

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**  
**TRƯỜNG ĐẠI HỌC NGUYỄN TẤT THÀNH**  
**KHOA CÔNG NGHỆ THÔNG TIN**

---



**TIỂU LUẬN KHAI THÁC DỮ LIỆU VÀ ỨNG DỤNG**  
**MÔ HÌNH DỰ ĐOÁN BỆNH UNG THƯ TUYẾN TIỀN LIỆT**  
**BẰNG PHƯƠNG PHÁP K-NN**

**Giảng viên giảng dạy: TS Võ Thị Hồng Thắm**

**Sinh viên thực hiện : Chu Doãn Đức**

**MSSV : 2000003917**

**Môn học : Khai thác dữ liệu và ứng dụng**

**Khóa : 2020**

**Tp.HCM, tháng 5 năm 2023**

## LỜI CẢM ƠN

Đầu tiên, em xin gửi lời cảm ơn chân thành đến Trường Đại học Nguyễn Tất Thành đã đưa môn học Khai thác dữ liệu và ứng dụng vào trương trình giảng dạy. Đặc biệt, em xin gửi lời cảm ơn sâu sắc đến giảng viên bộ môn – Cô Võ Thị Hồng Thắm đã dạy dỗ, truyền đạt những kiến thức quý báu cho em trong suốt thời gian học tập vừa qua. Trong thời gian tham gia lớp học của thầy, em đã có thêm cho mình nhiều kiến thức bổ ích, tinh thần học tập hiệu quả, nghiêm túc và đã cho em chắc chắn được hoạch định tương lai của mình.

Bộ môn Khai thác dữ liệu và ứng dụng là môn học thú vị, vô cùng bổ ích và có tính thực tế cao. Đảm bảo cung cấp đủ kiến thức, gắn liền với nhu cầu thực tiễn của sinh viên nói chung và bản thân em nói riêng. Tuy nhiên, do vốn kiến thức còn nhiều hạn chế và khả năng tiếp thu thực tế còn nhiều bỡ ngỡ và hạn hẹp. Mặc dù em đã cố gắng hết sức nhưng chắc chắn bài báo của em khó có thể tránh khỏi những thiếu sót và nhiều chỗ còn chưa chính xác, kính mong các thầy/cô chấm bài xem xét và góp ý để bài tiểu luận của em được hoàn thiện hơn.

Kính chúc thầy có nhiều sức khỏe, hạnh phúc, thành công trên con đường giảng dạy  
Em xin chân thành cảm ơn!

TRƯỜNG ĐẠI HỌC NGUYỄN TẤT THÀNH

KỲ THI KẾT THÚC HỌC PHẦN

TRUNG TÂM KHẢO THÍ

HỌC KỲ II - NĂM HỌC 2022 - 2023

**PHIẾU CHẤM THI TIỂU LUẬN/ĐỒ ÁN**

Môn thi: Khai thác dữ liệu và ứng dụng ..... Lớp học phần: 20DTH2A .....

Nhóm sinh viên thực hiện: .....

1. Chu Doãn Đức ..... Tham gia đóng góp: 100% .....

2. Lê Gia Bảo ..... Tham gia đóng góp: 100% .....

3. .... Tham gia đóng góp:.....

4. .... Tham gia đóng góp:.....

Ngày thi: ngày 29 tháng 05 năm 2023 ..... Phòng thi: L.505 .....

Đề tài tiểu luận/báo cáo của sinh viên: Mô hình dự đoán bệnh ung thư tuyến tiền liệt bằng phương pháp K-NN .....

Phản đánh giá của giảng viên (căn cứ trên thang rubrics của môn học):

Tiêu chí (theo CDR HP)	Đánh giá của GV	Điểm tối đa	Điểm đạt được
Cấu trúc của báo cáo	..... .....		
Nội dung			
- Các nội dung thành phần	..... .....		
- Lập luận	..... .....		
- Kết luận	.....		
Trình bày	.....		
<b>TỔNG ĐIỂM</b>			

**Giảng viên chấm thi**

(ký, ghi rõ họ tên)

## LỜI MỞ ĐẦU

Bệnh ung thư tuyến tiền liệt là một trong những loại ung thư phổ biến nhất ở nam giới. Việc phát hiện sớm bệnh ung thư tuyến tiền liệt có thể giúp cải thiện tỷ lệ sống sót và chất lượng cuộc sống của bệnh nhân. Tuy nhiên, các phương pháp chẩn đoán hiện tại như sinh thiết hay PSA vẫn còn nhiều hạn chế về độ chính xác, độ nhạy và độ đặc hiệu.

Do đó, nghiên cứu phát triển các mô hình dự đoán bệnh ung thư tuyến tiền liệt dựa trên các dữ liệu lâm sàng là rất cần thiết. Trong bài báo này, chúng tôi đề xuất một mô hình dự đoán bệnh ung thư tuyến tiền liệt bằng phương pháp K-NN (K-Nearest Neighbor). Phương pháp K-NN là một phương pháp học máy không tham số, dựa trên khoảng cách giữa các điểm dữ liệu để xác định nhãn cho điểm dữ liệu mới dùng để phân loại quan sát mới bằng cách tìm điểm tương đồng giữa quan sát mới này với dữ liệu sẵn có.

Mô hình KNN không chỉ đơn thuần là một công cụ dự đoán, mà còn mang trong mình tiềm năng để tìm hiểu sâu về mối quan hệ giữa các đặc trưng và bệnh tình. Qua việc phân tích cơ bản và mô phỏng sự tương đồng, KNN có thể giúp chúng ta nhìn thấy những yếu tố quan trọng trong quá trình phát triển ung thư tuyến tiền liệt. Điều này đặc biệt hữu ích để xác định các yếu tố nguy cơ và đưa ra các biện pháp phòng ngừa hiệu quả.

Chúng tôi sử dụng các dữ liệu có sẵn của thư viện python trong đó bao gồm dữ liệu của 569 bệnh nhân để huấn luyện và kiểm tra mô hình của chúng tôi.

[illegible]

**Giảng viên giảng dạy**  
(Ký tên và ghi rõ họ tên)

# MỤC LỤC

LỜI CẢM ƠN.....	
LỜI MỞ ĐẦU.....	
CHƯƠNG 1: GIỚI THIỆU .....	1
1.1 – Giới thiệu đề tài:.....	1
1.2 – Lý do chọn đề tài:.....	1
1.3 – Mục tiêu của đề tài: .....	2
1.4 – Công nghệ áp dụng:.....	2
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT .....	3
2.1 – Tổng quan về khai phá dữ liệu (Data Mining): .....	3
2.1.1 – Giới thiệu Data Mining: .....	3
2.1.2 - Ứng dụng: .....	3
2.2 – Tổng quan về thuật toán K-Nearest Neighbors (K-NN): .....	3
2.2.1 – Giới thiệu K – NN: .....	3
2.2.2 – Thuật toán K-NN:.....	4
2.2.3 – Ưu và nhược điểm của thuật toán:.....	4
CHƯƠNG 3: XÂY DỰNG GIẢI THUẬT/MÔ HÌNH VÀ THỰC NGHIỆM .....	6
3.1 – Khai báo thư viện và dữ liệu:.....	6
3.2 – Khám phá dữ liệu: .....	7
3.2.1 – Tổng quát về dữ liệu: .....	7
3.2.2 – Tên tính năng và ý nghĩa:.....	8
3.2.3 – Làm sạch và sắp xếp lại dữ liệu: .....	9
3.2 – Trực quan hóa dữ liệu:.....	10
3.3 – Khởi tạo mô hình:.....	15
3.3 – Khởi tạo mô hình:.....	16
3.6 – Mô hình K-NN: .....	20
3.7 – Thực nghiệm: .....	21
CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN .....	23
4.1 – Kết luận:.....	23
4.2 – Hướng phát triển: .....	23
Tài liệu tham khảo .....	26

# CHƯƠNG 1: GIỚI THIỆU

## 1.1 – Giới thiệu đề tài:

Trong những năm gần đây, ung thư tiền liệt tuyến là một trong những bệnh ung thư thường gặp, hiện đứng thứ 2 về tỷ lệ mắc và thứ 5 về tỷ lệ tử vong trong các bệnh ung thư ở nam giới. Ước tính có 1.3 triệu ca mới mắc và 359.000 ca chết liên quan đến ung thư tiền liệt tuyến trên thế giới vào năm 2018. Tại Việt Nam năm 2012, theo số liệu của cơ quan nghiên cứu ung thư thế giới, tỷ lệ mắc và tử vong chuẩn theo tuổi lần lượt là 3.4 và 2.5/ 100.000 dân. Việc chẩn đoán sớm và dự báo chính xác bệnh ung thư tuyến tiền liệt đóng vai trò quan trọng trong việc tăng khả năng chữa trị và cải thiện chất lượng cuộc sống cho các bệnh nhân. Trong lĩnh vực này, mô hình dự đoán bệnh ung thư tuyến tiền liệt bằng phương pháp K-Nearest Neighbor (KNN) đã thu hút sự quan tâm đáng kể.

## 1.2 – Lý do chọn đề tài:

Chúng tôi đã chọn đề tài “Mô hình dự đoán bệnh ung thư tuyến tiền liệt bằng phương pháp K-NN” vì chúng tôi muốn học cách sử dụng Python để giải quyết một bài toán dự báo chuỗi thời gian thực tế. Đồng muốn khám phá các kỹ thuật khai phá dữ liệu cũng như các phương pháp khai phá. Chúng tôi hy vọng rằng bằng cách làm đề tài này, sẽ nâng cao kỹ năng lập trình Python của mình và hiểu sâu hơn về khai phá dữ liệu. Chúng tôi cũng mong muốn áp dụng những kiến thức và kinh nghiệm từ đề tài này vào các bài toán khác trong tương lai. Chúng tôi tin rằng khai phá dữ liệu là một kỹ thuật có giá trị cao và có nhiều ứng dụng trong nhiều lĩnh vực khác nhau.

Và hơn hết, vì chúng tôi quan tâm đến việc ứng dụng học máy trong lĩnh vực y tế, đặc biệt là trong việc chẩn đoán bệnh ung thư. Chúng tôi nhận thấy rằng bệnh ung thư tuyến tiền liệt là một vấn đề sức khỏe quan trọng ở nam giới và cần có những phương pháp chẩn đoán hiệu quả hơn. Chúng tôi lựa chọn phương pháp k-nn vì nó là một phương pháp đơn giản nhưng hiệu quả, có thể xử lý được các dữ liệu không cân bằng và có tính khả diễn giải cao.

### 1.3 – Mục tiêu của đề tài:

Học cách sử dụng Python để dự đoán bệnh ung thư tuyến tiền liệt bằng phương pháp K-NN. Đây là một bài toán dự đoán thực tế, có tính ứng dụng cao trong lĩnh vực y tế công cộng. Bằng cách làm đề tài này, chúng ta sẽ nâng cao kỹ năng lập trình Python của bản thân và hiểu sâu hơn về các kỹ thuật khai phá và mô hình hóa dữ liệu. Chúng ta cũng sẽ có cơ hội so sánh hiệu quả của phương pháp K-Nearest Neighbor (KNN) với các phương pháp khác nhau và khám phá các ứng dụng khác của khai phá dữ liệu. Xây dựng và đánh giá một mô hình dự đoán bệnh ung thư tuyến tiền liệt bằng phương pháp K-NN, sử dụng dữ liệu có sẵn trên thư viện Python. Mô hình được mong đợi sẽ có độ chính xác, độ nhạy và độ đặc hiệu cao.

### 1.4 – Công nghệ áp dụng:

- NumPy: một thư viện dùng để làm việc với mảng và ma trận đa chiều. Nó cung cấp một tập hợp các chức năng và phương pháp để thao tác dữ liệu số học nhanh chóng, hiệu quả và linh hoạt.
- Pandas: một thư viện dùng để phân tích dữ liệu và xử lý dữ liệu dạng bảng. Nó cung cấp cấu trúc dữ liệu và công cụ để làm việc với dữ liệu dạng bảng, cho phép bạn đọc, ghi, xử lý và thao tác với các tập tin dữ liệu dạng bảng. Và cung cấp nhiều chức năng và phương thức để thao tác, truy vấn, lọc, xử lý, phân tích, và biến đổi dữ liệu. Nó cũng tích hợp tốt với các thư viện khác trong hệ sinh thái khoa học dữ liệu của Python như NumPy, Matplotlib và scikit-learn.
- Scikit – learn (sklearn): dùng cho machine learning và data mining. Nó cung cấp các công cụ và thuật toán để thực hiện nhiều tác vụ liên quan đến machine learning như phân loại, hồi quy, phân cụm, giảm chiều dữ liệu, và phân tích véc-tơ hỗ trợ. Scikit-learn cung cấp một loạt các thuật toán học máy tiêu biểu trong đó có K-Nearest Neighbors (K-NN).
- Google Colab: cung cấp một môi trường linh hoạt và tiện lợi để thực hiện các dự án machine learning và phân tích dữ liệu. Nó cung cấp sức mạnh tính toán và tài nguyên ngay trên trình duyệt web của bạn, giúp bạn tiết kiệm thời gian và công sức khi cài đặt và cấu hình môi trường máy tính cá nhân của mình.



## **CHƯƠNG 2: CƠ SỞ LÝ THUYẾT**

### **2.1 – Tổng quan về khai phá dữ liệu (Data Mining):**

#### **2.1.1 – Giới thiệu Data Mining:**

Data mining – khai phá dữ liệu là quá trình phân loại, sắp xếp các tập hợp dữ liệu lớn để xác định các mẫu và thiết lập các mối liên hệ nhằm giải quyết các vấn đề nhờ phân tích dữ liệu. Các MCU khai phá dữ liệu cho phép các doanh nghiệp có thể dự đoán được xu hướng tương lai.

Quá trình khai phá dữ liệu là một quá trình phức tạp bao gồm kho dữ liệu chuyên sâu cũng như các công nghệ tính toán. Hơn nữa, Data Mining không chỉ giới hạn trong việc trích xuất dữ liệu mà còn được sử dụng để chuyển đổi, làm sạch, tích hợp dữ liệu và phân tích mẫu.

Có nhiều tham số quan trọng khác nhau trong Data Mining, chẳng hạn như quy tắc kết hợp, phân loại, phân cụm và dự báo. Một số tính năng chính của Data Mining:

- Dự đoán các mẫu dựa trên xu hướng trong dữ liệu.
- Tính toán dự đoán kết quả.
- Tạo thông tin phản hồi để phân tích.
- Tập trung vào cơ sở dữ liệu lớn hơn.
- Phân cụm dữ liệu trực quan.

#### **2.1.2 - Ứng dụng:**

Có nhiều ứng dụng của Data Mining thường thấy như:

- Phân tích thị trường và chứng khoán.
- Phát hiện gian lận.
- Quản lý rủi ro và phân tích doanh nghiệp.
- Phân tích giá trị trọn đời của khách hàng.

### **2.2 – Tổng quan về thuật toán K-Nearest Neighbors (K-NN):**

#### **2.2.1 – Giới thiệu K – NN:**

KNN (K-Nearest Neighbors) là một thuật toán đơn giản nhất trong nhóm thuật toán Học có giám sát. Ý tưởng của thuật toán này đó là tìm output của một dữ liệu mới dựa

trên output của K điểm gần nhất xung quanh nó. KNN được ứng dụng nhiều trong khai phá dữ liệu và học máy. Trong thực tế, việc đo khoảng cách giữa các điểm dữ liệu, chúng ta có thể sử dụng rất nhiều độ đo, tiêu biểu như là Mahatan, O-clit, cosine,...

Thuật toán của KNN có thể được mô tả như sau:

### 2.2.2 – Thuật toán K-NN:

Các bước để hoàn thành thuật toán K-Nearest Neighbors:

- Tính khoảng cách của dữ liệu mới với những dữ liệu đã biết.
- Đưa tất cả các khoảng cách trên vào một matrix và sắp xếp từ bé đến lớn.
- Lấy K điểm nhỏ nhất ở matrix bên trên để tính tỷ lệ.
- Đưa ra đáp án.

Trong top K giá trị vừa lấy, ta thống kê số lượng của mỗi lớp, chọn phân lớp cho số lượng lớn nhất.

Một câu hỏi đặt ra đó là có phải cứ chọn K càng lớn thì càng tốt, thì câu trả lời đó là còn tùy thuộc vào dữ liệu đó như thế nào. Không phải lúc nào K càng lớn thì cho kết quả tốt và ngược lại. Việc lựa chọn tham số K của mô hình sẽ tiến hành thông qua thực nghiệm nhiều lần để chọn ra kết quả tốt nhất.

Với các bước như trên, chúng ta nhận thấy rằng thuật toán của KNN rất đơn giản, dễ thực hiện, dễ cài đặt. Việc dự đoán kết quả thật là dễ dàng, độ phức tạp của thuật toán nhỏ.

### 2.2.3 – Ưu và nhược điểm của thuật toán:

a) Ưu điểm:

- Độ phức tạp tính toán của quá trình training là bằng 0.
- Việc dự đoán kết quả của dữ liệu mới rất đơn giản.
- Không cần giả sử gì về phân phối của các class.

b) Nhược điểm:

- KNN rất nhạy cảm với nhiễu khi K nhỏ.
- Như đã nói, KNN là một thuật toán mà mọi tính toán đều nằm ở khâu test. Trong đó việc tính khoảng cách tới từng điểm dữ liệu trong training set sẽ tốn rất nhiều thời gian, đặc biệt là với các cơ sở dữ liệu có số chiều lớn và có nhiều điểm dữ

liệu. Với  $K$  càng lớn thì độ phức tạp cũng sẽ tăng lên. Ngoài ra, việc lưu toàn bộ dữ liệu trong bộ nhớ cũng ảnh hưởng tới hiệu năng của KNN.

## CHƯƠNG 3: XÂY DỰNG GIẢI THUẬT/MÔ HÌNH VÀ THỰC NGHIỆM

### 3.1 – Khai báo thư viện và dữ liệu:

Sử dụng các thư viện như pandas, numpy, matplotlib.pyplot, seaborn, sklearn.datasets, sklearn.model\_selection, sklearn.neighbors, sklearn.preprocessing và sklearn.metrics. Mỗi thư viện có chức năng khác nhau trong việc xử lý dữ liệu và phân tích dữ liệu.

```
KHAI BÁO THƯ VIỆN

[29] import pandas as pd #Đọc dữ liệu
import numpy as np #Xử lý dữ liệu
import matplotlib.pyplot as plt #Vẽ biểu đồ
import seaborn as sns #Vẽ biểu đồ thống kê.
import sklearn.datasets #Truy cập vào các tập dữ liệu mẫu có sẵn.
from sklearn.model_selection import train_test_split #Chia dữ liệu thành tập huấn luyện và tập kiểm tra
from sklearn.neighbors import KNeighborsClassifier #Mô hình phân loại k-nearest neighbors
from sklearn.preprocessing import MinMaxScaler #Tỉ lệ hóa dữ liệu
from sklearn.metrics import accuracy_score #Tính toán độ chính xác của mô hình phân loại
from sklearn.metrics import mean_squared_error #Tính toán giá trị bình phương trung bình của sai số
from sklearn.metrics import r2_score #Tính toán điểm R^2 cho mô hình hồi quy
from sklearn.metrics import confusion_matrix #tính toán điểm R^2 cho mô hình hồi quy
```

Như hình trên thư viện pandas được sử dụng để đọc và xử lý dữ liệu, thư viện numpy được sử dụng để xử lý các phép tính toán trên ma trận, thư viện matplotlib.pyplot được sử dụng để vẽ biểu đồ, thư viện seaborn được sử dụng để vẽ biểu đồ thống kê, thư viện sklearn.datasets được sử dụng để truy cập vào các tập dữ liệu mẫu có sẵn, thư viện sklearn.model\_selection được sử dụng để chia dữ liệu thành tập huấn luyện và tập kiểm tra, thư viện sklearn.neighbors được sử dụng để xây dựng mô hình phân loại k-nearest neighbors, thư viện sklearn.preprocessing được sử dụng để tỉ lệ hóa dữ liệu và cuối cùng là thư viện sklearn.metrics được sử dụng để tính toán độ chính xác của mô hình phân loại, giá trị bình phương trung bình của sai số và điểm  $R^2$  cho mô hình hồi quy.

```
[30] #Tải tập dữ liệu về ung thư vú từ sklearn.datasets.
breast_cancer_dataset = sklearn.datasets.load_breast_cancer()

[31] #Tạo một DataFrame từ dữ liệu tải về từ tập dữ liệu
df = pd.DataFrame(breast_cancer_dataset.data, columns = breast_cancer_dataset.feature_names)
```

Tải dữ liệu đã lấy về từ chính thư viện của sklearn.datasets. Sử dụng phương thức pd.DataFrame() để tạo DataFrame từ dữ liệu tải về và truyền vào dữ liệu và các cột của DataFrame.

```
[32] df.head()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst radius	worst texture	worst perimeter	worst area	worst smoothness	worst compactness	worst concavity	worst concave points	worst symmetry	worst fractal dimension
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	25.38	17.33	184.60	2019.0	0.1622	0.6656	0.7119	0.2654	0.4601	0.11890
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	24.99	23.41	158.80	1956.0	0.1238	0.1866	0.2416	0.1860	0.2790	0.08902
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	23.57	25.53	152.50	1709.0	0.1444	0.4245	0.4504	0.2430	0.3613	0.08758
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	14.91	26.50	98.87	567.7	0.2098	0.8663	0.6869	0.2575	0.6638	0.17300
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	22.54	16.67	152.20	1575.0	0.1374	0.2050	0.4000	0.1625	0.2364	0.07678

5 rows x 30 columns

## 3.2 – Khám phá dữ liệu:

### 3.2.1 – Tổng quát về dữ liệu:

```
[33] df.info() #Xác định kiểu dữ liệu
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 30 columns):
#   Column                                Non-Null Count  Dtype
---  ---                                ---
0   mean radius                           569 non-null    float64
1   mean texture                           569 non-null    float64
2   mean perimeter                         569 non-null    float64
3   mean area                             569 non-null    float64
4   mean smoothness                       569 non-null    float64
5   mean compactness                       569 non-null    float64
6   mean concavity                         569 non-null    float64
7   mean concave points                    569 non-null    float64
8   mean symmetry                         569 non-null    float64
9   mean fractal dimension                 569 non-null    float64
10  radius error                           569 non-null    float64
11  texture error                           569 non-null    float64
12  perimeter error                         569 non-null    float64
13  area error                             569 non-null    float64
14  smoothness error                       569 non-null    float64
15  compactness error                      569 non-null    float64
16  concavity error                        569 non-null    float64
17  concave points error                   569 non-null    float64
18  symmetry error                         569 non-null    float64
19  fractal dimension error                 569 non-null    float64
20  worst radius                           569 non-null    float64
21  worst texture                           569 non-null    float64
22  worst perimeter                         569 non-null    float64
23  worst area                             569 non-null    float64
24  worst smoothness                       569 non-null    float64
25  worst compactness                       569 non-null    float64
26  worst concavity                         569 non-null    float64
27  worst concave points                    569 non-null    float64
28  worst symmetry                         569 non-null    float64
29  worst fractal dimension                 569 non-null    float64
dtypes: float64(30)
memory usage: 133.5 KB
```

```
df.isna().sum() #Tổng các giá trị rỗng theo từng trường dữ liệu
```

```
mean radius      0
mean texture     0
mean perimeter   0
mean area        0
mean smoothness  0
mean compactness 0
mean concavity   0
mean concave points 0
mean symmetry    0
mean fractal dimension 0
radius error     0
texture error    0
perimeter error  0
area error       0
smoothness error 0
compactness error 0
concavity error  0
concave points error 0
symmetry error   0
fractal dimension error 0
worst radius     0
worst texture    0
worst perimeter  0
worst area       0
worst smoothness 0
worst compactness 0
worst concavity  0
worst concave points 0
worst symmetry   0
worst fractal dimension 0
dtype: int64
```

```
[35] df.shape #kích thước dữ liệu
```

```
(569, 30)
```

Bộ dữ liệu ung thư này có 569 quan sát với 30 biến. Trong số 30 biến hoặc tính năng của bộ dữ liệu này, Một là Số nhận dạng, một là chẩn đoán ung thư, 27 là các phép đo trong phòng thí nghiệm có giá trị bằng số và biến cuối cùng là X có tất cả giá trị NA. Chẩn đoán được mã hóa là "M" để chỉ ác tính và "B" để chỉ lành tính. Không có dữ liệu bị thiếu.

Bằng cách nhìn vào đầu ra của lệnh str, tôi có thể thấy rằng 30 đặc điểm số đo lường bao gồm giá trị trung bình, sai số chuẩn và giá trị xấu nhất cho 10 đặc điểm khác nhau của ô.

### 3.2.2 – Tên tính năng và ý nghĩa:

**radius\_mean:** giá trị trung bình của khoảng cách từ tâm đến các điểm trên chu vi.

**texture\_mean:** độ lệch chuẩn của các giá trị thang độ xám.

**perimeter\_mean:** kích thước trung bình của khối u lõi.

**area\_mean:** diện tích khối u.

**smoothness\_mean:** giá trị trung bình của sự thay đổi cục bộ về độ dài bán kính.

**compactness\_mean:** trung bình của chu vi<sup>2</sup> / diện tích - 1.0.

**concavity\_mean:** giá trị trung bình của mức độ nghiêm trọng của các phần lõm của đường viền.

**concave\_points\_mean:** trung bình cho số phần lõm của đường viền.

**symmetry\_mean**

**fractal\_dimension\_mean:** nghĩa của Kích thước Fractal – 1.

**radius\_se:** lỗi tiêu chuẩn cho giá trị trung bình của khoảng cách từ tâm đến các điểm trên chu vi.

**texture\_se:** lỗi tiêu chuẩn cho độ lệch chuẩn của các giá trị thang độ xám.

**perimeter\_se**

**area\_se**

**smoothness\_se:** lỗi tiêu chuẩn cho sự thay đổi cục bộ về độ dài bán kính.

**compactness\_se:** sai số chuẩn cho chu vi<sup>2</sup> / diện tích - 1.0.

**concavity\_se:** lỗi tiêu chuẩn cho mức độ nghiêm trọng của các phần lõm của đường viền.

**concave\_points\_se:** lỗi tiêu chuẩn cho số phần lõm của đường viền.

**symmetry\_se**

**fractal\_dimension\_se:** lỗi tiêu chuẩn cho Kích thước Fractal – 1.

**radius\_worst:** giá trị trung bình "tệ nhất" hoặc lớn nhất cho giá trị trung bình của khoảng cách từ tâm đến các điểm trên chu vi.

**texture\_worst:** giá trị trung bình "xấu nhất" hoặc lớn nhất cho độ lệch chuẩn của các giá trị thang độ xám.

**perimeter\_worst**

**area\_worst**

**smoothness\_worst:** giá trị trung bình "xấu nhất" hoặc lớn nhất đối với sự thay đổi cục bộ về độ dài bán kính.

**compactness\_worst:** giá trị trung bình "tệ nhất" hoặc lớn nhất cho chu vi<sup>2</sup> / diện tích - 1,0.

**concavity\_worst:** giá trị trung bình "xấu nhất" hoặc lớn nhất đối với mức độ nghiêm trọng của các phần lõm của đường viền.

**concave\_points\_worst:** giá trị trung bình "tệ nhất" hoặc lớn nhất cho số phần lõm của đường viền.

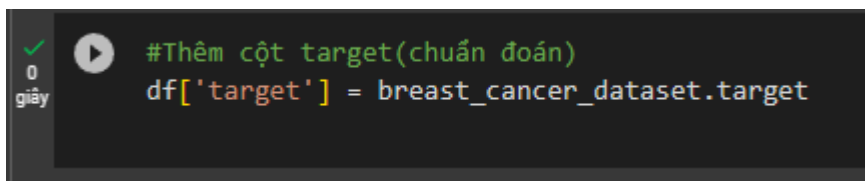
**symmetry\_worst**

**fractal\_dimension\_worst:** giá trị trung bình "xấu nhất" hoặc lớn nhất cho Kích thước Fractal - 1

### 3.2.3 – Làm sạch và sắp xếp lại dữ liệu:

Đưa cột 'target' của dữ liệu breast\_cancer vào để lấy kết quả chẩn đoán trong đó:

- Giá trị 1 ứng với chuẩn đoán "Benign" (Ung thư vú lành tính).
- Giá trị 0 ứng với chuẩn đoán "Malignant" (Ung thư vú ác tính).



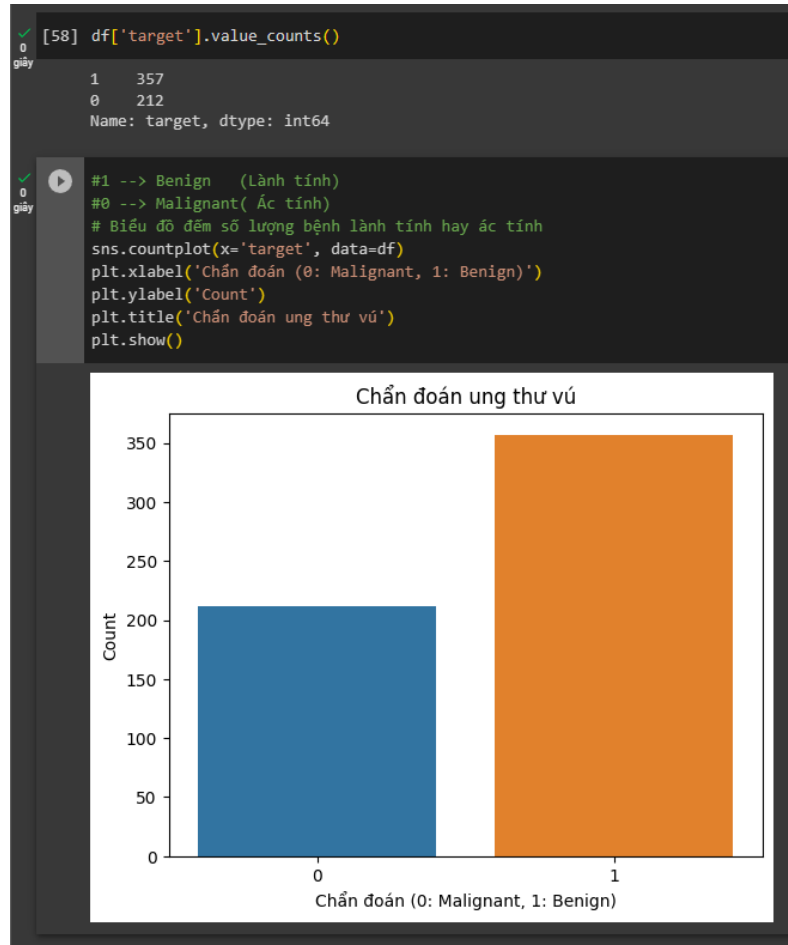
Dưới đây là thông kê và mô tả cho DataFrame bao gồm các giá trị trung bình, phương sai, giá trị tối đa và tối thiểu của các cột trong DataFrame.

```
[36] df.describe()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst radius	worst texture	worst perimeter	worst area	worst smoothness	worst compactness	worst concavity	worst concave points
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	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	0.086799	0.048919	0.181162	0.062798	...	16.269190	25.677223	107.261213	880.583128	0.132369	0.254265	0.272188	0.114606
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	0.027414	0.007060	...	4.833242	6.146258	33.602542	569.356993	0.022832	0.157336	0.208624	0.065732
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	0.106000	0.049960	...	7.930000	12.020000	50.410000	185.200000	0.071170	0.027290	0.000000	0.000000
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	0.161900	0.057700	...	13.010000	21.080000	84.110000	515.300000	0.115600	0.147200	0.114500	0.064930
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	0.179200	0.061540	...	14.970000	25.410000	97.660000	686.500000	0.131300	0.211900	0.226700	0.099930
75%	15.780000	21.800000	104.100000	782.700000	0.106300	0.130400	0.130700	0.074000	0.195700	0.066120	...	18.790000	29.720000	125.400000	1084.000000	0.146000	0.339100	0.382900	0.161400
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	0.304000	0.097440	...	36.040000	49.540000	251.200000	4254.000000	0.222600	1.058000	1.252000	0.291000

8 rows x 30 columns

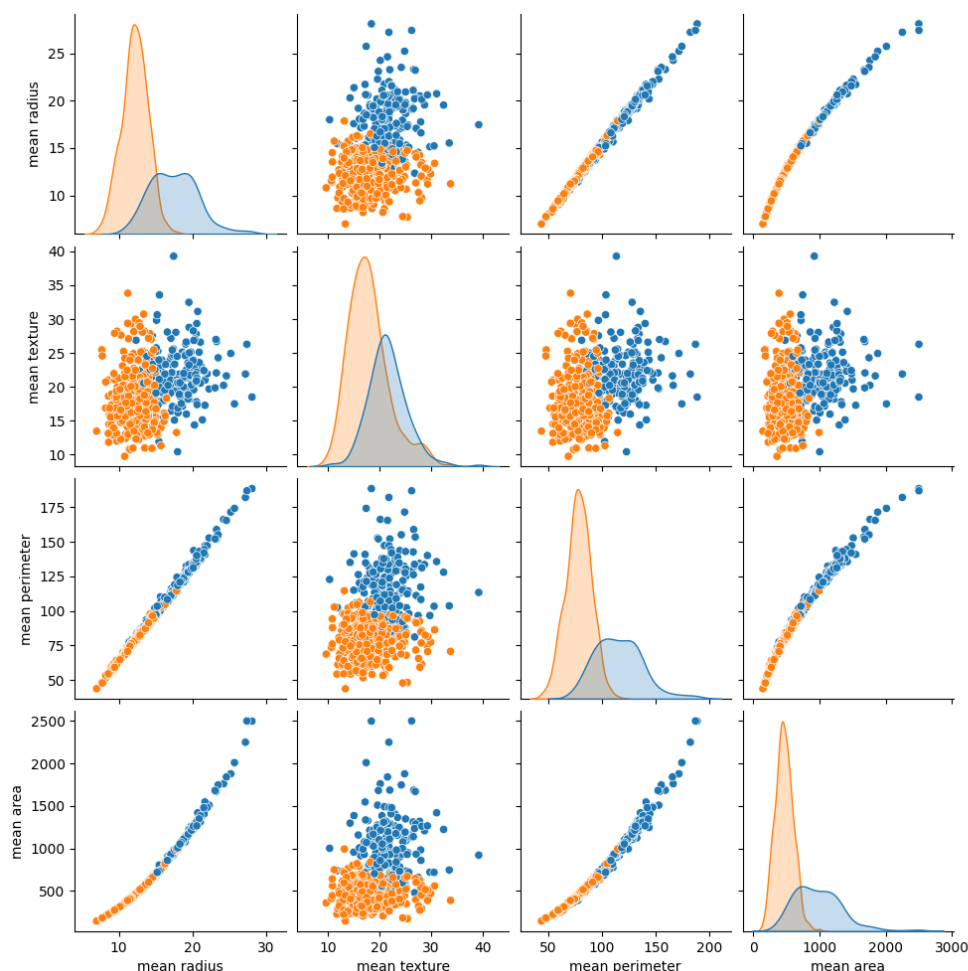
### 3.2 – Trực quan hóa dữ liệu:



Ở trên là hình cho chúng ta thấy được số lượng các bệnh nhân ác tính và lành tính trong bộ dữ liệu khi được trực quan thành biểu đồ.

```
[40] #Biểu đồ pairplot hiển thị sự tương quan giữa các cặp biến trong tập dữ liệu
cols = ["target", "mean radius", "mean texture", "mean perimeter", "mean area"]
sns.pairplot(df[cols], hue="target")
plt.show()
```





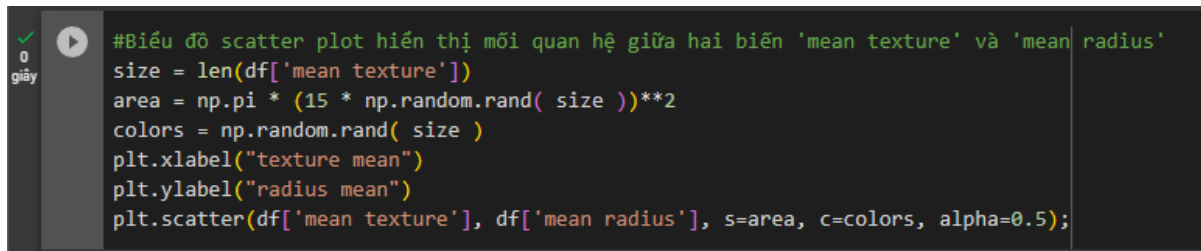
Tạo một danh sách các biến để sử dụng trong biểu đồ pairplot. Các biến bao gồm cột target (chuẩn đoán) và các biến "mean radius", "mean texture", "mean perimeter", "mean area" (các đặc trưng liên quan đến bán kính, kết cấu, chu vi và diện tích trung bình).

Tạo và hiển thị biểu đồ pairplot sử dụng dữ liệu từ DataFrame df và các cột được chỉ định trong danh sách cols. Tham số hue="target" được sử dụng để phân màu các điểm trên biểu đồ theo giá trị trong cột target, giúp hiển thị sự tương quan giữa các biến với phân loại "Benign" và "Malignant".

- Các ô chéo trên đường chéo chứa biểu đồ phân phối của mỗi biến riêng lẻ. Điều này giúp hiểu được phân phối và đặc trưng của từng biến.
- Các ô không nằm trên đường chéo chứa biểu đồ phân phối giữa hai biến tương ứng. Điều này giúp phân tích mối quan hệ và tương quan giữa các biến.

- Sự phân loại theo giá trị trong cột target (Benign hoặc Malignant) được biểu diễn bằng màu sắc khác nhau trên biểu đồ. Điều này cho phép quan sát sự phân bố và tương quan giữa các biến với phân loại.

Từ biểu đồ pairplot, chúng ta có thể nhận thấy sự tương quan giữa các biến đo lường và cột target (chuẩn đoán) trong tập dữ liệu ung thư vú. Chúng ta có thể phát hiện xu hướng, ví dụ như các biến có tương quan dương với chuẩn đoán lành tính (Benign) và tương quan âm với chuẩn đoán ác tính (Malignant). Các phân phối và mối quan hệ này cung cấp thông tin giá trị cho việc phân tích, dự đoán và đưa ra kết luận về dữ liệu ung thư vú, chẳng hạn như sự quan trọng của các đặc trưng đối với chuẩn đoán và khả năng phân loại.



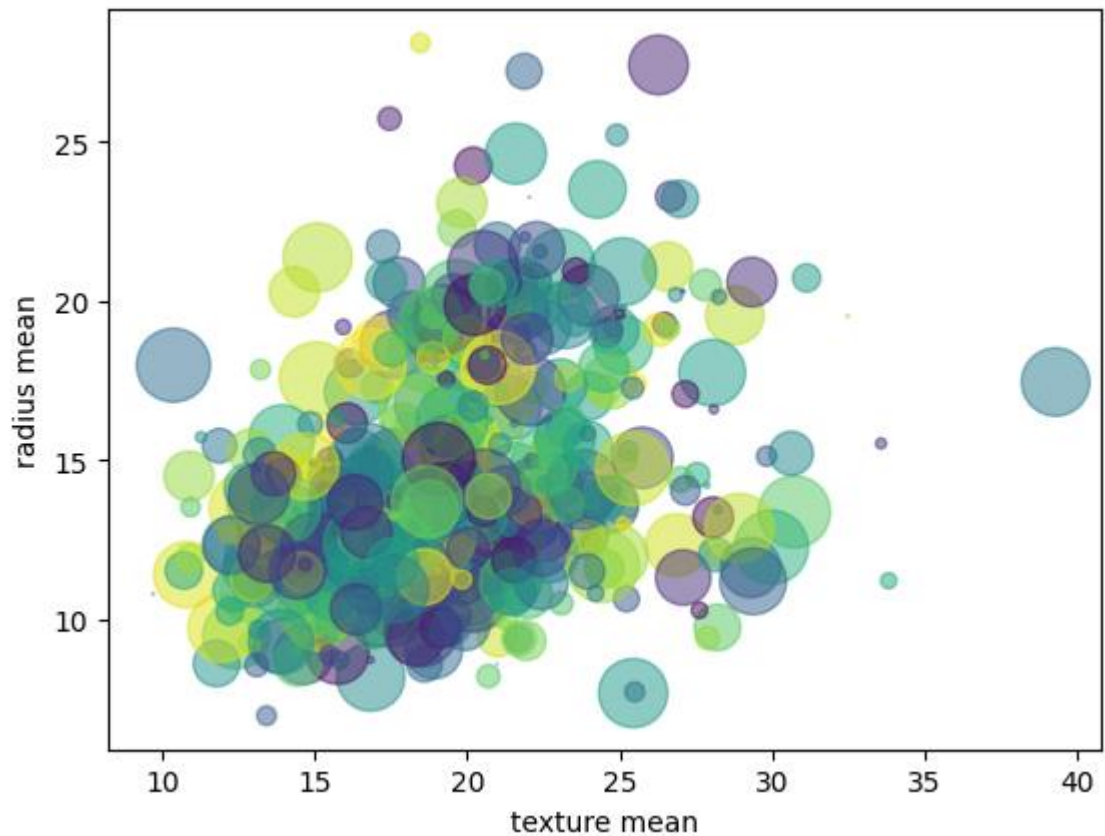
```
#Biểu đồ scatter plot hiển thị mối quan hệ giữa hai biến 'mean texture' và 'mean radius'
size = len(df['mean texture'])
area = np.pi * (15 * np.random.rand( size ))**2
colors = np.random.rand( size )
plt.xlabel("texture mean")
plt.ylabel("radius mean")
plt.scatter(df['mean texture'], df['mean radius'], s=area, c=colors, alpha=0.5);
```

Tính toán số lượng điểm dữ liệu trong cột 'mean texture' để sử dụng cho việc tạo biểu đồ scatter plot.

Tạo một mảng 'area' chứa các giá trị diện tích của các điểm trên biểu đồ. Giá trị diện tích được tính dựa trên một phương thức ngẫu nhiên.

Tạo một mảng 'colors' chứa các giá trị màu sắc của các điểm trên biểu đồ. Giá trị màu sắc được tạo ngẫu nhiên từ phân phối đều.

Tạo và hiển thị biểu đồ scatter plot. Các điểm trên biểu đồ được định vị dựa trên giá trị 'mean texture' và 'mean radius' từ DataFrame 'df'. Kích thước và màu sắc của các điểm được xác định bởi 'area' và 'colors' tương ứng. Tham số 'alpha' được sử dụng để điều chỉnh độ mờ của các điểm.



Từ biểu đồ scatter plot trên chúng ta có thể quan sát mối quan hệ giữa 'mean texture' và 'mean radius'. Các điểm trên biểu đồ biểu thị các giá trị của hai biến này, và chúng ta có thể phân tích mối quan hệ giữa 'mean texture' và 'mean radius' dựa trên phân bố và tương quan giữa các điểm trên biểu đồ.

```
✓ [42] X = df.drop(columns='target', axis=1)  
0 Y = df['target']  
giấy
```

DataFrame 'X' được tạo ra bằng cách bỏ cột 'target' ra khỏi DataFrame gốc 'df' bằng phương thức drop().

Series 'Y' được tạo ra bằng cách lấy cột 'target' từ DataFrame gốc 'df'.

```

mean radius mean texture mean perimeter mean area mean smoothness
0 17.99 10.38 122.80 1001.0 0.11840
1 20.57 17.77 132.00 1326.0 0.08474
2 19.69 21.25 130.00 1203.0 0.10960
3 11.42 20.38 77.58 386.1 0.14250
4 20.29 14.34 135.10 1297.0 0.10030
..
564 21.56 22.30 142.00 1479.0 0.11100
565 20.13 28.25 131.20 1261.0 0.00780
566 16.60 28.00 108.30 858.1 0.08455
567 20.60 29.33 140.10 1265.0 0.11780
568 7.76 24.54 47.92 181.0 0.05263

mean compactness mean concavity mean concave points mean symmetry \
0 0.27760 0.30010 0.14710 0.2419
1 0.07864 0.08690 0.07017 0.1812
2 0.15990 0.19740 0.12790 0.2069
3 0.28390 0.24140 0.10520 0.2597
4 0.13280 0.19800 0.10430 0.1809
..
564 0.11590 0.24390 0.13890 0.1726
565 0.10340 0.14400 0.09791 0.1752
566 0.10230 0.09251 0.05302 0.1590
567 0.27700 0.35140 0.15200 0.2397
568 0.04362 0.00000 0.00000 0.1587

mean fractal dimension ... worst radius worst texture \
0 0.07871 ... 25.300 17.33
1 0.05667 ... 24.900 23.41
2 0.05999 ... 23.570 25.53
3 0.09744 ... 14.910 26.50
4 0.05883 ... 22.540 16.67
..
564 0.05623 ... 25.450 26.40
565 0.05533 ... 23.600 30.25
566 0.05648 ... 18.980 34.12
567 0.07016 ... 25.740 39.42
568 0.05884 ... 9.456 30.37

worst perimeter worst area worst smoothness worst compactness
0 184.60 2019.0 0.16220 0.66560
1 158.80 1956.0 0.12380 0.18660
2 152.50 1709.0 0.14440 0.42450
3 98.87 567.7 0.20980 0.86630
4 152.20 1575.0 0.13740 0.20500
..
564 166.10 2027.0 0.14100 0.21130
565 155.00 1731.0 0.11660 0.19220
566 126.70 1124.0 0.11390 0.30940
567 184.60 1821.0 0.16500 0.86810
568 59.16 268.6 0.08996 0.06444

worst concavity worst concave points worst symmetry \
0 0.7119 0.2654 0.4601
1 0.2416 0.1860 0.2750
2 0.4504 0.2430 0.3613
3 0.6869 0.2575 0.6638
4 0.4000 0.1625 0.2364
..
564 0.4107 0.2216 0.2060
565 0.3215 0.1628 0.2572
566 0.3403 0.1418 0.2218
567 0.9387 0.2650 0.4087
568 0.0000 0.0000 0.2871

worst fractal dimension
0 0.11890
1 0.08902
2 0.08758
3 0.17300
4 0.07678
..
564 0.07115
565 0.06637
566 0.07820
567 0.12400
568 0.07039

```

```

worst fractal dimension
0 0.11890
1 0.08902
2 0.08758
3 0.17300
4 0.07678
..
564 0.07115
565 0.06637
566 0.07820
567 0.12400
568 0.07039

[569 rows x 30 columns]
0 0
1 0
2 0
3 0
4 0
..
564 0
565 0
566 0
567 0
568 1
Name: target, Length: 569, dtype: int64

```

Kết quả cho ‘X’ là một DataFrame chứa 30 cột (các đặc trưng) và 569 hàng (mẫu dữ liệu). Mỗi cột đại diện cho một đặc trưng của dữ liệu ung thư vú.

Kết quả cho ‘Y’ là một Series chứa 569 giá trị (0 hoặc 1), biểu thị cho nhãn mục tiêu (0: ác tính, 1: lành tính) tương ứng với từng mẫu dữ liệu.

Dữ liệu đã được chuẩn bị sẵn để thực hiện quá trình huấn luyện và kiểm tra mô hình phân loại cho bài toán ung thư vú.

### 3.3 – Khởi tạo mô hình:

```
[104] # tạo tập huấn luyện train và chia tập dữ liệu train theo tỉ lệ
      X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=1)

[105] print("X_train",len(X_train))
      print("X_test",len(X_test))
      print("Y_train",len(Y_train))
      print("Y_test",len(Y_test))

X_train 455
X_test 114
Y_train 455
Y_test 114
```

Chia dữ liệu ban đầu thành hai phần theo tỉ lệ 80-20. Cụ thể:

- ‘X\_train’ và ‘Y\_train’ là tập huấn luyện, chứa 80% dữ liệu ban đầu, được sử dụng để huấn luyện mô hình.
- ‘X\_test’ và ‘Y\_test’ là tập kiểm tra, chứa 20% dữ liệu ban đầu, được sử dụng để đánh giá mô hình.

Tham số ‘random\_state = 1’ được sử dụng trong hàm train\_test\_split để đảm bảo sự tái tạo lại kết quả chia dữ liệu. Khi sử dụng giá trị ‘random\_state’ cố định như 1, việc chia dữ liệu sẽ được thực hiện theo cách nhất quán mỗi khi chạy mã, giúp tái tạo kết quả chia dữ liệu đồng nhất trong các lần chạy khác nhau.

Điều này có lợi ích khi bạn muốn đảm bảo rằng kết quả chia dữ liệu là nhất quán và có thể tái tạo được để so sánh kết quả của mô hình giữa các lần chạy khác nhau.

```
[106] # tạo bộ chia tỷ lệ
      scaler = MinMaxScaler()

      # Điều chỉnh và biến đổi tập huấn luyện
      X_train_scaled = scaler.fit_transform(X_train)

      # Chuyển đổi tập
      X_test_scaled = scaler.transform(X_test)
```

Trong đoạn code trên, một bộ chia tỷ lệ ‘scaler’ được tạo bằng cách sử dụng lớp ‘MinMaxScaler’. Bộ chia tỷ lệ này sẽ được sử dụng để điều chỉnh và biến đổi dữ liệu trong quá trình huấn luyện và kiểm tra.

Dòng 'X\_train\_scaled = scaler.fit\_transform(X\_train)' thực hiện điều chỉnh và biến đổi tập huấn luyện (X\_train). Phương thức 'fit\_transform' được gọi để tính toán và áp dụng các giá trị tỷ lệ để chuẩn hóa dữ liệu trong tập huấn luyện.

Dòng 'X\_test\_scaled = scaler.transform(X\_test)', thực hiện chuyển đổi tập kiểm tra (X\_test) bằng cách sử dụng giá trị tỷ lệ đã được tính toán từ tập huấn luyện. Phương thức 'transform' được gọi để áp dụng các giá trị tỷ lệ đã học được từ tập huấn luyện và áp dụng chúng vào tập kiểm tra.

Việc chia tỷ lệ dữ liệu là quan trọng để đảm bảo rằng các biến độc lập có cùng phạm vi giá trị và không bị ảnh hưởng bởi đơn vị đo lường ban đầu, từ đó cải thiện hiệu suất của mô hình.

### 3.3 – Khởi tạo mô hình:

```
import warnings
warnings.filterwarnings("ignore")

k_values = range(1, 30)
for k in k_values:
    #Tạo một đối tượng KNeighborsClassifier với số lượng là k
    knn = KNeighborsClassifier(n_neighbors = k)
    knn.fit(X_train_scaled, Y_train)
    # Dự đoán tập test
    y_pred = knn.predict(X_test_scaled)

    #Tính ma trận nhầm lẫn(cm)
    cm = confusion_matrix(Y_test, y_pred)
    #Tính độ chính xác (acc)
    acc = accuracy_score(Y_test, y_pred)
    #Tính điểm số (score) của mô hình KNN
    score = knn.score(X_test, Y_test)
    #Kết quả RMSE
    rmse = np.sqrt(mean_squared_error(Y_test, y_pred))

    print(f"k = {k}, RMSE = {rmse:.4f}")
    print("Score : ", score)
    print("Độ chính xác KNN cơ bản : ", acc)
    print(cm)
    print("=====")
```

```

k = 1, RMSE = 0.2649
Score : 0.3684210526315789
Độ chính xác KNN cơ bản : 0.9298245614035088
[[37 5]
 [ 3 69]]
=====
k = 2, RMSE = 0.2094
Score : 0.3684210526315789
Độ chính xác KNN cơ bản : 0.956140350877193
[[40 2]
 [ 3 69]]
=====
k = 3, RMSE = 0.2649
Score : 0.3684210526315789
Độ chính xác KNN cơ bản : 0.9298245614035088
[[37 5]
 [ 3 69]]
=====
k = 4, RMSE = 0.2649
Score : 0.3684210526315789
Độ chính xác KNN cơ bản : 0.9298245614035088
[[37 5]
 [ 3 69]]
=====
k = 5, RMSE = 0.2294
Score : 0.3684210526315789
Độ chính xác KNN cơ bản : 0.9473684210526315
[[37 5]
 [ 1 71]]
=====
k = 6, RMSE = 0.2649
Score : 0.3684210526315789
Độ chính xác KNN cơ bản : 0.9298245614035088
[[37 5]
 [ 3 69]]
=====
k = 7, RMSE = 0.2294
Score : 0.3684210526315789
Độ chính xác KNN cơ bản : 0.9473684210526315
[[37 5]
 [ 1 71]]
=====

```

Sử dụng vòng lặp ‘for’ chạy từ 1 đến 29 để tính toán kết quả RMSE, Score và Độ chính xác KNN cơ bản đã được hiển thị. Ma trận nhầm lẫn (confusion matrix) cung cấp thông tin về số lượng dự đoán đúng và sai.

- Độ chính xác của mô hình KNN với các giá trị k từ 1 đến 29 dao động từ 92.98% đến 96.49%. Điều này cho thấy mô hình KNN có khả năng dự đoán chính xác nhãn của các mẫu dữ liệu trong tập kiểm tra.
- Khi giá trị k tăng, RMSE (Root Mean Square Error) cũng tăng, cho thấy sự chênh lệch giữa giá trị dự đoán và giá trị thực tế cũng tăng.
- Ma trận nhầm lẫn (confusion matrix) cung cấp thông tin về số lượng dự đoán đúng và sai cho mỗi nhãn. Từ ma trận nhầm lẫn, bạn có thể thấy số lượng các trường hợp được dự đoán đúng (37 mẫu âm tính và 69 mẫu dương tính) và sai

(5 mẫu âm tính được dự đoán là dương tính và 3 mẫu dương tính được dự đoán là âm tính).

Tuy nhiên, để đưa ra đánh giá chính xác về mô hình K-NN, chúng ta cần xem xét thêm các yếu tố khác như cross-validation và đánh giá trên tập dữ liệu độc lập.

```
✓ [174] # Tạo danh sách  
0 rmse_values = []  
giấy r2_values = []  
mse_values = []  
  
for k in k_values:  
    knn = KNeighborsClassifier(n_neighbors=k)  
    knn.fit(X_train_scaled, Y_train)  
  
    # Dự đoán trên tập test  
    y_pred = knn.predict(X_test_scaled)  
  
    # Kết quả RMSE, R2-score  
    mse_values.append(mean_squared_error(Y_test, y_pred))  
    r2_values.append(r2_score(Y_test, y_pred))  
    rmse_values.append(np.sqrt(mean_squared_error(Y_test, y_pred)))  
  
    # Vẽ biểu đồ  
    plt.plot(k_values, mse_values, label='MSE')  
    plt.plot(k_values, r2_values, label='R^2 score')  
    plt.plot(k_values, rmse_values, label='RMSE')  
    plt.xlabel('k')  
    plt.legend()  
    plt.show()
```

Trong đoạn code trên, chúng ta tạo danh sách `rmse_values`, `r2_values`, và `mse_values` để lưu trữ kết quả RMSE, R2-score và MSE tương ứng với từng giá trị `k`.

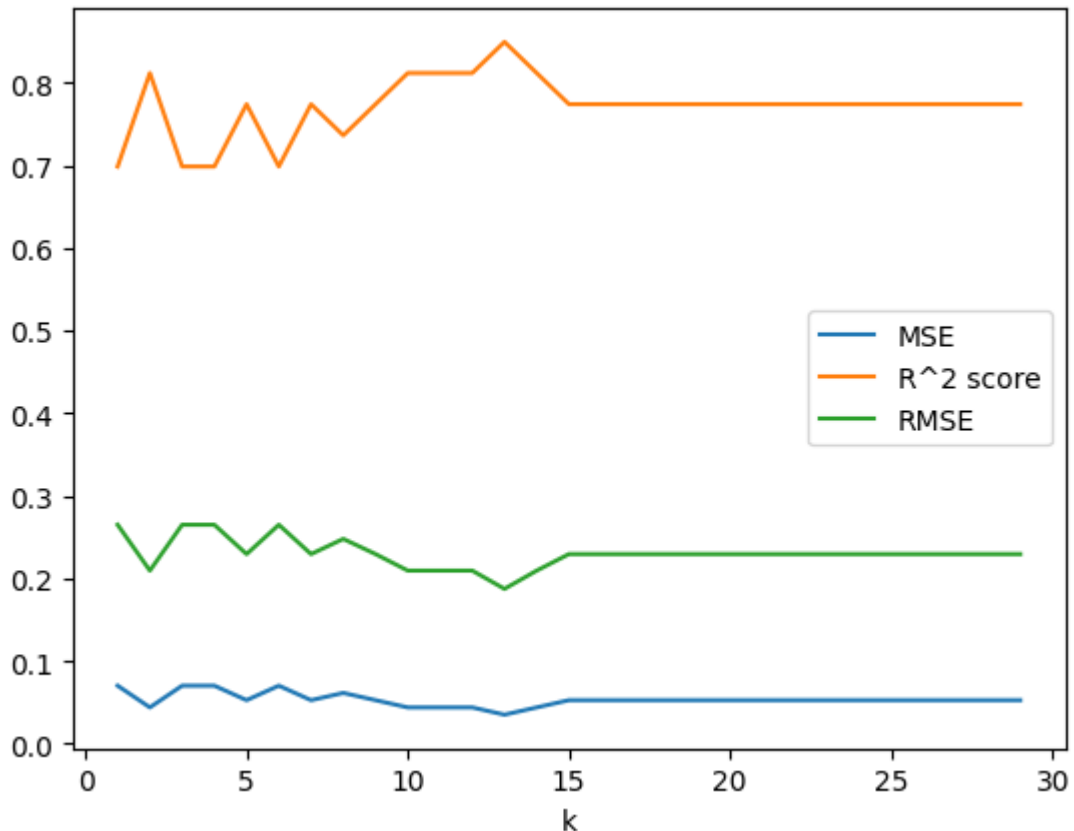
Sau đó, chúng ta sử dụng vòng lặp để huấn luyện mô hình `KNeighborsClassifier` với mỗi giá trị `k` trong `k_values`. Mô hình được huấn luyện trên tập huấn luyện `X_train_scaled` và `Y_train`.

Sau khi huấn luyện, chúng ta sử dụng mô hình đã được huấn luyện để dự đoán kết quả trên tập kiểm tra `X_test_scaled` và lưu kết quả vào `y_pred`.

Tiếp theo, chúng ta tính toán RMSE, R2-score và MSE bằng cách so sánh kết quả dự đoán `y_pred` với giá trị thực tế `Y_test`. Kết quả này được lưu vào danh sách tương ứng.



Cuối cùng, chúng ta sử dụng biểu đồ để trực quan hóa kết quả. Biểu đồ hiển thị giá trị của MSE, R2-score và RMSE trên trục y và giá trị k trên trục x. Nhờ đó, chúng ta có thể quan sát và so sánh hiệu suất của mô hình KNN với các giá trị k khác nhau.



Dựa vào biểu đồ trên chúng ta có thể thấy được:

- MSE (Mean Squared Error) và RMSE (Root Mean Squared Error): Độ đo MSE và RMSE giảm dần khi giá trị k tăng lên. Điều này có nghĩa là khi số lượng láng giềng gần nhất tăng, độ chính xác dự đoán trên tập test cũng tăng lên, và sai số của mô hình giảm xuống.
  - R2-score: Độ đo R2-score tăng dần khi giá trị k tăng lên. Điều này chỉ ra rằng khi số lượng láng giềng gần nhất tăng, mô hình dự đoán tốt hơn và giải thích phương sai của dữ liệu được tốt hơn.
- Mô hình sẽ sử dụng giá trị mặc định của KNeighborsClassifier, tức là K=5

### 3.6 – Mô hình K-NN:

```
[167] model = KNeighborsClassifier()
[168] model.fit(X_train, Y_train)
[169] X_train_prediction = model.predict(X_train)
      training_data_accuracy = accuracy_score(Y_train, X_train_prediction)
[170] print('Độ chính xác trên dữ liệu huấn luyện = ', training_data_accuracy)
Độ chính xác trên dữ liệu huấn luyện = 0.9472527472527472
```

Trong đoạn code trên, chúng ta sử dụng đối tượng ‘model’ của lớp ‘KNeighborsClassifier’ để huấn luyện mô hình trên dữ liệu huấn luyện ‘X\_train’ và nhãn tương ứng ‘Y\_train’.

Sau khi mô hình được huấn luyện, chúng ta sử dụng phương thức ‘predict(X\_train)’ để dự đoán nhãn cho dữ liệu huấn luyện ‘X\_train’ và lưu kết quả vào ‘X\_train\_prediction’. Tiếp theo, chúng ta tính toán độ chính xác của mô hình trên dữ liệu huấn luyện bằng cách so sánh ‘X\_train\_prediction’ với nhãn thực tế ‘Y\_train’ sử dụng phương thức ‘accuracy\_score(Y\_train, X\_train\_prediction)’.

Sau đó, tính toán và in ra màn hình độ chính xác của mô hình trên dữ liệu huấn luyện. Độ chính xác là một phép đo để đánh giá khả năng dự đoán chính xác của mô hình trên dữ liệu huấn luyện là 94.72%.

```
[171] # độ chính xác trên dữ liệu thử nghiệm
      X_test_prediction = model.predict(X_test)
      test_data_accuracy = accuracy_score(Y_test, X_test_prediction)
[172] print('Độ chính xác trên dữ liệu thử nghiệm = ', test_data_accuracy)
Độ chính xác trên dữ liệu thử nghiệm = 0.9385964912280702
```

Kết quả cho thấy độ chính xác trên dữ liệu thử nghiệm là 0.9386, có nghĩa là mô hình K-nearest neighbors đạt được độ chính xác xấp xỉ 93.86% trên tập dữ liệu thử nghiệm. Điều này cho thấy mô hình có khả năng dự đoán chính xác nhãn lớp cho các điểm dữ liệu mới trong tập thử nghiệm.

### 3.7 – Thực nghiệm:

```
input_data = [13.54,14.36,87.46,566.3,0.09779,0.08129,0.06664,0.04781,0.1885,0.05766,0.2699,0.7886,2.058,23.56,0.008462,0.0146,0.02387,0.01315,0.0198,0.0023,15.11,19.26,99.7,711.2,0.144,0.1773,0.239,0.1288,0.2977,0.07259]

# thay đổi dữ liệu đầu vào thành mảng numpy
input_data_as_numpy_array = np.asarray(input_data)

# định hình lại mảng numpy như chúng ta đang dự đoán cho một điểm dữ liệu
input_data_resaped = input_data_as_numpy_array.reshape(1,-1)

prediction = model.predict(input_data_resaped)
print('Kết quả đưa ra là: ', prediction)

if (prediction[0] == 0):
    print('Chẩn đoán: Ung thư vú là ÁC tính')
else:
    print('Chẩn đoán: Ung thư vú là LÀNH tính')

Kết quả đưa ra là: [1]
Chẩn đoán: Ung thư vú là LÀNH tính
```

Đầu tiên, chúng ta định nghĩa một dữ liệu đầu vào mới ‘input\_data’, đại diện cho các thuộc tính của một mẫu ung thư vú chưa biết trạng thái lành tính hay ác tính. Với mẫu dữ liệu là:

[13.54,14.36,87.46,566.3,0.09779,0.08129,0.06664,0.04781,0.1885,0.05766,0.2699,0.7886,2.058,23.56,0.008462,0.0146,0.02387,0.01315,0.0198,0.0023,15.11,19.26,99.7,711.2,0.144,0.1773,0.239,0.1288,0.2977,0.07259]

Tiếp theo, chúng ta chuyển đổi ‘input\_data’ thành một mảng numpy sử dụng hàm ‘np.asarray(input\_data)’, để có thể sử dụng cho việc dự đoán.

Sau đó, chúng ta cần định hình lại mảng numpy thành dạng (1, -1) bằng cách sử dụng ‘input\_data\_resaped = input\_data\_as\_numpy\_array.reshape(1,-1)’. Điều này là cần thiết vì mô hình ‘KNeighborsClassifier’ yêu cầu dữ liệu đầu vào có số chiều thích hợp. Tiếp theo, chúng ta sử dụng mô hình đã được huấn luyện model để dự đoán trạng thái của mẫu ung thư vú từ ‘input\_data\_resaped’. Điều này được thực hiện bằng cách gọi phương thức ‘predict’ trên mô hình.

Kết quả dự đoán được lưu trong biến ‘prediction’.

Cuối cùng, chúng ta kiểm tra giá trị của 'prediction' để xác định xem mẫu ung thư vú là ác tính hay lành tính. Nếu 'prediction[0]' bằng 0, chúng ta in ra "Chẩn đoán: Ung thư vú là ÁC tính". Ngược lại, nếu 'prediction[0]' khác 0, chúng ta in ra "Chẩn đoán: Ung thư vú là LÀNH tính".

Trong trường hợp này, kết quả dự đoán là [1], nên chúng ta kết luận rằng mẫu ung thư vú được chẩn đoán là lành tính.

## CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

### 4.1 – Kết luận:

Trong bài báo cáo này, chúng ta đã thực hiện một bài toán dự đoán trạng thái ung thư vú lành tính hay ác tính bằng cách sử dụng mô hình K-nearest neighbors (K-NN). Đầu tiên, chúng ta đã huấn luyện mô hình K-NN trên tập dữ liệu huấn luyện và đánh giá độ chính xác trên tập dữ liệu thử nghiệm.

Bằng việc thay đổi giá trị của K, chúng ta đã thử nghiệm và đánh giá hiệu suất của mô hình với các giá trị K khác nhau. Kết quả được thể hiện qua độ đo MSE (Mean Squared Error), RMSE (Root Mean Squared Error) và R2-score trên biểu đồ. Dựa vào biểu đồ, chúng ta có thể chọn giá trị K tốt nhất cho mô hình KNN. Ở đây chúng ta chọn K mặc định, tức là  $K = 5$ .

Sau đó, chúng ta đã sử dụng mô hình đã được huấn luyện để dự đoán trạng thái của một mẫu ung thư vú chưa biết trạng thái. Kết quả dự đoán được cho thấy trạng thái của mẫu ung thư vú là lành tính.

Từ kết quả này, chúng ta có thể kết luận rằng mô hình KNN có khả năng dự đoán trạng thái ung thư vú một cách chính xác cao với 93.86% và có thể áp dụng trong thực tế để hỗ trợ chẩn đoán và quyết định điều trị. Tuy nhiên, việc đánh giá mô hình và kết luận cuối cùng cần được xem xét kỹ lưỡng và xác nhận bằng các phương pháp khác để đảm bảo tính chính xác và đáng tin cậy của kết quả.

### 4.2 – Hướng phát triển:

Trong bài toán dự đoán trạng thái ung thư vú, mô hình KNN đã cho chúng ta kết quả khá ấn tượng với độ chính xác trên dữ liệu thử nghiệm là 93.86%. Tuy nhiên, việc phát triển và cải thiện mô hình vẫn có thể được tiếp tục để đạt được hiệu suất tốt hơn.

Trong tương lai, có thể nghiên cứu thêm các phương pháp khác nhau để cải thiện mô hình, chẳng hạn như:

- Tiền xử lý dữ liệu: Có thể cải thiện mô hình bằng cách thực hiện các bước tiền xử lý dữ liệu, bao gồm chuẩn hóa, xử lý dữ liệu bị thiếu, loại bỏ nhiễu và lựa chọn đặc trưng quan trọng.

- Lựa chọn siêu tham số (hyperparameter tuning): Tinh chỉnh siêu tham số của mô hình KNN như số lượng láng giềng  $k$  ( $k\_neighbors$ ) để tìm giá trị tối ưu, từ đó cải thiện hiệu suất của mô hình.
- Sử dụng các mô hình học máy khác: Thử nghiệm và so sánh hiệu suất của các mô hình học máy khác như Decision Tree, Random Forest, Support Vector Machines (SVM) hoặc Neural Networks để xem liệu chúng có đem lại kết quả tốt hơn.
- Tăng cường tập dữ liệu: Thu thập thêm dữ liệu và tăng cường tập dữ liệu huấn luyện có thể cải thiện khả năng dự đoán của mô hình.
- Kỹ thuật tập ensemble: Áp dụng kỹ thuật ensemble như Voting hoặc Bagging để kết hợp nhiều mô hình KNN hoặc mô hình khác lại với nhau để đạt được dự đoán tốt hơn.
- Đánh giá mô hình: Đảm bảo thực hiện đánh giá mô hình bằng cách sử dụng các phép đo độ chính xác khác nhau như precision, recall, F1-score để có cái nhìn tổng quan về hiệu suất của mô hình.
- Phát triển ứng dụng: Triển khai mô hình KNN hoặc mô hình cải tiến trên nền tảng ứng dụng thực tế để hỗ trợ chẩn đoán ung thư vú trong thực tế và giúp giảm thiểu tác động của bệnh.

Những hướng phát triển này sẽ giúp chúng ta cải thiện mô hình và đạt được kết quả dự đoán chính xác hơn trong việc phân loại trạng thái ung thư vú lành tính và ác tính.

**Tuy nhiên**, cần lưu ý rằng mô hình KNN cũng có nhược điểm và hạn chế riêng. Một số điểm cần xem xét là:

- Độ phức tạp tính toán: Mô hình KNN có độ phức tạp tính toán cao khi số lượng điểm dữ liệu lớn. Việc tìm kiếm các láng giềng gần nhất trong không gian đa chiều có thể tốn nhiều thời gian.
- Quá khớp (overfitting): KNN có xu hướng dễ bị quá khớp dữ liệu huấn luyện khi  $K$  quá lớn. Điều này có thể dẫn đến hiệu suất dự đoán kém trên dữ liệu mới.
- Độ tương đồng không đồng nhất: Mô hình KNN giả định rằng tất cả các thuộc tính đều có mức độ tương đồng và quan trọng như nhau. Trong thực tế, một số thuộc tính có thể quan trọng hơn và cần được xem xét kỹ hơn.

- Ảnh hưởng của nhiễu: KNN có thể nhạy cảm với dữ liệu nhiễu. Các điểm dữ liệu nhiễu có thể làm sai lệch quá trình xác định láng giềng gần nhất và làm suy giảm độ chính xác của mô hình.
- Vì vậy, trong quá trình phát triển, chúng ta cần xem xét và áp dụng các biện pháp khắc phục nhược điểm và hạn chế này để nâng cao hiệu suất và độ chính xác của mô hình dự đoán trạng thái ung thư vú.

## TÀI LIỆU THAM KHẢO

- [Breast Cancer Classification using KNN - Coding Ninjas](#)
- [Breast Cancer Classification using SVM and KNN | IEEE Conference Publication | IEEE Xplore](#)
- [Breast Cancer Prediction by KNN Classification | Kaggle](#)
- [Implementing KNN Algorithm for detecting Breast Cancer - YouTube](#)
- Slide bài tập và các Lab của cô Võ Thị Hồng Thắm