

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC XÃ HỘI VÀ NHÂN VĂN TPHCM
KHOA THƯ VIỆN - THÔNG TIN HỌC

-----<>-----



ĐỒ ÁN CUỐI KỲ

CHẨN ĐOÁN UNG THƯ VÚ BẰNG MÔ HÌNH K-NEAREST NEIGHBORS

Môn: Lưu trữ và khai thác dữ liệu

Giảng viên : Nguyễn Tân Công

Nhóm 1 – Lớp Quản lý thông tin B

STT	Họ và tên	Mã số sinh viên
1	Hoàng Xuân Quốc	2156210125
2	Đặng Hoàng Chiến	2156210095
3	Nguyễn Viết Đức	2156210100



MỤC LỤC

DANH MỤC HÌNH	4
DANH MỤC BẢNG	5
I. TỔNG QUAN.....	6
1. Giới thiệu nhóm /lý do chọn dự án.....	6
2. Sơ đồ tổng quan trình bày (phân rã chức năng)	6
3. Công cụ thực hiện	8
3.1 Giới thiệu Python.....	8
3.2 Các thư viện sử dụng	9
3.3 Tổng quan về dữ liệu sử dụng	11
4. Thông tin dự án của nhóm và phân chia nhiệm vụ.....	11
II. LÝ THUYẾT THUẬT TOÁN	12
1. Giới thiệu về data mining và bài toán phân loại/phân cụm/tìm luật kết hợp.	12
1.1 Giới thiệu về data mining	12
1.2 Bài toán phân loại/phân cụm/tìm tập kết hợp	13
2. Thuật toán K – nearest-neighbour.....	16
2.1 Giới thiệu về thuật toán và bài toán mà thuật toán giải quyết	16
2.1.1 Tổng quan về thuật toán	16
2.1.2 Vấn đề mà bài toán giải quyết	17
2.2 Đặc trưng của thuật toán	18
2.3 Nguyên lý thuật toán.....	20
2.4 Minh họa thuật toán	24
2.4.1 Dữ liệu	24
2.4.2 Các bước thực hiện	25
2.4.3 Kết quả	28
III. CHUẨN BỊ DỮ LIỆU	28
1. Giới thiệu dữ liệu	28
2. Tiền xử lý dữ liệu	30
2.1. Làm sạch dữ liệu	30
2.2. Biến đổi dữ liệu	31



3. Mô tả dữ liệu (thông kê mô tả)	32
IV. KẾT QUẢ VÀ ĐÁNH GIÁ THUẬT TOÁN	36
1. Xây dựng thuật toán.....	36
2. Áp dụng thuật toán cho bài toán.....	37
3. Thủ nghiệm thuật toán.....	38
3.1. Thủ nghiệm với sự thay đổi của k.....	38
3.2. Thủ nghiệm với sự thay đổi của p.....	39
4. Đánh giá và so sánh thuật toán	40
4.1 So sánh thuật toán	40
4.2 Đánh giá hiệu suất thuật toán	40
5. Tối ưu hóa siêu tham số:	42
V. GIAO DIỆN DEMO THUẬT TOÁN	44
1. Giới thiệu công cụ và trang web demo	44
2. Phương thức và hàm được sử dụng	44
3. Kết quả thực hiện	45
3.1 Giao diện trang overview	45
3.2 Giao diện trang descriptive statistics	46
3.3 Giao diện trang model	46
3.4 Giao diện trang contact.....	47
VI. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN.....	47
1. Kết luận	47
2. Hướng phát triển	48
VII. TÀI LIỆU THAM KHẢO.....	49
VIII. PHỤ LỤC – CÁC ĐƯỜNG DẪN QUAN TRỌNG.....	49



DANH MỤC HÌNH

Hình 1. 1 Sơ đồ phân rã chức năng (link xem: full_image_bfd).....	7
Hình 1. 2 Cấu trúc thư mục dự án Github	12
Hình 1. 3 Sơ đồ nguyên lý	21
Hình 1. 4 Minh họa K.....	24
Hình 3. 1 Insert thư viện sử dụng	29
Hình 3. 2 Tải dataset.....	29
Hình 3. 3 Mô tả dữ liệu.....	29
Hình 3. 4 10 dòng đầu của dữ liệu.....	30
Hình 3. 5 Kiểm tra dữ liệu trống	30
Hình 3. 6 Xóa dữ liệu trống.....	30
Hình 3. 7 Kiểm tra dữ liệu lặp	31
Hình 3. 8 Đặt lại index.....	31
Hình 3. 9 Đặt hình dạng của dữ liệu.....	31
Hình 3. 10 Xem lại dữ liệu	31
Hình 3. 11 Chia dữ liệu thử nghiệm	31
Hình 3. 12 Encode tập y	32
Hình 3. 13 Một số thông tin cơ bản	32
Hình 3. 14 Kiểm tra phân phối dữ liệu	34
Hình 3. 15 Độ tương quan giữa các biến	35
Hình 3. 16 Heatmap.....	36
Hình 3. 17 Chia dữ liệu	36
Hình 3. 18 Xây dựng mô hình	37
Hình 3. 19 Dựng hàm đánh giá.....	37
Hình 3. 20 Chạy mô hình đã xây dựng.....	38
Hình 3. 21 Thử nghiệm sự thay đổi của k	38
Hình 3. 22 Thử nghiệm với sự thay đổi của p	39
Hình 3. 23 So sánh thuật toán.....	40
Hình 3. 24 Confusion matrix	41
Hình 3. 25 Classification report.....	42
Hình 3. 26 Tối ưu hóa tham số Euclidean	42
Hình 3. 27 Tối ưu hóa tham số với độ đo Manhattan.....	43
Hình 3. 28 Tối ưu hóa tham số với độ đo Minkowski.....	43



DANH MỤC BẢNG

Bảng 1. 1 Thư viện sử dụng.....	10
Bảng 1. 2 Phân công nhiệm vụ	12
Bảng 2. 1 Dữ liệu demo.....	25
Bảng 2. 2 Euclidean demo	26
Bảng 2. 3 Phân tập khách hàng	26
Bảng 2. 4 Manhattan demo.....	27
Bảng 2. 5 Phân tập khách hàng manhattan	27
Bảng 5. 1 Danh sách class	44
Bảng 5. 2 Các phương thức của class TRAIN_MODELS	45
Bảng 5. 3 Các phương thức của class DESCRIPTIVE_STATISTICS.....	45
Bảng 5. 4 Các phương thức của class KNN_MODELS.....	45
Bảng 5. 5 Các phương thức khác.....	45
Bảng 5. 6 Giao diện overview	46
Bảng 5. 7 Giao diện trang descriptive statistic	46
Bảng 5. 8 Giao diện trang model	47
Bảng 5. 9 Giao diện trang Contact	47



I. TỔNG QUAN

1. Giới thiệu nhóm / lý do chọn dự án

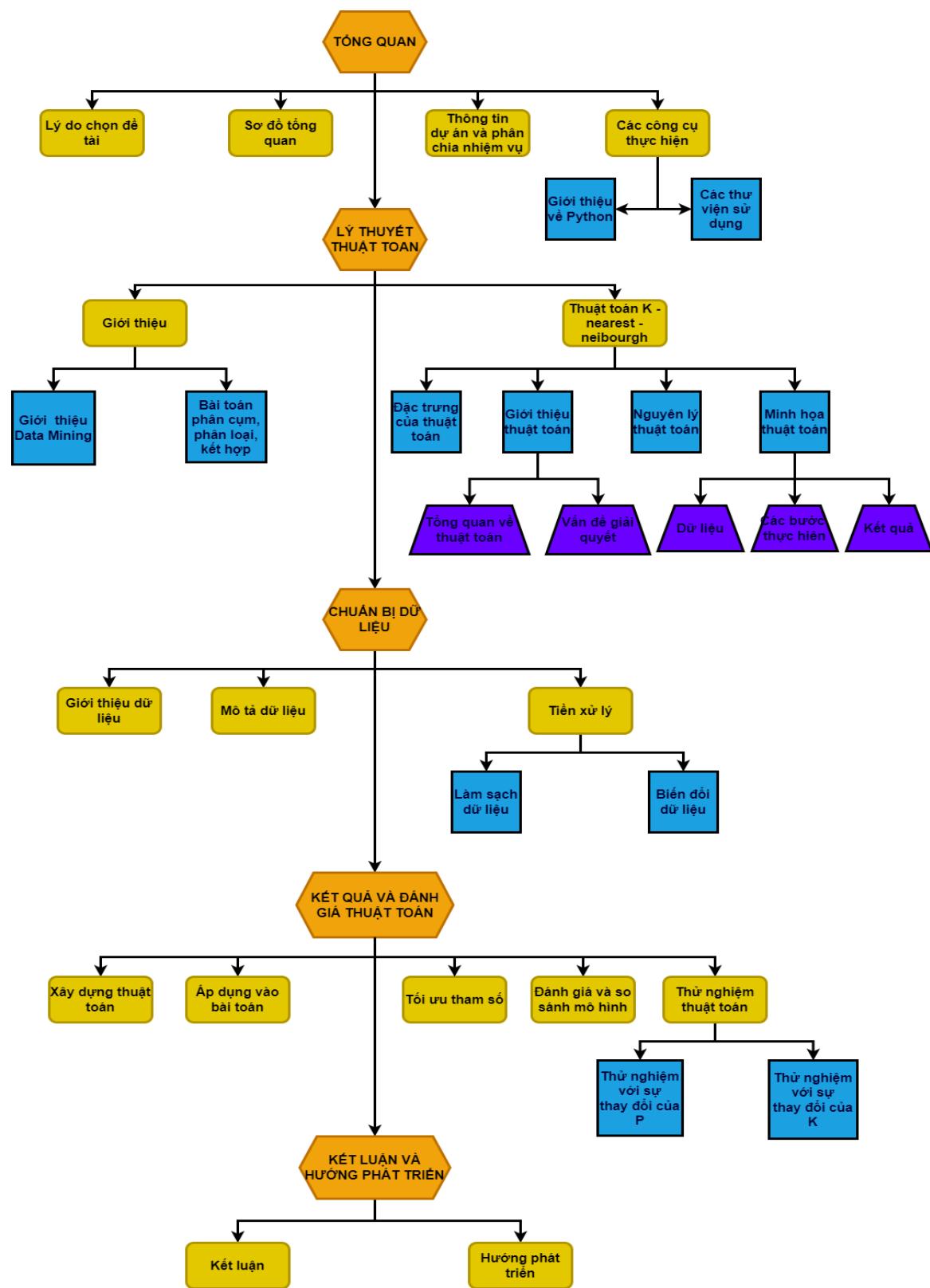
Học máy (ML - Machine Learning) đã trở thành phần công nghệ cốt lõi trong nhiều trường hợp sử dụng từ xử lý ngôn ngữ tự nhiên và thị giác máy tính đến phát hiện gian lận, dự báo nhu cầu, đề xuất sản phẩm, bảo trì phòng ngừa và xử lý tài liệu. Với những kiến thức đã học được trong môn “ Lưu trữ và khai thác hệ thống” của thầy “ NT Công” chúng em quyết định chọn đề tài là một trong những thuật toán phổ biến trong lĩnh vực học máy là K-Nearest Neighbors (KNN), một thuật toán phân loại dựa trên việc đo khoảng cách giữa các điểm dữ liệu. KNN là thuật toán học máy đóng vai trò quan trọng trong việc phân loại văn bản, nhận dạng hình ảnh và nhiều tác vụ khác trong lĩnh vực xử lý ngôn ngữ tự nhiên và thị giác máy tính. Nó là ngôn ngữ học máy trong quy mô lớn. Để tận dụng cần phải tiêu chuẩn hóa quy trình phát triển ML hiện đại trên toàn doanh nghiệp.

Điều quan trọng hơn hết đến việc lựa chọn đề tài KNN cho phép các nhà phát triển và nhà khoa học dữ liệu ở mọi cấp độ kỹ năng ML truy cập và quản lý tài nguyên một cách hiệu quả, đồng thời duy trì chi phí thấp. Nhờ vào khả năng phân loại chính xác và tính linh hoạt của nó, KNN là một công cụ mạnh mẽ trong việc xử lý ngôn ngữ tự nhiên và thị giác máy tính, và việc nghiên cứu về nó có thể mang lại nhiều giá trị cho các dự án ML và doanh nghiệp mà chúng em đang hướng tới.

2. Sơ đồ tổng quan trình bày (phân rã chức năng)

Dự án của nhóm đi qua những phần chính sau :

- Tổng quan
- Lý thuyết thuật toán
- Chuẩn bị dữ liệu
- Kết quả và đánh giá thuật toán
- Kết luận và hướng phát triển

Hình 1. 1 Sơ đồ phân rã chức năng (link xem: [full_image_bfd](#))



3. Công cụ thực hiện

3.1 Giới thiệu Python

a) Giới thiệu chung về Python

Python là một ngôn ngữ lập trình mạnh mẽ, đa năng và dễ học. Được phát triển vào cuối những năm 1980 bởi Guido van Rossum, Python nhanh chóng trở thành một trong những ngôn ngữ phổ biến nhất trên thế giới và được sử dụng rộng rãi trong nhiều lĩnh vực khác nhau.

Một trong những ưu điểm nổi bật của Python là cú pháp rõ ràng và dễ đọc, giúp cho việc viết mã trở nên dễ dàng và dễ hiểu. Python cung cấp một loạt các cấu trúc dữ liệu và phương pháp lập trình mạnh mẽ như danh sách (list), từ điển (dictionary), tuple và bộ (set), cho phép lập trình viên làm việc với dữ liệu một cách linh hoạt. Ngoài ra, Python cũng có một hệ sinh thái phong phú các thư viện và frameworks, cho phép lập trình viên xây dựng các ứng dụng phức tạp một cách nhanh chóng và hiệu quả. Ví dụ, thư viện NumPy cung cấp các công cụ mạnh mẽ cho tính toán khoa học và toán học, trong khi thư viện Pandas hỗ trợ xử lý và phân tích dữ liệu. TensorFlow và PyTorch là hai thư viện phổ biến trong lĩnh vực học sâu (deep learning), trong khi Django và Flask là các frameworks phổ biến để phát triển ứng dụng web...

Python có một ưu điểm lớn là tính di động, nghĩa là nó có thể chạy trên nhiều nền tảng khác nhau, bao gồm Windows, macOS và Linux. Điều này giúp cho việc phát triển và triển khai ứng dụng dễ dàng và thuận tiện. Được đánh giá cao với cộng đồng lập trình viên rộng lớn và nhiệt tình. Cộng đồng này cung cấp hàng ngàn tài liệu, ví dụ và hỗ trợ trực tuyến thông qua các diễn đàn, nhóm người dùng và trang web chia sẻ kiến thức.

b) Python trong lĩnh vực học máy

Trong lĩnh vực học máy (ML), Python đã trở thành một ngôn ngữ phổ biến và được ưa chuộng nhờ vào tính linh hoạt, hiệu suất và hệ sinh thái phong phú mà nó cung cấp. Với cú pháp đơn giản và dễ đọc, Python cho phép người dùng tập trung vào việc xây dựng mô hình và thử nghiệm các thuật toán học máy một cách dễ dàng.

Python cung cấp nhiều thư viện và frameworks mạnh mẽ để phát triển các ứng dụng học máy. Một trong số đó là scikit-learn, một thư viện phổ biến được sử dụng để triển khai các thuật toán học máy cơ bản và tiên tiến như K-Nearest Neighbors (KNN), Decision Trees, Random Forests và Support Vector Machines. Scikit-learn cung cấp các công cụ cho việc tiền xử lý dữ liệu, đánh giá mô hình và tinh chỉnh tham số, giúp người dùng dễ dàng xây dựng và đánh giá các mô hình học máy.

Trong số các thuật toán học máy, K-Nearest Neighbors (KNN) là một thuật toán phân loại đơn giản và hiệu quả. Nó dựa trên việc tính toán khoảng cách giữa các điểm dữ liệu để



xác định lớp phân loại của điểm dữ liệu mới. Python cung cấp các công cụ mạnh mẽ để thực hiện KNN, và scikit-learn là một trong những thư viện phổ biến cho việc triển khai chúng. Với sự trợ giúp của scikit-learn, việc xây dựng và đánh giá mô hình KNN trở nên đơn giản và thuận tiện. Python không chỉ hỗ trợ trong việc triển khai thuật toán KNN, mà còn là công cụ lý tưởng cho việc khám phá dữ liệu và tiền xử lý. Với thư viện như NumPy và Pandas, người dùng có thể thực hiện các phép toán số học, xử lý dữ liệu và trực quan hóa một cách dễ dàng. Điều này giúp người dùng có thể chuẩn bị dữ liệu cho mô hình KNN và tối ưu hóa quy trình làm việc của họ.

3.2 Các thư viện sử dụng

STT	Tên thư viện	Chức năng chính
1	Pandas	Pandas được thiết kế chủ yếu để xử lý và phân tích dữ liệu. Chức năng chính của Pandas là cung cấp cấu trúc dữ liệu linh hoạt như DataFrame và Series để dễ dàng thao tác và phân tích dữ liệu tabular: DataFrame và Series, Trực quan hóa dữ liệu, Thao tác dữ liệu với thời gian, Đánh chỉ số và multilindexing, Xử lý dữ liệu thiếu, Kết hợp và gộp dữ liệu,...
2	Numpy	NumPy tập trung vào cung cấp đối tượng mảng nhiều chiều và hỗ trợ các phép toán toán học hiệu quả trên các mảng này. Chức năng chính của NumPy là cung cấp một nền tảng mạnh mẽ cho tính toán khoa học và số học: Mảng, Thao tác mảng, Phép toán toán học, Truy cập và thay đổi phần tử mảng, Đại số tuyến tính, Tạo mảng, Thao tác dữ liệu thời gian, Hỗ trợ C-API,..
3	Math	Thư viện math trong Python cung cấp các hàm toán học cơ bản. Chức năng chính là hỗ trợ các phép toán và tính toán toán học trong các chương trình Python: Hàm số cơ học, Hàm số làm tròn và số nguyên, Hàm Trigonometry, Hàm số ngẫu nhiên, Hàm số tự nhiên,...
4	Datetime	Thư viện này cung cấp các lớp và hàm để làm việc với ngày tháng và thời gian trong Python. Lấy thông tin thời gian, Định dạng chuỗi thời gian, Các phép toán thời gian, lớp ‘datetime’ và ‘date’,...
5	Pyplot	Vẽ đồ thị cơ bản, Định dạng biểu đồ, Định dạng dữ liệu, Biểu đồ phân phối, Lưu và hiển thị biểu đồ,...



6	Seaborn	cung cấp một giao diện cao cấp để vẽ các biểu đồ thống kê hấp dẫn và dễ đọc: Biểu đồ phân phối, Biểu đồ tương quan, Biểu đồ phân loại, Thiết lập giao diện,...
7	Optuna	được thiết kế để tìm kiếm không gian siêu tham số để tối ưu hóa mục tiêu cụ thể, thường là một hàm mất mát trong quá trình huấn luyện mô hình máy học: Tìm kiếm tối ưu siêu tham số, Quản lý thông tin và kết quả thử nghiệm, Kiểm tra kết quả tìm kiếm,....
8	sklearn.metrics	Cung cấp các hàm và phương thức để đánh giá hiệu suất của mô hình học máy <ul style="list-style-type: none">• confusion_matrix• classification_report
9	sklearn.model_selection	Cung cấp phương thức train_test_split giúp chia dữ liệu thành tập train và tập thử.
10	Sklearn.neighbours	Cung cấp phương thức KneighborsClassifier giúp So sánh với mô hình tự xây dựng.
11	Plotly	Plotly là thư viện Python mạnh mẽ cho việc tạo đồ thị và biểu đồ tương tác. Nó hỗ trợ nhiều loại biểu đồ, tích hợp dễ dàng với các framework web, và có khả năng tương tác trực tiếp trên nền web. Plotly cũng được sử dụng rộng rãi cho việc thực hiện trực tiếp trong Jupyter Notebooks và hỗ trợ nhiều ngôn ngữ lập trình khác nhau.
12	Streamlit	Streamlit là một framework mã nguồn mở được sử dụng để xây dựng ứng dụng web với Python một cách nhanh chóng và dễ dàng. Đây là một công cụ phổ biến cho việc tạo ra các ứng dụng dựa trên dữ liệu hoặc máy học mà không cần kiến thức sâu về front-end. Bằng cách sử dụng các thư viện như Pandas, Matplotlib và Plotly, nó có thể hiển thị và tương tác với dữ liệu một cách linh hoạt. Điểm mạnh của Streamlit là khả năng tập trung vào logic của ứng dụng.

Bảng 1.1 Thư viện sử dụng



3.3 Tổng quan về dữ liệu sử dụng

Dữ liệu chính ở trong bài để xây dựng thuật toán được lấy từ trang web y tế : [UCI-EDU-DIAGNOSTIC](#). Dữ liệu này mô tả về các đặc điểm được tính toán từ hình ảnh số hóa của chọc hút bằng kim nhỏ (FNA) của khối vú. Chúng mô tả đặc điểm của nhân tế bào có trong hình ảnh.

Bằng dữ liệu này người ta có thể sử dụng để tiến hành chẩn đoán ung thư.

4. Thông tin dự án của nhóm và phân chia nhiệm vụ

Nhóm thực hiện đề tài trong vòng 2 tuần, với 3 thành viên chính, mỗi thành viên đảm nhiệm những nhiệm vụ khác nhau như bảng 1.2. Nhóm đã thực hiện tải dữ liệu và tiến hành phân tích dự báo, tất cả công cụ, thư viện và những yếu tố phần mềm cần thiết để demo dữ liệu nhóm đã tiến hành đóng gói lại vào dự án cộng đồng trên nền tảng GitHub.

a) Giới thiệu về nền tảng Github và dự án của nhóm

GitHub là một nền tảng lưu trữ mã nguồn và quản lý dự án phổ biến dành cho các nhà phát triển phần mềm. Nó cho phép người dùng lưu trữ mã nguồn của dự án, theo dõi sự thay đổi của dữ liệu.

Các tính năng chính của GitHub bao gồm:

- Lưu trữ mã nguồn: Người dùng có thể tạo và quản lý các kho lưu trữ (repositories) để lưu trữ mã nguồn của dự án.
- Quản lý phiên bản: GitHub ghi lại lịch sử thay đổi của mã nguồn, cho phép người dùng theo dõi các sửa đổi, trở về phiên bản cũ, và hợp nhất các thay đổi.
- Hợp tác và Phản hồi: Nền tảng này cho phép các nhà phát triển cùng làm việc trên dự án, đưa ra đề xuất (pull request), xem xét và thảo luận về mã nguồn.
- Công cụ Quản lý Dự án: GitHub cung cấp các công cụ quản lý dự án như quản lý nhiệm vụ (issues), wiki, và dự án kanban board để theo dõi tiến độ công việc.

GitHub đã trở thành một trong những công cụ quan trọng cho cộng đồng phát triển phần mềm, cung cấp nền tảng cho sự hợp tác, kiểm soát phiên bản, và lưu trữ mã nguồn dự án một cách dễ dàng và hiệu quả. Hợp tác với đồng nghiệp và kiểm soát phiên bản.

Dự án của nhóm trên nền tảng GitHub:

Link truy cập dự án: [GITHUB-DIANOSING CANCER WITH KNN UCI-EDU-DIAGNOSTIC](#)

- Thông tin cách cài đặt và sử dụng: README.md
- Thông tin các thư viện cần thiết: requirements.txt
- Cấu trúc thư mục dự án github:



☞ Cấu Trúc Thư Mục (Directory Structure)

- `data/` : Chứa dữ liệu của dự án.
- `model/` : Chứa notebook phân tích chính.
- `asset/image/` : Lưu trữ ảnh của dự án (Stores project images)
- `info/` : Lưu trữ thông tin giới thiệu (Stores information).
- `introduction` : Bản thảo word của dự án (Word draft of the project).
- `visualization/app.py` : Tệp chứa giao diện demo thuật toán (The file contains the algorithm demo interface)
- `Report/` : Lưu trữ thông tin báo cáo chi tiết.

Hình 1. 2 Cấu trúc thư mục dự án Github

b) Phân chia nhiệm vụ

STT	Họ và tên	MSSV	Nhiệm vụ
1	Hoàng Xuân Quốc	2156210125	Phân tích mô tả, code lại mô hình thuật toán
2	Nguyễn Viết Đức	2156210100	Chuẩn bị dữ liệu, code lại mô hình thuật toán
3	Đặng Hoàng Chiến	2156210095	Làm báo cáo, code mô hình thuật toán

Bảng 1. 2 Phân công nhiệm vụ

II. LÝ THUYẾT THUẬT TOÁN

1. Giới thiệu về data mining và bài toán phân loại/phân cụm/tìm luật kết hợp

1.1 Giới thiệu về data mining

a) Data mining là gì?

Data Mining là quá trình khai phá dữ liệu và phân tích để tìm ra các liên hệ, thông tin hữu ích và tiềm ẩn trong dữ liệu. Mục tiêu của việc này là giúp cho dự đoán xu hướng tương lai chính xác hơn và đưa ra quyết định được hỗ trợ từ tập dữ liệu khổng lồ.

b) Các bước để thực hiện Data Mining

Thu thập và tiền xử lý dữ liệu: Thu thập dữ liệu và tiền xử lý để đảm bảo tính chính xác và đầy đủ của dữ liệu sử dụng cho phân tích.

Lựa chọn phương pháp Data Mining: Xác định phương pháp phù hợp cho mục đích phân tích, bao gồm Clustering, Classification, Regression, Association Rules, và nhiều phương pháp khác.

Thực hiện phân tích dữ liệu: Áp dụng phương pháp Data Mining đã chọn để phân tích dữ liệu, tìm ra các mẫu và thông tin quan trọng.



Đánh giá kết quả: Trình bày và đánh giá kết quả phân tích để đảm bảo tính chính xác và đáng tin cậy của thông tin được tìm thấy.

c) Các phương pháp data mining phổ biến

Clustering (Phân cụm): Phương pháp này nhóm các đối tượng tương tự lại với nhau và phân tách chúng khỏi các đối tượng khác. Kết quả thu được là các nhóm đối tượng giống nhau hoặc có quan hệ với nhau.

Classification (Phân loại): Là phương pháp phân loại các đối tượng dựa trên các thuộc tính và giá trị đã được xác định trước. Kết quả thu được là các đối tượng được phân loại vào các nhóm khác nhau dựa trên các thuộc tính cụ thể.

Regression (Hồi quy): Dự đoán giá trị của một biến dựa trên các biến khác. Kết quả thu được là một mô hình dự đoán cho giá trị biến phụ thuộc.

Association Rules (Quy tắc kết hợp): Phân tích các quan hệ giữa các đối tượng trong tập dữ liệu. Kết quả thu được là các tập hợp các đối tượng có các quan hệ tương tự với nhau.

Neural Networks (Mạng nơ-ron): Phương pháp Machine Learning được lấy cảm hứng từ cấu trúc của hệ thống thần kinh sinh học, mô phỏng hoạt động của não người. Neural Networks được sử dụng để giải quyết các bài toán phức tạp trong nhiều lĩnh vực như xử lý ngôn ngữ tự nhiên, nhận dạng hình ảnh, dự báo tài chính và nhiều lĩnh vực khác.

Sequence Analysis (Phân tích chuỗi): Phương pháp này cho phép phát hiện các chuỗi sự kiện thường xuyên xảy ra cùng nhau hoặc các chuỗi sự kiện xảy ra trong một thời gian cụ thể. Kết quả thu được là các mô hình chuỗi sự kiện, giúp hiểu rõ hơn về các quá trình xảy ra trong tập dữ liệu.

Path Analysis (Phân tích quỹ đạo): Tập trung vào việc phát hiện các mối quan hệ giữa các sự kiện và xây dựng các mô hình quan hệ để giải thích các quá trình xảy ra trong tập dữ liệu. Kết quả thu được là các mô hình quan hệ giữa các sự kiện, giúp hiểu rõ hơn về sự tương tác và phụ thuộc giữa các biến.

d) Các công cụ hỗ trợ

Weka: Phần mềm mã nguồn mở hỗ trợ Data Mining, có thể chạy trên nhiều hệ điều hành khác nhau.

RapidMiner: Phần mềm có giao diện trực quan và dễ sử dụng, hỗ trợ nhiều phương pháp Data Mining.

KNIME: Phần mềm đa năng và mở rộng, có thể tích hợp nhiều công cụ và phương pháp Data Mining khác nhau.

1.2 Bài toán phân loại/phân cụm/tìm tập kết hợp

a) Bài toán phân loại

Bài toán phân loại là bài toán liên quan đến việc xây dựng một mô hình dự đoán để phân loại các đối tượng vào các nhóm khác nhau dựa trên các thuộc tính đã cho. Để thực hiện bài toán phân loại, ta sử dụng một tập dữ liệu huấn luyện đã được gán nhãn (có các đối tượng và nhãn tương ứng cho từng đối tượng). Mô hình phân loại được huấn luyện trên tập dữ liệu này và sau đó được sử dụng để dự đoán nhãn cho các đối tượng mới. Ví dụ, trong



bài toán phân loại email, mô hình sẽ học từ các email đã được gán nhãn là "thư rác" hoặc "thư thường" dựa trên các thuộc tính như từ khóa, độ dài, địa chỉ nguồn, v.v. Khi mô hình đã được huấn luyện, ta có thể sử dụng nó để dự đoán xem một email mới có phải là thư rác hay không. Kết quả của bài toán phân loại là một quyết định hoặc xác suất thuộc về từng nhóm.

Quá trình phân loại bao gồm các bước sau:

Thu thập dữ liệu: Bước đầu tiên là thu thập dữ liệu huấn luyện, trong đó mỗi mẫu được gán nhãn hoặc phân loại dựa trên thông tin đã biết. Dữ liệu này thường bao gồm các đặc trưng (features) mô tả mỗi mẫu và nhãn (label) tương ứng với phân loại mong muốn.

Tiền xử lý dữ liệu: Trước khi xây dựng mô hình, dữ liệu thường cần được tiền xử lý để chuẩn hóa, loại bỏ nhiễu và xử lý các giá trị thiếu. Điều này đảm bảo rằng dữ liệu sẽ đáng tin cậy và phù hợp cho quá trình phân loại.

Chọn mô hình phân loại: Chọn mô hình phân loại phù hợp là một bước quan trọng. Có nhiều loại mô hình phân loại khác nhau như cây quyết định (decision tree), hỗn hợp Gaussian (Gaussian mixture), máy vector hỗ trợ (support vector machine), mạng nơ-ron (neural network) và rất nhiều mô hình khác. Việc chọn mô hình thích hợp phụ thuộc vào loại dữ liệu, số lượng đặc trưng và yêu cầu phân loại cụ thể.

Xây dựng mô hình: Trong bước này, mô hình phân loại được xây dựng bằng cách huấn luyện với dữ liệu huấn luyện. Quá trình huấn luyện là quá trình tìm ra các tham số của mô hình để tối ưu hóa hiệu suất phân loại. Các thuật toán huấn luyện khác nhau sẽ có các cách tiếp cận khác nhau để tìm ra các tham số này.

Đánh giá mô hình: Sau khi mô hình được xây dựng, nó cần được đánh giá để đảm bảo rằng nó có thể hoạt động tốt trên dữ liệu mới. Đánh giá mô hình thường được thực hiện bằng cách sử dụng dữ liệu kiểm tra (test data) mà mô hình chưa từng được tiếp xúc. Các độ đo phổ biến để đánh giá mô hình phân loại bao gồm độ chính xác (accuracy), độ phủ (recall), độ chính xác dương tính (precision) và F1-score.

Sử dụng mô hình: Sau khi mô hình đã được đánh giá và chứng minh hiệu suất tốt, nó có thể được sử dụng để dự đoán và phân loại các mẫu mới. Mô hình sẽ xử lý các đặc trưng của các mẫu mới và dự đoán nhãn hoặc phân loại tương ứng.

b) Bài toán phân cụm

Phân cụm dữ liệu là bài toán gom nhóm các đối tượng dữ liệu vào thành từng cụm (cluster) sao cho các đối tượng trong cùng một cụm có sự tương đồng theo một tiêu chí nào đó.

Mục tiêu là tìm ra cấu trúc tự nhiên trong dữ liệu mà không cần có thông tin trước về nhãn hay nhóm. Điều này có nghĩa là chúng ta không biết trước các nhóm cụ thể trong dữ



liệu. Các thuật toán phân cụm sẽ phân chia dữ liệu thành các nhóm dựa trên sự tương đồng giữa các đối tượng dữ liệu. Ví dụ, trong phân cụm khách hàng, chúng ta có thể sử dụng các thuật toán phân cụm để nhóm các khách hàng có cùng hành vi mua hàng lại với nhau. Kết quả của bài toán phân cụm là các nhóm đối tượng tương tự hoặc có quan hệ với nhau.

Dưới đây là các bước cơ bản trong bài toán phân cụm:

Thu thập dữ liệu: Bước đầu tiên là thu thập dữ liệu mà chúng ta muốn phân cụm. Dữ liệu này thường bao gồm các đặc trưng mô tả mỗi mẫu, và không yêu cầu có nhãn như trong bài toán phân loại.

Tiền xử lý dữ liệu: Trước khi thực hiện phân cụm, dữ liệu thường cần được tiền xử lý để loại bỏ nhiễu, chuẩn hóa và xử lý các giá trị thiếu. Điều này giúp đảm bảo rằng dữ liệu sẽ đáng tin cậy và phù hợp cho quá trình phân cụm.

Chọn phương pháp phân cụm: Có nhiều phương pháp phân cụm khác nhau, và lựa chọn phương pháp thích hợp phụ thuộc vào loại dữ liệu và yêu cầu cụ thể của bài toán. Một số phương pháp phân cụm phổ biến bao gồm phân cụm theo k-means, phân cụm theo độ tương đồng (hierarchical clustering), phân cụm dựa trên mô hình Gaussian hỗn hợp (Gaussian mixture model), và phân cụm dựa trên mạng nơ-ron tự tổ chức (self-organizing map).

Xây dựng mô hình phân cụm: Trong bước này, mô hình phân cụm được xây dựng bằng cách áp dụng phương pháp phân cụm đã chọn lên dữ liệu. Mục tiêu là tìm ra cách chia dữ liệu thành các cụm sao cho các mẫu trong cùng một cụm có đặc điểm tương tự.

Đánh giá và tinh chỉnh: Sau khi mô hình được xây dựng, chúng ta cần đánh giá chất lượng của phân cụm. Các phương pháp đánh giá phân cụm thường đo lường sự tương đồng bên trong các cụm và sự khác biệt giữa các cụm. Các phương pháp như độ tương tự silhouette (silhouette similarity) và chỉ số Calinski-Harabasz (Calinski-Harabasz index) thường được sử dụng.

Sử dụng phân cụm: Sau khi mô hình phân cụm đã được đánh giá và tinh chỉnh, chúng ta có thể sử dụng nó để phân cụm các mẫu mới. Mô hình sẽ xử lý các đặc trưng của mẫu mới và gán chúng vào các cụm tương ứng.

c) Tìm luật kết hợp

Bài toán tìm luật kết hợp là một bài toán khám phá các quy tắc kết hợp (association rules) giữa các mục (items) trong một tập dữ liệu đã cho. Mục tiêu chính của bài toán tìm luật kết hợp là tìm ra các quy tắc kết hợp có độ tin cậy và hỗ trợ xác định. Độ tin cậy (confidence) đo lường mức độ chính xác của quy tắc và độ hỗ trợ (support) đo lường mức độ phổ biến của các mục trong quy tắc.



Để có thể hiểu rõ hơn, hãy xem một ví dụ cụ thể. Giả sử chúng ta có một tập dữ liệu về hóa đơn mua hàng của khách hàng tại một cửa hàng. Tập dữ liệu này ghi lại thông tin về các mặt hàng mà khách hàng đã mua. Bài toán tìm luật kết hợp sẽ giúp chúng ta khám phá các mẫu mua hàng kết hợp, ví dụ "nếu khách hàng mua sản phẩm A thì họ cũng thường mua sản phẩm B".

Quy trình thực hiện bài toán tìm luật kết hợp bao gồm các bước sau:

Xác định các mục (items) trong tập dữ liệu: Đầu tiên, chúng ta cần xác định các mục hoặc sản phẩm trong tập dữ liệu. Trong ví dụ của chúng ta, các mục có thể là các sản phẩm như A, B, C...

Tính toán độ hỗ trợ (support): Độ hỗ trợ là một yếu tố quan trọng để đánh giá mức độ phổ biến của các mục trong tập dữ liệu. Nó được tính bằng tỷ lệ giữa số lần xuất hiện của một tập hợp các mục cụ thể và tổng số hồ sơ trong tập dữ liệu. Ví dụ, độ hỗ trợ của quy tắc "A và B" là số lần mua cả sản phẩm A và B chia cho tổng số hóa đơn.

Xác định độ tin cậy (confidence): Độ tin cậy đo lường mức độ chính xác của một quy tắc. Nó được tính bằng tỷ lệ giữa số lần xuất hiện của quy tắc và số lần xuất hiện của mục xuất hiện đầu tiên trong quy tắc. Ví dụ, độ tin cậy của quy tắc "A dẫn đến B" là số lần mua cả sản phẩm A và B chia cho số lần mua sản phẩm A.

Áp dụng ràng buộc: Để giới hạn số lượng quy tắc kết hợp được tạo ra, chúng ta có thể áp dụng ràng buộc về độ hỗ trợ và độ tin cậy. Chỉ các quy tắc vượt qua các ràng buộc này mới được chấp nhận.

Tạo ra các quy tắc kết hợp: Cuối cùng, chúng ta sử dụng các bước trên để tạo ra các quy tắc kết hợp. Các quy tắc này sẽ cho chúng ta thông tin về mối liên hệ giữa các mục trong tập dữ liệu.

Sau quá trình tìm luật kết hợp, chúng ta có thể nhận được các quy tắc như:

- Nếu khách hàng mua sản phẩm A và B, thì khách hàng cũng thường mua sản phẩm C.
- Nếu khách hàng mua sản phẩm A và C, thì khách hàng cũng thường mua sản phẩm D.

2. Thuật toán K – nearest-neighbour

2.1 Giới thiệu về thuật toán và bài toán mà thuật toán giải quyết

2.1.1 Tổng quan về thuật toán

a) Khái niệm

Thuật toán K-Nearest Neighbor là thuật toán có nghĩa tiếng Việt là K láng giềng gần nhất, viết tắt là KNN. Thuật toán KNN là một kĩ thuật học có giám sát (supervised learning) dùng để phân loại quan sát mới bằng cách tìm điểm tương đồng giữa quan sát mới này với



dữ liệu sẵn có. Thuật toán này được sử dụng rộng rãi trong nhiều lĩnh vực khác nhau từ phân loại, hồi quy cho đến hệ thống gợi ý.

b) Nguyên lý hoạt động cơ bản

KNN hoạt động dựa trên nguyên tắc rằng các điểm dữ liệu tương tự thường nằm gần nhau. Để dự đoán nhãn cho một điểm dữ liệu mới, thuật toán này xem xét ‘K’ điểm dữ liệu gần nhất và dựa trên đa số nhãn của chúng để xác định nhãn cho điểm dữ liệu mới đó.

Giá trị ‘K’ là một siêu tham số quan trọng, quy định số lượng hàng xóm gần nhất mà thuật toán sẽ xem xét. Việc chọn K đúng là rất quan trọng: K quá nhỏ có thể làm mô hình bị ảnh hưởng bởi nhiều dữ liệu, trong khi K quá lớn có thể làm mô hình không nhạy với đặc điểm cụ thể của dữ liệu.

Tính toán khoảng cách giữa các điểm dữ liệu là một phần quan trọng của KNN dùng để đo lường sự tương tự giữa các điểm dữ liệu. Phương pháp phổ biến nhất là khoảng cách Euclidean. Tuy nhiên, có nhiều phương pháp khác như khoảng cách Manhattan được sử dụng tùy thuộc vào bản chất của dữ liệu.

c) Lịch sử hình thành và phát triển

Khởi nguyên: Nguồn gốc và lịch sử phát triển của thuật toán K-Nearest Neighbors (KNN) bắt đầu trong giai đoạn đầu của lĩnh vực học máy và trí tuệ nhân tạo. KNN xuất hiện lần đầu vào những năm 1950 và 1960 trong quá trình nghiên cứu về mô hình hóa mẫu và nhận dạng. Mặc dù không rõ người chính thức sáng tạo KNN, thuật toán này nhanh chóng trở nên phổ biến do tính đơn giản và dễ hiểu của nó.

Phát triển và cải tiến: Trong thập kỷ tiếp theo, KNN đã được cải tiến và tinh chỉnh để nâng cao hiệu suất và độ chính xác. Các nghiên cứu tập trung vào việc tối ưu hóa lựa chọn giá trị 'K', phương pháp tính khoảng cách và xử lý dữ liệu lớn. Một trong những cải tiến quan trọng là áp dụng các kỹ thuật giảm chiều dữ liệu như Principal Component Analysis (PCA) và t-SNE để giảm tải tính toán và cải thiện khả năng phân loại.

2.1.2 Vấn đề mà bài toán giải quyết

a) Về mặt lý thuyết

Về mặt lý thuyết, bài toán mà thuật toán K-Nearest Neighbors giải quyết là bài toán phân loại và gọi ý dựa trên các đặc trưng của dữ liệu. KNN được sử dụng để xác định lớp hoặc nhãn của một điểm dữ liệu mới dựa trên các điểm dữ liệu đã biết trước trong tập huấn luyện. KNN thuộc vào lớp các thuật toán học máy không giám sát (unsupervised learning) và phụ thuộc vào việc tính toán khoảng cách giữa các điểm dữ liệu. Thuật toán này dựa trên giả định rằng các điểm dữ liệu cùng lớp sẽ có tính chất tương tự và nằm gần nhau trong không gian đặc trưng.

b) Về những ứng dụng thực tế



- Phân loại văn bản: sử dụng để phân loại văn bản, chẳng hạn như phân loại email vào hộp thư rác hoặc không phải hộp thư rác dựa trên nội dung và các đặc trưng của email.
- Nhận dạng khuôn mặt: KNN có thể được áp dụng trong hệ thống nhận dạng khuôn mặt để xác định xem một khuôn mặt mới thuộc về người nào dựa trên các khuôn mặt đã biết trước trong tập huấn luyện.
- Hệ thống gợi ý: Có thể được sử dụng trong các hệ thống gợi ý sản phẩm hoặc nội dung. Điều này có thể bao gồm việc đề xuất các sản phẩm tương tự cho người dùng dựa trên lịch sử mua hàng hoặc đánh giá của họ.
- Phát hiện gian lận tín dụng: KNN có thể được sử dụng để phát hiện gian lận tín dụng bằng cách xác định xem một giao dịch mới có khả năng là gian lận dựa trên các đặc trưng của giao dịch và thông tin từ các giao dịch trước đó.
- Phân loại hình ảnh: Được áp dụng trong việc phân loại hình ảnh vào các nhóm khác nhau. Ví dụ, xác định xem một bức ảnh là chó, mèo hoặc chim cút dựa trên các đặc trưng hình ảnh và tập huấn luyện các hình ảnh đã được gán nhãn trước đó.
- Nhận dạng tiếng nói: áp dụng trong các hệ thống nhận dạng tiếng nói để xác định người nói dựa trên các đặc trưng âm thanh của giọng nói.
- Phân loại sản phẩm: KNN có thể được sử dụng trong các cửa hàng trực tuyến để phân loại sản phẩm vào các danh mục khác nhau dựa trên các đặc trưng của sản phẩm như mô tả, giá, nhãn hiệu, đánh giá và thông tin liên quan khác.
- Dự báo thời tiết: nó có thể được áp dụng trong việc dự báo thời tiết dựa trên các đặc trưng thời tiết trước đó như nhiệt độ, độ ẩm, áp suất không khí và các yếu tố khí hậu khác.
- Hỗ trợ quyết định y tế: KNN có thể được sử dụng để hỗ trợ quyết định trong lĩnh vực y tế, chẳng hạn như xác định liệu một bệnh nhân có nguy cơ cao mắc bệnh tim mạch hay không dựa trên các chỉ số sức khỏe và lịch sử bệnh lý...

2.2 Đặc trưng của thuật toán

a) Đặc trưng chung

Dựa trên sự tương tự: KNN sẽ đo lường tính toán độ tương đồng hoặc khoảng cách giữa các điểm dữ liệu trong không gian đặc trưng. Nó sẽ sử dụng một phép đo khoảng cách để xác định các điểm gần nhất với điểm dữ liệu đang được dự đoán. Các điểm gần nhất này sau đó được sử dụng để suy ra nhãn hoặc giá trị dự đoán cho điểm dữ liệu đó.

- **Ưu điểm:** Với đặc trưng này, việc xác định và tính toán giữa các điểm dữ liệu là khá đơn giản và dễ hiểu. Nó cho phép đo lường sự tương đồng dựa trên khoảng cách giữa các điểm, giúp xác định các điểm dữ liệu gần nhau.
- **Nhược điểm:** Nó có thể không hoạt động tốt đối với các tập dữ liệu có đặc trưng không đồng nhất hoặc khi các lớp không phân tách rõ ràng trong không gian đặc trưng của dữ liệu.



Không giám sát: KNN là một thuật toán học không giám sát, nghĩa là nó không yêu cầu một giai đoạn huấn luyện. Nó chỉ cần dữ liệu huấn luyện đã được gán nhãn để thực hiện dự đoán.

- **Ưu điểm:** Chúng ta chỉ cần dữ liệu huấn luyện đã được gán nhãn để thực hiện dự đoán, giảm thiểu được khá công việc chuẩn bị dữ liệu huấn luyện.
- **Nhược điểm:** Vì KNN không yêu cầu huấn luyện tường minh, nó có thể không học được các mẫu, quy luật hoặc đặc trưng phức tạp trong dữ liệu. Điều này có thể dẫn đến hiệu suất dự đoán không tốt trên các tập dữ liệu phức tạp.

Khoảng cách: KNN dựa trên các điểm dữ liệu gần nhất trong không gian đặc trưng để đưa ra dự đoán. Giá trị của K trong thuật toán KNN xác định số lượng các hàng xóm gần nhất cần xem xét.

- **Ưu điểm:** KNN dựa trên các điểm dữ liệu gần nhất để đưa ra dự đoán. Điều này có nghĩa là KNN tập trung vào các điểm dữ liệu có tương đồng về mặt không gian đặc trưng, giúp xác định nhóm và phân loại dữ liệu.
- **Nhược điểm:** Việc tìm kiếm các hàng xóm gần nhất trong quá trình dự đoán có độ phức tạp tính toán cao. Đặc biệt là khi số lượng điểm dữ liệu huấn luyện lớn, việc tìm kiếm các hàng xóm xung quanh có thể tốn nhiều thời gian và tài nguyên.

Linh hoạt với số chiều: KNN có khả năng làm việc tốt với các tập dữ liệu có số chiều cao. Điều này là do KNN không giả định về phân phối của dữ liệu và không áp dụng một mô hình toán học phức tạp. Tuy nhiên, khi số chiều tăng lên, việc tính toán khoảng cách có thể trở nên tốn kém và hiệu quả của thuật toán có thể giảm. Để giải quyết ta có thể sử dụng kỹ thuật giảm chiều dữ liệu như PCA hoặc LLE trước khi áp dụng KNN. Điều này giúp giảm số chiều của dữ liệu trong khi vẫn giữ lại thông tin quan trọng và cấu trúc của dữ liệu.

- **Ưu điểm:** KNN có khả năng làm việc tốt với các tập dữ liệu có số chiều cao. Điều này là do KNN không giả định về phân phối của dữ liệu và không áp dụng một mô hình toán học phức tạp. Nó có thể áp dụng trực tiếp trên không gian đặc trưng có số chiều cao mà không cần quá nhiều điều chỉnh.
- **Nhược điểm:** Mặc dù linh hoạt với số chiều, KNN có thể gặp khó khăn trong việc xử lý các tập dữ liệu có số chiều rất lớn. Việc tính toán khoảng cách giữa các điểm dữ liệu có thể trở nên rất tốn kém và không hiệu quả trong trường hợp này.

b) **Ưu điểm chung**

- **Đơn giản và dễ hiểu:** KNN là một thuật toán đơn giản và dễ hiểu. Nguyên tắc hoạt động của nó dựa trên ý tưởng rằng các điểm dữ liệu gần nhau trong không gian đặc trưng có khả năng thuộc cùng một nhãn.



- **Khả năng xử lý dữ liệu phi cấu trúc:** KNN có khả năng xử lý dữ liệu phi cấu trúc, không yêu cầu các giả định về cấu trúc dữ liệu. Điều này cho phép nó áp dụng cho nhiều loại dữ liệu, bao gồm cả dữ liệu văn bản, hình ảnh, và âm thanh.
 - **Khả năng mở rộng:** Thuật toán này không yêu cầu huấn luyện tƣờng minh, do đó nó có khả năng mở rộng cho việc thêm dữ liệu mới một cách dễ dàng.
 - **Hiệu suất tốt đối với tập dữ liệu nhỏ:** KNN hoạt động tốt trên các tập dữ liệu nhỏ với số lượng mẫu ít. Nó không đòi hỏi quá nhiều tính toán trước khi thực hiện dự đoán.
 - **Dễ dàng tinh chỉnh tham số:** K là tham số quan trọng trong thuật toán KNN. Việc điều chỉnh giá trị K có thể ảnh hưởng đến hiệu suất của thuật toán. Tuy nhiên, việc tinh chỉnh K là khá dễ dàng và có thể được thực hiện thông qua quá trình thử và sai.
 - **Sử dụng được trong cả phân loại và hồi quy:** KNN có thể được sử dụng cho cả hai mục đích là phân loại và hồi quy.
 - **Hoạt động tốt trong trường hợp phân loại với nhiều lớp:** KNN có thể được sử dụng để phân loại dữ liệu với nhiều lớp.
 - **Không cần giả định gì về phân phối của các class:** KNN là một thuật toán phi tham số, vì vậy nó không cần giả định gì về phân phối của các class. Điều này giúp KNN có thể hoạt động tốt với các tập dữ liệu có phân phối phức tạp hoặc không xác định.
- c) Nhược điểm chung
- **Phụ thuộc vào kích thước dữ liệu:** hiệu suất sẽ giảm khi tập dữ liệu lớn vì việc tính toán khoảng cách giữa các điểm dữ liệu mất nhiều thời gian. KNN không phải là thuật toán phù hợp cho việc xử lý các tập dữ liệu lớn.
 - **Tính toán tài nguyên:** KNN có đòi hỏi khá nhiều tài nguyên tính toán khi tập dữ liệu lớn, đặc biệt là khi số chiều của không gian đặc trưng cao.
 - **Nhạy cảm với nhiễu và dữ liệu không đồng nhất:** Dễ bị ảnh hưởng bởi các nhiễu trong dữ liệu và các điểm dữ liệu nằm trong các lớp khác nhau gần nhau. Điều này có thể dẫn đến việc phân loại không chính xác hoặc không ổn định.
 - **Cần xử lý các biến số và mất cân bằng dữ liệu:** KNN không xử lý được các biến số khác nhau và cần sự cân bằng dữ liệu trong các lớp khác nhau. Nếu một lớp có số lượng mẫu nhiều hơn so với lớp khác, KNN có thể dễ dàng bị thiên vị và cho ra kết quả không chính xác.
 - **Lưu trữ dữ liệu:** KNN cần lưu trữ toàn bộ tập dữ liệu huấn luyện trong bộ nhớ để thực hiện việc tìm kiếm hàng xóm gần nhất. Điều này có thể là một vấn đề khi tập dữ liệu rất lớn.
 - **Lựa chọn tham số K:** Tham số K trong KNN cần được chọn một cách thích hợp. Một K quá nhỏ có thể dẫn đến overfitting, trong khi một K quá lớn có thể dẫn đến underfitting.

2.3 Nguyên lý thuật toán

Nguyên lý hoạt động của thuật toán K-Nearest Neighbors (KNN) là dựa trên giả định rằng các điểm dữ liệu có tính chất tương đồng với nhau thường nằm gần nhau trong



không gian đặc trưng. Khi có một điểm dữ liệu mới, KNN sẽ tìm ra các điểm dữ liệu gần nó nhất trong không gian đặc trưng, và sử dụng nhãn của các điểm dữ liệu này để dự đoán nhãn của điểm dữ liệu mới.



Hình 1. 3 Sơ đồ nguyên lý



Giải thích chi tiết sơ đồ:

- Chuẩn bị dữ liệu huấn luyện: Thuật toán KNN yêu cầu một tập dữ liệu huấn luyện đã được gán nhãn, trong đó mỗi điểm dữ liệu có các đặc trưng và nhãn tương ứng
- Xác định giá trị 'K': Người dùng cần quyết định giá trị 'K', tức là số lượng hàng xóm gần nhất mà thuật toán sẽ xem xét trong quá trình dự đoán.
- Tính toán khoảng cách: KNN sử dụng một phương pháp tính khoảng cách để đo lường sự tương tự giữa các điểm dữ liệu. Khoảng cách được tính giữa điểm dữ liệu mới và các điểm dữ liệu trong tập huấn luyện.
- Các phép đo khoảng cách chủ yếu trong KNN là
- Khoảng cách Euclidean: Đây là phép đo khoảng cách phổ biến nhất trong KNN. Khoảng cách Euclidean giữa hai điểm dữ liệu trong không gian đặc trưng được tính bằng:

$$D(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

Trong đó

- $D(x,y)$ là khoảng cách giữa hai điểm x và y
- n là số chiều của không gian, x_i và y_i là các thành phần tương ứng của điểm x và y .

Độ đo Euclidean là đặc trưng của không gian Euclidean (không gian hình học thông thường). Khoảng cách Euclidean giữa hai điểm là đường đi ngắn nhất giữa chúng, tương đương với độ dài của đoạn thẳng nối chúng trong không gian Euclidean.

- Khoảng cách Manhattan: Còn được gọi là khoảng cách thành phố, khoảng cách Manhattan tính toán tổng các độ lệch tuyệt đối giữa các giá trị của các đặc trưng tương ứng của hai điểm dữ liệu với công thức.

$$D(x, y) = \sum_{i=1}^n |x_i - y_i|$$

Trong đó

- $D(x,y)$ là khoảng cách Manhattan giữa hai điểm x và y ,
- n là số chiều của không gian, x_i và y_i là các thành phần tương ứng của điểm x và y .



Độ đo Manhattan tương ứng với tổng của độ chênh lệch giữa các thành phần của hai điểm. Nó được gọi là "Manhattan" vì nó đề xuất một cách di chuyển giữa hai điểm trong thành phố Manhattan: bạn chỉ có thể đi theo các đường ngang và dọc, không thể đi chéo.

- Khoảng cách Minkowski: Minkowski là một loại độ đo khoảng cách được sử dụng để đo khoảng cách giữa hai điểm trong không gian nhiều chiều. Công thức của độ đo Minkowski được xác định như sau:

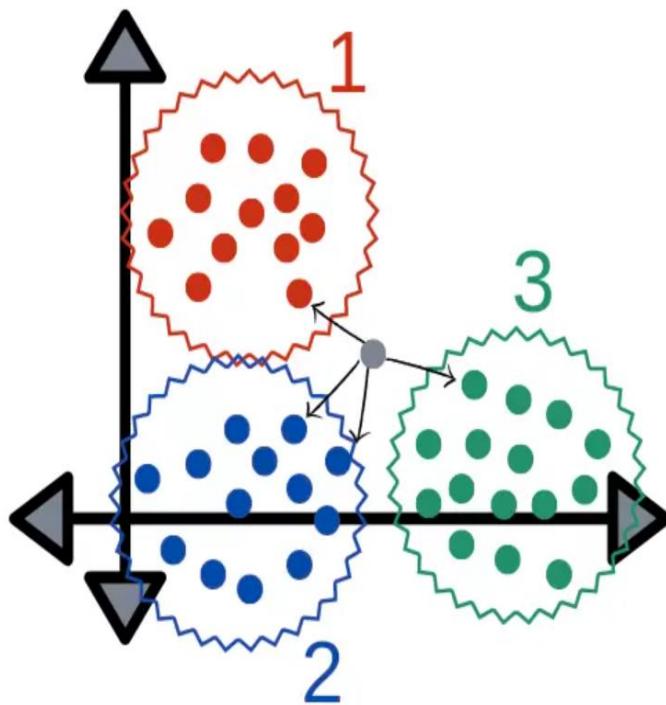
$$D(x, y) = \left(\sum_{i=1}^n |x_i - y_i|^p \right)^{\frac{1}{p}}$$

Trong đó

- $D(x, y)$ là khoảng cách giữa hai điểm x và y
- n là số chiều của không gian,
- x_i và y_i là các thành phần tương ứng của điểm x và y , p là một tham số dương, thường được gọi là "bậc" (power) hoặc "độ dài bậc p ".

Khi $1p=1$, ta có độ đo Manhattan (khoảng cách theo chiều ngang và dọc), khi $2p=2$, ta có độ đo Euclidean, và khi $p \rightarrow \infty$, ta có độ đo Chebyshev (tìm khoảng cách tối đa giữa các chiều). Độ đo Minkowski cho phép điều chỉnh độ nhạy của khoảng cách dựa trên giá trị của tham số p . Khi p tăng lên, độ đo Minkowski trở nên nhạy cảm hơn đối với sự khác biệt lớn trong các chiều của không gian.

- Xác định 'K' hàng xóm gần nhất: KNN chọn 'K' điểm dữ liệu gần nhất (hàng xóm gần nhất) dựa trên khoảng cách tính được.



Hình 1. 4 Minh họa K

- Đa số phiếu bầu: KNN sử dụng đa số phiếu bầu để quyết định nhãn cho điểm dữ liệu mới. Điểm dữ liệu mới sẽ được gán nhãn bằng nhãn xuất hiện nhiều nhất trong 'K' hàng xóm gần nhất.
- Đánh giá mô hình: Đánh giá hiệu suất của mô hình bằng cách đo lường độ chính xác, độ phủ (recall), độ chính xác (precision) và các độ đo khác trên tập dữ liệu kiểm tra hoặc sử dụng các kỹ thuật cross-validation.
- Tinh chỉnh tham số: Nếu kết quả đánh giá không đạt yêu cầu, bạn có thể tinh chỉnh tham số như K, phương pháp đo khoảng cách hoặc tiền xử lý dữ liệu để cải thiện hiệu suất của mô hình.

2.4 Minh họa thuật toán

2.4.1 Dữ liệu

Tuổi	Thu nhập	Đối tượng khách hàng
15	30 Triệu	Khách hàng không tiềm năng
20	100 Triệu	Khách hàng tiềm năng
22	70 Triệu	Khách hàng không tiềm năng
24	120 Triệu	Khách hàng tiềm năng



30	200 Triệu	Khách hàng tiềm năng
35	110 Triệu	Khách hàng không tiềm năng
40	300 Triệu	Khách hàng tiềm năng
45	90 Triệu	Khách hàng không tiềm năng
50	400 Triệu	Khách hàng tiềm năng
55	420 Triệu	Khách hàng không tiềm năng
27	80 Triệu	???

Bảng 2. 1 Dữ liệu demo

Dữ liệu cần phân loại là (27 tuổi, thu nhập 80 Triệu). Chúng ta dự đoán xem khách hàng này sẽ là khách hàng tiềm năng hay là khách hàng không tiềm năng.

2.4.2 Các bước thực hiện

a) Với độ đo Euclidean

Bước 1: Chuẩn bị dữ liệu

Bước 2: Chọn tham số K

Trong ví dụ này chúng ta sẽ chọn tham số K=3

Bước 3: Tính khoảng cách giữa dữ liệu mới với tất cả các dữ liệu còn lại: ở đây chúng ta sẽ sử dụng công thức Euclidean để tính toán khoảng cách:

Euclidean
$$\sqrt{\sum_{i=1}^k (x_i - y_i)^2}$$

$$\text{Euclidean} = \sqrt{(27-15)^2 + (80-30)^2} = 51.4\dots$$

Tuổi	Thu nhập	Đối tượng khách hàng	Khoảng cách
15	30 Triệu	Khách hàng không tiềm năng	51.4
20	100 Triệu	Khách hàng tiềm năng	21.2



22	70 Triệu	Khách hàng không tiềm năng	11.2
24	120 Triệu	Khách hàng tiềm năng	40.1
30	200 Triệu	Khách hàng tiềm năng	120
35	110 Triệu	Khách hàng không tiềm năng	31.1
40	300 Triệu	Khách hàng tiềm năng	220.4
45	90 Triệu	Khách hàng không tiềm năng	20.6
50	400 Triệu	Khách hàng tiềm năng	320.8
55	420 Triệu	Khách hàng tiềm năng	341.1
27	80 Triệu	???	

Bảng 2. 2 Euclidean demo

Bước 4: Lựa chọn k điểm có khoảng cách gần nhất: Ở đây chúng ta thấy 3 điểm có khoảng cách gần nhất là:

Tuổi	Thu nhập	Đối tượng khách hàng	Khoảng cách
20	100 Triệu	Khách hàng tiềm năng	21.2
22	70 Triệu	Khách hàng không tiềm năng	11.2
45	90 Triệu	Khách hàng không tiềm năng	20.6

Bảng 2. 3 Phân tập khách hàng

Với 3 khoảng cách này chúng ta nhận ra rằng 3 kết quả là 2 (Khách hàng không tiềm năng) và 1 (Khách hàng tiềm năng).

Bước 5: Dự đoán dữ liệu mới dựa trên K gần nhất

Trong 3 kết quả này Khách hàng không tiềm năng xuất hiện nhiều hơn nên chúng ta đưa ra dự đoán khách hàng này là khách hàng không tiềm năng.

b) Với độ đo Manhattan

Bước 1: Chuẩn bị dữ liệu

Bước 2: Chọn tham số K

Trong ví dụ này chúng ta sẽ chọn tham số K=3



Bước 3: Tính khoảng cách giữa dữ liệu mới với tất cả các dữ liệu còn lại: ở đây chúng ta sẽ sử dụng công thức Manhattan để tính toán khoảng cách:

Manhattan
$$\sum_{i=1}^k |x_i - y_i|$$

Tuổi	Thu nhập	Đối tượng khách hàng	Khoảng cách
15	30 Triệu	Khách hàng không tiềm năng	62
20	100 Triệu	Khách hàng tiềm năng	27
22	70 Triệu	Khách hàng không tiềm năng	15
24	120 Triệu	Khách hàng tiềm năng	43
30	200 Triệu	Khách hàng tiềm năng	123
35	110 Triệu	Khách hàng không tiềm năng	38
40	300 Triệu	Khách hàng tiềm năng	233
45	90 Triệu	Khách hàng không tiềm năng	28
50	400 Triệu	Khách hàng tiềm năng	343
55	420 Triệu	Khách hàng tiềm năng	368

Bảng 2. 4 Manhattan demo

Bước 4: Lựa chọn k điểm có khoảng cách gần nhất: Ở đây chúng ta thấy 3 điểm có khoảng cách gần nhất là:

Tuổi	Thu nhập	Đối tượng khách hàng	Khoảng cách
20	100 Triệu	Khách hàng tiềm năng	27
22	70 Triệu	Khách hàng không tiềm năng	15
45	90 Triệu	Khách hàng không tiềm năng	28

Bảng 2. 5 Phân tập khách hàng manhattan



Với 3 khoảng cách này chúng ta nhận ra rằng 3 kết quả là 2 (Khách hàng không tiềm năng) và 1 (Khách hàng tiềm năng).

Bước 5: Dự đoán dữ liệu mới dựa trên K gần nhất

Trong 3 kết quả này Khách hàng không tiềm năng xuất hiện nhiều hơn nên chúng ta đưa ra dự đoán khách hàng này là khách hàng không tiềm năng.

c) Với độ đo Minkowski

Minkowski
$$\left(\sum_{i=1}^k (|x_i - y_i|)^q \right)^{1/q}$$

Trong ví dụ này:

p=1 đối với công thức Manhattan kết quả khoảng cách giống với ví dụ đã thực hiện ở trên.

p=2 đối với công thức Euclidean kết quả khoảng cách giống với ví dụ đã thực hiện ở trên.

2.4.3 Kết quả

- Với Euclidean: Kết quả dự đoán của thuật toán KNN cho dữ liệu mới là khách hàng không tiềm năng.
- Với Manhattan: Kết quả dự đoán của thuật toán KNN cho dữ liệu mới là khách hàng không tiềm năng.
- Với Minkowski : Kết quả dự đoán của thuật toán KNN cho dữ liệu mới là khách hàng không tiềm năng.

III. CHUẨN BỊ DỮ LIỆU

1. Giới thiệu dữ liệu

- Insert các thư viện được sử dụng trong bài làm.



```
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn import metrics
sns.set(style='darkgrid', font_scale=1.4)
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.model_selection import train_test_split
from collections import Counter
import time
import optuna
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
# Import thư viện sklearn nhằm mục đích so sánh với mô hình tự xây dựng
from sklearn.neighbors import KNeighborsClassifier
```

Hình 3. 1 Insert thư viện sử dụng

- Tải dataset.

```
df = pd.read_csv('../data/data.csv')
```

Hình 3. 2 Tải dataset

- Mô tả từng cột dữ liệu trong dataset

```
# Mô tả kiểu dữ liệu của từng cột trong data set
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 32 columns):
 #   Column            Non-Null Count  Dtype  
--- 
 0   id                569 non-null    int64  
 1   diagnosis         569 non-null    object  
 2   radius_mean       569 non-null    float64 
 3   texture_mean     569 non-null    float64 
 4   perimeter_mean   569 non-null    float64 
 5   area_mean         569 non-null    float64 
 6   smoothness_mean  569 non-null    float64 
 7   compactness_mean 569 non-null    float64 
 8   concavity_mean   569 non-null    float64 
 9   concave_points_mean 569 non-null    float64 
 10  symmetry_mean   569 non-null    float64 
 11  fractal_dimension_mean 569 non-null    float64 
 12  radius_se        569 non-null    float64 
 13  texture_se       569 non-null    float64 
 14  perimeter_se    569 non-null    float64 
 15  area_se          569 non-null    float64 
 16  smoothness_se   569 non-null    float64 
 17  compactness_se  569 non-null    float64
```

Hình 3. 3 Mô tả dữ liệu

- Xem 10 dòng đầu tiên của dataset



Hình 3. 4 10 dòng đầu của dữ liệu

2. Tiền xử lý dữ liệu

2.1. Làm sạch dữ liệu

- Kiểm tra các dữ liệu thiếu và dữ liệu trống.

```
# Kiểm tra dữ liệu thiếu
print("Null values", df.isnull().values.sum())
print("NA values:", df.isna().values.any())

Null values 0
NA values: False
```

Hình 3. 5 Kiểm tra dữ liệu trống

- Xóa dữ liệu trống.

```
df.dropna()
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800
...
564	926424	M	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390
565	926682	M	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400
566	926954	M	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251
567	927241	M	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140
568	92751	B	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000

Hình 3. 6 Xóa dữ liệu trống

- Kiểm tra dữ liệu lặp



```
# Kiểm tra dữ liệu lặp  
df.duplicated().sum()  
0
```

Hình 3. 7 Kiểm tra dữ liệu lặp

2.2. Biến đổi dữ liệu

- Đặt lại chỉ số của DataFrame và loại bỏ chỉ số cũ và loại bỏ cột có tên 'id' khỏi DataFrame và thực hiện thay đổi trực tiếp trên DataFrame hiện tại.

```
df.reset_index(drop=True)  
df.drop('id', axis=1, inplace=True)
```

Hình 3. 8 Đặt lại index

- In ra hình dữ liệu:

```
print("Total rows: ", df.shape[0])  
print("Total columns: ", df.shape[1])  
  
Total rows: 569  
Total columns: 31
```

Hình 3. 9 Đặt hình dạng của dữ liệu

- Xem lại dataset

```
df.head()  
  
diagnosis  radius_mean  texture_mean  perimeter_mean  area_mean  smoothness_mean  compactness_mean  concavity_mean  concave points_mean  symmetry_mean  fractal_dimension_mean  
0          M         17.99        10.38       122.80      1001.0         0.11840         0.27760        0.3001        0.14710  
1          M         20.57        17.77       132.90      1326.0         0.08474         0.07864        0.0869        0.07017  
2          M         19.69        21.25       130.00      1203.0         0.10960         0.15990        0.1974        0.12790  
3          M         11.42        20.38       77.58       386.1         0.14250         0.28390        0.2414        0.10520  
4          M         20.29        14.34       135.10      1297.0         0.10030         0.13280        0.1980        0.10430  
5 rows × 31 columns
```

Hình 3. 10 Xem lại dữ liệu

- Chia dữ liệu để thử nghiệm

```
X = df.drop('diagnosis', axis=1)  
y = df['diagnosis']
```

Hình 3. 11 Chia dữ liệu thử nghiệm

- Encode tập y

```
y = (y=='M').astype('int')
```



Hình 3. 12 Encode tập y

3. Mô tả dữ liệu (thống kê mô tả)

- Đặt ra các câu hỏi thống kê nhằm hiểu rõ hơn về dữ liệu như:
- Count (Số lượng): Số lượng giá trị không rỗng (non-null) trong mỗi cột.
- Mean (Trung bình): Giá trị trung bình của dữ liệu.
- Std (Độ lệch chuẩn): Độ lệch chuẩn của dữ liệu, đo độ biến động của dữ liệu.
- Min (Giá trị nhỏ nhất): Giá trị nhỏ nhất trong dữ liệu.
- Phân vị 25%, 50%, 75%: Các phân vị của dữ liệu, cung cấp thông tin về phân phối của dữ liệu.
- Max (Giá trị lớn nhất): Giá trị lớn nhất trong dữ liệu.

df.describe()									
	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	0.18
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	0.02
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	0.10
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	0.16
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	0.17
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	0.19
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	0.30

Hình 3. 13 Một số thông tin cơ bản

Giải thích các biến

- radius_mean, texture_mean, perimeter_mean, area_mean : các biến liên quan chặt chẽ tới kích thước và hình dạng của tế bào.
- smoothness_mean, compactness_mean: độ mịn và độ đặc của tế bào , phản ánh cấu trúc tế bào.
- concavity_mean, concave points_mean: Liên quan đến sự uốn cong và các điểm lõm của tế bào.
- Symmetry_mean, fractal_dimension_mean: tính đối xứng và chiều mịn của tế bào
- Các biến hậu tố _se là các giá trị lỗi chuẩn của các đặc trưng trên.
- Các biến với hậu tố _worst là các giá trị lớn nhất trong 3 lần đo của các đặc trưng trên.
- Một số quan sát rút ra:
- Tổng số quan sát là 569.

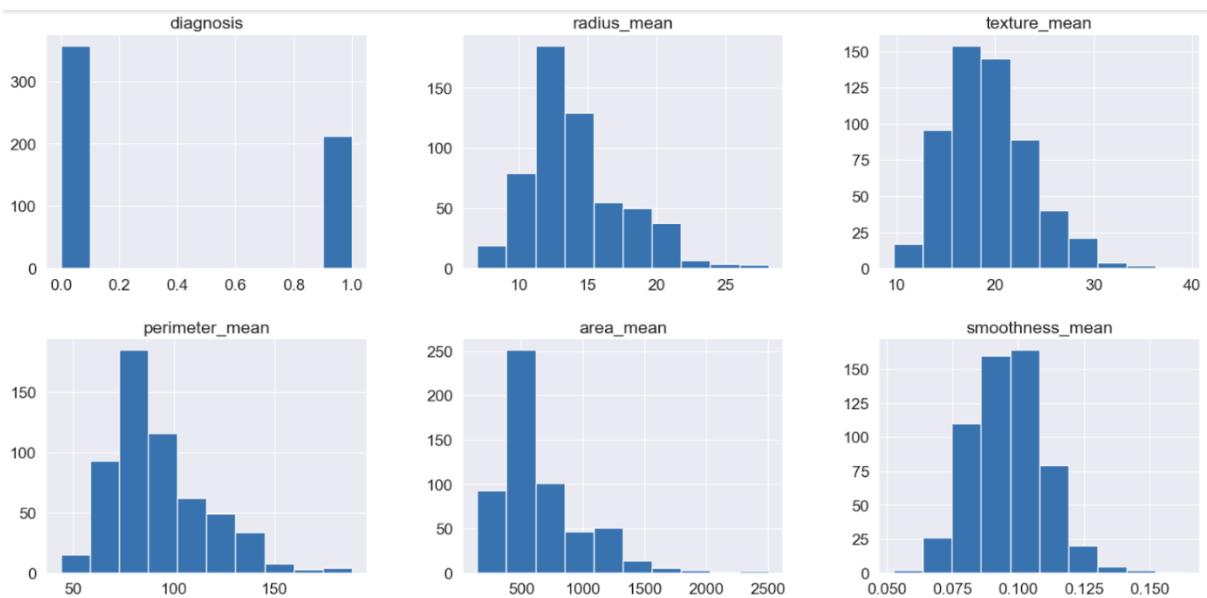


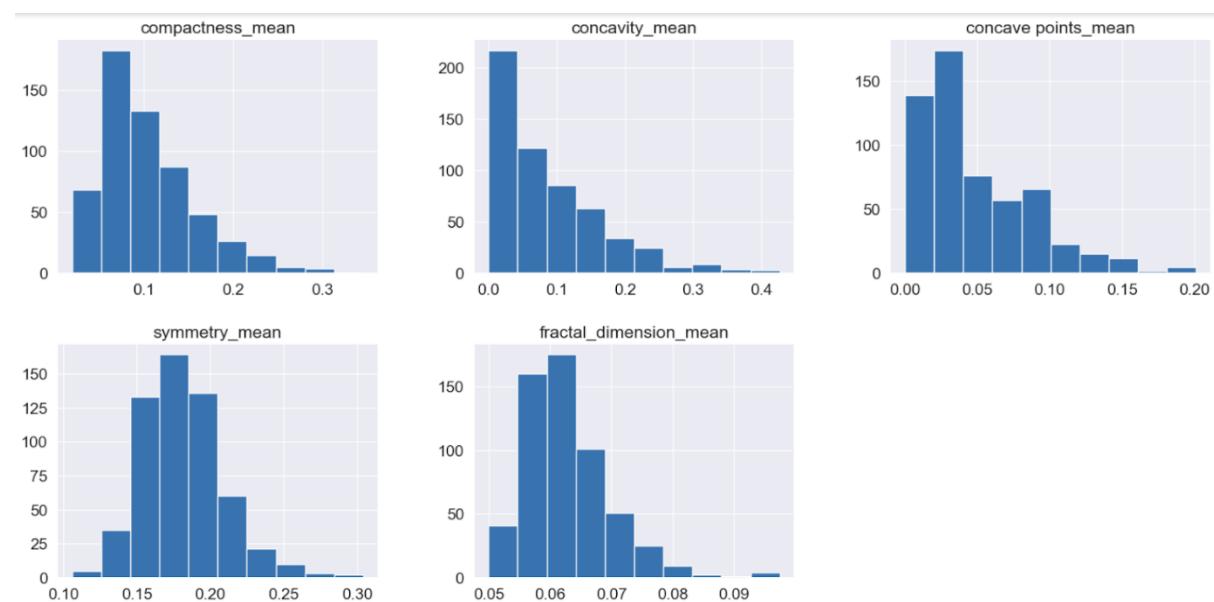
- Giá trị cao nhất, thấp nhất của các cột dữ liệu khác nhau đàng kẽ cho thấy phạm vi giá trị lớn của các biến.
- Độ lệch chuẩn của từng biến khá lớn, cho thấy mức độ biến động cao của dữ liệu.
- Kiểm tra phân phối dữ liệu

```
df_statistic = df[['diagnosis','radius_mean','texture_mean','perimeter_mean','area_mean','smoothness_mean','compactness_mean','concavity_mean','concave points_mean','symmetry_mean','fractal_dimension_mean']]  
df_statistic['diagnosis'] = (df_statistic['diagnosis'] == 'M').astype('int')
```

```
df_statistic.hist(figsize = (20,20))
```

```
array([[[<Axes: title={'center': 'diagnosis'}>,  
        <Axes: title={'center': 'radius_mean'}>,  
        <Axes: title={'center': 'texture_mean'}>],  
       [<Axes: title={'center': 'perimeter_mean'}>,  
        <Axes: title={'center': 'area_mean'}>,  
        <Axes: title={'center': 'smoothness_mean'}>],  
       [<Axes: title={'center': 'compactness_mean'}>,  
        <Axes: title={'center': 'concavity_mean'}>,  
        <Axes: title={'center': 'concave points_mean'}>],  
       [<Axes: title={'center': 'symmetry_mean'}>,  
        <Axes: title={'center': 'fractal_dimension_mean'}>, <Axes: >]],  
      dtype=object)
```

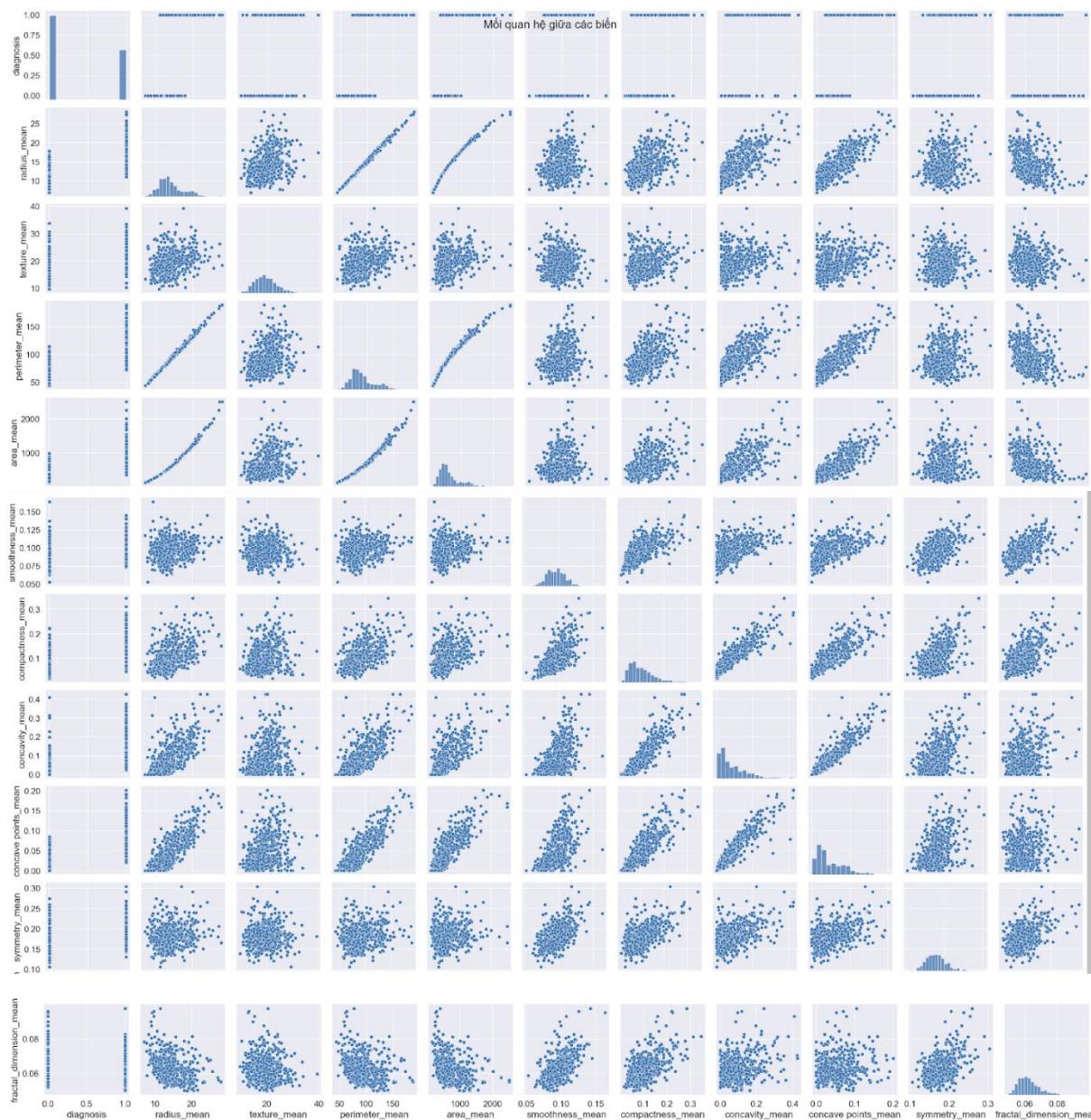




Hình 3. 14 Kiểm tra phân phối dữ liệu

- Độ tương quan giữa các biến:

```
sns.pairplot(df_statistic)
plt.suptitle('Mối quan hệ giữa các biến')
plt.show()
```

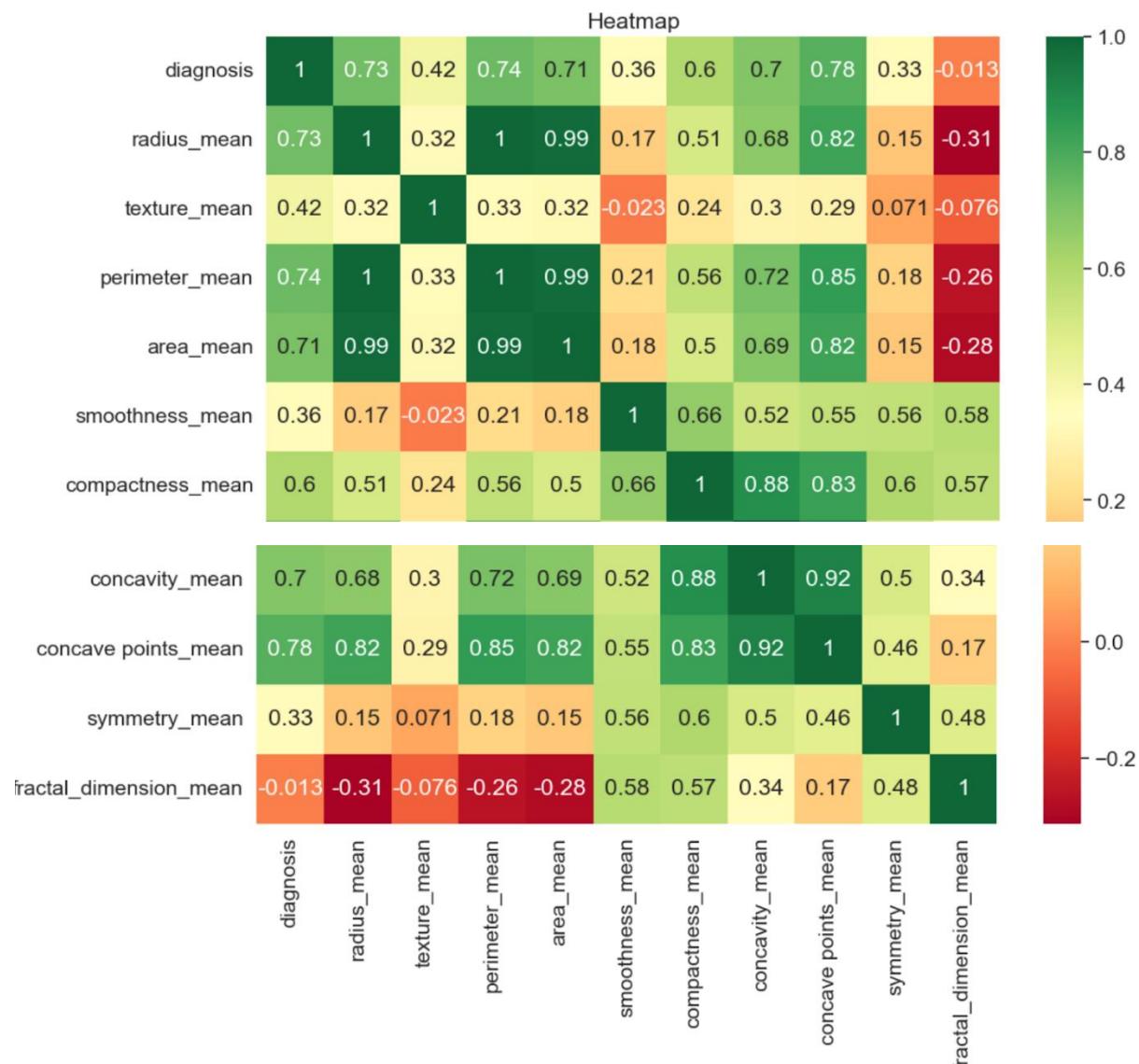


Hình 3. 15 Độ tương quan giữa các biến

Hệ số Tương quan Pearson: giúp tìm hiểu mối quan hệ giữa hai lượng. Nó cung cấp độ lưỡng về sức mạnh của mối liên kết giữa hai biến. Giá trị của Hệ số Tương quan Pearson có thể nằm trong khoảng từ -1 đến +1. Giá trị 1 có nghĩa là chúng có mối tương quan cao và giá trị 0 có nghĩa là không có tương quan.

- Vẽ biểu đồ Heatmap

```
plt.figure(figsize=(12,10))
plt.title('Heatmap')
p=sns.heatmap(df_statistic.corr(), annot=True, cmap = 'RdYlGn')
```



Hình 3. 16 Heatmap

IV. KẾT QUẢ VÀ ĐÁNH GIÁ THUẬT TOÁN

1. Xây dựng thuật toán

- Chia dữ liệu

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=4)
```

Hình 3. 17 Chia dữ liệu

- Xây dựng class chứa mô hình KNN



```
class KNN_Model():
    # Khởi tạo các biến trong hàm init
    def __init__(self, k=3, metric='euclidean', p=None):
        self.k = k
        self.metric = metric
        self.p = p

    # Xây dựng độ đo euclidean
    def euclidean(self, v1, v2):
        return np.sqrt(np.sum((v1-v2)**2))

    # Xây dựng độ đo manhattan
    def manhattan(self, v1, v2):
        return np.sum(np.abs(v1-v2))

    # Xây dựng độ đo minkowski
    def minkowski(self, v1, v2, p=2):
        return np.sum(np.abs(v1-v2)**p)**(1/p)

    # Xây dựng hàm fit
    def fit(self, X_train, y_train):
        self.X_train = X_train
        self.y_train = y_train

    # Xây dựng hàm Predict
    def predict(self, X_test):
        preds = []
        for test_row in X_test:
            nearest_neighbours = self.get_neighbours(test_row)
            majority = Counter(nearest_neighbours).most_common(1)[0][0]
            preds.append(majority)
        return np.array(preds)

    # Xây dựng hàm lấy điểm gần nhất
    def get_neighbours(self, test_row):
        distances = list()

        # Tính khoảng cách tất cả các điểm trong tập train
        for (train_row, train_class) in zip(self.X_train, self.y_train):
            if self.metric=='euclidean':
                dist = self.euclidean(train_row, test_row)
            elif self.metric=='manhattan':
                dist = self.manhattan(train_row, test_row)
            elif self.metric=='minkowski':
                dist = self.minkowski(train_row, test_row, self.p)
            else:
                raise NameError('Supported metrics are euclidean, manhattan and minkowski')
            distances.append((dist, train_class))

        # sắp xếp lại
        distances.sort(key=lambda x: x[0])

        # Xác định k
        neighbours = list()
        for i in range(self.k):
            neighbours.append(distances[i][1])

        return neighbours
```

Hình 3. 18 Xây dựng mô hình

- Dùng hàm đánh giá.

```
def accuracy(preds, y_test):
    return 100 * (preds == y_test).mean()
```

Hình 3. 19 Dụng hàm đánh giá

2. Áp dụng thuật toán cho bài toán



- Chạy mô hình đã xây dựng.

```
# Lặp qua các độ đo được xây dựng
for metric in ['euclidean', 'manhattan', 'minkowski']:
    clf = KNN_Model(k=5, metric=metric, p=2)
    clf.fit(X_train.values, y_train.values)
    preds = clf.predict(X_test.values)
    accuracy_value = accuracy(preds, y_test)
    print(f'Metric: {metric}, accuracy: {accuracy_value:.3f} %')

Metric: euclidean, accuracy: 87.719 %
Metric: manhattan, accuracy: 91.228 %
Metric: minkowski, accuracy: 87.719 %
```

Hình 3. 20 Chạy mô hình đã xây dựng

3. Thử nghiệm thuật toán

3.1. Thử nghiệm với sự thay đổi của k

- Thử nghiệm mô hình với độ đo manhattan

```
k_max = 30
accuracies = []
times = []
# Sử dụng vòng lặp thử nghiệm các giá trị k từ 0 đến k_max với accuracy
for k in range(3,k_max):
    clf = KNN_Model(k=k, metric='manhattan')
    clf.fit(X_train.values, y_train.values)

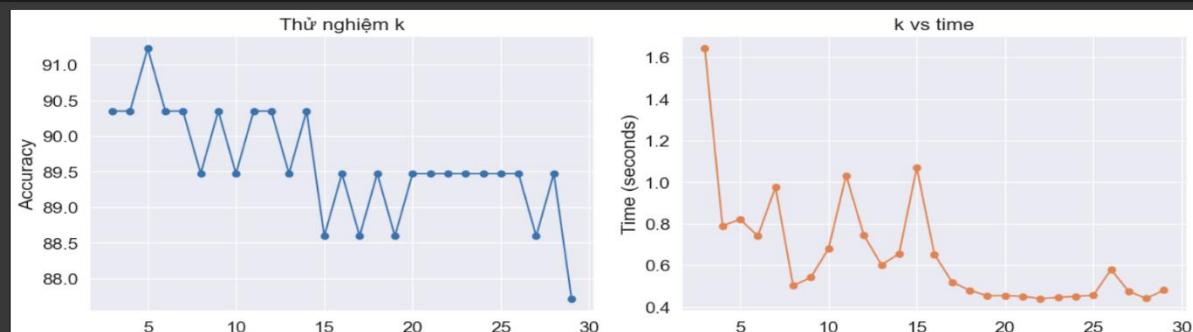
    start = time.time()
    preds = clf.predict(X_test.values)
    stop = time.time()

    acc = accuracy(preds, y_test)
    accuracies.append(acc)
    times.append(stop-start)

# Vẽ biểu đồ các giá trị accuracy tương ứng với k
plt.figure(figsize=(14,5))
plt.subplot(1,2,1)
plt.plot(range(3,k_max), accuracies, marker='o')
plt.xlabel('k')
plt.ylabel('Accuracy')
plt.title('Thử nghiệm k')

# Vẽ biểu đồ các giá trị accuracy theo thời gian
plt.subplot(1,2,2)
plt.plot(range(3,k_max), times, marker='o', c='C1')
plt.xlabel('k')
```

```
# Vẽ biểu đồ các giá trị accuracy theo thời gian
plt.subplot(1,2,2)
plt.plot(range(3,k_max), times, marker='o', c='C1')
plt.xlabel('k')
plt.ylabel('Time (seconds)')
plt.title('k vs time')
plt.tight_layout()
plt.show()
```



Hình 3. 21 Thử nghiệm sự thay đổi của k



Nhân xét

- Accuracy giảm dần từ các giá trị k nhỏ.
 - Sự thay đổi của k không ảnh hưởng rõ ràng đến sự biến động của thời gian.
- 3.2. Thử nghiệm với sự thay đổi của p
- Thử nghiệm với sự thay đổi của p độ đo Minkowski

```
p_grid = np.arange(0.5,10,0.5)
accuracies = []
times = []

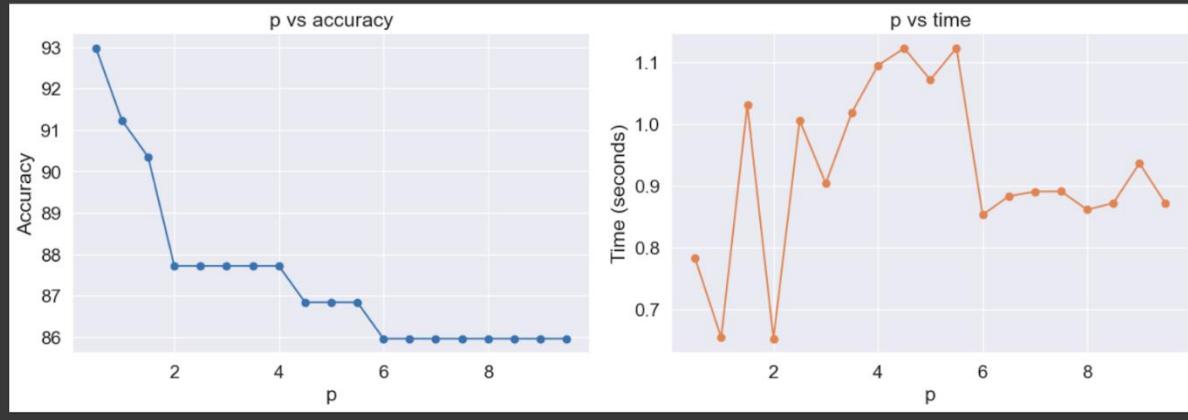
for p in p_grid:
    clf = KNN_Model(k=5, metric='minkowski',p=p)
    clf.fit(X_train.values, y_train.values)

    start = time.time()
    preds = clf.predict(X_test.values)
    stop = time.time()

    acc = accuracy(preds, y_test)
    accuracies.append(acc)
    times.append(stop-start)

plt.figure(figsize=(14,5))
plt.subplot(1,2,1)
plt.plot(p_grid, accuracies, marker='o')
plt.xlabel('p')
plt.ylabel('Accuracy')
plt.title('p vs accuracy')

plt.subplot(1,2,2)
plt.plot(p_grid, times, marker='o', c='C1')
plt.xlabel('p')
plt.ylabel('Time (seconds)')
plt.title('p vs time')
plt.tight_layout()
plt.show()
```



Hình 3. 22 Thử nghiệm với sự thay đổi của p

Nhân xét

- Độ chính xác cao nhất với các giá trị nhỏ của p và sau đó giảm đột ngột.
- Thời gian dự đoán không phụ thuộc vào p. Điều này là do phương pháp không thay đổi bất kỳ phần quan trọng nào của thuật toán.



4. Đánh giá và so sánh thuật toán

4.1 So sánh thuật toán

Sử dụng thư viện sklearn với module KNeighborsClassifier để so sánh với mô hình tự xây dựng (tham số mặc định là độ đo euclidean).

```
clf = KNeighborsClassifier(n_neighbors=5)
clf.fit(X_train.values, y_train.values)
preds = clf.predict(X_test.values)

def accuracy(preds, y_test):
    return 100 * (preds == y_test).mean()
print(F'Sklearn accuracy: {accuracy(preds, y_test):.3f} %')

Sklearn accuracy: 87.719 %
```

Hình 3. 23 So sánh thuật toán

Có thể thấy hiệu quả tương tự giữa thuật toán tự xây dựng và thuật toán của Sklearn trong việc đưa ra dự đoán đối với trường hợp này.

4.2 Đánh giá hiệu suất thuật toán

a) Confusion matrix

Confusion matrix là một kỹ thuật được sử dụng để tổng hợp hiệu suất của một thuật toán phân loại, nó có đầu ra nhị phân.

Ví dụ về bệnh ung thư:

- Các trường hợp mà bác sĩ dự đoán YES (họ có bệnh), và họ thực sự có bệnh sẽ được gọi là TRUE POSITIVES (TP). Bác sĩ đã dự đoán đúng rằng bệnh nhân có bệnh.
- Các trường hợp mà bác sĩ dự đoán NO (họ không có bệnh), và họ thực sự không có bệnh sẽ được gọi là TRUE NEGATIVES (TN). Bác sĩ đã dự đoán đúng rằng bệnh nhân không có bệnh.
- Các trường hợp mà bác sĩ dự đoán YES, nhưng họ không có bệnh sẽ được gọi là FALSE POSITIVES (FP). Còn được gọi là “Type I error”.
- Các trường hợp mà bác sĩ dự đoán NO, nhưng họ có bệnh sẽ được gọi là FALSE NEGATIVES (FN). Còn được gọi là “Type II error”.



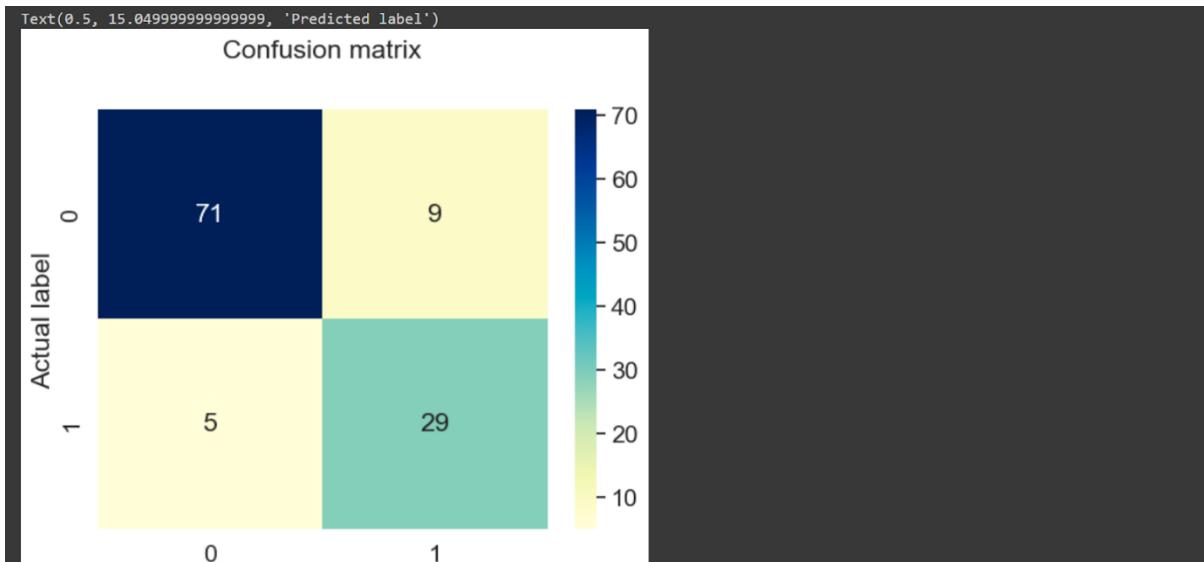
```
y_pred = clf.predict(X_test.values)

# Tính ma trận nhầm lẫn
cm = confusion_matrix(y_test.values, y_pred)

# Hiển thị ma trận nhầm lẫn
print("Confusion Matrix:")
print(pd.crosstab(y_test.values, y_pred, rownames=['True'], colnames=['Predicted'], margins=True))

Confusion Matrix:
Predicted    0   1   All
True
0           71   9   80
1           5  29   34
All         76  38  114

cnf_matrix = metrics.confusion_matrix(y_test.values, y_pred)
p = sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu", fmt='g')
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```



Hình 3. 24 Confusion matrix

b) Classification report

Báo cáo chi tiết về hiệu suất của mô hình phân loại trên tập dữ liệu kiểm thử:

- Precision (Độ chính xác): Là tỷ lệ giữa số lượng dự đoán đúng tích cực (True Positives) và tổng số lượng dự đoán tích cực (True Positives + False Positives). Precision cao đồng nghĩa với việc có ít dự đoán dương giả mạo.
- Recall (Tỉ lệ nhận diện): Là tỷ lệ giữa số lượng dự đoán đúng tích cực (True Positives) và tổng số lượng thực sự tích cực (True Positives + False Negatives). Recall cao đồng nghĩa với việc có ít trường hợp tích cực thực sự bị bỏ sót.
- F1-Score: Là một trung bình điều hòa giữa Precision và Recall. Nó được tính bằng công thức: $F1\text{-Score} = 2 * (\text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall})$. F1-Score cao nếu cả Precision và Recall đều cao.
- Support: Số lượng mẫu thực sự thuộc mỗi lớp.



Hình 3. 25 Classification report

Nhân xét

- Precision (Độ chính xác): Cho lớp 0: 93% - Độ chính xác cao, chỉ có một tỷ lệ nhỏ các trường hợp dự đoán tích cực nhưng thực sự là âm. Cho lớp 1: 76% - Có một số dự đoán tích cực nhưng thực sự là âm.
- Recall (Tỉ lệ nhận diện): Cho lớp 0: 89% - Tỉ lệ đánh giá cao, chỉ có một tỷ lệ nhỏ các trường hợp thực sự tích cực bị bỏ sót. Cho lớp 1: 85% - Tỉ lệ đánh giá khá cao.
- F1-Score: Cho lớp 0: 91% - F1-score là một trung bình điều hòa giữa precision và recall, độ chính xác cao. Cho lớp 1: 81% - F1-score có vẻ giảm do precision và recall không cân bằng.
- Accuracy (Độ chính xác tổng thể): 88% - Tỉ lệ dự đoán đúng trên toàn bộ tập kiểm thử.

5. Tối ưu hóa siêu tham số:

- Với độ đo Euclidean:

```
# Dùng hàm tối ưu
def objective_mks(trial):
    k = trial.suggest_int('k', 1, 30)

    clf = KNN_Model(k=k, metric='euclidean')
    clf.fit(X_train.values, y_train.values)

    preds = clf.predict(X_test.values)
    acc = accuracy(y_test.values, preds)

    return acc

study = optuna.create_study(direction='maximize')
study.optimize(objective_mks, n_trials=100)

best_params = study.best_params
best_k = best_params['k']

best_clf = KNN_Model(k=best_k, metric='euclidean')
best_clf.fit(X_train.values, y_train.values)
best_preds = best_clf.predict(X_test.values)
best_accuracy = accuracy(y_test.values, best_preds)

print(f"Giá trị k tốt nhất: {best_k}")
print(f"Accuracy tốt nhất : {best_accuracy:.3f}")

Giá trị k tốt nhất: 9
Accuracy tốt nhất : 89.474
```

Hình 3. 26 Tối ưu hóa tham số Euclidean



- Với độ đo Manhattan

```
# Dùng hàm tối ưu
def objective_mht(trial):
    k = trial.suggest_int('k', 1, 30)

    clf = KNN_Model(k=k, metric='manhattan')
    clf.fit(X_train.values, y_train.values)

    preds = clf.predict(X_test.values)
    acc = accuracy(y_test.values, preds)

    return acc

study = optuna.create_study(direction='maximize')
study.optimize(objective_mht, n_trials=100)

best_params = study.best_params
best_k = best_params['k']

best_clf = KNN_Model(k=best_k, metric='manhattan')
best_clf.fit(X_train.values, y_train.values)
best_preds = best_clf.predict(X_test.values)
best_accuracy = accuracy(y_test.values, best_preds)

print(f"Giá trị k tốt nhất: {best_k}")
print(f"Accuracy tốt nhất : {best_accuracy:.3f}")

Giá trị k tốt nhất: 5
Accuracy tốt nhất : 91.228
```

Hình 3. 27 Tối ưu hóa tham số với độ đo Manhattan

- Đôi với độ đo Minkowski.

```
# Dùng hàm tối ưu
def objective_mks(trial):
    p = trial.suggest_float('p', 0.5, 10.0, step=0.5)
    k = trial.suggest_int('k', 1, 30)

    clf = KNN_Model(k=k, metric='minkowski', p=p)
    clf.fit(X_train.values, y_train.values)

    preds = clf.predict(X_test.values)
    acc = accuracy(y_test.values, preds)

    return acc

# Tiên hành tối ưu
study = optuna.create_study(direction='maximize')
study.optimize(objective_mks, n_trials=100)

best_params = study.best_params
best_p = best_params['p']
best_k = best_params['k']

best_clf = KNN_Model(k=best_k, p=best_p, metric='minkowski')
best_clf.fit(X_train.values, y_train.values)
best_preds = best_clf.predict(X_test.values)
best_accuracy = accuracy(y_test.values, best_preds)

print(f"Giá trị p - k tốt nhất: {best_p}, k: {best_k}")
print(f"Accuracy tốt nhất : {best_accuracy:.3f}")

Giá trị p - k tốt nhất: 0.5, k: 10
Accuracy tốt nhất : 91.228
```

Hình 3. 28 Tối ưu hóa tham số với độ đo Minkowski



V. GIAO DIỆN DEMO THUẬT TOÁN

1. Giới thiệu công cụ và trang web demo

Những công cụ chính mà nhóm sử dụng để thiết lập trang web phân tích và dự báo chứng khoán bao gồm:

- Streamlit: Thiết kế khung trang web, là framework chính để thiết các phần tử của trang như sidebar, các trường input, option, tiêu đề, đồng thời chuyển kết quả lên server lưu trữ.
- Plotly: giống như matplotlib, thư viện này dùng để tạo ra các biểu đồ thể hiện dữ liệu, tuy nhiên thư viện này cho phép tạo ra các biểu đồ có tính tương tác cao.

Nhóm đã tiến hành đưa thiết kế giao diện lên nền tảng Streamlit (nền tảng hỗ trợ public trang web có sử dụng streamlit). Việc đưa phần thiết kế được thực hiện thông qua những bước sau:

- Thiết kế giao diện local, chỉnh sửa, chạy thử các model và các biểu đồ.
- Đăng kí nền tảng Streamlit bằng tài khoản Github.
- Sử dụng kho của Github làm nơi lưu trữ data và các file sử dụng cho trang web.
- Tiến hành xây dựng file requirements.txt để khai báo cho Streamlit server biết các thư viện với các phiên bản tương ứng của python để thiết lập môi trường chính của trang web.
- Reset lại cloud server của Streamlit để tải giao diện từ kho trên Github.

Kết quả đã được thể hiện ở trang web sau của nhóm : [DEMO-WEB-KNN](#)

2. Phương thức và hàm được sử dụng

STT	Tên class	Chức năng chính
1	TRAIN_MODELS	Thực hiện các lệnh bên trong các tab. Ở mỗi tab, class này sẽ kế thừa các phương thức từ các class tương ứng để thực hiện.
2	DESCRIPTIVE_STATISTICS	Bao gồm nhiều hàm được xây dựng để tạo ra các biểu đồ ở tab thống kê mô tả của trang web
3	KNN_MODELS	Class xây dựng thuật toán

Bảng 5. 1 Danh sách class

TRAIN MODELS		
STT	Tên phương thức	Chức năng chính
1	train_test_split	Chia dữ liệu thành tập test và tập train
2	accuracy	Hàm đánh giá mô hình
3	run_model_option	Nhận tham số và chạy thuật toán xây dựng ở class KNN MODELS



4	plot_confusion_matrix	Vẽ confusion matrix
5	run_knn	Hàm nhập tham số và điều hướng hiển thị

Bảng 5. 2 Các phương thức của class TRAIN_MODELS

DESCRIPTIVE_STATISTICS		
STT	Tên phương thức	Chức năng chính
1	describe	Hàm thống kê mô tả, xuất ra các chỉ số cơ bản
2	subplot_histograms	Hàm vẽ Histogram
3	pairplot	Vẽ pairplot
4	heatmap	Vẽ heatmap

Bảng 5. 3 Các phương thức của class DESCRIPTIVE_STATISTICS

KNN_MODELS		
STT	Tên phương thức	Chức năng chính
1	euclidean	Xây dựng độ đo euclidean
2	manhattan	Xây dựng độ đo manhattan
3	minkowski	Xây dựng độ đo min kowski
4	fit	Hàm fit mô hình
5	predict	Hàm dự báo
6	get_neighbours	Lấy điểm gần nhất

Bảng 5. 4 Các phương thức của class KNN_MODELS

CÁC PHƯƠNG THỨC KHÁC		
STT	Tên phương thức	Chức năng chính
1	run	Hàm tải giao diện
2	streamlit_menu	Cấu hình menu
3	upload_data	Hàm load dữ liệu
4	embed_image	Lấy đường dẫn ảnh
5	display_file_content	Lấy đường dẫn file
6	introduction	Hàm điều hướng giao diện overview
7	main	Hàm điều hướng thao tác từ menu

Bảng 5. 5 Các phương thức khác

3. Kết quả thực hiện

3.1 Giao diện trang overview

TEAM 1

BÁO CÁO ĐÓ ÁN – NHÓM 3

Diagnosing cancer with the KNN algorithm

K-Nearest Neighbors

Total Rows: 569 rows

Total Columns: 31 columns

Dimension: (569, 31)

Overview

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean
0	M	17.99	10.38	122.8	1,001	0.1184	0.2776	0.3001	0.1471
1	M	20.57	17.77	132.9	1,326	0.0847	0.0786	0.0869	0.0702
2	M	19.69	21.25	130	1,203	0.1096	0.1599	0.1974	0.1279
3	M	11.42	20.38	77.58	386.1	0.1425	0.2839	0.2414	0.1052
4	M	20.29	14.34	135.1	1,297	0.1003	0.1328	0.198	0.1043

x_2 Before training x_2 After training

< Manage app

Bảng 5. 6 Giao diện overview

3.2 Giao diện trang descriptive statistics

TEAM 1

BÁO CÁO ĐÓ ÁN – NHÓM 3

Diagnosing cancer with the KNN algorithm

Thống kê mô tả

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean
count	569	569	569	569	569	569	569	569	569
mean	14.1273	19.2896	91.969	654.8891	0.0964	0.1043	0.0888	0.0489	0.
std	3.524	4.301	24.299	351.9141	0.0141	0.0528	0.0797	0.0388	0.
min	6.981	9.71	43.79	143.5	0.0526	0.0194	0	0	0
25%	11.7	16.17	75.17	420.3	0.0864	0.0649	0.0296	0.0203	0.
50%	13.37	18.84	86.24	551.1	0.0959	0.0926	0.0615	0.0335	0.
75%	15.78	21.8	104.1	782.7	0.1053	0.1304	0.1307	0.074	0.
max	28.11	39.28	188.5	2,501	0.1634	0.3454	0.4268	0.2012	0.

Histogram

Histogram Subplots

diagnosis radius_mean texture_mean perimeter_mean

Bảng 5. 7 Giao diện trang descriptive statistic

3.3 Giao diện trang model



Bảng 5. 8 Giao diện trang model

3.4 Giao diện trang contact

Bảng 5. 9 Giao diện trang Contact

VI. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1. Kết luận

Việc nghiên cứu ví dụ là công tác chẩn đoán ung thư vú đã phản ánh được hiệu suất cũng như những đặc điểm của mô hình knearestneibough . Việc sử dụng thuật



toán tuy đơn giản những vẫn mang lại hiệu quả cao đối với những trường hợp đặc thù. Trong quá trình nghiên cứu và phát triển thuật toán, nhóm đã rút ra được những đặc điểm sau nếu áp dụng thuật toán vào thực tế cuộc sống:

Về ưu điểm:

- Việc triển khai và sử dụng KNN không đòi hỏi nhiều kiến thức toán học phức tạp.
- KNN linh hoạt trong việc sử dụng trên nhiều dữ liệu bởi nó không phụ thuộc nhiều vào phân phối.
- Có thể chia nhóm dữ liệu có cấu trúc phức tạp và không tuyến tính.

Về nhược điểm:

- Có thể nhạy cảm với dữ liệu nhiễu và có thể bị ảnh hưởng bởi chiều cao khác biệt của các đặc trưng.
- Cần lưu toàn bộ dữ liệu đào tạo trong bộ nhớ. Điều này có thể tạo ra vấn đề về hiệu suất khi dữ liệu lớn.
- Khi số chiều của dữ liệu tăng, khoảng cách giữa các điểm dữ liệu có thể trở nên không hiệu quả và dẫn đến hiệu suất kém.
- Việc chọn giá trị k phù hợp có thể ảnh hưởng đến kết quả của thuật toán. Giá trị k quá nhỏ có thể dẫn đến tình trạng quá mức tương quan, trong khi giá trị k quá lớn có thể làm giảm độ chính xác.

2. Hướng phát triển

Dựa trên những phân tích của nhóm, các yếu tố đánh giá ở trên, nhóm đưa ra những hướng phát triển khả dụng sau cho đề tài:

- Support Vector Machine (SVM): SVM là một mô hình mạnh mẽ cho việc phân loại và học máy.
- Random Forest: Random Forest là một mô hình thuật toán học tập dựa trên cây quyết định, có khả năng xử lý cả dữ liệu có cấu trúc và không có cấu trúc.
- Neural Networks (Deep Learning): Mạng nơ-ron sâu (Deep Neural Networks) có khả năng học tự động các đặc trưng phức tạp từ dữ liệu.
- Logistic Regression: Phù hợp cho các nhiệm vụ phân loại nhị phân và có thể được mở rộng cho nhiều lớp.
- Mô Hình Học Sâu (Deep Learning): Sử dụng các mô hình học sâu như mạng nơ-ron tích chập (CNN) để tự động rút trích đặc trưng từ hình ảnh và cải thiện khả năng chẩn đoán.
- Mô Hình Học Máy Tăng Cường (Ensemble Models): Kết hợp nhiều mô hình học máy để cải thiện độ chính xác và ổn định.



VII. TÀI LIỆU THAM KHẢO

[1] *CodeLearn, Thuật Toán K-nearest neighbors (KNN) siêu Cơ Bản.* Retrieve from https://codelearn.io/sharing/thuat-toan-k-nearest-neighbors-knn?fbclid=IwAR2vI2Ca6dsLMbCBh4sWbIZZN6hPvqZaJ6CjEklb-wVixYw3px_XtoSF-E

[2] *Trịnh D.D, Giới Thiệu về Thuật Toán Knn 2024.* Retrieve from <https://websitehcm.com/gioi-thieu-ve-thuat-toan-knn/>

[3] *Nguyễn & Trần, Tối ưu Mô Hình Phân lớp dữ liệu dựa trên thuật toán k nearest neighbor 1970) .* Retrieve from <http://elib.vku.udn.vn/bitstream/123456789/765/1/B32.227-232.pdf>

[4] *Srivastava, A complete guide to K-Nearest Neighbors (2024).* Retrieve from <https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/>

[5] *Dvms.vn, Thuật Toán Knn và Ví Dụ đơn Giản trong Ngành Ngân Hàng(2023).* Retrieve from <https://dvms.com.vn/tin-tuc/big-data/76009-thuat-toan-knn-va-vi-du-don-gian-trong-nganh-ngan-hang.html#gsc.tab=0>

[6] *Vu, Bài 6: K-Nearest Neighbors (2017).* Retrieve from <https://machinelearningcoban.com/2017/01/08/knn/?fbclid=IwAR07LReBw7K3Ycg1Cxz6r5QbSOJgKOz-ypW5X5A9ugGkVcba3pdtdhAO49s#-gioi-thieu>

VIII. PHỤ LỤC – CÁC ĐƯỜNG DẪN QUAN TRỌNG

LINK DỰ ÁN CỦA NHÓM TRÊN GITHUB: [GITHUB-DIANOSING CANCER WITH KNN](https://github.com/DIANOSING-CANCER-WITH-KNN)

LINK DỮ LIỆU CHÍNH : [UCI-EDU-DIANOSTIC](https://archive.ics.uci.edu/ml/datasets/UCI+EDU+DIANOSTIC)

LINK TRANG WEB DEMO: [DEMO-WEB-KNN](https://www.thesourcecodehub.com/demo-web-knn)