VIETNAM GENERAL CONFEDERATION OF LABOUR

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**PHAN LE MINH NHUT- 519H0211**

**FINAL PROJECT**

**INTRODUCTION TO MACHINE LEARNING**

**SOFTWARE ENGINEERING**

Advised by

**MSc. Le Anh Cuong**

**HO CHI MINH CITY, YEAR 2023**

**Bài 1 (3 điểm): làm riêng từng người**

Trình bày một bài nghiên cứu, đánh giá của em về các vấn đề sau:

1. Tìm hiểu, so sánh các phương pháp Optimizer trong huấn luyện mô hình học máy;
2. Tìm hiểu về Continual Learning và Test Production khi xây dựng một giải pháp học máy để giải quyết một bài toán nào đó.

------------------------------------------------------------------------------------------------------------

1. Các phương pháp tối ưu hóa hay còn gọi là “Optimizer” đóng vai trò rất quan trọng trong việc tìm ra tham số tốt nhất cho mô hình, bới nó chính là thuật toán giúp cho việc cập nhật và điều chỉnh các tham số của mô hình (như trọng số và bias) để giảm thiểu hàm mất mát (loss function). Nhờ vậy kết quả dự đoán của mô hình sẽ chính xác hơn.

Một số phương pháp tối ưu hóa thông dụng như là :

* + **Gradient Descent**: Là một trong những phương pháp cơ bản và thông dụng nhất, phương pháp này cập nhật trọng số mô hình bằng cách di chuyển theo hướng ngược lại với gradient của hàm mất mát. Sử dụng toàn bộ tập dữ liệu để cập nhật trọng số mô hình theo hướng giảm thiểu hàm mất mát.
    - **Áp dụng**:
    - Tính gradient của hàm mất mát đối với mỗi trọng số.
    - Cập nhật trọng số theo hướng ngược lại với gradient.
    - Lặp lại cho đến khi đạt đến điều kiện dừng (ví dụ: số lần lặp, sự thay đổi nhỏ trong mất mát).
    - **Ưu điểm**: Phương pháp này rất đơn giản và dễ hiểu,rất hợp lí khi sử dụng cho các tập dữ liệu nhỏ và trung bình. Cập nhật trọng số sau mỗi vòng lặp
    - **Nhược điểm**: Không hiệu quả với dữ liệu lớn do cần tính toán gradient trên toàn bộ tập dữ liệu.Vẫn còn phụ thuộc vào nghiệm tạo ban đầu và LR (Learning Rate)
    - **So sánh**:
    - Nếu so sánh, GD sẽ chậm hơn SGD và Mini-Batch GD trong việc xử lý các dữ liệu lớn.
    - Sẽ ít bị “nhiễu” hơn so với SGD, nhưng sẽ dễ mắc kẹt ở các điểm cực tiểu địa phương.
  + **Stochastic Gradient Descent (SGD):** Biến thể của GD, nhưng thay vì sử dụng toàn bộ tập dữ liệu, SGD sẽ cập nhật trọng số cho mỗi mẫu dữ liệu, giúp hội tụ nhanh hơn nhưng có thể không ổn định trong một số trường hợp.
* **Áp dụng**:
  + - Tính gradient của hàm mất mát cho mỗi mẫu dữ liệu.
    - Cập nhật trọng số dựa trên gradient này.
    - Lặp lại quá trình cho toàn bộ tập dữ liệu.
* **Ưu điểm:** Nhanh hơn GD khi xử lý dữ liệu lớn, có thể thoát khỏi điểm cực tiểu địa phương.
* **Nhược điểm:** Cập nhật trọng số không ổn định, có thể dẫn đến hội tụ chậm
* **So sánh:**
  + - Quy trình xử lý nhanh hơn và tương đồng so với GD nhưng cập nhật không ổn định, dẫn đến sự “nhiễu” lớn.
    - Thích hợp với dữ liệu lớn
  + **Mini-Batch Gradient Descent**: Là sự kết hợp giữa GD và SGD, sử dụng một lô (batch) của dữ liệu thay vì toàn bộ tập dữ liệu hoặc một mẫu dữ liệu, giúp giảm thiểu việc bị “nhiễu” trong quá trình học máy.
    - **Áp dụng:**
    - Chia tập dữ liệu thành nhiều lô.
    - Tính gradient và cập nhật trọng số cho mỗi lô.
    - Lặp lại cho tới khi hoàn thành tất cả các lô.
    - **Ưu điểm:** Cung cấp sự cân bằng giữa hiệu suất xử lý của GD và sự nhanh nhẹn của SGD.
    - **Nhược điểm:** Cần lựa chọn kích thước lô một cách phù hợp
    - **So sánh:**
    - Là phương pháp cân bằng tốt giữa hiệu suất và ổn định trong tất cả các phương pháp khác và cần lựa chọn phù hợp kích thước lô
  + **Momentum**: Là phương pháp SGD nhưng cải tiến và thêm "quán tính" vào để cập nhật trọng số, giúp tăng tốc độ học và vượt qua các điểm cực tiểu
* **Áp dụng:**
  + - Tính gradient như trong SGD.
    - Áp dụng một phần của cập nhật trọng số trước đó vào cập nhật hiện tại để tăng quán tính.
    - Cập nhật trọng số dựa trên gradient và quán tính.
* **Ưu điểm:** Giúp thoát khỏi các điểm cực tiểu địa phương, tăng tốc độ hội tụ.
* **Nhược điểm:** Thêm siêu tham số (ví dụ: tỷ lệ quán tính) cần được tinh chỉnh. Các tỷ lệ quán tính cần được tinh chỉnh
* **So sánh:**
  + - Nhanh hơn SGD.
    - Phù hợp với các bài toán có dạng địa hình phức tạp.
  + **Adagrad**: Điều chỉnh tốc độ học tự động cho mỗi tham số, rất hữu dụng với các dữ liệu thưa thớt và các tham số không cập nhật thường xuyên.
    - **Áp dụng:**
    - Lưu trữ tổng bình phương của các gradient trước đó cho mỗi tham số.
    - Điều chỉnh tốc độ học dựa trên tổng bình phương này.
    - Cập nhật trọng số theo tỷ lệ ngược với kích thước của gradient.
    - **Ưu điểm:** Hiệu quả với dữ liệu thưa và các bài toán yêu cầu chính xác cao.
    - **Nhược điểm:** Tốc độ học có thể giảm quá nhanh, dẫn đến hội tụ sớm
    - **So sánh:**
    - Tránh được vấn đề lựa chọn tốc độ học cố định.
    - Có thể giảm tốc độ học quá nhanh về cuối quá trình huấn luyện.
  + **RMSprop**: Phiên bản Cải tiến Adagrad bằng cách sử dụng trung bình di động cho gradient, giúp tránh việc tốc độ học giảm quá nhanh.
    - **Áp dụng:**
    - Tính bình phương của gradient và áp dụng trung bình di động.
    - Điều chỉnh tốc độ học dựa trên trung bình di động này.
    - Cập nhật trọng số tương tự như Adagrad nhưng với tốc độ học được hiệu chỉnh.
    - **Ưu điểm:** Giải quyết vấn đề giảm tốc độ học quá nhanh của Adagrad.
    - **Nhược điểm:** Cần cấu hình thêm siêu tham số.
    - **So sánh:**
    - Hiệu quả hơn Adagrad, đặc biệt trong các tình huống cần tốc độ học linh hoạt.
  + **Adam (Adaptive Moment Estimation):** Là phương pháp kết hợp các ý tưởng từ RMSprop và Momentum để điều chỉnh tốc độ học dựa trên trung bình di động của cả gradient và bình phương gradient.
* **Áp dụng:**
  + - Tính trung bình di động của gradient và bình phương gradient.
    - Điều chỉnh cả gradient và tốc độ học dựa trên trung bình di động này.
    - Cập nhật trọng số sử dụng cả hai yếu tố này.
* **Ưu điểm:** Cân bằng tốt giữa hiệu suất và ổn định, ít cần điều chỉnh siêu tham số.
* **Nhược điểm:** Có thể không ổn định trong một số trường hợp cụ thể.
* **So sánh:**
  + - Rất phổ biến do cân bằng giữa hiệu suất và ổn định.
    - Hiệu quả trong nhiều bài toán và mô hình khác nhau.

1. *Continual Learning (Học Tập Liên Tục)*

Học liên tục là khái niệm học mô hình cho một số lượng lớn bài học/nhiệm vụ một cách tuần tự mà không quên kiến ​​thức thu được từ các bài học/nhiệm vụ trước, trong đó dữ liệu trong các nhiệm vụ cũ không còn nữa trong quá trình đào tạo những cái mới.

Nói ngắn gọn Continual Learning tập trung vào việc phát triển các mô hình có khả năng học liên tục từ dữ liệu mới mà không quên những gì đã học trước đó. Điểm quan trọng của Continual Learning là giải quyết vấn đề "quên kiến thức" (catastrophic forgetting), nơi mô hình mất đi khả năng thực hiện tốt trên dữ liệu cũ khi được huấn luyện trên dữ liệu mới.

**Đặc điểm của Học Liên Tục:**

* Khả năng Thích Ứng: Mô hình có thể thích ứng với những thay đổi trong dữ liệu hoặc môi trường.
* Tránh Quên Kiến Thức: Mô hình duy trì hiệu suất trên dữ liệu cũ khi học dữ liệu mới.
* Học Từ Dữ Liệu Liên Tục: Khả năng học từ dòng dữ liệu không ngừng và không cố định.

**Những thách thức chính của Học Liên Tục:**

* Không biết được chính xác vấn đề :

Nếu chúng ta đang làm việc trong lĩnh vực Học máy ứng dụng, việc xác định vấn đề là cực kỳ quan trọng vì nó sẽ ảnh hưởng đến các quyết định ta đưa ra về công nghệ, nguồn dữ liệu và những người sẽ tham gia vào quá trình phát triển sản phẩm.

Vấn đề này giải thích tại sao chỉ có 20% lập trình viên AI của các tập đoàn lớn là đạt đến giai đoạn sản xuất và nhiều người trong số họ không phục vụ được nhu cầu của người tiêu dùng như dự định trong quá trình phát triển. Có hai khả năng cho thống kê này: một là vấn đề sai đang được giải quyết, và khả năng còn lại là không phải tất cả các biến đều được tính đến trong quá trình phát triển.

* Thu thập dữ liệu cần thiết

Khi nói đến Học máy liên tục, một trong những khía cạnh khó khăn nhất là thu thập và sắp xếp dữ liệu cần thiết để đào tạo mô hình. Điều này ngược lại với nghiên cứu khoa học, nơi dữ liệu đào tạo thường có thể truy cập được và mục đích là phát triển mô hình Machine Learning phù hợp nhất.

Theo Real World AI của Rochwerger và Pang, "khi phát triển Trí tuệ nhân tạo trong thế giới thực, dữ liệu được sử dụng để đào tạo mô hình quan trọng hơn nhiều so với chính mô hình đó." Trong giới học thuật, các Tiến sĩ Khoa học Dữ liệu dành phần lớn thời gian và công sức để phát triển các mô hình mới. Tuy nhiên, trong trường hợp Học máy liên tục thì ngược lại. Dữ liệu được sử dụng để đào tạo các mô hình trong học thuật chỉ nhằm mục đích chứng minh rằng mô hình đó hoạt động hiệu quả, không phải để giải quyết các vấn đề trong thế giới thực Việc thu thập dữ liệu chính xác và chất lượng cao có thể được sử dụng để đào tạo một mô hình AI hoạt động là rất khó thực hiện trong thế giới thực.

* Duy trì các mô hình Machine Learning

Như đã nêu trước đó, các mô hình Machine Learning hoạt động trong môi trường dữ liệu động, nơi dữ liệu liên tục thay đổi. “Sự sai lệch khái niệm” có thể xảy ra, điều này sẽ ảnh hưởng tiêu cực đến độ chính xác của mô hình nếu nó không được sửa chữa, đó là lý do tại sao nó được gọi là học tập “liên tục”.

Thật không may, AI cần có sự can thiệp của con người, bảo trì, quản lý liên tục và điều chỉnh hướng đi để mang lại kết quả có ý nghĩa. Ví dụ, trong đại dịch COVID-19, nhiều mẫu máy gặp trục trặc trong quá trình ngừng hoạt động do có sự thay đổi lớn so với tiêu chuẩn.

Theo Harvard Business Review, tác động đến hành vi của người tiêu dùng đã tạo ra một vấn đề không lường trước được: lỗ hổng thông tin, vì dữ liệu được thu thập trước cuộc khủng hoảng, không còn có thể được sử dụng để dự đoán chính xác các mô hình trong tương lai. Thành phần quan trọng cho các chương trình khách hàng thân thiết của khách hàng bán lẻ, đề xuất sản phẩm do AI điều khiển và một loạt các lựa chọn kinh doanh quan trọng có vấn đề nghiêm trọng về chất lượng. Vì điều này, các mô hình ML được yêu cầu phải đào tạo lại.

Tóm lại, việc đảm bảo rằng bạn có sẵn cơ sở hạ tầng và quy trình để thu thập và cập nhật các mô hình của mình một cách liên tục là chìa khóa để đảm bảo rằng mô hình Machine Learning của bạn sẽ có hiệu quả về lâu dài.

Các bộ dữ liệu công khai không hữu ích cho các mô hình đào tạo trong nhiều ứng dụng Machine Learning được áp dụng, chẳng hạn như Học máy liên tục, vì chúng không được quản lý tốt. Bạn thu thập dữ liệu của riêng mình hoặc bạn mua dữ liệu đó từ bên thứ ba. Không có lựa chọn nào là lý tưởng. Cả hai lựa chọn thay thế đều có những khó khăn riêng.

**Sự quan trọng của việc Học Liên Tục:**

Đào tạo liên tục định kỳ giúp mô hình ML cập nhật dữ liệu mới nhất.

Các mô hình ML cần được đào tạo lại một cách thường xuyên. Tuy nhiên, nếu không có sự khác biệt về khái niệm hoặc lý do quan trọng để đào tạo lại, chẳng hạn như trong đại dịch nêu trên, thì điều này có thể rất tốn kém về lâu dài.

Đôi khi các mô hình ML giảm xuống dưới ngưỡng chấp nhận được.

Một vấn đề lớn với điều này là phải mất một thời gian để xác định sự thật cơ bản, hay còn gọi là dữ liệu chính xác.

Dữ liệu có thể trở nên quá khác so với dữ liệu mà mô hình ML được đào tạo ban đầu.

Để ngăn điều này xảy ra, điều cần thiết là phải cập nhật cho nhóm hoặc cá nhân biết về dữ liệu đầu vào ban đầu.

Tóm lại, Học máy liên tục phá vỡ khái niệm AI thu hẹp và cung cấp khả năng cho máy móc thu thập dữ liệu và tinh chỉnh dữ liệu, giống như con người sẽ làm trong suốt cuộc đời của họ.

Một số thách thức đi kèm với Học máy liên tục, chẳng hạn như tìm ra vấn đề cần giải quyết, thu thập dữ liệu cần thiết và duy trì các mô hình ML. Tuy nhiên, khả năng diễn giải dữ liệu mới và thay đổi đầu ra theo xu hướng và thông tin mới là một phần quan trọng trong sự tiến bộ của Trí tuệ nhân tạo.

*Test Production (Kiểm Thử Trong Môi Trường Sản Xuất)*

Test Production, hay kiểm thử trong môi trường sản xuất, là quá trình kiểm tra và đánh giá các ứng dụng, hệ thống, hoặc mô hình AI trong một môi trường thực tế, nơi chúng sẽ được triển khai và sử dụng. Mục tiêu là đảm bảo rằng sản phẩm hoạt động chính xác, hiệu quả và an toàn trong điều kiện thực tế.

**Đặc điểm của Test Production:**

* Môi Trường Thực Tế: Kiểm thử trong điều kiện và môi trường sử dụng thực tế.
* Đánh Giá Hiệu Suất: Đảm bảo rằng sản phẩm hoạt động như mong đợi khi đối mặt với dữ liệu thực tế và tình huống không dự đoán trước.
* Phát Hiện và Giải Quyết Lỗi: Tìm và khắc phục các sự cố không được phát hiện trong quá trình kiểm thử trước đó.

Kiểm thử trong ngành công nghiệp phần mềm là một lĩnh vực được nghiên cứu và phát triển kỹ lưỡng. Những kinh nghiệm tốt đã học được từ vô số dự án thất bại giúp chúng ta phát hành những bản vá thường xuyên và có ít cơ hội hơn để nhìn thấy những khiếm khuyết trong sản xuất.

Một mô hình bao gồm mã (thuật toán, tiền xử lý, hậu xử lý, v.v.), dữ liệu và cơ sở hạ tầng tạo điều kiện thuận lợi cho thời gian chạy.

Các loại thử nghiệm khác nhau bao gồm việc đảm bảo chất lượng cho các thành phần khác nhau của hệ thống.

**Kiểm tra dữ liệu**: đảm bảo dữ liệu mới đáp ứng các giả định của bạn. Việc kiểm tra này là cần thiết trước khi chúng tôi đào tạo một mô hình và đưa ra dự đoán. Trước khi đào tạo mô hình, X và y (nhãn)

**Kiểm tra đường ống(Pipeline)**: đảm bảo đường ống của bạn được thiết lập chính xác. Đối với hệ thống ML, nó cũng có thể đo lường tính nhất quán (khả năng tái tạo).

**Đánh giá mô hình**: đánh giá đường ống ML của bạn tốt như thế nào. Tùy thuộc vào số liệu và tập dữ liệu bạn đang sử dụng, nó có thể đề cập đến những thứ khác nhau.

* Đánh giá về tập dữ liệu nắm giữ/xác thực chéo.
* Đánh giá các đường ống đã triển khai và thực tế cơ bản (đánh giá liên tục).
* Đánh giá dựa trên phản hồi của người dùng hệ thống (các số liệu liên quan đến doanh nghiệp, không phải proxy ML có thể đo lường được)

Có một loạt các kỹ thuật có thể được áp dụng trong quy trình, như đánh giá dựa trên lát cắt, phân tích nhóm/mẫu MVP (một tập hợp con quan trọng), nghiên cứu cắt bỏ, thử nghiệm dựa trên nhóm con người dùng (như thử nghiệm Beta và A/B thử nghiệm).

**Thử nghiệm mô hình**: bao gồm việc kiểm tra rõ ràng các hành vi mà chúng ta mong đợi mô hình của mình tuân theo. Loại thử nghiệm này không phải để cho chúng tôi biết các chỉ số liên quan đến độ chính xác mà để ngăn chúng tôi xử lý không tốt trong quá trình sản xuất. Các loại thử nghiệm phổ biến bao gồm, nhưng không giới hạn ở:

* Kiểm tra tính bất biến (nhiễu loạn): nhiễu loạn của đầu vào mà không ảnh hưởng đến đầu ra của mô hình.
* Kiểm tra kỳ vọng định hướng: để đạt được mục tiêu, chúng ta phải có tác động có thể dự đoán được đối với đầu ra của mô hình. Ví dụ, nếu lượng máu mất trong một cuộc phẫu thuật tăng lên thì lượng máu để truyền cũng sẽ tăng lên.
* Hồi quy điểm chuẩn: sử dụng các mẫu được xác định trước và cổng chính xác để đảm bảo phiên bản của mô hình sẽ không gây ra các vấn đề điên rồ.
* Kiểm tra overfitting : điều chỉnh overfit của mô hình với một phần nhỏ của tập dữ liệu đầy đủ và xác nhận xem mô hình có thể ghi nhớ dữ liệu hay không.

Quy trình Test Production :

**Đánh giá mô hình: Công cụ và số liệu Imp**

Đánh giá mô hình là giai đoạn 0 của thử nghiệm mô hình và chỉ giới hạn ở chức năng của mô hình. Dưới đây là số liệu cho từng loại mô hình để đánh giá chất lượng của nó. Đối với các tập dữ liệu không cân bằng, điểm F1 và điểm AUC là tốt nhất để phân loại và đối với dữ liệu nặng ngoại lệ, MAE sẽ tốt hơn cho hồi quy. Đối với một số mô hình như mô hình giám sát dựa trên cây quyết định tổng hợp (XGBoost, RF). SHAP có thể được sử dụng để hiểu logic dự đoán như mô hình hộp đen ở chế độ xem tổng thể và LIME để kiểm tra các trường hợp cụ thể là gì

**Kiểm tra trước đào tạo/Kiểm tra đơn vị**

Khi chúng tôi áp dụng phương pháp học trực tuyến hoặc học theo đợt, trước quá trình đào tạo, một số thử nghiệm đơn vị sẽ được thực hiện để đáp ứng các yêu cầu về phần mềm/mô hình như đúng định dạng dữ liệu, đủ dữ liệu, v.v. như dưới đây

**Bài kiểm tra sau đào tạo - Học hàng loạt và học trực tuyến**

Kiểm tra sau đào tạo được thực hiện sau đào tạo theo đợt nhưng trong quá trình đào tạo học trực tuyến

Kiểm tra độ trễ: Kiểm tra xem dự đoán có được thực hiện trong vòng một phần giây hay không để mô hình có thể mở rộng và xử lý lưu lượng. Nếu mất ≥ một phút thì phần lớn thiết kế mô hình cần phải thay đổi. Bài kiểm tra này rất quan trọng nếu nó là một phương pháp học máy trực tuyến. Một kỹ thuật để giải quyết vấn đề này trong học tập trực tuyến là đặt siêu tham số làm giá trị tĩnh và không sử dụng cv tìm kiếm dạng lưới để điều chỉnh tham số mỗi khi có dữ liệu mới vì nó có thể làm tăng độ trễ. Do đó, có thể tìm thấy giá trị tĩnh của siêu tham số bằng cách chạy tập dữ liệu được lấy mẫu/tập hợp con và huấn luyện mô hình về các kỹ thuật như cv tìm kiếm ngẫu nhiên để có được tham số tốt nhất dựa trên dữ liệu xác thực đa số

Kiểm tra tải: Kiểm tra xem mô hình có thể xử lý bao nhiêu dữ liệu kiểm tra cùng một lúc. Điều này rất quan trọng cho cả việc học theo đợt và học trực tuyến. Một kỹ thuật sử dụng SQLalchemy để kiểm tra xem tất cả DB (cơ sở dữ liệu) có được truy cập song song hay không, điều này sẽ cho phép xử lý lượng lớn dữ liệu. Ngoài ra, bộ chứa AWS còn giúp giám sát CPU/bộ nhớ. Locust là một công cụ để tự động hóa việc kiểm tra này.

**Thử nghiệm A/B: Đào tạo lại mô hình**

Trong một số mô hình ML/AI, theo thời gian, các đặc điểm dữ liệu thay đổi, do đó mô hình được đào tạo trên dữ liệu cũ có thể không hoạt động tốt trong dữ liệu mới. Điều này được gọi là trôi dạt dữ liệu. Tương tự, hiện tượng lệch khái niệm xảy ra khi các giả định do mô hình học máy đưa ra không còn đúng trong dữ liệu trong thế giới thực mà nó gặp trong quá trình triển khai.

Do đó, cần phải đào tạo lại sau khi triển khai ML. Thử nghiệm A/B giúp quyết định xem phiên bản mới của mô hình có nên thay thế mô hình cũ hay không.

Trong thử nghiệm A/B (được AWS Sagemaker hỗ trợ tự động), 80% lưu lượng truy cập được cung cấp cho mô hình ml hiện tại/cũ trong khi 20% lưu lượng truy cập được cung cấp cho mô hình mới. Dựa trên (các) chỉ số cơ bản, mô hình mới có thể thay thế mô hình cũ nếu mô hình mới hoạt động tốt hơn.

**Kiểm tra giai đoạn/Kiểm tra bóng**

Giai đoạn thử nghiệm là một trong những thử nghiệm cuối cùng để kiểm tra xem mô hình có cho kết quả đầu ra như mong muốn hay không. Thử nghiệm này diễn ra sau quá trình dockerization và AWS containerization để triển khai tự động trong các quy trình như Bitbucket và GitLab. Điều này bao gồm dữ liệu thử nghiệm đa dạng (bao gồm các kịch bản mở rộng) nắm bắt các đặc điểm dữ liệu trong thế giới thực và được cung cấp làm đầu vào cho mô hình ở môi trường giai đoạn.

Kiểm tra bóng (Cách an toàn để kiểm tra độ tỉnh táo khi triển khai các mô hình ML/AI lớn):

* Triển khai mô hình song song với mô hình hiện có (nếu đã có sẵn).
* Đối với mỗi yêu cầu, hãy định tuyến nó đến cả hai mô hình để đưa ra dự đoán nhưng chỉ phân phát yêu cầu hiện có cho người dùng
* Sử dụng dự đoán trên mô hình mới để đánh giá/phân tích

**Kiểm tra API**

Đây là thử nghiệm cuối cùng, nơi chúng tôi kiểm tra xem người dùng thực tế sẽ thấy phản hồi từ mô hình như thế nào. Do đó, nó nắm bắt tất cả các trường hợp có thể xảy ra trong đầu vào của người dùng có thể gây ra lỗi khi phản hồi. Mỗi lỗi được mã hóa dưới dạng một Id duy nhất (dưới dạng status\_code, mã lỗi). Và khía cạnh bảo mật của đầu vào và đầu ra được kiểm tra cho từng khách hàng riêng biệt.

**Kết luận**

Khi kết hợp cả Continual Learning và Test Production, chúng ta có thể tạo ra một hệ thống học máy không chỉ học hỏi và thích ứng liên tục với dữ liệu và môi trường mới mà còn đảm bảo hoạt động hiệu quả và chính xác trong môi trường sản xuất thực tế. Điều này giúp giải pháp không chỉ thông minh và cập nhật mà còn đáng tin cậy và ổn định khi triển khai.