**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT HƯNG YÊN**

****

**ĐỒ ÁN 2**

**ỨNG DỤNG HỌC MÁY TRONG**

**DỰ ĐOÁN BỆNH UNG THƯ PHỔI**

NGÀNH: KHOA HỌC MÁY TÍNH

SINH VIÊN: **DƯƠNG VIỆT HÙNG**

LỚP: **124221**

NGƯỜI HƯỚNG DẪN: **PGS. TS. NGUYỄN MINH TIẾN**

**HƯNG YÊN – 2025**

**NHẬN XÉT**

**Nhận xét của giáo viên hướng dẫn**

**GIÁO VIÊN HƯỚNG DẪN**

**Nguyễn Minh Tiến**

**LỜI CAM ĐOAN**

Em xin cam đoan đồ án “Ứng dụng học máy trong dự đoán bệnh ung thư phổi” là kết quả thực hiện của bản thân em dưới sự hướng dẫn của thầy PSG. TS. Nguyễn Minh Tiến.

Những phần sử dụng tài liệu tham khảo trong đồ án đã được nêu rõ trong phần tài liệu tham khảo. Các kết quả trình bày trong đồ án và chương trình xây dựng được hoàn toàn là kết quả do bản thân em thực hiện.

Nếu vi phạm lời cam đoan này, em xin chịu hoàn toàn trách nhiệm trước khoa và nhà trường.

*Hưng Yên, ngày 30 tháng 05 năm 2025*

Sinh viên

Dương Việt Hùng

**LỜI CẢM ƠN**

Để có thể hoàn thành bài tập lớn này, lời đầu tiên em xin phép gửi lời cảm ơn tới bộ môn Khoa học máy tính, Khoa Công nghệ thông tin – Trường Đại học Sư phạm Kỹ thuật Hưng Yên đã tạo điều kiện thuận lợi cho em thực hiện bài tập lớn môn học này.

Đặc biệt em xin chân thành cảm ơn thầy Nguyễn Minh Tiến đã rất tận tình hướng dẫn, chỉ bảo em trong suốt thời gian thực hiện bài tập lớn vừa qua.

Em cũng xin chân thành cảm ơn tất cả các Thầy, các Cô trong Trường đã tận tình giảng dạy, trang bị cho em những kiến thức cần thiết, quý báu để giúp em thực hiện được bài tập lớn này.

Mặc dù em đã có cố gắng hết sức, nhưng với trình độ còn hạn chế, trong quá trình thực hiện đề tài không tránh khỏi những thiếu sót. Em hi vọng sẽ nhận được những ý kiến nhận xét, góp ý của các Thầy giáo, cô giáo về những kết quả triển khai trong bài tập lớn.

Em xin trân trọng cảm ơn!

**MỤC LỤC**

[CHƯƠNG 1: GIỚI THIỆU BÀI TOÁN 9](#_Toc3409)

[1.1 Bài toán 9](#_Toc25007)

[1.2 Trình bày dữ liệu bài toán 9](#_Toc17106)

[1.3 Tiền xử lý dữ liệu 12](#_Toc22504)

[1.4 Trực quan hoá dữ liệu 12](#_Toc13721)

[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 13](#_Toc12890)

[2.1 Pandas 13](#_Toc29830)

[2.2 Matplotlib 15](#_Toc22245)

[2.3 Sklearn 17](#_Toc32398)

[2.4 Synthetic Minority Over-sampling Technique (SMOTE) 19](#_Toc32500)

[2.5 Random Oversampling Technique 20](#_Toc11410)

[2.6 Data Scaling 21](#_Toc6267)

[2.7 HyperParameter Tuning 22](#_Toc13026)

[2.8 Cross Validation 22](#_Toc21493)

[2.9 Machine Learning 24](#_Toc16341)

[2.9.1 Decision Tree 24](#_Toc28439)

[2.9.2 Naive Bayes 25](#_Toc10270)

[2.9.3 XGBoost 26](#_Toc2644)

[2.9.4 Random Forest 27](#_Toc22210)

[2.9.5 KNN 28](#_Toc1906)

[2.10 Confusion Matrix 29](#_Toc9702)

[2.11 Evaluated Metrics 30](#_Toc8627)

[CHƯƠNG 3: GIẢI PHÁP 32](#_Toc11969)

[3.1. Mã nguồn tiền xử lý dữ liệu 32](#_Toc13413)

[3.2. Mã nguồn trực quan hóa dữ liệu 38](#_Toc21118)

[3.3. Mã nguồn Machine Learning Models 44](#_Toc2775)

[KẾT LUẬN 55](#_Toc2770)

[TÀI LIỆU THAM KHẢO 56](#_Toc11518)

**DANH MỤC CÁC HÌNH VẼ**

[Hình 1.1 Tổng quan về bộ dữ liệu Air Quality 11](#_Toc18573)

[Hình 1.2 Thông tin cơ bản về kiểu dữ liệu của từng đặc trưng 11](#_Toc30298)

[Hình 2.1: Synthetic Minority Oversampling Techique 20](#_Toc6004)

[Hình 2.2: Confusion matrix 30](#_Toc28883)

[Hình 3.1: Mã nguồn kiểm tra dữ liệu 33](#_Toc14881)

[Hình 3.2: Mã nguồn encode data 34](#_Toc31652)

[Hình 3.3: Dữ liệu sau khi được mã hóa 34](#_Toc17861)

[Hình 3.4: Mã nguồn thống kê dữ liệu khuyết thiếu 35](#_Toc18070)

[Hình 3.5: Mã nguồn xử lý dữ liệu khuyết thiếu 35](#_Toc17652)

[Hình 3.6: Dữ liệu khuyết thiếu đã được xử lý 36](#_Toc11287)

[Hình 3.7: Mã nguồn xử lý các bản ghi trùng lặp 37](#_Toc25066)

[Hình 3.8: Mã nguồn rescale dữ liệu 37](#_Toc2282)

[Hình 3.9: Mã nguồn cân bằng dữ liệu 38](#_Toc23534)

[Hình 3.10: Mã nguồn phân tách dữ liệu 38](#_Toc26250)

[Hình 3.11: Phân phối của nhãn dữ liệu 39](#_Toc26801)

[Hình 3.12: Phân phối PM2.5 40](#_Toc13685)

[Hình 3.13: Phân phối của mật độ dân số 41](#_Toc27114)

[Hình 3.14: Phân tích PM10 theo chất lượng không khí 42](#_Toc32505)

[Hình 3.15: Biểu đồ phân tán của PM2.5 và PM10 43](#_Toc24693)

[Hình 3.16: Biểu đồ hộp các đặc trưng 44](#_Toc29239)

[Hình 3.17: Mã nguồn Import thư viện 45](#_Toc4541)

[Hình 3.18: Mã nguồn hàm Model\_ML 45](#_Toc3706)

[Hình 3.19: Mã nguồn hàm gridsearch\_params 46](#_Toc14659)

[Hình 3.20: Mã nguồn hàm gridsearch\_params 46](#_Toc32026)

[Hình 3.21: Mã nguồn Decision Tree model 47](#_Toc12148)

[Hình 3.22: Hiệu suất Decision Tree với Random Oversampler 47](#_Toc32627)

[Hình 3.23: Hiệu suất Decision Tree với Smote 48](#_Toc30778)

[Hình 3.24: Mã nguồn Random Forest model 48](#_Toc14794)

[Hình 3.25: Hiệu suất Random Forest với Random Oversampler 49](#_Toc32478)

[Hình 3.26: Hiệu suất Random Forest với Smote 50](#_Toc10763)

[Hình 3.27: Mã nguồn KNN model 50](#_Toc12856)

[Hình 3.28: Hiệu suất KNN với Random Oversampler 51](#_Toc20934)

[Hình 3.29: Hiệu suất KNN với Smote 52](#_Toc2891)

[Hình 3.30: Mã nguồn Naive Bayes model 52](#_Toc5342)

[Hình 3.31: Hiệu suất Naive Bayes với Random Oversampler 53](#_Toc6579)

[Hình 3.32: Hiệu suất Naive Bayes với Smote 54](#_Toc6840)

[Hình 3.33: Mã nguồn XGBoost model 54](#_Toc19602)

[Hình 3.34: Hiệu suất XGBoost với Random Oversampler 55](#_Toc11765)

[Hình 3.35: Hiệu suất XGBoost với Smote 55](#_Toc9188)

# GIỚI THIỆU BÀI TOÁN

## Bài toán

Chẩn đoán và dự đoán bệnh ung thư phổi là một trong những ứng dụng nổi bật và đầy tiềm năng của học máy (machine learning) trong lĩnh vực y học hiện đại. Ung thư phổi hiện là một trong những nguyên nhân gây tử vong hàng đầu trên toàn cầu, đặc biệt tại các quốc gia đang phát triển, nơi mà điều kiện tầm soát và chẩn đoán còn hạn chế. Tại Việt Nam, số ca mắc mới và tử vong do ung thư phổi vẫn ở mức cao, phần lớn do bệnh được phát hiện ở giai đoạn muộn. Việc ứng dụng học máy mang lại khả năng phân tích và xử lý dữ liệu để hỗ trợ phát hiện sớm và phân loại nguy cơ mắc bệnh với độ chính xác cao. Các thuật toán tiên tiến như rừng ngẫu nhiên (Random Forest), XGBoost đã chứng minh khả năng vượt trội trong việc nhận diện tổn thương phổi, phân biệt khối u lành tính và ác tính, cũng như dự đoán tiến triển của bệnh theo thời gian.

Ngoài ra, học máy còn hỗ trợ cá nhân hóa phương pháp điều trị thông qua phân tích đặc điểm di truyền của từng bệnh nhân (genomics), từ đó tối ưu hóa phác đồ hóa trị, xạ trị hoặc liệu pháp miễn dịch. Tuy nhiên, để các mô hình này phát huy hiệu quả trong thực tế lâm sàng, cần đảm bảo nguồn dữ liệu chất lượng cao, đa dạng và được chú thích chính xác, đồng thời đáp ứng các yêu cầu đạo đức và bảo mật thông tin y tế. Tại Việt Nam, những thách thức về hạ tầng công nghệ, nguồn nhân lực chuyên môn và hành lang pháp lý vẫn đang là rào cản lớn. Do đó, việc phát triển và triển khai hệ thống dự đoán ung thư phổi thông minh cần có sự phối hợp liên ngành giữa các cơ sở y tế, viện nghiên cứu, trường đại học và doanh nghiệp công nghệ nhằm xây dựng nền tảng dữ liệu y tế số hóa, thúc đẩy nghiên cứu ứng dụng AI trong y học, góp phần nâng cao chất lượng khám chữa bệnh và giảm tỷ lệ tử vong do ung thư phổi trong tương lai.

## Trình bày dữ liệu bài toán

Link dữ liệu trên Kaggle:

Lung Cancer Dataset [| Kaggle](https://www.kaggle.com/datasets/rabieelkharoua/alzheimers-disease-dataset/data)

A screenshot of a computer

AI-generated content may be incorrect.

Hình 1.1 Tổng quan về bộ dữ liệu Lung Cancer

A screenshot of a computer program

AI-generated content may be incorrect.

Hình 1.2 Thông tin cơ bản về kiểu dữ liệu của từng đặc trưng

* Dữ liệu bài toán gồm các feature sau:

- **Temperature (°C)**: Average temperature of the region.

- **Humidity (%)**: Relative humidity recorded in the region.

- **PM2.5 Concentration (µg/m³)**: Fine particulate matter levels.

- **PM10 Concentration (µg/m³)**: Coarse particulate matter levels.

- **NO2 Concentration (ppb)**: Nitrogen dioxide levels.

- **SO2 Concentration (ppb)**: Sulfur dioxide levels.

- **CO Concentration (ppm)**: Carbon monoxide levels.

- **Proximity to Industrial Areas (km)**: Distance to the nearest industrial zone.

- **Population Density (people/km²)**: Number of people per square kilometer in the region.

- **Air Quality**: Level of air pollution

* Dữ liệu bài toán là 1 file csv gồm 5000 rows × 10 columns

Tương ứng với có 10 features và mỗi feature có 5000 dữ liệu đầu vào.

* Sau khi mô tả dữ liệu ta có:

## Tiền xử lý dữ liệu

1. Mã hóa các cột chứa dữ liệu văn bản
2. Xử lý dữ liệu mất mát
3. Xử lý dữ liệu trùng lặp
4. Xử lý chuẩn hóa dữ liệu
5. Xử lý mất cân bằng nhãn

## Trực quan hoá dữ liệu

1. Biểu đồ phân phối nhãn dữ liệu
2. Biểu đồ cột phân phối độ tuổi
3. Biểu đồ tỉ lệ trình độ học vấn
4. Biểu đồ phân phối MMSE
5. Biểu đồ phân phối BMI
6. Biểu đồ phân phối người bênh hút thuốc
7. Biểu đồ tương quan giữa BMI và MMSE

# CƠ SỞ LÝ THUYẾT

## Pandas

Pandas là một thư viện mã nguồn mở mạnh mẽ được sử dụng rộng rãi trong Python để thao tác và phân tích dữ liệu, đặc biệt trong lĩnh vực khoa học dữ liệu và học máy. Tên "Pandas" xuất phát từ cụm từ "Panel Data," một thuật ngữ trong kinh tế lượng, nhấn mạnh khả năng xử lý dữ liệu đa chiều. Pandas cung cấp hai cấu trúc dữ liệu chính: \*\*DataFrame\*\* (một bảng dữ liệu hai chiều với nhãn dòng và cột, tương tự như bảng trong Excel hoặc cơ sở dữ liệu) và \*\*Series\*\* (một mảng một chiều, giống như một cột trong DataFrame). Các tính năng chính của Pandas bao gồm: xử lý dữ liệu từ nhiều nguồn khác nhau như CSV, Excel, SQL, và JSON; cung cấp các phương thức mạnh mẽ để sắp xếp, lọc, nhóm, nối và tóm tắt dữ liệu; hỗ trợ thao tác dữ liệu chuỗi thời gian và xử lý dữ liệu thiếu (missing data). Ngoài ra, Pandas tích hợp tốt với các thư viện Python khác như NumPy, Matplotlib, và Scikit-learn, làm cho nó trở thành công cụ linh hoạt trong việc tiền xử lý dữ liệu và phân tích.

Việc sử dụng Pandas mang lại nhiều lợi ích rõ rệt. Trước hết, Pandas có một cú pháp đơn giản, dễ học, và tài liệu phong phú, giúp ngay cả những người mới bắt đầu cũng có thể nhanh chóng làm quen. Thứ hai, các thao tác của Pandas được tối ưu hóa để xử lý tập dữ liệu lớn, nhờ được xây dựng dựa trên NumPy, một thư viện xử lý mảng hiệu suất cao. Thứ ba, Pandas cung cấp các công cụ trực quan hóa dữ liệu cơ bản và tích hợp chặt chẽ với các thư viện vẽ biểu đồ khác, giúp người dùng dễ dàng khám phá và trình bày dữ liệu. Cuối cùng, khả năng quản lý và thao tác dữ liệu dễ dàng, từ việc chuẩn hóa dữ liệu đầu vào đến xử lý dữ liệu phức tạp, khiến Pandas trở thành lựa chọn lý tưởng cho các nhà khoa học dữ liệu, nhà phân tích, và kỹ sư học máy. Trong thực tế, Pandas không chỉ giúp tăng năng suất làm việc mà còn giảm nguy cơ sai sót trong xử lý dữ liệu, điều này rất quan trọng trong các dự án yêu cầu độ chính xác cao.

Việc sử dụng Pandas để xử lý bộ dữ liệu về bệnh Alzheimer có nhiều lợi ích rõ rệt nhờ các tính năng mạnh mẽ và linh hoạt của thư viện này, đặc biệt khi làm việc với dữ liệu y tế phức tạp. Dưới đây là các lý do chính:

1. Quản lý dữ liệu dễ dàng:

Bộ dữ liệu chứa nhiều đặc trưng (features) và giá trị đa dạng, từ các thông số số liệu (tuổi, huyết áp, cholesterol) đến thông tin danh mục (giới tính, dân tộc) và các giá trị nhị phân (hút thuốc, tiền sử gia đình). Pandas cung cấp cấu trúc DataFrame, cho phép tổ chức dữ liệu một cách rõ ràng, dễ dàng truy xuất, và thao tác với các cột cụ thể.

1. Xử lý dữ liệu thiếu (Missing Data):

Trong dữ liệu y tế, các giá trị thiếu (missing values) rất phổ biến. Pandas cung cấp các công cụ để phát hiện, thay thế hoặc loại bỏ dữ liệu thiếu thông qua các hàm như isnull() hoặc fillna(), giúp đảm bảo dữ liệu sạch và nhất quán.

1. Chuyển đổi dữ liệu:

Pandas hỗ trợ chuyển đổi dữ liệu hiệu quả, như chuẩn hóa các cột số liệu (tuổi, BMI) hoặc mã hóa dữ liệu danh mục (giới tính, dân tộc) thành dạng mà các thuật toán học máy có thể sử dụng, nhờ các phương thức như apply(), map(), và get\_dummies().

1. Khả năng phân tích dữ liệu ban đầu( EDA):

Với Pandas, bạn có thể dễ dàng tính toán các thống kê cơ bản (trung bình, độ lệch chuẩn, tần suất xuất hiện) để hiểu sâu hơn về từng đặc trưng. Ví dụ, describe() cung cấp tổng quan về dữ liệu số, giúp nhanh chóng xác định các giá trị ngoại lệ (outliers) hoặc xu hướng bất thường.

1. Lọc và nhóm dữ liệu (Filtering and Grouping):

Pandas hỗ trợ thao tác với các nhóm dữ liệu dựa trên điều kiện cụ thể. Ví dụ, bạn có thể dễ dàng nhóm bệnh nhân theo giới tính, độ tuổi, hoặc chẩn đoán để so sánh các thông số khác nhau bằng cách sử dụng groupby() và filter().

1. Tích hợp tốt với học máy:

Bộ dữ liệu này là tiền đề cho các mô hình học máy. Pandas hỗ trợ chuẩn bị dữ liệu đầu vào cho các thư viện học máy như Scikit-learn hoặc TensorFlow, từ việc chia nhỏ dữ liệu (train-test split) đến xuất dữ liệu dưới dạng NumPy array.

1. Hiệu quả trong thao tác dữ liệu lớn:

Pandas được tối ưu hóa để xử lý lượng lớn dữ liệu một cách nhanh chóng và hiệu quả. Bộ dữ liệu về Alzheimer có thể có nhiều dòng (bệnh nhân) và cột (đặc trưng), nhưng Pandas giúp quản lý và thao tác dễ dàng nhờ hiệu suất cao.

1. Tích hợp với trực quan hóa:

Với Pandas, bạn có thể dễ dàng tạo các biểu đồ như histogram, boxplot, hoặc scatter plot để phân tích mối quan hệ giữa các đặc trưng (ví dụ: mối quan hệ giữa tuổi và nguy cơ Alzheimer).

## Matplotlib

1. Trực quan hóa dữ liệu dễ dàng:

Matplotlib cung cấp các công cụ mạnh mẽ để tạo ra nhiều loại biểu đồ như histogram, scatter plot, line plot, bar chart, box plot, phù hợp với dữ liệu đa dạng trong bộ dữ liệu Alzheimer. Ví dụ: biểu đồ scatter có thể giúp minh họa mối quan hệ giữa tuổi và điểm MMSE, trong khi biểu đồ histogram giúp phân phối các đặc trưng như chỉ số BMI hoặc cholesterol.

1. Tùy chỉnh mạnh mẽ:

Matplotlib cho phép tùy chỉnh toàn diện các yếu tố của biểu đồ, bao gồm màu sắc, kiểu đường, nhãn, tiêu đề, kích thước, và chú giải. Điều này rất hữu ích để tạo ra các biểu đồ chuyên nghiệp hoặc nhấn mạnh vào các yếu tố quan trọng, chẳng hạn như hiển thị tỷ lệ bệnh nhân với huyết áp cao theo từng nhóm tuổi.

1. Khám phá mối quan hệ giữa các đặc trưng:

Matplotlib hỗ trợ minh họa mối quan hệ giữa các đặc trưng qua biểu đồ hai chiều hoặc ba chiều. Chẳng hạn, bạn có thể sử dụng scatter plot để kiểm tra sự tương quan giữa cholesterol LDL và huyết áp tâm thu trong việc dự đoán nguy cơ Alzheimer, hoặc biểu đồ heatmap để hiển thị mức độ liên quan giữa nhiều đặc trưng.

1. Hỗ trợ phân tích so sánh:

Với Matplotlib, bạn có thể dễ dàng so sánh các nhóm dữ liệu khác nhau. Ví dụ: sử dụng bar chart để so sánh tỷ lệ mắc bệnh Alzheimer giữa nam và nữ, hoặc box plot để so sánh giá trị cholesterol giữa các nhóm bệnh nhân có hoặc không có tiền sử gia đình.

1. Xử lý dữ liệu thời gian:

Bộ dữ liệu Alzheimer thường có các đặc trưng liên quan đến thời gian, chẳng hạn như diễn biến triệu chứng hoặc kết quả đánh giá theo thời gian. Matplotlib hỗ trợ vẽ biểu đồ line để minh họa xu hướng thay đổi của điểm MMSE hoặc chức năng ADL qua các năm.

1. Tích hợp tốt với Pandas:

Matplotlib hoạt động mượt mà với các DataFrame của Pandas. Bạn có thể dễ dàng truyền dữ liệu trực tiếp từ Pandas vào Matplotlib để tạo biểu đồ, ví dụ như biểu đồ phân phối tuổi hoặc biểu đồ scatter giữa huyết áp và chỉ số BMI.

1. Phân phối dữ liệu và giá trị ngoại lệ:

Matplotlib hỗ trợ trực quan hóa phân phối dữ liệu và phát hiện các giá trị ngoại lệ thông qua các biểu đồ như box plot hoặc violin plot, giúp bạn nhanh chóng nhận ra bất kỳ sự bất thường nào trong dữ liệu, chẳng hạn như huyết áp cao bất thường hoặc BMI rất thấp.

1. Tạo báo cáo và trình bày dữ liệu:

Với Matplotlib, bạn có thể tạo các biểu đồ có chất lượng cao phù hợp cho việc trình bày báo cáo, nghiên cứu khoa học, hoặc chia sẻ trong nhóm. Ví dụ, biểu đồ cột có thể được sử dụng để minh họa tỷ lệ phần trăm bệnh nhân có các triệu chứng như mất định hướng hoặc thay đổi tính cách.

1. Hiệu quả và linh hoạt:

Matplotlib rất linh hoạt khi làm việc với các tập dữ liệu lớn, cho phép bạn chia nhỏ các tập con hoặc tổng hợp dữ liệu thành các biểu đồ trực quan. Điều này rất quan trọng khi cần so sánh nhiều nhóm bệnh nhân, chẳng hạn giữa các nhóm tuổi hoặc các mức độ nghiêm trọng của bệnh.

1. Hỗ trợ biểu đồ nâng cao với Seaborn:

Mặc dù Matplotlib là công cụ cơ bản, nó tích hợp rất tốt với Seaborn (một thư viện cao cấp hơn) để tạo ra các biểu đồ thống kê đẹp mắt và dễ dàng minh họa các mối quan hệ phức tạp, như biểu đồ heatmap giữa các đặc trưng hoặc pair plot để kiểm tra tương quan giữa các cột số liệu.

## Sklearn

Scikit-learn (sklearn) cung cấp một loạt các công cụ mạnh mẽ giúp xử lý và tiền xử lý dữ liệu một cách hiệu quả. Trong bộ dữ liệu về bệnh Alzheimer, bạn có thể sử dụng các công cụ như `StandardScaler` và `MinMaxScaler` để chuẩn hóa và điều chỉnh các đặc trưng đầu vào như huyết áp, cholesterol, BMI. Việc chuẩn hóa này giúp mô hình học máy có thể xử lý dữ liệu hiệu quả hơn, đặc biệt là khi các đặc trưng có phạm vi hoặc đơn vị khác nhau.

Scikit-learn cũng cung cấp một hàm hữu ích gọi là `train\_test\_split`, cho phép bạn chia bộ dữ liệu thành các tập huấn luyện và kiểm tra một cách dễ dàng. Điều này rất quan trọng trong việc kiểm tra độ chính xác của mô hình sau khi huấn luyện, giúp đảm bảo rằng mô hình có thể tổng quát hóa tốt trên dữ liệu chưa thấy và tránh tình trạng overfitting.

Với Scikit-learn, bạn có thể dễ dàng chọn các mô hình học máy phù hợp. Thư viện này hỗ trợ nhiều thuật toán học giám sát như hồi quy tuyến tính, cây quyết định, SVM (Support Vector Machine), hay các mô hình học không giám sát như k-means. Ví dụ, bạn có thể sử dụng hồi quy logistic để dự đoán nguy cơ mắc bệnh Alzheimer dựa trên các đặc trưng như tuổi tác, huyết áp, cholesterol, hoặc SVM để phân loại bệnh nhân thành các nhóm nguy cơ cao và thấp.

Khi đã lựa chọn được mô hình, Scikit-learn cung cấp nhiều công cụ để đánh giá hiệu suất của mô hình như kỹ thuật cross-validation và grid search. Những công cụ này giúp bạn tối ưu hóa các siêu tham số của mô hình, từ đó nâng cao hiệu quả dự đoán. Bạn có thể sử dụng `cross\_val\_score` để đánh giá mô hình qua nhiều lần phân chia dữ liệu khác nhau, hoặc dùng `GridSearchCV` để tìm bộ tham số tối ưu cho mô hình của mình.

Một vấn đề thường gặp trong các bộ dữ liệu y tế là giá trị thiếu. Scikit-learn cung cấp các công cụ như SimpleImputer để thay thế các giá trị thiếu bằng các giá trị trung bình, trung vị hoặc mode của cột, giúp duy trì tính toàn vẹn của bộ dữ liệu và không làm gián đoạn quá trình huấn luyện mô hình.

Về mặt kỹ thuật biến đổi đặc trưng, Scikit-learn hỗ trợ các công cụ như OneHotEncoding và PolynomialFeatures, giúp cải thiện khả năng học của mô hình. Ví dụ, các đặc trưng phân loại như giới tính hay dân tộc có thể được mã hóa dưới dạng các cột nhị phân bằng `OneHotEncoder`, giúp mô hình học máy có thể xử lý chúng một cách hiệu quả.

Một điểm mạnh của Scikit-learn là khả năng cung cấp các mô hình học máy có thể giải thích được. Các mô hình như cây quyết định hay hồi quy tuyến tính có thể đưa ra các quyết định rõ ràng về cách các đặc trưng ảnh hưởng đến kết quả, điều này rất quan trọng trong các bài toán y tế như Alzheimer, nơi việc giải thích các quyết định của mô hình có thể giúp bác sĩ và chuyên gia y tế hiểu được các yếu tố ảnh hưởng đến nguy cơ bệnh.

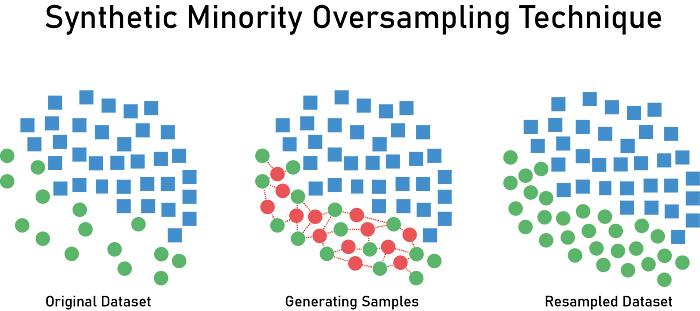
Ngoài ra, Scikit-learn có thể dễ dàng tích hợp với các thư viện khác như Pandas và Matplotlib, giúp người dùng dễ dàng thao tác với dữ liệu và trực quan hóa kết quả. Sau khi huấn luyện mô hình, bạn có thể sử dụng Pandas để phân tích kết quả, sau đó sử dụng Matplotlib để trực quan hóa các chỉ số quan trọng như độ chính xác, độ nhạy và độ đặc hiệu.

Scikit-learn cũng rất hiệu quả trong việc xử lý các bộ dữ liệu lớn, giúp bạn huấn luyện các mô hình học máy phức tạp mà không gặp phải vấn đề về hiệu suất. Điều này đặc biệt quan trọng khi làm việc với các bộ dữ liệu có quy mô lớn như bộ dữ liệu bệnh Alzheimer.

Mặc dù Scikit-learn chủ yếu tập trung vào các thuật toán học máy truyền thống, thư viện này cũng hỗ trợ các mô hình học sâu đơn giản thông qua các công cụ như MLPClassifier và MLPRegressor, cho phép giải quyết các bài toán phức tạp hơn trong việc dự đoán các bệnh lý như Alzheimer, đặc biệt khi dữ liệu có sự tương tác phức tạp giữa các đặc trưng.

## Synthetic Minority Over-sampling Technique (SMOTE)

SMOTE là một kỹ thuật xử lý dữ liệu mất cân bằng bằng cách tạo thêm các mẫu giả lập cho lớp thiểu số. Phương pháp này sử dụng nội suy giữa các điểm dữ liệu hiện có để sinh ra các mẫu mới.



Hình 2.1: Synthetic Minority Oversampling Techique

Ưu điểm:

* Cân bằng dữ liệu hiệu quả mà không làm mất dữ liệu ban đầu.
* Tăng cường khả năng học của mô hình trên lớp thiểu số.

Nhược điểm:

* Có thể sinh ra các mẫu không tự nhiên, dẫn đến giảm tính chính xác.
* Không xử lý được sự chồng lấn giữa các lớp dữ liệu.

## Random Oversampling Technique

Random Oversampling là một kỹ thuật xử lý dữ liệu mất cân bằng bằng cách nhân bản ngẫu nhiên các mẫu hiện có của lớp thiểu số để cân bằng dữ liệu. Phương pháp này đơn giản và không yêu cầu nội suy giữa các điểm dữ liệu.

Ưu điểm:

* Cân bằng dữ liệu hiệu quả mà không làm mất dữ liệu ban đầu.
* Giữ nguyên cấu trúc, giúp mô hình dễ dàng học tập hơn trên lớp thiểu số.

Nhược điểm:

* Có thể dẫn đến hiện tượng overfitting do việc nhân bản dữ liệu.
* Không tạo ra thêm thông tin mới, chỉ lặp lại dữ liệu hiện có.

## Data Scaling

**Scaling** là một kỹ thuật trong **Data Transformation** nhằm thay đổi phạm vi giá trị của các đặc trưng (features) trong dữ liệu để đảm bảo sự đồng nhất về độ lớn giữa các đặc trưng và giúp các thuật toán hoạt động hiệu quả hơn.

Phương pháp:

* **Min - Max Scaler**: Biến đổi dữ liệu về một phạm vi giá trị cụ thể (thường là [0, 1]).

Ưu điểm:

* Scaling giúp tăng độ chính xác cho các thuật toán phụ thuộc vào khoảng cách, như K-Means hoặc KNN.
* Giảm sự chênh lệch về độ lớn giữa các đặc trưng, giúp các thuật toán như SVM, KNN, hoặc Gradient Descent hoạt động hiệu quả hơn.

Nhược điểm:

* Sau khi scale, giá trị dữ liệu không còn phản ánh ý nghĩa ban đầu (ví dụ: chiều cao từ cm chuyển thành số nhỏ như 0.6).
* Phương pháp như **Min-Max Scaling** rất nhạy cảm với giá trị ngoại lệ, có thể làm méo mó phạm vi giá trị của dữ liệu.

## HyperParameter Tuning

Tinh chỉnh siêu tham số là quá trình điều chỉnh các tham số của mô hình học máy (siêu tham số) để tối ưu hóa hiệu suất của mô hình. Siêu tham số không được học trực tiếp từ dữ liệu mà được thiết lập trước khi huấn luyện mô hình, như số lượng cây trong rừng ngẫu nhiên, tỷ lệ học trong mô hình mạng nơ-ron, hoặc độ sâu của cây quyết định.

Phương pháp:

* **Grid Search**: Kiểm tra tất cả các kết hợp có thể của siêu tham số trong một phạm vi đã định.

Ưu điểm:

* Giúp tối ưu hóa hiệu suất mô hình.
* Cải thiện khả năng dự đoán của mô hình.

Nhược điểm:

* Tốn thời gian và tài nguyên tính toán, đặc biệt đối với mô hình phức tạp.
* Không phải lúc nào cũng dễ dàng xác định phạm vi siêu tham số hợp lý.

## Cross Validation

Kiểm định chéo là một kỹ thuật đánh giá mô hình học máy nhằm giảm thiểu vấn đề overfitting và tăng tính tổng quát của mô hình. Quá trình kiểm định chéo chia dữ liệu thành nhiều phần (folds), mỗi phần sẽ lần lượt được sử dụng làm dữ liệu kiểm tra trong khi các phần còn lại được sử dụng làm dữ liệu huấn luyện.

Phương pháp:

* **K-fold Cross Validation:** Dữ liệu được chia thành K phần, và mô hình được huấn luyện và đánh giá K lần, mỗi lần sử dụng một phần khác nhau làm dữ liệu kiểm tra.

Ưu điểm:

* Cung cấp một ước tính chính xác hơn về hiệu suất mô hình.
* Giảm thiểu bias và overfitting bằng cách sử dụng tất cả dữ liệu để huấn luyện và kiểm tra.

Nhược điểm:

* Tốn thời gian tính toán, đặc biệt với các mô hình phức tạp hoặc tập dữ liệu lớn.
* Các phương pháp kiểm định chéo có thể gây ra sự chồng chéo nếu dữ liệu không được chia đúng cách.

## Machine Learning

### Decision Tree

1. **Nền tảng lý thuyết**

Decision Tree (Cây quyết định) là một thuật toán học máy có giám sát (supervised learning) được sử dụng cho cả bài toán phân loại và hồi quy. Thuật toán xây dựng một cây quyết định, trong đó mỗi nút đại diện cho một đặc trưng (feature), mỗi nhánh là một điều kiện, và mỗi lá (leaf) đại diện cho kết quả dự đoán.

1. **Công thức tính toán**

- Tiêu chí phân tách:

Decision Tree sử dụng các phép đo để đánh giá chất lượng phân tách tại mỗi nút. Một số tiêu chí phổ biến:

**Entropy** (đo lường mức độ hỗn loạn của dữ liệu):

Entropy(S)=−i=1∑npilog2(pi)

**Information Gain** (đo lường hiệu quả phân tách):

Gain(S,A)=Entropy(S)−v∈Values(A)∑∣S| Sv| .Entropy(Sv)

**Gini Impurity** (đo lường xác suất chọn sai lớp nếu phân loại ngẫu nhiên):

Gini(S)=1−i=1∑n pi2​​

1. **Ưu điểm**

Kết quả của Decision Tree có thể được trực quan hóa, dễ dàng giải thích cho người không chuyên về kỹ thuật.

Không cần chuẩn hóa hay scale dữ liệu. Decision Tree hoạt động tốt với cả dữ liệu danh mục (categorical) và số (numerical).

Cây có thể điều chỉnh để tập trung vào các lớp hiếm thông qua việc tối ưu tiêu chí phân tách.

1. **Nhược điểm**

Cây sâu hoặc quá chi tiết có thể dẫn đến việc học thuộc dữ liệu huấn luyện, giảm khả năng tổng quát hóa.

Các giá trị ngoại lệ hoặc nhiễu trong dữ liệu có thể làm giảm độ chính xác của cây.

Khi số lượng đặc trưng hoặc giá trị của các đặc trưng rất lớn, việc xây dựng cây có thể trở nên tốn kém về thời gian và tài nguyên.

### Naive Bayes

1. **Nền tảng lý thuyết**

Naive Bayes là một thuật toán học máy có giám sát (supervised learning) được sử dụng chủ yếu trong các bài toán phân loại. Dựa trên Định lý Bayes, NB đưa ra giả định "naive" rằng các đặc trưng (features) là độc lập có điều kiện với nhau, tức là không có sự phụ thuộc giữa các đặc trưng khi đã biết lớp (class).

1. **Công thức tính toán**

- Giả định độc lập có điều kiện:

NB giả định ràng tất cả các đặc trưng , , …, trong tập đặc trưng X là độc lập:

P(X|C) = P(|C) + P(|C) + … + P(|C)

Phương trình phân loại trở thành:

C = arg max(C)

- Tính xác suất:

Xác suất tiên nghiệm P(C): Tỷ lệ xuất hiện của mỗi lớp trong tập huấn luyện.

Xác suất có điều kiện : Tỷ lệ xuất hiện của đặc trưng trong các văn bản thuộc lớp C.

- NB trong phân loại:

Multinomial Naive Bayes: Dùng cho dữ liệu rời rạc.

Bernoulli Naive Bayes: Dùng cho dữ liệu nhị phân.

Gaussian Naive Bayes: Dùng cho dữ liệu liên tục.

1. **Ưu điểm**

Tốc độ nhanh: Naive Bayes rất nhẹ, phù hợp với tập dữ liệu lớn.

Hiệu quả tốt trên dữ liệu nhiều chiều: Thích hợp cho dữ liệu văn bản, nơi mỗi từ là một đặc trưng.

Không đòi hỏi nhiều siêu tham số: NB không cần tinh chỉnh tham số phức tạp.

Khả năng mở rộng: NB có thể được mở rộng với các dạng phân phối khác nhau (Multinomial, Gaussian)

1. **Nhược điểm**

Giả định độc lập không thực tế: Các từ trong văn bản thường có liên quan với nhau, điều này có thể làm giảm độ chính xác.

Không linh hoạt với dữ liệu phức tạp: NB không thể nắm bắt tốt các mối quan hệ phức tạp giữa các đặc trưng.

### XGBoost

1. **Nền tảng lý thuyết**

XGBoost (Extreme Gradient Boosting), được thiết kế cho bài toán phân loại. Đây là một thuật toán học máy mạnh mẽ dựa trên cây quyết định, áp dụng kỹ thuật boosting để cải thiện hiệu suất của mô hình.

**1. Boosting**

Boosting là một kỹ thuật ensemble học máy, nơi nhiều mô hình yếu (weak learners) được huấn luyện tuần tự, mỗi mô hình cố gắng sửa lỗi của mô hình trước đó. Mục tiêu là kết hợp các mô hình yếu để tạo ra một mô hình mạnh với hiệu suất cao hơn.

**2. Gradient Boosting**

Gradient Boosting tối ưu hóa mô hình thông qua việc giảm thiểu hàm mất mát bằng cách xây dựng tuần tự các cây quyết định, sử dụng đạo hàm gradient để hướng dẫn.

**3. XGBoost**

XGBoost là một cải tiến của Gradient Boosting, được tối ưu hóa về tốc độ và hiệu suất. Một số đặc điểm nổi bật:

Tối ưu hóa hiệu suất: Tích hợp xử lý song song và quản lý bộ nhớ hiệu quả.

Hàm mất mát tùy chỉnh: Hỗ trợ hàm mất mát log-loss cho bài toán phân loại.

Regularization (L1 và L2): Giảm overfitting thông qua điều chuẩn trọng số.

Xử lý dữ liệu thiếu: Tự động xử lý các giá trị bị thiếu trong dữ liệu.

Công thức tính toán

Các tham số quan trọng:

max\_depth: Độ sâu tối đa của mỗi cây quyết định (quyết định khả năng phân chia dữ liệu).

learning\_rate: Tốc độ học, xác định mức độ điều chỉnh mô hình qua mỗi cây.

n\_estimators: Số lượng cây quyết định trong rừng.

subsample: Tỷ lệ mẫu dữ liệu sử dụng cho mỗi cây (để giảm overfitting).

colsample\_bytree: Tỷ lệ cột được chọn cho mỗi cây.

objective: Hàm mục tiêu, thường là "binary: logistic" cho phân loại nhị phân hoặc "multi: softprob" cho đa nhãn.

reg\_lambda và reg\_alpha: Điều chuẩn L2 và L1.

Ưu điểm

Hiệu suất cao: XGBoost thường đạt kết quả tốt trên nhiều bài toán thực tế.

Khả năng tổng quát hóa tốt: Điều chỉnh regularization giúp giảm overfitting.

Tùy biến cao: Hỗ trợ nhiều tham số và hàm mất mát tùy chỉnh.

Xử lý dữ liệu thiếu: Không yêu cầu tiền xử lý dữ liệu thiếu.

d, Nhược điểm

Chi phí tính toán cao: Việc huấn luyện có thể mất nhiều thời gian, đặc biệt với dữ liệu lớn.

Nhạy cảm với siêu tham số: Cần tinh chỉnh tham số (tuning) để đạt hiệu suất tối ưu.

Không phù hợp cho dữ liệu ít: XGBoost cần một lượng dữ liệu đáng kể để phát huy hiệu quả.

### Random Forest

Random Forest là một thuật toán học máy có giám sát dựa trên kỹ thuật học tổ hợp (**ensemble learning**) nhằm cải thiện hiệu suất và độ chính xác bằng cách kết hợp nhiều cây quyết định (**decision trees**). Ý tưởng chính của Random Forest là sử dụng một tập hợp các mô hình yếu (weak learners) để tạo ra một mô hình mạnh hơn, giảm thiểu overfitting và tăng khả năng tổng quát hóa. Để xây dựng Random Forest, thuật toán áp dụng phương pháp **bagging (Bootstrap Aggregation)**, trong đó tạo ra nhiều tập dữ liệu con bằng cách chọn ngẫu nhiên các mẫu (có hoàn lại) từ tập dữ liệu gốc, và mỗi tập con này được sử dụng để huấn luyện một cây quyết định độc lập. Tại mỗi nút của cây, thuật toán chỉ xem xét một tập con ngẫu nhiên của các đặc trưng để thực hiện việc phân tách, nhằm tăng tính đa dạng giữa các cây và giảm tương quan giữa chúng.

Khi dự đoán, Random Forest kết hợp đầu ra của các cây bằng cách lấy **đa số phiếu** (đối với bài toán phân loại) hoặc tính **trung bình** (đối với bài toán hồi quy). Một số tham số quan trọng của Random Forest bao gồm: số lượng cây (**n\_estimators**), càng nhiều cây thì mô hình càng ổn định nhưng cũng tốn thời gian tính toán hơn; số lượng đặc trưng được xem xét tại mỗi nút (**max\_features**), với các giá trị thường dùng như d\sqrt{d}d​ cho bài toán phân loại và d/3d/3d/3 cho bài toán hồi quy (với ddd là tổng số đặc trưng); và chiều sâu tối đa của cây (**max\_depth**) để kiểm soát độ phức tạp của mỗi cây.

Random Forest có nhiều ưu điểm như khả năng xử lý tốt dữ liệu mất cân bằng, dữ liệu có nhiều nhiễu, và cả dữ liệu phi tuyến. Ngoài ra, nó không yêu cầu tiền xử lý dữ liệu như chuẩn hóa hay xử lý giá trị ngoại lai. Tuy nhiên, Random Forest cũng có một số hạn chế, bao gồm việc tốn tài nguyên tính toán, khó giải thích trực quan khi có nhiều cây, và có nguy cơ overfitting nếu số lượng cây hoặc độ sâu cây không được kiểm soát tốt.

### KNN

**K-Nearest Neighbors (KNN)** là một thuật toán học máy thuộc nhóm **học không tham số** và **có giám sát**, được sử dụng phổ biến trong các bài toán phân loại và hồi quy. KNN hoạt động dựa trên nguyên tắc đơn giản: một điểm dữ liệu mới sẽ được phân loại hoặc dự đoán dựa trên các "láng giềng gần nhất" của nó trong không gian đặc trưng.

Nguyên lý hoạt động của KNN bao gồm các bước chính sau: Đầu tiên, thuật toán tính khoảng cách giữa điểm dữ liệu mới và tất cả các điểm trong tập huấn luyện. Khoảng cách này có thể được đo bằng các phương pháp phổ biến như **Euclidean**, **Manhattan**, hoặc **Minkowski**, tùy thuộc vào bài toán. Sau đó, thuật toán xác định KKK điểm gần nhất với điểm dữ liệu mới (với KKK là một tham số do người dùng chọn). Cuối cùng, trong bài toán phân loại, KNN sẽ dự đoán lớp cho điểm mới dựa trên **đa số phiếu** từ các điểm láng giềng gần nhất. Trong bài toán hồi quy, giá trị dự đoán được tính là **trung bình** giá trị của các điểm láng giềng này.

KNN có một số **ưu điểm** nổi bật, như dễ hiểu và dễ triển khai, không yêu cầu quá trình huấn luyện mô hình phức tạp, và phù hợp với dữ liệu phi tuyến. Ngoài ra, KNN có thể xử lý cả dữ liệu phân loại và hồi quy một cách hiệu quả. Tuy nhiên, KNN cũng có **nhược điểm**, bao gồm việc tốn tài nguyên tính toán khi xử lý tập dữ liệu lớn do phải tính toán khoảng cách cho mọi điểm trong tập huấn luyện. Kết quả của KNN cũng phụ thuộc mạnh vào giá trị của KKK và cách chọn khoảng cách. Nếu KKK quá nhỏ, mô hình có thể bị overfitting, trong khi KKK quá lớn có thể làm giảm độ chính xác. Ngoài ra, KNN nhạy cảm với dữ liệu mất cân bằng, khi các lớp có số lượng không đồng đều sẽ ảnh hưởng đến kết quả phân loại.

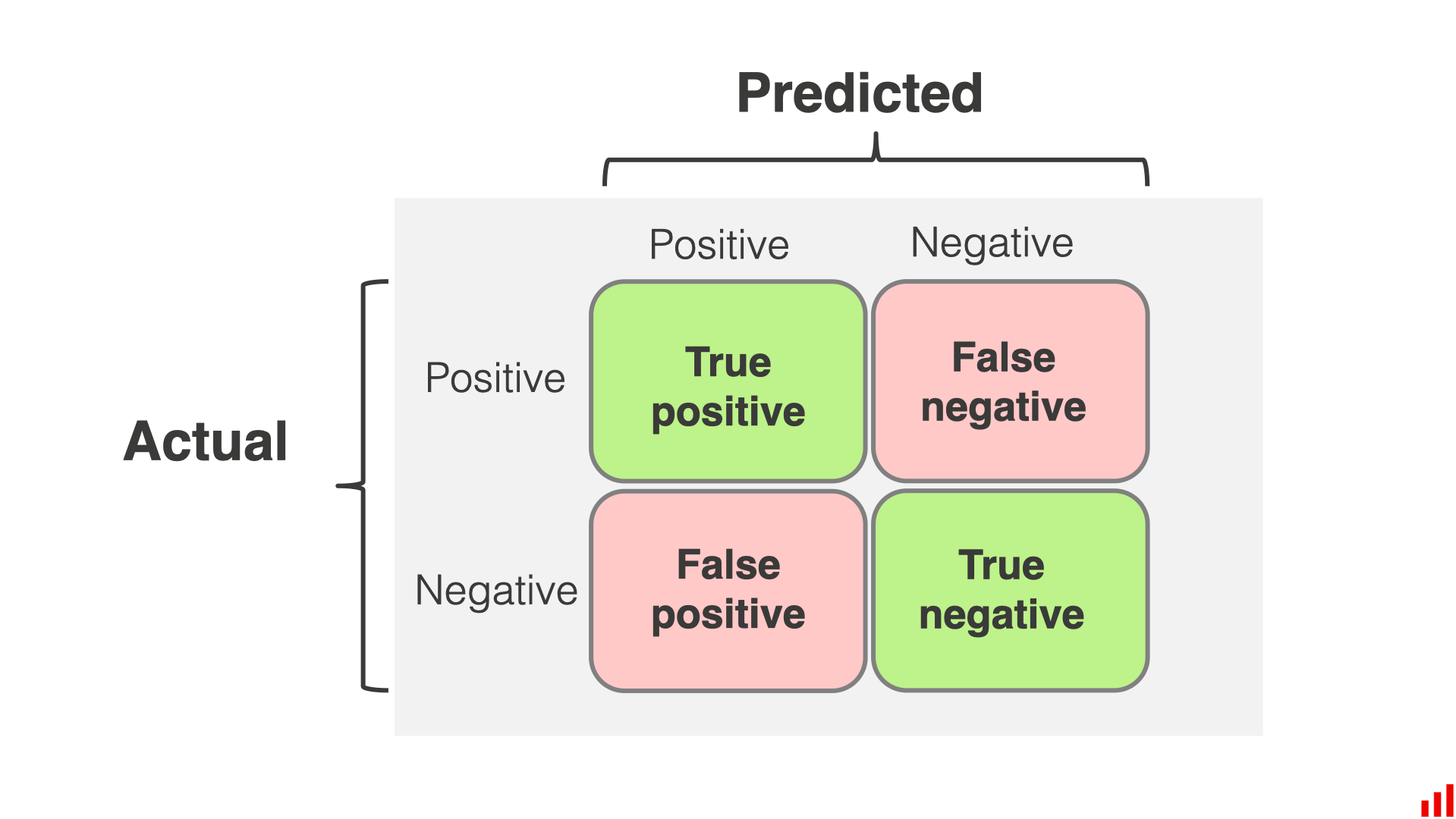
Trong thực tế, KNN thường được sử dụng trong các bài toán như phân loại hình ảnh, nhận dạng mẫu, hệ thống khuyến nghị, và phân tích dữ liệu y tế. Để tối ưu hiệu suất của KNN, các kỹ thuật như chuẩn hóa dữ liệu (normalization), giảm chiều dữ liệu (dimensionality reduction), và sử dụng cấu trúc dữ liệu như **KD-tree** hoặc **Ball-tree** để tăng tốc độ tìm kiếm láng giềng thường được áp dụng.

## Confusion Matrix

Confusion Matrix (Ma trận nhầm lẫn) là một công cụ đánh giá mô hình phân loại, giúp so sánh giữa các dự đoán của mô hình và các giá trị thực tế trong tập kiểm tra. Ma trận này cung cấp cái nhìn tổng quan về hiệu suất của mô hình, từ đó giúp chúng ta nhận diện được các lỗi mà mô hình mắc phải.

Ma trận nhầm lẫn cho một bài toán phân loại nhị phân sẽ có 4 ô:

* True Positive (TP): Số lượng mẫu mà mô hình dự đoán đúng là lớp dương (positive).
* False Positive (FP): Số lượng mẫu mà mô hình dự đoán sai là lớp dương, trong khi thực tế là lớp âm (negative).
* True Negative (TN): Số lượng mẫu mà mô hình dự đoán đúng là lớp âm.
* False Negative (FN): Số lượng mẫu mà mô hình dự đoán sai là lớp âm, trong khi thực tế là lớp dương.



Hình 2.2: Confusion matrix

**Ưu điểm**:

* Giúp phân tích chi tiết các loại lỗi mà mô hình mắc phải (ví dụ: sai thành lớp dương hay lớp âm).
* Cung cấp cái nhìn rõ ràng về hiệu suất của mô hình trong từng lớp.

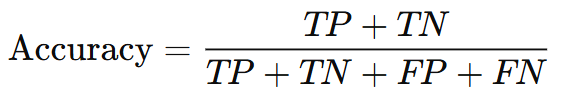
**Nhược điểm**:

* Không cung cấp thông tin về các lỗi liên quan đến độ chính xác tổng thể của mô hình, mà chỉ tập trung vào các phân loại chính xác và sai lệch giữa các lớp.

## Evaluated Metrics

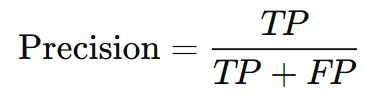
Các **metrics đánh giá** giúp chúng ta đo lường hiệu suất của mô hình từ nhiều góc độ khác nhau. Dưới đây là một số metric quan trọng trong việc đánh giá mô hình phân loại:

1. **Accuracy (Độ chính xác)**:
   * **Định nghĩa**: Là tỷ lệ giữa số mẫu được phân loại đúng trên tổng số mẫu.
   * **Công thức**:



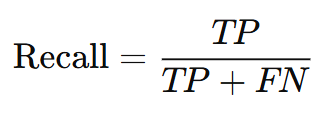
* + **Ưu điểm**: Dễ hiểu và tính toán nhanh.
  + **Nhược điểm**: Không hiệu quả khi dữ liệu mất cân bằng (class imbalance), vì mô hình có thể đạt độ chính xác cao bằng cách dự đoán đúng lớp chiếm ưu thế.

1. **Precision (Độ chính xác)**
   * **Định nghĩa**: Là tỷ lệ giữa số mẫu dự đoán đúng là lớp dương so với tổng số mẫu được mô hình dự đoán là lớp dương.
   * **Công thức**:



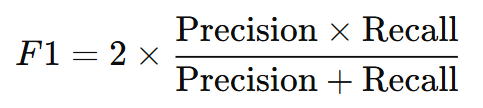
* + **Ưu điểm**: Chỉ ra được độ chính xác của mô hình khi dự đoán lớp dương.
  + **Nhược điểm**: Nếu lớp dương hiếm, precision có thể không phản ánh được khả năng mô hình phân biệt tốt giữa các lớp.

1. **Recall (Độ nhạy)**
   * **Định nghĩa**: Là tỷ lệ giữa số mẫu dự đoán đúng là lớp dương so với tổng số mẫu thực sự thuộc lớp dương.
   * **Công thức**:



* + **Ưu điểm**: Giúp đánh giá khả năng của mô hình trong việc nhận diện tất cả các mẫu dương.
  + **Nhược điểm**: Một mô hình có recall cao có thể dẫn đến nhiều dự đoán sai (false positives).

1. **F1-Score**
   * **Định nghĩa**: F1-Score là trung bình hài hòa của Precision và Recall. Nó giúp cân bằng giữa độ chính xác và độ nhạy, đặc biệt hữu ích khi dữ liệu bị mất cân bằng.
   * **Công thức**:

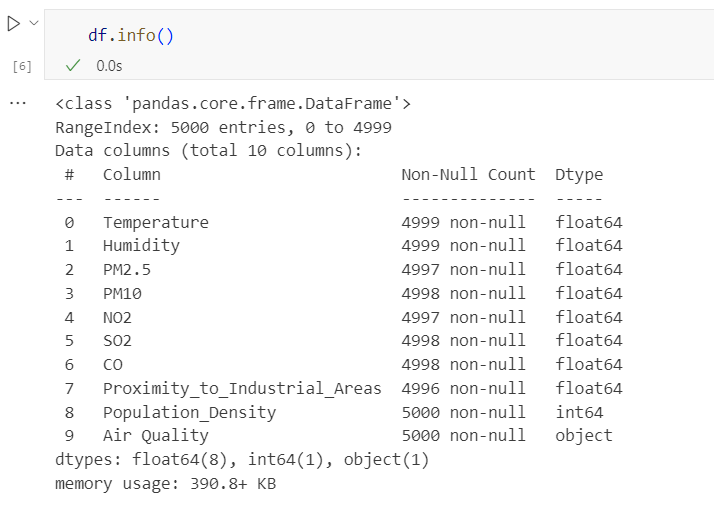


* + **Ưu điểm**: Cung cấp cái nhìn cân bằng về độ chính xác và độ nhạy, đặc biệt trong các bài toán phân loại mất cân bằng.
  + **Nhược điểm**: Không phải lúc nào cũng cung cấp thông tin đầy đủ nếu không có sự cân nhắc về các yếu tố khác như Accuracy.

# GIẢI PHÁP

## Mã nguồn tiền xử lý dữ liệu

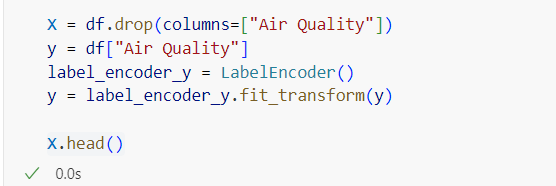
Đầu tiên em sử dụng hàm df.info() để kiểm tra các đặc trưng nào bị thiếu và có kiểu dữ liệu thế nào. Qua đó có thể biết thêm 1 số thông tin về kích thước của bộ dữ liệu, tên của các đặc trưng đó sẽ như nào, kiểu dữ liệu đã chuẩn hóa đúng chưa.



Hình 3.1: Mã nguồn kiểm tra dữ liệu

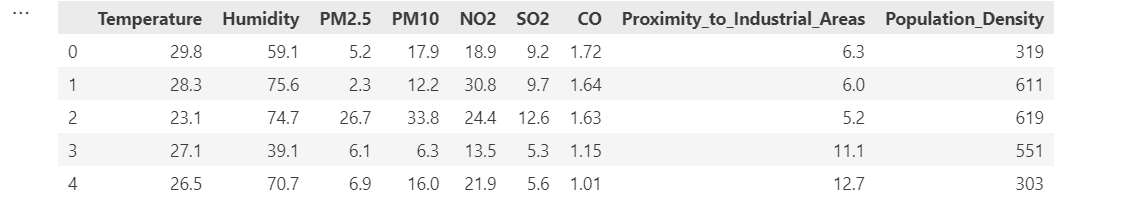
1. **Mã hóa các cột chứa dữ liệu văn bản**

Em mã hóa các giá trị văn bản bằng cách sử dụng LabelEncoder()



Hình 3.2: Mã nguồn encode data

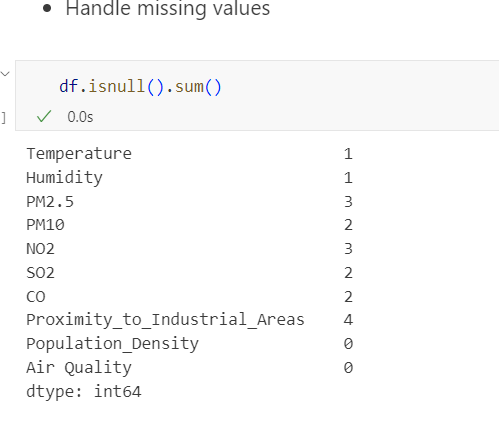
Dữ liệu sau khi được mã hóa



Hình 3.3: Dữ liệu sau khi được mã hóa

1. **Xử lý dữ liệu mất mát**

Các cột bên dưới được in ra dưới màn hình là những cột đều mang các giá trị thiếu Các cột này sẽ cần phải được thêm dữ liệu vào những chỗ trống để có thể phù hợp cho mô hình huấn luyện và phân tích.



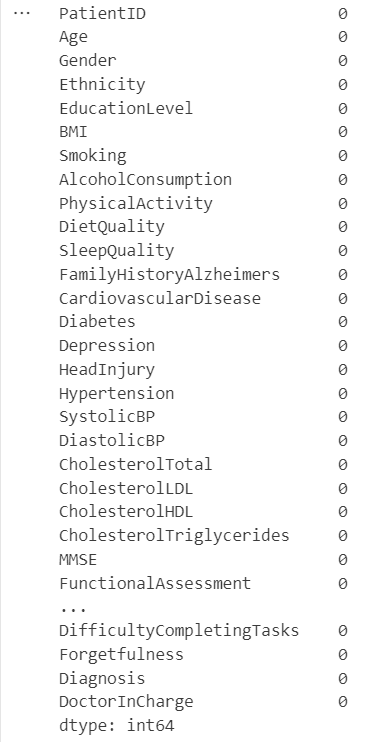
Hình 3.4: Mã nguồn thống kê dữ liệu khuyết thiếu

Em điền giá trị trung bình của mỗi cột cho những dữ liệu khuyết thiếu.



Hình 3.5: Mã nguồn xử lý dữ liệu khuyết thiếu

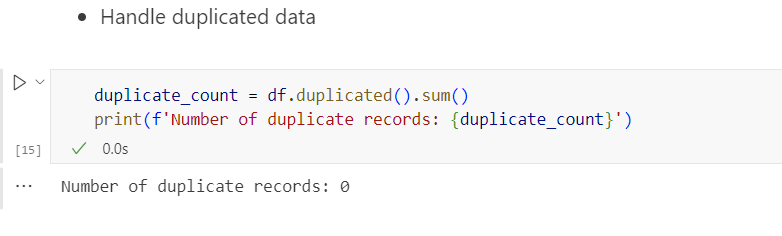
Dữ liệu sau khi xử lý:



Hình 3.6: Dữ liệu khuyết thiếu đã được xử lý

1. **Xóa các bản ghi trùng lặp**

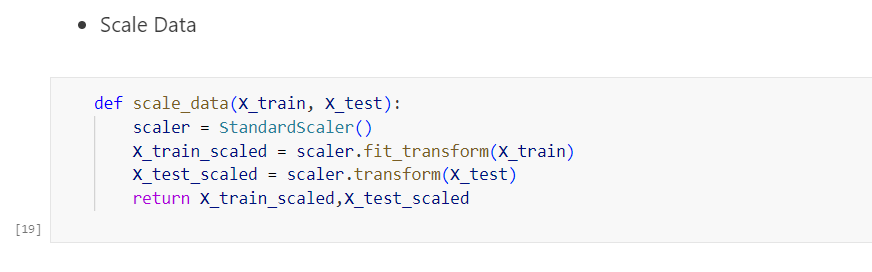
Em sử dụng hàm df.drop\_duplicate() để xóa bỏ những bản ghi trùng lặp. Các bản ghi trùng lặp sẽ bị xóa bỏ để tránh dư thừa dữ liệu cho quá trình huấn luyện mô hình và tiền xử lý dữ liệu.



Hình 3.7: Mã nguồn xử lý các bản ghi trùng lặp

1. **Rescale data**

Để rescale data em tạo ra 1 hàm scale\_date với 2 tham số đầu vào là x\_train và x\_test, hàm này có tác dụng rescale lại dữ liệu được truyền vào.



Hình 3.8: Mã nguồn rescale dữ liệu

1. **Cân bằng dữ liệu**

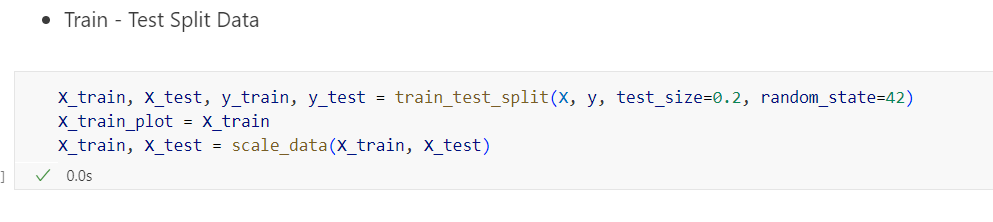
Để cân bằng dữ liệu của tập nhãn, em tạo ra 1 hàm Imbalance\_data nhằm cân bằng dữ liệu với tùy chọn smote và random oversampler



Hình 3.9: Mã nguồn cân bằng dữ liệu

1. **Phân tách dữ liệu**

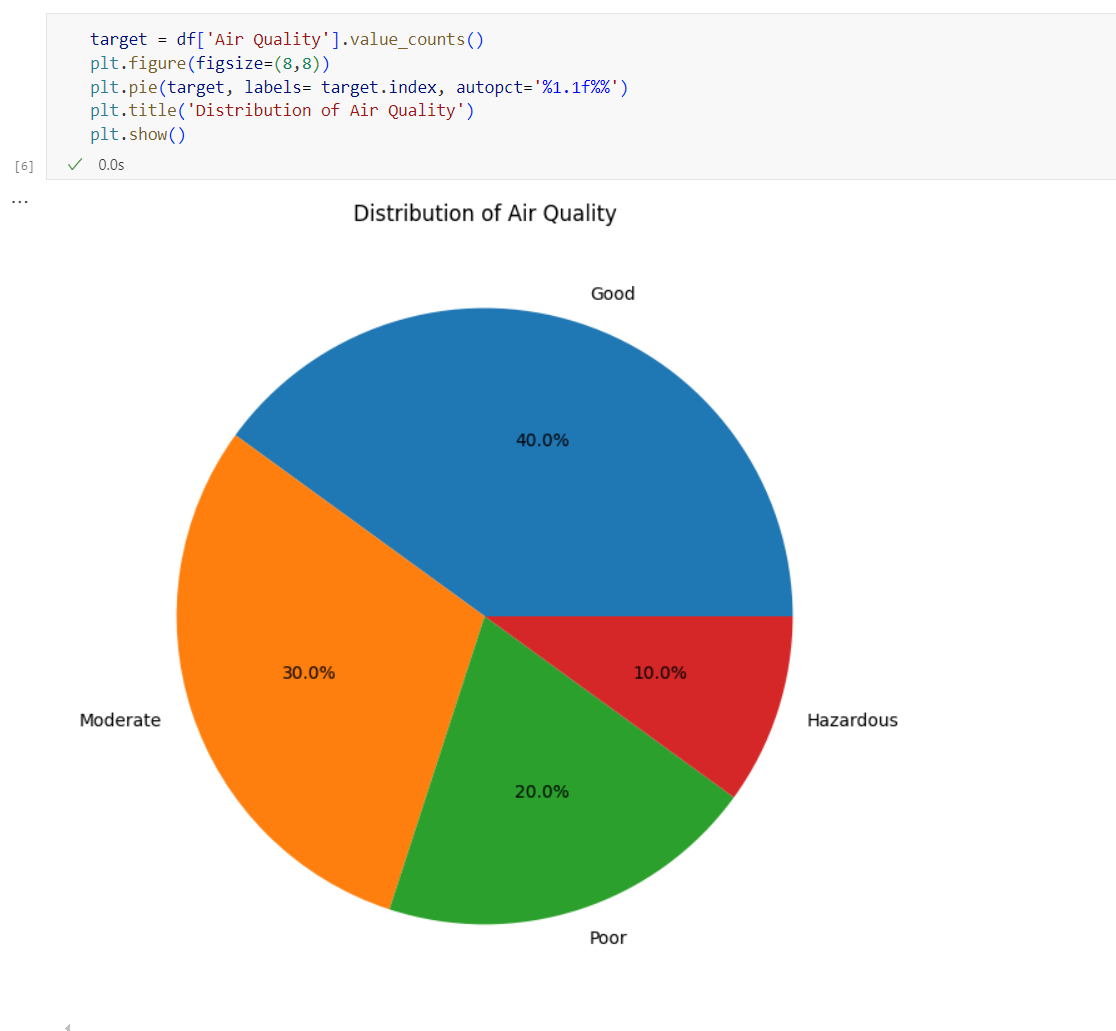
Dữ liệu trong bài này được em chia thành 80/20 với 80% là tập train và 20% là tập test



Hình 3.10: Mã nguồn phân tách dữ liệu

## Mã nguồn trực quan hóa dữ liệu

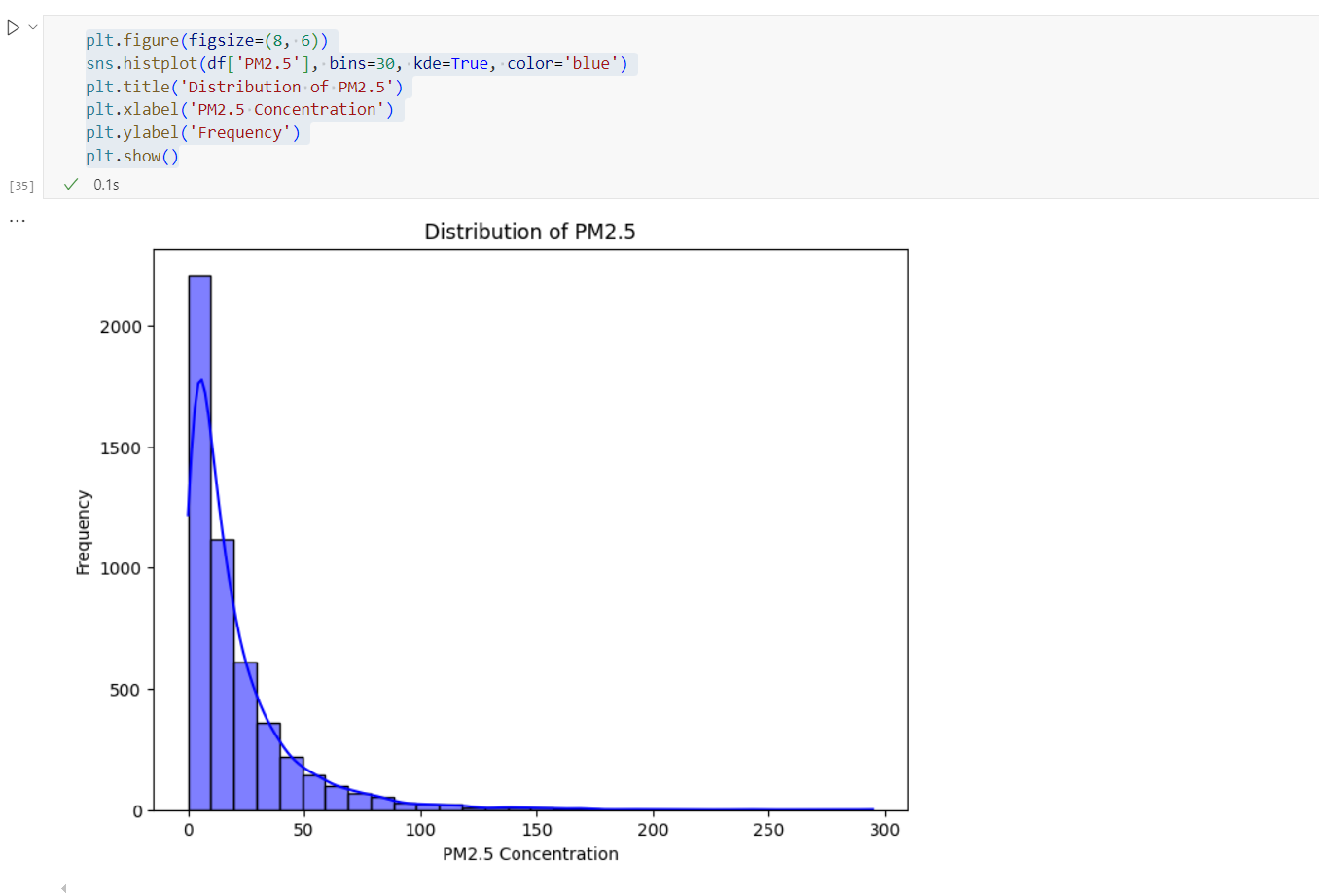
1. Phân phối nhãn dữ liệu



Hình 3.11: Phân phối của nhãn dữ liệu

* Mục đích: Xác định sự phân bổ của tập nhãn.

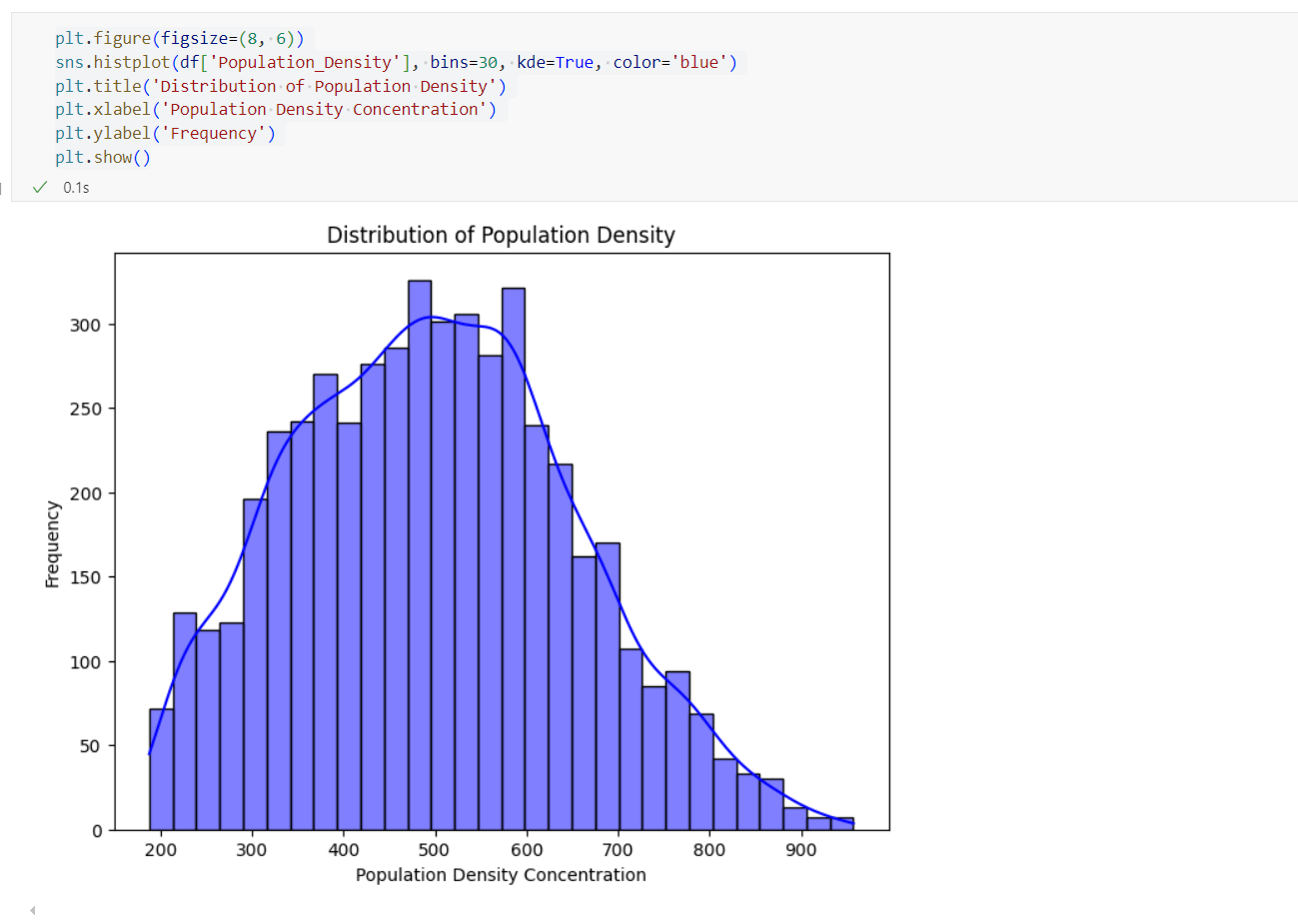
1. Phân phối của PM2.5



Hình 3.12: Phân phối PM2.5

* Mục đích: Thống kê phân phối các đặc trưng PM2.5.

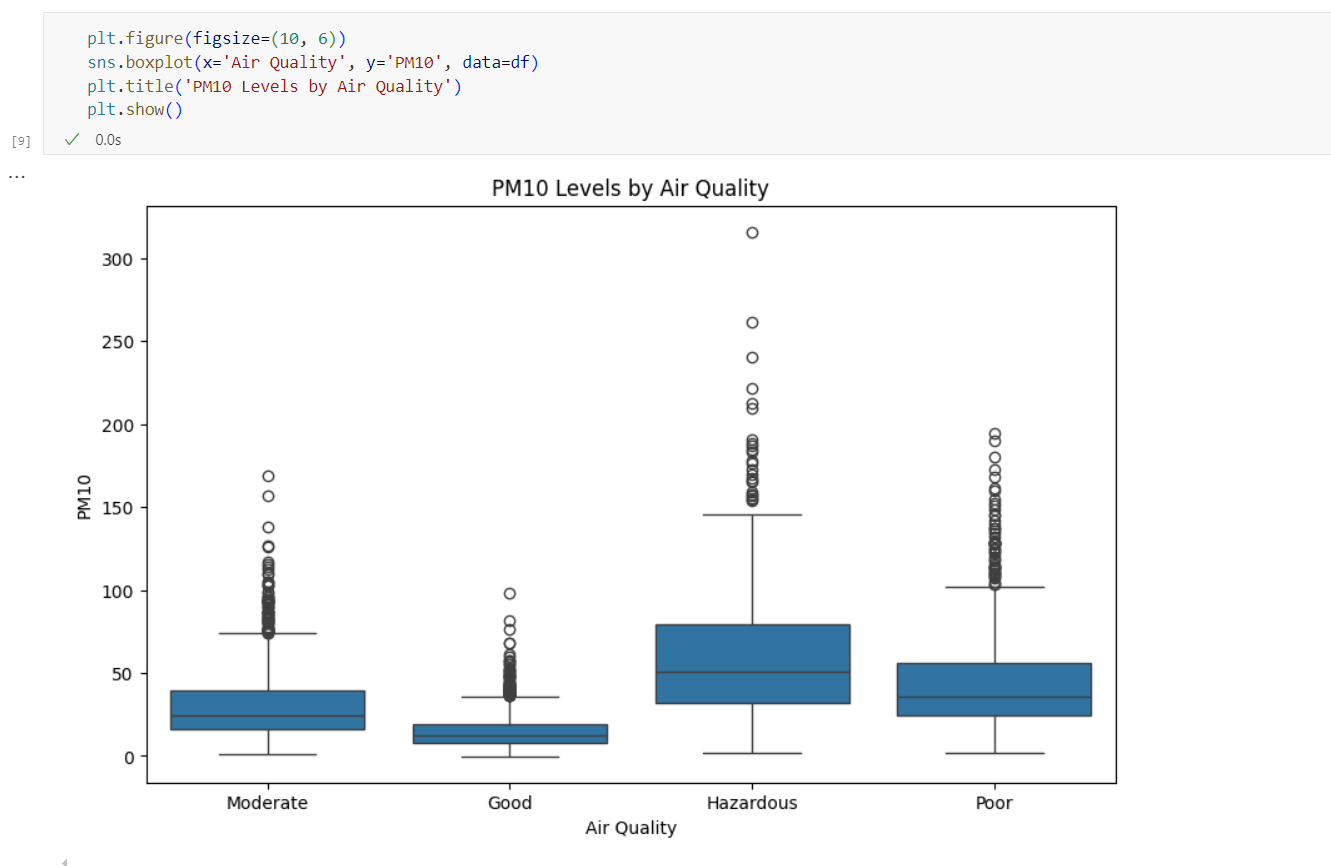
1. Phân phối mật độ dân số



Hình 3.13: Phân phối của mật độ dân số

* Mục đích: Thống kê mật độ dân số của dữ liệu.

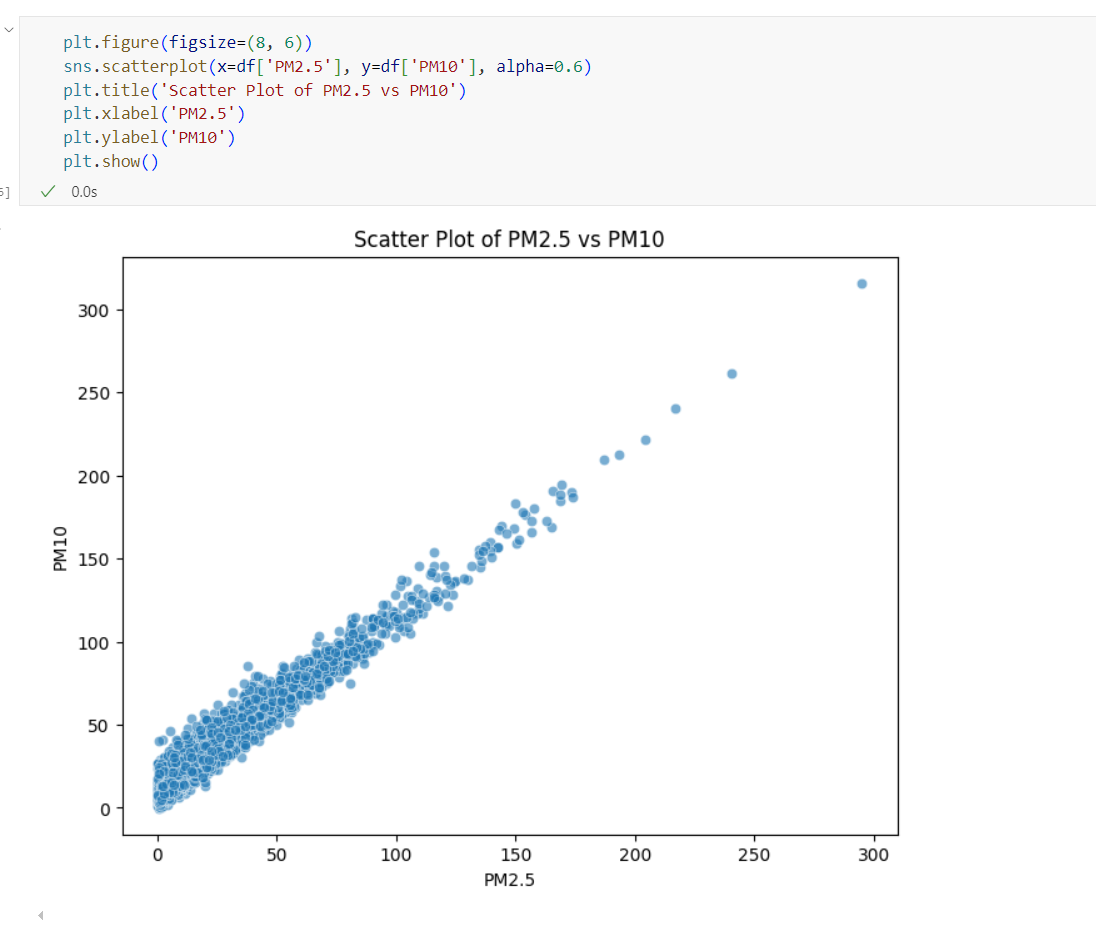
1. Phân tích PM10 theo chất lượng không khí



Hình 3.14: Phân tích PM10 theo chất lượng không khí

* Mục đích: Phân tích PM10 theop chất lượng không khí.

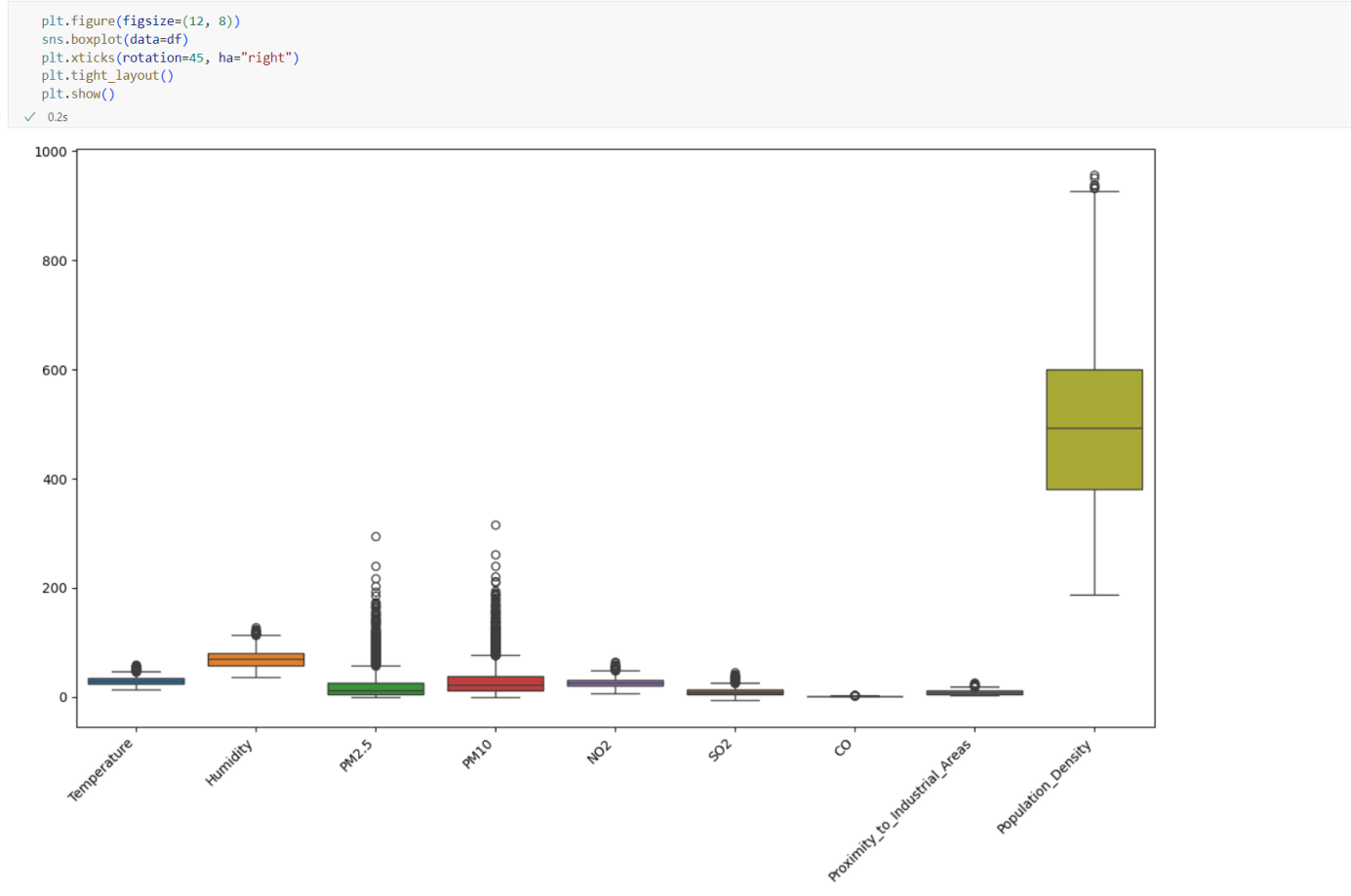
1. Biểu đồ phân tán của PM2.5 và PM10



Hình 3.15: Biểu đồ phân tán của PM2.5 và PM10

* Mục đích: Biểu đồ phân tán của PM2.5 và PM10.

1. Biểu đồ hộp các đặc trưng



Hình 3.16: Biểu đồ hộp các đặc trưng

* Mục đích: Biểu đồ hộp các đặc trưng.

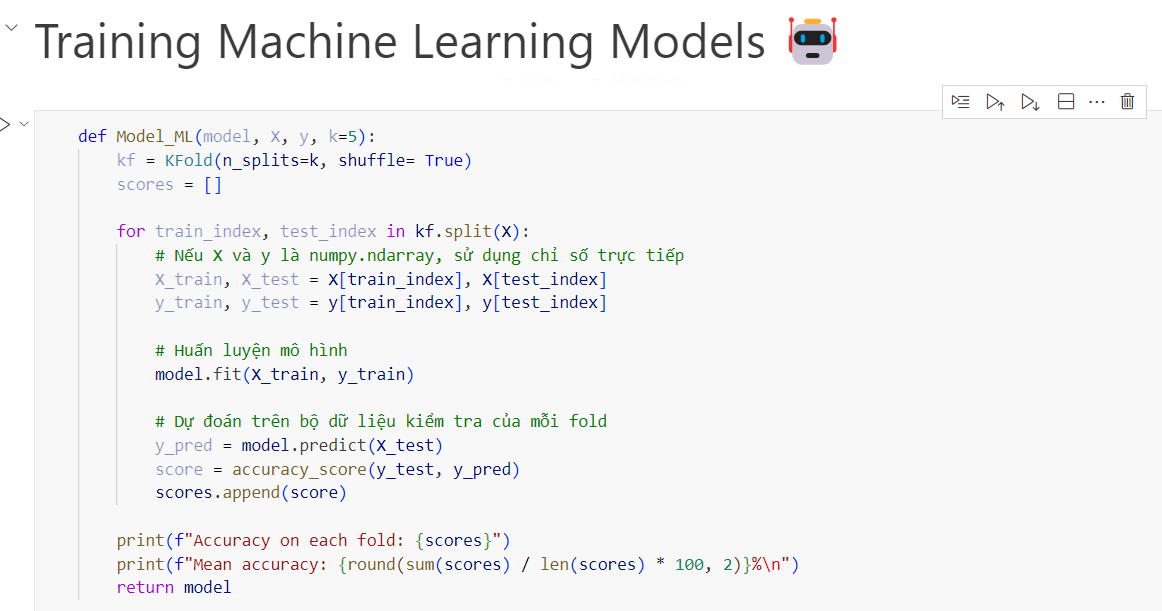
## Mã nguồn Machine Learning Models

**Mã nguồn Import thư viện**



Hình 3.17: Mã nguồn Import thư viện

**Mã nguồn hàm Model\_ML**



Hình 3.18: Mã nguồn hàm Model\_ML

Hàm này em dùng kĩ thuật K-fold để đánh giá một mô hình trên nhiều tập dữ liệu để có một cái nhìn tổng quan hơn về mô hình.

**Mã nguồn hàm Gridsearch\_params**



Hình 3.19: Mã nguồn hàm gridsearch\_params

Hàm này em sử dụng kỹ thuật gridsearch để tìm ra các tham số tối ưu nhất cho mô hình.

**Mã nguồn hàm Evaluation\_models**



Hình 3.20: Mã nguồn hàm gridsearch\_params

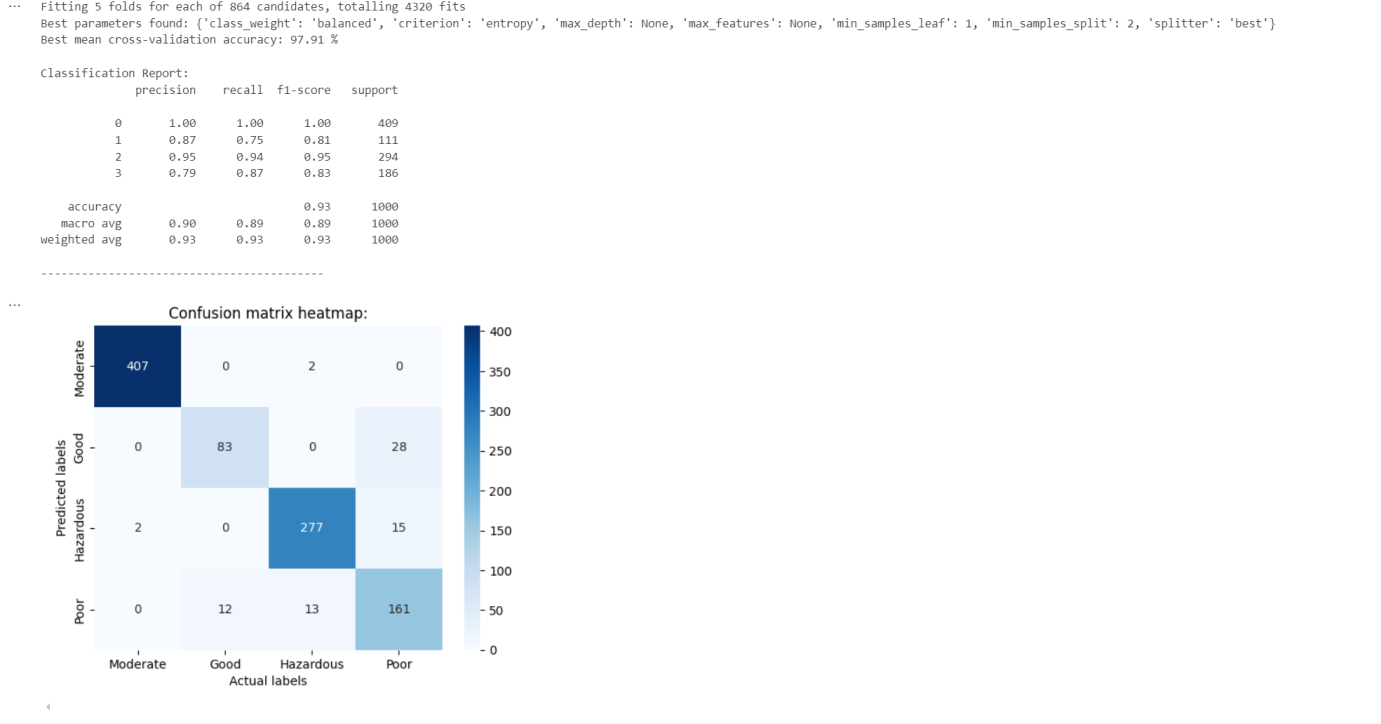
Hàm này để đánh giá độ hiệu quả của mô hình bằng classsification report.

**Mã nguồn mô hình Decision Tree**



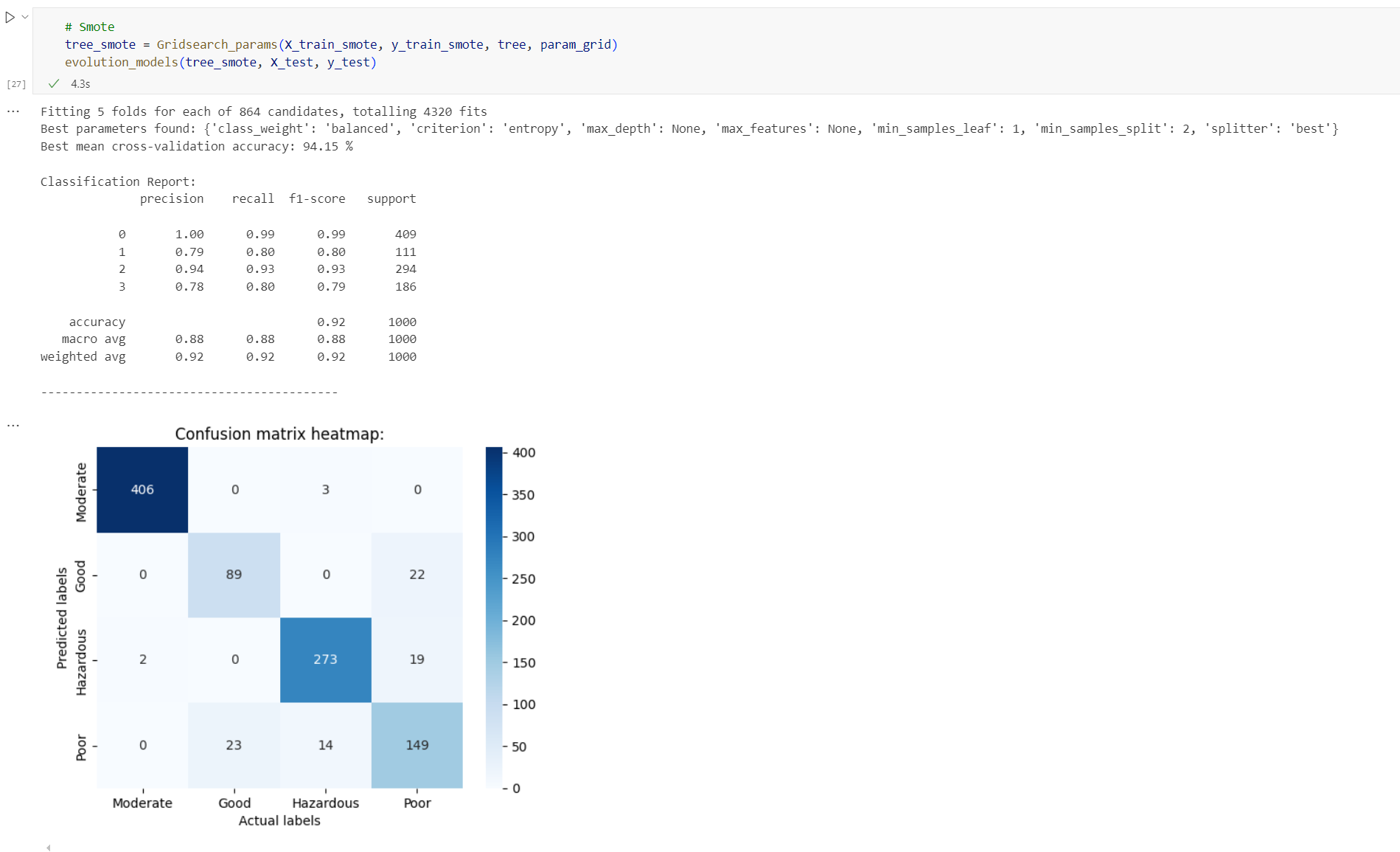
Hình 3.21: Mã nguồn Decision Tree model

**Đánh giá mô hình với cân bằng dữ liệu random oversampler**



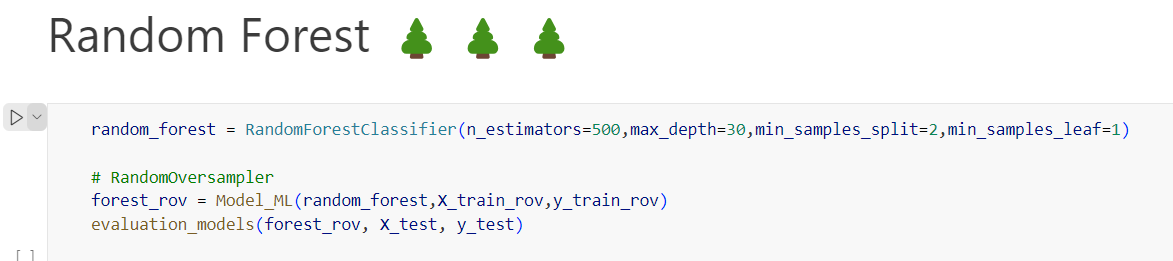
Hình 3.22: Hiệu suất Decision Tree với Random Oversampler

**Đánh giá mô hình với cân bằng dữ liệu smote**



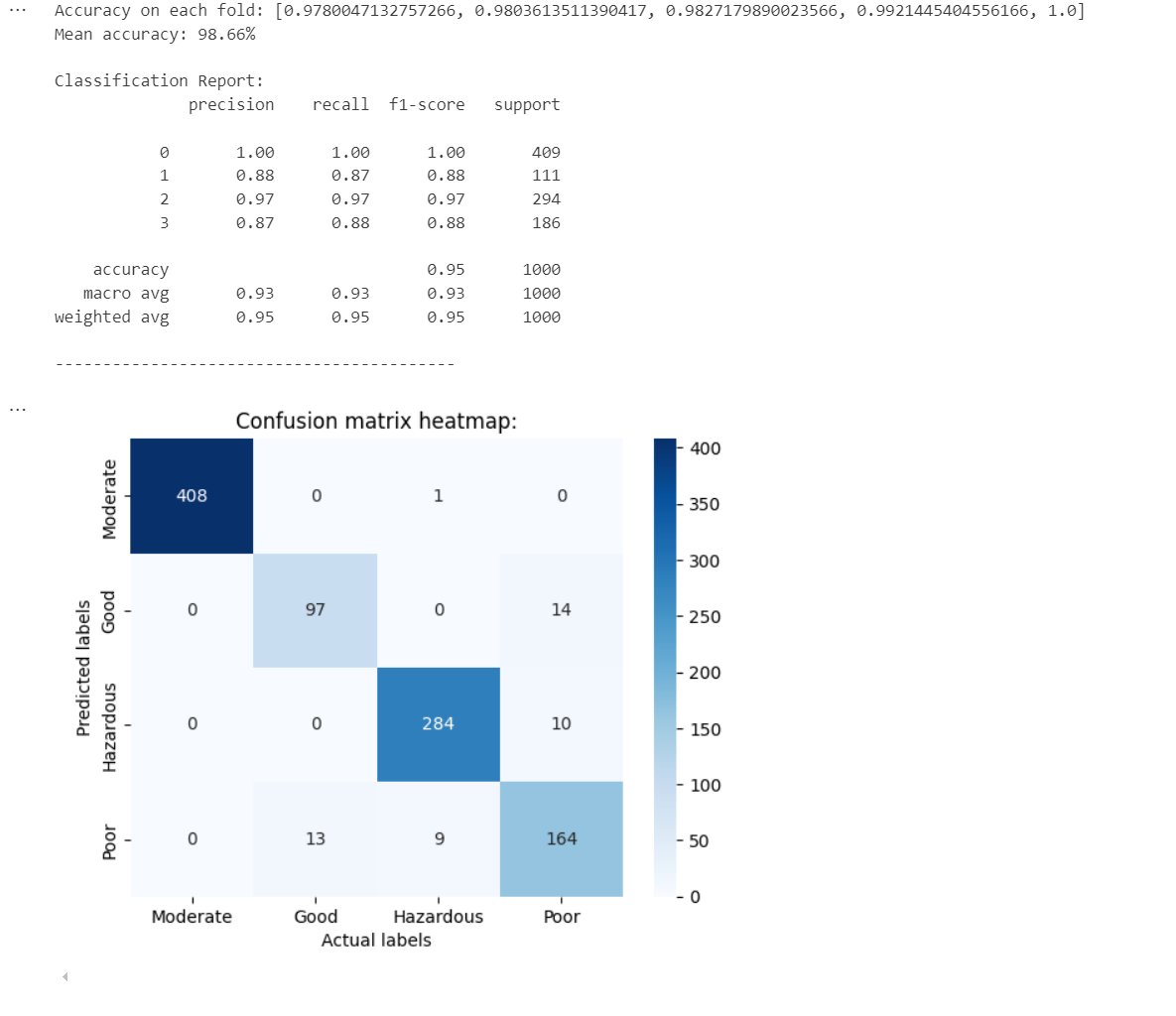
Hình 3.23: Hiệu suất Decision Tree với Smote

**Mã nguồn mô hình Random forest**



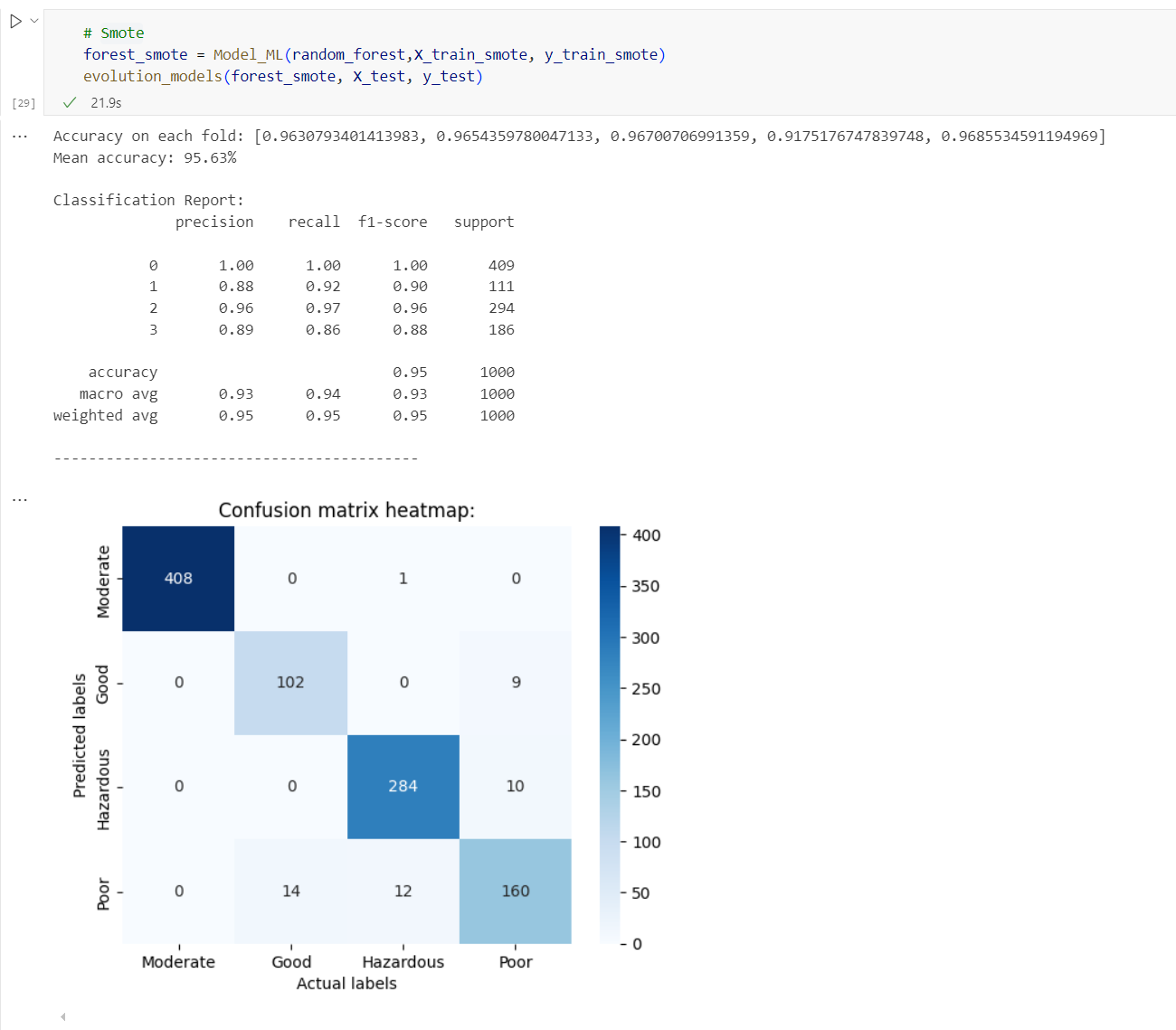
Hình 3.24: Mã nguồn Random Forest model

**Đánh giá mô hình với cân bằng dữ liệu random oversampler**



Hình 3.25: Hiệu suất Random Forest với Random Oversampler

**Đánh giá mô hình với cân bằng dữ liệu smote**



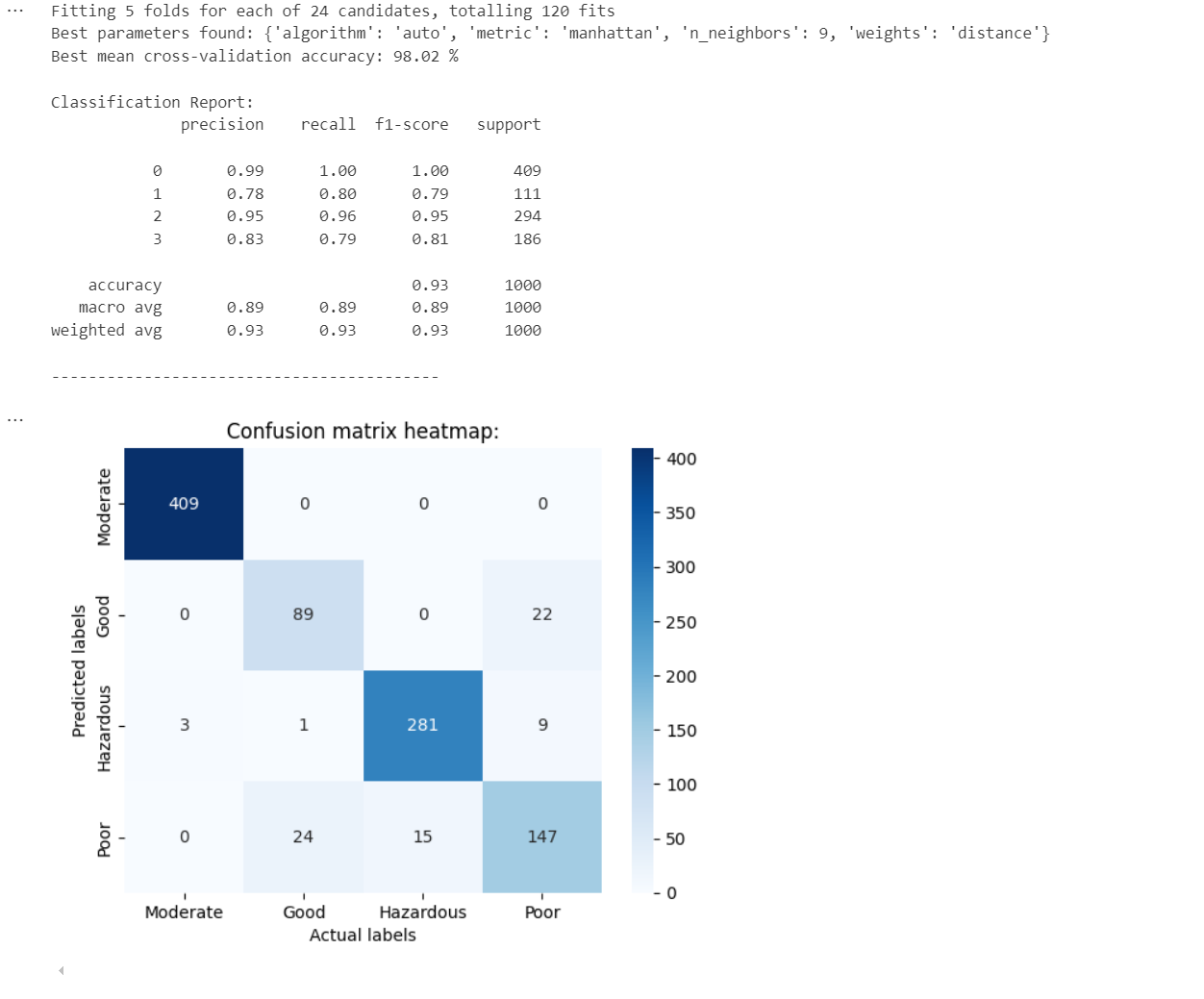
Hình 3.26: Hiệu suất Random Forest với Smote

**Mã nguồn mô hình KNN**

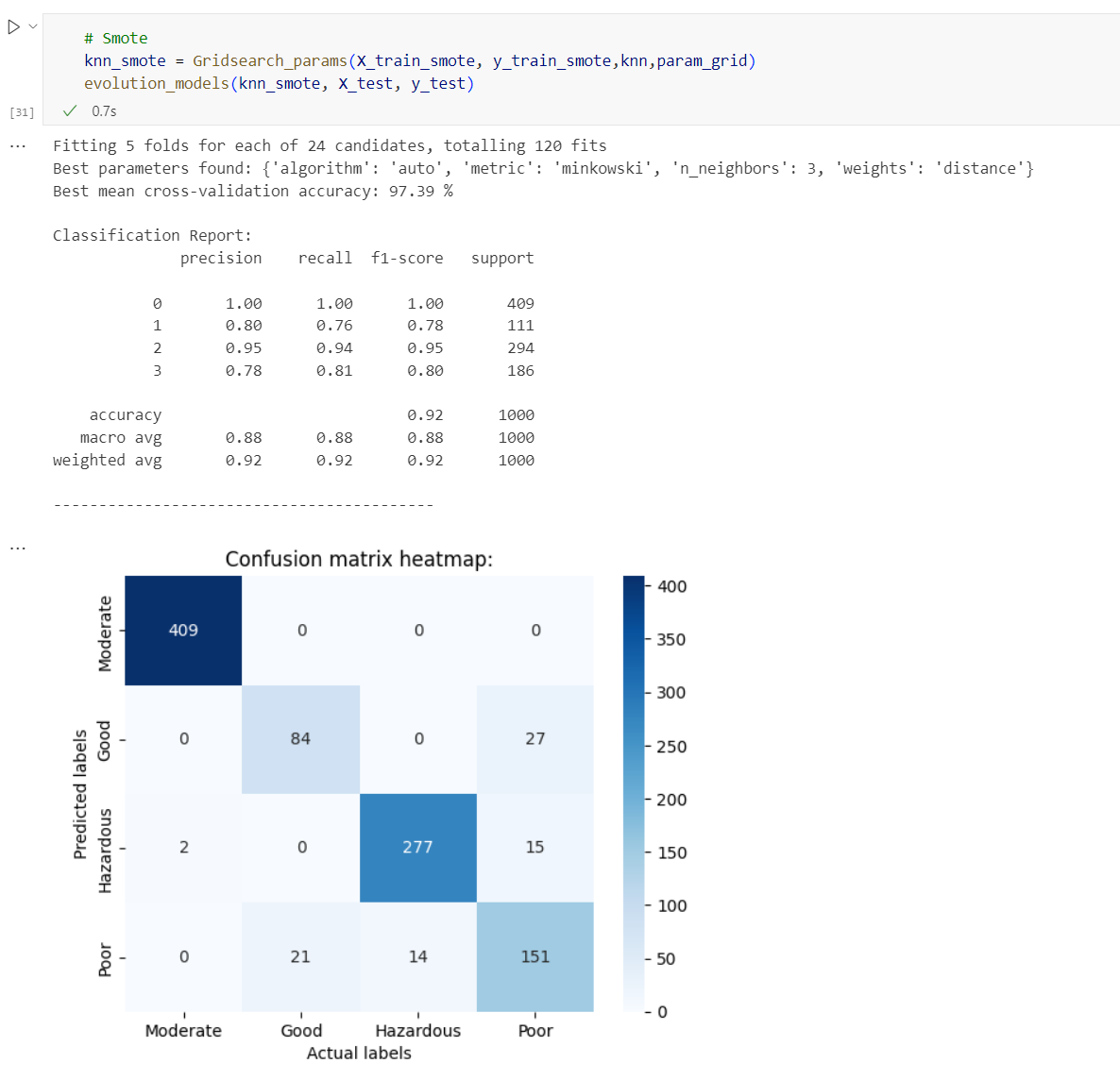


Hình 3.27: Mã nguồn KNN model

**Đánh giá mô hình với cân bằng dữ liệu random oversampler**

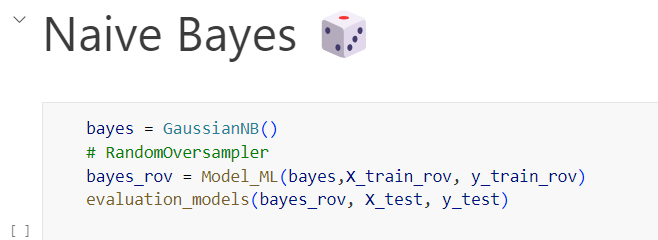
Hình 3.28: Hiệu suất KNN với Random Oversampler

**Đánh giá mô hình với cân bằng dữ liệu smote**



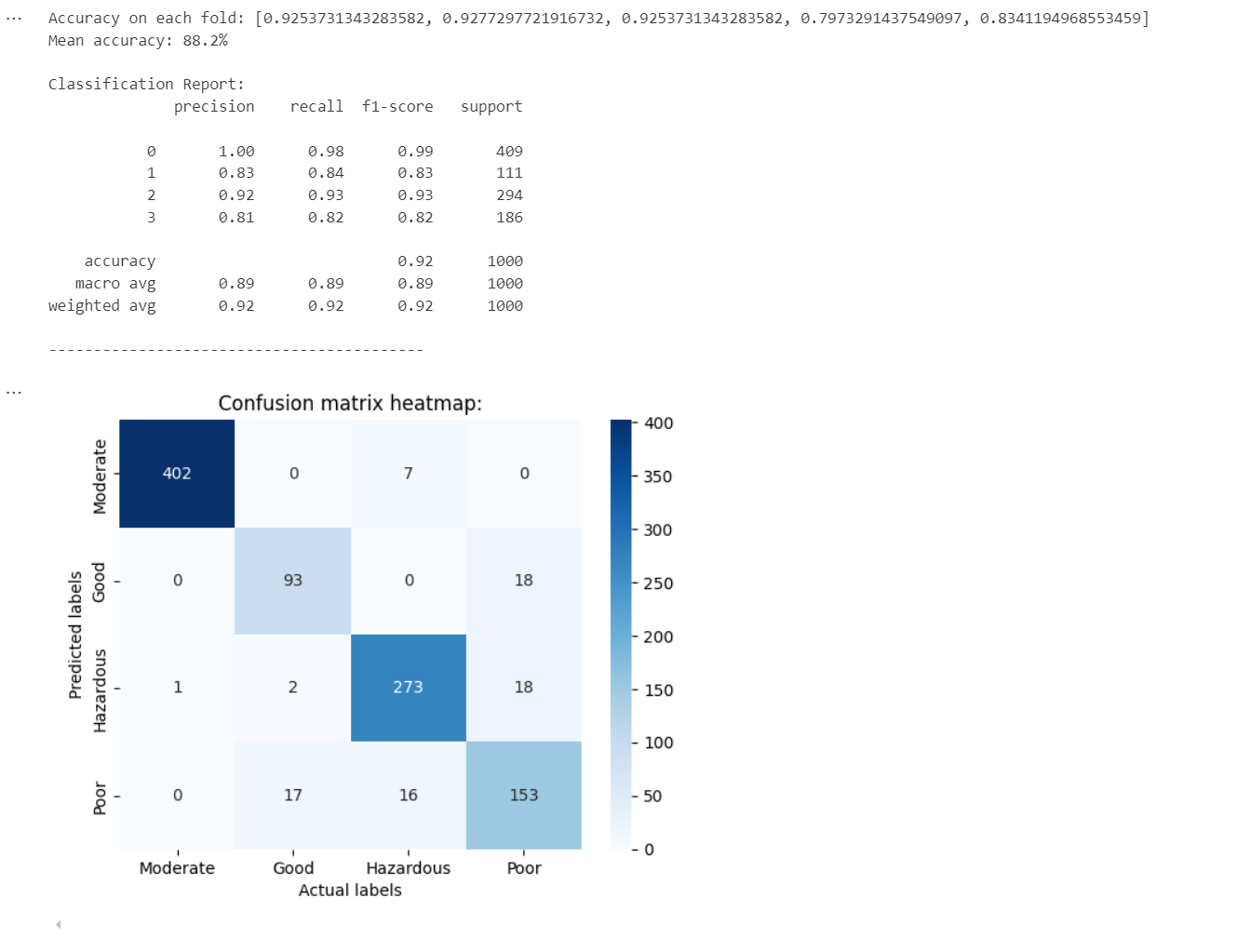
Hình 3.29: Hiệu suất KNN với Smote

**Mã nguồn mô hình Naive Bayes**



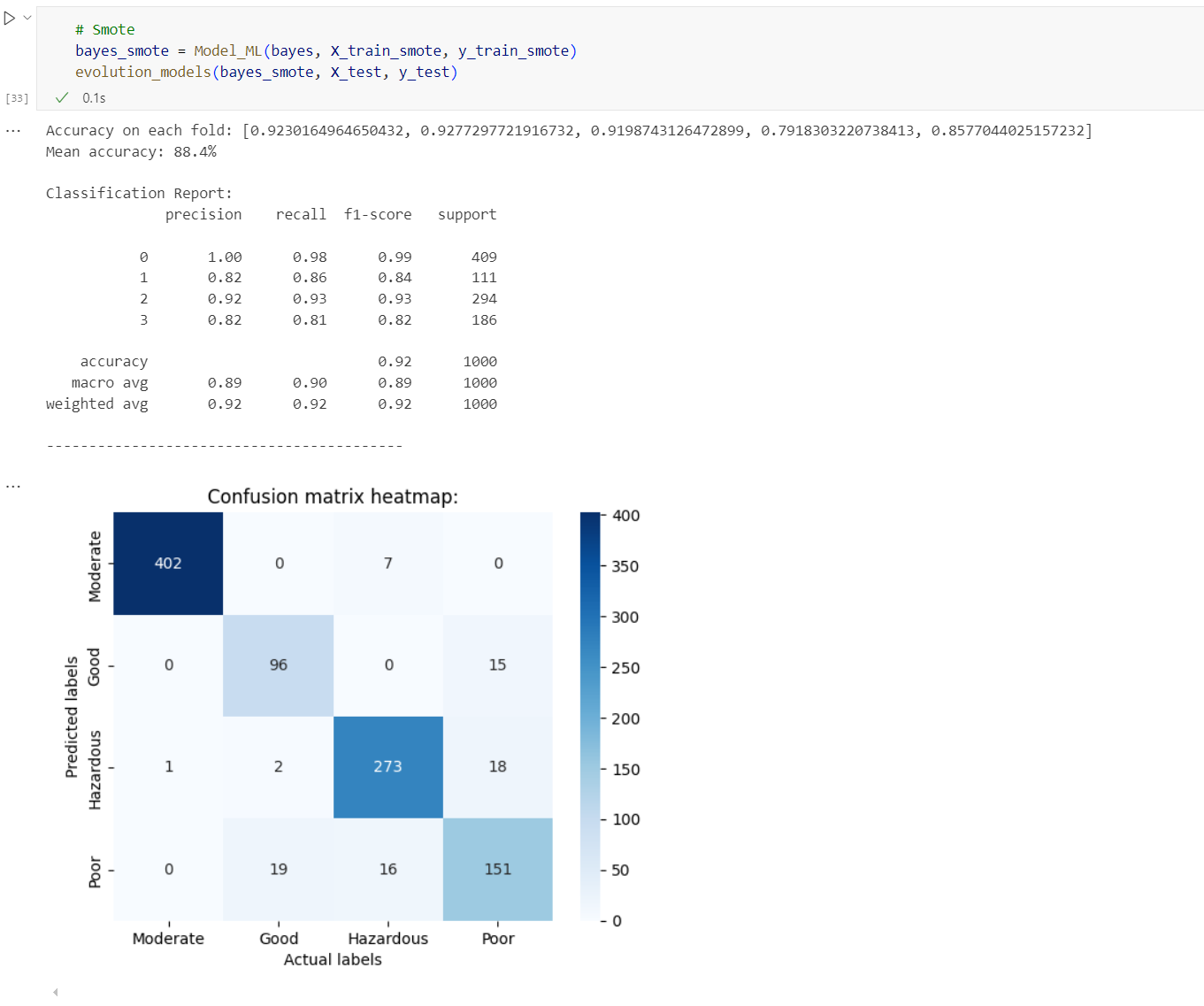
Hình 3.30: Mã nguồn Naive Bayes model

**Đánh giá mô hình với cân bằng dữ liệu random oversampler**



Hình 3.31: Hiệu suất Naive Bayes với Random Oversampler

**Đánh giá mô hình với cân bằng dữ liệu smote**



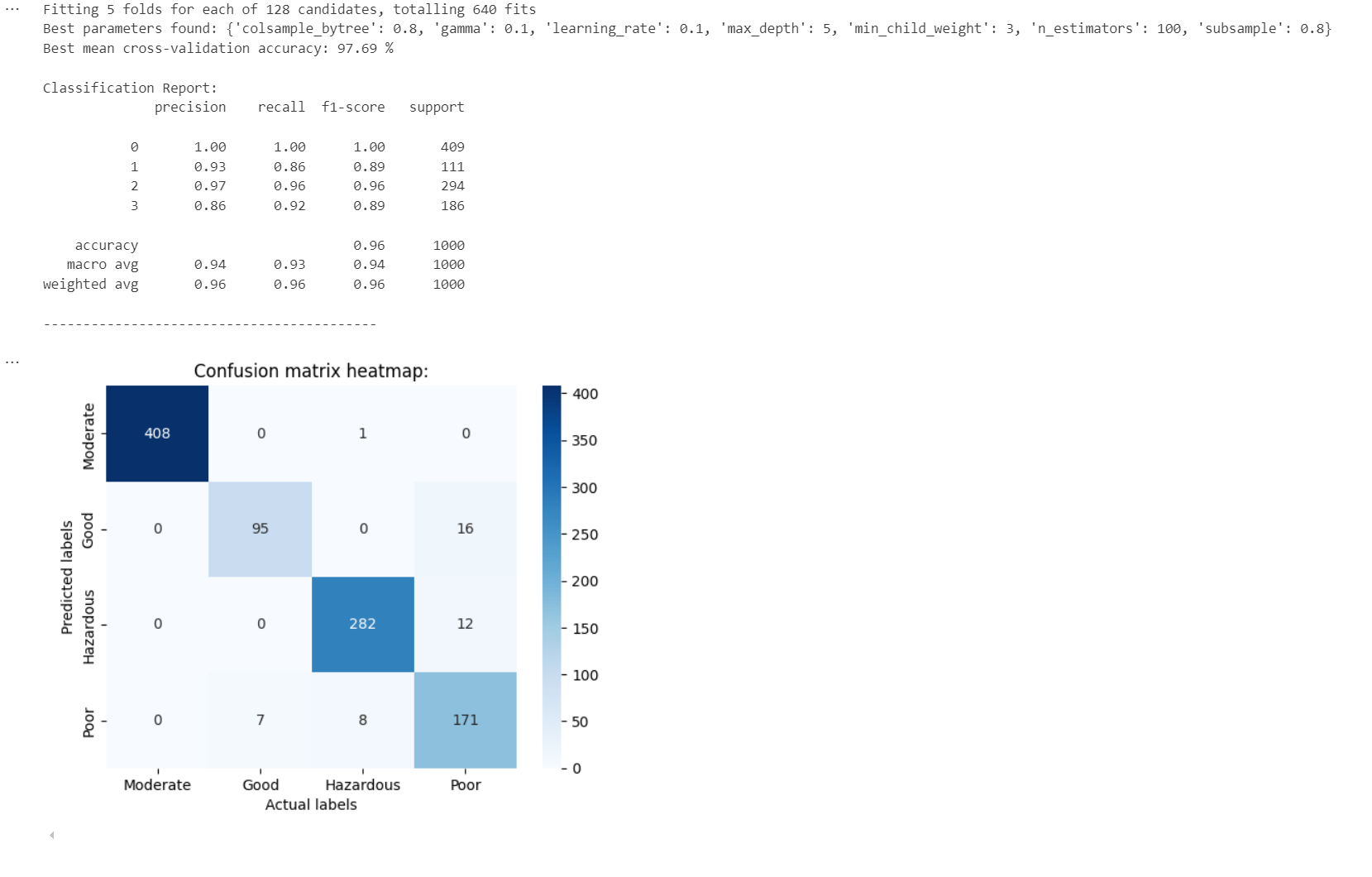
Hình 3.32: Hiệu suất Naive Bayes với Smote

**Mã nguồn mô hình XGBoost**



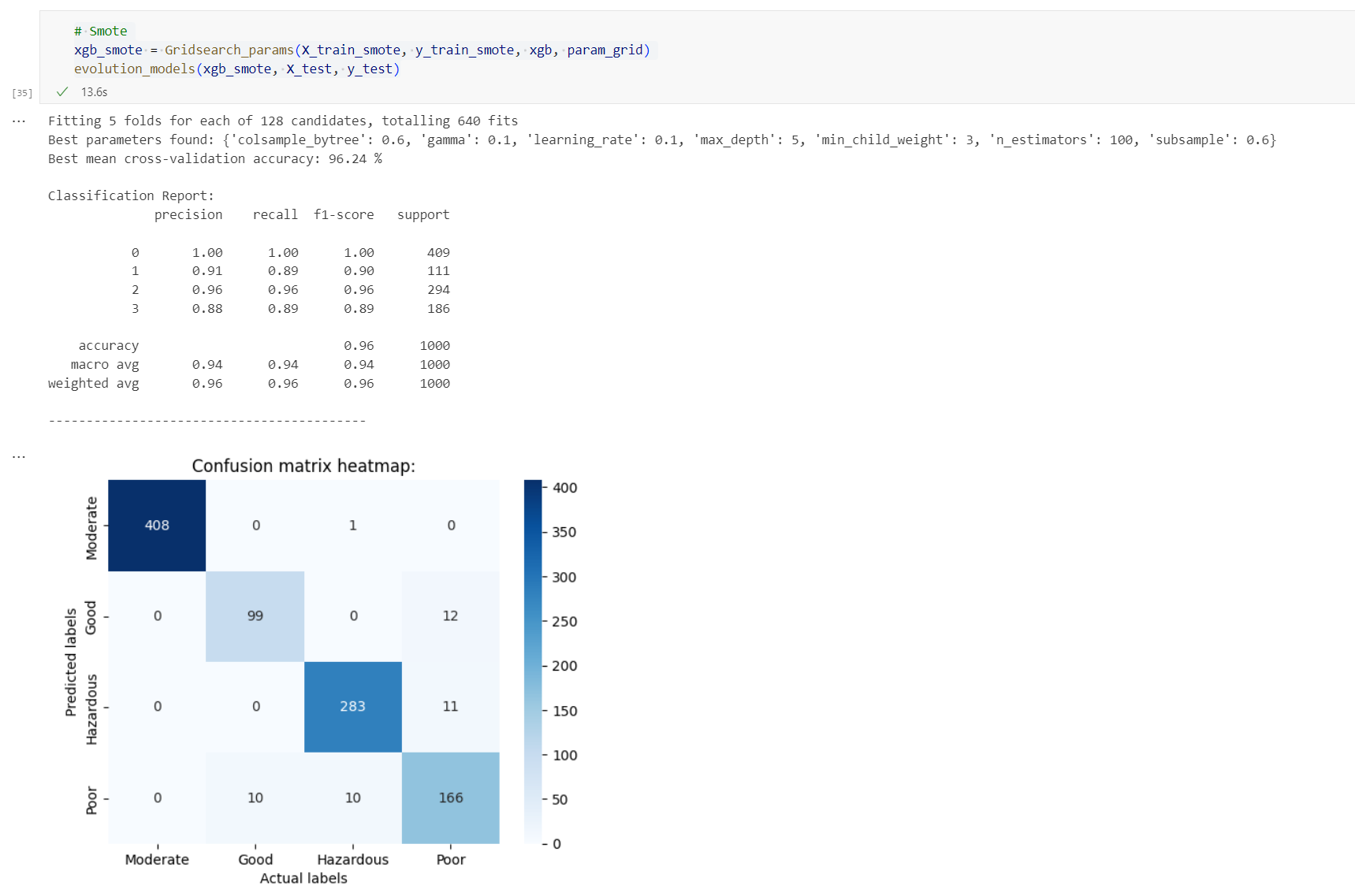
Hình 3.33: Mã nguồn XGBoost model

**Đánh giá mô hình với cân bằng dữ liệu random oversampler**



Hình 3.34: Hiệu suất XGBoost với Random Oversampler

**Đánh giá mô hình với cân bằng dữ liệu smote**



Hình 3.35: Hiệu suất XGBoost với Smote

# KẾT LUẬN

1. **Thành tựu:**

* Dự án đã cải thiện đáng kể việc phân loại chất lượng không khí bằng cách sử dụng các mô hình học máy như Random Forest và XGBoost. Những mô hình này cho thấy hiệu quả cao trong việc phân tích dữ liệu môi trường, từ đó phân loại chính xác các mức độ chất lượng không khí.
* Việc tích hợp AI vào các hệ thống giám sát môi trường đã giúp các cơ quan quản lý đưa ra cảnh báo và biện pháp phòng ngừa nhanh chóng, từ đó giảm thiểu rủi ro sức khỏe cộng đồng.
* Khả năng xử lý và phân tích các bộ dữ liệu lớn, bao gồm nhiệt độ, độ ẩm, nồng độ các chất gây ô nhiễm, và các yếu tố khác, đã nâng cao khả năng dự đoán và cung cấp thông tin chi tiết cho các chính sách bảo vệ môi trường.

1. **Hạn chế:**

* **Hạn chế dữ liệu:** Thiếu các bộ dữ liệu quy mô lớn, toàn diện và được chú thích tốt về chất lượng không khí, đặc biệt ở các khu vực nông thôn hoặc các quốc gia có hệ thống giám sát kém phát triển.
* **Khả năng tổng quát hóa:** Các mô hình được huấn luyện trên dữ liệu từ một khu vực cụ thể có thể không hoạt động tốt ở các vùng khác với điều kiện khí hậu hoặc nguồn ô nhiễm khác biệt.

1. **Hướng phát triển:**

* **Cải thiện dữ liệu:** Cần thu thập và tích hợp các bộ dữ liệu đa dạng hơn, bao gồm dữ liệu thời gian thực từ các cảm biến môi trường, dữ liệu khí hậu, và nghiên cứu theo thời gian dài, nhằm tăng cường khả năng áp dụng của mô hình.
* **Tăng độ chính xác:** Áp dụng các kỹ thuật học máy tiên tiến như học sâu (deep learning) và học chuyển giao (transfer learning) để nâng cao độ chính xác và khả năng phát hiện các dấu hiệu ô nhiễm ở giai đoạn sớm.

# TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | Dataset link: *https://www.kaggle.com/datasets/mujtabamatin/air-quality-and-pollution-assessment/data* |
| [2] | Slide của thầy Nguyễn Văn Quyết |