

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**KHOA ……….**

-----&-----

**Nhận dạng khuyết tật tấm**

**kim loại dựa trên mạng ANN**

***Sinh viên thực hiện:***

* ***Trần Hữu Đạt*** ***B21DCCN221***
* ***Hoàng Đức Huyên***  ***B21DCCN***

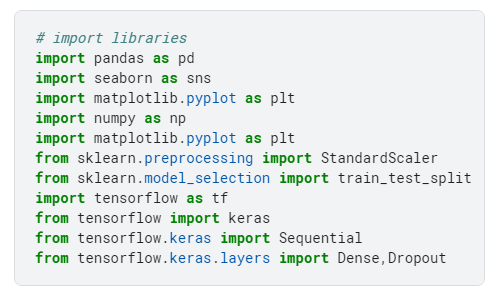
**Hà Nội, 2024**

1. Giới thiệu

“Steel Plate Defect Prediction” là một lĩnh vực quan trọng trong công nghiệp sản xuất thép, nhằm phát hiện và dự đoán sớm các lỗi trên đĩa, ngăn chặn hư hỏng tiếp theo và giảm chi phí sửa chữa. Điều này đóng vai trò quan trọng trong việc đảm bảo chất lượng sản phẩm thép và tiết kiệm chi phí cho các nhà sản xuất.

Nguồn tham khảo: <https://www.kaggle.com/code/akhilpm1996/steel-plate-defect-prediction-ann-h2oautoml?rvi=1>

1. Huấn luyện mạng ann để tạo mô hình nhận lỗi trên tấm thép
2. Nhập các thư viện và hàm





* Pandas:
  + Pandas cung cấp một loạt các công cụ và cấu trúc dữ liệu, chủ yếu là DataFrame.
  + Thư viện này chuyên dùng để thực hiện các công việc liên quan đến khoa học dữ liệu và phân tích dữ liệu.



* Seaborn:
  + Được xây dựng trên thư viện Matplotlib. Nó cung cấp giao diện để vẽ đồ họa thống kê hấp dẫn với đầy đủ thông tin.
  + seaborn thường được dùng để trực quan hóa dữ liệu



* Matplotlib.pyplot:
  + pylot cung cấp một giao diện trạng thái (state-machine interface) tới thư viện vẽ đồ thị cơ bản trong matplotlib.
  + Nó giúp tạo ra các đồ thị phức tạp một cách dễ dàng hơn.



* Numpy:
  + numpy cung cấp nhiều cấu trúc dữ liệu mạnh mẽ khác nhau nhằm mục đích triển khai các ma trận và mảng đa chiều.
  + Các cấu trúc dữ liệu này được sử dụng cho các tính toán tối ưu liên quan đến mảng và ma trận.



* StandardScaler:
  + Nhập từ sklearn.preprocessing. Đây là công cụ tiền xử lý dữ liệu,
  + nó loại bỏ giá trị trung bình và chia tỷ lệ để đạt đến phương sai đơn vị.
  + Hàm này được sử dụng để tránh trường hợp các đặc trưng không tuân theo phân phối chuẩn



* train\_test\_split
  + Nhập từ sklearn.model\_selection
  + Hàm này giúp chia dữ liệu thành 2 tập con
  + Tập huấn luyện: sẽ giúp huấn luyện mô hình, Tập kiểm tra: sẽ giúp kiểm tra hiệu suất của mô hình



* Tensorflow
  + tensorflow được sử dụng rộng rãi trong học máy và học sâu để xây dựng và huấn luyện các mô hình
  + Tensorflow giúp: tính toán với tensor, tính toán song song và phân tán, xây dựng và huấn luyện mô hình học sâu



* Keras
  + Là modul của tensorflow
  + Là api cao cấp để xây dựng và huấn luyện các mô hình trong tensorflow



* Sequential
  + Là một mô hình trong tensorflow
  + Sequential là mô hình mạng neuron theo kiểu chồng chất, có thể thêm các lớp một cách tuần tự
  + Mạng neuron kiểu chồng chất sẽ hữu ích khi chỉ cần 1 mạng neuron đơn giản và không có các liên kết phức tạp



* Dense:
  + Là 1 lớp của tensorflow.keras.layers
  + mỗi nơ-ron trong lớp i được kết nối với tất cả nơ-ron trong lớp i-1 và lớp i+1



* Dropout
  + Là 1 lớp của tensorflow.keras.layers
  + ngẫu nhiên bỏ qua một số nơ-ron trong quá trình huấn luyện, giúp mô hình không phụ thuộc quá mức vào bất kỳ nơ-ron cụ thể nào

1. Nhập dữ liệu

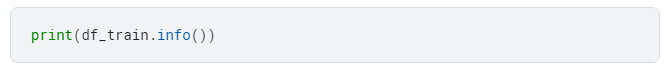
Đọc dữ liệu và hiển thị 5 hàng đầu tiên





1. Đánh giá dữ liệu

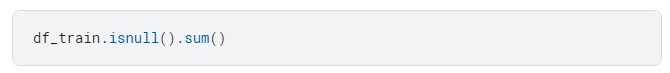
In ra thông tin của df\_train gồm tên các cột, số cột không null, kiểu dữ liệu



Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, tài liệu

Mô tả được tạo tự động

Tính tổng số giá trị null trong mỗi cột của df\_train

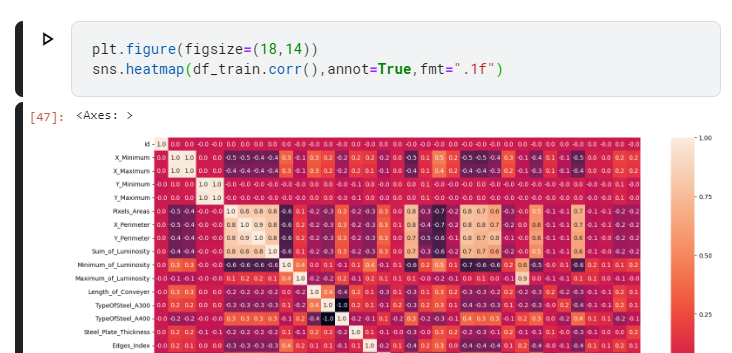




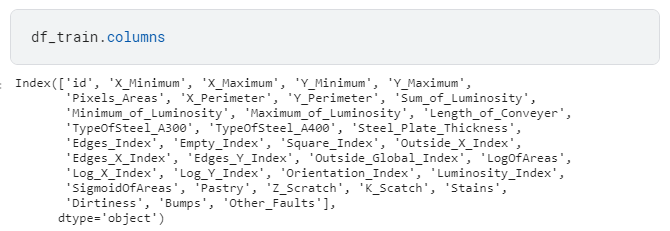
Tính tổng số các dòng bị trùng lặp trong df\_train



Vẽ biểu đồ nhiệt biểu hiện mối tương quan giữa các cột trong df\_train



In ra tên các cột của df\_train

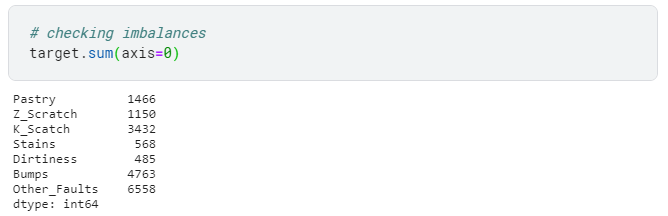


1. Chuẩn hóa dữ liệu

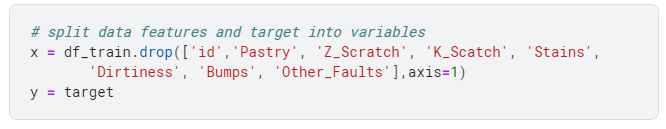
Chia các cột nhãn ra khỏi df\_train và gán vào biến target, (để chuẩn bị cho việc huấn luyện)



Tính tổng của mỗi cột trong DataFrame target, kiểm tra sự mất cân bằng

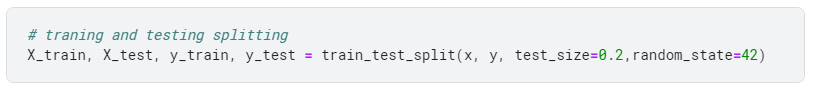


Chia dữ liệu thành hai tập: tập đặc trưng và tập mục tiêu.



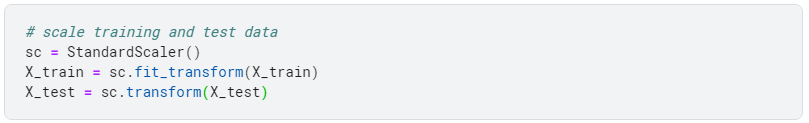
* x= tập đặc trưng, y= tập mục tiêu

Chia tiếp thành 2 tập: tập huấn luyện và tập kiểm tra



* X\_train: tập đặc trưng dùng để huấn luyện
* y\_train: tập mục tiêu dùng để huấn luyện
* X\_test: tập đặc trưng dùng để kiểm tra
* y\_test: tập mục tiêu dùng để kiểm tra
* test\_size= 0,2 : 20% dữ liệu sẽ dành cho việc kiểm tra
* random\_state= 42: đảm bảo tính nhất quán khi chia dữ liệu

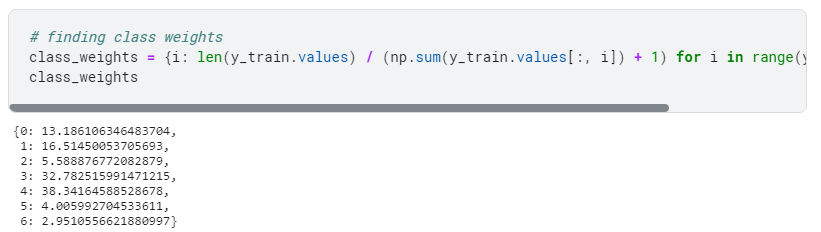
Tính toán giá trị trung bình, độ lệch chuẩn của X\_train, sau đó chuẩn hóa X\_train



Trả về các tuple kích thước (số hàng, số cột) của X\_train, X\_test, y\_train, y\_test



Tạo một từ điển class\_weights trong đó khóa là chỉ số của lớp và giá trị là trọng số của lớp đó.



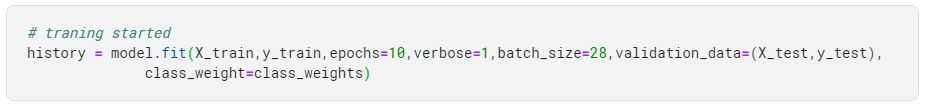
* Trọng số= (tổng số mẫu huấn luyện)/(số mẫu thuộc lớp đó)+1
  + Trọng số lớp có thể được sử dụng trong quá trình huấn luyện mô hình học máy để cân bằng dữ liệu,
  + khi có sự mất cân đối giữa các lớp. Trọng số lớp cao hơn sẽ cho mô hình biết rằng nó nên tập trung hơn vào các mẫu thuộc lớp đó.

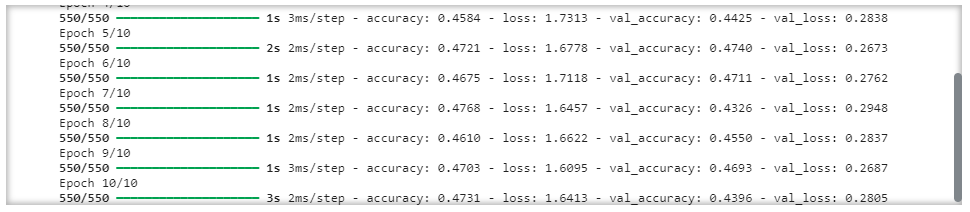
1. Xây dựng mô hình



* model = Sequential([...]): khởi tạo một mô hình Sequential. Mô hình Sequential xếp các lớp nơ-ron lên nhau theo thứ tự tuần tự.
* Các lớp nơ-ron:
  + Dense(256, activation='relu', input\_dim=27): lớp nơ-ron đầu tiên với 256 nơ-ron, hàm kích hoạt là ReLU (Rectified Linear Unit), và 27 đầu vào (input features).
  + Dropout(0.2): Lớp Dropout với tỷ lệ 0.2. Lớp này giúp ngăn overfitting bằng cách loại bỏ ngẫu nhiên một phần nơ-ron trong quá trình huấn luyện.
  + Dense(128, activation='relu'): Lớp nơ-ron thứ hai với 128 nơ-ron và hàm kích hoạt ReLU.
  + Dropout(0.2): Lớp Dropout khác với tỷ lệ 0.2.
  + Dense(64, activation='relu'): Lớp nơ-ron thứ ba với 64 nơ-ron và hàm kích hoạt ReLU.
  + Dropout(0.2): Lớp Dropout cuối cùng với tỷ lệ 0.2.
  + Dense(7, activation='sigmoid'): Lớp nơ-ron cuối cùng với 7 nơ-ron và hàm kích hoạt Sigmoid. Đây là một mô hình phân loại đa nhãn (multi-label classification) với 7 lớp đầu ra.
* model.compile(...): Dòng này cấu hình mô hình với optimizer là Adam, hàm mất mát là binary\_crossentropy (phù hợp cho phân loại đa nhãn), và độ đo hiệu suất là accuracy.

1. Biên dịch mô hình



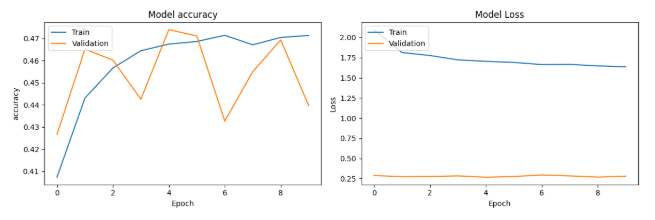


* history = model.fit(...): huấn luyện mô hình. Nó thực hiện đào tạo mô hình trên tập dữ liệu huấn luyện (X\_train, y\_train). Các tham số được sử dụng:
  + epochs=10: Số lần lặp qua toàn bộ dữ liệu huấn luyện.
  + verbose=1: Chế độ hiển thị thông tin trong quá trình huấn luyện (1: hiển thị, 0: không hiển thị).
  + batch\_size=28: Kích thước của các batch dữ liệu được sử dụng trong mỗi lần cập nhật mô hình.
  + validation\_data=(X\_test, y\_test): Dữ liệu kiểm tra để đánh giá hiệu suất mô hình trong quá trình huấn luyện.
  + class\_weight=class\_weights: Trọng số của từng lớp được sử dụng để cân bằng mẫu dữ liệu.
  + history: Biến lưu trữ thông tin về quá trình huấn luyện, bao gồm các giá trị mất mát và độ chính xác trong từng epoch.

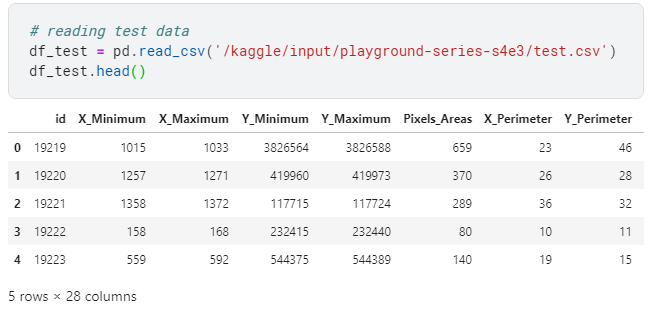
1. Đánh giá kết quả huấn luyện

Vẽ biểu đồ mô tả về độ chính xác và độ mất mát của việc huấn luyện



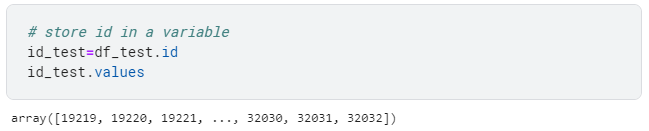


1. Đọc dữ liệu kiểm tra và in



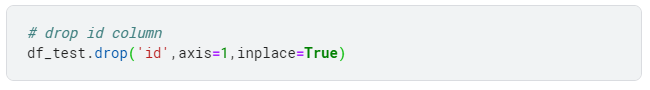
1. Chuẩn hóa dữ liệu kiểm tra

Lưu id của dữ liệu kiểm tra trong 1 biến và in giá trị của chúng

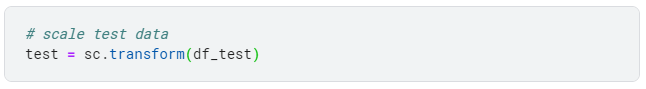


* (giúp cải thiện hiệu suất và đảm bảo tính hợp lý trong việc đánh giá)

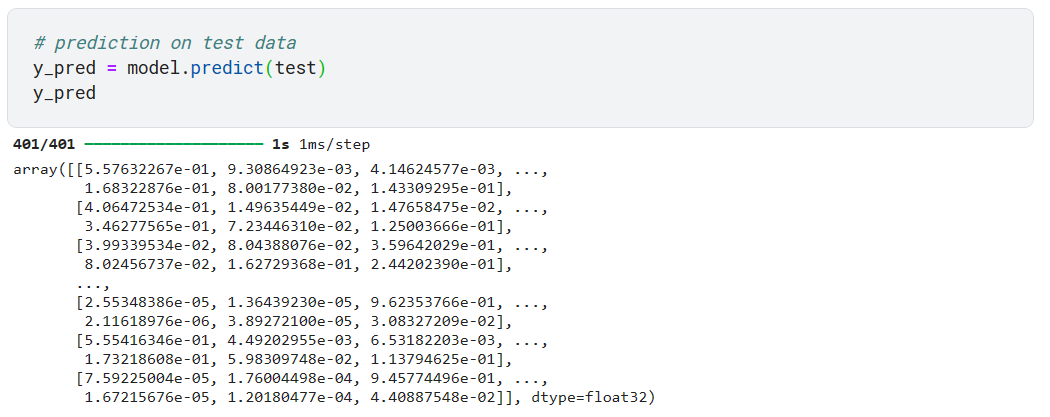
Xóa cột id khỏi df\_test



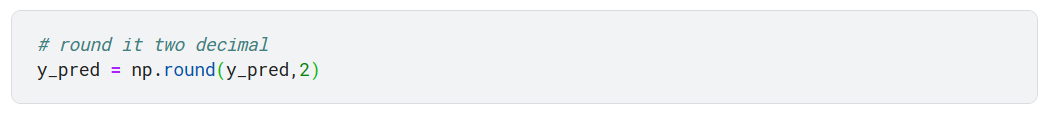
Chuẩn hóa df\_test



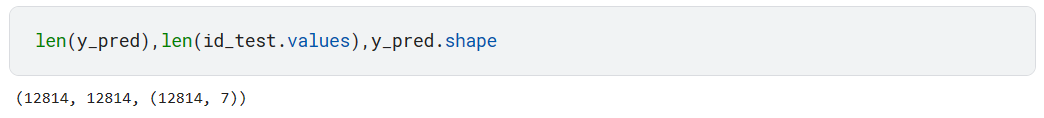
1. Dự đoán kết quả từ tập dữ liệu kiểm tra, sau đó in ra kết quả



1. Làm tròn kết quả dự đoán trong 2 chữ số thập phân

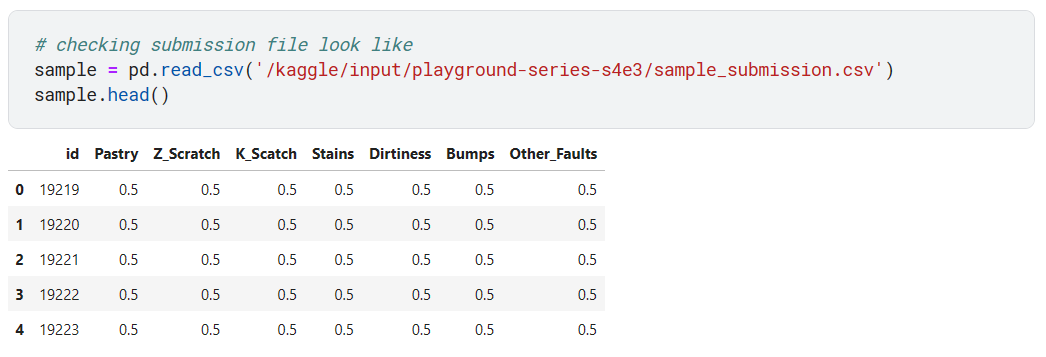


1. In ra độ dài của y\_pred, id\_test.values, và kích thước của y\_pred



1. Nộp dữ liệu

Đọc dữ liệu từ tệp mẫu và lưu nó vào biến sample, hiển thị 5 hàng đầu tiên của tệp mẫu

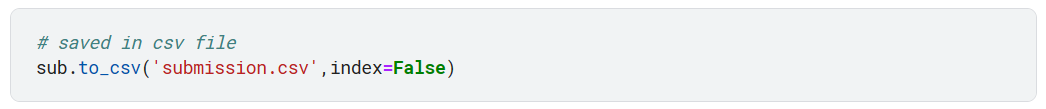


Lưu dataframe kết quả vào biến sub, hiển thị 5 hàng đầu tiên của nó



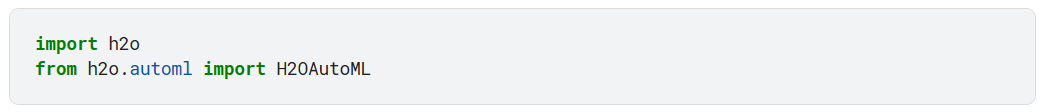
* Tạo dataframe với các cột: id từ id\_test, tên các lỗi từ y\_pred (Pastry, Z\_Scratch, K\_Scatch, Stains, Dirtiness, Bumps, Other\_Faults)
* Và gán nó vào biến sub

Lưu kết quả vào file submission



1. Sử dụng H2O autoML để tự động tinh chỉnh mô hình học máy
2. Nhập thư viện, khởi tạo phiên làm việc

Nhập thư viện



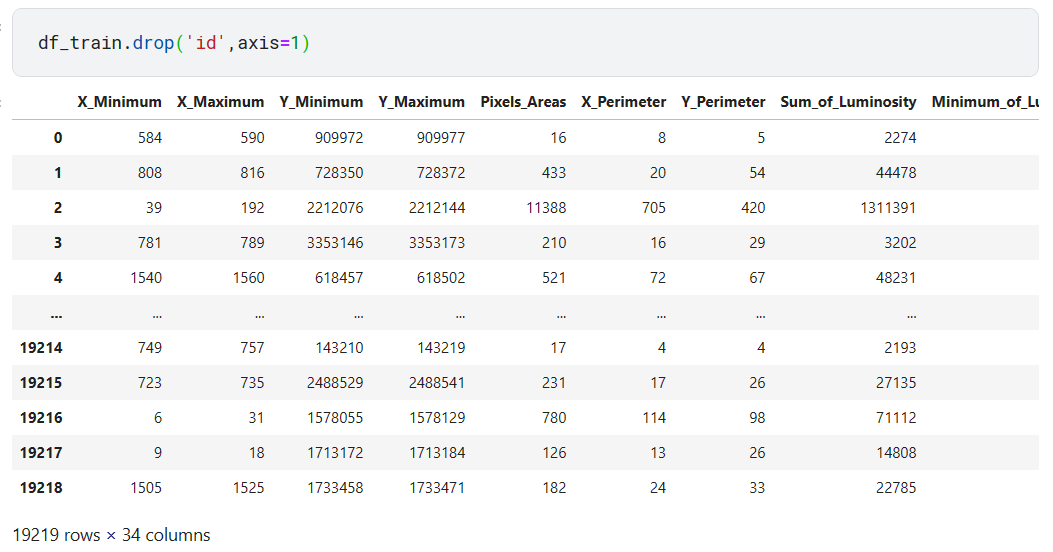
* H2O là một thư viện mã nguồn mở dành cho học máy và khai phá dữ liệu.
* H2O AutoML giúp tự động tìm ra mô hình tốt nhất cho bài toán mà không cần phải tinh chỉnh thủ công các siêu tham số. Các tính năng
  + Tự động tạo ra một loạt các mô hình máy học khác nhau từ dữ liệu đầu vào, người dùng không cần phải thử nghiệm từng thuật toán một.
  + Tự động điều chỉnh tham số của mô hình để đạt hiệu suất tốt nhất.
  + Tự động chọn mô hình tốt nhất dựa trên độ đo hiệu suất (ví dụ: độ chính xác, AUC, RMSE).

khởi tạo phiên làm việc với H2O cluster cục bộ



1. Chuẩn hóa dữ liệu

Loại bỏ cột id khỏi data frame df\_train



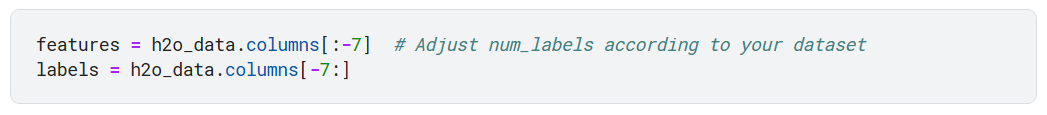
* Cải thiện hiệu suất và đảm bảo tính hợp lý khi đánh giá mô hình

Loại bỏ cột id khỏi df\_train và dùng nó tạo nên H2OFrame h2o\_data

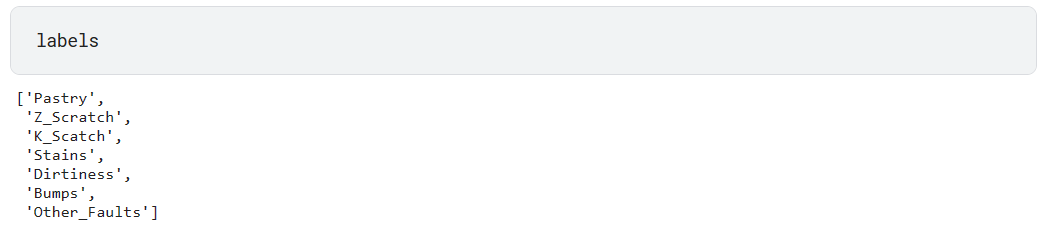


* H2OFrame là một cấu trúc dữ liệu tương tự DataFrame của thư viện Pandas
* H2OFrame không lưu dữ liệu trong bộ nhớ mà tham chiếu đến một cụm H2O có thể ở nơi khác (có thể cách xa)

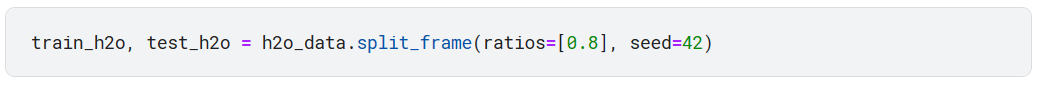
Chia 7 cột cuối cùng thành H2OFrame labels, các cột còn lại thành H2OFrame features



In các cột của H2OFrame labels

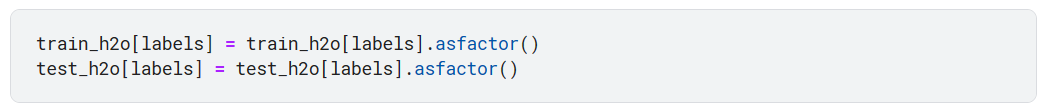


Chia dữ liệu từ h2o\_data, thành 2 phần train\_h2o (tập huấn luyện), test\_h2o (tập kiểm tra)



* ratios: tỉ lệ phân chia dữ liệu huấn luyện/kiểm tra= 0.8/0.2
* seed: đảm bảo dữ liệu phân chia là nhất quán dù có nhiều lần phân chia lại

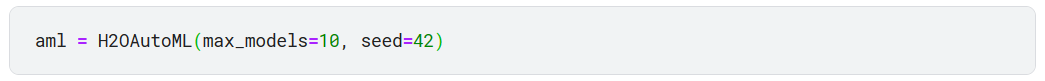
Chuyển đổi nhãn thành kiểu factor



* Khi chuyển đổi nhãn sang kiểu factor, H2O sẽ tự động tạo ra các chỉ mục cho nhãn dựa trên các giá trị duy nhất trong cột.

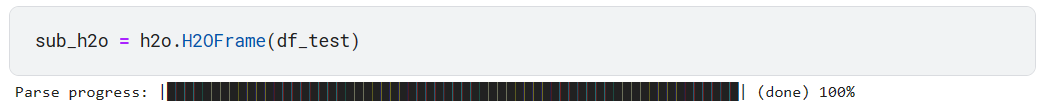
1. Huấn luyện mô hình

Huấn luyện và tinh chỉnh tối đa 10 mô hình khác nhau, seed= 42, lưu kết quả vào biến aml



* seed=42, đảm bảo tính nhất quán khi chạy lại
* biến aml, chứa các thông tin về các mô hình huấn luyện và tinh chỉnh

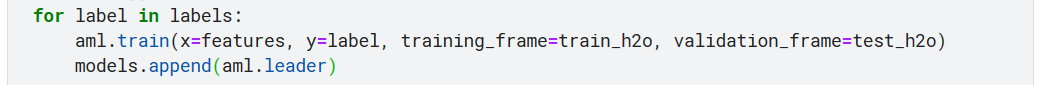
Tạo H2OFrame sub\_h2o từ biến df\_test



Tạo biến models rỗng để lưu các mô hình huấn luyện



Với mỗi label trong mảng labels, huấn luyện mô hình dựa trên H2O AutoML



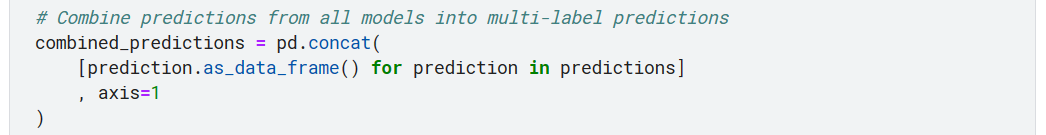
* aml.train(...): huấn luyện mô hình học máy tự động bằng H2O AutoML,
  + X=features, danh sách các biến đầu vào (đặc trưng) mà mô hình sẽ sử dụng để dự đoán.
  + y=label: biến mục tiêu mà mô hình sẽ dự đoán.
  + training\_frame= train\_h2o: Dữ liệu huấn luyện được sử dụng để xây dựng mô hình.
  + validation\_frame= test\_h2o: Dữ liệu kiểm tra dùng để đánh giá hiệu suất mô hình.
* models.append(aml.leader) : thêm mô hình tốt nhất của quá trình huấn luyện vào biến models

Biến predictions chứa các dự đoán từ tất cả các mô hình.

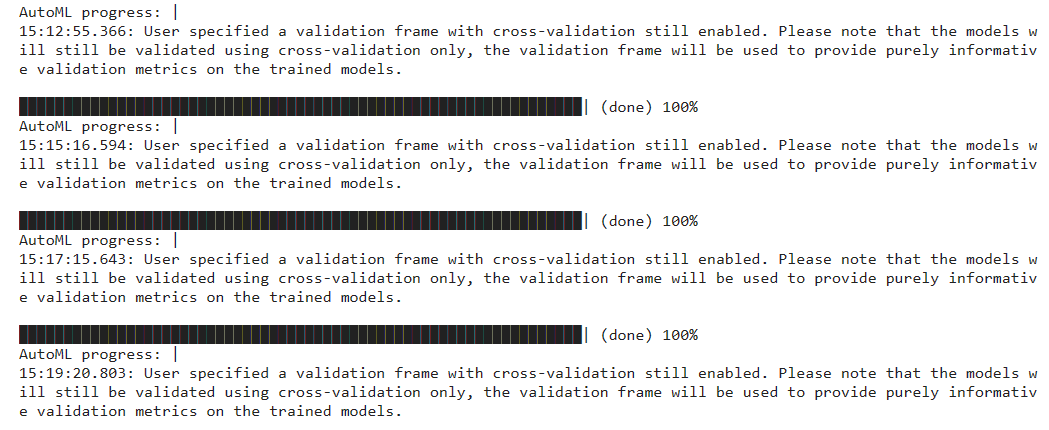


* Với mỗi mô hình trong biến models, gọi hàm predict(sub\_h2o), tạo ra dự đoán cho từng nhãn
* Kết quả được lưu vào biến predictions

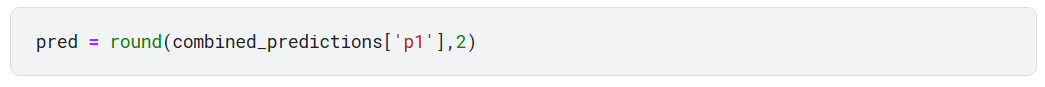
Kết hợp dữ đoán từ tất cả các mô hình thành dự đoán nhiều nhãn



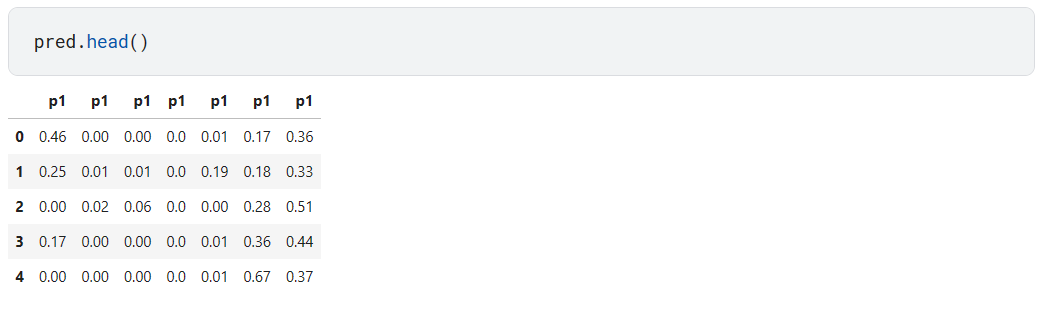
* Với mỗi phần từ trong “predictions”, chuyển chúng thành dataFrame .
  + 
* Kết hợp tất cả các dataFrame thành 1 dataFrame duy nhất với lệnh concat
  + Việc kết hợp được thực hiện theo cột “axis=1”



Lưu vào biến pred là kết quả combined\_prediction làm tròn trong 2 chữ số thập phân

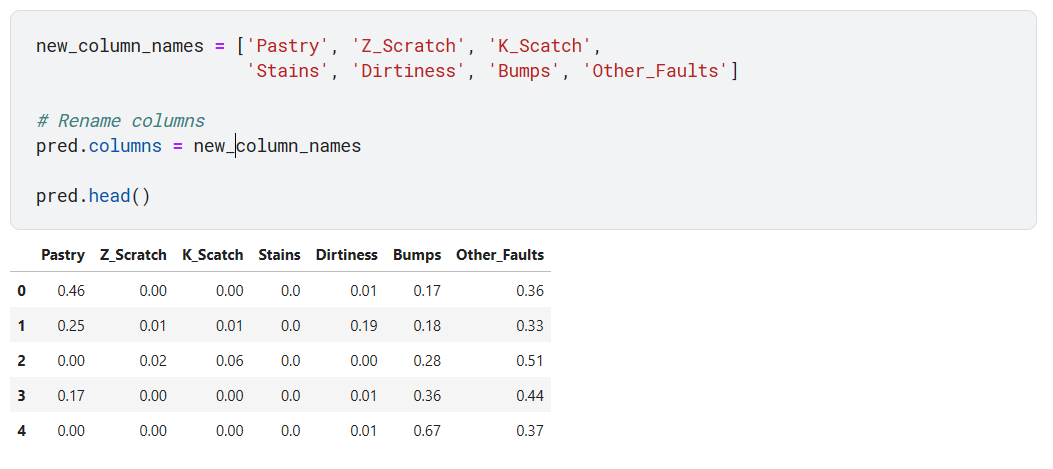


hiển thị 5 hàng đầu tiên của biến pred

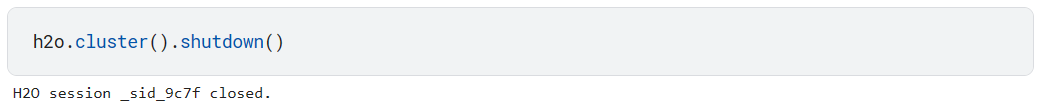


1. Kết thúc và nộp dữ liệu

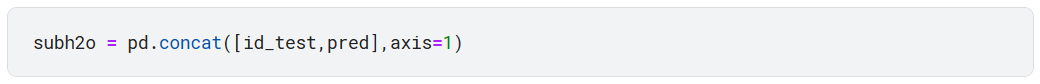
Tạo mảng chứa tên mới cho các cột, đổi tên của các cột, in 5 hàng đầu tiên của pred



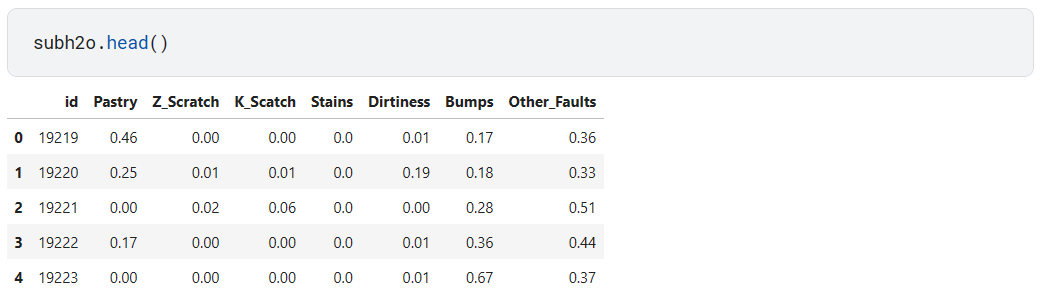
Tắt h2o\_cluster cục bộ



Kết hợp dataFrame id\_test và pred theo chiều ngang thành dataFrame mới, gán nó vào biến subh2o



In 5 hàng đầu của biến subh2o



Lưu biến subh2o vào file submission.csv

