứng với các đặc trung.	Polynomial_Regression.pyx
6	<p>ve2D</p> <p>Input: Mảng các đặc trung và đặc trung vẽ.</p> <p>Output: vẽ 2d.</p>	Biểu diễn trên không gian 2 chiều.	Polynomial_Regression.pyx
7	<p>tinhToan</p> <p>Input: Mảng các đặc trung.</p> <p>Output: kết quả tính MR.</p>	Hàm tính toán kết quả sử dụng hàm MR.	Polynomial_Regression.pyx



### c. Mã nguồn

- Thư viện

```
#dataset(0) -> dataset_for_Kernel_PCA
#dataset(1) -> dataset_for_Logistic_regression
#dataset(2) -> dataset_for_MultiLinear_regression
#dataset(3) -> dataset_for_PCA_LDA
#dataset(4) -> dataset_for_Poly_regression
import sys
sys.path.append('../DAO/')
# chia tap du lieu ban dau thanh 2 tap la training va testing
from sklearn.model_selection import train_test_split
import DataProcessing as dp
import operator
# thu vien thuan toan LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
# danh gia
from sklearn.metrics import r2_score
#chon dac trung dung cay quyet dinh
from sklearn import tree
#chuyen sang anh
import pydotplus
#xu ly matrix
import numpy as np
# thu vien ve cua python
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.pipeline import make_pipeline
from mpl_toolkits.mplot3d import Axes3D
```

- Chọn dataset

```
data=dp.data(4)
label_=dp.label_(4)
```

- Chạy thuật toán

```
def PLR(x_train,y_train,x_test,y_test):
    poly_reg = PolynomialFeatures(degree=2)
    X_poly = poly_reg.fit_transform(x_train)
    lin_reg_2 = LinearRegression()
```

```

lin_reg_2.fit(X_poly, y_train)
X_poly1 = poly_reg.fit_transform(x_test)
y_pred = lin_reg_2.predict(X_poly1)
score = r2_score(y_test, y_pred)
k=[]
k.append([score,y_pred])
return np.array(k)

```

```

def vitriDacTrung(dacTrung):
    if dacTrung=='Rating':
        return 0
    elif dacTrung=='Reviews':
        return 1
    elif dacTrung=='Size':
        return 2

```

```

def tenDacTrung(viTri):
    switcher={
        0:'Rating',
        1:'Reviews',
        2:'Size'
    }
    return switcher.get(viTri)

```

```

def layviTriDacTrung(mangCacDacTrung):
    m=[]
    for i in mangCacDacTrung:
        m.append(vitriDacTrung(i))
    return m

```

```

def layDacTrung(mangCacDacTrung):
    return pd.DataFrame(data)[layviTriDacTrung(mangCacDacTrung)]

```

- Hàm vẽ

```

def ve2D(mangCacDacTrung,dacTrungVe):
    x_train,x_test,y_train,y_test=train_test_split(layDacTrung(mangCacDacTrung),label_,test_size=0.2)
    m1=PLR(x_train[layviTriDacTrung(dacTrungVe)],y_train,x_test[layviTriDacTrung(dacTrungVe)],y_test)
    y_pred=[]
    for i in range(len((m1[:,1])[0])):
        y_pred.append((m1[:,1])[0][i])

```

```

# pr = LinearRegression()
# quadratic = PolynomialFeatures(degree=2)
# X_quad =
quadratic.fit_transform(x_train[layviTriDacTrung(dacTrungVe)])
# pr.fit(X_quad, y_train)
# y_quad_fit =
pr.predict(quadratic.fit_transform(x_test[layviTriDacTrung(dacTrungVe)]))

plt.scatter(x_train[layviTriDacTrung(dacTrungVe)],y_train,s=10,c='blue')
plt.plot(x_test[layviTriDacTrung(dacTrungVe)],y_pred,color='red')
plt.title('Polynomial Regression')
plt.xlabel("%s(Xj)"%dacTrungVe)
plt.ylabel('Y')

plt.legend(loc='upper left')
plt.tight_layout()
plt.show()

```

- Hàm tính toán

```

def tinhToan(mangCacDacTrung):
    x_train, x_test, y_train,
    y_test=train_test_split(layDacTrung(mangCacDacTrung),label_test_size=0.2)
    return float(PLR(x_train,y_train,x_test,y_test)[:,0])*100

```

- Chạy thuật toán

```

ve2D(['Rating','Reviews','Size'],['Rating'])
print(tinhToan(['Rating','Reviews','Size']))

```

## 6. Logistic Regression

### A. Cơ sở lý thuyết

Phương pháp hồi quy logistic là một mô hình hồi quy nhằm dự đoán giá trị đầu ra rời rạc (*discrete target variable*)  $y$  ứng với một véc-tơ đầu vào  $\mathbf{x}$ . Việc này tương đương với chuyển phân loại các đầu vào  $\mathbf{x}$  vào các nhóm  $y$  tương ứng.

### B. Bảng hàm

STT	Tên hàm	Mục đích của hàm	File lưu trữ
1	logistic Input: Tập dữ liệu đã chia training và testing. Output: Kết quả qua thuật toán MR.	Là hàm tính của thuật toán logistic_regression.	Logistic_Regression.py
2	vitriDacTrung Input: Tên đặc trưng. Output: Vị trí đặc trưng trong dataset.	Là hàm trả về vị trí đặc trưng trong dataset cung cấp cho hàm layviTriDacTrung	Logistic_Regression.py
3	tenDacTrung Input: Vị trí đặc trưng. Output: Tên đặc trưng.	Là hàm trả về tên đặc trưng.	Logistic_Regression.py
4	layviTriDacTrung Input: Mảng các đặc trưng. Output: Mảng vị trí đặc trưng tương ứng.	Hàm trả về mảng vị trí đặc trưng từ mảng tên đặc trưng.	Logistic_Regression.py
5	layDacTrung	Trả về 1 DataFrame	Logistic_Regression.py

	Input: Mảng các đặc trung. Output: Mảng các cột đặc trưng tương ứng trong dataset	tương ứng với các đặc trung.	y
6	ve2D Input: Mảng các đặc trung và đặc trưng vẽ. Output: vẽ 2d.	Vẽ trên không gian 2 chiều.	Logistic_Regression.p y
7	ve3D Input: Mảng các đặc trung và đặc trưng vẽ. Output: vẽ 3d.	Vẽ trên không gian 3 chiều.	Logistic_Regression.p y
8	tinhToan Input: Mảng các đặc trung. Output: kết quả tính MR.	Hàm tính toán kết quả sử dụng hàm logistic.	Logistic_Regression.p y

### ***C. Mã nguồn***

- Các thư viện và trở liên kết tới DAO

```

#dataset(0) -> dataset_for_Kernel_PCA
#dataset(1) -> dataset_for_Logistic_regression
#dataset(2) -> dataset_for_MultiLinear_regression
#dataset(3) -> dataset_for_PCA_LDA
#dataset(4) -> dataset_for_Poly_regression
import sys
sys.path.append('../DAO/')
# chia tap du lieu ban dau thanh 2 tap la training va testing
from sklearn.model_selection import train_test_split
import DataProcessing as dp
from sklearn.linear_model import LogisticRegression
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import r2_score

Chọn dataset
data=dp.data(1)
label_=dp.label_(1)

```

- Hàm logistic trả về kết quả tính toán

```

def logistic(x_train,y_train,x_test,y_test):
    logic = LogisticRegression()
    logic.fit(x_train, y_train)
    y_pred = logic.predict(x_test)
    score=r2_score(y_test,y_pred)
    k=[]
    k.append([score,y_pred])
    return np.array(k)

```

- Hàm `vitriDacTrung` trả về vị trí của đặc trưng

```
def vitriDacTrung(dacTrung):  
    if dacTrung=='Sex':  
        return 0  
    elif dacTrung=='AgeRange':  
        return 1  
    elif dacTrung=='Class_':  
        return 2  
    elif dacTrung=='SiblingSpouse':  
        return 3  
    elif dacTrung=='ParentChild':  
        return 4  
    elif dacTrung=='Embarked_':  
        return 5
```

- Hàm `tenDacTrung` trả về tên của đặc trưng

```
def tenDacTrung(viTri):  
    switcher={  
        0:'Sex',  
        1:'AgeRange',  
        2:'Class_',  
        3:'SiblingSpouse',  
        4:'ParentChild',  
        5:'Embarked_',  
    }  
    return switcher.get(viTri)
```

- Hàm `layviTriDacTrung` trả về mảng các đặc trưng

```
def layviTriDacTrung(mangCacDacTrung):
```

```

m=[]
for i in mangCacDacTrung:
    m.append(vitriDacTrung(i))
return m

```

- Hàm layDacTrung trả về 1 DataFrame tương ứng với mảng đặc trưng

```

def layDacTrung(mangCacDacTrung):
    return pd.DataFrame(data)[layviTriDacTrung(mangCacDacTrung)]

```

- Hàm ve2D biểu diễn tương ứng với 1 đặc trưng

```

def ve2D(mangCacDacTrung,dacTrungVe):
    x_train, x_test, y_train,
    y_test=train_test_split(layDacTrung(mangCacDacTrung),label_,test_size=
    0.2)
    m1=logistic(x_train[layviTriDacTrung(dacTrungVe)],y_train,x_test[layvi
    TriDacTrung(dacTrungVe)],y_test)
    y_pred=[]
    for i in range(len((m1[:,1])[0])):
        y_pred.append((m1[:,1])[0][i])

```

```

plt.scatter(x_test[layviTriDacTrung(dacTrungVe)],y_test,s=10,c="#3366c
c")

```

```

plt.plot(x_test[layviTriDacTrung(dacTrungVe)],y_pred,
color='#cc0099', linewidth=3)

```

```

plt.xlabel("%s(Xj)"%dacTrungVe)

```

```

plt.ylabel('Y')

```

```

plt.show()

```

- Hàm ve3D biểu diễn tương ứng với 2 đặc trưng



```

def ve3D(mangCacDacTrung,dacTrungVe):
    x_train, x_test, y_train,
y_test=train_test_split(layDacTrung(mangCacDacTrung),label_,test_size=
0.2)
    m0=logistic(x_train,y_train,x_test,y_test)
    y_pred0=[]
    for i in range(len((m0[:,1])[0])):
y_pred0.append((m0[:,1])[0][i])
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')

ax.scatter(x_test[layviTriDacTrung([dacTrungVe[0]])],x_test[layviTriDac
Trung([dacTrungVe[1]])],y_test)

ax.plot(x_test[layviTriDacTrung([dacTrungVe[0]])],x_test[layviTriDacTr
ung([dacTrungVe[1]])], y_pred0)

plt.show()

```

- Hàm tinhToan là hàm tính toán chính cho thuật toán logistic sử dụng lại hàm logistic.

```

def tinhToan(mangCacDacTrung):
    x_train, x_test, y_train,
y_test=train_test_split(layDacTrung(mangCacDacTrung),label_,test_size
=0.2)
    return float(logistic(x_train,y_train,x_test,y_test)[:,0])*100

```

## ***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 VÀ LỚP**

Bảng 1: Hàm

<b>STT</b>	<b>Tên hàm</b>	<b>Mục đích của hàm</b>	<b>File lưu trữ</b>
1	create_widgets Input: self. Output: Không.	Là bộ khởi tạo các đối tượng đồ họa sử dụng tkinter.	App.pyx
2	__init__ Input: self. Output: Không.	Là bộ khởi tạo các đối tượng đồ họa sử dụng tkinter.	App.pyx
3	logbtn_Click Input: self. Output: Không.	Là event login cho giao diện.	App.pyx
4	back_Click Input: self. Output: Không.	Là event cho nút back.	App.pyx
5	on_select Input: self. Output: Không.	Là event cho combobox.	App.pyx
6	PCA_danhGia_bt_Click Input: self. Output: Không.	Là event cho nút đánh giá của PCA.	App.pyx
7	PCA_ve2d_bt_Click Input: self. Output: Không.	Là event cho nút ve2D của PCA.	App.pyx

8	PCA_ve3d_bt_Click Input: self. Output: Không.	Là event cho nút ve3D của PCA.	App.pyx
9	PCA_new_Data_bt_Click Input: self. Output: Không.	Là event cho nút new_Data của PCA.	App.pyx
10	PCA_Eigen_Vectors_bt_Click Input: self. Output: Không.	Là event cho nút Eigen_Vectors của PCA.	App.pyx
11	PCA_Eigen_Values_bt_Click Input: self. Output: Không.	Là event cho nút Eigen_Values của PCA.	App.pyx
12	PCA_Selecting_Pri_Components_bt_Click Input: self. Output: Không.	Là event cho nút ketQua của PCA.	App.pyx
13	PCA_Mean_bt_Click Input: self. Output: Không.	Là event cho nút Mean của PCA.	App.pyx
14	PCA_Click Input: self. Output: Không.	Là event cho nút PCA của PCA.	App.pyx
15	MR_Click Input: self. Output: Không.	Là event cho nút MR của MR.	App.pyx
16	MR_Age_Click Input: self. Output: Không.	Là event cho nút Age của MR.	App.pyx
17	MR_Weight_Click Input: self.	Là event cho nút Weight của MR.	App.pyx

	Output: Không.		
18	MR_Height_Click Input: self. Output: Không.	Là event cho nút Height của MR.	App.pyx
19	MR_Neck_Click Input: self. Output: Không.	Là event cho nút Neck của MR.	App.pyx
20	MR_Chest_Click Input: self. Output: Không.	Là event cho nút Chest của MR.	App.pyx
21	MR_Abdomen_Click Input: self. Output: Không.	Là event cho nút Abdomen của MR.	App.pyx
22	MR_Hip_Click Input: self. Output: Không.	Là event cho nút Hip của MR.	App.pyx
23	MR_Thigh_Click Input: self. Output: Không.	Là event cho nút Thigh của MR.	App.pyx
24	MR_Knee_Click Input: self. Output: Không.	Là event cho nút Knee của MR.	App.pyx
25	MR_Ankle_Click Input: self. Output: Không.	Là event cho nút Ankle của MR.	App.pyx
26	MR_Biceps_Click Input: self. Output: Không.	Là event cho nút Biceps của MR.	App.pyx
27	MR_Forearm_Click Input: self. Output: Không.	Là event cho nút Forearm của MR.	App.pyx
28	MR_Wrist_Click Input: self. Output: Không.	Là event cho nút Wrist của MR.	App.pyx

29	MR_All_Click Input: self. Output: Không.	Là event cho nút All của MR.	App.pyx
30	MR_Age2_Click Input: self. Output: Không.	Là event cho nút Age2 của MR.	App.pyx
31	MR_Weight2_Click Input: self. Output: Không.	Là event cho nút Weight2 của MR.	App.pyx
32	MR_Height2_Click Input: self. Output: Không.	Là event cho nút Height2 của MR.	App.pyx
33	MR_Neck2_Click Input: self. Output: Không.	Là event cho nút Neck2 của MR.	App.pyx
34	MR_Chest2_Click Input: self. Output: Không.	Là event cho nút Chest2 của MR.	App.pyx
35	MR_Abdomen2_Click Input: self. Output: Không.	Là event cho nút Abdomen2 của MR.	App.pyx
36	MR_Hip2_Click Input: self. Output: Không.	Là event cho nút Hip2 của MR.	App.pyx
37	MR_Thigh2_Click Input: self. Output: Không.	Là event cho nút Thigh2 của MR.	App.pyx
38	MR_Knee2_Click Input: self. Output: Không.	Là event cho nút Knee2 của MR.	App.pyx
39	MR_Ankle2_Click Input: self.	Là event cho nút Ankle2 của MR.	App.pyx

	Output: Không.		
40	MR_Biceps2_Click Input: self. Output: Không.	Là event cho nút Biceps2_ của MR.	App.pyx
41	MR_Forearm2_Click Input: self. Output: Không.	Là event cho nút Forearm2 của MR.	App.pyx
42	MR_Wrist2_Click Input: self. Output: Không.	Là event cho nút Wrist2 của MR.	App.pyx
43	MR_tinhToan_Click Input: self. Output: Không.	Là event cho nút tinhToan của MR.	App.pyx
44	MR_ve2d_Click Input: self. Output: Không.	Là event cho nút ve2d của MR.	App.pyx
45	MR_ve3d_Click Input: self. Output: Không.	Là event cho nút ve3d của MR.	App.pyx

Bảng 1: Lớp

STT	Tên lớp	Mục đích của lớp
1	Application	Tạo ra đối tượng giao diện

### 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('App.pyx'))
setup(ext_modules=cythonize('../BUS/Multivariable_Regression.pyx'))
setup(ext_modules=cythonize('../DAO/DataProcessing.pyx'))
setup(ext_modules=cythonize('../DTO/DataAdapter.pyx'))
setup(ext_modules=cythonize('../BUS/Poly_Regression.pyx'))
setup(ext_modules=cythonize('../BUS/PCA.pyx'))
setup(ext_modules=cythonize('../BUS/KPCA.pyx'))
setup(ext_modules=cythonize('../BUS/LDA.pyx'))
setup(ext_modules=cythonize('../BUS/Logistic Regression.pyx'))
```

## ***B. 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 App.o khi mới tạo ra**

```
import App
```

## ***C. File App.py***

- **Chức năng**

Mục đích để sử dụng các nghiệp vụ từ tầng BUS triển khai trên giao diện.

- **Các thư viện xử dụng và trở liên kết tới DAO**

```
import tkinter as tk
import tkinter.ttk as ttk
from tkinter import *
from tkinter import messagebox
# thu vien xu ly anh
import cv2
import PIL.Image, PIL.ImageTk
# thu vien xu ly duong dan
import sys
sys.path.append('../BUS/')
import Multivariable_Regression as MR
import PCA
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
isLogin=[]
tinhToan=[]#bien tam cho tinh toan
isAll=[]#bien tam cho nut all
ve=[]#bien tam cho ve
PCA_n_pc=[1]
```

- **Các hàm khởi tạo trong lớp đối tượng giao diện**

```
class Application(tk.Frame):
    def __init__(self, master=None):
        super().__init__(master)
        self.pack()
        self.create_widgets()
```



```

def create_widgets(self):
##### background
#####

cv_img = cv2.imread("img/b.png")
cv_img = cv2.blur(cv_img, (10,10))
self.canvas = tk.Canvas(self.master, width = 800, height = 600)
self.canvas.pack()

self.photo = PIL.ImageTk.PhotoImage(image =
PIL.Image.fromarray(cv_img))

self.canvas.create_image(0, 0, image=self.photo, anchor=tk.NW)
##### Login
#####
#####

self.label_loginForm
=tk.Label(bg="#A19A99",fg="#f2f2f2",text="Đăng Nhập")

self.label_username = tk.Label(bg="#A19A99",fg="#f2f2f2",
text="Tài khoản")

self.label_password =
tk.Label(bg="#A19A99",fg="#f2f2f2",text="Mật khẩu")

self.entry_username = Entry(self.master)
self.entry_password = Entry(self.master, show="*")
self.logbtn = Button(self.master, text="Đăng Nhập")
self.label_loginForm.config(font=("Courier bold", 40))
self.label_username.config(font=("Courier bold", 20))
self.label_password.config(font=("Courier bold", 20))
self.entry_username.config(font=("Courier bold", 20))
self.entry_password.config(font=("Courier bold", 20))
self.logbtn.config(font=("Courier bold", 20))
self.label_loginForm.pack()

```

```

self.label_username.pack()
self.label_password.pack()
self.entry_username.pack()
self.entry_password.pack()
self.logbtn.pack()
self.label_loginForm.place(x=250,y=10)
self.label_username.place(x=40,y=250)
self.label_password.place(x=40,y=300)
self.entry_username.place(x=180,y=250)
self.entry_password.place(x=180,y=300)
self.logbtn.place(x=220,y=350, width=250, height=50)
#
self.Back_bt=tk.Button(bg="#ff5c33",fg="#ffebe6")
self.Back_bt['text']='<--'
self.Back_bt.pack()
self.Back_bt.place(x=-2000+0,y=-2000+560,width=60, height=40)
##### MR
#####
##
self.MR_bt=Button(self.master, text="MultiLinear
Regression",width=20, height=2)
self.MR_bt.config(font=("Courier bold", 20))
self.MR_bt.pack()
self.MR_bt.place(x=-2000+220,y=-2000+47)
self.MR_ketQua_bt = tk.Button(bg="ffffff")
self.MR_ketQua_bt.pack()
self.MR_ketQua_bt.place(x=-2000+250, y=-2000+300, width=300,
height=100)
#####

```

```

self.MR_tinhToan_bt = tk.Button(bg="#C9DCEA")
self.MR_tinhToan_bt['text'] = 'Tính'
self.MR_tinhToan_bt.pack()
self.MR_tinhToan_bt.place(x=-2000+252, y=-2000+420,
width=90, height=50)
###
self.MR_ve2d_bt = tk.Button(bg="#C9DCEA")
self.MR_ve2d_bt['text'] = 'Vẽ 2d'
self.MR_ve2d_bt.pack()
self.MR_ve2d_bt.place(x=-2000+352, y=-2000+420, width=90,
height=50)
###
self.MR_ve3d_bt = tk.Button(bg="#C9DCEA")
self.MR_ve3d_bt['text'] = 'Vẽ 3d'
self.MR_ve3d_bt.pack()
self.MR_ve3d_bt.place(x=-2000+452, y=-2000+420, width=90,
height=50)
self.MR_label = tk.Label(text="MR",bg="#A19A99",fg="#f2f2f2")
self.MR_label.pack()
self.MR_label.place(x=-2000+260, y=-2000+5, width=280,
height=60)
self.MR_label.config(font=("Courier bold", 40))
self.MR_dactrungtinh_label = tk.Label(text="Chọn đặc trưng tính
toán",bg="#A19A99",fg="#f2f2f2")
self.MR_dactrungtinh_label.config(font=("Courier bold", 15))
self.MR_dactrungtinh_label.pack()
self.MR_dactrungtinh_label.place(x=-2000+30, y=-2000+10,
width=280, height=30)

```

```
#####

self.MR_Age_bt = tk.Button(bg="#C9DCEA")
self.MR_Age_bt['text'] = 'Age'
self.MR_Age_bt.pack()
self.MR_Age_bt.place(x=-2000+30, y=-2000+50, width=70,
height=30)

###

self.MR_Weight_bt = tk.Button(bg="#C9DCEA")
self.MR_Weight_bt['text'] = 'Weight'
self.MR_Weight_bt.pack()
self.MR_Weight_bt.place(x=-2000+100, y=-2000+50, width=70,
height=30)

###

self.MR_Height_bt = tk.Button(bg="#C9DCEA")
self.MR_Height_bt['text'] = 'Height'
self.MR_Height_bt.pack()
self.MR_Height_bt.place(x=-2000+170, y=-2000+50, width=70,
height=30)

###

self.MR_Neck_bt = tk.Button(bg="#C9DCEA")
self.MR_Neck_bt['text'] = 'Neck'
self.MR_Neck_bt.pack()
self.MR_Neck_bt.place(x=-2000+240, y=-2000+50, width=70,
height=30)

#####

self.MR_Chest_bt = tk.Button(bg="#C9DCEA")
self.MR_Chest_bt['text'] = 'Chest'
self.MR_Chest_bt.pack()
```

```

self.MR_Chest_bt.place(x=-2000+30, y=-2000+80, width=70,
height=30)

####

self.MR_Abdomen_bt = tk.Button(bg="#C9DCEA")
self.MR_Abdomen_bt['text'] = 'Abdomen'
self.MR_Abdomen_bt.pack()
self.MR_Abdomen_bt.place(x=-2000+100, y=-2000+80, width=70,
height=30)

####

self.MR_Hip_bt = tk.Button(bg="#C9DCEA")
self.MR_Hip_bt['text'] = 'Hip'
self.MR_Hip_bt.pack()
self.MR_Hip_bt.place(x=-2000+170, y=-2000+80, width=70,
height=30)

####

self.MR_Thigh_bt = tk.Button(bg="#C9DCEA")
self.MR_Thigh_bt['text'] = 'Thigh'
self.MR_Thigh_bt.pack()
self.MR_Thigh_bt.place(x=-2000+240, y=-2000+80, width=70,
height=30)

####

self.MR_Knee_bt = tk.Button(bg="#C9DCEA")
self.MR_Knee_bt['text'] = 'Knee'
self.MR_Knee_bt.pack()
self.MR_Knee_bt.place(x=-2000+30, y=-2000+110, width=70,
height=30)

####

self.MR_Ankle_bt = tk.Button(bg="#C9DCEA")
self.MR_Ankle_bt['text'] = 'Ankle'

```

```

self.MR_Ankle_bt.pack()
self.MR_Ankle_bt.place(x=-2000+100, y=-2000+110, width=70,
height=30)
####
self.MR_Biceps_bt = tk.Button(bg="#C9DCEA")
self.MR_Biceps_bt['text'] = 'Biceps'
self.MR_Biceps_bt.pack()
self.MR_Biceps_bt.place(x=-2000+170, y=-2000+110, width=70,
height=30)
####
self.MR_Forearm_bt = tk.Button(bg="#C9DCEA")
self.MR_Forearm_bt['text'] = 'Forearm'
self.MR_Forearm_bt.pack()
self.MR_Forearm_bt.place(x=-2000+240, y=-2000+110,
width=70, height=30)
####
self.MR_Wrist_bt = tk.Button(bg="#C9DCEA")
self.MR_Wrist_bt['text'] = 'Wrist'
self.MR_Wrist_bt.pack()
self.MR_Wrist_bt.place(x=-2000+30, y=-2000+140, width=70,
height=30)
####
self.MR_All_bt = tk.Button(bg="#C9DCEA")
self.MR_All_bt['text'] = 'All'
self.MR_All_bt.pack()
self.MR_All_bt.place(x=-2000+100, y=-2000+140, width=70,
height=30)
#

```

```

self.MR_dactrungtinh2_label = tk.Label(text="Chọn đặc trưng
về",bg="#A19A99",fg="#f2f2f2")
self.MR_dactrungtinh2_label.pack()
self.MR_dactrungtinh2_label.config(font=("Courier bold", 15))
self.MR_dactrungtinh2_label.place(x=-2000+480, y=-2000+10,
width=280, height=30)

self.MR_Age2_bt = tk.Button(bg="#C9DCEA")
self.MR_Age2_bt['text'] = 'Age'
self.MR_Age2_bt.pack()
self.MR_Age2_bt.place(x=-2000+480, y=-2000+50, width=70,
height=30)

###

self.MR_Weight2_bt = tk.Button(bg="#C9DCEA")
self.MR_Weight2_bt['text'] = 'Weight'
self.MR_Weight2_bt.pack()
self.MR_Weight2_bt.place(x=-2000+550, y=-2000+50, width=70,
height=30)

###

self.MR_Height2_bt = tk.Button(bg="#C9DCEA")
self.MR_Height2_bt['text'] = 'Height'
self.MR_Height2_bt.pack()
self.MR_Height2_bt.place(x=-2000+620, y=-2000+50, width=70,
height=30)

###

self.MR_Neck2_bt = tk.Button(bg="#C9DCEA")
self.MR_Neck2_bt['text'] = 'Neck'
self.MR_Neck2_bt.pack()
self.MR_Neck2_bt.place(x=-2000+690, y=-2000+50, width=70,
height=30)

```

```
#####  
self.MR_Chest2_bt = tk.Button(bg="#C9DCEA")  
self.MR_Chest2_bt['text'] = 'Chest'  
self.MR_Chest2_bt.pack()  
self.MR_Chest2_bt.place(x=-2000+480, y=-2000+80, width=70,  
height=30)  
#####  
self.MR_Abdomen2_bt = tk.Button(bg="#C9DCEA")  
self.MR_Abdomen2_bt['text'] = 'Abdomen'  
self.MR_Abdomen2_bt.pack()  
self.MR_Abdomen2_bt.place(x=-2000+550, y=-2000+80,  
width=70, height=30)  
#####  
self.MR_Hip2_bt = tk.Button(bg="#C9DCEA")  
self.MR_Hip2_bt['text'] = 'Hip'  
self.MR_Hip2_bt.pack()  
self.MR_Hip2_bt.place(x=-2000+620, y=-2000+80, width=70,  
height=30)  
#####  
self.MR_Thigh2_bt = tk.Button(bg="#C9DCEA")  
self.MR_Thigh2_bt['text'] = 'Thigh'  
self.MR_Thigh2_bt.pack()  
self.MR_Thigh2_bt.place(x=-2000+690, y=-2000+80, width=70,  
height=30)  
#####  
self.MR_Knee2_bt = tk.Button(bg="#C9DCEA")  
self.MR_Knee2_bt['text'] = 'Knee'  
self.MR_Knee2_bt.pack()
```



```

        self.MR_Knee2_bt.place(x=-2000+480, y=-2000+110, width=70,
height=30)

        #####

        self.MR_Ankle2_bt = tk.Button(bg="#C9DCEA")
        self.MR_Ankle2_bt['text'] = 'Ankle'
        self.MR_Ankle2_bt.pack()
        self.MR_Ankle2_bt.place(x=-2000+550, y=-2000+110, width=70,
height=30)

        #####

        self.MR_Biceps2_bt = tk.Button(bg="#C9DCEA")
        self.MR_Biceps2_bt['text'] = 'Biceps'
        self.MR_Biceps2_bt.pack()
        self.MR_Biceps2_bt.place(x=-2000+620, y=-2000+110, width=70,
height=30)

        #####

        self.MR_Forearm2_bt = tk.Button(bg="#C9DCEA")
        self.MR_Forearm2_bt['text'] = 'Forearm'
        self.MR_Forearm2_bt.pack()
        self.MR_Forearm2_bt.place(x=-2000+690, y=-2000+110,
width=70, height=30)

        #####

        self.MR_Wrist2_bt = tk.Button(bg="#C9DCEA")
        self.MR_Wrist2_bt['text'] = 'Wrist'
        self.MR_Wrist2_bt.pack()
        self.MR_Wrist2_bt.place(x=-2000+480, y=-2000+140, width=70,
height=30)

##### Kernel_PCA
#####
#####

```

```

        self.Kernel_PCA_bt=Button(self.master, text="Kernel
PCA",width=20, height=2)
        self.Kernel_PCA_bt.config(font=("Courier bold", 20))
        self.Kernel_PCA_bt.pack()
        self.Kernel_PCA_bt.place(x=-2000+220,y=-2000+127)
##### Logistic_regression
#####
#####

        self.Logistic_regression_bt=Button(self.master, text="Logistic
Regression",width=20, height=2)
        self.Logistic_regression_bt.config(font=("Courier bold", 20))
        self.Logistic_regression_bt.pack()
        self.Logistic_regression_bt.place(x=-2000+220,y=-2000+207)
##### PCA
#####
#####

        self.PCA_bt=Button(self.master, text="PCA",width=20, height=2)
        self.PCA_bt.config(font=("Courier bold", 20))
        self.PCA_bt.pack()
        self.PCA_bt.place(x=-2000+220,y=-2000+287)
#
        self.PCA_Dactrung_label=tk.Label(text="Chọn số đặc trưng
mới",bg="#A19A99",fg="#f2f2f2")
        self.PCA_Dactrung_label.config(font=("Courier bold", 12))
        self.PCA_Dactrung_label.pack()
        self.PCA_Dactrung_label.place(x=-2000+565, y=-2000+95,
width=280, height=30)
        val=[]
        for i in range(1,39):

```

```

        val.append(i)

self.PCA_label=tk.Label(text="PCA",bg="#A19A99",fg="#f2f2f2")
    self.PCA_label.config(font=("Courier bold", 40))
    self.PCA_label.pack()
    self.PCA_label.place(x=-2000+250, y=-2000+10, width=280,
height=70)
        self.PCA_Selecting_Pri_Components= ttk.Combobox(self.master,
values=val,width=14, height=5)
        self.PCA_Selecting_Pri_Components.set(1)
        self.PCA_Selecting_Pri_Components.pack()

self.PCA_Selecting_Pri_Components.place(x=-2000+640,y=-2000+125)
    self.PCA_Selecting_Pri_Components.config(font=("Courier bold",
12))
        self.PCA_ve2d_bt=Button(self.master, text="Vẽ 2d",width=13,
height=1)
        self.PCA_ve2d_bt.config(font=("Courier bold", 12))
        self.PCA_ve2d_bt.pack()
        self.PCA_ve2d_bt.place(x=-2000+640,y=-2000+360)
        self.PCA_ve3d_bt=Button(self.master, text="Vẽ 3d",width=13,
height=1)
        self.PCA_ve3d_bt.config(font=("Courier bold", 12))
        self.PCA_ve3d_bt.pack()
        self.PCA_ve3d_bt.place(x=-2000+640,y=-2000+395)
        self.PCA_Eigen_Values_bt=Button(self.master, text="Eigen
Values",width=13, height=1)
        self.PCA_Eigen_Values_bt.config(font=("Courier bold", 12))
        self.PCA_Eigen_Values_bt.pack()

```

```

self.PCA_Eigen_Values_bt.place(x=-2000+640,y=-2000+220)
self.PCA_Eigen_Vectors_bt=Button(self.master, text="Eigen
Vectors",width=13, height=1)
self.PCA_Eigen_Vectors_bt.config(font=("Courier bold", 12))
self.PCA_Eigen_Vectors_bt.pack()
self.PCA_Eigen_Vectors_bt.place(x=-2000+640,y=-2000+255)
self.PCA_new_Data_bt=Button(self.master, text="New
Data",width=13, height=1)
self.PCA_new_Data_bt.config(font=("Courier bold", 12))
self.PCA_new_Data_bt.pack()
self.PCA_new_Data_bt.place(x=-2000+640,y=-2000+290)
self.PCA_danhGia_bt=Button(self.master, text="Đánh
Giá",width=13, height=1)
self.PCA_danhGia_bt.config(font=("Courier bold", 12))
self.PCA_danhGia_bt.pack()
self.PCA_danhGia_bt.place(x=-2000+640,y=-2000+325)
self.PCA_Mean_bt=Button(self.master, text="Mean",width=13,
height=1)
self.PCA_Mean_bt.config(font=("Courier bold", 12))
self.PCA_Mean_bt.pack()
self.PCA_Mean_bt.place(x=-2000+640,y=-2000+150)
self.PCA_Selecting_Pri_Components_bt=Button(self.master,
text="Se_Pri_Components",width=13, height=1)
self.PCA_Selecting_Pri_Components_bt.config(font=("Courier
bold", 12))
self.PCA_Selecting_Pri_Components_bt. pack()

self.PCA_Selecting_Pri_Components_bt.place(x=-2000+640,y=-2000+18
5)

```

```

        self.PCA_ketQua_bt=Button(self.master, text="",width=25,
height=8,bg="#ffffff")
        self.PCA_ketQua_bt.config(font=("Courier bold", 20))
        self.PCA_ketQua_bt.pack()
        self.PCA_ketQua_bt.place(x=-2000+100,y=-2000+155)
#####

        self.LDA_bt=Button(self.master, text="LDA",width=20, height=2)
        self.LDA_bt.config(font=("Courier bold", 20))
        self.LDA_bt.pack()
        self.LDA_bt.place(x=-2000+220,y=-2000+367)
##### Poly_regression
#####
#####

        self.Poly_regression_bt=Button(self.master, text="Poly
Regression",width=20, height=2)
        self.Poly_regression_bt.config(font=("Courier bold", 20))
        self.Poly_regression_bt.pack()
        self.Poly_regression_bt.place(x=-2000+220,y=-2000+448)

##### Event
#####
#####

        ##### Login #####
        self.logbtn['command']=self.logbtn_Click
        ##### MR #####
        self.MR_tinhToan_bt['command']=self.MR_tinhToan_Click
        self.MR_ve2d_bt['command']=self.MR_ve2d_Click
        self.MR_ve3d_bt['command']=self.MR_ve3d_Click
        self.MR_Age_bt['command']=self.MR_Age_Click

```

```
self.MR_Weight_bt['command']=self.MR_Weight_Click
self.MR_Height_bt['command']=self.MR_Height_Click
self.MR_Neck_bt['command']=self.MR_Neck_Click
self.MR_Chest_bt['command']=self.MR_Chest_Click
self.MR_Abdomen_bt['command']=self.MR_Abdomen_Click
self.MR_Hip_bt['command']=self.MR_Hip_Click
self.MR_Thigh_bt['command']=self.MR_Thigh_Click
self.MR_Knee_bt['command']=self.MR_Knee_Click
self.MR_Ankle_bt['command']=self.MR_Ankle_Click
self.MR_Biceps_bt['command']=self.MR_Biceps_Click
self.MR_Forearm_bt ['command']=self.MR_Forearm_Click
self.MR_Wrist_bt ['command']=self.MR_Wrist_Click
self.Back_bt['command']=self.back_Click
self.MR_bt['command']=self.MR_Click
self.MR_All_bt['command']=self.MR_All_Click
self.MR_Age2_bt['command']=self.MR_Age2_Click
self.MR_Weight2_bt['command']=self.MR_Weight2_Click
self.MR_Height2_bt['command']=self.MR_Height2_Click
self.MR_Neck2_bt['command']=self.MR_Neck2_Click
self.MR_Chest2_bt['command']=self.MR_Chest2_Click
self.MR_Abdomen2_bt['command']=self.MR_Abdomen2_Click
self.MR_Hip2_bt['command']=self.MR_Hip2_Click
self.MR_Thigh2_bt['command']=self.MR_Thigh2_Click
self.MR_Knee2_bt['command']=self.MR_Knee2_Click
self.MR_Ankle2_bt['command']=self.MR_Ankle2_Click
self.MR_Biceps2_bt['command']=self.MR_Biceps2_Click
self.MR_Forearm2_bt['command']=self.MR_Forearm2_Click
self.MR_Wrist2_bt['command']=self.MR_Wrist2_Click
```

```

##### PCA
#####

self.PCA_bt['command']=self.PCA_Click

self.PCA_Selecting_Pri_Components.bind('<<ComboboxSelected>>',
self.on_select)

self.PCA_Mean_bt['command']=self.PCA_Mean_bt_Click

self.PCA_Eigen_Values_bt['command']=self.PCA_Eigen_Values_bt_Click

self.PCA_Selecting_Pri_Components_bt['command']=self.PCA_Selecting_Pri_Components_bt_Click

self.PCA_Eigen_Vectors_bt['command']=self.PCA_Eigen_Vectors_bt_Click

self.PCA_new_Data_bt['command']=self.PCA_new_Data_bt_Click
self.PCA_ve2d_bt['command']=self.PCA_ve2d_bt_Click
self.PCA_ve3d_bt['command']=self.PCA_ve3d_bt_Click
self.PCA_danhGia_bt['command']=self.PCA_danhGia_bt_Click

def logbtn_Click(self,event=None):
    if self.entry_username.get()=='Nhom1' and
self.entry_password.get()=='123456':
        isLogin.append(1)
        isAll.clear()
        ve.clear()
        tinhToan.clear()
#####

```

```

self.label_loginForm.place(x=-2000+250,y=-2000+10)
self.label_username.place(x=-2000+40,y=-2000+250)
self.label_password.place(x=-2000+40,y=-2000+300)
self.entry_username.place(x=-2000+180,y=-2000+250)
self.entry_password.place(x=-2000+180,y=-2000+300)
self.logbtn.place(x=-2000+220,y=-2000+350, width=250,
height=50)

self.Back_bt.place(x=0,y=560,width=60, height=40)
#####

self.MR_bt.place(x=220,y=47)
self.Kernel_PCA_bt.place(x=220,y=127)
self.Logistic_regression_bt.place(x=220,y=207)
self.PCA_bt.place(x=220,y=287)
self.LDA_bt.place(x=220,y=367)
self.Poly_regression_bt.place(x=220,y=448)
#####

else:
messagebox.showerror('Lỗi','Sai tài khoản hoặc mật khẩu!')
if 1 not in isLogin:
self.label_loginForm.place(x=250,y=10)
self.label_username.place(x=40,y=250)
self.label_password.place(x=40,y=300)
self.entry_username.place(x=180,y=250)
self.entry_password.place(x=180,y=300)
self.logbtn.place(x=220,y=350, width=250, height=50)
self.Back_bt.place(x=-2000+0,y=-2000+560,width=60, height=40)

#####
#####

```



```
self.MR_ketQua_bt.place(x=-2000+250, y=-2000+300, width=300,
height=100)
self.MR_tinhToan_bt.place(x=-2000+252, y=-2000+420,
width=90, height=50)
self.MR_ve2d_bt.place(x=-2000+352, y=-2000+420, width=90,
height=50)
self.MR_ve3d_bt.place(x=-2000+452, y=-2000+420, width=90,
height=50)
self.MR_label.place(x=-2000+260, y=-2000+5, width=280,
height=60)
self.MR_dactrungtinh_label.place(x=-2000+30, y=-2000+10,
width=280, height=30)
self.MR_Age_bt.place(x=-2000+30, y=-2000+50, width=70,
height=30)
self.MR_Weight_bt.place(x=-2000+100, y=-2000+50, width=70,
height=30)
self.MR_Height_bt.place(x=-2000+170, y=-2000+50, width=70,
height=30)
self.MR_Neck_bt.place(x=-2000+240, y=-2000+50, width=70,
height=30)
self.MR_Chest_bt.place(x=-2000+30, y=-2000+80, width=70,
height=30)
self.MR_Abdomen_bt.place(x=-2000+100, y=-2000+80, width=70,
height=30)
self.MR_Hip_bt.place(x=-2000+170, y=-2000+80, width=70,
height=30)
self.MR_Thigh_bt.place(x=-2000+240, y=-2000+80, width=70,
height=30)
```

```
self.MR_Knee_bt.place(x=-2000+30, y=-2000+110, width=70,
height=30)
self.MR_Ankle_bt.place(x=-2000+100, y=-2000+110, width=70,
height=30)
self.MR_Biceps_bt.place(x=-2000+170, y=-2000+110, width=70,
height=30)
self.MR_Forearm_bt .place(x=-2000+240, y=-2000+110,
width=70, height=30)
self.MR_Wrist_bt .place(x=-2000+30, y=-2000+140, width=70,
height=30)
self.MR_All_bt.place(x=-2000+100, y=-2000+140, width=70,
height=30)
self.MR_dactrungtinh2_label.place(x=-2000+480, y=-2000+10,
width=280, height=30)
self.MR_Age2_bt.place(x=-2000+480, y=-2000+50, width=70,
height=30)
self.MR_Weight2_bt.place(x=-2000+550, y=-2000+50, width=70,
height=30)
self.MR_Height2_bt.place(x=-2000+620, y=-2000+50, width=70,
height=30)
self.MR_Neck2_bt.place(x=-2000+690, y=-2000+50, width=70,
height=30)
self.MR_Chest2_bt.place(x=-2000+480, y=-2000+80, width=70,
height=30)
self.MR_Abdomen2_bt.place(x=-2000+550, y=-2000+80,
width=70, height=30)
self.MR_Hip2_bt.place(x=-2000+620, y=-2000+80, width=70,
height=30)
```

```

        self.MR_Thigh2_bt.place(x=-2000+690, y=-2000+80, width=70,
height=30)
        self.MR_Knee2_bt.place(x=-2000+480, y=-2000+110, width=70,
height=30)
        self.MR_Ankle2_bt.place(x=-2000+550, y=-2000+110, width=70,
height=30)
        self.MR_Biceps2_bt.place(x=-2000+620, y=-2000+110, width=70,
height=30)
        self.MR_Forearm2_bt.place(x=-2000+690, y=-2000+110,
width=70, height=30)
        self.MR_Wrist2_bt.place(x=-2000+480, y=-2000+140, width=70,
height=30)

```

```

#####

```

```

        self.Back_bt.place(x=-2000+0,y=-2000+560,width=60, height=40)
        self.MR_bt.place(x=-2000+220,y=-2000+47)
        self.Kernel_PCA_bt.place(x=-2000+220,y=-2000+127)
        self.Logistic_regression_bt.place(x=-2000+220,y=-2000+207)
        self.PCA_bt.place(x=-2000+220,y=-2000+287)
        self.LDA_bt.place(x=-2000+220,y=-2000+367)
        self.Poly_regression_bt.place(x=-2000+220,y=-2000+448)
        def back_Click(self,event=None):
            if 2 in isLogin:
                isAll.clear()
                tinhToan.clear()
                ve.clear()
                isLogin.remove(2)
            self.Back_bt.place(x=0,y=560,width=60, height=40)
            self.MR_bt.place(x=220,y=47)

```

```
self.Kernel_PCA_bt.place(x=220,y=127)
self.Logistic_regression_bt.place(x=220,y=207)
self.PCA_bt.place(x=220,y=287)
self.LDA_bt.place(x=220,y=367)
self.Poly_regression_bt.place(x=220,y=448)
```

```
#####
```

```
#####
```

```
self.MR_ketQua_bt.place(x=-2000+250, y=-2000+300, width=300,
height=100)
```

```
self.MR_tinhToan_bt.place(x=-2000+252, y=-2000+420,
width=90, height=50)
```

```
self.MR_ve2d_bt.place(x=-2000+352, y=-2000+420, width=90,
height=50)
```

```
self.MR_ve3d_bt.place(x=-2000+452, y=-2000+420, width=90,
height=50)
```

```
self.MR_label.place(x=-2000+260, y=-2000+5, width=280,
height=60)
```

```
self.MR_dactrungtinh_label.place(x=-2000+30, y=-2000+10,
width=280, height=30)
```

```
self.MR_Age_bt.place(x=-2000+30, y=-2000+50, width=70,
height=30)
```

```
self.MR_Weight_bt.place(x=-2000+100, y=-2000+50, width=70,
height=30)
```

```
self.MR_Height_bt.place(x=-2000+170, y=-2000+50, width=70,
height=30)
```

```
self.MR_Neck_bt.place(x=-2000+240, y=-2000+50, width=70,
height=30)
```

```
self.MR_Chest_bt.place(x=-2000+30, y=-2000+80, width=70,
height=30)
self.MR_Abdomen_bt.place(x=-2000+100, y=-2000+80, width=70,
height=30)
self.MR_Hip_bt.place(x=-2000+170, y=-2000+80, width=70,
height=30)
self.MR_Thigh_bt.place(x=-2000+240, y=-2000+80, width=70,
height=30)
self.MR_Knee_bt.place(x=-2000+30, y=-2000+110, width=70,
height=30)
self.MR_Ankle_bt.place(x=-2000+100, y=-2000+110, width=70,
height=30)
self.MR_Biceps_bt.place(x=-2000+170, y=-2000+110, width=70,
height=30)
self.MR_Forearm_bt .place(x=-2000+240, y=-2000+110,
width=70, height=30)
self.MR_Wrist_bt .place(x=-2000+30, y=-2000+140, width=70,
height=30)
self.MR_All_bt.place(x=-2000+100, y=-2000+140, width=70,
height=30)
self.MR_dactrungtinhh2_label.place(x=-2000+480, y=-2000+10,
width=280, height=30)
self.MR_Age2_bt.place(x=-2000+480, y=-2000+50, width=70,
height=30)
self.MR_Weight2_bt.place(x=-2000+550, y=-2000+50, width=70,
height=30)
self.MR_Height2_bt.place(x=-2000+620, y=-2000+50, width=70,
height=30)
```

```

        self.MR_Neck2_bt.place(x=-2000+690, y=-2000+50, width=70,
height=30)
        self.MR_Chest2_bt.place(x=-2000+480, y=-2000+80, width=70,
height=30)
        self.MR_Abdomen2_bt.place(x=-2000+550, y=-2000+80,
width=70, height=30)
        self.MR_Hip2_bt.place(x=-2000+620, y=-2000+80, width=70,
height=30)
        self.MR_Thigh2_bt.place(x=-2000+690, y=-2000+80, width=70,
height=30)
        self.MR_Knee2_bt.place(x=-2000+480, y=-2000+110, width=70,
height=30)
        self.MR_Ankle2_bt.place(x=-2000+550, y=-2000+110, width=70,
height=30)
        self.MR_Biceps2_bt.place(x=-2000+620, y=-2000+110, width=70,
height=30)
        self.MR_Forearm2_bt.place(x=-2000+690, y=-2000+110,
width=70, height=30)
        self.MR_Wrist2_bt.place(x=-2000+480, y=-2000+140, width=70,
height=30)
        ##### PCA
        #####
        self.PCA_Dactrung_label.place(x=-2000+565, y=-2000+95,
width=280, height=30)
        self.PCA_label.place(x=-2000+250, y=-2000+10, width=280,
height=70)

self.PCA_Selecting_Pri_Components.place(x=-2000+640,y=-2000+125)
        self.PCA_ve2d_bt.place(x=-2000+640,y=-2000+360)

```

```

self.PCA_ve3d_bt.place(x=-2000+640,y=-2000+395)
self.PCA_Eigen_Values_bt.place(x=-2000+640,y=-2000+220)
self.PCA_Eigen_Vectors_bt.place(x=-2000+640,y=-2000+255)
self.PCA_new_Data_bt.place(x=-2000+640,y=-2000+290)
self.PCA_danhGia_bt.place(x=-2000+640,y=-2000+325)
self.PCA_Mean_bt.place(x=-2000+640,y=-2000+150)

self.PCA_Selecting_Pri_Components_bt.place(x=-2000+640,y=-2000+18
5)

self.PCA_ketQua_bt.place(x=-2000+100,y=-2000+155)
elif 1 in isLogin:
isLogin.clear()
self.label_loginForm.place(x=250,y=10)
self.label_username.place(x=40,y=250)
self.label_password.place(x=40,y=300)
self.entry_username.place(x=180,y=250)
self.entry_password.place(x=180,y=300)
self.logbtn.place(x=220,y=350, width=250, height=50)

#####

self.Back_bt.place(x=-2000+0,y=-2000+560,width=60, height=40)
self.MR_bt.place(x=-2000+220,y=-2000+47)
self.Kernel_PCA_bt.place(x=-2000+220,y=-2000+127)
self.Logistic_regression_bt.place(x=-2000+220,y=-2000+207)
self.PCA_bt.place(x=-2000+220,y=-2000+287)
self.LDA_bt.place(x=-2000+220,y=-2000+367)
self.Poly_regression_bt.place(x=-2000+220,y=-2000+448)

```

```
#####  
#####
```

```
self.MR_ketQua_bt.place(x=-2000+250, y=-2000+300, width=300,  
height=100)
```

```
self.MR_tinhToan_bt.place(x=-2000+252, y=-2000+420,  
width=90, height=50)
```

```
self.MR_ve2d_bt.place(x=-2000+352, y=-2000+420, width=90,  
height=50)
```

```
self.MR_ve3d_bt.place(x=-2000+452, y=-2000+420, width=90,  
height=50)
```

```
self.MR_label.place(x=-2000+260, y=-2000+5, width=280,  
height=60)
```

```
self.MR_dactrungtinh_label.place(x=-2000+30, y=-2000+10,  
width=280, height=30)
```

```
self.MR_Age_bt.place(x=-2000+30, y=-2000+50, width=70,  
height=30)
```

```
self.MR_Weight_bt.place(x=-2000+100, y=-2000+50, width=70,  
height=30)
```

```
self.MR_Height_bt.place(x=-2000+170, y=-2000+50, width=70,  
height=30)
```

```
self.MR_Neck_bt.place(x=-2000+240, y=-2000+50, width=70,  
height=30)
```

```
self.MR_Chest_bt.place(x=-2000+30, y=-2000+80, width=70,  
height=30)
```

```
self.MR_Abdomen_bt.place(x=-2000+100, y=-2000+80, width=70,  
height=30)
```

```
self.MR_Hip_bt.place(x=-2000+170, y=-2000+80, width=70,  
height=30)
```



```
self.MR_Thigh_bt.place(x=-2000+240, y=-2000+80, width=70,
height=30)
self.MR_Knee_bt.place(x=-2000+30, y=-2000+110, width=70,
height=30)
self.MR_Ankle_bt.place(x=-2000+100, y=-2000+110, width=70,
height=30)
self.MR_Biceps_bt.place(x=-2000+170, y=-2000+110, width=70,
height=30)
self.MR_Forearm_bt .place(x=-2000+240, y=-2000+110,
width=70, height=30)
self.MR_Wrist_bt .place(x=-2000+30, y=-2000+140, width=70,
height=30)
self.MR_All_bt.place(x=-2000+100, y=-2000+140, width=70,
height=30)
self.MR_dactrungtinh2_label.place(x=-2000+480, y=-2000+10,
width=280, height=30)
self.MR_Age2_bt.place(x=-2000+480, y=-2000+50, width=70,
height=30)
self.MR_Weight2_bt.place(x=-2000+550, y=-2000+50, width=70,
height=30)
self.MR_Height2_bt.place(x=-2000+620, y=-2000+50, width=70,
height=30)
self.MR_Neck2_bt.place(x=-2000+690, y=-2000+50, width=70,
height=30)
self.MR_Chest2_bt.place(x=-2000+480, y=-2000+80, width=70,
height=30)
self.MR_Abdomen2_bt.place(x=-2000+550, y=-2000+80,
width=70, height=30)
```

```

self.MR_Hip2_bt.place(x=-2000+620, y=-2000+80, width=70,
height=30)
self.MR_Thigh2_bt.place(x=-2000+690, y=-2000+80, width=70,
height=30)
self.MR_Knee2_bt.place(x=-2000+480, y=-2000+110, width=70,
height=30)
self.MR_Ankle2_bt.place(x=-2000+550, y=-2000+110, width=70,
height=30)
self.MR_Biceps2_bt.place(x=-2000+620, y=-2000+110, width=70,
height=30)
self.MR_Forearm2_bt.place(x=-2000+690, y=-2000+110,
width=70, height=30)
self.MR_Wrist2_bt.place(x=-2000+480, y=-2000+140, width=70,
height=30)

```

- **Các hàm sự kiện tương ứng với các đối tượng giao diện**

```

def on_select(self,event=None):
    if event: #
        PCA_n_pc.clear()
        PCA_n_pc.append(int( event.widget.get()))
    def PCA_danhGia_bt_Click(self,event=None):
        knn=PCA.KNN(n_pc=PCA_n_pc[0],k=1)
        s="Kết quả:\n F(knn_bd)=%.3f"%knn[0]
        s1="\n F(knn_PCA)=%.3f"%knn[1]
        s+=s1
        self.PCA_ketQua_bt.config(text=s)
        self.PCA_ketQua_bt.place(x=100,y=155)

    def PCA_ve2d_bt_Click(self,event=None):
        PCA.Draw_2d(1,"Vẽ 2d cho tập gốc")

```

```
PCA.Draw_2d(2,"Vẽ 2d cho tập iris")
self.PCA_ketQua_bt.place(x=-2000+100,y=-2000+155)
```

```
def PCA_ve3d_bt_Click(self,event=None):
PCA.Draw_3d(1,"Vẽ 3d cho tập gốc")
PCA.Draw_3d(2,"Vẽ 3d cho tập iris")
self.PCA_ketQua_bt.place(x=-2000+100,y=-2000+155)
```

```
def PCA_new_Data_bt_Click(self,event=None):
print("\n\n\n\t\t\tNew Data")
new_Data=PCA.new_Data(n_pc=PCA_n_pc[0],k=1)
print(new_Data)
print("\n\n\n")
self.PCA_ketQua_bt.place(x=-2000+100,y=-2000+155)
```

```
def PCA_Eigen_Vectors_bt_Click(self,event=None):
print("\n\n\n\t\t\tEigen_Vectors")
print(PCA.Eigen_Vectors(1))
print("\n\n\n")
self.PCA_ketQua_bt.place(x=-2000+100,y=-2000+155)
```

```
def PCA_Eigen_Values_bt_Click(self,event=None):
self.PCA_ketQua_bt.place(x=-2000+100,y=-2000+155)
Eigen_Values=PCA.Eigen_Values(1)
fig, ax = plt.subplots()
fig.patch.set_visible(False)
ax.axis('off')
ax.axis('tight')
a=[]
```

```

b=[]
c=[]
dem=0
for i in range(len(Eigen_Values)):
    if i<4:
        a.append(Eigen_Values[i])
    elif i>=4 :
        dem+=1
        b.append(Eigen_Values[i])
        if dem==4:
            c.append((np.array(b)).T)
            dem=0
            b.clear()
            if dem==3 and i==len(Eigen_Values)-1:
                b.append(" ")
                c.append((np.array(b)).T)
df = pd.DataFrame(c, columns=a)
ax.table(cellText=df.values, colLabels=df.columns, loc='center')
fig.tight_layout()
plt.show()

```

```

def PCA_Selecting_Pri_Components_bt_Click(self,event=None):
    dt=PCA.Selecting_Pri_Components(1)
    self.PCA_ketQua_bt.place(x=-2000+100,y=-2000+155)

```

```

def PCA_Mean_bt_Click(self,event=None):
    self.PCA_ketQua_bt.place(x=-2000+100,y=-2000+155)
    fig, ax = plt.subplots()

```

```

fig.patch.set_visible(False)
ax.axis('off')
ax.axis('tight')
dt=PCA.Standardizing(1)
mean_vec=PCA.Mean(dt)
a=[]
b=[]
c=[]
dem=0
for i in range(len(mean_vec)):
    if i<4:
        a.append(mean_vec[i])
    elif i>=4 :
        dem+=1
        b.append(mean_vec[i])
        if dem==4:
            c.append((np.array(b)).T)
            dem=0
            b.clear()
            if dem==3 and i==len(mean_vec)-1:
                b.append(" ")
                c.append((np.array(b)).T)
df = pd.DataFrame(c, columns=a)
ax.table(cellText=df.values, colLabels=df.columns, loc='center')
fig.tight_layout()
plt.show()

def PCA_Click(self,event=None):
    isLogin.append(2)

```

```

k=2000
self.MR_bt.place(x=-k+220,y=47)
self.Kernel_PCA_bt.place(x=-k+220,y=127)
self.Logistic_regression_bt.place(x=-k+220,y=207)
self.PCA_bt.place(x=-k+220,y=287)
self.LDA_bt.place(x=-k+220,y=367)
self.Poly_regression_bt.place(x=-k+220,y=448)
#
self.PCA_Dactrung_label.place(x=565, y=95, width=280,
height=30)
self.PCA_label.place(x=250, y=10, width=280, height=70)
self.PCA_Selecting_Pri_Components.place(x=640,y=125)
self.PCA_ve2d_bt.place(x=640,y=360)
self.PCA_ve3d_bt.place(x=640,y=395)
self.PCA_Eigen_Values_bt.place(x=640,y=220)
self.PCA_Eigen_Vectors_bt.place(x=640,y=255)
self.PCA_new_Data_bt.place(x=640,y=290)
self.PCA_danhGia_bt.place(x=640,y=325)
self.PCA_Mean_bt.place(x=640,y=150)
self.PCA_Selecting_Pri_Components_bt.place(x=640,y=185)
self.PCA_ketQua_bt.place(x=-2000+100,y=-2000+155)

```

```

def MR_Click(self,event=None):

```

```
isLogin.append(2)
k=2000
self.MR_bt.place(x=-k+220,y=47)
self.Kernel_PCA_bt.place(x=-k+220,y=127)
self.Logistic_regression_bt.place(x=-k+220,y=207)
self.PCA_bt.place(x=-k+220,y=287)
self.LDA_bt.place(x=-k+220,y=367)
self.Poly_regression_bt.place(x=-k+220,y=448)
```

```
#####
```

```
#
```

```
self.MR_Age_bt.config(bg="#C9DCEA")
self.MR_Weight_bt.config(bg="#C9DCEA")
self.MR_Height_bt.config(bg="#C9DCEA")
self.MR_Neck_bt.config(bg="#C9DCEA")
self.MR_Chest_bt.config(bg="#C9DCEA")
self.MR_Abdomen_bt.config(bg="#C9DCEA")
self.MR_Hip_bt.config(bg="#C9DCEA")
self.MR_Thigh_bt.config(bg="#C9DCEA")
self.MR_Knee_bt.config(bg="#C9DCEA")
self.MR_Ankle_bt.config(bg="#C9DCEA")
self.MR_Biceps_bt.config(bg="#C9DCEA")
self.MR_Forearm_bt.config(bg="#C9DCEA")
self.MR_Wrist_bt.config(bg="#C9DCEA")
self.MR_Age2_bt.config(bg="#C9DCEA")
self.MR_Weight2_bt.config(bg="#C9DCEA")
self.MR_Height2_bt.config(bg="#C9DCEA")
self.MR_Neck2_bt.config(bg="#C9DCEA")
self.MR_Chest2_bt.config(bg="#C9DCEA")
```

```
self.MR_Abdomen2_bt.config(bg="#C9DCEA")
self.MR_Hip2_bt.config(bg="#C9DCEA")
self.MR_Thigh2_bt.config(bg="#C9DCEA")
self.MR_Knee2_bt.config(bg="#C9DCEA")
self.MR_Ankle2_bt.config(bg="#C9DCEA")
self.MR_Biceps2_bt.config(bg="#C9DCEA")
self.MR_Forearm2_bt.config(bg="#C9DCEA")
self.MR_Wrist2_bt.config(bg="#C9DCEA")
self.MR_All_bt.config(bg="#C9DCEA")
self.MR_ketQua_bt.config(text="")
```

```
#####
```

```
#####
```

```
#####
```

```
#####
```

```
self.MR_ketQua_bt.place(x=250, y=300, width=300, height=100)
self.MR_tinhToan_bt.place(x=252, y=420, width=90, height=50)
self.MR_ve2d_bt.place(x=352, y=420, width=90, height=50)
self.MR_ve3d_bt.place(x=452, y=420, width=90, height=50)
self.MR_label.place(x=260, y=5, width=280, height=60)
self.MR_dactrungtinh_label.place(x=30, y=10, width=280,
height=30)
self.MR_Age_bt.place(x=30, y=50, width=70, height=30)
self.MR_Weight_bt.place(x=100, y=50, width=70, height=30)
self.MR_Height_bt.place(x=170, y=50, width=70, height=30)
self.MR_Neck_bt.place(x=240, y=50, width=70, height=30)
self.MR_Chest_bt.place(x=30, y=80, width=70, height=30)
self.MR_Abdomen_bt.place(x=100, y=80, width=70, height=30)
```



```

self.MR_Hip_bt.place(x=170, y=80, width=70, height=30)
self.MR_Thigh_bt.place(x=240, y=80, width=70, height=30)
self.MR_Knee_bt.place(x=30, y=110, width=70, height=30)
self.MR_Ankle_bt.place(x=100, y=110, width=70, height=30)
self.MR_Biceps_bt.place(x=170, y=110, width=70, height=30)
self.MR_Forearm_bt .place(x=240, y=110, width=70, height=30)
self.MR_Wrist_bt .place(x=30, y=140, width=70, height=30)
self.MR_All_bt.place(x=100, y=140, width=70, height=30)
self.MR_dactrungtinht2_label.place(x=480, y=10, width=280,
height=30)
self.MR_Age2_bt.place(x=480, y=50, width=70, height=30)
self.MR_Weight2_bt.place(x=550, y=50, width=70, height=30)
self.MR_Height2_bt.place(x=620, y=50, width=70, height=30)
self.MR_Neck2_bt.place(x=690, y=50, width=70, height=30)
self.MR_Chest2_bt.place(x=480, y=80, width=70, height=30)
self.MR_Abdomen2_bt.place(x=550, y=80, width=70, height=30)
self.MR_Hip2_bt.place(x=620, y=80, width=70, height=30)
self.MR_Thigh2_bt.place(x=690, y=80, width=70, height=30)
self.MR_Knee2_bt.place(x=480, y=110, width=70, height=30)
self.MR_Ankle2_bt.place(x=550, y=110, width=70, height=30)
self.MR_Biceps2_bt.place(x=620, y=110, width=70, height=30)
self.MR_Forearm2_bt.place(x=690, y=110, width=70, height=30)
self.MR_Wrist2_bt.place(x=480, y=140, width=70, height=30)

```

```

def MR_Age_Click(self,event=None):
if 'Age' not in tinhToan:
tinhToan.append('Age')
self.MR_Age_bt.config(bg="#FEB857")
else:

```

```
    tinhToan.remove('Age')
    self.MR_Age_bt.config(bg="#C9DCEA")
    def MR_Weight_Click(self,event=None):
        if 'Weight' not in tinhToan:
            tinhToan.append('Weight')
            self.MR_Weight_bt.config(bg="#FEB857")
        else:
            tinhToan.remove('Weight')
            self.MR_Weight_bt.config(bg="#C9DCEA")
        def MR_Height_Click(self,event=None):
            if 'Height' not in tinhToan:
                tinhToan.append('Height')
                self.MR_Height_bt.config(bg="#FEB857")
            else:
                tinhToan.remove('Height')
                self.MR_Height_bt.config(bg="#C9DCEA")
            def MR_Neck_Click(self,event=None):
                if 'Neck' not in tinhToan:
                    tinhToan.append('Neck')
                    self.MR_Neck_bt.config(bg="#FEB857")
                else:
                    tinhToan.remove('Neck')
                    self.MR_Neck_bt.config(bg="#C9DCEA")
                def MR_Chest_Click(self,event=None):
                    if 'Chest' not in tinhToan:
                        tinhToan.append('Chest')
                        self.MR_Chest_bt.config(bg="#FEB857")
                    else:
                        tinhToan.remove('Chest')
```

```
self.MR_Chest_bt.config(bg="#C9DCEA")
def MR_Abdomen_Click(self,event=None):
    if 'Abdomen' not in tinhToan:
        tinhToan.append('Abdomen')
        self.MR_Abdomen_bt.config(bg="#FEB857")
    else:
        tinhToan.remove('Abdomen')
        self.MR_Abdomen_bt.config(bg="#C9DCEA")
def MR_Hip_Click(self,event=None):
    if 'Hip' not in tinhToan:
        tinhToan.append('Hip')
        self.MR_Hip_bt.config(bg="#FEB857")
    else:
        tinhToan.remove('Hip')
        self.MR_Hip_bt.config(bg="#C9DCEA")
def MR_Thigh_Click(self,event=None):
    if 'Thigh' not in tinhToan:
        tinhToan.append('Thigh')
        self.MR_Thigh_bt.config(bg="#FEB857")
    else:
        tinhToan.remove('Thigh')
        self.MR_Thigh_bt.config(bg="#C9DCEA")
def MR_Knee_Click(self,event=None):
    if 'Knee' not in tinhToan:
        tinhToan.append('Knee')
        self.MR_Knee_bt.config(bg="#FEB857")
    else:
        tinhToan.remove('Knee')
        self.MR_Knee_bt.config(bg="#C9DCEA")
```

```

def MR_Ankle_Click(self,event=None):
    if 'Ankle' not in tinhToan:
        tinhToan.append('Ankle')
        self.MR_Ankle_bt.config(bg="#FEB857")
    else:
        tinhToan.remove('Ankle')
        self.MR_Ankle_bt.config(bg="#C9DCEA")
    def MR_Biceps_Click(self,event=None):
        if 'Biceps' not in tinhToan:
            tinhToan.append('Biceps')
            self.MR_Biceps_bt.config(bg="#FEB857")
        else:
            tinhToan.remove('Biceps')
            self.MR_Biceps_bt.config(bg="#C9DCEA")
    def MR_Forearm_Click(self,event=None):
        if 'Forearm' not in tinhToan:
            tinhToan.append('Forearm')
            self.MR_Forearm_bt .config(bg="#FEB857")
        else:
            tinhToan.remove('Forearm')
            self.MR_Forearm_bt .config(bg="#C9DCEA")
    def MR_Wrist_Click(self,event=None):
        if 'Wrist' not in tinhToan:
            tinhToan.append('Wrist')
            self.MR_Wrist_bt .config(bg="#FEB857")
        else:
            tinhToan.remove('Wrist')
            self.MR_Wrist_bt .config(bg="#C9DCEA")
    def MR_All_Click(self,event=None):

```

if 1 not in isAll:

t=['Age','Weight','Height','Neck','Chest','Abdomen','Hip','Thigh','Knee','Ankle','Biceps','Forearm','Wrist']

for i in t:

if i not in tinhToan:

tinhToan.append(i)

self.MR\_Age\_bt.config(bg="#FEB857")

self.MR\_Weight\_bt.config(bg="#FEB857")

self.MR\_Height\_bt.config(bg="#FEB857")

self.MR\_Neck\_bt.config(bg="#FEB857")

self.MR\_Chest\_bt.config(bg="#FEB857")

self.MR\_Abdomen\_bt.config(bg="#FEB857")

self.MR\_Hip\_bt.config(bg="#FEB857")

self.MR\_Thigh\_bt.config(bg="#FEB857")

self.MR\_Knee\_bt.config(bg="#FEB857")

self.MR\_Ankle\_bt.config(bg="#FEB857")

self.MR\_Biceps\_bt.config(bg="#FEB857")

self.MR\_Forearm\_bt.config(bg="#FEB857")

self.MR\_Wrist\_bt.config(bg="#FEB857")

self.MR\_All\_bt.config(bg="#FEB857")

isAll.append(1)

else:

tinhToan.clear()

isAll.remove(1)

self.MR\_Age\_bt.config(bg="#C9DCEA")

self.MR\_Weight\_bt.config(bg="#C9DCEA")

self.MR\_Height\_bt.config(bg="#C9DCEA")

self.MR\_Neck\_bt.config(bg="#C9DCEA")

```
self.MR_Chest_bt.config(bg="#C9DCEA")
self.MR_Abdomen_bt.config(bg="#C9DCEA")
self.MR_Hip_bt.config(bg="#C9DCEA")
self.MR_Thigh_bt.config(bg="#C9DCEA")
self.MR_Knee_bt.config(bg="#C9DCEA")
self.MR_Ankle_bt.config(bg="#C9DCEA")
self.MR_Biceps_bt.config(bg="#C9DCEA")
self.MR_Forearm_bt.config(bg="#C9DCEA")
self.MR_Wrist_bt.config(bg="#C9DCEA")
self.MR_All_bt.config(bg="#C9DCEA")
```

```
#####
```

```
#####
```

```
def MR_Age2_Click(self,event=None):
    if 'Age' not in ve:
        ve.append('Age')
        self.MR_Age2_bt.config(bg="#FEB857")
    else:
        ve.remove('Age')
        self.MR_Age2_bt.config(bg="#C9DCEA")
    def MR_Weight2_Click(self,event=None):
        if 'Weight' not in ve:
            ve.append('Weight')
            self.MR_Weight2_bt.config(bg="#FEB857")
        else:
            ve.remove('Weight')
            self.MR_Weight2_bt.config(bg="#C9DCEA")
    def MR_Height2_Click(self,event=None):
        if 'Height' not in ve:
            ve.append('Height')
```

```
self.MR_Height2_bt.config(bg="#FEB857")
else:
ve.remove('Height')
self.MR_Height2_bt.config(bg="#C9DCEA")
def MR_Neck2_Click(self,event=None):
if 'Neck' not in ve:
ve.append('Neck')
self.MR_Neck2_bt.config(bg="#FEB857")
else:
ve.remove('Neck')
self.MR_Neck2_bt.config(bg="#C9DCEA")
def MR_Chest2_Click(self,event=None):
if 'Chest' not in ve:
ve.append('Chest')
self.MR_Chest2_bt.config(bg="#FEB857")
else:
ve.remove('Chest')
self.MR_Chest2_bt.config(bg="#C9DCEA")
def MR_Abdomen2_Click(self,event=None):
if 'Abdomen' not in ve:
ve.append('Abdomen')
self.MR_Abdomen2_bt.config(bg="#FEB857")
else:
ve.remove('Abdomen')
self.MR_Abdomen2_bt.config(bg="#C9DCEA")
def MR_Hip2_Click(self,event=None):
if 'Hip' not in ve:
ve.append('Hip')
self.MR_Hip2_bt.config(bg="#FEB857")
```

```
else:
ve.remove('Hip')
self.MR_Hip2_bt.config(bg="#C9DCEA")
def MR_Thigh2_Click(self,event=None):
if 'Thigh' not in ve:
ve.append('Thigh')
self.MR_Thigh2_bt.config(bg="#FEB857")
else:
ve.remove('Thigh')
self.MR_Thigh2_bt.config(bg="#C9DCEA")
def MR_Knee2_Click(self,event=None):
if 'Knee' not in ve:
ve.append('Knee')
self.MR_Knee2_bt.config(bg="#FEB857")
else:
ve.remove('Knee')
self.MR_Knee2_bt.config(bg="#C9DCEA")
def MR_Ankle2_Click(self,event=None):
if 'Ankle' not in ve:
ve.append('Ankle')
self.MR_Ankle2_bt.config(bg="#FEB857")
else:
ve.remove('Ankle')
self.MR_Ankle2_bt.config(bg="#C9DCEA")
def MR_Biceps2_Click(self,event=None):
if 'Biceps' not in ve:
ve.append('Biceps')
self.MR_Biceps2_bt.config(bg="#FEB857")
else:
```



```

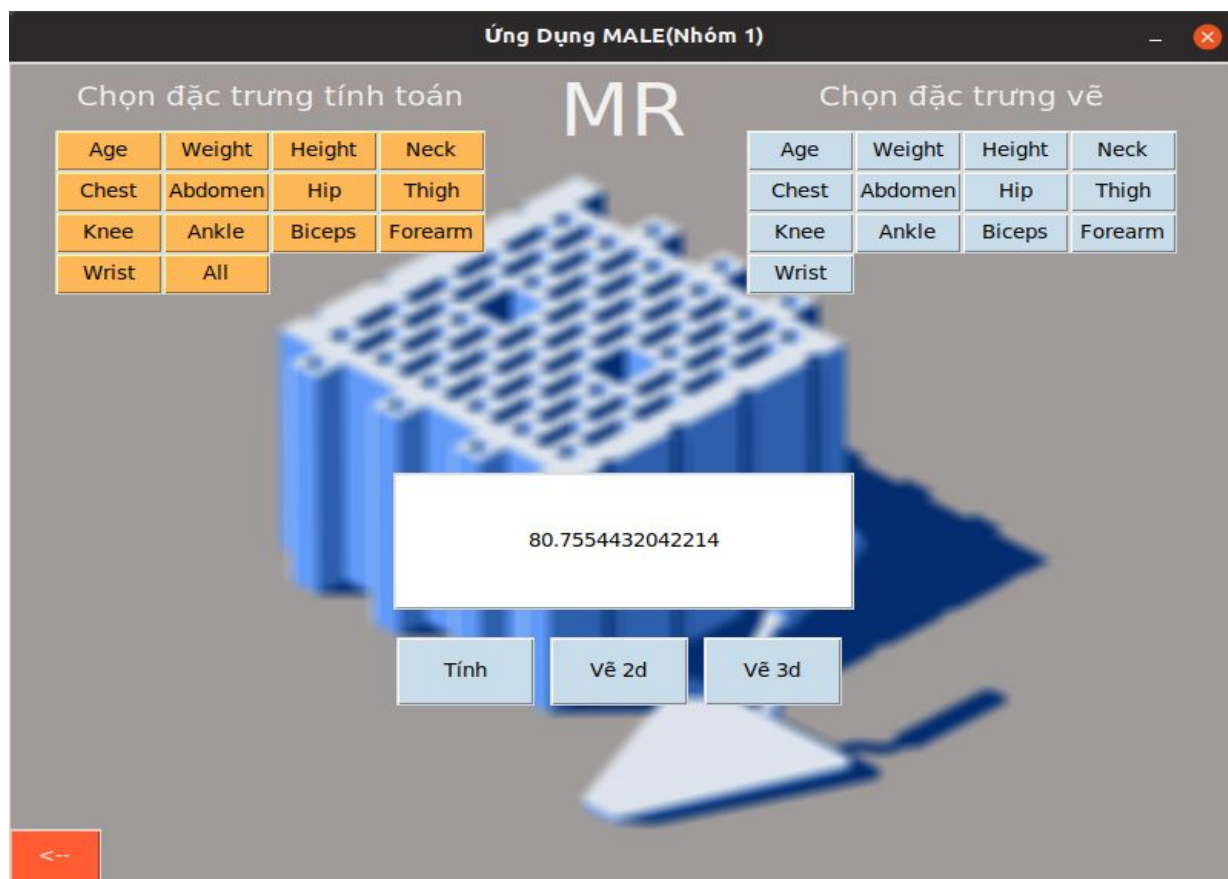
ve.remove('Biceps')
self.MR_Biceps2_bt.config(bg="#C9DCEA")
def MR_Forearm2_Click(self,event=None):
    if 'Forearm' not in ve:
        ve.append('Forearm')
        self.MR_Forearm2_bt.config(bg="#FEB857")
    else:
        ve.remove('Forearm')
        self.MR_Forearm2_bt.config(bg="#C9DCEA")
def MR_Wrist2_Click(self,event=None):
    if 'Wrist' not in ve:
        ve.append('Wrist')
        self.MR_Wrist2_bt.config(bg="#FEB857")
    else:
        ve.remove('Wrist')
        self.MR_Wrist2_bt.config(bg="#C9DCEA")
def MR_tinhToan_Click(self,event=None):
    if len(tinhToan)<1:
        messagebox.showerror("Lỗi", "Vui lòng chọn đặc trưng để tính!")
    if tinhToan is not None:
        self.MR_ketQua_bt.config(text="Lỗi bạn chưa chọn đặc trưng")
        m=MR.tinhToan(tinhToan)
        self.MR_ketQua_bt.config(text="%s"%m)
def MR_ve2d_Click(self,event=None):
    if len(ve)>=2 :
        messagebox.showerror("Lỗi", "Vẽ 2d bạn vui lòng chọn duy nhất 1
đặc trưng vẽ!")
    elif len(ve)==1 and len(tinhToan)>0:
        m=MR.ve2D(tinhToan,ve)

```

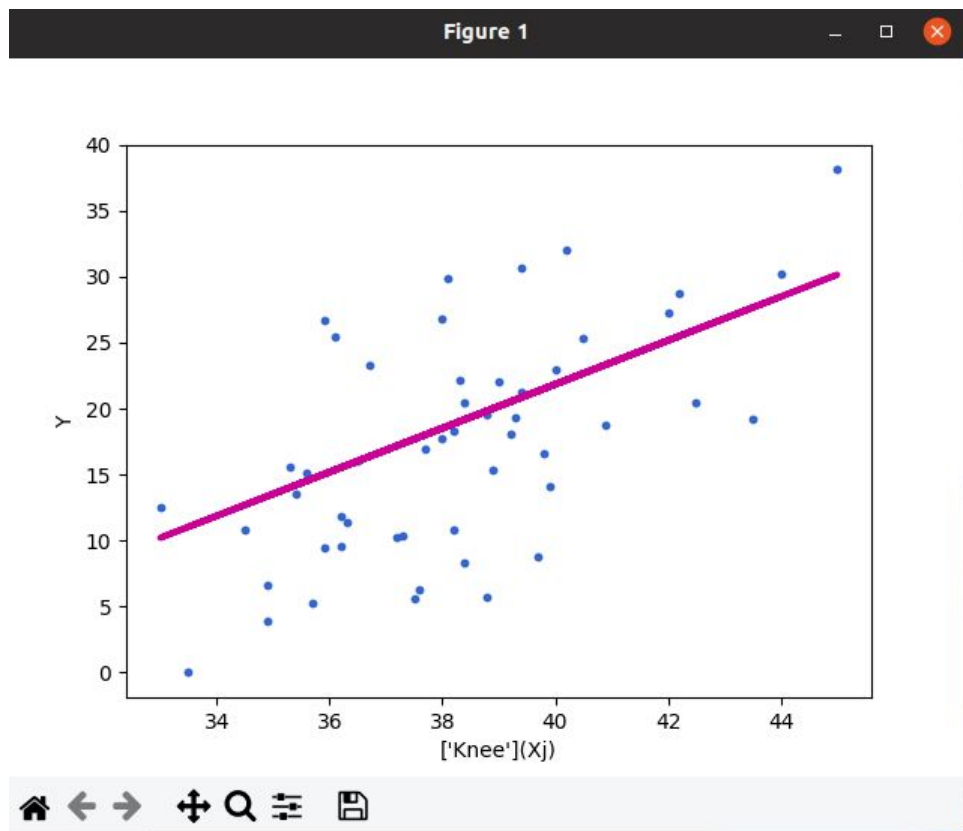
```
elif len(ve)<1:
    messagebox.showerror("Lỗi", "Chưa chọn đặc trưng về!")
def MR_ve3d_Click(self,event=None):
    if len(ve)>=3:
        messagebox.showerror("Lỗi", "Vẽ 3d bạn vui lòng chọn 2 đặc
trung về!")
    elif len(ve)==1:
        messagebox.showerror("Lỗi", "Vẽ 3d bạn vui lòng chọn 2 đặc
trung về!")
    elif len(ve)==2 and len(tinhToan)>1:
        m=MR.ve3D(tinhToan,ve)
    elif len(ve)<1:
        messagebox.showerror("Lỗi", "Chưa chọn đặc trưng về!")
```

## **PHẦN 3: ĐÁNH GIÁ**

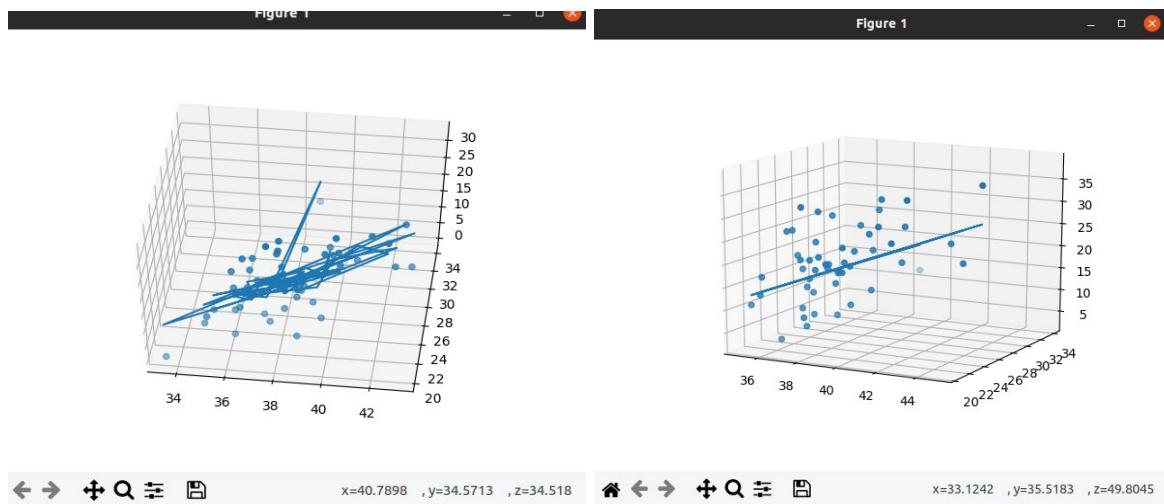
### ***I. MR***



Hình 9: Kết quả tính toán MR.



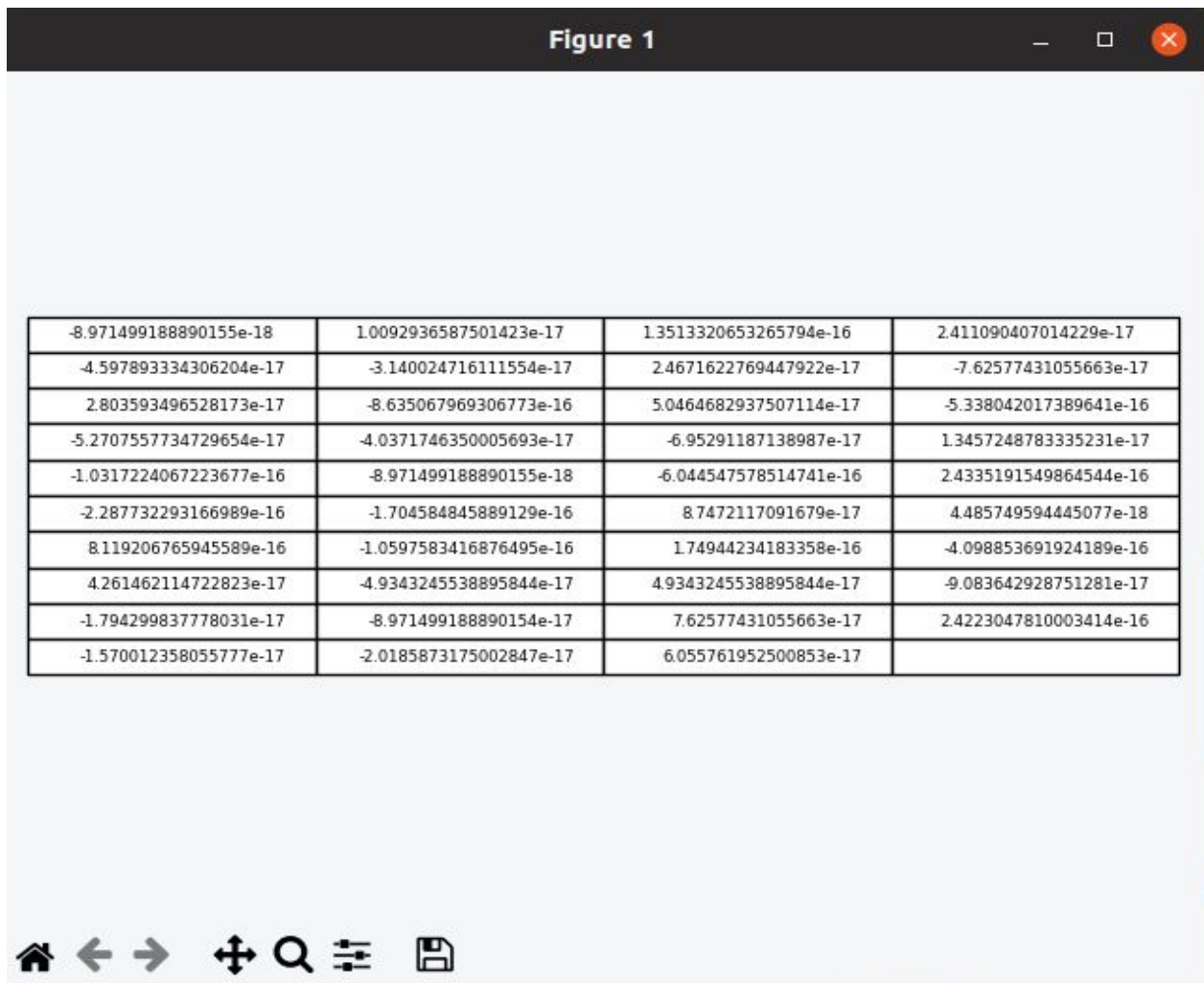
Hình 10: Vẽ 2d MR.



Hình 11: Vẽ 3d MR.

Hình 10, 11 là hình vẽ 2 chiều và 3 chiều cho MR tương ứng 2 chiều là đường thẳng và 3 chiều là mặt phẳng. Ở đây tụi em chọn independent là FAT\_PER còn variables là các đặc trưng còn lại.

## II. PCA



Hình 12: Mean của PCA.

```
tiendat@DELL: ~/Documents/Git/DoAnMale/CuoiKy/Src/BUS
tiendat@DELL:~/Documents/Git/DoAnMale/CuoiKy/Src/BUS$ python3 PCA.py
CovarianceMatrix
[[1.01020408 0.76907398 0.70923948 ... 0.91668656 0.4694096 0.8007302 ]
 [0.76907398 1.01020408 0.63948899 ... 0.93432572 0.20309564 0.9501019 ]
 [0.70923948 0.63948899 1.01020408 ... 0.70766733 0.32780695 0.6406625 ]
 ...
 [0.91668656 0.93432572 0.70766733 ... 1.01020408 0.3725168 0.98037405]
 [0.4694096 0.20309564 0.32780695 ... 0.3725168 1.01020408 0.24667852]
 [0.8007302 0.9501019 0.6406625 ... 0.98037405 0.24667852 1.01020408]]
tiendat@DELL:~/Documents/Git/DoAnMale/CuoiKy/Src/BUS$
```

Hình 13: Covariance Matrix của PCA.

```
tiendat@DELL:~/Documents/Git/DoAnMale/CuoiKy/Src/BUS$ python3 PCA.py
[ 2.47893759e+01  3.49325767e+00  1.99681773e+00  1.62604406e+00
  1.32115645e+00  1.18922531e+00  9.86848252e-01  8.58733657e-01
  6.97469512e-01  5.10826334e-01  4.58978168e-01  3.72916209e-01
  2.81429048e-01  1.99979463e-01  1.52123997e-01  1.13059902e-01
  8.11768677e-02  6.91998152e-02  5.35532432e-02  4.08412460e-02
  2.61660160e-02  2.40698774e-02  1.47564462e-02  1.25093927e-02
  1.06298096e-02  6.93138928e-03  5.31859219e-03  1.81625915e-03
  1.70108683e-03  5.54907914e-04  4.27490735e-04  3.38019242e-05
  2.33452527e-05  7.92911445e-06  3.66516975e-14  9.47103243e-16
 -4.43799531e-16  4.01283668e-16 -4.05893396e-17]
tiendat@DELL:~/Documents/Git/DoAnMale/CuoiKy/Src/BUS$
```

Hình 13: Eigen Values của PCA.

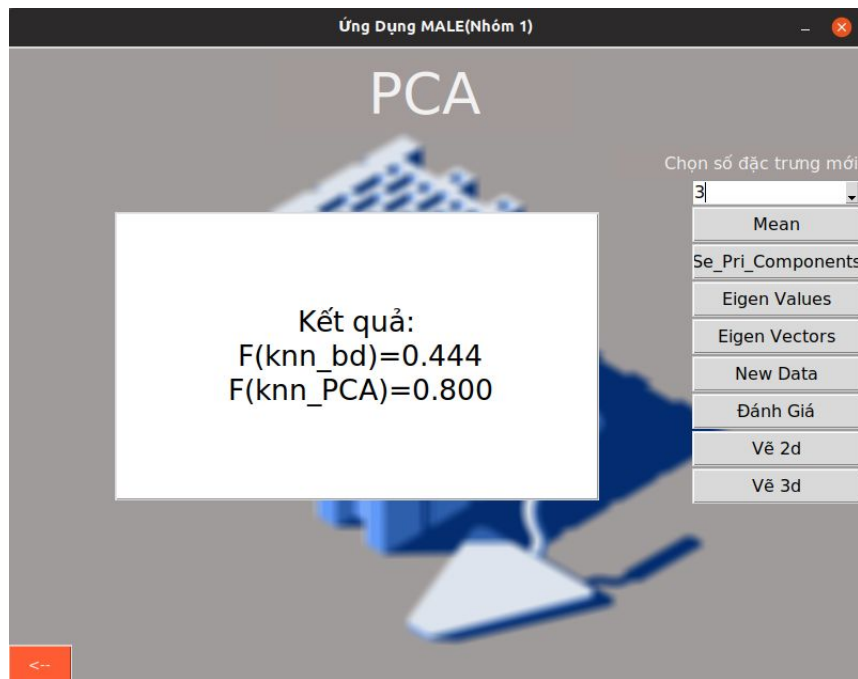


```
tiendat@DELL: ~/Documents/Git/DoAnMale/CuoiKy/Src
Eigen_Vectors
[-0.42961383 -0.12309475 -0.68915611 -0.55533964 -0.47850188 -0.09975351
-0.13545554 -0.29056413 -0.09447168 0.28557577 -0.57314096 -1.32858713
-0.28019584 0.02268305 1.06723179 -0.08770939 0.17038855 0.03873446
0.76242086 -0.7937378 1.36271812 -0.05882757 -0.1823814 -0.08993134
-0.709888 -0.05882706 -0.18254927 0.84279379 -0.10303858 -0.10085961
-0.32558425 0.01783781 -0.15861208 -0.04065884 -0.62311759 0.58358818
-0.28608385 -0.80174994 -0.13115917]
-8.971499188890155e-18
[[ 1.66461451e-01 1.16107168e-01 -3.63100364e-02 ... 3.94133924e-09
-9.02264889e-10 3.93696626e-10]
[ 1.94816701e-01 -1.04521055e-01 1.02224672e-02 ... 5.67315686e-08
-1.29951415e-08 5.66995344e-09]
[ 1.35959500e-01 1.34672827e-01 -2.17913423e-01 ... 1.40661021e-09
-3.21987535e-10 1.40462706e-10]
...
[ 1.97178981e-01 5.17940488e-02 -5.99906827e-03 ... -1.78303395e-08
4.08190677e-09 -1.78102817e-09]
[ 5.90326113e-02 3.00460120e-01 -2.06162206e-02 ... -7.88848145e-10
1.80512286e-10 -7.87997477e-11]
[ 1.98488301e-01 1.38974812e-02 2.91869466e-03 ... 6.23083194e-02
-4.33370123e-01 3.52643407e-01]]
```

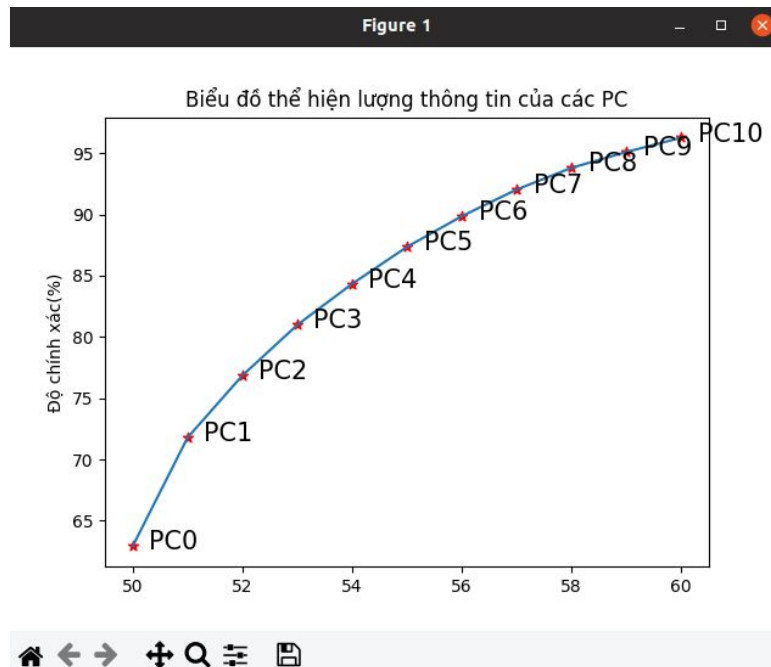
Hình 14: Eigen Vectors của PCA.

```
tiendat@DELL: ~/Documents/Git/DoAnMale/CuoiKy/Src
[-6.06459261e-01 -1.23962184e+00 8.99029470e-01]
[-2.61612072e+00 1.36521655e+00 3.68473195e-01]
[-2.86693173e+00 9.01697129e-01 -2.33857445e-01]
[-8.54143424e-01 -2.77154328e+00 -2.34373116e+00]
[ 1.57434042e-01 -4.42581822e-01 -2.28597950e+00]
[-1.41477997e+00 5.89743878e-01 1.55604118e-01]
[-3.50075369e+00 1.00041696e+00 -6.00098707e-01]
[ 1.11082258e+00 4.80709824e+00 -1.02548733e-01]
[-2.87526218e+00 1.13354498e+00 -5.27114962e-01]
[ 2.99136015e+00 4.68193182e+00 1.00801223e+00]
[-3.66428270e+00 -8.36216864e-01 3.80260569e-01]
[ 3.62039056e+00 2.70687501e+00 1.72215058e+00]
[-1.62032087e+00 -1.19681669e+00 1.23438692e+00]
[ 5.43270510e+00 -1.94595968e+00 -3.20282306e+00]
[-1.35255345e+00 8.50404600e-01 3.72765550e-01]
[-2.32280982e+00 9.66171414e-01 8.30409424e-01]
[ 1.69857763e+00 4.12008950e+00 7.85514067e-01]
[-1.99595432e+00 3.00629157e+00 -1.87166595e+00]
[-4.45804271e+00 3.58288507e-01 -2.31832876e+00]
[-2.54795877e+00 8.29736645e-01 3.06431359e-01]
[-3.18839559e+00 -1.78772810e+00 2.56176784e+00]
[-3.76344651e+00 -1.91426850e+00 1.39645958e+00]
[-1.18276033e+00 -9.43118919e-01 8.56221464e-01]
[-3.80321802e+00 1.41530918e-01 -5.51725207e-01]
```

Hình 15: Dữ liệu sau biến đổi PCA và chọn 3 pc.

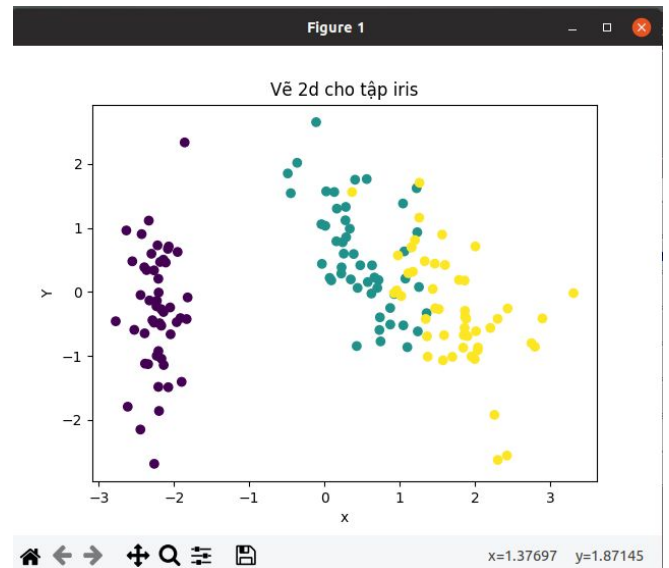
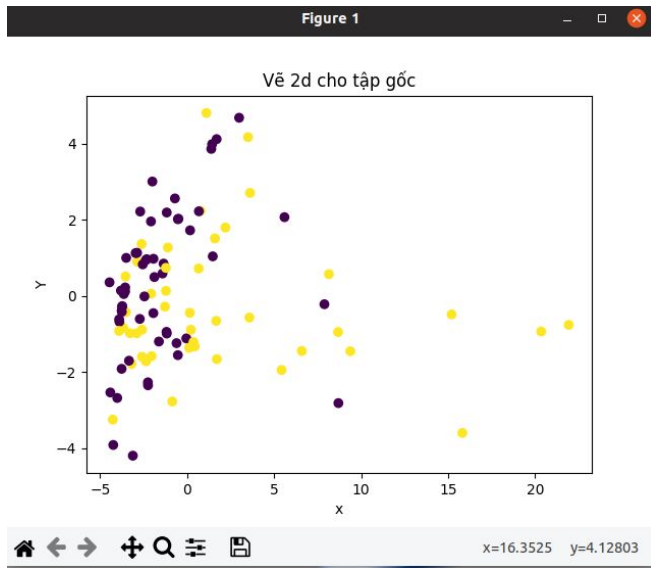


Hình 16: So sánh dữ liệu trước sau biến đổi PCA dùng KNN.

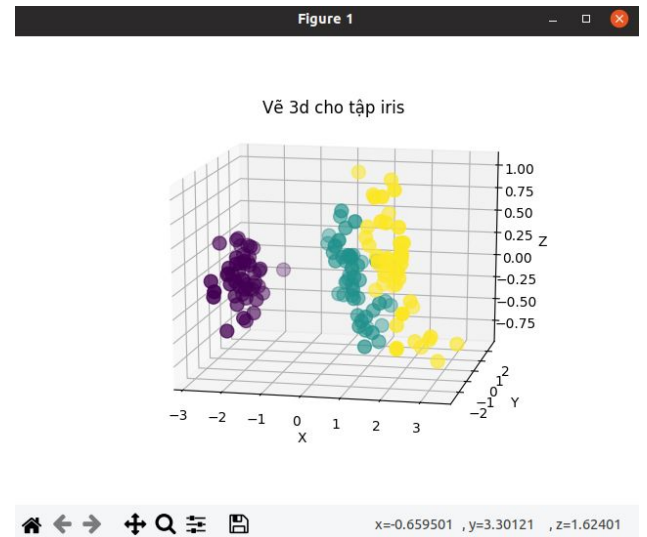
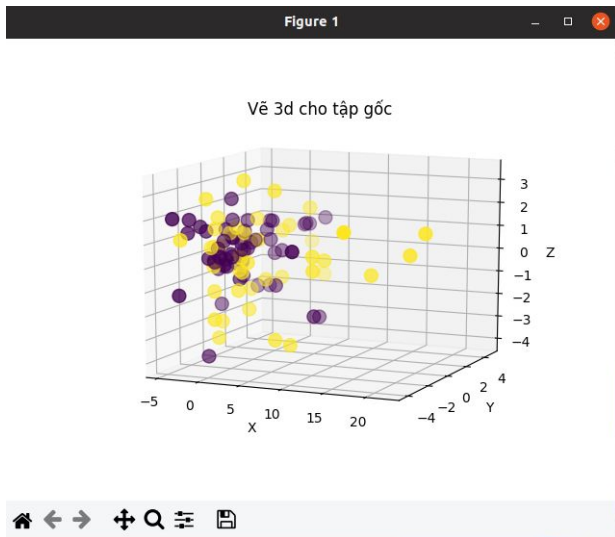


Hình 17: Lượng thông tin dữ liệu sau biến đổi PCA còn giữ lại.





Hình 18: Vẽ 2d tập gốc và tập iris khi qua PCA.



Hình 19: Vẽ 3d tập gốc và tập iris khi qua PCA.

Từ hình 18, 19 cho thấy tập dữ liệu vẫn chưa phân tách mạnh khi qua PCA như iris.

### III. LDA

```
tiendat@DELL: ~/Documents/Git/DoAnMale/CuoiKy/Src/BUS
tiendat@DELL:~/Documents/Git/DoAnMale/CuoiKy/Src/BUS$ python3 LDA.py
mean cho class 0
[ 4.03717464e-18  1.21115239e-17  2.11951668e-17  1.02947953e-16
 1.61486985e-17 -8.88178420e-17  3.63345717e-17  3.55271368e-16
-7.26691434e-17  4.78405194e-16  6.25762068e-17  2.22044605e-17
-1.61486985e-17  2.82602224e-17  2.82602224e-17 -6.45947942e-17
 4.44089210e-17  4.03717464e-17 -1.45338287e-16 -9.28550166e-17
-6.45947942e-17 -8.27620800e-17  6.98431212e-16  1.21115239e-17
 2.36174716e-16 -7.26691434e-17 -8.27620800e-17  1.61486985e-17
 1.61486985e-17 -8.88178420e-17  1.31208176e-16  3.43159844e-16
 2.82602224e-17  7.26691434e-17  1.13040890e-16  1.17078064e-16
-3.83531590e-17 -4.66293670e-16  2.22044605e-17]
mean cho class 1
[-1.76626390e-17 -1.51394049e-17  1.00613962e-16  5.04646829e-17
-7.31737903e-17 -1.00929366e-17  2.77555756e-17  9.65137061e-17
 8.70515781e-17 -1.26161707e-16  2.52323415e-17  1.94289029e-16
-2.27091073e-17 -4.03717464e-17  1.00929366e-17 -1.76626390e-17
 5.04646829e-18  5.67727683e-17  2.30875924e-16 -5.80343854e-17
-1.17961196e-16  3.78485122e-18  1.64010220e-16  1.00929366e-17
-2.39707244e-16 -2.14474902e-17 -8.70515781e-17 -5.65204449e-16
 4.79414488e-17  1.26161707e-17  2.39707244e-17  9.08364293e-17
 1.03452600e-16  3.02788098e-17 -6.56040878e-17  1.00929366e-16
 3.02788098e-17 -2.01858732e-16  3.78485122e-17]
tiendat@DELL:~/Documents/Git/DoAnMale/CuoiKy/Src/BUS$
```

Hình 20: Mean tương ứng với 2 lớp.

```
tiendat@DELL: ~/Documents/Git/DoAnMale/CuoiKy/Src/BUS
tiendat@DELL:~/Documents/Git/DoAnMale/CuoiKy/Src/BUS$ python3 LDA.py
Sw
[[ 99.          73.00457123  72.90629906 ...  91.47382715  48.24726713
 80.22244853]
 [ 73.00457123  99.          63.57152043 ...  87.18368501  16.16734983
 88.9056591 ]
 [ 72.90629906  63.57152043  99.          ...  70.63768608  30.01026411
 62.56678138]
 ...
 [ 91.47382715  87.18368501  70.63768608 ...  99.          37.46133509
 95.66735547]
 [ 48.24726713  16.16734983  30.01026411 ...  37.46133509  99.
 23.87797116]
 [ 80.22244853  88.9056591  62.56678138 ...  95.66735547  23.87797116
 99.          ]]
Sb
8.735612972150694e-28
tiendat@DELL:~/Documents/Git/DoAnMale/CuoiKy/Src/BUS$
```

Hình 20: Sw và Sb tương ứng.

```
tiendat@DELL: ~/Documents/Git/DoAnMale/CuoiKy/Src/BUS
tiendat@DELL:~/Documents/Git/DoAnMale/CuoiKy/Src/BUS$ python3 LDA.py
[ 5.03755454e-28  1.72973771e-28 -3.52487966e-29 ... -1.80287980e-27
 -1.01644492e-29  3.30775130e-27]
[ 1.72981380e-28  3.57006355e-26 -2.63957352e-28 ... -9.56906326e-28
 4.81346708e-30 -2.00666014e-27]
[ -3.52490725e-29 -2.63956828e-28  7.71545730e-29 ...  1.24411519e-28
 1.50577317e-29  6.67307278e-28]
...
[ -1.80288033e-27 -9.56876854e-28  1.24410487e-28 ...  7.91520945e-27
 1.28855280e-28 -1.02122223e-26]
[ -1.01647052e-29  4.81409107e-30  1.50577267e-29 ...  1.28856228e-28
 4.48526726e-29 -1.27315573e-28]
[ 3.30773948e-27 -2.00674246e-27  6.67311076e-28 ... -1.02121708e-26
 -1.27311833e-28  9.35456198e-26]]
tiendat@DELL:~/Documents/Git/DoAnMale/CuoiKy/Src/BUS$
```

Hình 21: Hàm mục tiêu  $(S_w^{-1}) * S_b$  tương ứng.

```
1.91636906e-17 1.75704542e-20 1.86253404e-24 2.58873178e-23
2.46804360e-25 1.42945900e-25 1.00127888e-25 5.00929310e-26
7.40597591e-27 4.75745824e-27 2.08721214e-27 1.68752615e-27
8.48186329e-28 4.25030659e-28 3.39740869e-28 2.43325871e-28
1.96574362e-28 1.75197357e-28 1.35962630e-28 1.00471690e-28
7.48663748e-29 6.33975415e-29 3.69412229e-29 3.05067658e-29
2.74666674e-29 1.88460374e-29 1.40472292e-29 1.16251771e-29
8.90905725e-30 6.24629554e-30 1.61965005e-31 5.53801227e-30
9.58938513e-31 4.26355930e-30 1.67750959e-30 2.02305207e-30
3.40922188e-30 2.90527470e-30 3.06714815e-30]
```

Hình 22: Eigen Values



```

tiendat@DELL:~/Documents/Git/DoAnMale/Cuoiky/Src/BUS$ python3 LDA.py
[[ 7.67554287e-07  8.39645529e-05  5.32590998e-03 ... -4.88779769e-02
 -1.65822722e-01  1.27090106e-03]
 [ 5.75284003e-06 -1.77757720e-04  1.09433228e-03 ... -5.84648842e-02
 -5.28289696e-02  3.96942384e-02]
 [-2.61279558e-07  9.38859824e-07  5.22891324e-04 ...  1.74694177e-02
 4.58797001e-02 -8.20294254e-02]
 ...
 [-3.36976714e-06 -3.11089968e-04 -1.17354609e-02 ... -6.17290092e-02
 -5.09942774e-02  9.35683716e-03]
 [-2.56436280e-07 -7.39837070e-07 -2.28738794e-04 ... -9.50002367e-03
 -5.24321026e-01  1.56914308e-01]
 [-6.28036004e-07  1.20116927e-03  8.60287883e-02 ... -5.07328244e-02
 6.68002196e-02  8.02664997e-03]]
tiendat@DELL:~/Documents/Git/DoAnMale/Cuoiky/Src/BUS$ █

```

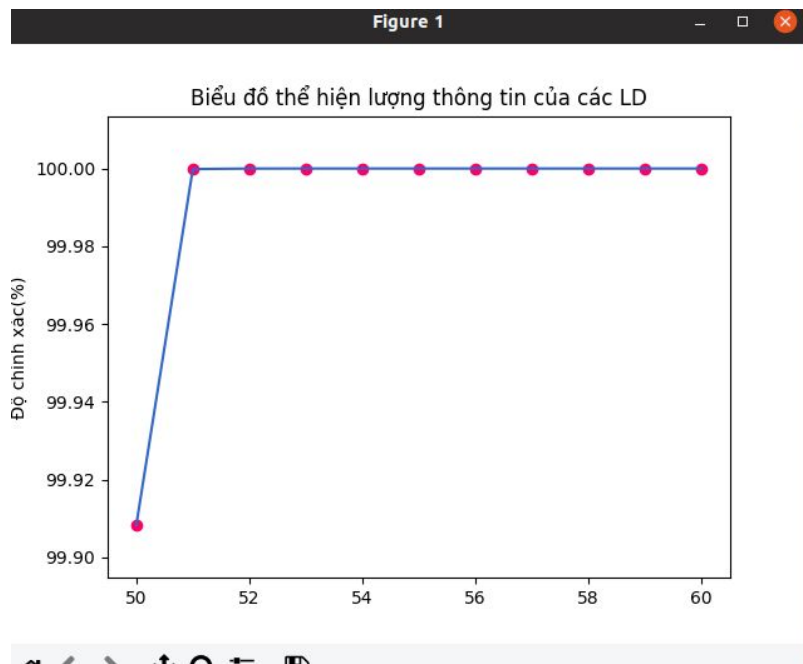
Hình 23: Eigen Vectors

```

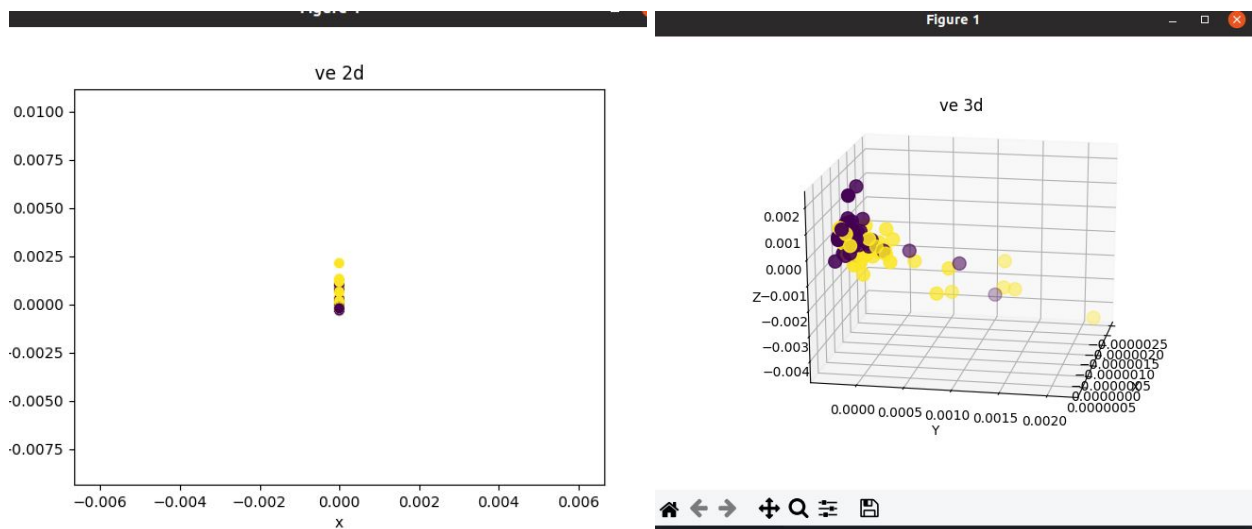
tiendat@DELL:~/Documents/Git/DoAnMale/Cuoiky/Src/BUS$ python3 LDA.py
[-2.10890878e-07 -4.78818126e-05 -5.81403333e-05]
[-1.29957889e-07 -1.71346236e-04  1.64668749e-04]
[ 3.17948523e-07 -4.15812827e-05 -1.56884300e-04]
[ 2.38359402e-07 -2.14989148e-04  2.68605676e-04]
[ 1.93075615e-07 -1.91762670e-04  9.91401473e-04]
[-2.11605594e-07 -1.09278863e-06 -6.19357204e-04]
[ 5.37086625e-07 -2.75100979e-06 -3.03799889e-04]
[ 9.40289854e-08 -5.16826922e-05  3.97304548e-06]
[-2.12480720e-07 -2.16543390e-04  7.21696015e-04]
[ 2.76616949e-07 -1.30915931e-04  7.29935503e-04]
[-5.90577953e-08 -1.82479758e-04 -8.54329117e-05]
[-6.97881090e-08 -1.72365835e-04  3.08978012e-04]
[ 5.63575298e-07 -2.00083747e-04  8.00345314e-04]
[-2.06745922e-07  6.95586941e-05 -5.05078695e-05]
[ 2.61978949e-07 -1.02496012e-04 -2.33070646e-04]
[-6.04965175e-07  3.83460544e-04 -1.02831385e-03]
[-1.21068954e-07 -7.53541546e-05 -1.60558078e-04]
[ 2.90578353e-07 -1.96254033e-04  3.40035840e-04]
[ 5.69254773e-08 -1.60030071e-04  5.81996822e-04]
[ 3.23580261e-07 -2.40942046e-04  7.98925549e-04]
[ 4.76663702e-07 -1.84114150e-04  8.00750839e-04]
[ 1.34023912e-08 -1.79921698e-04  5.95140613e-04]
[ 3.33234817e-08 -1.55759256e-04 -2.99814199e-04]

```

Hình 24: Dữ liệu mới sau khi biến đổi qua LDA ở đây chọn 3 cột.



Hình 25: Lượng thông tin dữ liệu sau biến đổi LDA còn giữ lại.

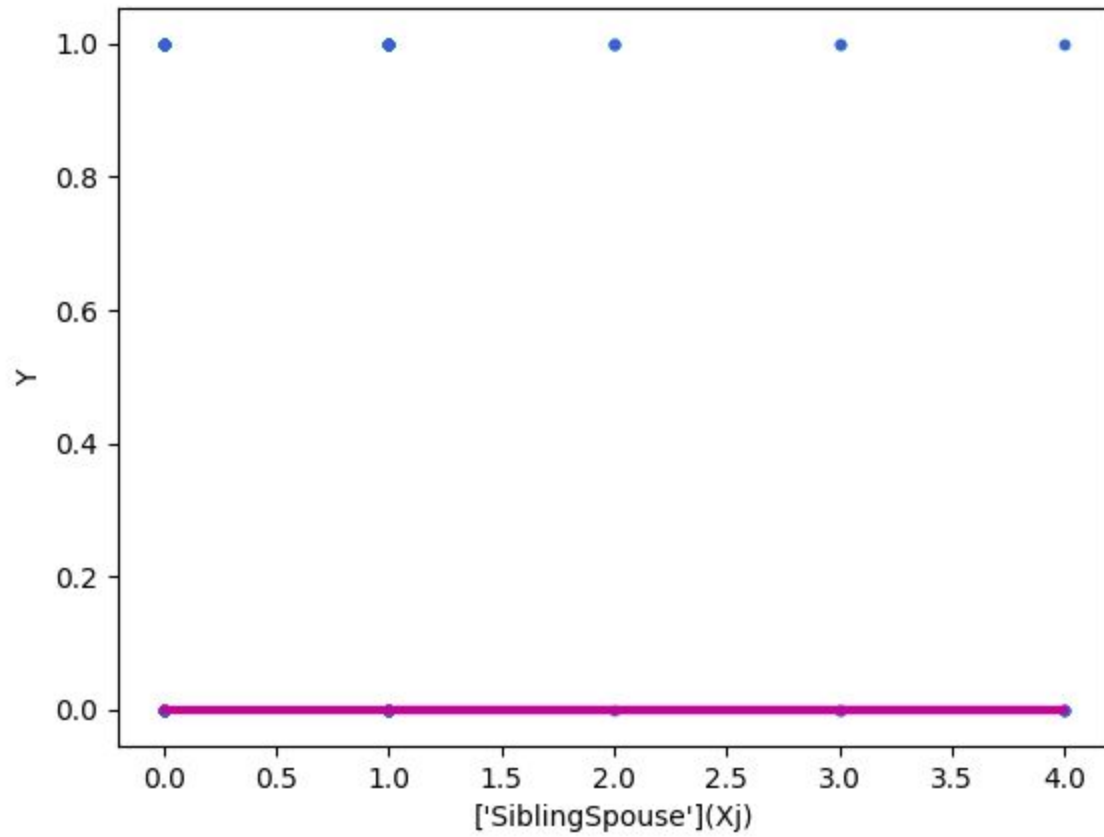


Hình 26: Vẽ 2d và 3d LDA.

```
tiendat@DELL:~/Documents/Git/DoAnMale/Cuoiky/Src/BUS$ python3 LDA.py  
knn ban dau  
0.2857142857142857  
knn qua LDA  
0.6666666666666666  
tiendat@DELL:~/Documents/Git/DoAnMale/Cuoiky/Src/BUS$
```

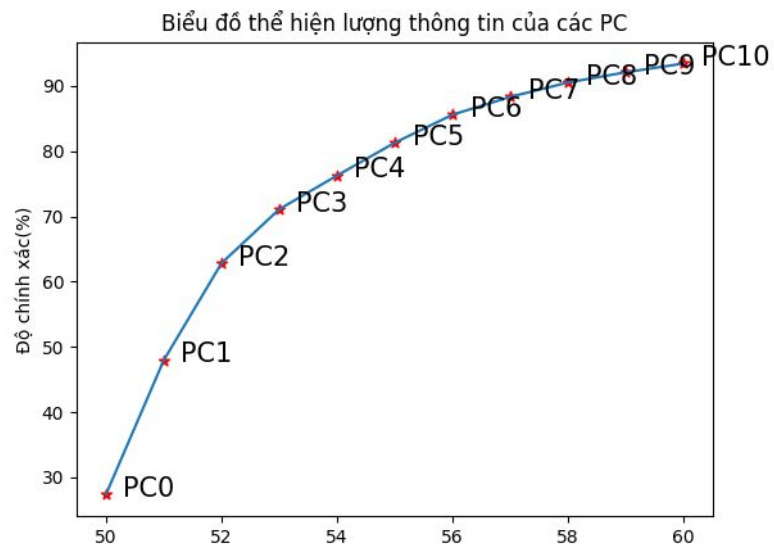
Hình 26: So sánh tập dữ liệu ban đầu và tập qua biến đổi LDA với KNN.

#### *IV. Logistic regression*



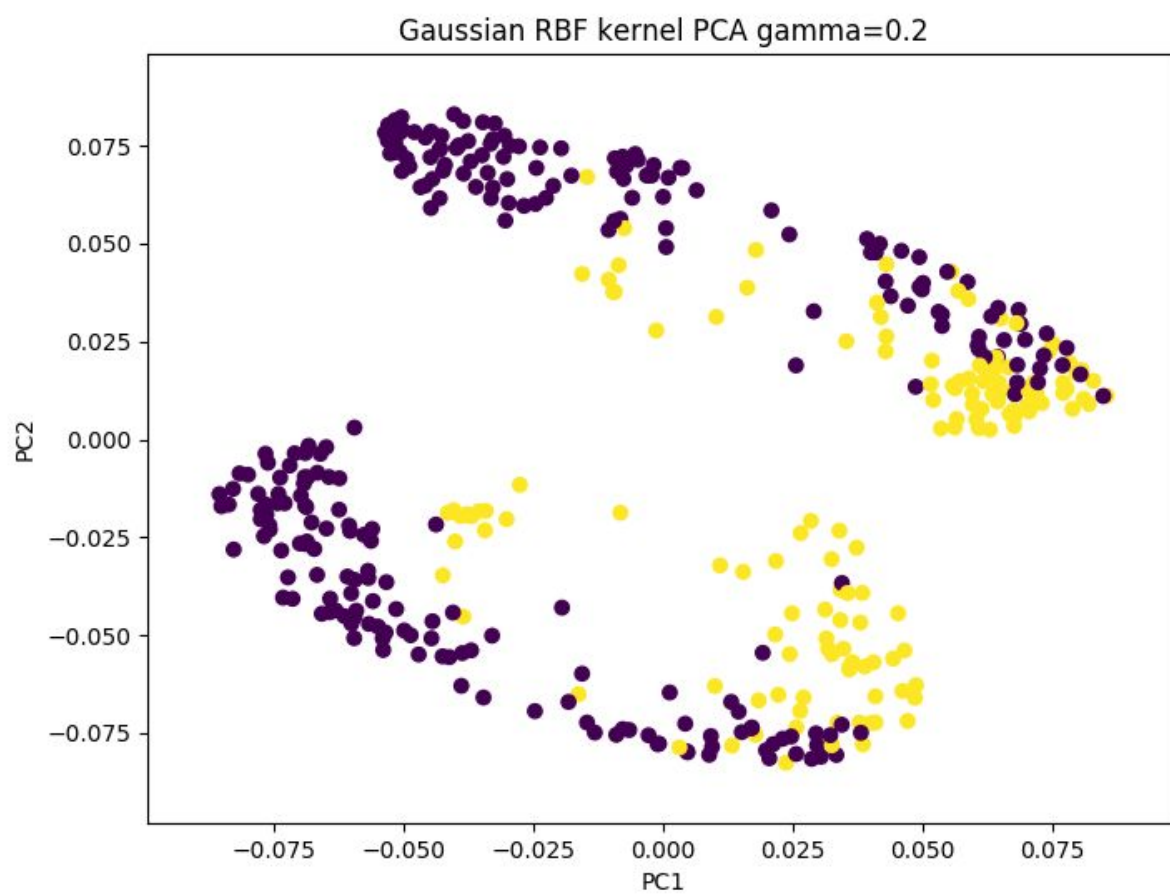
Hình 27: Hình biểu diễn Logistic\_Regression.

## V. KPCA



Hình 28: Lượng thông tin dữ liệu sau biến đổi KPCA còn giữ lại.





Hình 29: Vẽ 2d tập gốc khi qua KPCA.

The image shows a Kali Linux desktop environment with a terminal window open. The terminal has a dark background and white text. At the top of the terminal, there's a title bar with "T4 12:18 PM" and some system icons. Below the title bar, the prompt is "root@kali: ~/DoAnMale/Cuoiky/Src/BUS".

The first session shows the user navigating to the PyCharm community edition bin directory and running the script. The output includes several warnings from OpenJDK and IntelliJ IDEA.

```
root@kali:~/DoAnMale/Cuoiky/Src/BUS# cd /opt/pycharm-community-2019.1.3/bin/
root@kali:/opt/pycharm-community-2019.1.3/bin# ./pycharm.sh
OpenJDK 64-Bit Server VM warning: Option UseConcMarkSweepGC was deprecated in version 9.0 and will likely be removed in a future release.
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by com.intellij.ide.ClassUtilCore to field sun.net.www.protocol.jar.JarFileFactory.fileCache
WARNING: Please consider reporting this to the maintainers of com.intellij.ide.ClassUtilCore
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
```

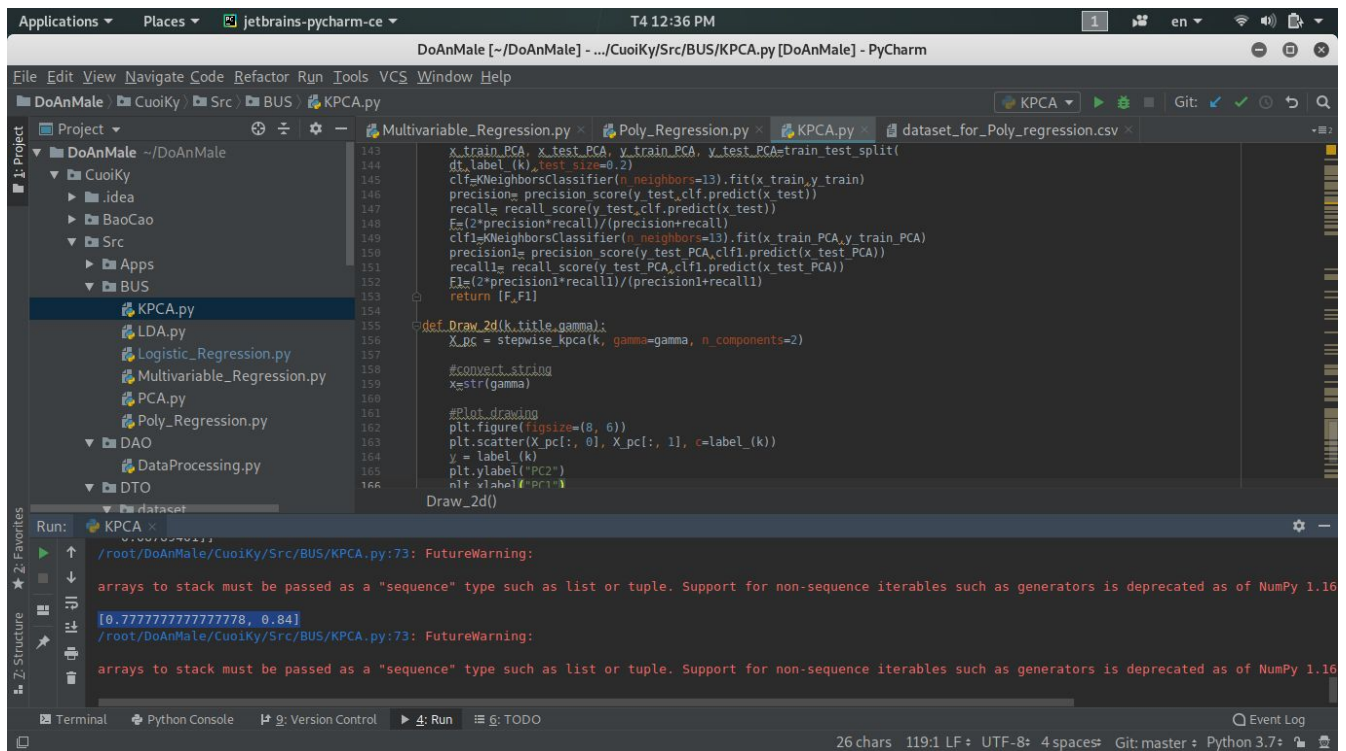
The second session shows the user running a script that outputs Eigen Vectors. The output is a matrix of floating-point numbers.

```
root@kali:~/DoAnMale/Cuoiky/Src/BUS 122x14
Eigen Vectors: [[-0.04999884 -0.00018289 -0.00012456 ... 0.00741021 0.00302032
-0.0595433 ]
[-0.04999157 -0.00092972 -0.00044955 ... 0.02786339 -0.00993587
-0.06245148]
[-0.04999744 -0.00021297 -0.00021802 ... 0.01846781 0.07301494
-0.05263348]
...
[-0.04999765 -0.00024256 -0.0001687 ... 0.07557559 0.00521734
0.05653333]
[-0.04999897 -0.00022923 -0.00016254 ... 0.05607414 -0.05448162
0.01914717]
[-0.05001089 -0.00059667 -0.0001194 ... 0.07443946 0.0058887
0.06769401]]
KPCA py: 73: FutureWarning:
```

Hình 30: Giá trị Eigen Vectors

[illegible]

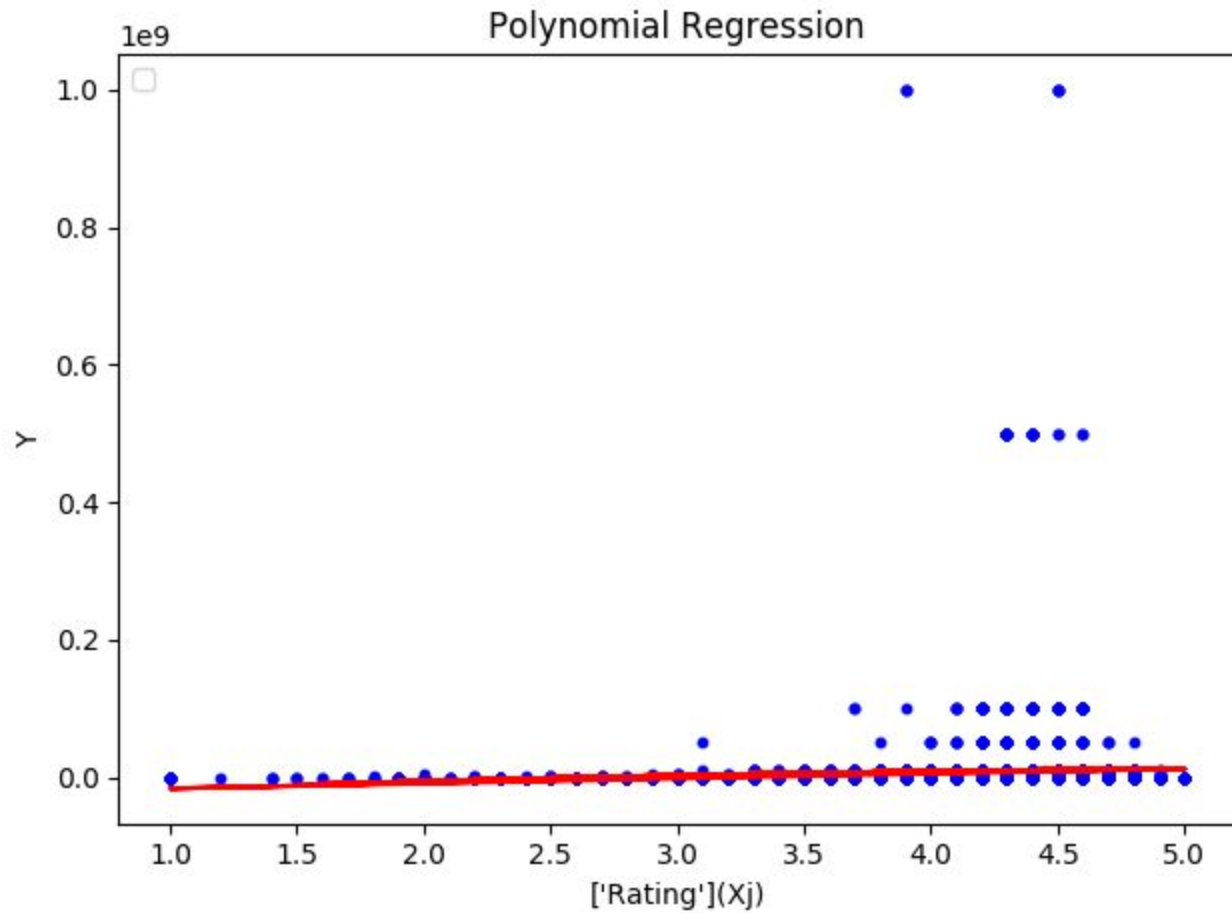
Hình 31: Giá trị Eigen Values



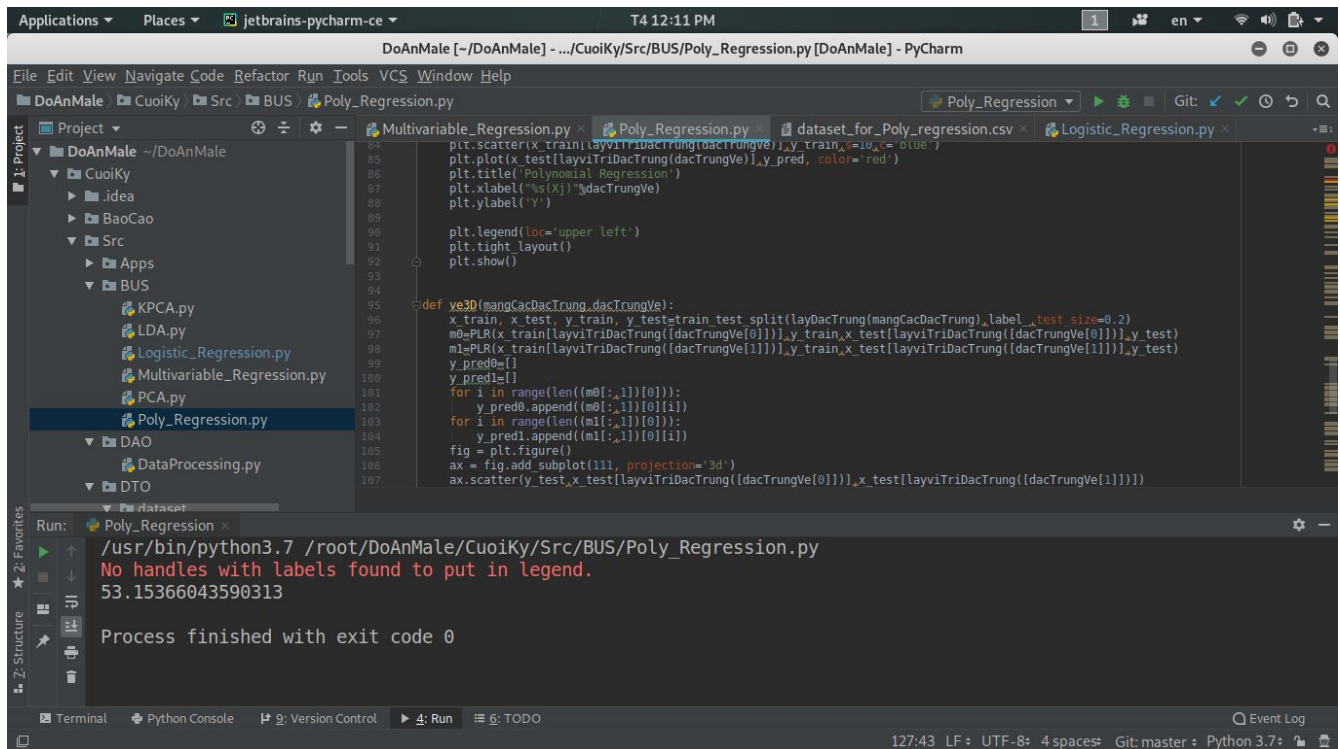
Hình 32: So sánh tập dữ liệu ban đầu và tập qua biến đổi KPCA với KNN.

## VI. *Polynomial Regression*

Ở đây ta chọn biến độc lập là 'installa'



Hình 33: Biểu diễn Polynomial Regression với bậc bằng 2



Hình 34: Kết quả sử dụng Polynomial Regression với bậc bằng 2

## PHẦN 4: TÀI LIỆU THAM KHẢO

<https://plot.ly/ipython-notebooks/principal-component-analysis/>

[https://scikit-learn.org/stable/modules/generated/sklearn.linear\\_model.LinearRegression.html](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html)

<https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.KernelPCA.html>

<https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PolynomialFeatures.html>

<https://scikit-learn.org/stable/modules/svm.html>

[https://scikit-learn.org/stable/modules/generated/sklearn.linear\\_model.LogisticRegression.html](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html)

<https://docs.python.org/2/library/tkinter.html>

[https://sebastianraschka.com/Articles/2014\\_python\\_lda.html](https://sebastianraschka.com/Articles/2014_python_lda.html)