

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM

KHOA ĐÀO TẠO CHẤT LƯỢNG CAO

-----□□🕮□□-----



**HỌC MÁY**

**TÌM HIỂU VỀ THUẬT TOÁN**

**HIERARCHICAL CLUSTERING**

GVHD: Cô Từ Tuyết Hồng

Nhóm :

17110213 Trần Cao Quyền

17110181 Nguyễn Trọng Luật

TP.HCM, tháng 12 năm 2020

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM**  **Độc lập – Tự do – Hạnh Phúc**  **\*\*\*\*\*\*\*** |  |  |  |  |  |

# PHIẾU NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

Họ và tên các sinh viên: Trần Cao Quyền MSSV: 17110213

Nguyễn Trọng Luật MSSV: 17110181

Ngành: Công nghệ thông tin

Chuyên ngành: Công nghệ phần mềm Lớp: 17110CLST

Họ và tên Giáo viên hướng dẫn: Cô Từ Tuyết Hồng

**NHẬN XÉT**

1. Về nội dung báo cáo & khối lượng thực hiện:

1. Ưu điểm:

1. Khuyết điểm:

1. Đề nghị cho bảo vệ hay không?

1. Đánh giá loại:

6. Điểm (Bằng chữ: )

*TP.Hồ Chí Minh, ngày tháng năm 2020*

Giáo viên hướng dẫn

*(Ký & ghi rõ họ tên)*

# LỜI CẢM ƠN

Lời đầu tiên, nhóm thực hiện báo cáo xin gửi lời cảm ơn chân thành nhất tới cô Từ Tuyết Hồng, giáo viên hướng dẫn đã giúp đỡ nhóm hoàn thành bài báo cáo môn học Kiểm thử phần mềm. Nhờ có sự hướng dẫn và truyền đạt kiến thức cần thiết từ đầu học kỳ tới giờ, luôn hỗ trợ và giúp đỡ cho nhóm khi khó khăn. Đồng thời, nhóm thực hiện báo cáo cũng gửi lời cảm ơn tới các thầy, cô trong khoa Đào Tạo Chất Lượng Cao nói chung và ngành Công Nghệ Thông Tin nói riêng, đã trang bị những kiến thức cần thiết, kinh nghiệm và luôn hỗ trợ nhóm trong quá trình học tập.

Xin chân thành cảm ơn!

*TP.Hồ Chí Minh, ngày tháng 12 năm 2020*

**Đại diện nhóm thực hiện báo cáo**

Trần Cao Quyền

# LỜI GIỚI THIỆU

Bài toán về Logistics nói chung và bài toán người giao hàng nói riêng luôn là vấn đề mà các nhà nghiên cứu, các doanh nghiệp, các đơn vị vận chuyển vẫn đang tiếp tục tìm hiểu và nghiên cứu để đưa ra giải pháp tốt nhất cho vấn đề này. Việc tìm ra thuật toán phù hợp, giải pháp cấp thiết để việc vận chuyển giữa các cơ sở, việc vận chuyển từ đơn vị đến tay người dùng làm sao cho nhanh và tiết kiệm chi phí. Bài toán này cũng có nhiều phương pháp khác nhau, thuật toán khác nhau. Hieararchical Clustering là thuật toán của Machine Learning, thuật toán phân cụm có thể sẽ là giải pháp hợp lý chăng?

Mục lục:

[PHIẾU NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN 2](#_Toc60688897)

[LỜI CẢM ƠN 3](#_Toc60688898)

[LỜI GIỚI THIỆU 4](#_Toc60688899)

[Chương 1 TỔNG QUAN 9](#_Toc60688900)

[1.1 Mục Đích 9](#_Toc60688901)

[Chương 2 CƠ SỞ LÝ THUYẾT 9](#_Toc60688902)

[2.1 Heirarchical Clustering 9](#_Toc60688903)

[2.1.1 Heirarchical Clustering 9](#_Toc60688904)

[2.1.2 Phân loại 9](#_Toc60688905)

[2.1.3 Phép Đo 10](#_Toc60688906)

[2.1.3.1 Euclid 10](#_Toc60688907)

[2.1.3.2 Manhattan 10](#_Toc60688908)

[2.1.4 Agglomerative 10](#_Toc60688909)

[2.1.4.1 Single link 10](#_Toc60688910)

[2.1.4.2 Complete link 11](#_Toc60688911)

[2.1.4.3 Average link 12](#_Toc60688912)

[2.1.4.4 Ward 13](#_Toc60688913)

[2.1.5 Ứng dụng 14](#_Toc60688914)

[2.1.6 So sánh với các thuật toán khác 14](#_Toc60688915)

[2.2 Python 14](#_Toc60688916)

[2.2.1 Giới thiệu về Python 14](#_Toc60688917)

[2.2.2 Scikit Learn AgglomerativeClustering 15](#_Toc60688918)

[2.2.3 Numpy 15](#_Toc60688919)

[2.2.4 Pandas 15](#_Toc60688920)

[2.2.5 Flask 16](#_Toc60688921)

[2.2.6 Apscheduler 16](#_Toc60688922)

[Chương 3 GIỚI THIỆU SẢN PHẨM 16](#_Toc60688923)

[3.1 Tổng quan hệ thống 16](#_Toc60688924)

[3.2 Thành phần và chức năng 16](#_Toc60688925)

[3.2.1 Hàm Fetch Api Từ Server 16](#_Toc60688926)

[3.2.2 Hàm Chuyển Json sang Numpy Array 17](#_Toc60688927)

[3.2.3 Hàm kiểm tra Kích thước 18](#_Toc60688928)

[3.2.4 Hàm Phân Cụm 18](#_Toc60688929)

[3.2.5 Hàm tính kích thước sản phẩm trong đơn hàng 19](#_Toc60688930)

[3.2.6 Hàm chuyển từ Numpy Array sang Json 19](#_Toc60688931)

[3.2.7 Hàm Một Nhánh 20](#_Toc60688932)

[3.2.8 Hàm Xử lý phân cụm 21](#_Toc60688933)

[3.2.9 Hàm chạy tất cả các xử lý phân cụm 22](#_Toc60688934)

[3.2.10 Hàm post Json lên server 23](#_Toc60688935)

[3.2.11 Hàm lập lịch trình theo thời gian 23](#_Toc60688936)

[3.2.12 Hàm lấy và xử lý thời gian 24](#_Toc60688937)

[Chương 4 CÀI ĐẶT VÀ KIỂM THỬ 25](#_Toc60688938)

[4.1 Cài Đặt 25](#_Toc60688939)

[4.1.1 Yêu cầu tiên quyết 25](#_Toc60688940)

[4.1.2 Các bước cài đặt 25](#_Toc60688941)

[4.1.3 Nâng cao 28](#_Toc60688942)

[4.2 Kiểm Thử 34](#_Toc60688943)

[Chương 5 KẾT LUẬN 34](#_Toc60688944)

[5.1 Kết quả đạt được 34](#_Toc60688945)

[5.2 Ưu điểm 34](#_Toc60688946)

[5.3 Khuyết điểm 34](#_Toc60688947)

[TÀI LIỆU THAM KHẢO 35](#_Toc60688948)

Mục Lục bảng:

[Bảng 1 getalldatda 17](#_Toc60688885)

[Bảng 2 getjsontoarray 18](#_Toc60688886)

[Bảng 3 checkSideOder 18](#_Toc60688887)

[Bảng 4 Cluster 19](#_Toc60688888)

[Bảng 5 getSizeSanPhamByIdDonHang 19](#_Toc60688889)

[Bảng 6 outputdatatojson 20](#_Toc60688890)

[Bảng 7 Motnhanh 21](#_Toc60688891)

[Bảng 8 runcluster 22](#_Toc60688892)

[Bảng 9 runall 23](#_Toc60688893)

[Bảng 10 getlapchuoidonhang 23](#_Toc60688894)

[Bảng 11 loopHourInDay 24](#_Toc60688895)

[Bảng 12 gettime 24](#_Toc60688896)

# Chương 1 TỔNG QUAN

## Mục Đích

Tìm hiểu về thuật toán Hierarchical Clustering và áp dụng thuật toán vào bài toán người giao hàng. Tìm hiểu về ngôn ngữ lập trình Python, công cụ lập trình và các thư viện hỗ trợ cho việc triển khai thuật toán.

# Chương 2 CƠ SỞ LÝ THUYẾT

## 2.1 Heirarchical Clustering

### 2.1.1 Heirarchical Clustering

Hierarchical Clustering hay còn gọi là phân cụm phân cấp hoặc phân cụm thứ cấp là giải thuật học tập không giám sát khác được sử dụng để nhóm các điểm dữ liệu không được gắn nhãn có các đặc điểm tương tự lại với nhau. Hệ thống phân cấp của các cụm này thường được biểu diễn dưới dạnh cây (hoặc biểu đồ hình hạt). Phần rễ (root) là một cụm duy nhất gồm tập hợp tất cả các mẫu và lá (leaf) là cụm mà chỉ có một mẫu.

### 2.1.2 Phân loại

Heirarchical Clustering được phân loại thành hai phương pháp là Agglomerative (tổng hợp) và Divisive (phân chia)

- Agglomerative là phương pháp tiếp cận “bottom up”, các đối tượng riêng lẻ sẽ đượng gom lại thành cụm theo một liên kết hoặc một sự tính toán cho đến khi hợp nhất lại thành một cụm duy nhất. Là phương pháp đi từ là đến gốc.

- Divisive là phương pháp tiếp cận “top down”, ban đầu là một cụm duy nhất chứa tất cả các đối tượng, sau đó được phân chia ra thành từng cụm nhỏ theo một sự liên kết hoặc sự tính toán cho đến khi mỗi đối tượng được nằm trong một cụm riêng. Là phương pháp đi từ gốc đến lá.

### 2.1.3 Phép Đo

#### 2.1.3.1 Euclid

d(x,y) =

#### 2.1.3.2 Manhattan

d(x,y) =

### 2.1.4 Agglomerative

#### 2.1.4.1 Single link

Single link là mối liên kết giữa các cụm theo mối quan hệ: khoảng cách giữa hai điểm gần nhất của hai cụm. Thuật toán sẽ tính khoảng cách giữa các cụm với nhau, sau đó xét các cặp cụm, cụm nào có điểm của hai cụm có khoảng cách nhỏ nhất, thuật toán sẽ gom cụm hai cụm đó lại và tiến hành tính lại khoảng cách giữa các cụm. Thuật toán sẽ làm tương tự cho đến khi chỉ còn một cụm chứa tất cả các đối tượng.

Ví dụ:

|  |  |  |  |
| --- | --- | --- | --- |
|  | A | B | C |
| A | 0 |  |  |
| B | 6 | 0 |  |
| C | 5 | 7 | 0 |

Bước 1: Xét thấy cụm A và cụm C có khoảng cách nhỏ nhất. Tiến hành gom cụm A và C lại và thực hiện tính khoảng cách giữa các cụm mới.

B 🡪 AC = max(B 🡪 A, B 🡪 C) = max(6, 7) = 6

|  |  |  |
| --- | --- | --- |
|  | A,C | B |
| A,C | 0 |  |
| B | 6 | 0 |

Bước 2: Xét thấy cụm AC và cụm B có khoảng cách nhỏ nhất. Tiến hành gom cụm AC và B lại và thực hiện khoảng cách giữa các cụm mới.

|  |  |
| --- | --- |
|  | A,B,C |
| A,B,C | 0 |

Bước 3 Kiếm tra số cụm thành một thì dừng lại

#### 2.1.4.2 Complete link

Complete link là mối liên kết giữa các cụm theo mối quan hệ: khoảng cách giữa hai điểm xa nhất của hai cụm. Thuật toán sẽ tính khoảng cách giữa các cụm với nhau, sau đó xét các cặp cụm, cụm nào có điểm của hai cụm có khoảng cách nhỏ nhất, thuật toán sẽ gom cụm hai cụm đó lại và tiến hành tính lại khoảng cách giữa các cụm. Thuật toán sẽ làm tương tự cho đến khi chỉ còn một cụm chứa tất cả các đối tượng.

Ví dụ:

|  |  |  |  |
| --- | --- | --- | --- |
|  | A | B | C |
| A | 0 |  |  |
| B | 6 | 0 |  |
| C | 5 | 7 | 0 |

Bước 1: Xét thấy cụm A và cụm C có khoảng cách nhỏ nhất. Tiến hành gom cụm A và C lại và thực hiện tính khoảng cách giữa các cụm mới.

B 🡪 AC = max(B 🡪 A, B 🡪 C) = max(6, 7) = 7

|  |  |  |
| --- | --- | --- |
|  | A,C | B |
| A,C | 0 |  |
| B | 7 | 0 |

Bước 2: Xét thấy cụm AC và cụm B có khoảng cách nhỏ nhất. Tiến hành gom cụm AC và B lại và thực hiện khoảng cách giữa các cụm mới.

|  |  |
| --- | --- |
|  | A,B,C |
| A,B,C | 0 |

Bước 3 Kiếm tra số cụm thành một thì dừng lại

#### 2.1.4.3 Average link

Average link là mối liên kết giữa các cụm theo mối quan hệ: khoảng cách trung bình giữa các điểm của hai cụm. Thuật toán sẽ tính khoảng cách giữa các cụm với nhau, sau đó xét các cặp cụm, cụm nào có điểm của hai cụm có khoảng cách nhỏ nhất, thuật toán sẽ gom cụm hai cụm đó lại và tiến hành tính lại khoảng cách giữa các cụm. Thuật toán sẽ làm tương tự cho đến khi chỉ còn một cụm chứa tất cả các đối tượng.

Ví dụ:

|  |  |  |  |
| --- | --- | --- | --- |
|  | A | B | C |
| A | 0 |  |  |
| B | 6 | 0 |  |
| C | 5 | 7 | 0 |

Bước 1: Xét thấy cụm A và cụm C có khoảng cách nhỏ nhất. Tiến hành gom cụm A và C lại và thực hiện tính khoảng cách giữa các cụm mới.

B 🡪 AC =

=

= = 6.5

|  |  |  |
| --- | --- | --- |
|  | A,C | B |
| A,C | 0 |  |
| B | 6.5 | 0 |

Bước 2: Xét thấy cụm AC và cụm B có khoảng cách nhỏ nhất. Tiến hành gom cụm AC và B lại và thực hiện khoảng cách giữa các cụm mới.

|  |  |
| --- | --- |
|  | A,B,C |
| A,B,C | 0 |

Bước 3 Kiếm tra số cụm thành một thì dừng lại

#### 2.1.4.4 Ward

Ward là mối liên kết giữa các cụm theo phương pháp phương sai tối thiểu. Thuật toán sẽ tính khoảng cách trung bình giữa các điểm trong cụm, sau đó dùng phép tính khoảng cách Euclid để tính khoảng cách giữa các cụm với nhau, sau đó xét các cặp cụm, cụm nào có điểm của hai cụm có khoảng cách nhỏ nhất, thuật toán sẽ gom cụm hai cụm đó lại và tiến hành tính lại khoảng cách giữa các cụm. Thuật toán sẽ làm tương tự cho đến khi chỉ còn một cụm chứa tất cả các đối tượng.

Ví dụ:

|  |  |  |  |
| --- | --- | --- | --- |
|  | A | B | C |
| A | 0 |  |  |
| B | 9 | 0 |  |
| C | 6 | 15 | 0 |

Bước 1: Xét thấy cụm A và cụm C có khoảng cách nhỏ nhất. Tiến hành gom cụm A và C lại và thực hiện tính khoảng cách giữa các cụm mới.

B 🡪 AC =

=

= = 3 + 5 – 2 = 6

|  |  |  |
| --- | --- | --- |
|  | A,C | B |
| A,C | 0 |  |
| B | 6 | 0 |

Bước 2: Xét thấy cụm AC và cụm B có khoảng cách nhỏ nhất. Tiến hành gom cụm AC và B lại và thực hiện khoảng cách giữa các cụm mới.

|  |  |
| --- | --- |
|  | A,B,C |
| A,B,C | 0 |

Bước 3 Kiếm tra số cụm thành một thì dừng lại

### 2.1.5 Ứng dụng

Hierarchical Clustering được ứng dụng vào nhiều lĩnh vực khác nhau như: Phân tích gom cụm các khuôn mặt có đặc điểm giống nhau, gom cụm mặt hàng có đặc điểm giống nhau,… Ngoài ra ta có thể sử dụng Thuật toán này cho giao hàng, thuật toán có thể thu thập địa chỉ sau đó phần tích đường đi và sau đó có thể phân cụm thành những cụm nhỏ, mối cụm cho từng Shipper đi giao hàng.

### 2.1.6 So sánh với các thuật toán khác

- Flat clustering rả về một tập hợp các cụm không có cấu trúc phẳng, yêu cầu một số lượng xác định trước các cụm làm đầu vào và không xác định. Hierarchical Clustering xuất ra một cấu trúc phân cấp, một cấu trúc có nhiều thông tin hơn so với tập hợp các cụm không có cấu trúc được trả về bởi Flat clustering.

- Hierarchical Clustering không yêu cầu chúng ta xác định trước số lượng các cụm và hầu hết các thuật toán phân cấp đã được sử dụng trong IR là xác định. Những lợi thế của Hierarchical Clustering đi kèm với chi phí là hiệu quả thấp hơn. Các thuật toán Hierarchical Clustering phổ biến nhất có độ phức tạp ít nhất là bậc hai về số lượng tài liệu so với độ phức tạp tuyến tính của K-means và EM

## 2.2 Python

### 2.2.1 Giới thiệu về Python

Python là ngôn ngữ lập trình hướng đối tượng, cấp cao, mạnh mẽ, được tạo ra bởi Guido van Rossum. Nó dễ dàng để tìm hiểu và đang nổi lên như một trong những ngôn ngữ lập trình nhập môn tốt nhất cho người lần đầu tiếp xúc với ngôn ngữ lập trình. Python hoàn toàn tạo kiểu động và sử dụng cơ chế cấp phát bộ nhớ tự động. Python có cấu trúc dữ liệu cấp cao mạnh mẽ và cách tiếp cận đơn giản nhưng hiệu quả đối với lập trình hướng đối tượng. Cú pháp lệnh của Python là điểm cộng vô cùng lớn vì sự rõ ràng, dễ hiểu và cách gõ linh động làm cho nó nhanh chóng trở thành một ngôn ngữ lý tưởng để viết script và phát triển ứng dụng trong nhiều lĩnh vực, ở hầu hết các nền tảng.

### 2.2.2 Scikit Learn AgglomerativeClustering

Scikit-learn (Sklearn) là thư viện mạnh mẽ nhất dành cho các thuật toán học máy được viết trên ngôn ngữ Python. Thư viện cung cấp một tập các công cụ xử lý các bài toán machine learning và statistical modeling gồm: classification, regression, clustering, và dimensionality reduction.

AgglomerativeClustering là đối tượng trong được định nghĩa trong thư viện Sklearn. Đây là một trong hai phương pháp phân cụm của giải thuật Hierarchical Clustering. Đối tượng có các tham số như n\_clusters để xác định số lượng cụm, affinity là metric được sử dụng để tính toán linkage. Gồm có các loại affinity như: euclidean, l1, l2, manhattan, cosine, precomputed. Có điều đặc biệt là nếu linkage là “ward” thì affinity phải là euclidean, linkage: là tiêu chí liên kết, được dùng để xác định khoảng cách giữa các bộ quan sát. Thuật toán sẽ gộp các cặp cụm có khoảng cách ngắn nhất gồm có các linkage như: ward, complete, average, single, mặc định là ward

### 2.2.3 Numpy

NumPy là một thư viện Python được sử dụng để làm việc với các mảng. Nó cũng có các chức năng để làm việc trong miền đại số tuyến tính, biến đổi fourier và ma trận. NumPy được tạo ra vào năm 2005 bởi Travis Oliphant. Nó là một dự án mã nguồn mở và bạn có thể sử dụng nó một cách tự do. NumPy là viết tắt của Numerical Python. Trong Python, chúng ta có danh sách phục vụ mục đích của mảng, nhưng chúng xử lý chậm. NumPy nhằm mục đích cung cấp một đối tượng mảng nhanh hơn tới 50 lần so với danh sách Python truyền thống. Đối tượng mảng trong NumPy được gọi ndarray, nó cung cấp rất nhiều chức năng hỗ trợ giúp làm việc với nó ndarrayrất dễ dàng. Mảng được sử dụng rất thường xuyên trong khoa học dữ liệu, nơi mà tốc độ và tài nguyên là rất quan trọng.

### 2.2.4 Pandas

Pandas là một thư viện mã nguồn mở, hỗ trợ đắc lực trong thao tác dữ liệu. Đây cũng là bộ công cụ phân tích và xử lý dữ liệu mạnh mẽ của ngôn ngữ lập trình python. Thư viện này được sử dụng rộng rãi trong cả nghiên cứu lẫn phát triển các ứng dụng về khoa học dữ liệu. Thư viện này sử dụng một cấu trúc dữ liệu riêng là Dataframe. Pandas cung cấp rất nhiều chức năng xử lý và làm việc trên cấu trúc dữ liệu này. Chính sự linh hoạt và hiệu quả đã khiến cho pandas được sử dụng rộng rãi. Pandas nhằm mục đích trở thành khối xây dựng cấp cao cơ bản để thực hiện phân tích dữ liệu thực tế, thế giới thực bằng Python. Ngoài ra, nó có mục tiêu rộng hơn là trở thành công cụ phân tích / thao tác dữ liệu nguồn mở mạnh mẽ và linh hoạt nhất có sẵn bằng bất kỳ ngôn ngữ nào.

### 2.2.5 Flask

Flask là một web frameworks, nó thuộc loại micro-framework được xây dựng bằng ngôn ngữ lập trình Python. Flask cho phép bạn xây dựng các ứng dụng web từ đơn giản tới phức tạp. Nó có thể xây dựng các api nhỏ, ứng dụng web chẳng hạn như các trang web, blog, trang wiki hoặc một website dựa theo thời gian hay thậm chí là một trang web thương mại. Flask cung cấp cho bạn công cụ, các thư viện và các công nghệ hỗ trợ bạn làm những công việc trên.

### 2.2.6 Apscheduler

Advanced Python Scheduler (APScheduler) là một thư viện Python cho phép bạn lập lịch trình mã Python của mình được thực thi sau này, chỉ một lần hoặc định kỳ. Bạn có thể thêm công việc mới hoặc xóa nhanh công việc cũ tùy ý. Nếu lưu trữ các công việc trong cơ sở dữ liệu, chúng cũng sẽ tồn tại khi trình lập lịch khởi động lại và duy trì trạng thái của chúng. Khi bộ lập lịch được khởi động lại, sau đó nó sẽ chạy tất cả các công việc mà lẽ ra nó phải chạy trong khi ngoại tuyến

# Chương 3 GIỚI THIỆU SẢN PHẨM

## 3.1 Tổng quan hệ thống

## 3.2 Thành phần và chức năng

### 3.2.1 Hàm Fetch Api Từ Server

def getalldata():

    url = "http://servertlcn.herokuapp.com/diachi/search"

    r = requests.post(url, json=setjsontime(timerun, timeafter))

    return r.json()

|  |  |  |  |
| --- | --- | --- | --- |
| Tên hàm | Input | Output | Chức Năng |
| getalldata | url: api  timerun: thời gian bắt đầu  timeafter: thời gian kết thúc | Chuỗi Json | Fetch data từ dữ liệu từ server theo thời gian mong muốn |

Bảng 1 getalldatda

### 3.2.2 Hàm Chuyển Json sang Numpy Array

def getjsontoarray():

    X = np.array([[106,10]])

    url = ulrpage+"/diachi/search"

    r = requests.post(url, json={

        "id":1,

        "timerange":{

                "timeStart":"12/10/2020 "+tstart,

                "timeEnd":"12/10/2020 "+tend

                }

        })

    data=r.json()

    for i in range(0,len(data['data'])):

        X=np.concatenate([X,[[float(data['data'][i]['KinhDo']),

                              float(data['data'][i]['ViDo'])]]])

    X=np.delete(X,0,axis = 0)

    return X

|  |  |  |  |
| --- | --- | --- | --- |
| Tên hàm | Input | Output | Chức Năng |
| getjsontoarray | url: api  timerun: thời gian bắt đầu  timeafter: thời gian kết thúc | X: mảng numpy array | Chuyển chuỗi Json lấy từ server chuyển sang mảng hai chiều |

Bảng 2 getjsontoarray

### 3.2.3 Hàm kiểm tra Kích thước

def checkSizeOrder(json):

    clusternumber=cluster\_number

    for i in range(0,cluster\_number):

        soluong=0

        for j in range(0,len(json[i])):

            soluong+=getSizeSanPhamByIdDonHang(json[i][j]['DonhangId'])

        if soluong>size:

            clusternumber+=1

    if(soluongdon<clusternumber): return True

    if(clusternumber==cluster\_number): return True

    else: return False

|  |  |  |  |
| --- | --- | --- | --- |
| Tên hàm | Input | Output | Chức Năng |
| checkSizeOrder | json: Chuỗi json sau khi phân cụm  size: Kích thước ban đầu | True, False | Kiểm tra kích thước đơn hàng từng cụm xem có tổng kích thước có vừa với kích thước ban đầu không |

Bảng 3 checkSideOder

### 3.2.4 Hàm Phân Cụm

def Cluster(A):

    if(len(A)>1):

        hc=AgglomerativeClustering(n\_clusters=cluster\_number,

                                   affinity='euclidean',

                                   linkage='ward')

        return hc.fit\_predict(A)

|  |  |  |  |
| --- | --- | --- | --- |
| Tên hàm | Input | Output | Chức Năng |
| Cluster | A: mảng array numpy được chuyển từ Json trên server  linkage='ward': liên kết  n\_clusters=cluster\_number: số cụm  affinity='euclidean' phương thức tính toán | hc.fit\_predict(A): mảng ở dạng label chứa các số từ 0,1,2,… tương ứng với số cụm | Kiểm tra kích thước đơn hàng từng cụm xem có tổng kích thước có vừa với kích thước ban đầu không |

Bảng 4 Cluster

### 3.2.5 Hàm tính kích thước sản phẩm trong đơn hàng

def getSizeSanPhamByIdDonHang(id):

    url="http://servertlcn.herokuapp.com/dssanpham/"+str(id)+"/donhang"

    r=requests.get(url)

    data=r.json()

    soluong=0

    index=data['data']

    for i in range(0,len(index)):

        soluong+=float(index[i]['SanPham']['KichCo'])

    return soluong

|  |  |  |  |
| --- | --- | --- | --- |
| Tên hàm | Input | Output | Chức Năng |
| getSizeSanPhamByIdDonHang | url: api  id: id của đơn hàng | soluong: tổng kích thước của các sản phẩm trong một đơn hàng | Tính toán tổng kích thước của các sản phẩm trong một đơn hàng |

Bảng 5 getSizeSanPhamByIdDonHang

### 3.2.6 Hàm chuyển từ Numpy Array sang Json

def outputdatatojson():

    data=createjson()

    outputdata=data['data']

    cluster=0

    while(cluster<cluster\_number):

        for j in range(0,len(A[y\_hc==cluster])):

                for i in range(0,len(TempA)):

                    if(A[y\_hc==cluster][j][0]==TempA[i][0] and A[y\_hc==cluster][j][1]==TempA[i][1]):

                        outputdata[cluster].append(datatemp[i])

        cluster+=1

    return outputdata

|  |  |  |  |
| --- | --- | --- | --- |
| Tên hàm | Input | Output | Chức Năng |
| outputdatatojson | A: mảng array numpy  y\_hc: mảng label chứa các giá trị 0,1,2… sau khi phân cụm | Outputdata: Chuỗi Json | Chuyển mảng array numpy sau khi được phân cụm sẽ được chuyển lại thành Json |

Bảng 6 outputdatatojson

### 3.2.7 Hàm Một Nhánh

def Motnhanh():

    b='{"data": []}'

    b=json.loads(b)

    alldata=getalldata()

    global soluongdon

    soluongdon=len(alldata['data'])

    b=b['data']

    b.append(alldata['data'])

    return b

|  |  |  |  |
| --- | --- | --- | --- |
| Tên hàm | Input | Output | Chức Năng |
| Motnhanh | alldata:chuỗi json lấy từ server | b: chuỗi Json được bỏ vào một cụm | Chuyển các đơn hàng trên server vào một cụm |

Bảng 7 Motnhanh

### 3.2.8 Hàm Xử lý phân cụm

def runcluster():

    data=getalldata()

    if(checkresult(data)==False):

        json=createjson()

        outputdata=json['data']

        del outputdata[1]

        del outputdata[0]

        return outputdata,0

    else:

        global A

        A=getjsontoarray()

        global soluongdon

        soluongdon=len(A)

        global TempA

        TempA=getjsontoarray()

        if(len(A)<2):

            json=createjson()

            outputdata=json['data']

            outputdata[0].append(data['data'][0])

            del outputdata[1]

            return outputdata,0

        else:

            global y\_hc

            y\_hc=Cluster(A)

            global datatemp

            datatemp=data['data']

            j=outputdatatojson()

            return j,1

|  |  |  |  |
| --- | --- | --- | --- |
| Tên hàm | Input | Output | Chức Năng |
| runcluster | data: chuỗi json được lấy từ server  A, tempA: mảng array numpy | j:chuỗi json đã được phân cụm | Lấy data json từ server về, sau đó chuyển sang mảng và phân cụm, sau đó chuyển ngược lại thành json đã được phân cụm |

Bảng 8 runcluster

### 3.2.9 Hàm chạy tất cả các xử lý phân cụm

def runall():

    b=Motnhanh()

    if(checkSizeOrder(b) == True):

        return b

    else:

        j,status=runcluster()

        if(status==0): return j

        else:

            while checkSizeOrder(j) == False:

                global cluster\_number

                cluster\_number+=1

                j,status=runcluster()

        global finaljson

        finaljson=j

        return j

|  |  |  |  |
| --- | --- | --- | --- |
| Tên hàm | Input | Output | Chức Năng |
| runall | b: chuyển về một cụm  runcluster() | j:chuỗi json đã được phân cụm | Phân cụm sau đó kiểm tra xem kích thước đơn hàng trong cụm đó có phù hợp với kích thước ban đầu không |

Bảng 9 runall

### 3.2.10 Hàm post Json lên server

def getlapchuoidonhang(j):

    url = "http://servertlcn.herokuapp.com/lapchuoidonhang/donhang"

    r = requests.post(url, json={'timeStart':timestartdefault.strftime("%H:%M"),

                                 'data':j,'plus':plustime})

    data=r.json()

    return data['result']

|  |  |  |  |
| --- | --- | --- | --- |
| Tên hàm | Input | Output | Chức Năng |
| getlapchuoidonhang | j: chuỗi json đã được phân cụm và kiểm tra kích thước  timestartdefault: thời gian bắt đầu | data[‘result’]: tình trạng post lên server | Post Json lên server |

Bảng 10 getlapchuoidonhang

### 3.2.11 Hàm lập lịch trình theo thời gian

def loopHourInDay():

    timerun=timestartdefault

    timeafter = fplustime(timerun,plustime)

    solan=0

    while solan < timesloop:

        j=runall()

        check=getlapchuoidonhang(j)

        if(check=="ok" or check=="fail"):

            timerun=timeafter

            timeafter = fplustime(timerun,plustime)

            solan+=1

|  |  |  |  |
| --- | --- | --- | --- |
| Tên hàm | Input | Output | Chức Năng |
| loopHourInDay | timestartdefault: thời gian bắt đầu  timeloop: số lần lặp trong ngày |  | Thực hiện phân cụm theo thời gian, và lặp lại theo số lần giao cho trước |

Bảng 11 loopHourInDay

### 3.2.12 Hàm lấy và xử lý thời gian

def gettime(timestart,pltime,plday,timesloops):

    global timestartdefault

    timestartdefault=datetime.strptime(str(date.strftime("%m/%d/%Y")+" " +timestart),'%m/%d/%Y %H:%M')

    global plustime

    plustime=int(pltime)

    global plusday

    plusday=int(plday)

    global timesloop

    timesloop=int(timesloops)

|  |  |  |  |
| --- | --- | --- | --- |
| Tên hàm | Input | Output | Chức Năng |
| gettime | timestart: thời gian bắt đầu  timeloop: số lần lặp trong ngày |  | Thực hiện phân cụm theo thời gian, và lặp lại theo số lần giao cho trước |

Bảng 12 gettime

# Chương 4 CÀI ĐẶT VÀ KIỂM THỬ

## 4.1 Cài Đặt

### 4.1.1 Yêu cầu tiên quyết

* Cài đặt Anaconda

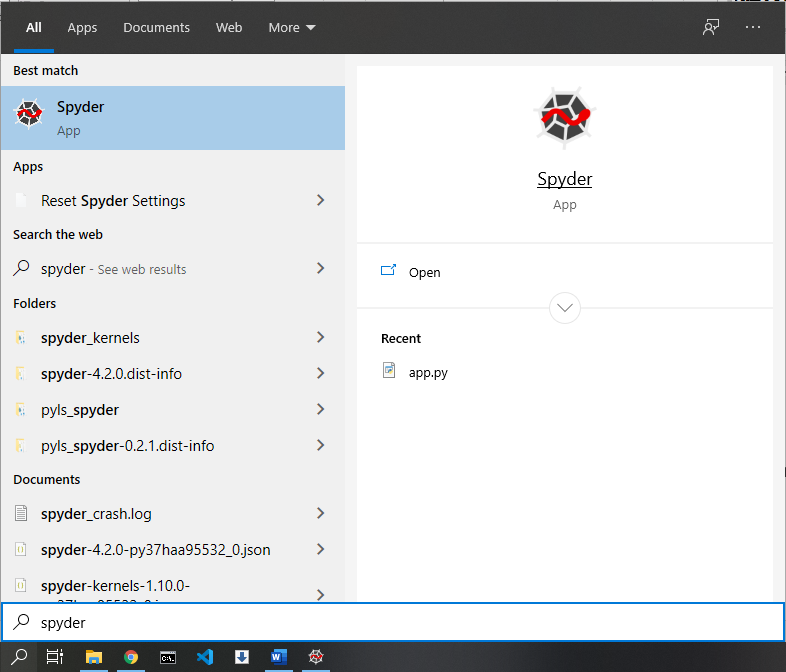
### 4.1.2 Các bước cài đặt

Bước 1: Clone hoặc download project từ link github:

<https://github.com/Caoquyen1913/DoAn3>

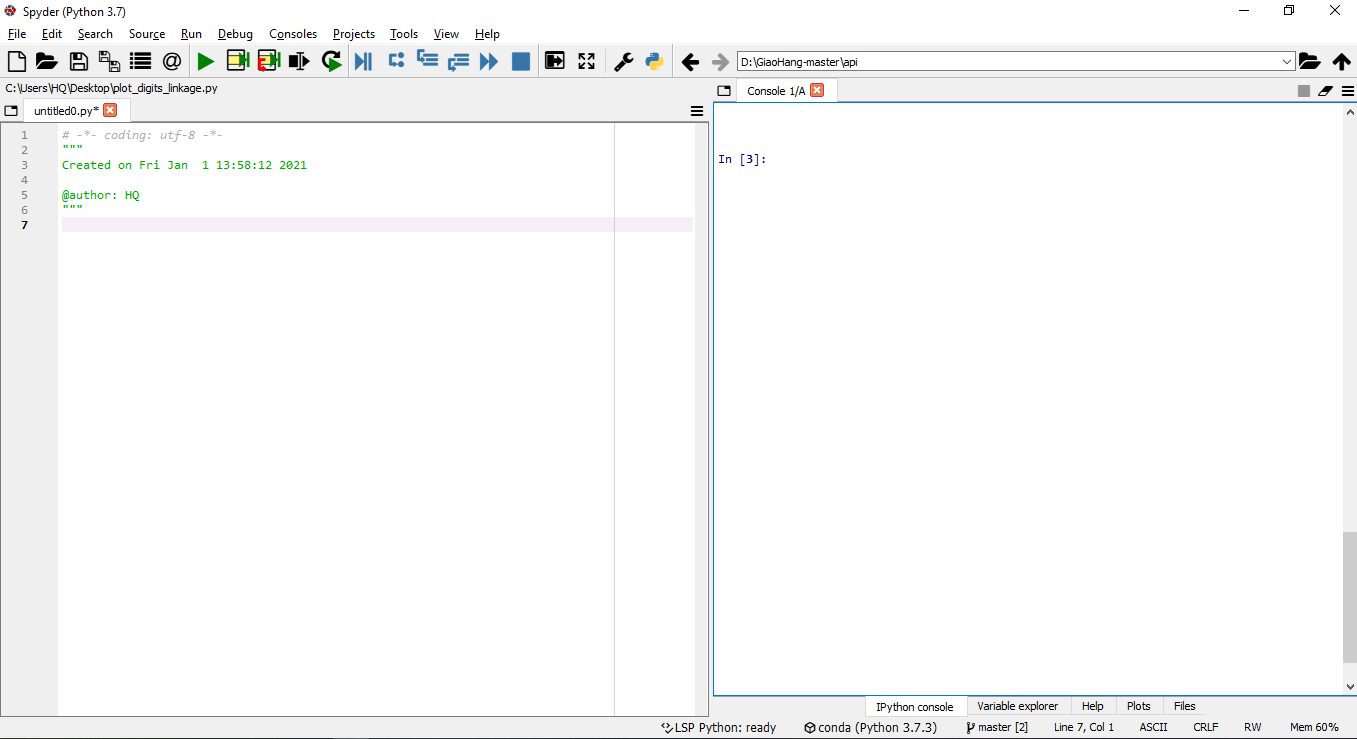
Bước 2: Một thư mục có tên là “DoAn3-main” sẽ xuất hiện sau khi clone project từ github hoặc giải nén file đính kèm báo cáo.

Bước 3: Ấn tổ hợp phím Win+Q để mở Window tìm kiếm, nhập Spyder, rồi chọn ứng dụng Spyder

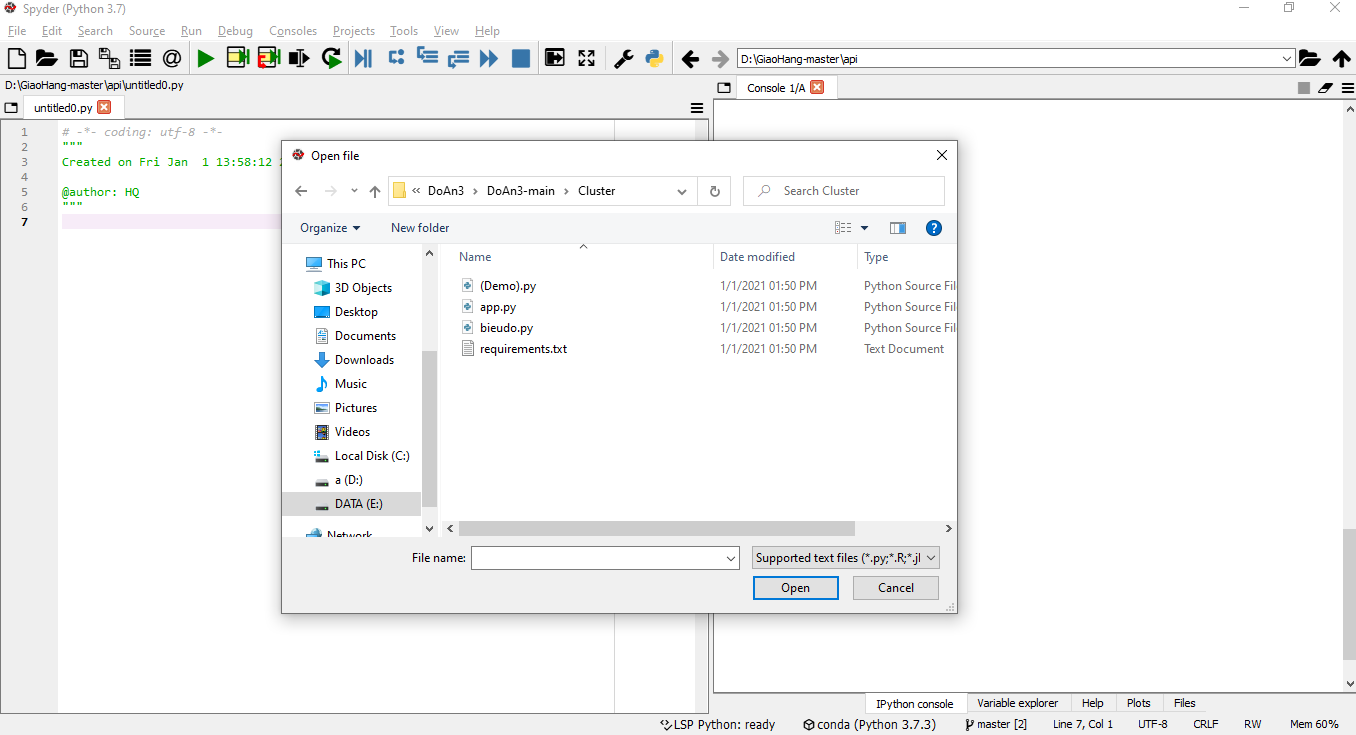


Hình 1

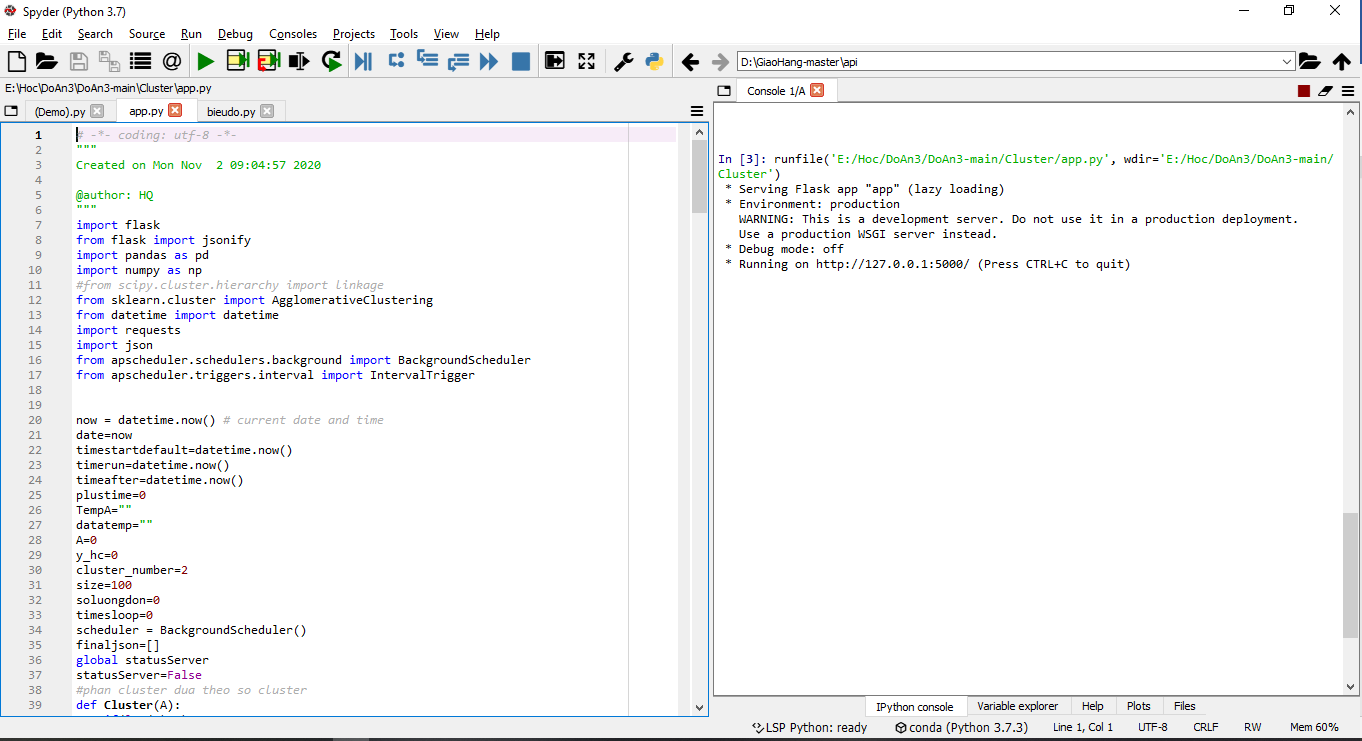
Bước 4: Mở ứng dụng Spyder



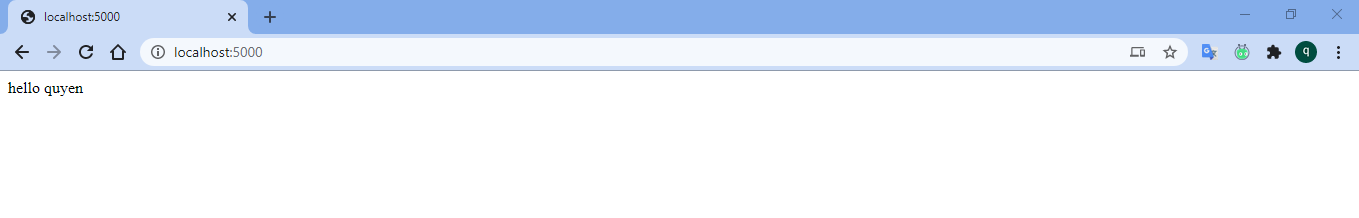
Bước 5: nhấn tổ hợp phìm Ctrl+O để mở file, chuyển đường dẫn tới thư mục “DoAn3-main” chọn thư mục “Cluster” và mở 3 file .py



Bước 6: Chạy file app.py (nhấn F5 để tiến hành run)



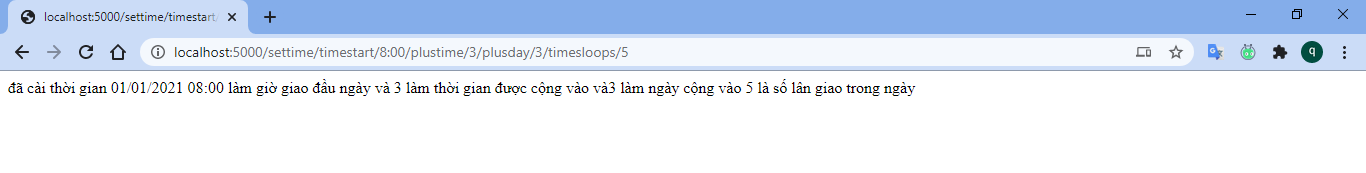
Bước 7: Vào trình duyệt web nhập URL: <http://localhost:5000/>



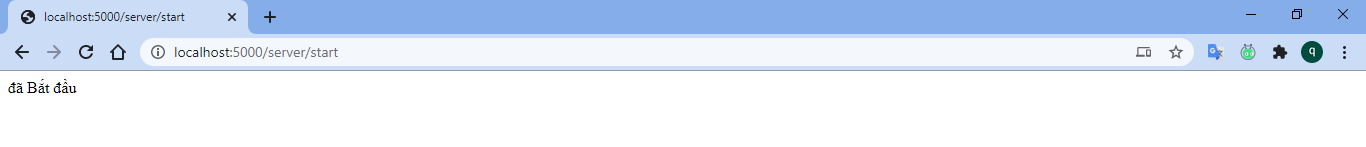
Bước 8: Nhập URL:

<http://localhost:5000/settime/timestart/8:00/plustime/3/plusday/3/timesloops/5>

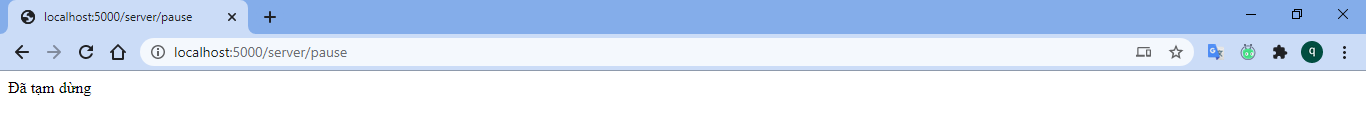
Trong đó timestart là thời gian bắt đầu, plustime là thời gian cộng vào (8:00 đến 11:00) plusday là số ngày được cộng vào mà server sẽ tính(ví dụ hôm nay là 01-01-2021 thì server sẽ tính đơn hàng của ngày 04-01-2021), timesloops là số lân giao trong ngày.



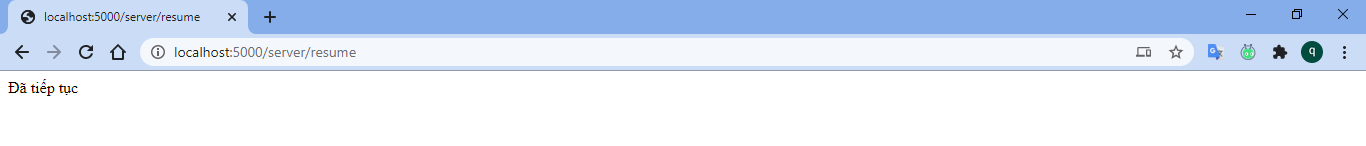
Bước 9: Nhập URL: <http://localhost:5000/server/start> để chạy server



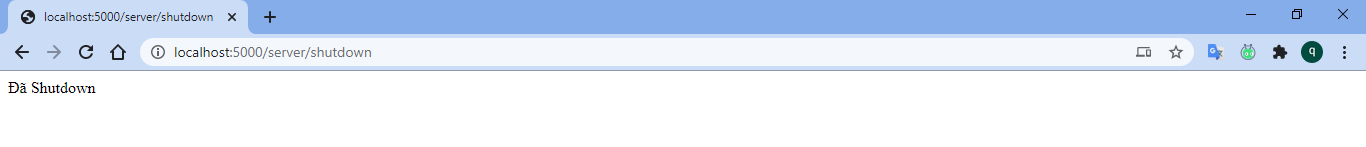
Để tạm dừng server nhập URL: <http://localhost:5000/server/pause>



Để tiếp tục chạy server nhập URL: <http://localhost:5000/server/resume>



Để tắt server nhập URL: <http://localhost:5000/server/shutdown>



### 4.1.3 Nâng cao

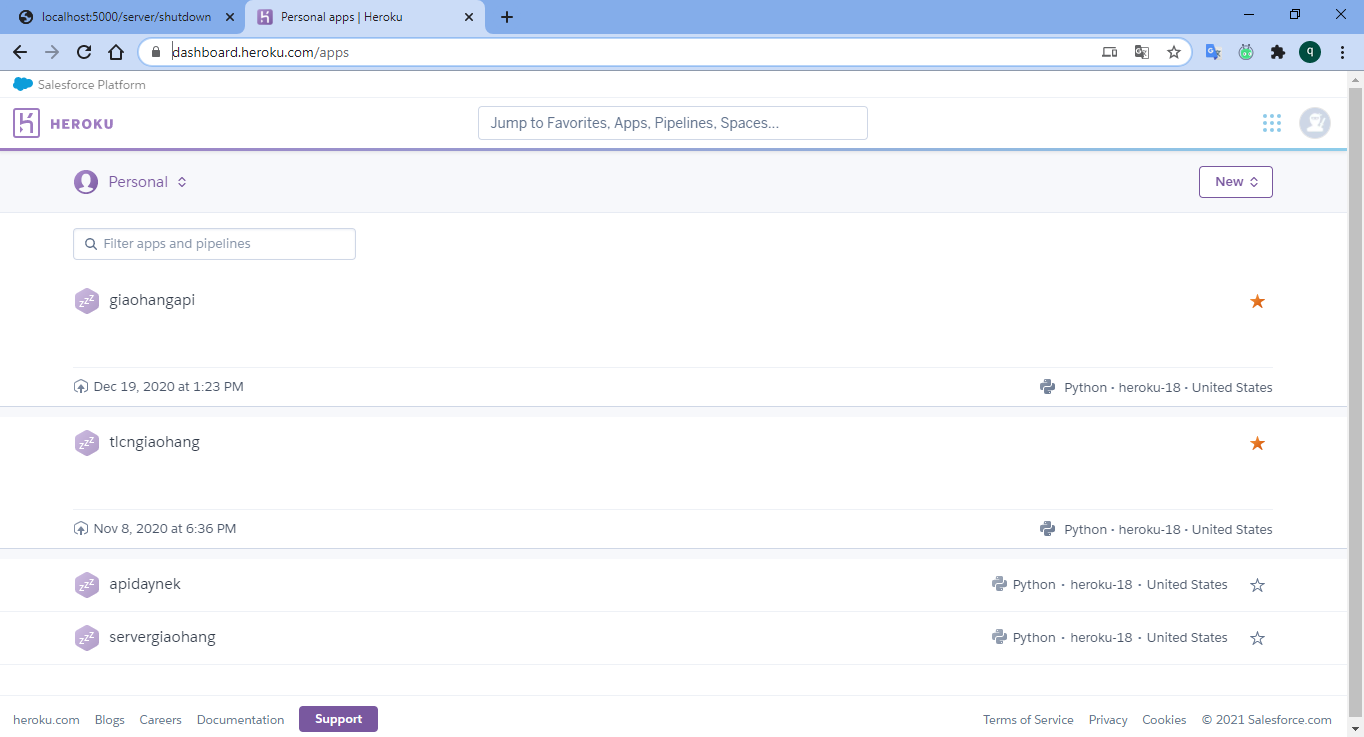
Để Deploy lên Heroku(cloud)

* Yêu cầu:
  + đã có tài khoán heroku
  + Tải PowerShell 7

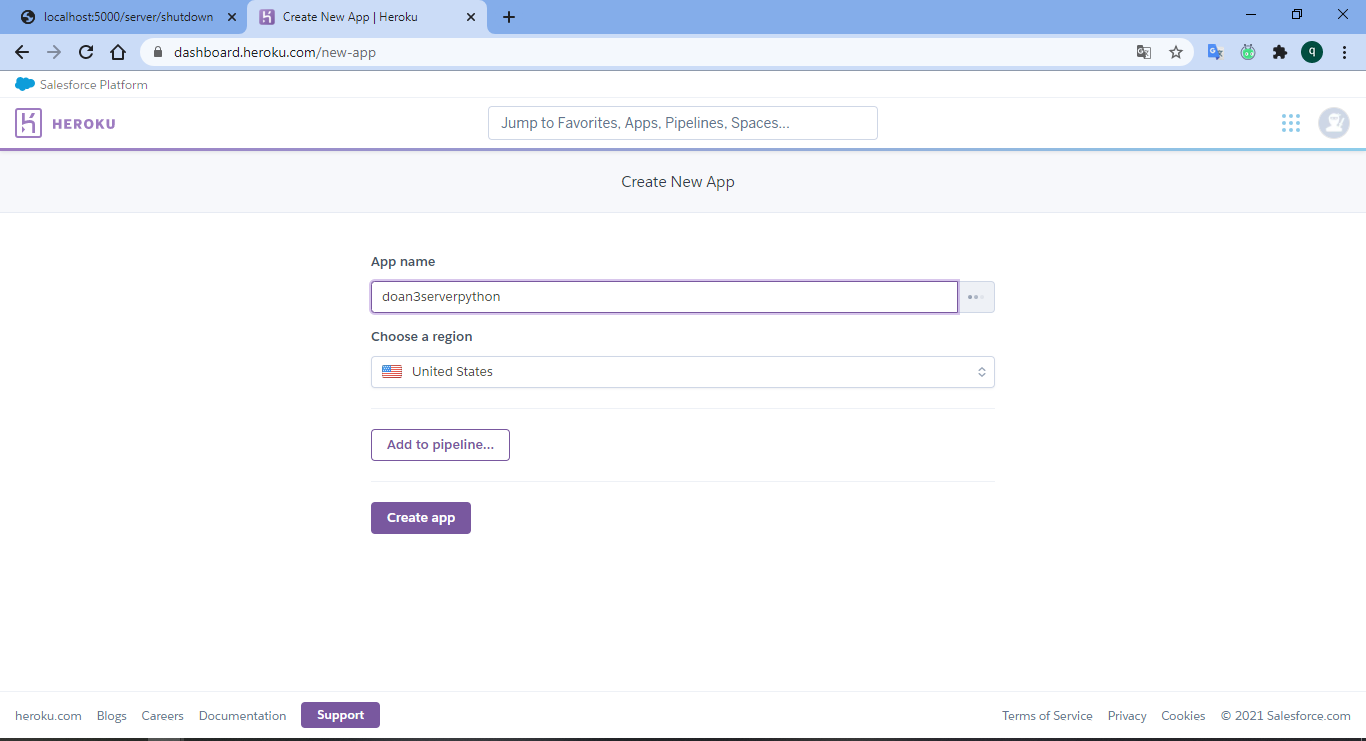
Bước thực hiện:

Làm tương tự bước 1,2

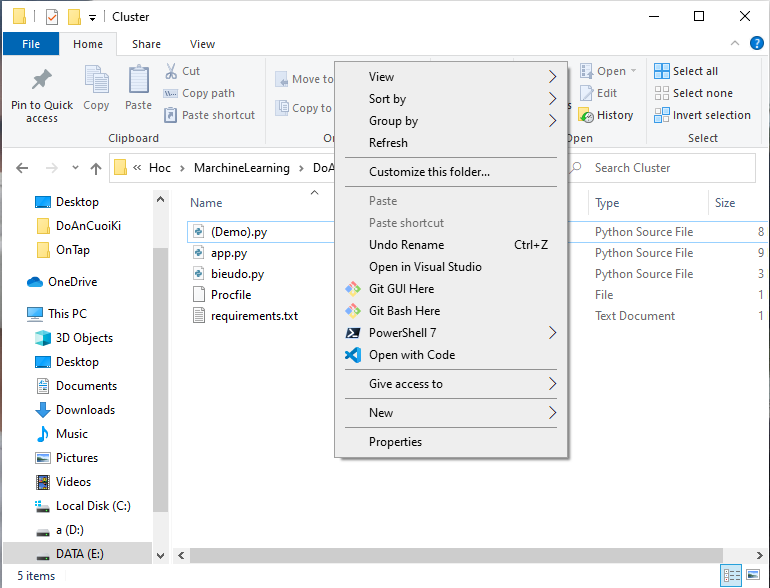
Bước 3: Mở trình duyệt vào URL: <https://dashboard.heroku.com/apps>



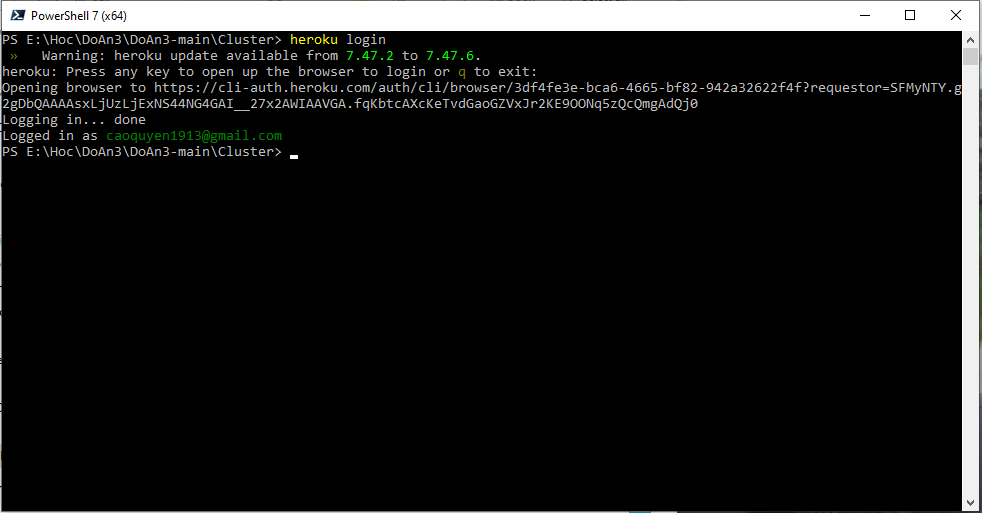
Bước 4: Chọn New 🡪 Create New App, sau đó đặt tên app



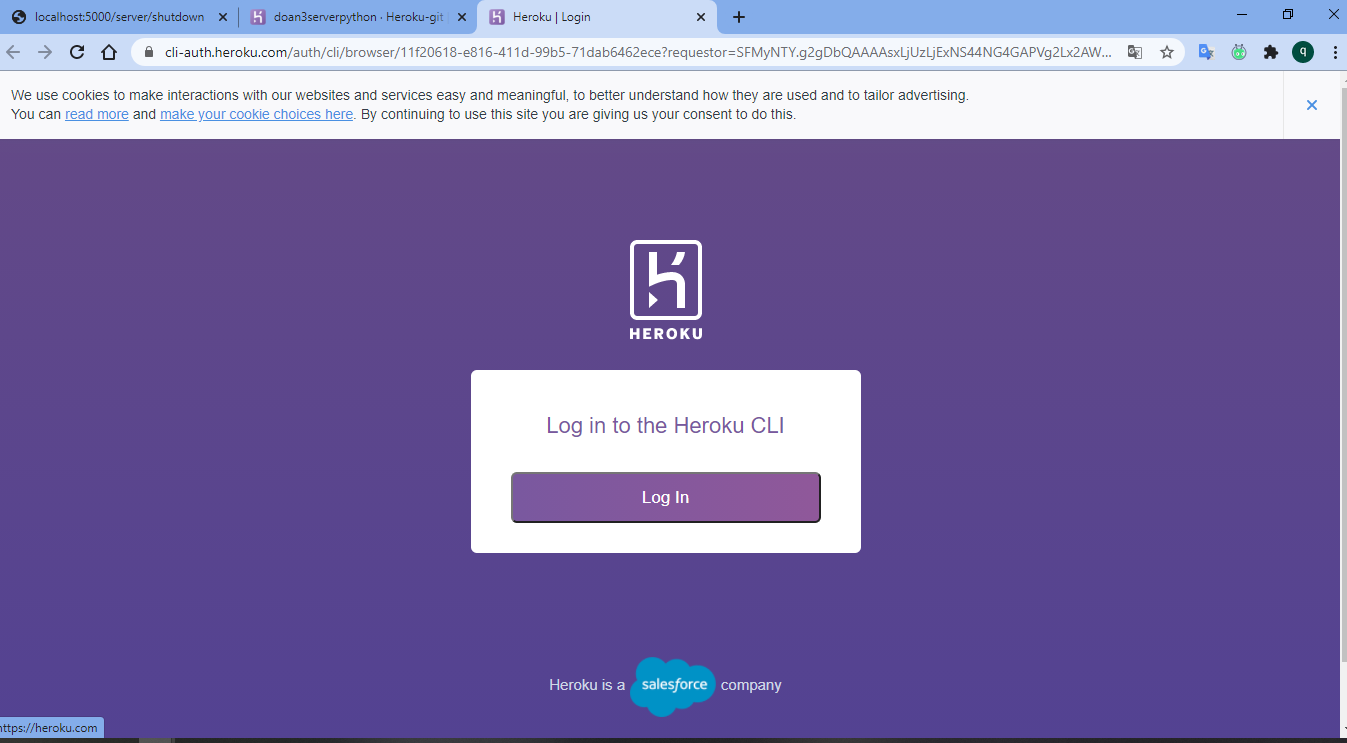
Bước 5: vào thư mục giải nén “DoAn3-main” vào thư mục “Cluset” click chuột phải chọn “PowerShell 7” chọn “Open Here”



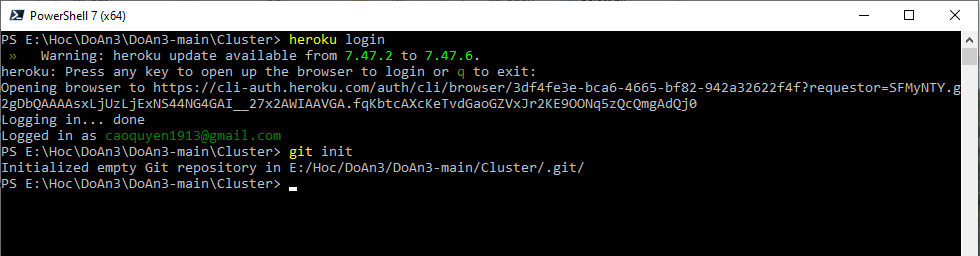
Bước 6: nhập “heroku login” rồi Enter, đợi 1 lúc rồi ấn enter lần nữa:



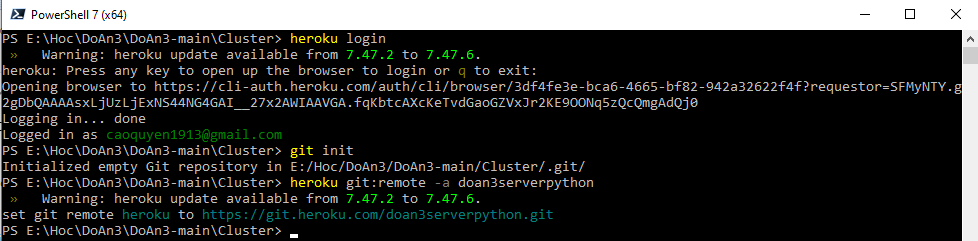
Bước 7: Đợi trình duyệt Web hiện ra chọn login



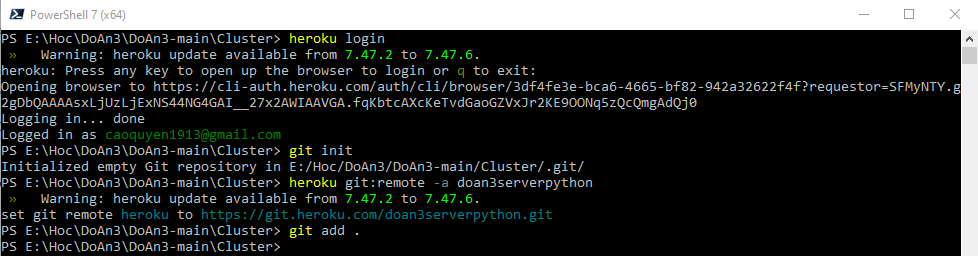
Bước 8: Quay lại PowerShell 7 nhập “git init”



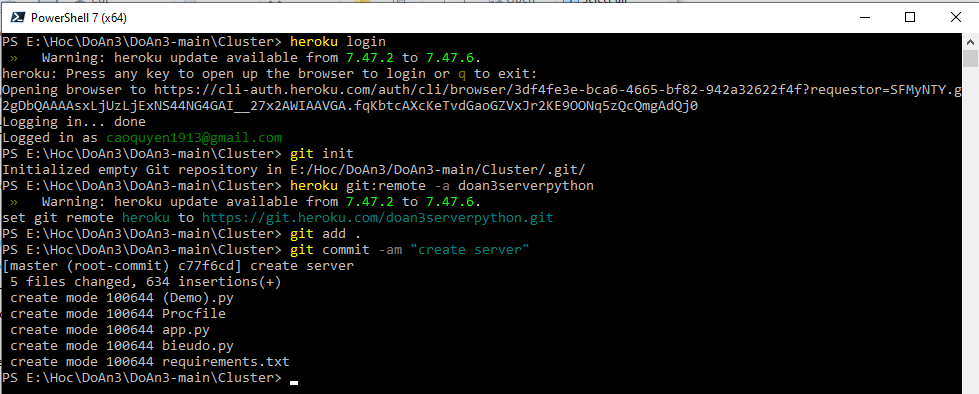
Bước 9: Nhập “heroku git:remote -a doan3serverpython” doan3serverpython là tên app đặt lúc nãy.



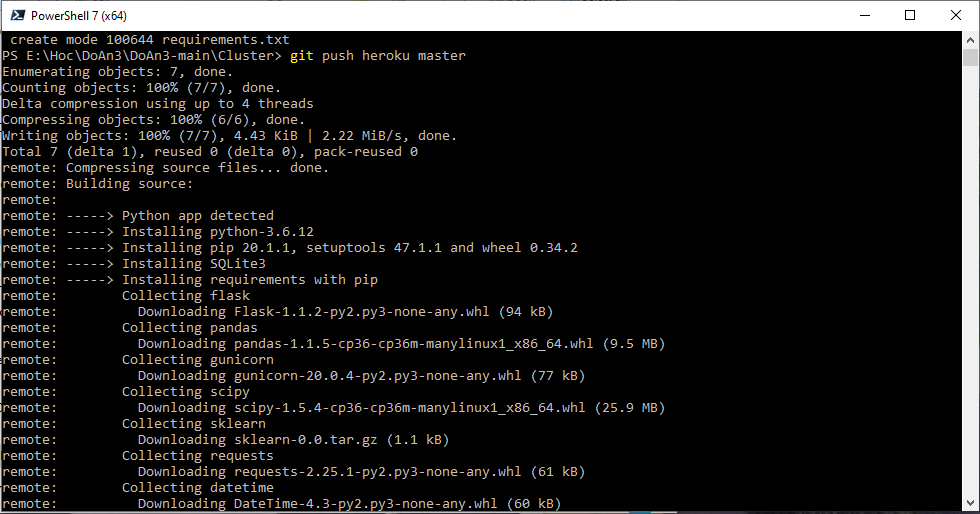
Bước 10: Nhập “git add .”



Bước 11: Nhập “git commit -am “create server”

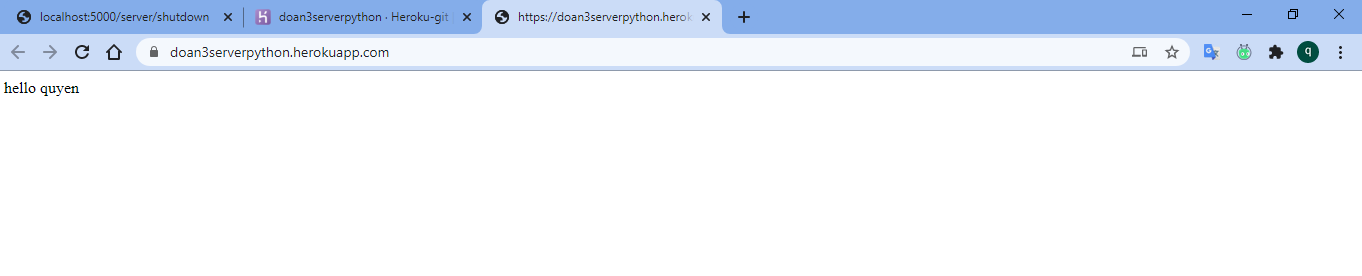


Bước 12: Nhập “git push heroku master” để deploy lên heroku



Bước 13: vào trình duyệt web nhập URL:

<https://doan3serverpython.herokuapp.com/> , doan3serverpython là tên app đặt lúc nãy



Bước 14: Làm như các bước 8-9 như trên phần cài đặt. thay <http://localhost:5000/> bằng [https://doan3serverpython.herokuapp.com/](https://doan3serverpython.herokuapp.com/%20)

## 4.2 Kiểm Thử

# Chương 5 KẾT LUẬN

## 5.1 Kết quả đạt được

* Kỹ năng làm việc nhóm tốt, kỹ năng giao tiếp trong team, phân công công việc khi làm dự án ngắn ngày.
* • Kỹ năng search tài liệu, kỹ năng tìm kiếm giải pháp trên mạng cho những khúc mắc trong quá trình xây dựng đồ án.

## 5.2 Ưu điểm

* Tìm được đường đi ngắn nhất trong từng cụm
* Quản lý, thu thập, phân cụm cho tất cả các đơn hàng mà không đơn hàng nào bị bỏ sót
* Server dễ sử dụng, triển khai tự động và độc lập. Không cần người túc trực để chạy

## 5.3 Khuyết điểm

* Việc phân nhỏ vì không phù hợp kích cỡ của cốp hàng dẫn đến có thể có cụm chỉ có một đơn hàng

# TÀI LIỆU THAM KHẢO

* Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze. (2009). An Introduction to Information Retrieval. Pp. 377- 400
* <https://en.wikipedia.org/wiki/Hierarchical_clustering> - Truy cập ngày 31/12/2020
* <https://en.wikipedia.org/wiki/Single-linkage_clustering> - Truy cập ngày 31/12/2020
* <https://en.wikipedia.org/wiki/Complete-linkage_clustering> - Truy cập ngày 31/12/2020
* <https://en.wikipedia.org/wiki/Ward%27s_method> - Truy cập ngày 31/12/2020