**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐH SƯ PHẠM KỸ THUẬT TP.HCM**

**KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**

****

**BÁO CÁO GIỮA KÌ MACHINE LEARNING**

**ĐỀ TÀI: CLUSTERING VÀ CLASSIFICATION**

**Giáo viên hướng dẫn: TS.Trần Lê Minh Sang**

**Sinh viên thực hiện: - Phan Nguyễn Tấn Tài --16110203**

**- Lê Duy Quí – 16110190**

***TP.HỒ CHÍ MINH, THÁNG 11 NĂM 2019***

**MỤC LỤC**

[I-LỜI CAM KẾT 4](#_Toc24036662)

[1.Tìm hiểu về đạo văn, đạo code 4](#_Toc24036663)

[2.Cách phòng tránh đạo văn, đạo code 4](#_Toc24036664)

[3.Lời cam kết 4](#_Toc24036665)

[II-TÌM HIỂU THƯ VIỆN SCIKIT-LEARN CHO GIẢI THUẬT HIERARCHICAL CLUSTERING 4](#_Toc24036666)

[1.Giới thiệu về Hierarchical Clustering 4](#_Toc24036667)

[2. Giới thiệu những thông số giải thuật của thư viện 5](#_Toc24036668)

[3. Giới thiệu những tiêu chí liên kết để xác định số liệu của thư viện 6](#_Toc24036669)

[a.WARD *(nợ cô)* 6](#_Toc24036670)

[b.COMPLETE 7](#_Toc24036671)

[c.AVERAGE 7](#_Toc24036672)

[d.SIGLE 7](#_Toc24036673)

[e.CENTROID 7](#_Toc24036674)

[III-DEMO THUẬT TOÁN VỀ PHÂN LOẠI (CLASSIFICATION) 8](#_Toc24036675)

[1.XỬ LÍ DỮ LIỆU CHUNG 8](#_Toc24036676)

[a.Đọc dữ liệu 8](#_Toc24036677)

[b.Xử lí dữ liệu từ chuỗi sang số 8](#_Toc24036678)

[c.Train Slit Data 9](#_Toc24036679)

[2.NAIVE BAYES 9](#_Toc24036680)

[a.GaussianNB 9](#_Toc24036681)

[b.MultinomialNB 10](#_Toc24036682)

[c.BernoulliNB 11](#_Toc24036683)

[d.ComplementNB 11](#_Toc24036684)

[d. GaussianNB với CrossValidation 12](#_Toc24036685)

[e.Biểu diễn các điểm dữ liệu 12](#_Toc24036686)

[3.KNN 14](#_Toc24036687)

[a. Chạy với dữ liệu bình thường (không sử dụng tham số của KNN) 14](#_Toc24036688)

[b. Chạy với tham số của KNN 14](#_Toc24036689)

[c.KNN với Cross Validation với algorithm='kd\_tree' 17](#_Toc24036690)

[d. Biểu diễn các điểm dữ liệu 17](#_Toc24036691)

[4.SVM 17](#_Toc24036692)

[a.Mã nguồn 17](#_Toc24036693)

[b.SVM với Cross Validation 18](#_Toc24036694)

[c. Biểu diễn các điểm dữ liệu 18](#_Toc24036695)

[5.SVM KERNEL 19](#_Toc24036696)

[a. Polynomial 19](#_Toc24036697)

[b. RBF(Radial Basis Function) 19](#_Toc24036698)

[c. Sigmoid 20](#_Toc24036699)

[d. Sigmoid với CrossVailidation 20](#_Toc24036700)

[e. Biểu diễn các điểm dữ liệu 21](#_Toc24036701)

[IV-DEMO THUẬT TOÁN VỀ PHÂN CỤM (CLUSTERING) 21](#_Toc24036702)

[1.XỬ LÍ DỮ LIỆU CHUNG 21](#_Toc24036703)

[a.Đọc dữ liệu 21](#_Toc24036704)

[b.Chuyển kiểu dữ liệu chữ sang label số: 21](#_Toc24036705)

[2. HIERARCHICAL CLUSTERING: 22](#_Toc24036706)

[a.Vẽ cấu trúc cây Dendrograms 22](#_Toc24036707)

[b.Vẽ mô hình phân cụm 2D 23](#_Toc24036708)

[c.Vẽ mô hình phân cụm 3D 24](#_Toc24036709)

[3.K-MEANS 25](#_Toc24036710)

[a. Tìm số cụm tối ưu để phân loại K-means 25](#_Toc24036711)

[c. Tạo một K-means classifier với số cluster là 6 25](#_Toc24036712)

[d. Hiển thị các cụm 2D 25](#_Toc24036713)

[e. Hiển thị các cụm 3D 27](#_Toc24036714)

[V-TÀI LIỆU THAM KHẢO 29](#_Toc24036715)

# I-LỜI CAM KẾT

## 1.Tìm hiểu về đạo văn, đạo code

-Đạo văn được hiểu là sử dụng công trình hay tác phầm của người khác, lấy ý tưởng của người khác, sao chép nguyên bản từ ngữ của người khác mà không ghi nguồn, sử dụng cấu trúc và cách lí giải của người khác mà không ghi nhận họ.

-Đạo văn được xem là hành vi thiếu trung thực về mặt học thuật và vi phạm đạo đức rất nghiêm trọng. Ở mức độ sinh viên, đạo văn sẽ khiến kết quả nghiên cứu bị hủy bỏ tùy vào mức độ nghiêm trọng của hành vi.

## 2.Cách phòng tránh đạo văn, đạo code

-Luôn trích dẫn nguồn khi sử dụng câu văn, dữ liệu thống kê.

-Thay đổi cấu trúc câu văn.

-Dùng từ đồng nghĩa.

-Thay đổi cách dạng câu văn.

-Trường hợp không cần trích dẫn nguồn như: Lý luận, tư tưởng, thông tin của chính tác giả hoặc một thông tin phổ biến và được nhiều người biết.

## 3.Lời cam kết

Chúng em (Nhóm 6) xin cam kết với cô (TS. Trần Lê Minh Sang) sẽ chỉ tham khảo và không đạo văn, đạo code trong quá trình thực hiện báo cáo giữa kì. Nếu có vi phạm, chúng em sẽ chịu trách nhiệm và chịu mọi xử lý theo yêu cầu của cô đưa ra.

# II-TÌM HIỂU THƯ VIỆN SCIKIT-LEARN CHO GIẢI THUẬT HIERARCHICAL CLUSTERING

## 1.Giới thiệu về Hierarchical Clustering

-Trong kỹ thuật phân nhóm theo thứ bậc (Hierarchical Clustering Techniques), có 2 phương pháp chính đó là:

+Agglomerative Aapproach (bottom up approach): Ban đầu, chúng ta xem mỗi đối tượng là 1 nhóm (cluster) và nhóm 2 đối tượng gần nhất thành 1 cluster. Quá trình này lặp lại cho đến khi tất cả các đối tượng được nhóm vào 1 cluster cuối cùng.

+Divisive Approach (top down approach): Quá trình ngược lại với Agglomerative Approach, ban đầu chúng ta xem tất cả các đối tượng thuộc cùng 1 cluster, sau đó tiến hành phân thành 2 nhóm con (thường dựa vào khoảng cách lớn nhất). Quá trình này được thực hiện cho đến khi mỗi nhóm chỉ còn 1 đối tượng.

-Trong phần này sẽ giới thiệu kỹ thuật phân nhóm Agglomerative Approach.

***Các bước trong kỹ thuật phân cụm Agglomerative Approach như sau:***

1.Chuyển đổi các đặc trưng (thuộc tính - Features) của đối tượng (objects) vào ma trận khoảng cách

2.Xem mỗi đối tượng là một cluster (chẳn hạn, nếu ta có 4 đối tượng, ban đầu chúng ta sẽ có 4 clusters)

3.Lặp lại 2 bước sau cho đến khi số cluster bằng 1

a.Gộp 2 cluster gần nhất

b.Cập nhật ma trận khoảng cách

## 2. Giới thiệu những thông số giải thuật của thư viện

-n\_clusters\_ : int **Số lượng các cụm để tìm. Nó sẽ phải là None nếu distance\_threshod là not None**

-affinity : string or callable, default: “euclidean” **Số liệu được sử dụng để tính toán các linkage (các liên kết). Có thể dùng “euclidean”, “l1”, “l2”, “manhattan”, “cosine” hoặc “precomputed”. Nếu linkage là “ward” thì chỉ sử dụng “eucline” mới được chấp nhận.**

-memory :None, str or object with the joblib **Được sử dụng để lưu trữ đầu ra tính toán của cây. Theo mặc định, không có bộ nhớ đệm được thực hiện. Nếu một chuỗi được đưa ra, đó là đường dẫn đến thư mục bộ đệm.**

-connectivity : array-like or callable, optional **Xác định cho từng mẫu các mẫu lân cận theo cấu trúc dữ liệu đã cho.**

-compute\_full\_tree : bool or ‘auto’ (optional) **Dừng sớm việc xây dựng cây tại n\_cluster. Điều này rất hữu ích để giảm thời gian tính toán nếu số lượng custers không nhỏ so với số lượng mẫu.**

-linkage : {“ward”, “complete”, “average”, “single”}, optional (default=”ward”). **Tiêu chí liên kết được sử dụng trong thuật toán. Tiêu chí liên kết xác định khoảng cách sử dụng giữa các bộ quan sát.**

-distance\_threshold : float, optional (default=None) **Ngưỡng khoảng cách liên kết ở trên đó, nơi mà các cụm sẽ không được hợp nhất.**

## 3. Giới thiệu những tiêu chí liên kết để xác định số liệu của thư viện

### a.WARD *(nợ cô)*

- Phương pháp phương sai tối thiểu của Ward là trường hợp đặc biệt của phương pháp hàm mục tiêu ban đầu. Ward đã đề xuất một quy trình phân cụm phân cấp tổng hợp chung, trong đó tiêu chí chọn cặp cụm để hợp nhất ở mỗi bước dựa trên giá trị tối ưu của hàm mục tiêu.

- Để minh họa, Ward đã sử dụng ví dụ trong đó hàm mục tiêu là tổng sai số của bình phương và ví dụ này được gọi là phương thức của Ward hoặc chính xác hơn là phương pháp phương sai tối thiểu của Ward.

- Tiêu chí phương sai tối thiểu: Để thực hiện phương pháp này, ở mỗi bước, hãy tìm cặp cụm dẫn đến tăng tối thiểu tổng phương sai trong cụm sau khi hợp nhất. Ở bước đầu, tất cả các cụm là singletons (cụm chứa một điểm duy nhất). Để áp dụng thuật toán đệ quy theo hàm mục tiêu này, khoảng cách ban đầu giữa các đối tượng riêng lẻ phải là (tỷ lệ với) khoảng cách Euclide bình phương.

Khoảng cách cụm ban đầu trong phương pháp phương sai tối thiểu của Ward được xác định là khoảng cách Euclide bình phương giữa các điểm:

-Phương pháp phương sai tối thiểu của Ward có thể được xác định và thực hiện đệ quy bằng thuật toán Lance siêu Williams.

*Giả sử rằng các cụm và đã được hợp nhất. Công thức đệ quy cung cấp khoảng cách cụm được cập nhật sau khi hợp nhất các cụm và .*

*-Biết:*

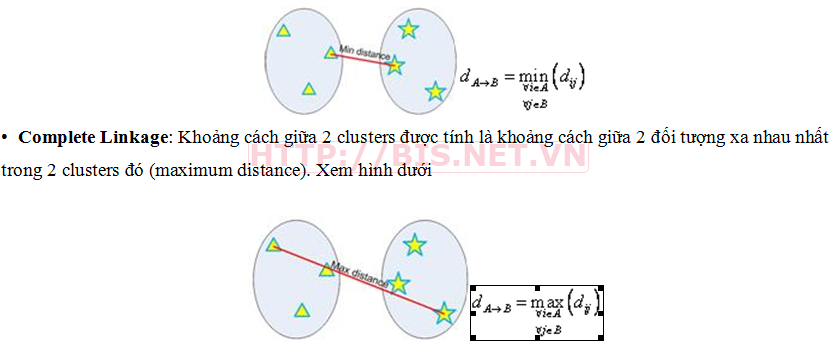
*+ ,, lần lượt là khoảng cách giữa các cụm ,, tương ứng.*

*+ là khoảng cách giữa một cụm mới và*

*Một thuật toán thuộc họ Lance-Williams nếu khoảng cách cụm được cập nhật có thể được tính toán đệ quy theo:*

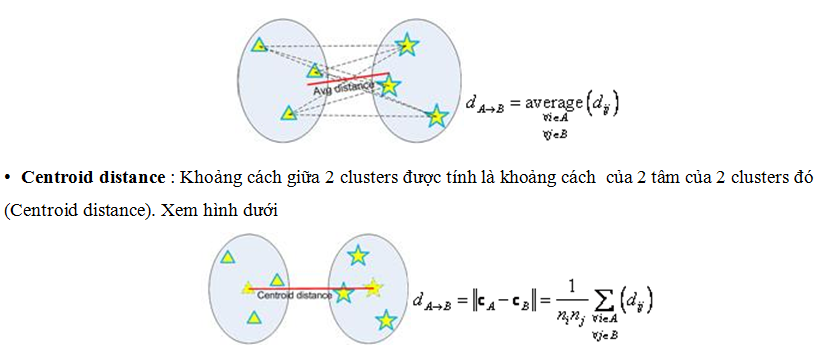
*=+++ ,với là những parameters (tham số), có thể phụ thuộc và size của cluster*

### b.COMPLETE

Khoảng cách giữa 2 cluster được tính là khoảng cách xa nhau nhất của 2 cluster đó

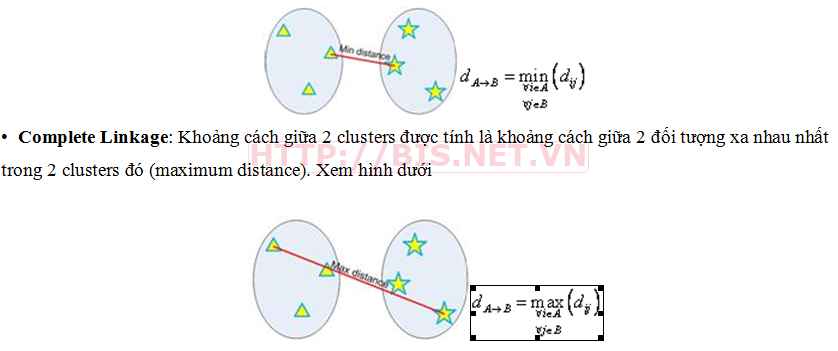
### c.AVERAGE

Khoảng cách giữa 2 clusters được tính là khoảng cách trung bình giữa các đối tượng trong 2 clusters đó (average distance).



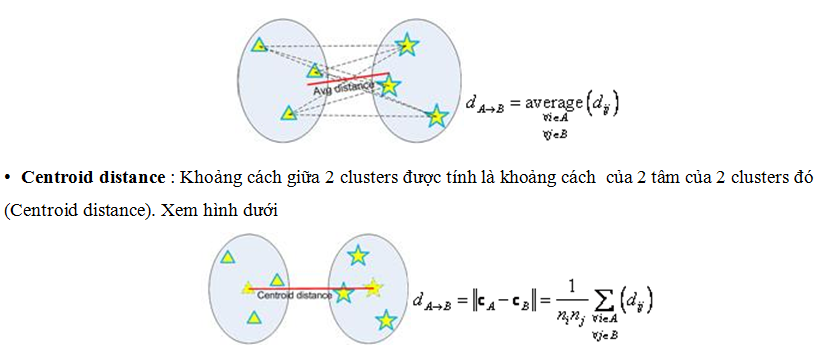
### d.SIGLE

Khoảng cách giữa 2 clusters được tính là khoảng cách giữa 2 đối tượng gần nhau nhất trong 2 clusters đó (minimum distance).



### e.CENTROID

Khoảng cách giữa 2 cluster được tính là khoảng cách của 2 tâm của 2 cluster đó (Centroid distance).



# III-DEMO THUẬT TOÁN VỀ PHÂN LOẠI (CLASSIFICATION)

## 1.XỬ LÍ DỮ LIỆU CHUNG

### a.Đọc dữ liệu

*import pandas as pd*

*DV=pd.read\_csv('dataset\_for\_classification\_cmc.csv', encoding='utf-8')*

(Dataset được chúng em đưa vào trong file code và mở vào trong Visual Code, khi get link để đọc file: Copy Relative Path)

### b.Xử lí dữ liệu từ chuỗi sang số

Với những thuật toán ở Classification thì dataset được xử lí phải ở dạng số, nếu trong dataset ta phát hiện dữ liệu ở dạng chuỗi thì phải xử lí dataset.

*from sklearn import preprocessing*

*from sklearn.preprocessing import LabelEncoder*

#thư việt scikit learn đã trình bày một transformer là LabelEncoder

*le =preprocessing.LabelEncoder()*

*(DV['education'])=le.fit\_transform(DV['education'])*

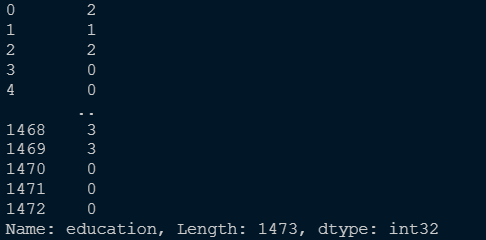
*(DV['spouse\_education'])=le.fit\_transform(DV['spouse\_education'])*

*(DV['religion'])=le.fit\_transform(DV['religion'])*

*(DV['use\_insurrance'])=le.fit\_transform(DV['use\_insurrance'])*

*(DV['now\_working'])=le.fit\_transform(DV['now\_working'])*

*(DV['spouse\_occupation'])=le.fit\_transform(DV['spouse\_occupation'])*

Sau khi transform thì kết quả sẽ được chạy (cột education):

### c.Train Slit Data

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

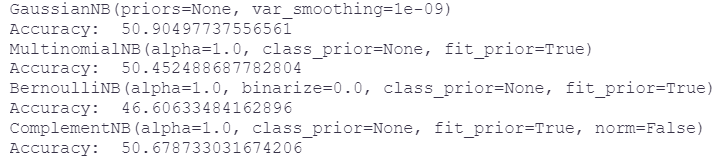
*X=DV.iloc[:,:-1].values* #Chọn tất cả feature trong dataset trừ cột cuối cùng

*Y=DV.iloc[:,9].values* #Chọn feature thứ 9 trong dataset (‘use\_insurrance’)

*X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X,Y,test\_size=0.3)* #70% training and 30% test

## 2.NAIVE BAYES

Thực hiện trên cả 4 công thức:



Nhận xét: Kết quả chạy được của nhóm em thì chưa được tốt (~50%), các công thức chỉ có độ chênh lệch từ 4 %

### a.GaussianNB

**Mã nguồn:**

*gaus=GaussianNB()*

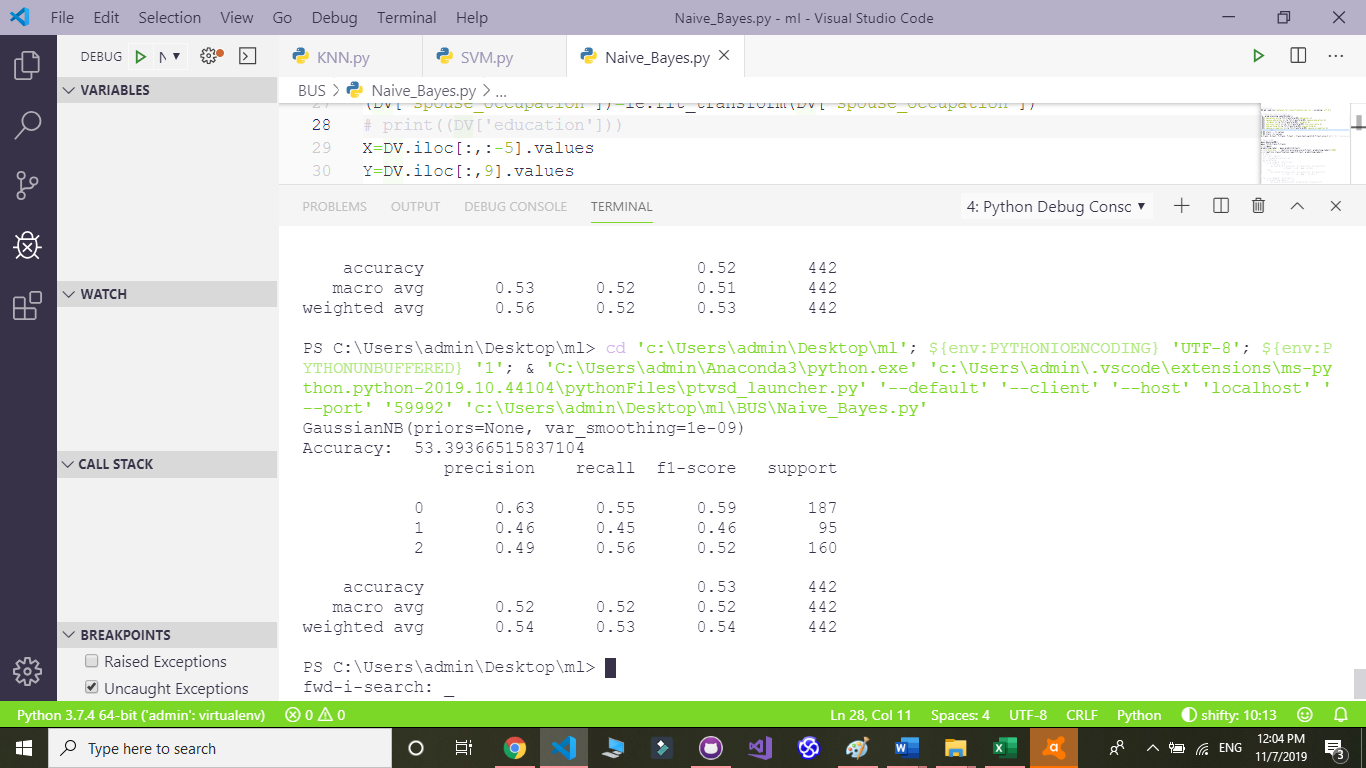
*gaus.fit(X\_train,Y\_train)*

*print(gaus)*

*predicting\_label = gaus.predict(X\_test)*

*print("Accuracy: ", (metrics.accuracy\_score(Y\_test, predicting\_label))\*100)*

Kết quả chạy:



Nhận xét: Độ chính xác ổn định nhất trong, nhãn 0 tỉ lệ đoán đúng ổn nhất trong các nhãn

### b.MultinomialNB

**Mã nguồn:**

*gaus=MultinomialNB()*

*gaus.fit(X\_train,Y\_train)*

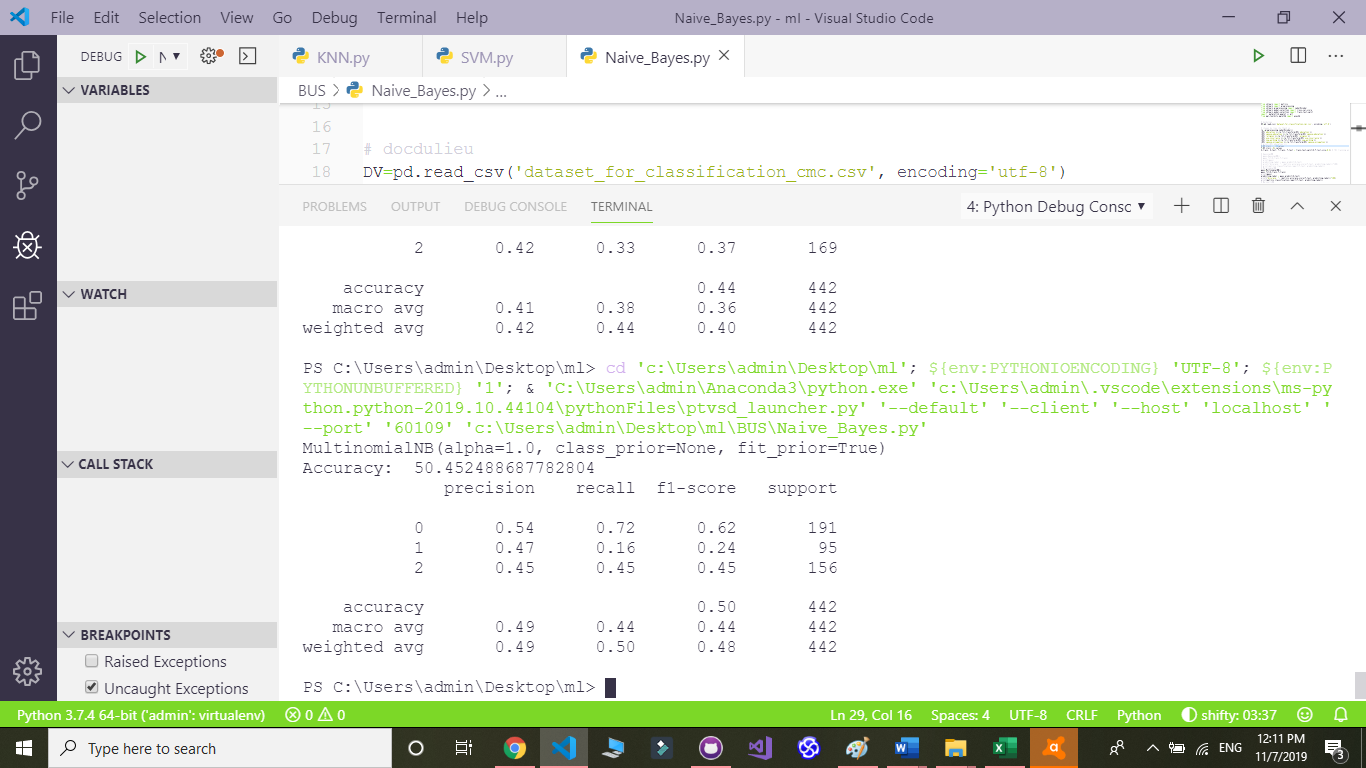
*print(gaus)*

*predicting\_label = gaus.predict(X\_test)*

*print("Accuracy: ", (metrics.accuracy\_score(Y\_test, predicting\_label))\*100)*

*print(metrics.classification\_report(Y\_test, predicting\_label))*

Kết quả chạy:



Nhận xét: Độ chính xác ~50%, với tỉ lệ này thì độ chính xác vẫn chưa cao. Cao nhất chỉ ở 52 %

### c.BernoulliNB

**Mã nguồn:**

*gaus=BernoulliNB()*

*gaus.fit(X\_train,Y\_train)*

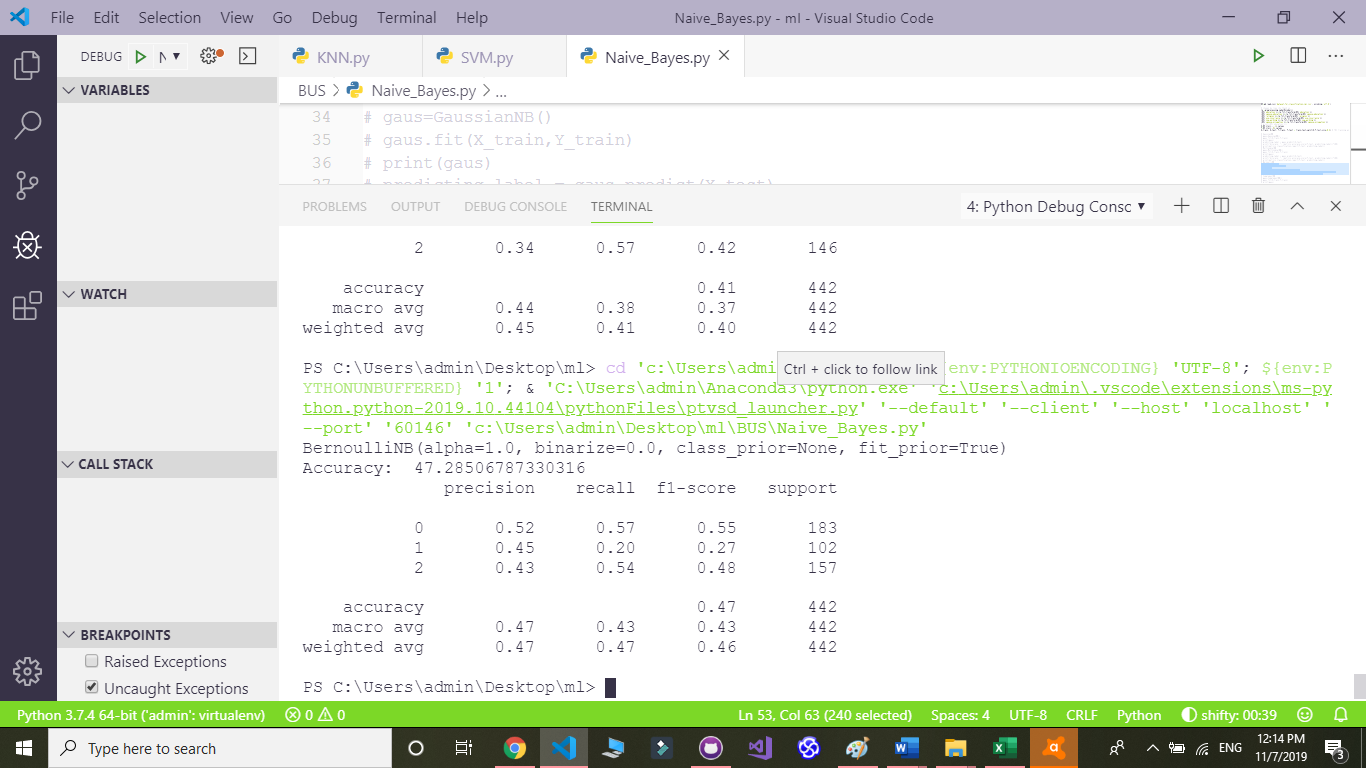
*print(gaus)*

*predicting\_label = gaus.predict(X\_test)*

*print("Accuracy: ", (metrics.accuracy\_score(Y\_test, predicting\_label))\*100)*

*print(metrics.classification\_report(Y\_test, predicting\_label))*

Kết quả chạy:



Nhận xét: Trong Naiver bayes thì đây là thuật toán chạy có độ chính xác thấp nhất, so với label 1 thì độ chính xác vẫn chưa cao.

### d.ComplementNB

**Mã nguồn:**

*gaus=ComplementNB()*

*gaus.fit(X\_train,Y\_train)*

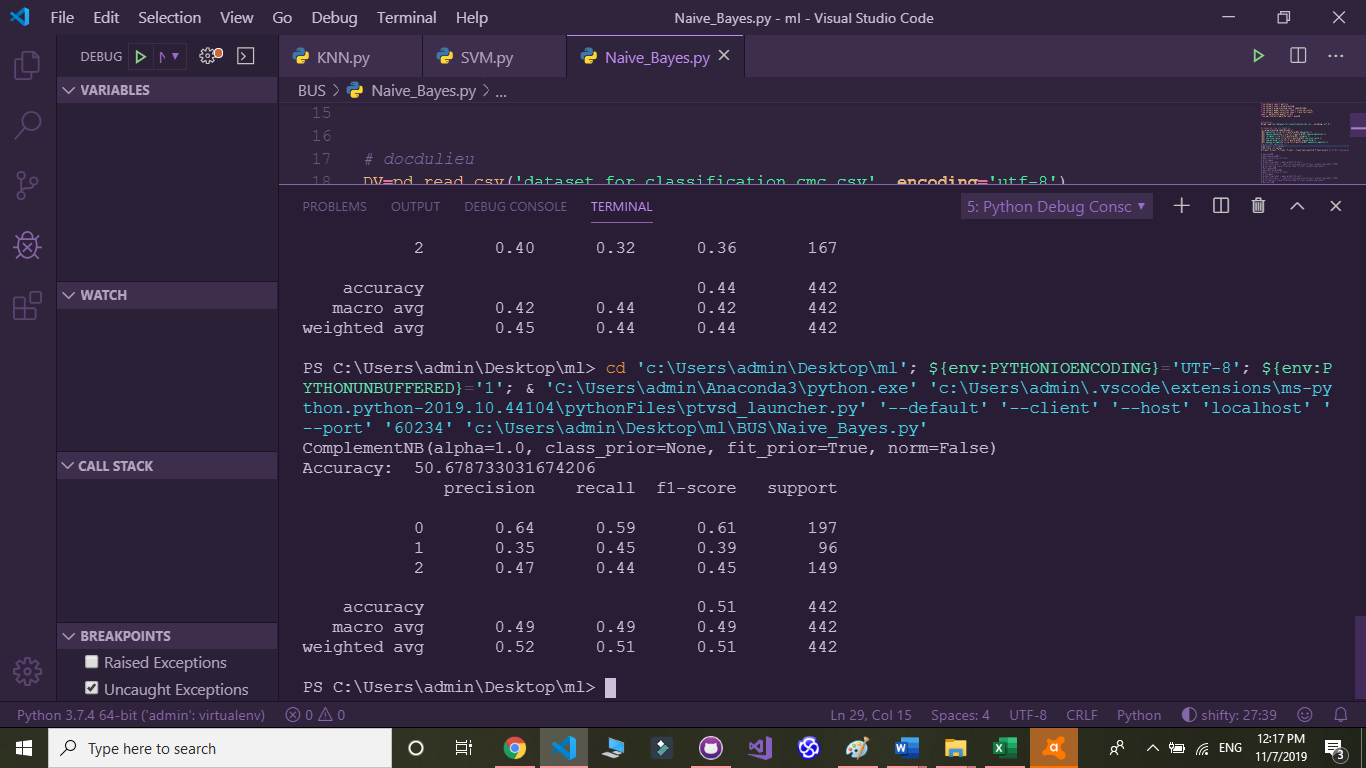
*print(gaus)*

*predicting\_label = gaus.predict(X\_test)*

*print("Accuracy: ", (metrics.accuracy\_score(Y\_test, predicting\_label))\*100)*

*print(metrics.classification\_report(Y\_test, predicting\_label))*

Kết quả chạy:



Nhận xét: Kết quả của thuật toán gần bằng với MultinomialNB. Độ chính xác ở label 1 chưa tốt.

### d. GaussianNB với CrossValidation

**Mã nguồn:**

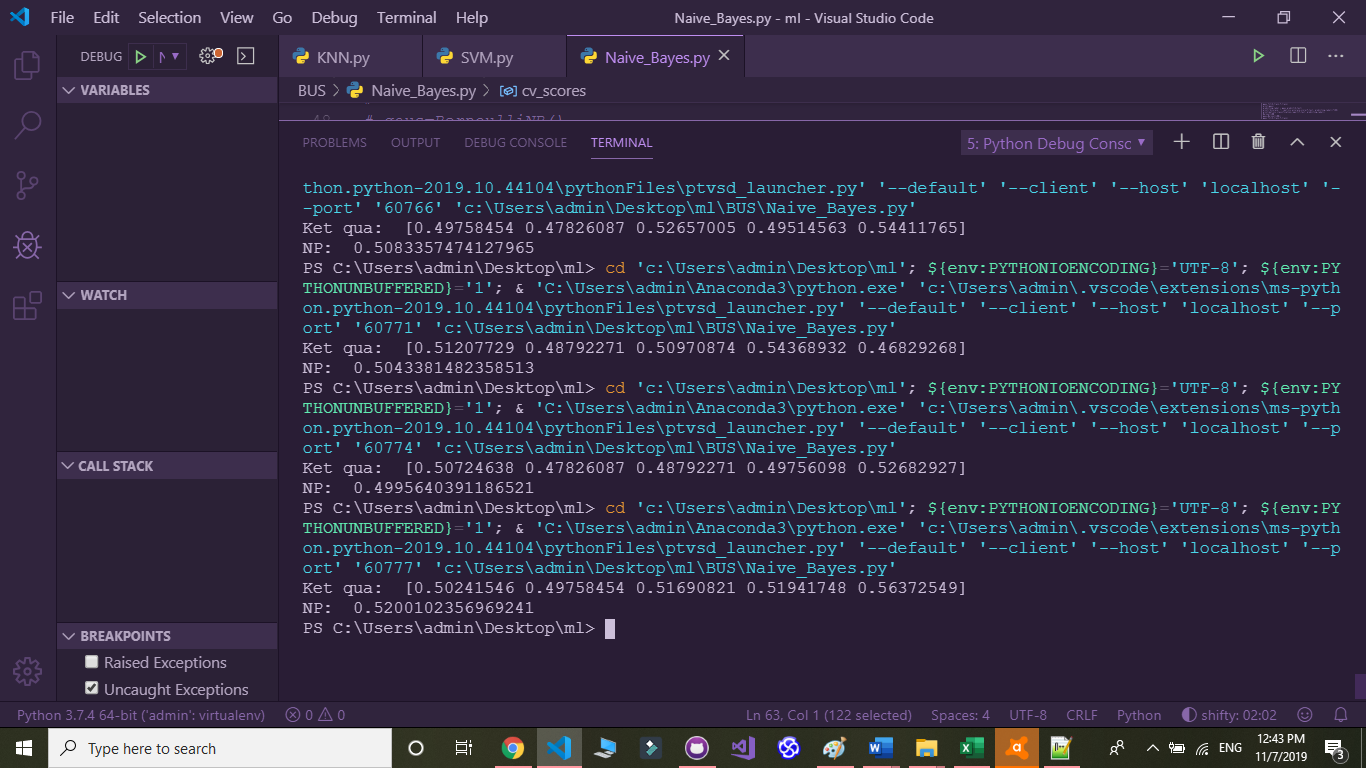
*gaus = GaussianNB()*

*cv\_scores = cross\_val\_score(gaus, X\_train, Y\_train, cv=5)*

*print("Ket qua: ", cv\_scores)*

*print("NP: ",np.mean(cv\_scores))*

Kết quả chạy được:



### e.Biểu diễn các điểm dữ liệu

**Mã nguồn:**

*import matplotlib.pyplot as plt*

*from mpl\_toolkits.mplot3d import axes3d* **#sử dụng thư viện matplotlib để vẽ đồ thị biểu diễn 2d và mpl\_toolkits.mplot3d để biểu diễn 3d**

*fig = plt.figure()*

*ax = fig.gca(projection='3d')*

*ax.grid(False)*

*for i in range(0, len(Y\_train)):*

*if Y\_train[i] == 0:*

*ax.scatter(X\_train[i][2], X\_train[i][3], X\_train[i][7],* #**Vì đã drop đi 1 cột trong dataset nên số liệu trong mảng chỉ được chọn dưới 9. Nếu chọn 9 thì dữ liệu sẽ bị báo đầy**

*c='black', s=10, cmap='viridis')*

*else:*

*ax.scatter(X\_train[i][2], X\_train[i][3], X\_train[i][7],*

*c='red', s = 10, cmap = 'viridis')*

*for i in range(0, len(X\_test)):*

*if predicting\_label[i] == 0:*

*ax.scatter(X\_test[i][2], X\_test[i][3], X\_test[i][7],*

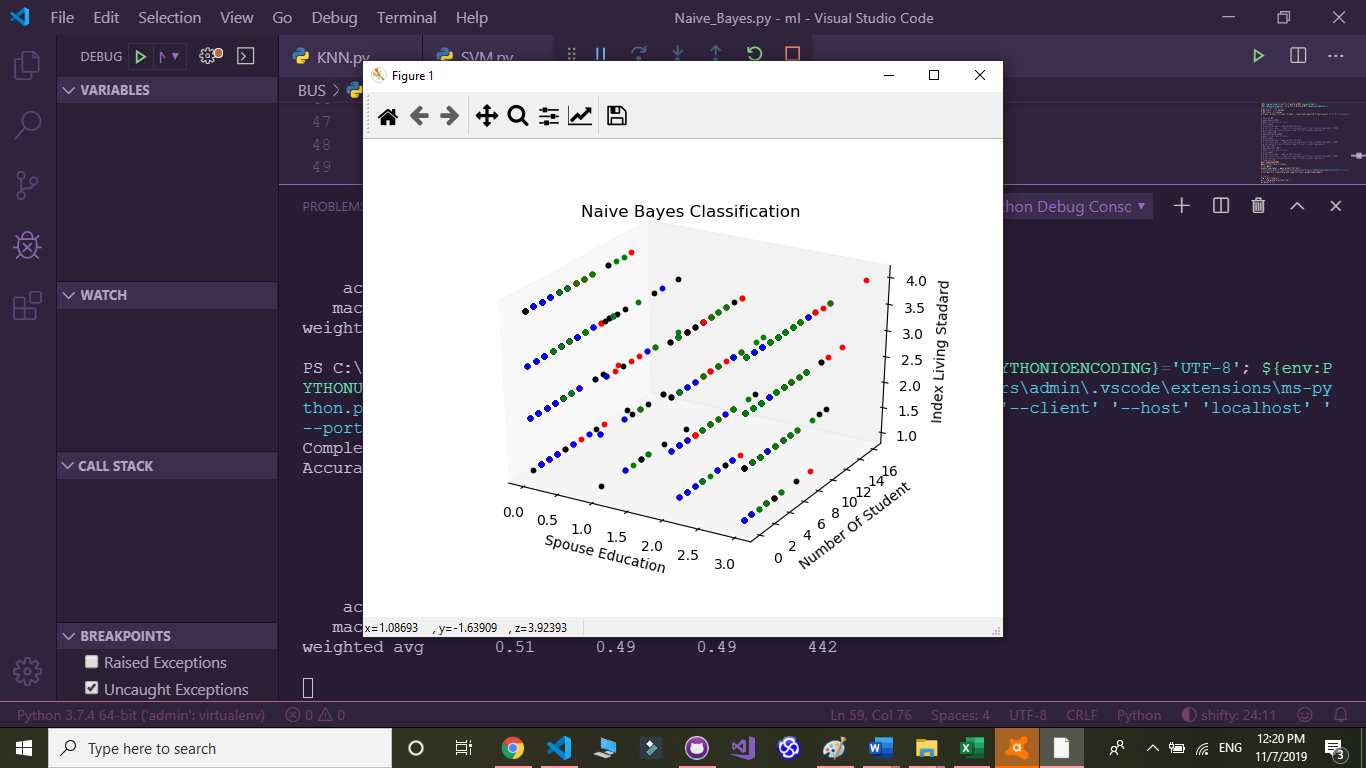
*c='blue', s=10, cmap='viridis')*

*else:*

*ax.scatter(X\_test[i][2], X\_test[i][3], X\_test[i][7],*

*c='green', s = 10, cmap = 'viridis')*

-Nhóm chọn 3 features để vẽ: spouse\_education, number\_of\_children, index\_living\_standard



## 3.KNN

### a. Chạy với dữ liệu bình thường (không sử dụng tham số của KNN)

**Mã nguồn:**

*knn = KNeighborsClassifier(n\_neighbors=15)*

*knn.fit(X\_train,Y\_train)*

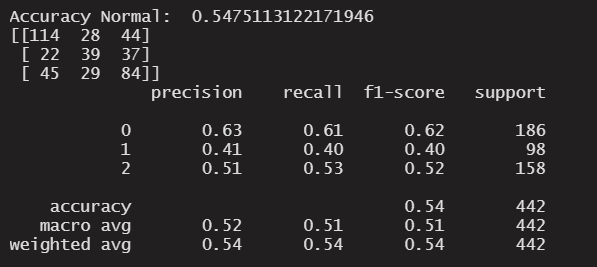
*predicting\_label = knn.predict(X\_test)*

*print(confusion\_matrix(Y\_test,predicting\_label))*

*print(classification\_report(Y\_test,predicting\_label))*

*print("Accuracy Normal: ", metrics.accuracy\_score(Y\_test,predicting\_label))*

Kết quả chạy:



Nhận xét: Kết quả chưa tốt, label 01 và 02 kết quả chưa tốt.

### b. Chạy với tham số của KNN

#### b.1.Tham số Weight:

Scikit-learn giúp chúng ta đơn giản hóa dữ liệu bằng cách gán gía trị weights = '**distance**'. (Giá trị mặc định của weights là '**uniform**', tương ứng với việc coi tất cả các điểm lân cận có giá trị như nhau như ở trên).

Với giá trị **distance** mức độ sẽ được đao dựa theo khoảng cách. Những điểm có khoảng cách gần sẽ có độ ảnh hưởng tốt hơn. Còn với **uniform** các điểm lân cận có mức độ là như nhau.

**Mã nguồn:**

#distance

*knn = KNeighborsClassifier(n\_neighbors=15, weights='distance')*

*knn.fit(X\_train,Y\_train)*

*predicting\_label = knn.predict(X\_test)*

*print(confusion\_matrix(Y\_test,predicting\_label))*

*print(classification\_report(Y\_test,predicting\_label))*

*print("Accuracy distance: ", metrics.accuracy\_score(Y\_test,predicting\_label))*

#uniform

*knn = KNeighborsClassifier(n\_neighbors=15, weights='uniform')*

*knn.fit(X\_train,Y\_train)*

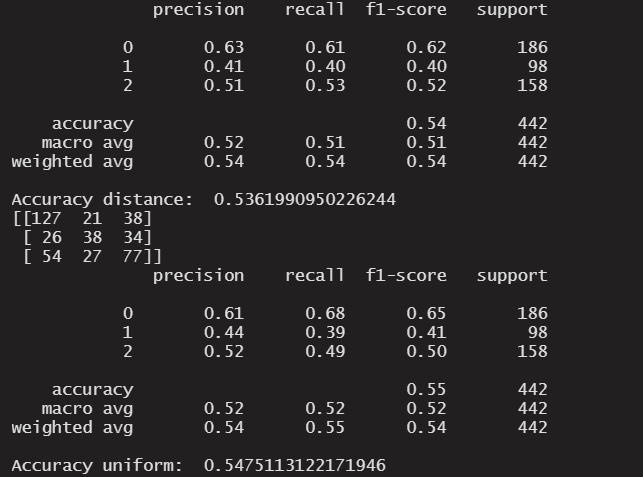
*predicting\_label = knn.predict(X\_test)*

*print(confusion\_matrix(Y\_test,predicting\_label))*

*print(classification\_report(Y\_test,predicting\_label))*

*print("Accuracy uniform: ", metrics.accuracy\_score(Y\_test,predicting\_label))*

Kết quả chạy:



Nhận xét: Kết quả có thể chênh lệch không nhiều nhưng thể hiện được sự tối ưu khi dùng tham số ‘uniform’

#### b.2.Tham số Algorithm:

Tham số Algorithm được tính toán dựa trên các thuật toán: Balltree, KDtree và Brute Force

**Mã nguồn:**

#BallTree

*knn = KNeighborsClassifier(n\_neighbors=15, algorithm='ball\_tree')*

*knn.fit(X\_train,Y\_train)*

*predicting\_label = knn.predict(X\_test)*

*print(confusion\_matrix(Y\_test,predicting\_label))*

*print(classification\_report(Y\_test,predicting\_label))*

*print("Accuracy ball\_tree: ", metrics.accuracy\_score(Y\_test,predicting\_label))*

#KDTree

*knn = KNeighborsClassifier(n\_neighbors=15, algorithm='kd\_tree')*

*knn.fit(X\_train,Y\_train)*

*predicting\_label = knn.predict(X\_test)*

*print(confusion\_matrix(Y\_test,predicting\_label))*

*print(classification\_report(Y\_test,predicting\_label))*

*print("Accuracy kd\_tree: ", metrics.accuracy\_score(Y\_test,predicting\_label))*

#BruteForce

*knn = KNeighborsClassifier(n\_neighbors=15, algorithm='brute')*

*knn.fit(X\_train,Y\_train)*

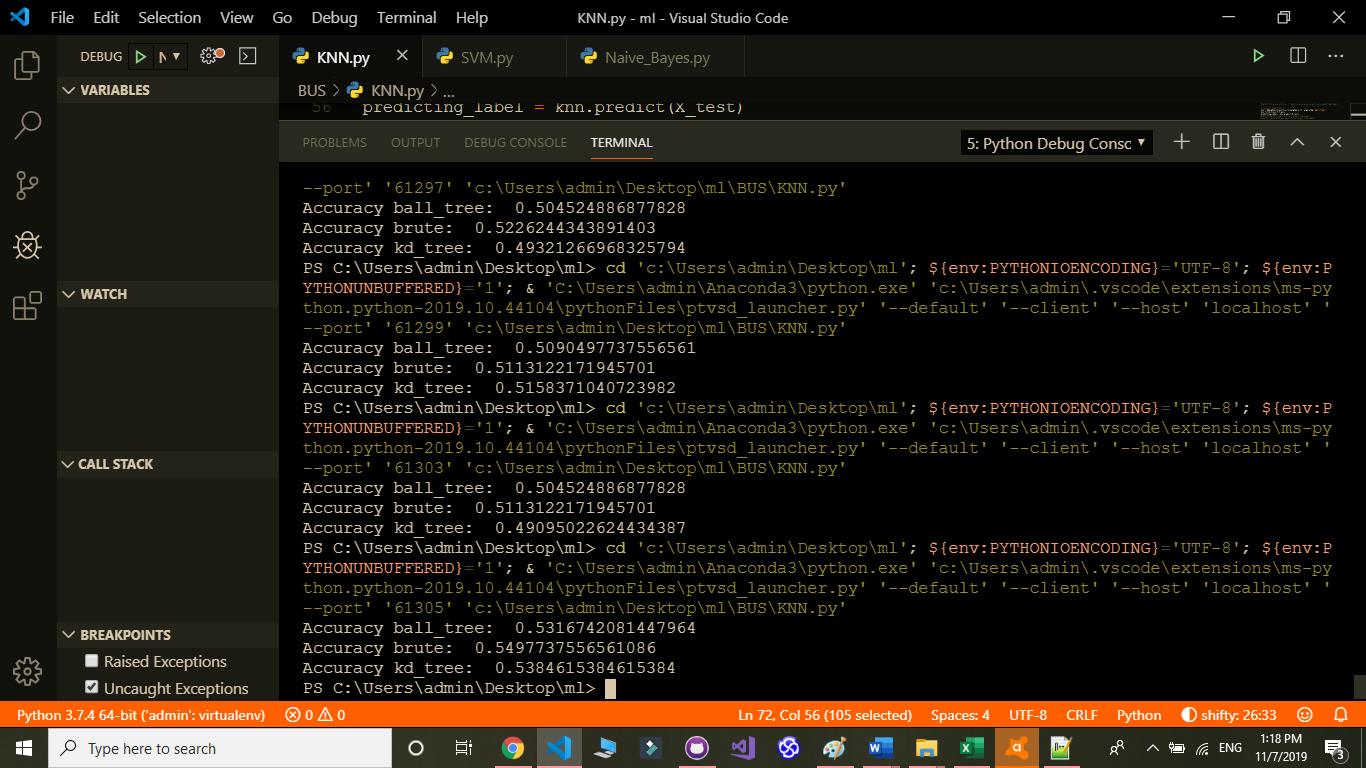
*predicting\_label = knn.predict(X\_test)*

*print(confusion\_matrix(Y\_test,predicting\_label))*

*print(classification\_report(Y\_test,predicting\_label))*

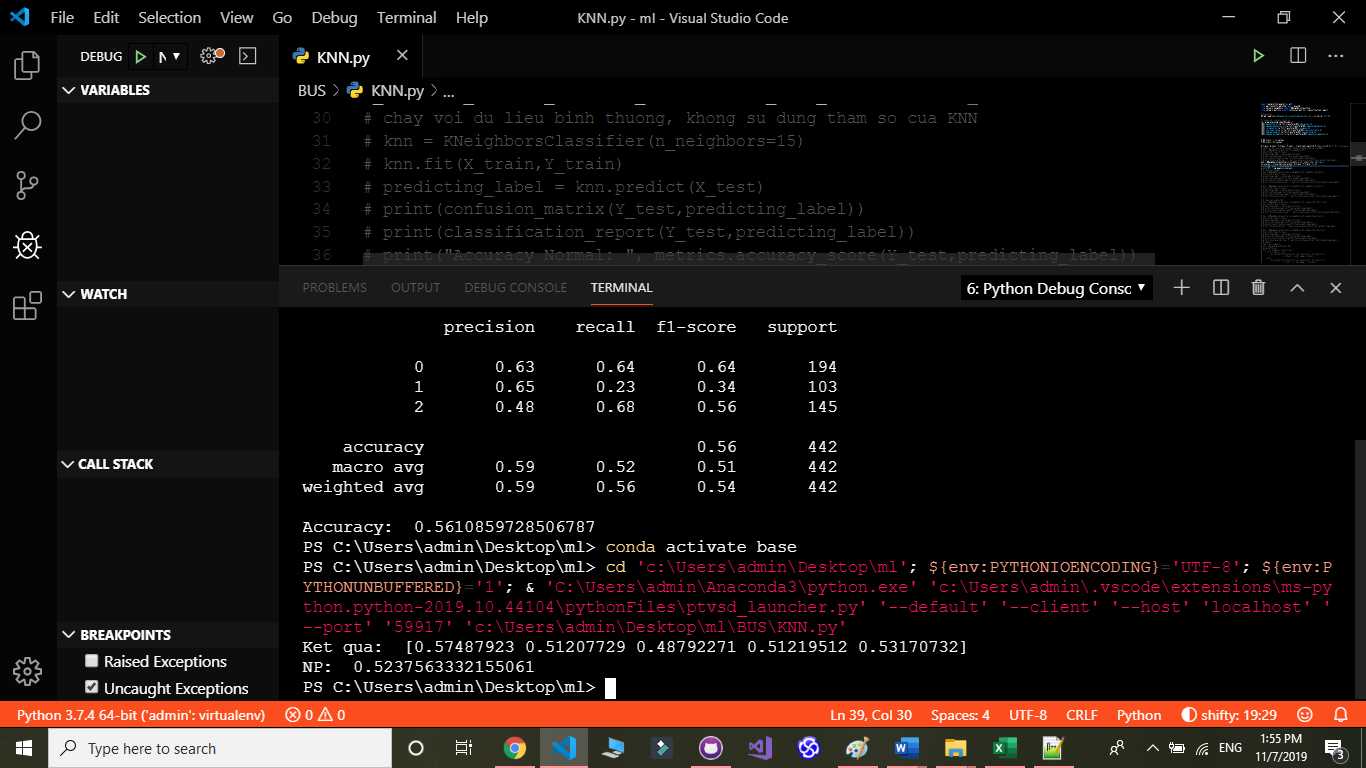
*print("Accuracy brute: ", metrics.accuracy\_score(Y\_test,predicting\_label))*

Kết quả chạy được:



Nhận xét: Kết quả chạy ra tốt nhất so với các kĩ thuật xử lí trước

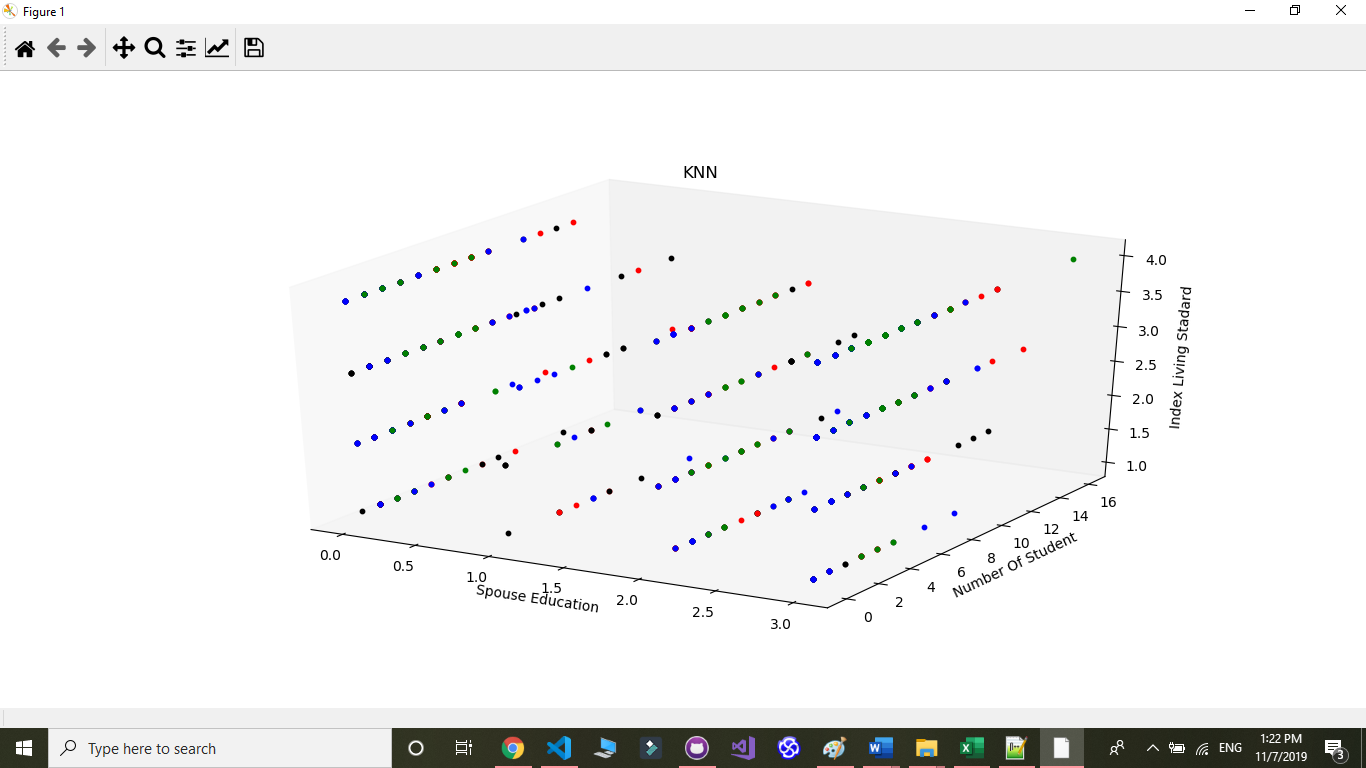
### c.KNN với Cross Validation với algorithm='kd\_tree'



### d. Biểu diễn các điểm dữ liệu

**Mã nguồn:** Tương tự thuật toán Naïve Bayes.

-Nhóm chọn 3 features để vẽ: spouse\_education, number\_of\_children, index\_living\_standard



## 4.SVM

### a.Mã nguồn

*SVM = SVC(kernel='linear')* #kernel sẽ là tuyến tính với thuật toán SVM bình thường

*SVM.fit(X\_train,Y\_train)*

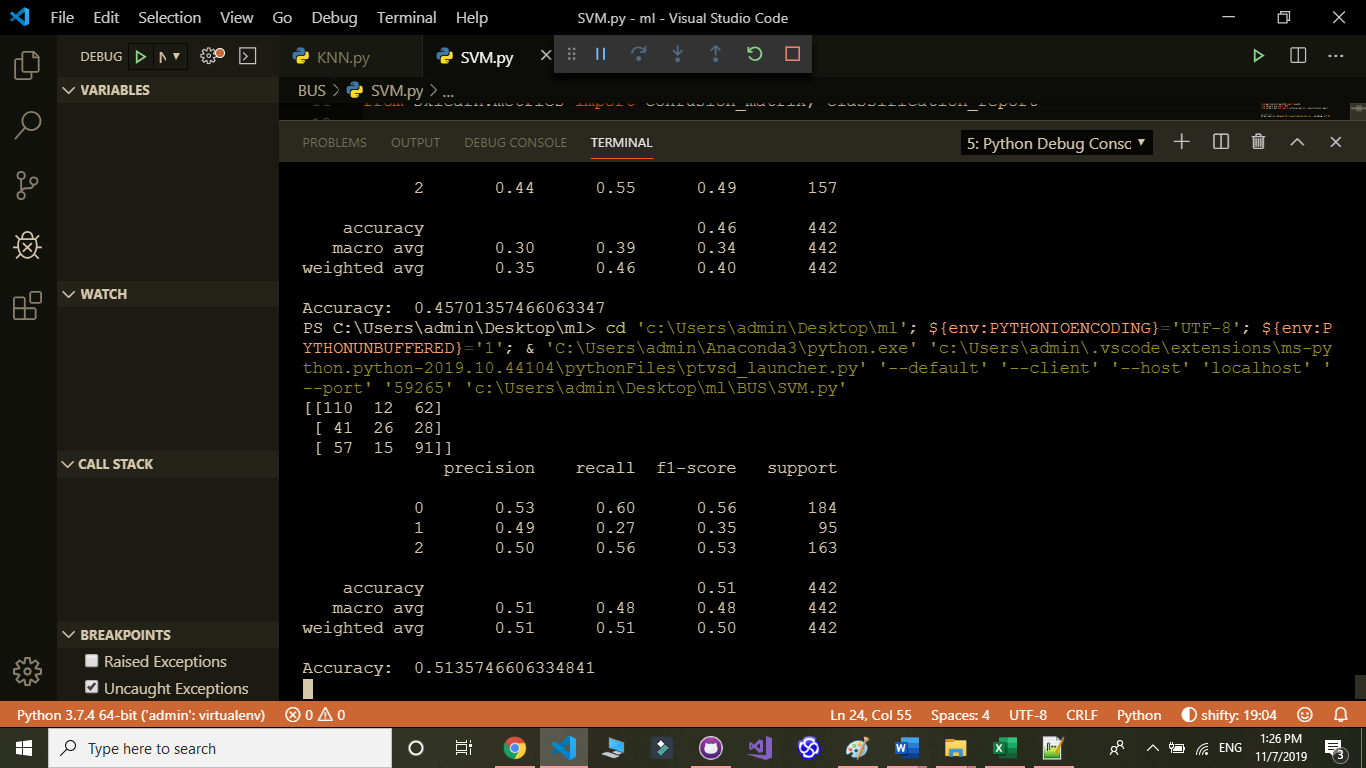
*predicting\_label = SVM.predict(X\_test)*

*print(confusion\_matrix(Y\_test,predicting\_label))*

*print(classification\_report(Y\_test,predicting\_label))*

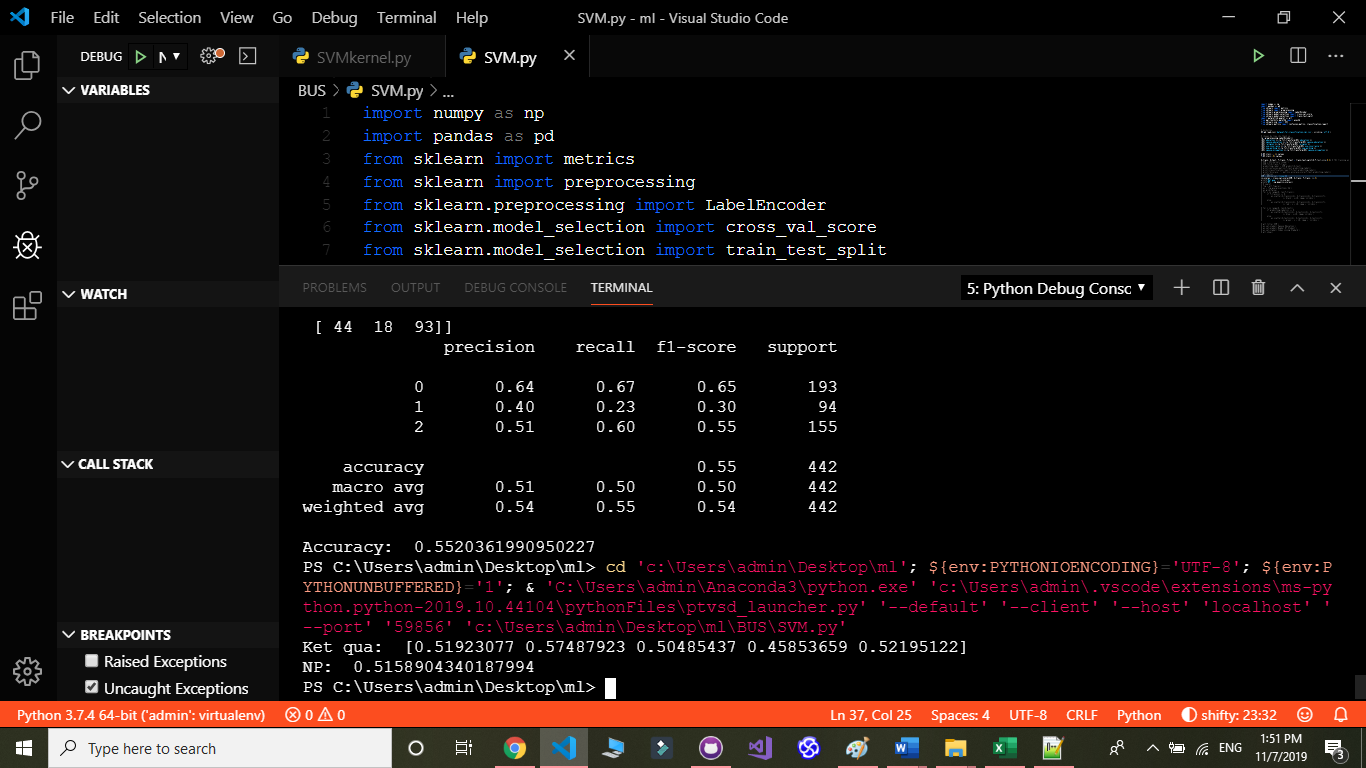
*print("Accuracy: ", metrics.accuracy\_score(Y\_test,predicting\_label))*

Kết quả chạy được:



Nhận xét: Kết quả chạy ra vẫn chưa tốt, đặc biệt là ở label 01

### b.SVM với Cross Validation



Nhận xét: Kết quả vẫn giống với trên.

### c. Biểu diễn các điểm dữ liệu

**Mã nguồn:** Tương tự thuật toán Naïve Bayes.

-Nhóm chọn 3 features để vẽ: spouse\_education, number\_of\_children, index\_living\_standard



## 5.SVM KERNEL

Với giải thuật SVM thì kernel sẽ là tuyến tính nhưng ở SVM kernel thì kernel là một tập nào đó không tuyến tính.

### a. Polynomial

**Mã nguồn:**

*SVM = SVC(kernel='poly',gamma=0.01 ,C=10)* # C là tham số xác định mức độ phạt, gamma là tham số của RBF

*SVM.fit(X\_train,Y\_train)*

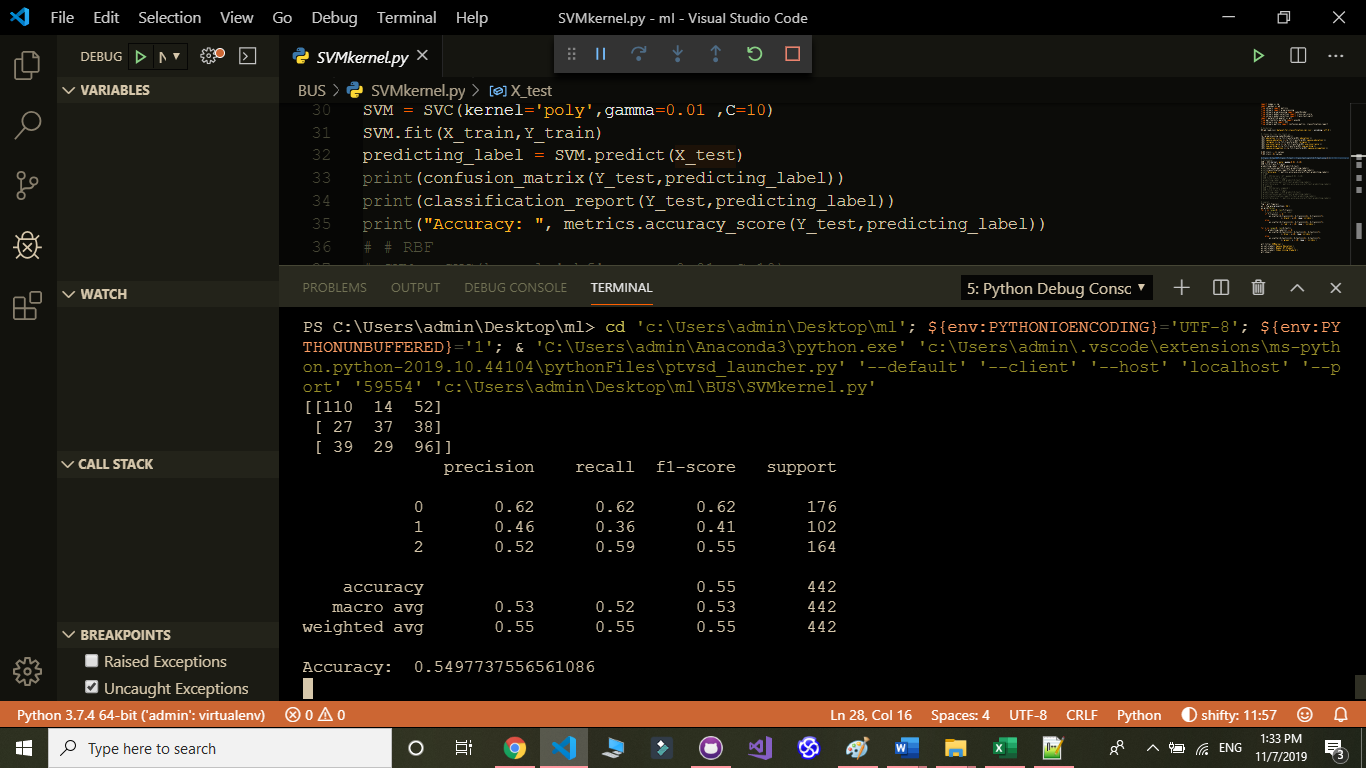
*predicting\_label = SVM.predict(X\_test)*

*print(confusion\_matrix(Y\_test,predicting\_label))*

*print(classification\_report(Y\_test,predicting\_label))*

*print("Accuracy: ", metrics.accuracy\_score(Y\_test,predicting\_label))*

Kết quả chạy được:



Nhận xét: Kết quả ôn hơn so với SVM không kernel nhưng vẫn chưa tốt

### b. RBF(Radial Basis Function)

**Mã nguồn:**

*SVM = SVC(kernel='rbf',gamma=0.01 ,C=10)*

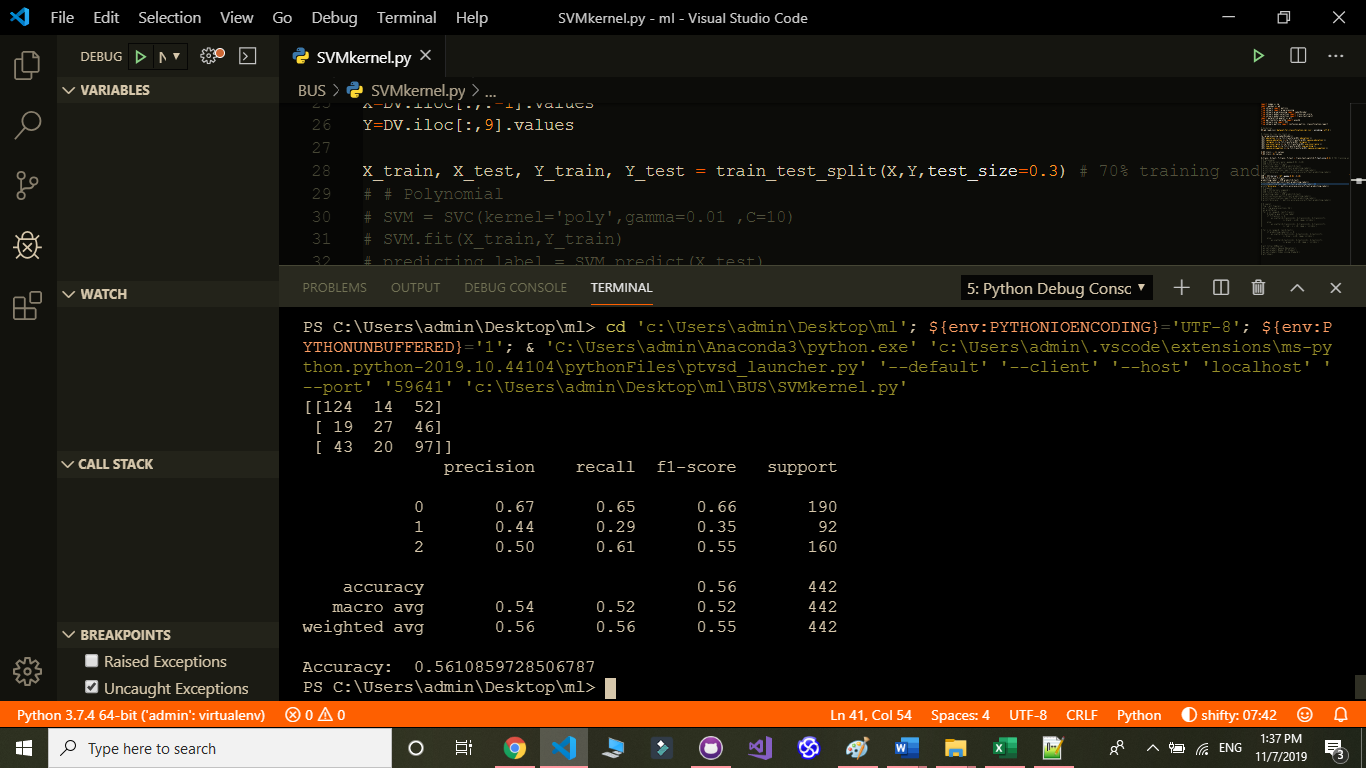
*SVM.fit(X\_train,Y\_train)*

*predicting\_label = SVM.predict(X\_test)*

*print(confusion\_matrix(Y\_test,predicting\_label))*

*print("Accuracy: ", metrics.accuracy\_score(Y\_test,predicting\_label))*

Kết quả chạy được:



Nhận xét: Kết quả khả quan hơn nhưng vẫn chưa tốt

### c. Sigmoid

**Mã nguồn:**

*SVM = SVC(kernel='sigmoid')*

*SVM.fit(X\_train,Y\_train)*

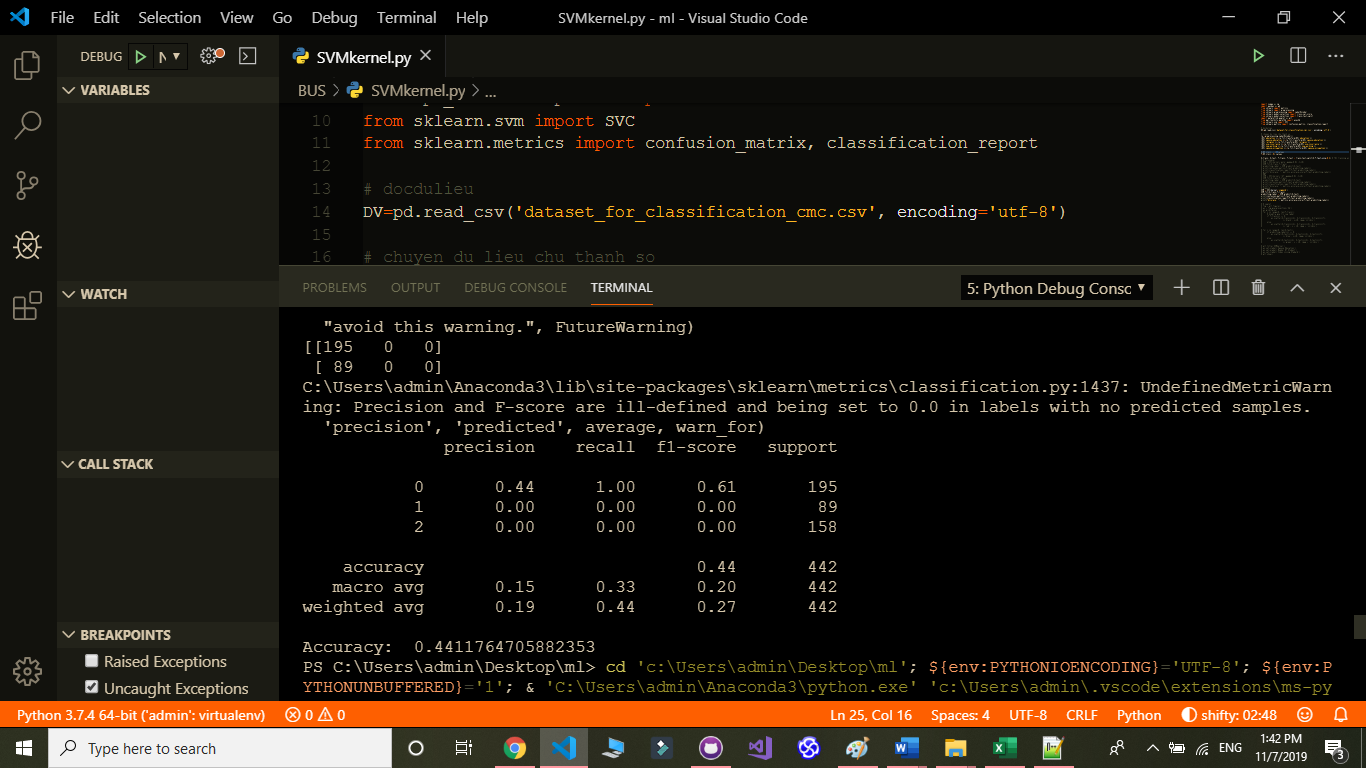
*predicting\_label = SVM.predict(X\_test)*

*print(confusion\_matrix(Y\_test,predicting\_label))*

*print(classification\_report(Y\_test,predicting\_label))*

*print("Accuracy: ", metrics.accuracy\_score(Y\_test,predicting\_label))*

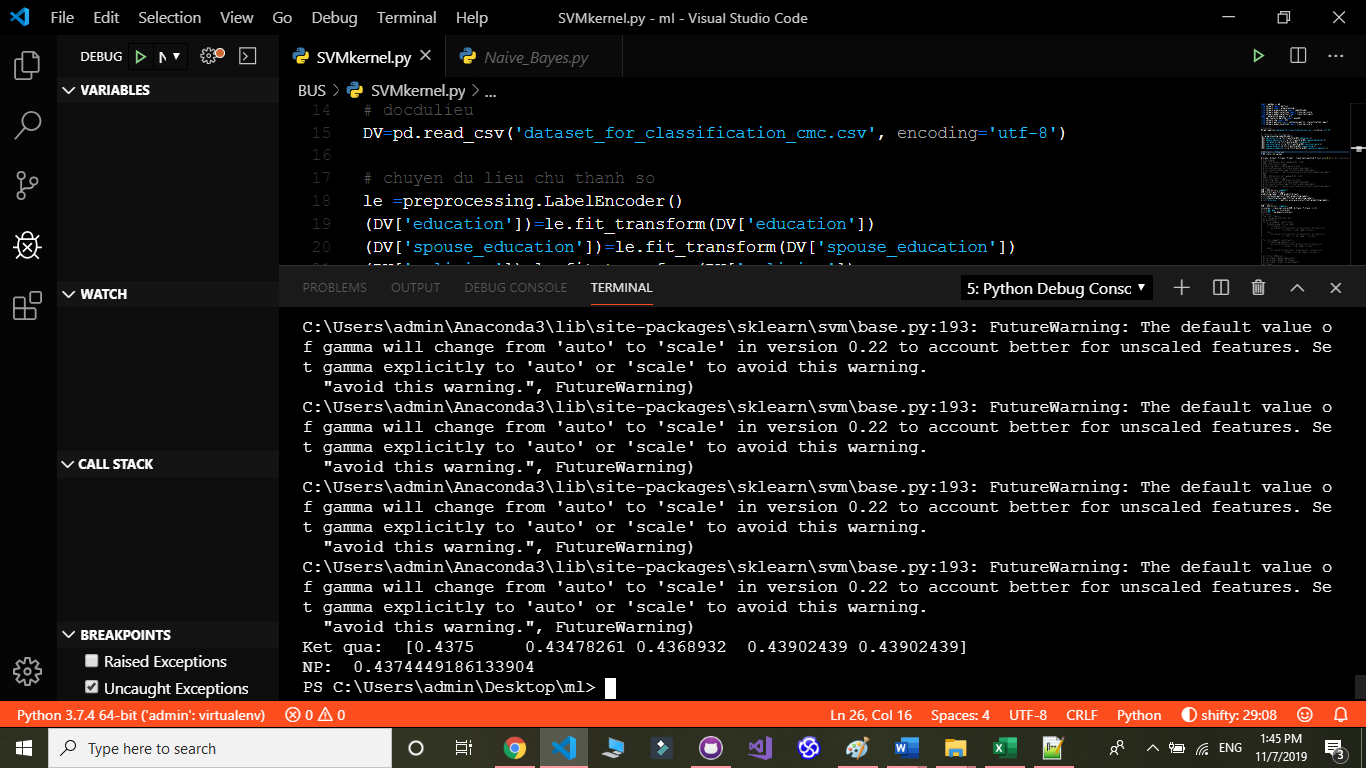
Kết quả chạy được:



Nhận xét: Kết quả chạy rất thấp so với 2 kernel còn lại.

### d. Sigmoid với CrossVailidation

Kết quả chạy được:



### e. Biểu diễn các điểm dữ liệu

**Mã nguồn:** Tương tự thuật toán Naïve Bayes.

-Nhóm chọn 3 features để vẽ: spouse\_education, number\_of\_children, index\_living\_standard



# IV-DEMO THUẬT TOÁN VỀ PHÂN CỤM (CLUSTERING)

## 1.XỬ LÍ DỮ LIỆU CHUNG

### a.Đọc dữ liệu

*import pandas as pd*

*customer\_data = pd.read\_csv('Dataset\dataset4clustering\_student.csv', delimiter=',')*

### b.Chuyển kiểu dữ liệu chữ sang label số:

## Lấy dữ liệu của các cột có số chỉ định

*data = customer\_data.iloc[:, [2, 6,8]].values*

*le =preprocessing.LabelEncoder()*

*le.fit(data[:,2])*

*data[:,2]=le.transform(data[:,2])*

##Tạo kích thước cho khung hình

*plt.figure(figsize=(8, 5))*

##Gán tên cho trục y

*plt.ylabel("Y")*

*##Gán tên cho trục x*

*plt.xlabel("X")*

**Thư viện:** from sklearn import preprocessing

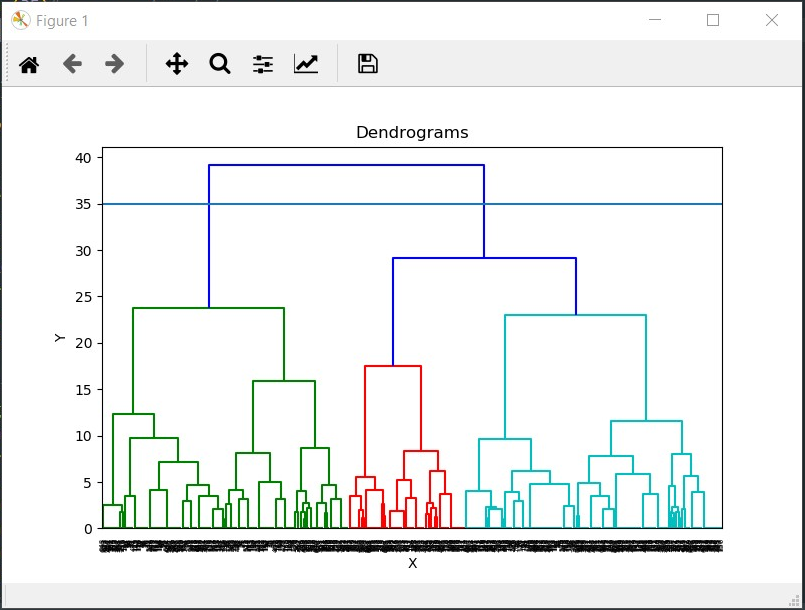
## 2. HIERARCHICAL CLUSTERING:

**Mã nguồn:**

*cluster = AgglomerativeClustering(n\_clusters=3, affinity='euclidean', linkage='ward')*

*cluster.fit(data)*

*print("Labels:", cluster.labels\_)*

a.Vẽ cấu trúc cây Dendrograms(3 label: Age, Medu,Mjob)

#Gán tiêu đề cho Dendrograms

*plt.title("Dendrograms")*

#Vẽ cấu trúc cây Dendrogram

*shc.dendrogram(shc.linkage(data, method='ward'))*

*plt.axhline(35)* **# Từ đây chúng ta sẽ vẽ đường Threshold cắt ngang cây dendrogam này tại đoạn có khoảng cách lớn nhất.**

*plt.show()*

*cluster = AgglomerativeClustering(n\_clusters=3, affinity='euclidean', linkage='ward')*

*cluster.fit(data)*

*print("Labels:", cluster.labels\_)*

### b.Vẽ mô hình phân cụm 2D

*plt.figure(figsize=(8, 5))*

*plt.ylabel("Y")*

*plt.xlabel("X")*

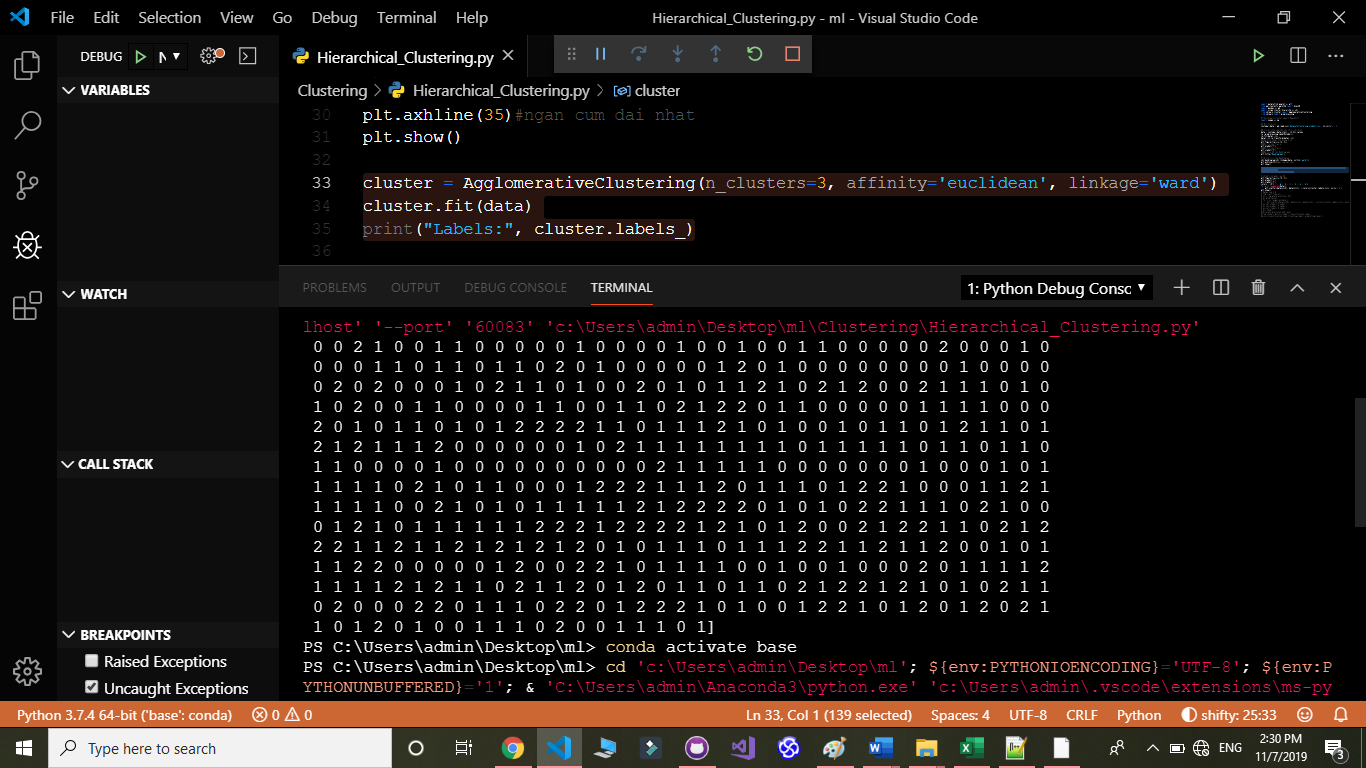
*colors = 10\*['r', 'g', 'b', 'c', 'k', 'y', 'm']*

*for i in range(len(data)):*

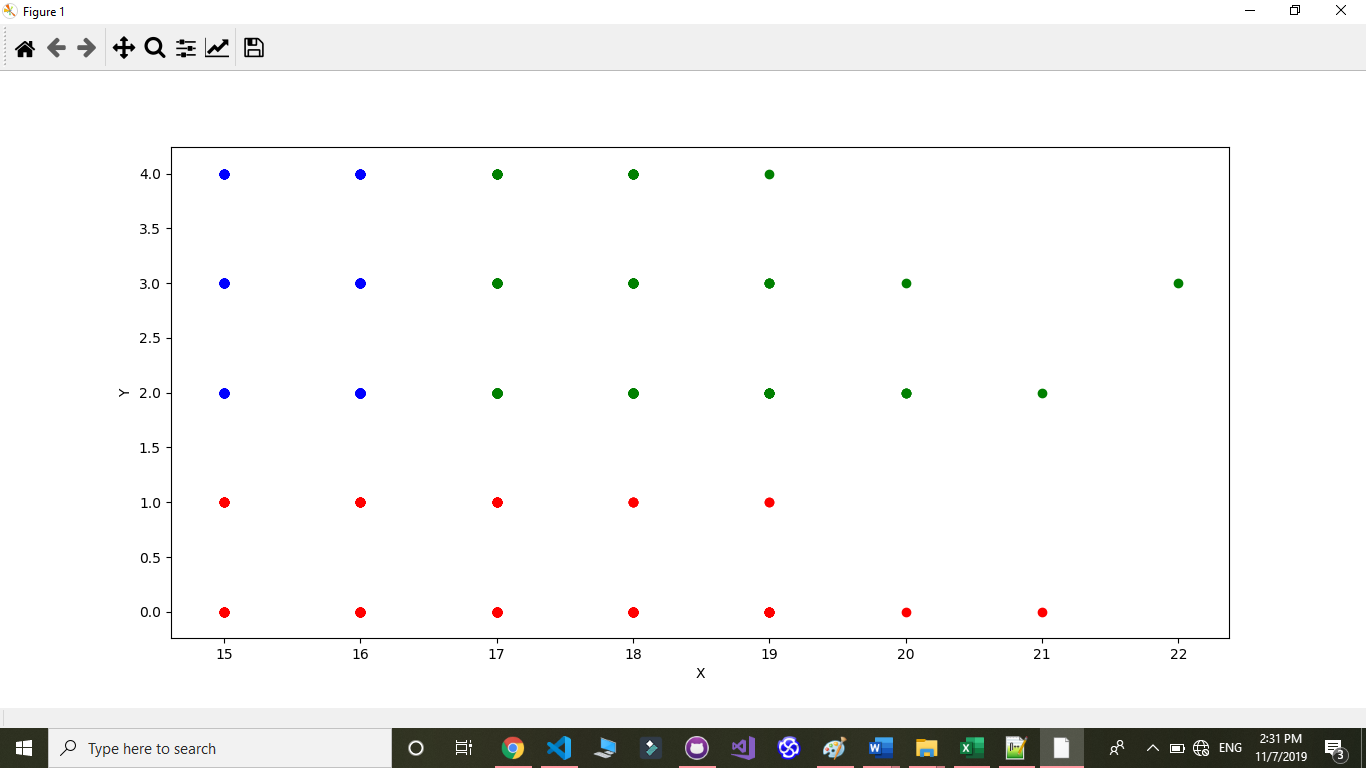
*plt.scatter(data[i][0], data[i][1], c=colors[cluster.labels\_[i]], marker='o')*

*plt.show()*

Kết quả in ra được 2 cụm được biểu diễn dưới dạng 0 và 1(chọn 2 label:age, mjob):



Mô hình phân cụm 2D (chọn 2 label: age, mjob):



### c.Vẽ mô hình phân cụm 3D

**Mã nguồn:**

*fig = plt.figure()*

*ax = fig.gca(projection='3d')*

*ax.grid(False)*

*for i in range(len(data)):*

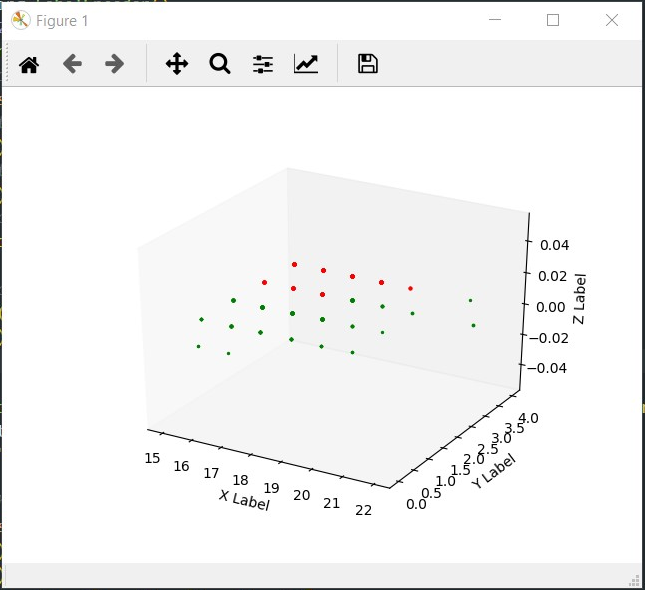
*plt.scatter(data[i][0], data[i][1], data[i][2], c=colors[cluster.labels\_[i]], marker='o')*

*ax.set\_xlabel('X Label')*

*ax.set\_ylabel('Y Label')*

*ax.set\_zlabel('Z Label')*

*plt.show()*



## 3.K-MEANS

### a. Tìm số cụm tối ưu để phân loại K-means

*wcss = []*

*for i in range(1, 14):*

*kmeans = KMeans(n\_clusters=i, init='k-means++', random\_state=42, max\_iter=300,n\_init = 10)*

*kmeans.fit(X)*

*wcss.append(kmeans.inertia\_)*

**b.** **Thể hiện kết quả tìm được lên biểu đồ**

*plt.plot(range(1, 14), wcss)*

*plt.title('The Elbow Method')*

*plt.xlabel('Number of clusters')*

*plt.ylabel('WCSS')*

*plt.show()*

### c. Tạo một K-means classifier với số cluster là 6

*kmeans = KMeans(n\_clusters=6, init='k-means++', random\_state=42, max\_iter=300,n\_init = 10)*

*y\_kmeans = kmeans.fit\_predict(X)*

d. Hiển thị các cụm 2D(chọn 3 label: age, mjob)

*plt.scatter(X[y\_kmeans == 0, 0], X[y\_kmeans == 0, 1], s=100, c='orange', label='Cluster 1')*

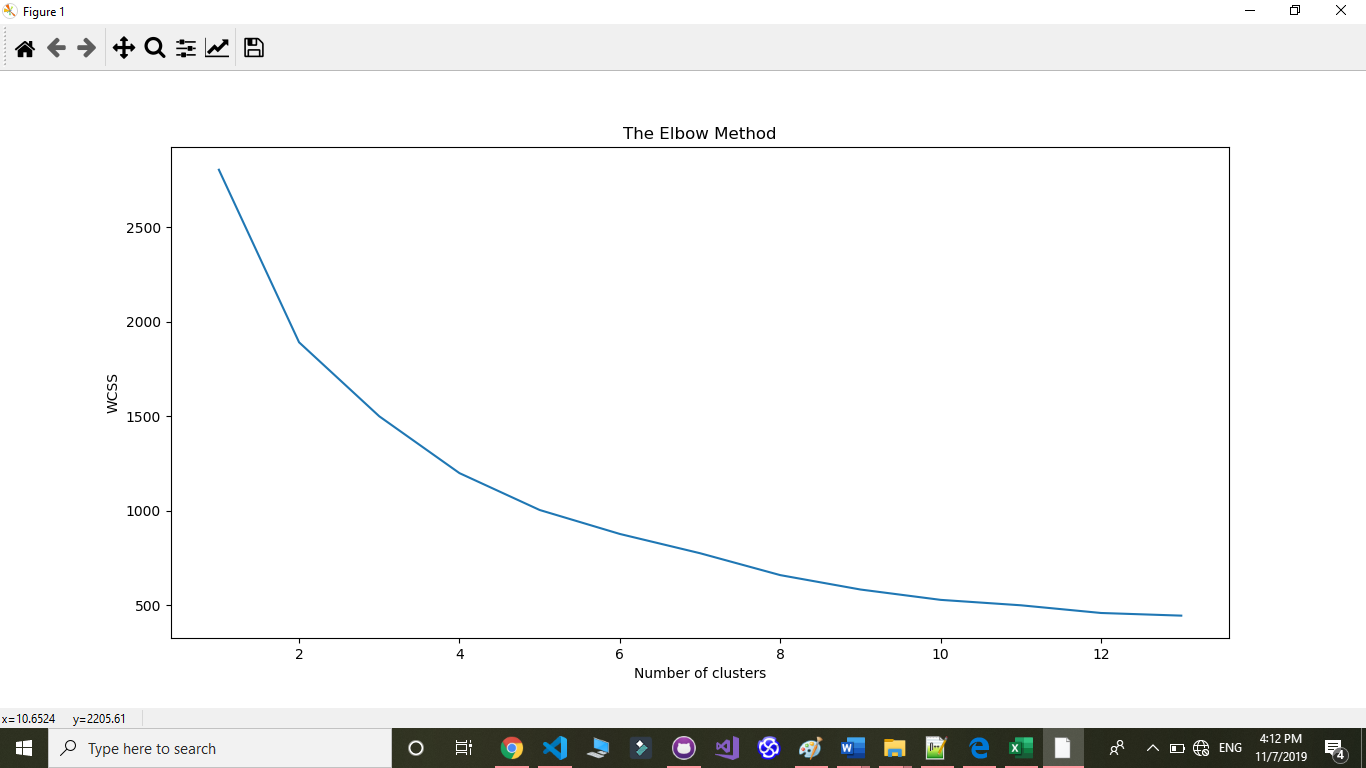
*plt.scatter(X[y\_kmeans == 1, 0], X[y\_kmeans == 1, 1], s=100, c='blue', label='Cluster 2')*

*plt.scatter(X[y\_kmeans == 2, 0], X[y\_kmeans == 2, 1], s=100, c='green', label='Cluster 3')*

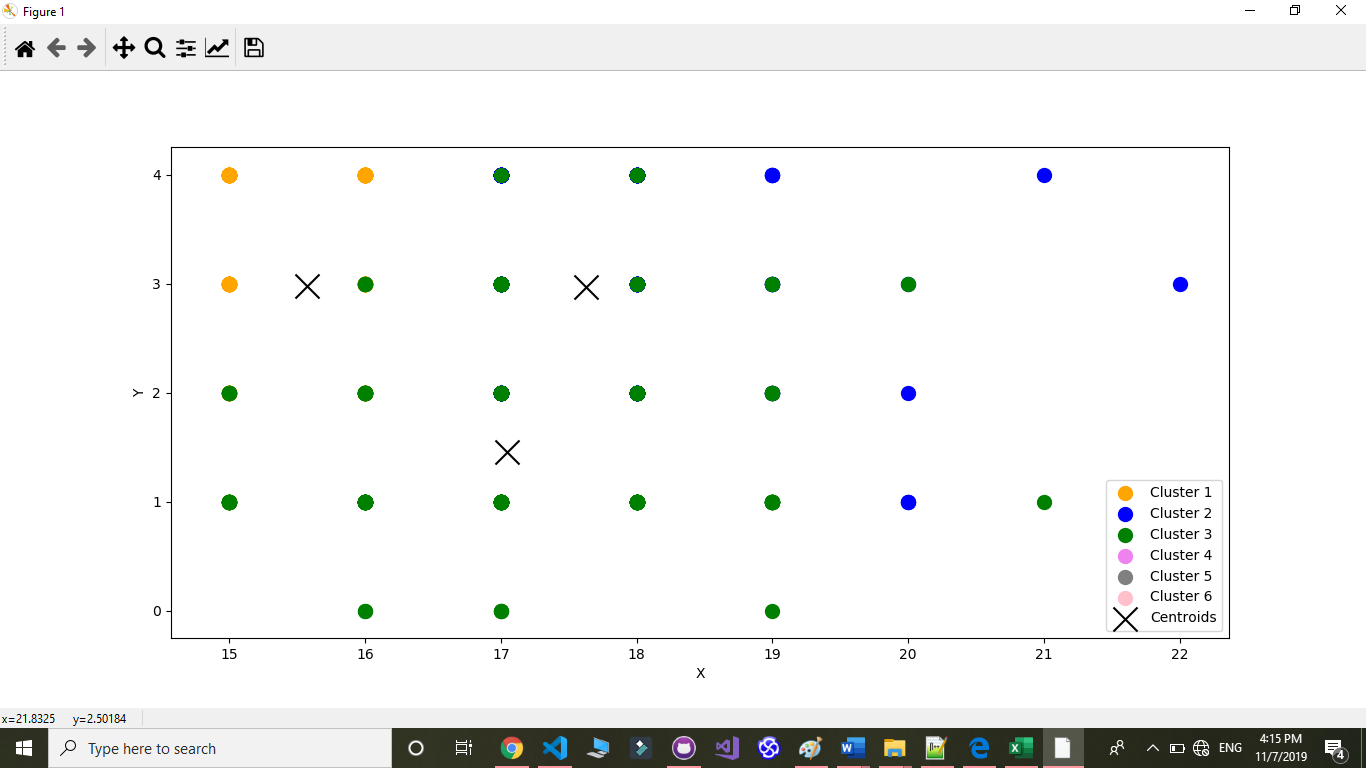
*plt.scatter(X[y\_kmeans == 3, 0], X[y\_kmeans == 3, 1], s=100, c='violet', label='Cluster 4')*

*plt.scatter(X[y\_kmeans == 4, 0], X[y\_kmeans == 4, 1], s=100, c='gray', label='Cluster 5')*

*plt.scatter(X[y\_kmeans == 5, 0], X[y\_kmeans == 5, 1], s=100, c='pink', label='Cluster 6')*



Biểu đồ cụm

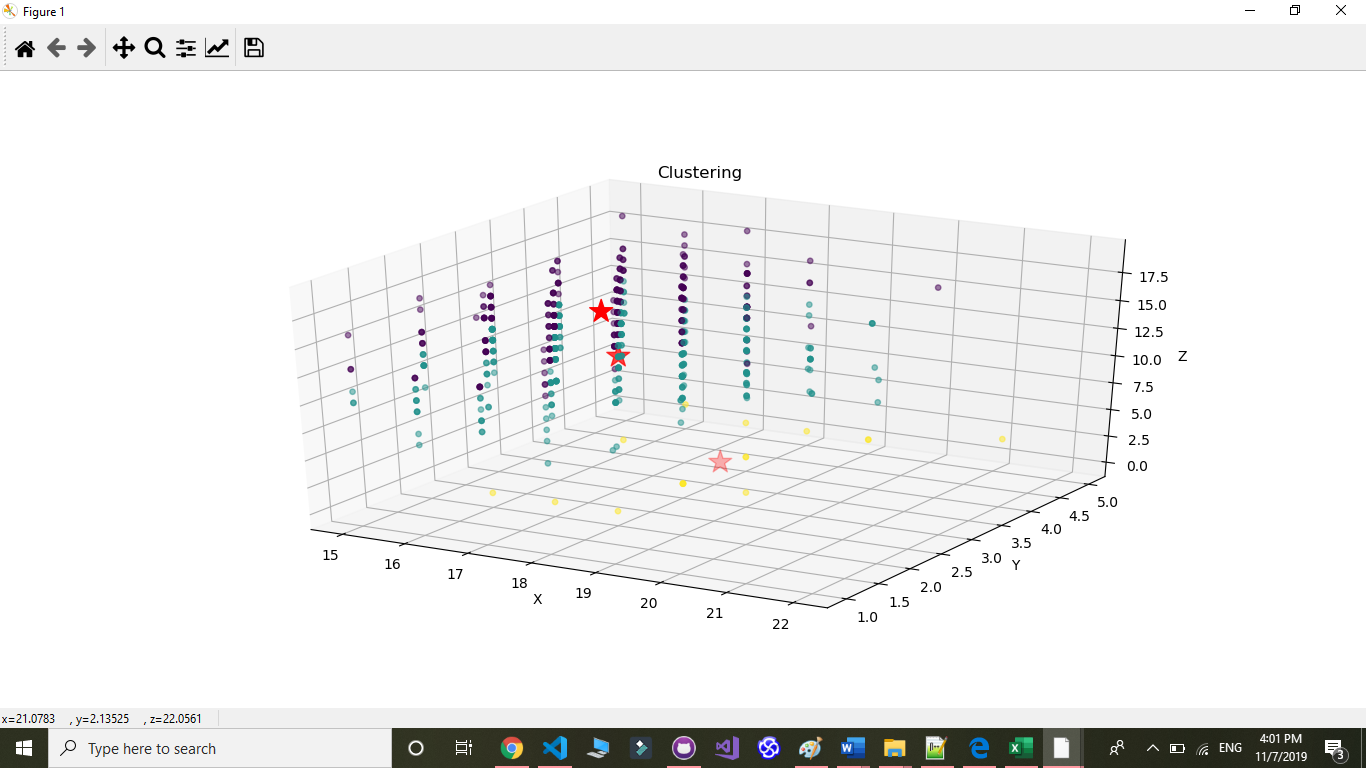


Phân cụm 2D với 3 label

Nhận xét: Khi phân cụm với label thì centroids hiển thị trên 2D với 3label không chính xác.

-Xét trường hợp phân cụm với 2 label trên 2D sẽ chính xác hơn (age,Mjob):



e. Hiển thị các cụm 3D(Chọn 3 label: age, freetime,G3)

*cols = ['age','freetime','G3']*

*X2 = pd.read\_csv("dataset4clustering\_student.csv", usecols = cols)*

*k\_means\_4pl = KMeans(*

*n\_clusters=3, init='k-means++',*

*n\_init=10, max\_iter=300, random\_state=0)*

*res = k\_means\_4pl.fit\_predict(X2)*

*center = k\_means\_4pl.cluster\_centers\_*

*fig =plt.figure()*

*ax = plt.axes(projection='3d')*

*ax.scatter(X2.iloc[:,0],X2.iloc[:,1],X2.iloc[:,2],alpha=0.5,marker='o',c= k\_means\_4pl.labels\_ ,s=15)*

*plt.title('Clustering')*

*ax.set\_xlabel('X')*

*ax.set\_ylabel("Y")*

*ax.set\_zlabel("Z")*

*ax.scatter(center[:,0], center[:,1], center[:,2], s = 300, c = 'r', marker='\*', label = 'Centroid')*

*plt.show()*

# V-TÀI LIỆU THAM KHẢO

ĐẠO VĂN: <http://rces.info/sinh-vien-kinh-te-nckh/loi-dao-van-trong-nghien-cuu-khoa-hoc-va-cach-phong-tranh/>

TÌM HIỂU THƯ VIỆN SCIKIT-LEARN:

<https://scikitlearn.org/stable/modules/generated/sklearn.cluster.FeatureAgglomeration.html>

<http://bis.net.vn/forums/t/571.aspx>

CLASSIFICATION:

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

<https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html>

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

CLUSTERING:

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

<https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html>