Flight Delay Forecasting Real-times using Big Data Technology

Lê Huy Quang, *20521804;* Đỗ trọng Hợp, *hopdt@uit.edu.vn;*

Tóm tắt nội dung

Trong nghiên cứu này, chúng tôi đã phát triển một hệ thống dự đoán độ trễ chuyến bay theo thời gian thực. Chúng tôi xây dựng và sử dụng bộ dữ liệu Flight Delays Dataset và các phương pháp học máy để dự đoán khả năng trễ của các chuyến bay. Chúng tôi xây dựng pipeline để xử lý dữ liệu và sử dụng các thuật toán Machine Learning được hỗ trợ bởi Machine Learning Library (MLlib) là một thư viện máy học của Spark để xây dựng mô hình dự đoán cho bài toán và so sánh kết quả đạt được thông qua các độ đo: Precision, Recall, Accuracy và F1– score. Kết quả tốt nhất theo độ đo F1-macro là 45.27 sử dụng Decision Tree. Đặc biệt, hệ thống này được thiết kế để hoạt động trong thời gian thực, giúp các hãng hàng không có thể dự đoán và quản lý thời gian bay một cách hiệu quả hơn. Nghiên cứu này mở ra khả năng cải thiện quản lý thời gian và nâng cao uy tín của các hãng hàng không thông qua việc ứng dụng các mô hình dự đoán tiên tiến.

Index Terms—FlightDelay, BigData

1 Giới thiệu

Trễ chuyến bay là một vấn đề nghiêm trọng trong ngành hàng không và vận tải, ảnh hưởng trực tiếp đến lịch trình của hành khách cũng như uy tín của các hãng hàng không. Mặc dù khoảng 80 phần trăm các chuyến bay diễn ra đúng giờ, vẫn còn tới 20 phần trăm chuyến bay bị chậm trễ. Tình trạng này không chỉ gây bực mình vì phải chờ đợi lâu mà còn làm lỡ kế hoạch của hành khách. Có nhiều nguyên nhân khiến chuyến bay khởi hành trễ so với thời gian dự kiến, như máy bay về muộn, thời tiết xấu, vấn đề an ninh và sự cố kỹ thuật. Việc dự đoán chính xác khả năng trễ và mức độ trễ của các chuyến bay có thể giúp các hãng hàng không quản lý lịch trình hiệu quả hơn, từ đó nâng cao chất lượng dịch vụ và sự hài lòng của khách hàng

Dự đoán độ trễ chuyến bay là một bài toán dựa trên phân tích các thông tin trước khi máy bay khởi hành để trích xuất những thông tin hữu ích, sau đó sử dụng các mô hình toán học để dự đoán khả năng và mức độ trễ. Mục tiêu của bài toán này là hỗ trợ các hãng hàng không dự đoán trước khả năng và mức độ trễ của chuyến bay, giúp họ xử lý các tình huống và thông báo kịp thời đến khách hàng, nâng cao trải nghiệm người dùng và cạnh tranh với các hãng khác.

Trong báo cáo này, chúng tôi sẽ trình bày phương pháp xây dựng bộ dữ liệu Flight Delays Dataset trong Phần 2. Phần 3 mô tả chi tiết hướng tiếp cận và Phần 4 trình bày quá trình xây dựng pipeline thực nghiệm các mô hình Machine Learning trên bộ dữ liệu, phân tích kết quả để tìm ra mô hình tốt nhất và các hạn chế của nó. Phần 5 mô tả quá trình xây dựng hệ thống giả lập dự đoán độ trễ chuyến bay theo thời gian thực. Cuối cùng, chúng tôi rút ra kết luận ở Phần 6.

2 Bộ dữ liệu

2.1 Thu thập và tiền xử lí dữ liệu

Bộ dữ liệu sử dụng trong bài toán dự đoán độ trễ chuyến bay được lấy từ website Bureau of Transportation Statistics (BTS) 6. BTS là một bộ phận thuộc Department of Transportation (DOT) là nguồn thống kê ưu việt về hàng không thương mại. Dữ liệu về thông tin các chuyến bay được cung cấp theo từng tháng với trung bình gần nửa triệu thông tin chuyến bay trong một tháng. Vì số lượng dữ liệu khá lớn và bị giới hạn về phần cứng nên chúng tôi chỉ sử dụng một vài tháng đại diện cho các quý trong năm để tạo ra bộ dữ liệu huấn luyện mô hình. Chúng tôi sử dụng dữ liệu thông tin các chuyến bay của tháng 6/9/12–2021, tháng 3/2022 để huấn luyện mô hình và dữ liệu thông tin các chuyến bay của tháng 4/2022 để treaming giả lập real–time cho hệ thống. Chúng tôi đã thu thập dữ liệu của 4 tháng kể trên và thu được bộ dữ liệu chứa 2,200,913 dữ liệu thông tin các chuyến bay và có nhiều chuyến bay bị missing values. Vì bộ dữ liệu có missing values, nên chúng tôi tiến hành các bước tiền xử lý để xử lý missing values. Với các chuyến bay có missing values, chúng tôi tôi sẽ xóa bỏ thông tin của chuyến bay đó. Thuộc tính DEP-DELAY được chúng tôi sử dụng làm thuộc tính dự đoán cho bài toán, nó chứa các giá trị liên tục là sự chênh lệch về số phút giữa thời gian khỏi hành dự kiến và thực tế của các chuyến bay. Để phù hợp với mục tiêu ban đầu và đơn giản hóa bài toán chúng tôi tiến hành chuẩn hóa thuộc tính DEPDELAY thành 3 nhãn như

sau: với các chuyến bay có thời gian khởi hành thực tế sớm hoặc đúng giờ sẽ được gán nhãn "0", các chuyến bay có thời gian khởi hành thực tế muộn không quá 30 phút sẽ được gán nhãn "1", còn lại các chuyến bay có thời gian khởi hành thực tế muộn hơn 30 phút hoặc có thể bị hủy chuyến sẽ được gán nhãn "2". Thuộc tính DEP-DELAY cũng được chúng tôi đổi tên thành LABEL để phù hợp với dữ liệu và bài toán. Sau khi xử lý xong, chúng tôi thu được bộ dữ liệu Flight Delays Dataset chứa 2,157,737 dòng dữ liệu và 11 thuộc tính là các thông tin chuyến bay. Tương tự, chúng tôi tạo tiếp bộ dữ liệu để streaming giả lập real–time cho hệ thống chứa 542,117 dòng dữ liệu và 11 thuộc tính.

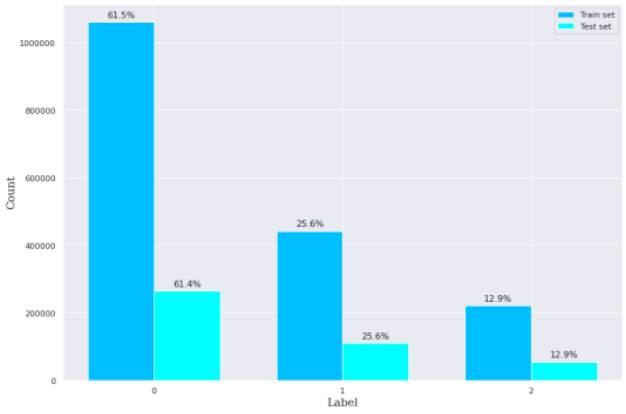
2.2 Phân tích dữ liệu

Thông tin chi tiết các thuộc tính của bộ dữ liệu được thể hiện tại Hình 1

Index	Thuộc tính	Ý nghĩa
0	ID	ID của chuyến bay.
1	QUARTER	Quý trong năm (1-4).
2	MONTH	Tháng trong năm (1-12).
3	DAY_OF_MONTH	Ngày trong tháng (1-31).
4	DAY_OF_WEEK	Ngày trong tuần (1-7).
5	OP_UNIQUE_CARRIER	Mã nhà cung cấp dịch vụ.
6	ORIGIN	Sân bay xuất phát.
7	DEST	Sân bay đến.
8	DISTANCE	Khoảng cách giữa các sân bay (dặm).
9	CRS_DEP_TIME	Giờ khởi hành dự kiến.
10	LABEL	Output của bài toán, gồm 3 nhãn: – 0: Không trễ. – 1: Trễ không quá 30 phút. – 2: Trễ hơn 30 phút hoặc hủy chuyến.

Hình 1: Bảng thông tin chi tiết các thuộc tính

Hình 1 thể hiện phân phối và tỉ lệ của ba nhãn trong tập train/test của bộ dữ liệu Flight Delays Dataset của chúng tôi. Ta có thể thấy phân phối các nhãn trên tập train và test là tương đồng nhau với các tỉ lệ nhãn 0 – 1 – 2 lần lượt là 61.5% – 25.6% – 12.9%. Bộ dữ liệu của chúng tôi có xu hướng bị mất cân bằng dữ liệu (imbalance data) khi mà số lượng nhãn 0 chiếm 61.5% của bộ dữ liệu, gấp 2.4 lần nhãn 1 (25.6%) và gấp 4.8 lần nhãn 2 (12.9%). Đây cũng là một thách thức của bộ dữ liệu



Hình 2: Thông tin dữ liệu train test

3 Hướng tiếp cận

Từ bộ dữ liệu Flight Delays Dataset mà chúng tôi đã tạo ra ở Phần 2, chúng tôi thực hiện các bước chuẩn hóa dữ liệu trước khi huấn luyện mô hình như: String Indexer, One Hot Encoding, Vector Assembler, Data Standardization (được trình bày rõ hơn tại Phần 3.1). Để tạo ra mô hình dự đoán độ trễ chuyến bay, chúng tôi tiến hành xây dựng các mô hình Máy Học (Machine Learning) như: Logistic Regression, Decision Tree Classifier, Random Forest Classifier, Naive Bayes được hỗ trợ bởi Machine Learning Library (MLlib) là một thư viện máy học của Spark. Để đánh giá mô hình chúng tôi sử dụng 4 độ đo là: Precision, Recall, Accuracy và F1–score (chi tiết Phần 3.3). Từ việc đánh giá mô hình, chúng tôi sẽ chọn ra mô hình tốt và khả thi nhất để tiến hành xây dựng hệ thống giả lập real–time dự đoán độ trễ chuyến bay của chúng tôi.

3.1 Chuẩn hóa dữ liệu

Chuẩn hóa dữ liệu (Data Normalization) là một khái niệm chung đề cập đến hành động chuyển đổi các giá trị ban đầu của tập dữ liệu thành các giá trị mới. Các giá trị mới thường được mã hóa liên quan đến chính tập dữ liệu và được chia tỷ lệ theo một cách nào đó.

String Indexer : Mã hóa một cột chuỗi nhãn thành một cột chỉ số nhãn. String Indexer có thể mã hóa nhiều cột. Nếu cột đầu vào là số, nó sẽ được ép kiểu thành kiểu chuỗi và lập chỉ mục các giá trị của chuỗi. Các chỉ số nằm trong [0, numLabels). Theo mặc định, điều này được sắp xếp theo tần số nhãn vì vậy nhãn thường xuyên nhất nhận được chỉ số 0.

One Hot Encoding: Ánh xạ một đặc trưng phân loại (categorical feature), được biểu thị dưới dạng nhãn chỉ mục thành một vector nhị phân có nhiều nhất một giá trị duy nhất cho biết sự hiện diện của một giá trị đặc trưng cụ thể trong số tất cả các giá trị đặc trưng. Mã hóa này cho phép các thuật toán mong đợi các đặc trưng liên tục chẳng hạn như Logistic Regression sử dụng các đặc trưng phân loại. Đối với dữ liệu đầu vào kiểu chuỗi, người ta thường mã hóa các đặc trưng phân loại bằng cách sử dụng String Indexer trước. OneHotEncoder có thể chuyển đổi nhiều cột, trả về một cột vector đầu ra được one-hot-encoded cho mỗi cột đầu vào. Thông thường, hợp nhất các vector này thành một vector đặc trưng duy nhất bằng cách sử dụng Vector Assembler.

Vector Assembler: Là một cách biến đổi kết hợp một danh sách các cột nhất định thành một cột vector duy nhất. Nó rất hữu ích để kết hợp các đặc trưng thô và các đặc trưng được tạo bởi các biến đổi đặc trưng khác nhau thành một vector đặc trưng duy nhất, để đào tạo các mô hình ML như Logistic Regression và Decision Trees. Vector Assembler chấp nhận các kiểu cột đầu vào sau: tất cả các kiểu số, kiểu boolean và kiểu vector. Trong mỗi hàng, giá trị của các cột đầu vào sẽ được nối thành một vector theo thứ tự được chỉ định.

3.2 Machine Learning Algorithmi

Hồi quy logistic (LR): là một quá trình mô hình hóa xác suất của một kết quả rời rạc cho một biến đầu vào. Hồi quy logistic là một phương pháp phổ biến để dự đoán một bài toán phân loại. Đây là một trường hợp đặc biệt của mô hình Tuyến tính tổng quát (Generalized Linear models) dự đoán

xác suất của các kết quả. Trong hồi quy logistic spark.ml có thể được sử dụng để dự đoán kết quả phân loại nhị phân bằng cách sử dụng hồi quy logistic nhị thức (binomial logistic regression) hoặc nó có thể được sử dụng để dự đoán kết quả phân loại đa lớp bằng cách sử dụng hồi quy logistic đa thức (multinomial logistic regression). Phân loại đa lớp (Multiclass classification) được hỗ trợ thông qua hồi quy logistic đa thức (softmax). Trong hồi quy logistic đa thức, thuật toán tạo ra K bộ hệ số hoặc ma trận có kích thước $K \times J$ trong đó K là số lớp kết quả và J là số đặc trưng. Nếu thuật toán phù hợp với một số hạng chặn (intercept) thì độ dài K vector của các điểm chặn có sẵn.

Xác suất có điều kiện của các lớp kết quả $k \in 1, 2, \ldots, K$ được mô hình hóa bằng cách sử dụng hàm softmax.

$$P(Y = k|X, \beta_k, \beta_{0k}) = \frac{e^{\beta_k \cdot X + \beta_{0k}}}{\sum_{k'=1}^{K} e^{\beta_{k'} \cdot X + \beta_{0k'}}}$$

Decision Tree (DT): là một họ phổ biến của các phương pháp phân loại và hồi quy. Cây quyết định và tập hợp của chúng là các phương pháp phổ biến cho các tác vụ phân loại và hồi quy của máy học. Cây quyết định được sử dụng rộng rãi vì chúng dễ diễn giải, xử lý các đặc trưng phân loại, mở rộng sang cài đặt phân loại đa lớp, không yêu cầu tỷ lệ đối tượng và có thể nắm bắt các tương tác đối tượng và phi tuyến tính. Cây quyết định là một thuật toán tham lam thực hiện phân vùng nhị phân đệ quy của không gian đặc trưng. Cây dự đoán cùng một nhãn cho mỗi phân vùng (lá) dưới cùng. Mỗi phân vùng được chọn một cách tham lam bằng cách chọn phần tách tốt nhất từ một tập hợp các phần tách có thể có, để tối đa hóa độ lợi thông tin (information gain) tại một nút cây

Random Forest (RF): là một nhóm phương pháp phân loại và hồi quy phổ biến. Rừng ngẫu nhiên là quần thể của các cây quyết định. Rừng ngẫu nhiên kết hợp nhiều cây quyết định, tuy nhiên mỗi cây quyết định sẽ khác nhau (có yếu tố random). Sau đó kết quả dự đoán được tổng hợp từ các cây quyết định, nhãn của đầu ra là nhãn được dự đoán nhiều nhất (chiếm đa số) trong tất cả các cây quyết định để giảm nguy cơ bị quá khớp (overfitting). Việc triển khai spark.ml hỗ trợ rừng ngẫu nhiên để phân loại nhị phân và đa lớp và cho hồi quy, sử dụng cả các đặc trưng liên tục và phân loại.

Naive Bayes (NB): là một thuật toán phân loại dựa trên tính toán xác suất áp dụng định lý Bayes. Thuật toán này thuộc nhóm học có giám sát (supervised learning). Đây là hướng tiếp cận phân lớp theo mô hình xác suất. Dự đoán xác suất một đối tượng mới thuộc về thành viên của lớp đang xét. Naive Bayes có thể được huấn luyện rất hiệu quả. Với một lần huấn luyện qua dữ liệu, nó sẽ tính toán phân phối xác suất có điều kiện của từng đặc trưng cho từng nhãn. Để dự đoán, nó áp dụng định lý Bayes để tính toán phân phối xác suất có điều kiện của mỗi nhãn cho một quan sát.

Theo định lý Bayes, ta có công thức tính xác suất ngẫu nhiên của sự kiện Y khi biết X như sau:

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)}$$

3.3 Apache Kafka

Apache Kafka: là một hệ thống giới thiệu thông tin (messaging system) mã nguồn mở, được phát triển bởi Apache Software Foundation. Nó được thiết kế để xử lý dòng dữ liệu lớn và phân tán qua nhiều máy chủ. Kafka thường được sử dụng để xây dựng các ứng dụng xử lý dữ liệu thời gian thực, như xử lý và phân tích logs, sự kiện, và dữ liệu từ nhiều nguồn khác nhau. Kafka thực hiện mô hình công bố-đăng ký, nơi các ứng dụng có thể gửi (publish) và nhận (subscribe) các thông điệp thông qua các chủ đề (topics). Dữ liệu trong Kafka được lưu trữ dưới dạng các thông điệp trong các partition trên nhiều máy chủ, giúp đảm bảo tính nhất quán và khả năng mở rộng.Kafka được thiết kế để chịu lỗi, có thể xử lý mất mát dữ liệu và tiếp tục hoạt động ngay cả khi một số máy chủ gặp sự cố, đảm bảo tính nhất quán của dữ liệu giữa các producer và consumer, và cung cấp đảm bảo giao hàng (delivery guarantee) để đảm bảo rằng thông điệp được đưa đến đúng nơi đích.

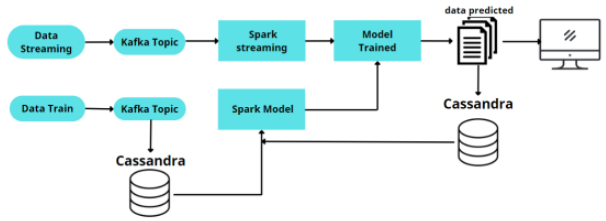
3.4 Apache Cassandra

Apache Cassandra: Nói về phần mềm trung gian NoSQL dẫn đầu là Google BigTable và Amazon Dynamo, nhưng nhiều thứ khác cũng xuất hiện trong thế giới open source. Trong số đó, Apache Cassandra (Cassandra) đang thu hút sự chú ý đặc biệt gần đây. Cassandra ban đầu được tạo ra bởi Facebook (hiện tại đã đổi tên thành Meta). Sau đó nó đã được tặng cho Apache Foundation vào tháng 2 năm 2010 và được nâng cấp lên thành dự án hàng đầu của Apache. Cassandra là một cơ sở dữ liệu phân tán kết hợp mô hình dữ liệu của Google Bigtable với thiết kế hệ thống phân tán như bản sao của Amazon Dynamo. Hình 7 cho thấy các đặt điểm của Cassandra. Các tính năng ưu việt của Cassandra bao gồm các điểm sau: Thích hợp để sử dụng thực tế, có khả năng chịu lỗi cao, kiến trúc không có SPOF (Single point of failure), mức độ tự do kiểm soát nhất quán, mô hình dữ liệu phong phú, Fast Linear-scale Performance, Cassandra tăng thông lượng vì chúng tạo điều kiện cho chúng ta tăng số lượng nút trong cụm. Do đó, Cassandra duy trì thời gian phản hồi nhanh chóng. Tính khả dụng cao, hỗ trợ các ngôn ngữ khác nhau dưới dạng client code, dễ dàng nắm bắt trạng thái bên trong của máy chủ bằng JMX (Java Management Extensions). Vì Bigdata có khối lượng rất lớn nên việc lưu trữ và quản lý dữ liệu trở nên khó khăn, nhờ vào các tính năng vượt trội của Cassandra chúng tôi sử dụng nó để quản lý dữ liệu lớn này. Mục đích của chúng tôi sử dụng Cassandra trong hệ thống này còn vì muốn tận dụng khả năng multi node, khi hai hay nhiều máy đều chứa dữ liệu cần thì có thể truy cập P2P (peer-to-peer) để truy vấn dữ liệu qua lại, từ đó có thể lấy dữ liệu từ nhiều node khác nhau, đó là bản chất của dữ liệu lớn vì nó có khối lượng rất lớn nên khả năng lưu trữ phân tán cao, dữ liệu khó mà tập trung tại 1 node.

4 KIẾN TRÚC HỆ THỐNG

Kiến trúc hệ thống của chúng tôi được mô tả ở hình **3**. Dữ liệu được tổng hợp từ nhiều nguồn khác nhau sẽ được Producer Kafka gửi vào Topic Kafka. Sau đó Consumer Kafka có vai trò chuyển nhận dữ liệu và truyền dữ liệu trực tiếp vào cơ sở dữ liệu Cassandra để lưu trữ. Tiếp theo, dữ liệu được load từ Cassandra để tiến hành huấn luyện các mô hình và chọn ra mô hình tốt nhất để sử dụng làm mô hình chính của hệ thống này. Kết quả các bước tiền xử lí dữ liệu

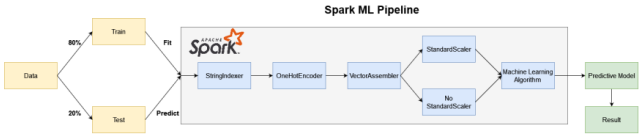
và thực nghiệm huấn luyện mô hình cho thấy mô hình mô hình Decision Tree đạt kết quả tốt nhất, nên chúng tôi sẽ sử dụng mô hình Decision Tree cho hệ thống này đạt kết quả tốt nhất, nên chúng tôi sẽ sử dụng mô hình LSTMForecaster cho hệ thống này. Chúng tôi sử dụng Apache Spark để load dữ liệu Streaming từ Cassandra để giả lập real-time. Dữ liệu sau khi được load ra sẽ được Producer Kafka gửi vào Topic Kafka, sau đó Consumer Kafka sẽ nhận dữ liệu liên tục từ Producer Kafka để đưa đến mô hình tiến hành quá trình dự đoán. Dữ liệu được dự đoán xong sẽ được lưu lại vào Cassandra để tiến hành cập nhật lại mô hình sau này. Đồng thời dữ liệu dự đoán cũng sẽ được đưa ra màn hình để người dùng có thể nhìn trực quan kết quả



Hình 3: Tổng quan kiến trúc hệ thống

5 Kết quả thực nghiệm và đánh giá

Chúng tôi tiến hành xây dựng một pipeline thực nghiệm các mô hình Machine Learning (ML): Logistic Regression (LR), Decision Tree (DT), Random Forest (RF), Naive Bayes (NB) sử dụng Framework Spark ML hình **4**



Hình 4: Spark ML pipeline

5.1 Độ đo đánh giá

Precision và Recall độ đo được sử dụng để đo lường hiệu suất của hệ thống truy xuất là precision và recall. Precision đo lường số lượng các trường hợp đúng được truy xuất chia cho tất cả các trường hợp được truy xuất, xem Công thức 7. Recall đo lường số lượng các trường hợp đúng được truy xuất chia cho tất cả các trường hợp đúng

$$\text{Precision} : P = \frac{TP}{TP + FP} \tag{1}$$

$$\text{Recall} : R = \frac{TP}{TP + FN} \tag{2}$$

F1–score được định nghĩa là giá trị trung bình có trọng số precision và recall tùy thuộc vào hàm trọng số , . F1–score có nghĩa là trung bình hài hòa giữa precision và recall, xem Công thức 10, khi nó được viết là F–score, nó thường có nghĩa là F1–score. F–score còn được gọi là F– Measure. F1–score có thể có các chỉ số khác nhau tạo ra các trọng số khác nhau về precision và recall Với $\beta = 1$, F–score tiêu chuẩn thu được, Công thức.

$$F1 = F = 2 \times \frac{P \times R}{P + R} \tag{3}$$

Accuracy là một độ đo khác được định nghĩa là tỷ lệ các trường hợp đúng được truy xuất, cả positive và negative, trong số tất cả các trường hợp được truy xuất. Accuracy là một trung bình có trọng số của precision và precision nghịch đảo. Accuracy càng cao, mô hình càng hiệu quả

5.2 Đánh giá kết quả

	Scale = None				Scale = StandardScaler			
	Train set		Test set		Train set		Test set	
	Accuracy	F1-macro	Accuracy	F1-macro	Accuracy	F1-macro	Accuracy	F1-macro
LR	63.33	36.23	63.32	36.24	63.34	36.23	63.32	36.25
DT	**67.24**	**49.04**	**64.95**	**45.28**	**66.56**	**47.34**	**64.86**	**44.58**
RF	61.52	25.62	61.49	25.60	61.90	27.14	61.86	27.13
NB	24.61	24.82	24.64	24.86	44.96	38.75	44.73	38.51

Hình 5: Kết quả

Để đảm bảo tính ổn định và tránh sự ngẫu nhiên của kết quả, mỗi mô hình được chạy 5 lần và kết quả trung bình được ghi lại. Kết quả cho thấy Logistic Regression (LR) hiệu suất không thay đổi khi áp dụng hoặc không áp dụng chuẩn hóa dữ liệu . Mô hình Decision Tree (DT) khi không áp dụng chuẩn hóa dữ liệu, hiệu suất cao hơn so với khi áp dụng chuẩn hóa bằng z-score (cao hơn 1.7% ở tập training và 0.69% ở tập testing), kết quả chênh lệch giữa tập traing và testing khá lớn. Mô hình Random Forest (RF) hiệu suất thấp hơn khi không áp dụng chuẩn hóa dữ liệu (thấp hơn 1.52% ở tập training và 1.54% ở tập testing), chênh lệch giữa tập training và testing không đáng kể , không dự đoán được nhãn "2" trong bộ dữ liệu. Mô hình Naive Bayes (NB) hiệu suất cao vượt trội khi áp dụng chuẩn hóa dữ liệu (cao hơn 13.92% ở tập training và 13.64% ở tập testing), chênh lệch giữa tập training và testing không đáng kể. Kết luận mô hình tốt nhất là Decision Tree (DT) không áp dụng chuẩn hóa dữ liệu với F1-macro test = 45.28%. Mô hình tốt thứ hai là Naive Bayes (NB) có áp dụng chuẩn hóa dữ liệu với F1-macro test = 38.51%. Mô hình tệ nhất là Random Forest (RF) với F1-macro test = 27.13% và không dự đoán được nhãn "2"

6 Kết luận và hướng phát triển

Trong báo cáo này, chúng tôi đã tiến hành xây dựng và thực nghiệm các mô hình Machine Learning để dự đoán độ trễ chuyến bay sử dụng dữ liệu từ Flight Delays Dataset. Các mô hình Logistic Regression (LR), Decision Tree (DT), Random Forest (RF), và Naive Bayes (NB) được triển khai trong môi trường Spark ML với việc thử nghiệm cả chuẩn hóa dữ liệu và không chuẩn hóa dữ liệu bằng z-score. Kết quả cho thấy mô hình Decision Tree (DT) không áp dụng chuẩn hóa dữ liệu cho hiệu suất tốt nhất với F1-macro test đạt 45.28%. Naive Bayes (NB) áp dụng chuẩn hóa dữ liệu bằng z-score cho kết quả tốt thứ hai với F1-macro test đạt 38.51%. Mô hình Random Forest (RF) có hiệu suất kém nhất với F1-macro test chỉ đạt 27.13% và không dự đoán được nhãn "2" trong bộ dữ liệu. Sự chênh lệch giữa các mô hình và ảnh hưởng của chuẩn hóa dữ liệu đã được phân tích

chi tiết, cho thấy tầm quan trọng của việc lựa chọn và điều chỉnh mô hình phù hợp với đặc thù của dữ liệu.

Hướng phát triển gồm cải thiện chất lượng dữ liệu như thu thập thêm dữ liệu từ nhiều nguồn khác nhau để tăng độ phong phú và đa dạng, áp dụng các kỹ thuật xử lý mất cân bằng dữ liệu như oversampling, undersampling, hoặc sử dụng các thuật toán cân bằng để cải thiện hiệu suất của mô hình đối với các nhãn ít xuất hiện.. Nâng cao mô hình bằng cách thử nghiệm thêm các mô hình Machine Learning tiên tiến hơn như Gradient Boosting, XGBoost, hoặc các mô hình dựa trên học sâu (Deep Learning).Thực hiện các kỹ thuật tối ưu hóa và tuning siêu tham số (hyperparameter tuning) để cải thiện hiệu suất của các mô hình. Áp dụng các phương pháp cross-validation nâng cao để đánh giá mô hình một cách chính xác hơn. Có thể triển khai ứng dụng vào thực tế như xây dựng và triển khai hệ thống dự đoán độ trễ chuyến bay theo thời gian thực để hỗ trợ các hãng hàng không trong việc quản lý và điều hành lịch bay hoặc thông qua thông qua các ứng dụng di động hoặc website của các hãng hàng không.

TÀI LIỆU

[1] X. Meng et al., “MLlib: Machine Learning in Apache Spark,” Journal of Machine Learning Research, vol. 17, pp. 1–7, 2016, Available: https://www.jmlr.org/papers/volume17/15-237/15-237.pdf

[2] 3. Taylor, Sean J., and Benjamin Letham. "Forecasting at scale." The American Statis- tician 72, no. 1 (2018): 37-45.

[3] On the relationship between selfattention and convolutional layers. In ICLR, 2020. Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi

[4] Momentum contrast for unsupervised visual representation learning. In CVPR, 2020. Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick

[5] “Understanding from Machine Learning Models | The British Journal for the Philosophy of Science: Vol 73, No 1,” The British Journal for the Philosophy of Science, 2022.

[6] Dai, Jason Jinquan, Ding Ding, Dongjie Shi, Shengsheng Huang, Jiao Wang, Xin Qiu, Kai Huang et al. "BigDL 2.0: Seamless Scaling of AI Pipelines from Laptops to Distributed Cluster." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 21439-21446. 2022

[7] Dai, Jason Jinquan, Yiheng Wang, Xin Qiu, Ding Ding, Yao Zhang, Yanzhang Wang, Xianyan Jia et al. "Bigdl: A distributed deep learning framework for big data." In Proceedings of the ACM Symposium on Cloud Computing, pp. 50-60. 2019.

[8] G. Varoquaux and Olivier Colliot, “Evaluating Machine Learning Models and Their Diagnostic Value,” Neuromethods, pp. 601–630, Jan. 2023