Tuần 1: Giới thiệu về machine learning:

Trong thế giới hiện đại, đặc biệt là tại thời đại công nghệ thông tin phát triển như hiện tại. Có rất nhiều các hệ thống như các hệ thống tự động hóa, robot,… ứng dụng machine learning. Mọi người khác có thể biết được các ứng dụng nhưng không biết đến thuật ngữ machine learning. Còn chúng ta, đặc biệt là những người học và hoạt động trong ngành CNTT thì việc nghe qua thuật ngữ này hẳn là rất bình thường, không những vậy còn được nghe rất thường xuyên và còn được học và tìm hiểu về nó nữa. Vậy machine learning là gì? Machine learning là một lĩnh vực nhánh của Trí tuệ nhân tạo (Artificial Intelligence), sử dụng thuật toán cho phép máy tính có thể học từ dữ liệu có sẵn (do việc thu thập dữ liệu) để thực hiện các công việc một cách tự động thay vì lập trình rõ rang cách giải. Có nghĩa là chúng ta sẽ lập trình cho các máy tính có khả năng tự học.

Lấy một số ví dụ để thấy rõ ứng dụng của machine learning:

* Recommendation của Youtube: khi chúng ta tìm kiếm hoặc xem vài video liên quan đến chủ đề nào đó thì lần sau khi vào youtube chúng ta sẽ thấy hang loạt các video có nội dung hoặc chủ đề liên quan được gợi ý ở trang chủ.
* Lọc thư rác trên mail: phân loại các thư không có chủ đề, các thư quảng cáo hoặc các thư chứa các tệp virus, mã độc cho vào thư rác.

Nói về ứng dụng đầu tiên của machine learning ta có thể nói về ứng dụng cho máy tính học cách chơi và các chiến thuật đánh trong cờ vây của Samuel 1959.

Khi nào chúng ta nên sử dụng machine learning:

* Tập dữ liệu cần xử lý quá lớn.
* Dữ liệu liên quan đến vấn đề cần xử lý cập nhật liên tục.
* Các vấn đề cần xử lý phức tạp và môi trường hoạt động thay đổi liên tục.

Dựa vào các điều kiện khác nhau ta có thể chia machine learning ra làm 3 loại:

* Có sự giám sát của con người hoặc không (with human supervision or not): có một số thuật toán thuộc nhóm này như: supervised learning, unsupervised learning, semisupervised learning, reinforcement learning
* Có vừa học vừa làm hoặc không (learn on the fly or not): một số thuật toán thuộc nhóm này là online learning, batch learning.
* Có sử dụng model hoặc không: một số thuật toán thuộc nhóm này là: instance-based learning, model-based learning.

Đi sâu vào từng vấn đề của mỗi loại, đầu tiên ta đến với supervised learning:

* Supervised learning là loại thuật toán xử lý trên tập dữ liệu có dán dùng nhãn. Gồm 2 loại nhỏ là: classification và regression.
* Các thuật toán quan trọng nhất của supervised learning là: k-Nearest Neighbors, Linear Regression, Logictic Regression, Support Vector Machines (SVMs), Decision Trees and Random Forests, Neural Networks.

Phần tiếp theo là unsupervised learning:

* Unsupervised learning là loại thuật toán xử lý trên tập dữ liệu không có dán nhãn.
* Các thuật toán thuộc nhóm Unsupervised learning:

+ Clustering gồm các thuật toán quan trọng: K-Means, DBSCAN, Hierarchical Cluster Analysis (HCA).

+ Anomaly detection: One-class SVM, Isolation Forest.

+ Dimensionality reduction: Principal Component Analysis (PCA), Locally-Linear Embedding (LLE), t-distributed Stochastic Neighbor Embedding (t-SNE)

+ Association rule learning: Apriori, Eclat.

+ Reinforcement learning: tác nhân được dạy dựa trên điểm thưởng và điểm phạt.

Tiếp đến là learn on the fly:

- Batch learning: là đưa một lượng lớn dữ liệu vào (data training), máy sẽ học dựa trên tất cả dữ liệu được đưa vào và bắt đầu vận hành mà không học thêm bất cứ gì nữa, chỉ áp dụng những thứ đã học trong tập dữ liệu đầu vào. Muốn xử lý loại dữ liệu mới thì phải đưa tập dữ liệu mới vào và cho máy học lại, sau đó mới vận hành lại được. Nên batch learning tiêu tốn nhiều thời gian và tài nguyên. VD: Chương trình đánh cờ do máy đánh.

- Online learning: Tiếp nhận và học từng bước bằng cách đưa vào máy từng nhóm dữ liệu nhỏ. Máy sẽ học rất nhanh những dữ liệu đó và giải phóng dữ liệu để tránh tiêu tốn tài nguyên. Thích hợp với những hệ thống có nguồn tài nguyên hạn chế và dữ liệu liên tục cập nhật và đổi mới.

VD: Chương trình dự báo thời tiết dựa vào nhiệt độ và độ ẩm.

Cuối cùng là Use model

* Instance-based learning (Học dựa trên ví dụ): so sánh dữ liệu mới đưa vào với các dữ liệu cũ đã được học bằng cách sử dụng thước đo sự tương đồng giữa các giá trị cũ và mới.
* Model-based learning (Học dựa trên mô hình): từ những gì đã học được trên tập dữ liệu mà xây dựng nên một mô hình.

Tuần 2: Những thách thức chính của machine learning: dữ liệu không tốt (bad data), thuật toán không tốt (bad algorithm)

1. Bad data
2. Sự thiếu hụt dữ liệu: Quá trình để cho máy học cách giải quyết một vấn đề cũng giống như cách con người chúng ta tiếp nhận và học các thông tin từ môi trường bên ngoài. Không phải chỉ cần đọc qua một lần hoặc học qua một lần mà đã có thể nhận ra và giải quyết các vấn đề liên quan được. Muốn nhìn nhận và giải quyết vấn đề chính xác chúng ta cần thời gian để học tập cũng như luyện tập và thực hành từ những thứ đã học được, dần dần sẽ có phản xạ để giải quyết vấn đề nhanh và chính xác. Vấn đề mà chúng ta cần ở đây không phải là thời gian, nhưng là lượng dữ liệu chúng ta cần tiếp thu để đủ giải quyết được vấn đề ấy. Đối với máy tính cũng vậy, chúng ta cần phải cung cấp một lượng dữ liệu đủ lớn để máy có thể học tập và đưa ra phán đoán, giải quyết vấn đề được đặt ra một cách chính xác nhất.
3. Dữ liệu huấn luyện không mang tính đại diện (Nonrepresentative training data)

Đối với loại dữ liệu ta có thể gọi nó là dữ liệu gây nhiễu. nếu có quá nhiều dữ liệu gây nhiễu thì máy sẽ học và cho ra một model phù hợp với dữ liệu đó nhất. Dẫn đến model đó sẽ khớp hơn với các dữ liệu gây nhiễu đó nhưng lại không khớp với các dữ liệu thông thường. Làm cho máy xử lý các dữ liệu thông thường thì bị lỗi trong khi đó là nguồn dữ liệu chính ta cần xử lý chứ không phải là vài dữ liệu gây nhiễu kia. Cho nên cách tốt nhất là chúng ta nên lọc các dữ liệu lỗi trước khi đưa vào cho máy để có thể huấn luyện máy hoạt động tốt nhất.

1. Thiếu dữ liệu (missing data): mỗi sample sẽ có một số features nhất định nhưng một số các samples bị khuyết mất một vài features dẫn đến việc huấn luyện máy không chính xác. Những samples này sẽ làm cho quá trình huấn luyện của máy sai lệch dẫn đến quá trình xử lý dữ liệu không chính xác. Ở đây chúng ta có nhiều cách để khắc phục lỗi này, nếu số lượng dữ liệu của chúng ta không đủ lớn thì chúng ta phải điền vào chỗ các features bị khuyết (có thể điền vào số 0 hoặc là số trung bình của feature này ở một số samples xung quanh,…). Nếu số lượng dữ liệu của chúng ta đủ lớn thì ta có thể lọc và xóa đi hết tất cả các dữ liệu bị khuyết features này.
2. Features không liên quan (Irrelevant features): hệ thống của chúng ta chỉ có thể học được nếu dữ liệu huấn luyện chứa đầy đủ các features thích hợp và không chứa quá nhiều features không liên quan. Một phần quan trọng trong thành công của một dự án Học máy là tập đầy đủ các features đủ tốt để huấn luyện. Quá trình lọc, chỉnh sửa dữ liệu để huấn luyện cho máy này được gọi là các kỹ thuật xử lý Feature, bao gồm:

* Feature selection: chọn ra các features hợp lý trong tất cả các features tồn tại để thực hiện quá trình huấn luyện.
* Feature extraction: gộp các features đang có để tạo thành các thành features hữu ích hơn.
* Tạo ra các features mới thông qua việc thu thập thêm dữ liệu để huấn luyện.

1. Bad Algorithm
2. Overfitting: Hiện tượng overfitting là hiện tượng phổ biến diễn ra trong quá trình huấn luyện cho máy. Overfitting xảy ra khi chúng ta cố gắng làm cho model khớp nhất đối với tập dữ liệu huấn luyện. Dẫn đến model sau khi huấn luyện xong bậc quá lớn để có thể bám sát đến hết tất cả các dữ liệu trong tập dữ liệu (gồm cả các dữ liệu gây nhiễu). Kết quả là khi xử lý các dữ liệu bình thường ở khu vực dữ liệu nhiễu thì model không thể biểu diễn được nên máy sẽ xử lý và đưa ra kết quả bị sai, làm cho thuật toán bị sai.

Để giải quyết hiện tượng overfitting ta có một số cách sau:

* Đơn giản hóa model bậc cao vừa được tạo ra bằng cách chọn các tham số để giữ lại và giảm bậc của model lại.
* Giảm số lượng features có trong mỗi sample của dữ liệu huấn luyện, hoặc hạn chế model.
* Thu thập thêm dữ liệu phù hợp.
* Lọc bớt dữ liệu gây nhiễu.
* Đối với phương pháp Ràng buộc model (Constraining Model): đây là phương pháp làm cho model đơn giản hơn và giảm nguy cơ bị overfitting. Ví dụ nếu ta có thể đoán được model của chúng ta sẽ là model hình bậc 2 hoặc bậc 3 thì ta có thể giới hạn trước số lượng tham số của model và buộc cho model máy học ra phải thuộc hàm bậc mà chúng ta quy định. Cách này sẽ làm cho model của chúng ta đủ đơn giản để khái quát tập dữ liệu nhưng chúng ta cũng phải cần xem xét và cân nhắc để chọn số bậc của model phù hợp nhất để khát quát được toàn bộ tập dữ liệu của chúng ta.
* Một số lượng các quy tắc (regularization) được thêm vào trong suốt quá trình huấn luyện có thể được kiểm soát bởi một siêu tham số (hyperparameter). Một siêu tham số không phải là một model mà là một tham số của một thuật toán học tập. Nó không bị ảnh hưởng bởi các thuật toán học tập mà nó phải được thiết lập trước khi quá trình huấn luyện diễn ra và không thay đổi trong suốt quá trình đó. Nếu cài đặt một siêu tham số quá lớn thì độ dốc của mô hình gần như bằng không, thuật toán sẽ học ra được một model gần như không bị overfitting nhưng sẽ khó có thể tìm ra được một giải pháp tốt.

1. Underfitting: Hiện tượng underfitting trái ngược với hiện tượng overfitting. Underfitting diễn ra khi model của chúng ta quá đơn giản để biểu diễn tập dữ liệu của chúng ta. Làm cho máy sau khi được huấn luyện vẫn không thể thực hiện được các dự đoán một cách chính xác bởi vì model được học ra không đủ phức tạp để biểu diễn cho tập dữ liệu của chúng ta.

Để giải quyết vấn đề underfitting ta có những cách chính sau:

* Chọn một model tốt hơn, mạnh hơn với nhiều tham số hơn để huấn luyện cho máy .
* Cung cấp thêm các features tốt hơn cho thuật toán học tập.
* Giảm bớt các ràng buộc trên model (ví dụ: Giảm các siêu tham số chính quy).

1. Test set and generalization error.

* Tập dữ liệu của chúng ta nên được chia ra thành 2 phần, một phần là training set và một phần là test set. Chúng ta phải chia ra thành 2 phần bởi vì nếu ta đã dùng tập training set để huấn luyện cho máy mà lại dùng tập đó để kiểm tra thì chắc chắn tỉ lệ máy đoán đúng sẽ gần như là tuyệt đối, rất ít khi đoán sai. Nên ta phải chia tập dữ liệu ra thành 2 phần để có thể dùng tập test set kiểm tra xem quá trình máy học có đúng và áp dụng quá trình học để đoán các dữ liệu mới có đúng hay không.
* Tỉ lệ lỗi trên các trường hợp mới thì được gọi là generalization error. Bằng cách đánh giá model trên tập test set, ta có thể ước tính được lỗi này. Giá trị này sẽ cho ta biết model của ta hoạt động tốt như thế nào trên các trường hợp mà máy chưa gặp bao giờ.
* Nếu lỗi huấn luyện thấp thì có nghĩa là model của ta mắc một số lỗi trên tập dữ liệu huấn luyện, nhưng nếu lỗi generalization cao có nghĩa là model của ta đã bị overfitting

1. Hyperparameter Tuning and Model Selection

* Hyperparameter Tuning: có nghĩa là để tìm được một siêu tham số cho quá trình huấn luyện, ta không thể lấy bừa một con số được mà phải dò xem những siêu tham số nào phù hợp với model và tạo ra một model hợp lý và khát quát được toàn bộ tập dữ liệu của chúng ta để chọn ra những siêu tham số chính xác nhất.
* Model Selection: Một tập dữ liệu của chúng ta có thể được biểu diễn bằng nhiều dạng model khác nhau. Vậy giữa các model đó chúng ta phải chọn model nào để biểu diễn cho dữ liệu của chúng ta. Chúng ta phải chọn một model nào có thể hoạt động tốt nhất và ít lỗi nhất trên tập validation set.

1. Validation set

* Muốn tạo ra một model đủ tốt để biểu diễn và hoạt động tốt trên tập dữ liệu thì trước tiên ta phải chia bộ dữ liệu của chúng ta ra làm 3 phần (training set, validation set và test set).
* Tập validation set dùng để đánh giá model vừa được huấn luyện xong của chúng ta trước khi đem cho tập test set vào thử nghiệm.
* Cho máy học trên tập training set và tạo nên nhiều model với nhiều siêu tham số khác nhau. Sau đó chọn model hoạt động tốt nhất trên tập validation set. Model này là model cuối cùng hoạt động tốt nhất trên tập huấn luyện đầy đủ (gồm training set và validation set). Cuối cùng ta đánh giá model trên tập test set để ước tính generalization error và xử lý nếu gặp các hiện tượng overfitting, underfitting,..
* Thông thường các model được huấn luyện bằng cách này thường hoạt động rất tốt và chính xác. Tuy nhiên nếu tập validation quá nhỏ thì sẽ không đánh giá đúng được model vừa huấn luyện ra có tốt hay không, và ta sẽ chọn nhầm một model không tối ưu trên tập dữ liệu của chúng ta. Ngược lại, nếu tập xác thực quá lớn thì tập training set của chúng ta sẽ bị nhỏ lại, điều này cực kỳ không tốt vì nếu tập training set quá nhỏ thì máy sẽ không thể nào học và cho ra một model đủ tốt để khái quát được dữ liệu của chúng ta. Vì vậy chúng ta cần chia bộ dữ liệu của chúng ta thành 3 tập training set, validation set, test set sao cho hợp lý để có thể huấn luyện ra một model hoàn hảo nhất có thể.

1. Tóm tắt một số quy lệnh của python
2. Một số kiểu dữ liệu:

Text Type: str

Numeric Types: int, float, complex

Sequence Types: list, tuple, range

Mapping Type: dict

Set Types: set, frozenset

Boolean Type: bool

Binary Types: bytes, bytearray, memoryview

1. If … else:

Equals: a == b

Not Equals: a != b

Less than: a < b

Less than or equal to: a <= b

Greater than: a > b

Greater than or equal to: a >= b

Cấu trúc: if b > a:

print("b is greater than a")

elif a == b:

print("a and b are equal")

else:

print("a is greater than b")

1. Vòng lặp while:

while i < 6:

print(i)

1. Vòng lặp For:

fruits = ["apple", "banana", "cherry"]

for x in fruits:

print(x)

1. Hàm:

def my\_function(fname):

print(fname + " Refsnes")

my\_function("Emil") 🡪 Emil Refsnes

my\_function("Tobias") 🡪 Tobias Refsnes

my\_function("Linus") 🡪 Linus Refsnes

1. Mảng: cars = ["Ford", "Volvo", "BMW"]

Một số phương thức có sẵn của mảng trong python: append(), clear(), copy(), count(), extend(), index(), insert(), pop(), remove(), reverse(), sort()

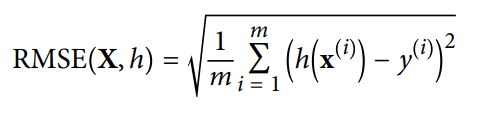
Tuần 3:

Các bước từ đầu đến cuối để thực hiện một dự án machine learning:

* 1. Look at the big picture. (Quan sát tổng quan về dự án đang làm)
* 2. Get the data. (Thu thập dữ liệu cần thiết)
* 3. Discover and visualize the data to gain insights. (Khai phá dữ liệu)
* 4. Prepare the data for Machine Learning algorithms. (Chuẩn bị dữ liệu cho thuật toán Machine learning)
* 5. Select a model and train it. (Lựa chọn model hợp lý để bắt đầu huấn luyện cho máy)
* 6. Fine-tune your model. (Tinh chỉnh lại model)
* 7. Present your solution. (Trình bày giải pháp giải quyết vấn đề)
* 8. Launch, monitor, and maintain your system. (Khởi chạy, giám sát và bảo trì hệ thống).

Để hiểu rõ hơn ta đi kỹ hơn từng bước trên.

Đầu tiên là bước 1: Quan sát một cách tổng quát về dự án chuẩn bị xử lý.

* Phải hỏi kỹ người đi săn dự án là mục tiêu của dự án cần đạt được chính xác là gì? Việc xây dựng một model hoạt động được chưa chắc là mục tiêu cuối cùng của dự án vì vậy phải xác nhận lại xem công ty dự kiến sử dụng và thu lợi từ model này như thế nào? Việc xác nhận này hết sức quan trọng bởi vì khi ta đã xác định được mục tiêu cần đạt được thì hướng đi đến mục tiêu đó của ta sẽ chính xác hơn. Qua đó ta có thể chọn được thuật toán phù hợp, thước đo để đánh giá độ hiệu quả của model và cách chỉnh sửa như thế nào để model chính xác nhất.
* Tiếp theo ta phải hỏi về yêu cầu của khách hàng là họ có bắt buộc phải làm theo hướng nào? Model nào không? Sau đó cần phải tìm hiểu xem phải chọn loại dữ liệu như nào để phù hợp với dự án của chúng ta.
* Tiếp đến là cần chọn một thước đo hiệu suất (performance measure). Thước đo hiệu suất điển hình mà người ta hay dùng là Root Mean Square Error (RMSE). RMSE là độ lệch chuẩn lấy căn bậc 2 của MSE, MSE là một hàm rủi ro được tính bằng trung bình của bình phương các sai số, nói cách khác MSE là một hàm rủi ro. RMSE được tính theo công thức:

Bước 2: Thu thập dữ liệu:

* Chuẩn bị đầy đủ các công cụ cần thiết để chuẩn bị thực hiện dự án.
  + Chọn môi trường làm việc (python hay một ngôn ngữ bất kỳ) và cài đặt đầy đủ các thư viện và các công cụ hỗ trợ việc thực hiện.
  + Nếu dùng python có thể tạo một môi trường làm việc biệt lập để tránh bị xung đột các phiên bản của thư viện giữa các dự án khác nhau.
* Bắt đầu thu thập dữ liệu:
  + Có thể tự đi thu thập dữ liệu để có được nguồn dữ liệu theo đúng yêu cầu của dự án và không cần phải xử lý lại dữ liệu. Nhưng việc này không được khuyến khích trừ trường hợp bất khả kháng vì việc đi thu thập một lượng dữ liệu khổng lồ đủ để máy học được thì mất rất nhiều thời gian và công sức.
  + Cách đơn giản hơn là download từ các nguồn dữ liệu có sẵn và đáng tin cậy trên mạng (có thể sử dụng trang TỔNG CỤC THỐNG KÊ của chính phủ - https://www.gso.gov.vn/du-lieu-va-so-lieu-thong-ke/) và cách này được khuyến khích sử dụng hơn vì chỉ tốn thời gian xử lý lại nguồn dữ liệu chứ không mất nhiều thời gian và công sức, tiền của để đi thu thập dữ liệu.
  + Việc tiếp theo cần làm là có nhìn một cách tổng quát và nắm được cấu trúc của nguồn dữ liệu vừa thu thập được (số cột (features), số dòng (samples), kiểu dữ liệu của từng cột) để thuận tiện cho việc xử lý và làm việc với dữ liệu.
  + Vẽ các biểu đồ của từng thuộc tính trong bảng dữ liệu để kiểm tra và thấy được mức phân bố của dữ liệu, những dữ liệu bất thường, hiểu rõ được phần nào dữ liệu đã thu thập được.
  + Việc quan trọng tiếp theo là cắt một phần trong tập dữ liệu ra làm tập test set và chỉ xử lý và huấn luyện trên phần còn lại, không dùng tập test set cho việc huấn luyện.
* Khám phá và trực quan hóa dữ liệu để nắm được thông tin chi tiết:
  + Từ các bước trước đến hiện tại, chúng ta chỉ mới xem sơ qua dữ liệu và nắm được tổng quát tập dữ liệu của chúng ta. Nên việc cần làm của bước hiện tại là đi sâu hơn một chút vào tập dữ liệu để hiểu rõ hơn về nó.
  + Phải chắc chắn rằng không được động đến tập test set và chỉ khai phá trên tập training set.
  + Nếu tập training của ta quá lớn thì ta có thể cắt ra một phần nhỏ làm đại diện để khai phá dữ liệu.

Bước 3: Khai phá và trực quan hóa dữ liệu để có cái nhìn chi tiết hơn:

* Tiếp theo là trực quan hóa miền dữ liệu: Sau khi vẽ được biểu đồ phân bố của miền dữ liệu, ta sẽ thấy được những vị trí có mật độ phân bố cao. Khi đó ta có thể xác định được miền dữ liệu bình thường và miền dữ liệu bất thường.
* Tìm mối tương quan của các dữ liệu. Trong python ta có thể dùng hàm corr() để thấy rõ được tương quan chuẩn của các features với kết quả dự đoán sau khi xử lý. Hệ số tương quan sẽ nằm trong khoảng -1 đến 1. Features nào có hệ số gần bằng 1 có nghĩa là có mối tương quan tích cực mạnh mẽ (tỉ lệ thuận với kết quả), features nào có hệ số gần bằng -1 thì có nghĩa là mối tương quan tiêu cực mạnh (có xu hướng tỉ lệ nghịch với kết quả).
* Thử dùng các thuộc tính kết hợp: Qua các bước trên ta có thể nhận thấy được những thuộc tính nào ảnh hưởng đến thuộc tính nào, những thuộc tính nào quyết định đến kết quả thì thử kết hợp chúng lại với nhau qua các phép toán để làm cho dữ liệu đầu vào của chúng càng sạch càng dễ hiểu thì việc huấn luyện cho máy càng dễ dàng và chính xác.

Bước 4: Chuẩn bị dữ liệu cho thuật toán Machine Learning:

* Thay vì làm việc này một cách thủ công thì ta nên viết các hàm để làm điều này bởi vì:
  + Có thể cung cấp lại các sự thay đổi trên dữ liệu một cách dễ dàng hoặc dễ dàng xử lý khi cung cấp lại một tập dữ liệu mới.
  + Dần dần tích góp và xây dựng nên một thư viện các hàm chuyển đổi mà ta có thể tái sử dụng trong các dự án trong tương lai.
  + Có thể sử dụng các hàm này cho các dự án đã hoàn thành và đang hoạt động để xử lý dữ liệu mới trước khi đưa vào thuật toán.
  + Dễ dàng thử nhiều hàm chuyển đổi khác nhau và so sánh độ hiệu quả và sự kết hợp giữa các hàm chuyển đổi nào hoạt động tốt nhất trong từng dự án cụ thể.
* Làm sạch dữ liệu (data cleaning):
  + Hầu hết các thuật toán Machine Learning đều không thể xử lý được với những features bị khuyết dữ liệu. Vì vậy ta phải tạo ra các hàm để xử lý những thuộc tính bị thiếu này.
  + Có nhiều cách để xử lý trường hợp này:
    - Loại bỏ các miền dữ liệu tương ứng với sự phân bố của dữ liệu bị khuyết này.
    - Loại bỏ feature bị khuyết trên toàn bộ samples của tập dữ liệu.
    - Điền vào các vị trí khuyết một giá trị theo quy định (bằng 0, giá trị trung bình của miền dữ liệu tương ứng với samples bị khuyết dữ liệu,….)
  + Nếu quyết định chọn cách thứ 3 thì ta nên tính giá trị trung bình trên tập training set và dùng giá trị này điền hết vào những dữ liệu bị khuyết trên tập training set.
* Xử lý dữ liệu văn bản và phân loại thuộc tính:
  + Loại bỏ các thuộc tính văn bản bởi vì ta không thể tính giá trị trung bình của thuộc tính này.
  + Hầu hết các thuật toán Machine Learning thì thường xử lý trên các dữ liệu số nên ta phải biến đổi các thuộc tính văn bản sang thuộc tính số.
* Tùy chỉnh các hàm chuyển đổi: Mặc dù thư viện scikit-Learn của python có cung cấp sẵn các hàm chuyển đổi rất hữu dụng nhưng chúng ta cũng nên tự viết cho riêng mình một hàm chuyển đổi để xử lý các việc làm sạch dữ liệu hoặc kết hợp các thuộc tính đặc biệt.
* Co giãn dữ liệu trong các features (features scaling): do số liệu ở mỗi features có thể sẽ cách biệt nhau rất lớn (có thể từ 0-10 cũng có thể từ 3000-5000,…) nên ta cần phải xử lý tất cả các dữ liệu trong từng feature về % hoặc về khoảng 0-1 để tạo ra một tập dữ liệu đủ trực quan cho máy học.

Bước 5: Chọn và huấn luyện một model: Sau khi đã định hình vấn đề, thu thập dữ liệu, chia tập training set và test set, viết các hàm chuyển đổi để làm sạch dữ liệu và chuẩn bị dữ liệu cho thuật toán Machine Learning thì bây giờ ta cho thể dễ dàng chọn và huấn luyện được một model Machine Learning.

Đánh giá tốt hơn bằng cách sử dụng Cross-Validation: Với cách này ta chia nhỏ tập training set của chúng ta ra thành các tập training set nhỏ hơn và một tập validation set bằng cách sử dụng K-fold cross-validation của python. Hàm này sẽ giúp chúng ta chia nhỏ tập dữ liệu ra một cách ngẫu nhiên thành các tập training set và validation set sau đó liên tục thay đổi các tập training set và validation set và xử lý thì ta sẽ thu được một kết quả tốt hơn nhiều.

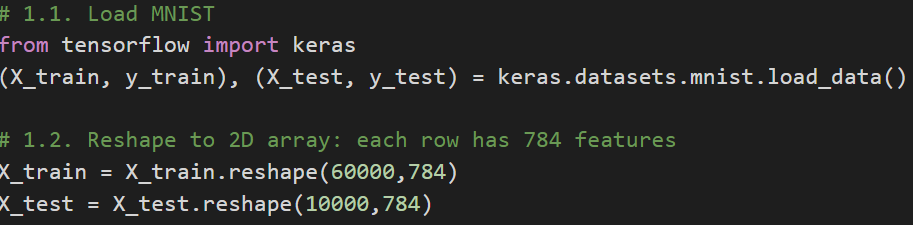
Bước 6: Tinh chỉnh lại model:

* Ta có thể dùng một số cách như grid search hoặc randomized search để dò tìm ra các siêu tham số thích hợp nhất cho model của chúng ta (những cách này python đã cung cấp sẵn các hàm xử lý hết sức tiện lợi cho ta dùng)
* Các phương thức kết hợp: Một cách khác để tinh chỉnh model là cố gắng kết hợp các mô hình hoạt động tốt. Bởi vì một nhóm sẽ thường hoạt động tốt hơn một cá thể tốt thất.
* Phân tích các models tốt nhất và lỗi của chúng: tính điểm của từng thuộc tính ảnh hưởng đến kết quả, sau đó có thể ta sẽ cần loại bỏ một vài features it hữu dụng để giải quyết vấn đề. Ta nên xem xét qua các lỗi cụ thể mà model của ta mắc phải và cố gắng tìm hiểu lý do tạo ra lỗi để khắc phục nó.
* Đánh giá lại model bằng tập test set: sau khi xử lý xong ta nên dùng tập test set để đánh giá lại thuật toán ta có hoạt động tốt trên những dữ liệu chưa gặp lần nào không? Nếu model hoạt động không tốt thì tìm hiểu nguyên nhân dẫn đến lỗi trên và khắc phục nó để model có thể hoạt động tốt nhất trên một tập dữ liệu hoàn toàn mới. Nếu model đã hoạt động tốt thì chuyển đến bước cuối cùng.

Bước 7: Khởi chạy, giám sát và bảo trì hệ thống: Sau khi đã hoàn thành các bước trên thì cơ bản model của chúng ta đã hoàn thành, cho model chạy trên nguồn dữ liệu thực tế. Kiểm tra xem model có gặp những lỗi sai gì không để xử lý kịp thời. Bảo trì thường niên hệ thống để có thể giúp hệ thống hoạt động tốt trên nguồn dữ liệu biến đổi liên tục từng ngày.

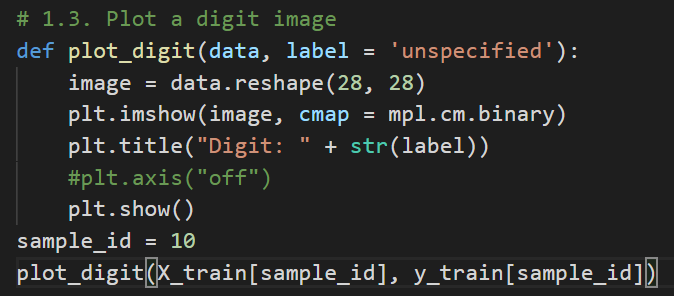
Tuần 5: Classification:

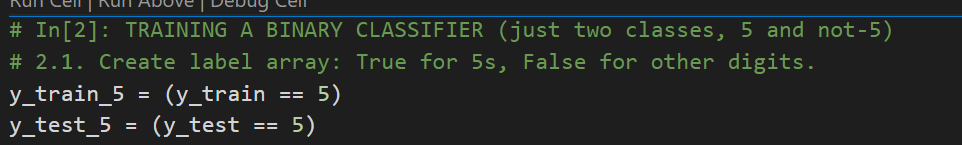
Sử dụng tập dữ liệu MNIST mà thư viện tensorflow – keras có cung cấp sẵn. Tập Mnist gồn 70000 hình ảnh nhỏ (kích thước 28x28 pixel) bao gồm các kí tự số viết tay từ 0-9 của các học sinh và nhân viên của Census Bureau. Mỗi hình ảnh ấy được đánh nhãn với ký tự tương ứng với hình ảnh ấy.

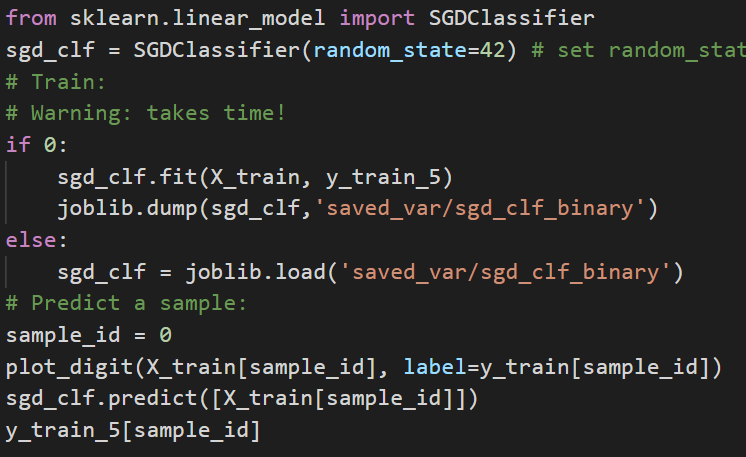
* Đầu tiên ta chia tập dữ liệu mnist của ta thành các tập training set và test set. Cụ thể ta chia thành các tập X\_train, y\_train, X\_test, y\_test. X\_train chứa dữ liệu của tập training set, y\_train chứa nhãn của tất cả các dữ liệu trong tập training set. Tương tự X\_test và y\_test cũng là tập dữ liệu và nhãn của tập dữ liệu test set. 

Tải dữ liệu cho tập các tập training set và test set (60000 samples cho tập training set và 10000 samples cho tập test set). Bởi vì mỗi sample là một bức ảnh có kích thước 28x28 là một mảng 2 chiều 28x28 nên ta cần biến đổi mảng 2 chiều này thành mảng 1 chiều có 784 phần tử (=28x28).

* Sau đó load một số samples lên xem thử đã chia và biến đổi các hình ảnh để lưu vào các tập đã chia đã đúng chưa.

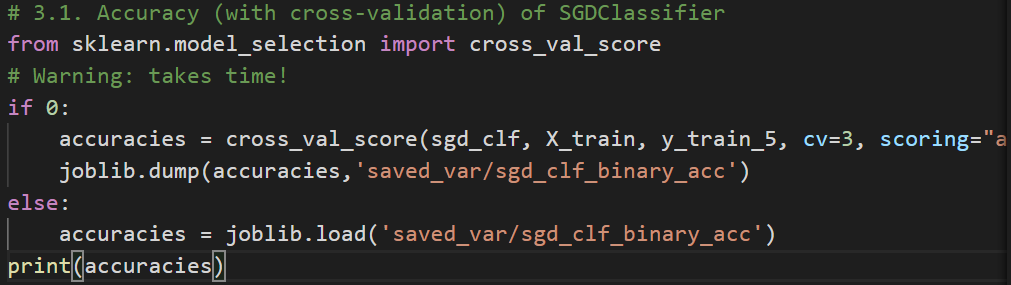


* Bài toán có 10 class là các ký tự từ 0-9 vì vậy để làm bài toán binary class thì ta chia tập dữ liệu của ta thành 2 class là ký tự 5 và không phải ký tự 5. 
* Ta thử với thuật toán Stochastic Gradient Descent (SGD) classifier:



Huấn luyện cho sgd\_clf trên binary class (5 và not-5) và thử đoán một sample.

* Dùng K-fold cross\_validation\_score để xem độ chính xác của từng cách chia nhỏ tập training set và validation set.



Độ chính xác khá cao >95%.

* Thử dùng DumpClassifier. Với DumpClassifier bất cứ giá trị nào đưa vào thuật toán này đều trả ra giá trị dự đoán là False nhưng độ chính xác của nó trên tập dữ liệu mnist thì cực kỳ cao >90%. Tại sao bất kỳ dữ liệu nào đưa vào nó đều đưa kết quả là False mà độ chính xác lại cao như vậy? Bởi vì tập dữ liệu này không cân bằng. Tập mnist của ta có 10 class nhưng ta lại biến nó thành binary class (2 class là 5 và not-5) nên số lượng samples là 5 (True) chỉ chiếm 1/10 số lượng samples của cả tập mnist nên đưa dữ liệu nào nó cũng đưa ra kết qua là False thì độ chính xác lên cao đến 90%.
* Confusion matrix: ma trận này thể hiện dự đoán đúng và sai của thuật toán chúng ta huấn luyện.

0 1

0 [[53892 687]

1 [ 1891 3530]]

# 0-0: là không phải 5 và máy đoán không phải 5

# 1-1: là 5 và máy đoán là 5

# 0-1: là không phải 5 nhưng máy dự đoán là 5

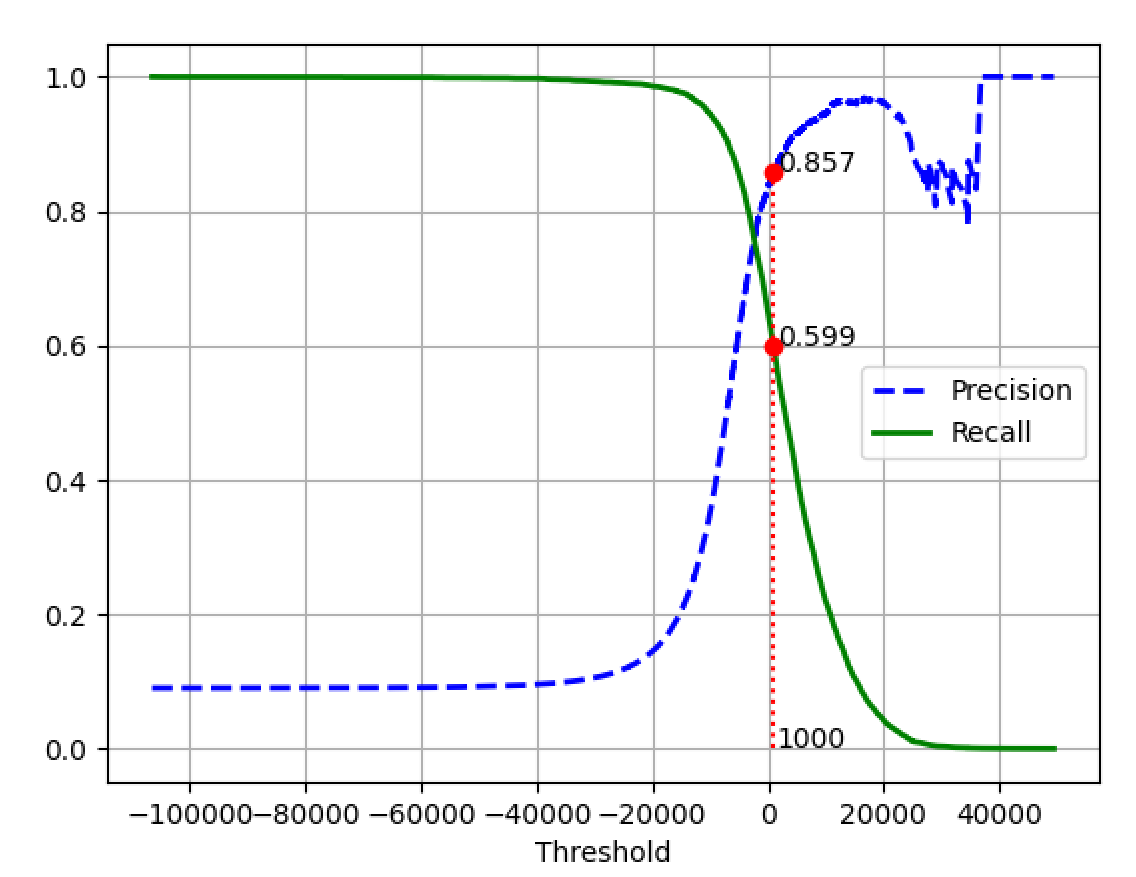
# 1-0: là 5 nhưng máy dự đoán không phải 5

[[54579 0]

[ 0 5421]]

Còn phía trên là một matrix lý tưởng có nghĩa là thuật toán đoán không sai sample nào.

* Precision và recall: dựa vào confusion matrix phía trên, ta có thể chia [0,0] là True Negative(TN), [0,1] là False Positive(FP), [1,0] là False Negative(FN), [1,1] True Positive (TP)
* Ta tính precision và recall bằng công thức:
  + Precision = TP/(TP+FP)
  + Recall = TP/(TP+FN)
  + Precision cho chúng ta thấy độ chính xác của thuật toán (độ chính xác của kết quả do thuật toán dự đoán ra)
  + Recall cho chúng ta thấy độ đầy đủ của dự đoán của thuật toán (Cho chúng ta trong tổng số các dự đoán đúng chúng ta chỉ ra được bao nhiêu trường hợp đúng)
* Precision and recall tradeoff:
  + Một model tốt nhất khi cả precision và recall đều đạt giá trị lớn.
  + Nhưng trong thực tế ta khó có thể làm cho cả 2 đều lớn được vì vậy mà tùy thuộc vào yêu cầu thực tế của thuật toán mà ta buộc phải đánh đổi mất cái này để được cái kia nhưng vẫn cố gắng giữ 2 giá trị này cùng cao nhất có thể.
  + Ví dụ trong bài toán ta đang làm thì ta có biểu đồ như sau:

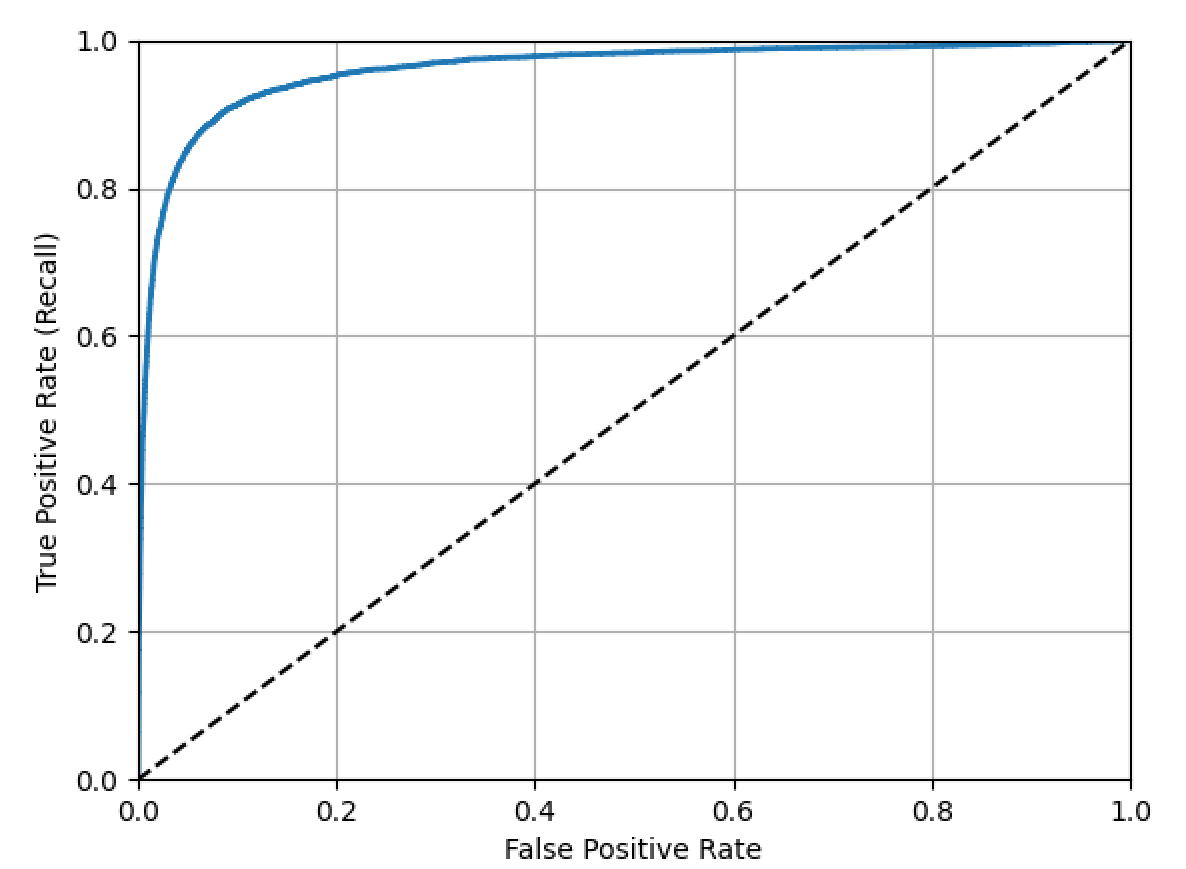


Ta chọn ở ngưỡng 1000 là được giá trị tốt nhất có thể.

* F1 score: cho ta thấy giá trị trung bình độ hài hòa của precision và recall.
  + Được tính theo công thức:

F1=2 x Precision x Recall / (Precision + Recall)

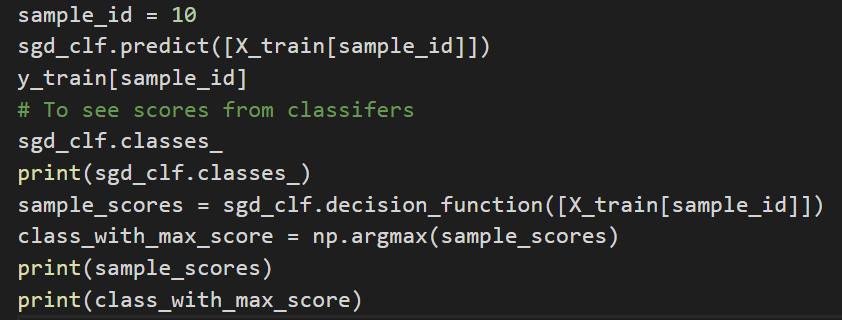
* + Giá trị của F1 nằm trong đoạn 0-1, tốt nhất khi F1 bằng 1 và tệ nhất khi F1 = 0.
* ROC-curve (Receiver operating characteristic curve)



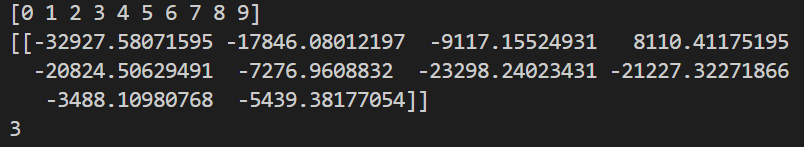
* + ROC curve cho ta thấy tỉ lệ True Positive Rate (Recall) / False Positive Rate (số lượng FP/(TN+FP))
  + FPR là tỉ lệ của các trường hợp Negative được phân loại nhầm thành Positive.
  + TPR càng cao thì bộ phân loại càng tạo ra nhiều FPR.
  + Bộ phân loại càng tốt nếu ROC-curve càng đi cao về góc trên bên trái của biểu đồ.

Tuần 6: classifier (continue)

* Thử sử dụng SGDClassifier:
  + Scikit-Learn sẽ tự động chạy OvA khi ta cố gắng sử dụng một binary classifier cho một multiclass classifier.
  + Sau khi train xong ta cho máy đoán thử một sample



* + Kết quả là:



Vì bây giờ chúng ta đang hoạt động trên tập multiclass nên máy sẽ cho ta thấy được điểm dự đoán của sample đó trên toàn bộ các class của tập dữ liệu. Điểm ở class nào cao nhất thì nó sẽ là kết quả dự đoán của máy. Ở ví dụ sample thứ 10 trên thì máy sẽ đoán nó là số 3 và máy đã dự đoán đúng.

* Buộc scikit-Learn chạy multiclass bằng OvO (OneVsOneClassifier) or OvA (OneVsRestClassifier)
  + Chạy Multiclass bằng OvA cũng giống như chạy SGD, kết quả tương tự trên.
  + Chạy multiclass bằng OvO: số class khi áp dụng phương pháp này là n(n-1)/2. Đối với tập dữ liệu mnist thì số class khi dùng OvO là 45.
* Đánh giá độ chính xác khi dùng SGD và RandomForest cho multiclass
  + Quan sát độ chính xác của 2 phương pháp:



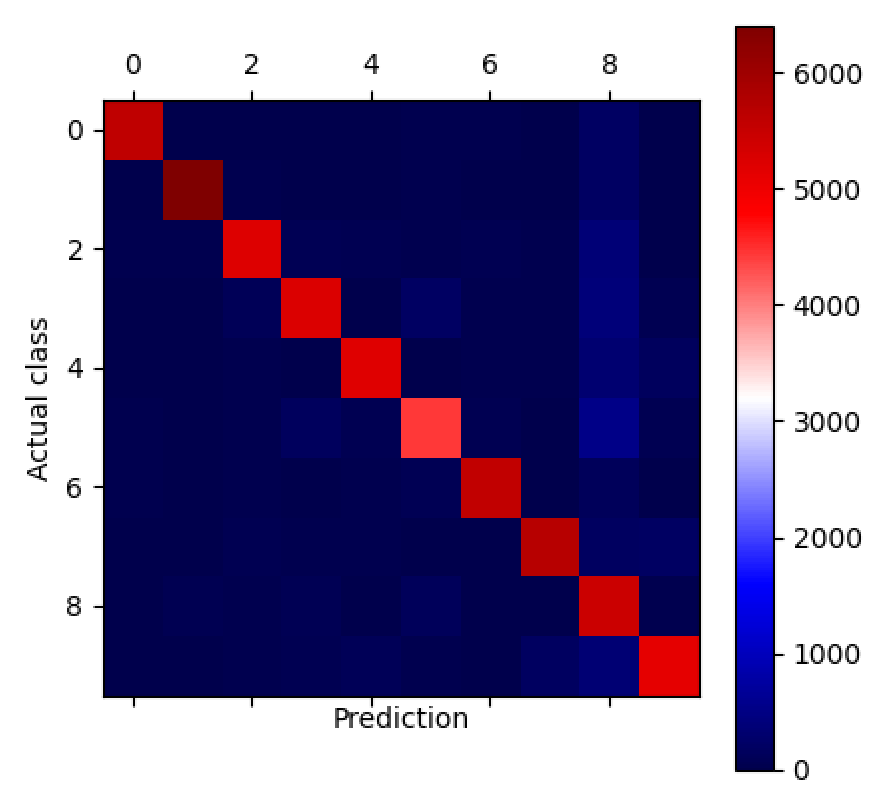
Ta có thể thấy độ chính xác của RandomForest lớn hơn rất nhiều so với SGD

* + Scale các features và đánh giá lại các phương pháp phân loại. Sau khi scale ta có được độ chính xác của 2 phương pháp trên như sau:



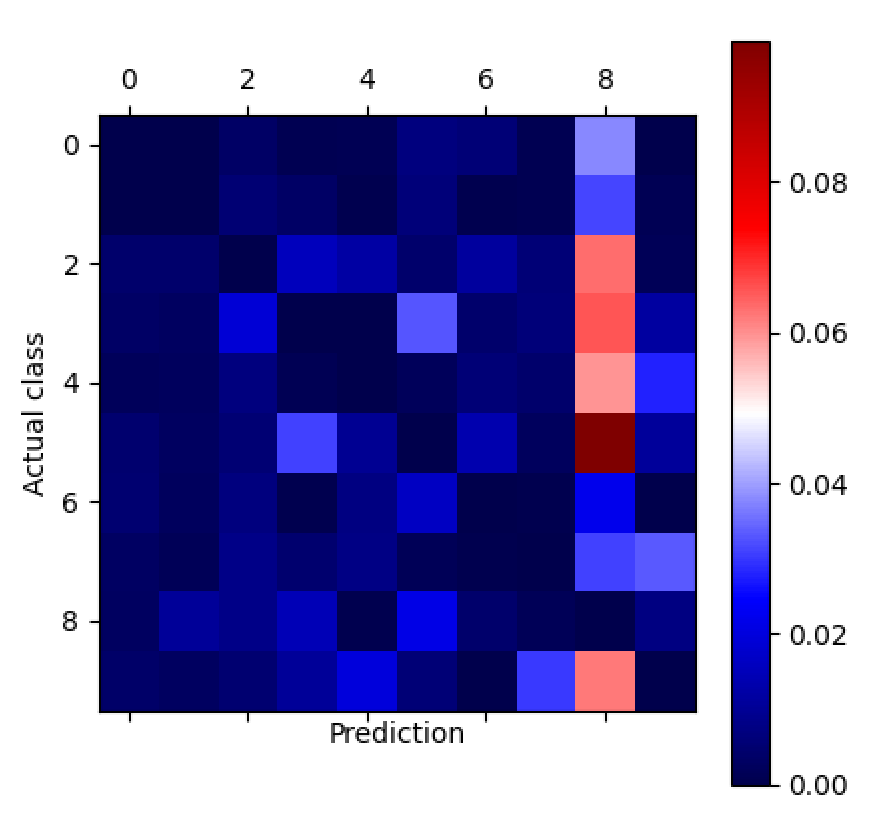
Ta có thể thấy được độ chính xác của phương pháp SGD được cải thiện lên rất nhiều, còn RandomForest thì không được cải thiện nhiều bởi vì bản thân nó đã rất tốt rồi.

* Phân tích các lỗi của model chúng ta vừa xử lý ra được:



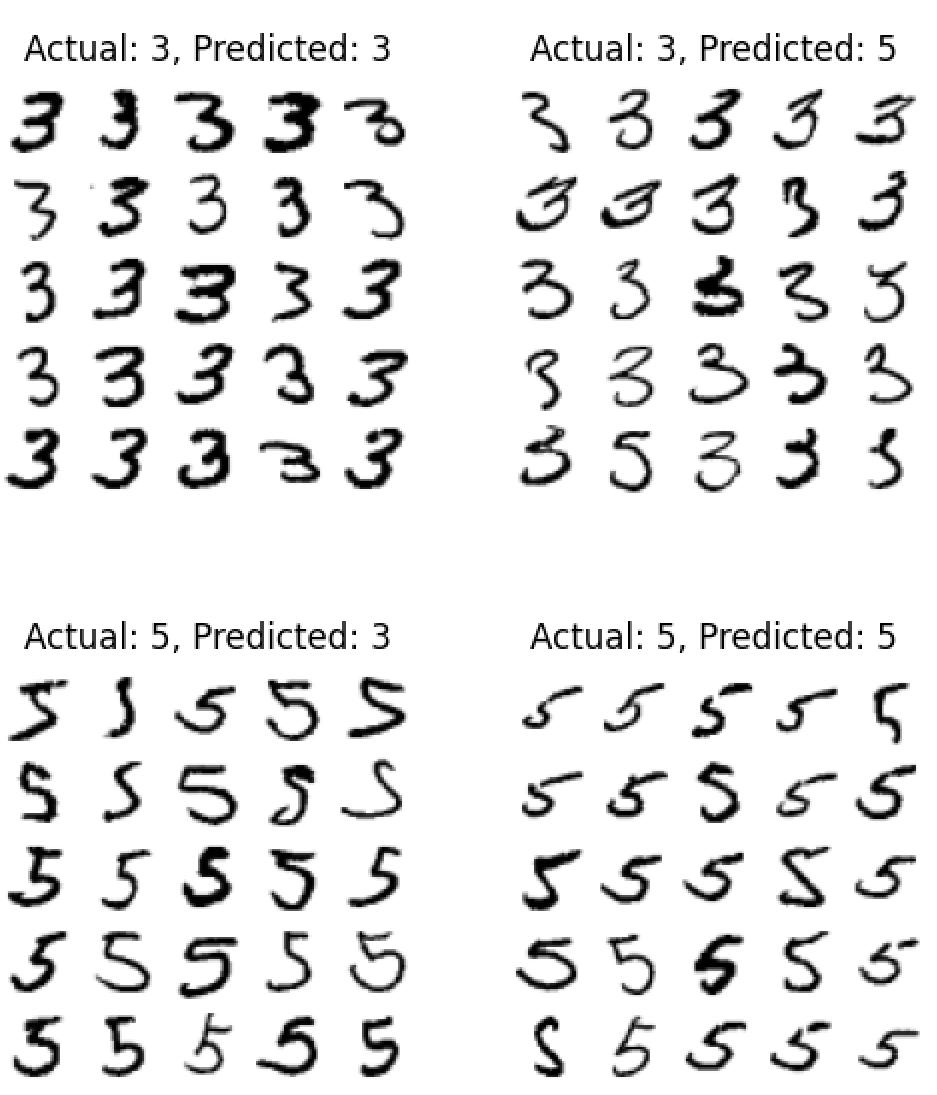
Máy dễ đoán sai ở một số trường hợp: ví dụ như số 3 nhưng lại nhầm thành 5, 6 nhưng lại nhầm thành 9,…

* + Chuyển đổi từ số lượng thành tỉ lệ



Ta có thể thấy máy đoán số 8 tỉ lệ sai rất cao.

* Vài cách để giảm sự đoán sai số 8 là:
  + Thu thập thêm dữ liệu huấn luyện
  + Đếm số lượng vòng lặp đã đóng
  + Xử lý hình ảnh trước khi cho máy học.
* Như lúc trên ta đã có đề cập số 3 và số 5 dễ nhầm lẫn ta sẽ in ra vài trường hợp cơ bản về việc dự đoán chính xác và nhầm lẫn giữa các trường hợp số 3 và 5



Một số trường hợp nhầm lẫn giữa 3 và 5 ta thấy nhầm lẫn là do các dữ liệu sai gây nhiễu đến mắt thường ta còn nhìn bị nhầm lẫn thì máy sai cũng là chuyện dễ hiểu.

* Multilabel classification: đối với một bài toán thông thường thì khi ta đưa một sample vào cho máy dự đoán thì dù đúng hay sai thì máy sẽ trả ra một label thích hợp cho sample đó. Nhưng đối với multilabel thì kết quả trả ra không phải là một label mà là một vector label (vector binary) hay dễ hiểu hơn là một mảng một chiều chứa các label dự đoán cho kết quả ấy. Tùy vào yêu cầu của bài toán mà ta sẽ đặt điều kiện cho vector để có được một kết quả binary vector tương ứng. Ví dụ một multilabel có điều kiện là >5 và là số chẵn thì khi ta nhập 6 vào ta sẽ được kết quả là vector [1,1], 4 thì [0,1], 9 thì [1,0],…
* Multioutput cũng tương tự như multilabel nhưng đối với multilabel thì kết quả trả ra của thuật toán là một binary vector thì multioutput kết quá trả ra là một multiclass vector. Có nghĩa là kết quả nằm trong vector trả ra không phải là những số 0 1 (true, false) mà là một vector chứ các giá trị tùy thuộc vào giá trị cần trả ra.