BỘ GIÁO DỤC VÀ ĐÀO TẠO TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN



KHOA CÔNG NGHỆ THÔNG TIN

NHẬP MÔN DỮ LIỆU LỚN

**BÁO CÁO BÀI TẬP MÔN HỌC NỘI DUNG: LAB 03 – MACHINE LEARNING ON SPARK**

**Giảng viên hướng dẫn** : Huỳnh Lâm Hải Đăng

**Lớp** : CQ2022/21

**Sinh viên thực hiện** : Trương Tiến Anh- 22120017 Đoàn Minh Cường- 22120043 Đinh Viết Lợi- 22120188 Nguyễn Trần Lợi-22120190

*Hồ Chí Minh, ngày 6 tháng 5 năm 2025*

# MỤC LỤC

[MỤC LỤC 1](#_Toc197795381)

[PHẦN 1: CLASSIFICATION WITH LOGISTIC REGRESSION 2](#_Toc197795382)

[I. Structured API Implementation (High Level) 2](#_Toc197795383)

[1. Ý tưởng thực hiện 2](#_Toc197795384)

[2. Mô tả chi tiết 2](#_Toc197795385)

[II. MLlib RDD-Based Implementation 8](#_Toc197795386)

[1. Ý tưởng thực hiện 8](#_Toc197795387)

[2. Mô tả chi tiết 8](#_Toc197795388)

[III. Low-Level Operations 13](#_Toc197795389)

[1. Ý tưởng thực hiện 13](#_Toc197795390)

[2. Mô tả chi tiết 13](#_Toc197795391)

[IV. So sánh các phương pháp giải quyết 18](#_Toc197795392)

[PHẦN 2: REGRESSION WITH DECISION TREES 19](#_Toc197795393)

[I. Structured API Implementation (High Level) 19](#_Toc197795394)

[1. Ý tưởng thực hiện 19](#_Toc197795395)

[2. Mô tả chi tiết 19](#_Toc197795396)

[II. MLlib RDD-Based Implementation 24](#_Toc197795397)

[1. Ý tưởng thực hiện 24](#_Toc197795398)

[III. Low-Level Operations 26](#_Toc197795399)

[1. Ý tưởng thực hiện 26](#_Toc197795400)

[2. Mô tả chi tiết 27](#_Toc197795401)

[IV. So sánh các phương pháp giải quyết 32](#_Toc197795402)

[PHẦN 3: BÁO CÁO NHÓM 32](#_Toc197795403)

# PHẦN 1: CLASSIFICATION WITH LOGISTIC REGRESSION

## Structured API Implementation (High Level)

### **Ý tưởng thực hiện**

* + - Structured API cung cấp lượng lớn các API giúp xử lý dữ liệu có scheme rõ ràng và hầu như không yêu cầu viết rất ít các hàm bổ trợ hay xử lý thủ công.
    - Tập dữ liệu được đưa ra cung cấp các giao dịch bằng thẻ tín dụng tại Châu Âu. Do yêu cầu về bảo mật riêng tư nên tên thuộc tính và giá trị tất cả các thuộc tính ngoại trừ `Thời gian giao dịch` và `Số tiền giao dịch` đều đã được chuẩn hoá và ẩn đi.
    - Một vấn đề lớn của tập dữ liệu này là hiện tượng mất cân bằng dữ liệu cực nặng khi mà chỉ có 492 trên 284,807 giao dịch là giao dịch lừa đảo.
    - Bài làm sẽ sử dụng Structured API với mục đích sau cùng là xây dựng một mô hình nhằm xác định xem giao dịch nào là lừa đảo. Quy trình sẽ bao gồm: khám phá dữ liệu, tiền xử lý dữ liệu cơ bản, phân chia tập huấn luyện, huấn luyện mô hình, cải thiện mô hình bằng phương pháp Undersampling.
    - Việc ghi nhận các giao dịch lừa đảo là rất quan trọng, hơn rất nhiều so với các giao dịch không phải lừa đảo nên mô hình cũng sẽ được xây dựng theo hướng tối ưu nhất các dự đoán giao dịch là lừa đảo.

### **Mô tả chi tiết**

* + - Khởi tạo Session và đọc dữ liệu đầu vào

*A screenshot of a computer

AI-generated content may be incorrect.*

*Hình 1: Tập dữ liệu*

* + - Kiểm tra missing values

A screenshot of a computer

AI-generated content may be incorrect.

*Hình 2: Kiểm tra missing values*

* + - Chuẩn hoá dữ liệu: Thực hiện chuẩn hoá Z-score đối với 2 dữ liệu chưa được chuẩn hoá là `time` và `Amount`, ta gom 2 thuộc tính lại thành 1 vector để chuẩn hoá sau đó tách ra lại.

A computer screen with white text

AI-generated content may be incorrect.

*Hình 3: Chuẩn hoá dữ liệu*

* + - Xoá dữ liệu trùng lắp: dữ liệu bị trùng sẽ ảnh hưởng đến chất lượng của mô hình

*A screen shot of a computer

AI-generated content may be incorrect.*

*Hình 4: Xoá dữ liệu trùng lặp*

* + - Phân chia tập dữ liệu: tập dữ liệu không những chỉ phải được phân chia với tỉ lệ train-test phù hợp mà còn phải đảm bảo số lượng nhãn được phân chia phù hợp với vấn đề mất cân bằng dữ liệu trầm trọng.

*A screenshot of a computer program

AI-generated content may be incorrect. Hình 5: Kết quả phân chia train-test*

* + - Huấn luyện mô hình cơ bản: sau khi tiền xử lý dữ liệu cơ bản, ta có thể tiến hành học từ tập huấn luyện nhanh chóng khi structured API hỗ trợ các API cho mô hình logistic regression sau khi assemble các thuộc tính thành vector.

*A screen shot of a computer code

AI-generated content may be incorrect.*

*Hình 6: Huấn luyện mô hình*

* + - Đánh giá mô hình trên tập huấn luyện: Đánh giá mô hình bằng các chỉ số Accuracy, AUC, Precision, Recall và F1-score. Mô hình hiện tại cho thấy hiệu suất đối với lớp 0 cực cao trong khi với lớp 1 thì chỉ đạt mức trung bình, điều này là do việc mất cân bằng dự liệu.

*A screenshot of a computer

AI-generated content may be incorrect.*

*Hình 7: Hiệu suất mô hình trên tập huấn luyện*

* + - Dự đoán và kiểm tra trên tập test: Áp dụng mô hình vừa học được ở trên cho dữ liệu của tập test để bổ sung thêm cột dự đoán, tính các chỉ số tp,tn,fp,fn để tính các thông số hiệu suất. Tuy nhiên kết quả chưa thể chấp nhận được đối với nhãn 1 nên ta sẽ tiếp tục cải thiện hiệu suất mô hình.

*A screen shot of a computer

AI-generated content may be incorrect.*

*Hình 8: Chỉ số trên tập test*

* + - Undersampling: áp dụng phương pháp undersampling để giảm kích thước của nhãn 0 xuống để giúp mô hình học tốt hơn, có thể phân biệt rõ ràng nhãn 0 và nhãn 1. Lưu ý chỉ oversampling sau khi split tập train-test. Ta lấy tỉ lệ 1:80 thay vì 1:577 như tập dữ liệu gốc.

A screen shot of a computer program

AI-generated content may be incorrect.

Hình 9: Undersampling

* + - Vẽ biểu đồ tương quan để thấy kết quả của undersampling ảnh hưởng thế nào đến mô hình

*A graph of numbers and symbols

AI-generated content may be incorrect.*

*Hình 10: Ma trận tương quan sau Undersampling*

* + - Huấn luyện lại mô hình trên tập Undersampling: các thông số đánh giá đối với nhãn 1 có sự cải thiện đáng kể.

*A computer screen shot of a number

AI-generated content may be incorrect.*

*Hình 11: Hiệu suất mô hình trên tập Undersampling*

* + - Dự đoán và kiểm tra tập test sau khi huấn luyện lại mô hình: Chỉ số Recall của mô hình đã có sự cải thiện đáng kể, giúp việc dự đoán một giao dịch là lừa đảo trở nên chính xác hơn.

*A screen shot of a computer

AI-generated content may be incorrect.*

*Hình 12: Chỉ số trên tập test*

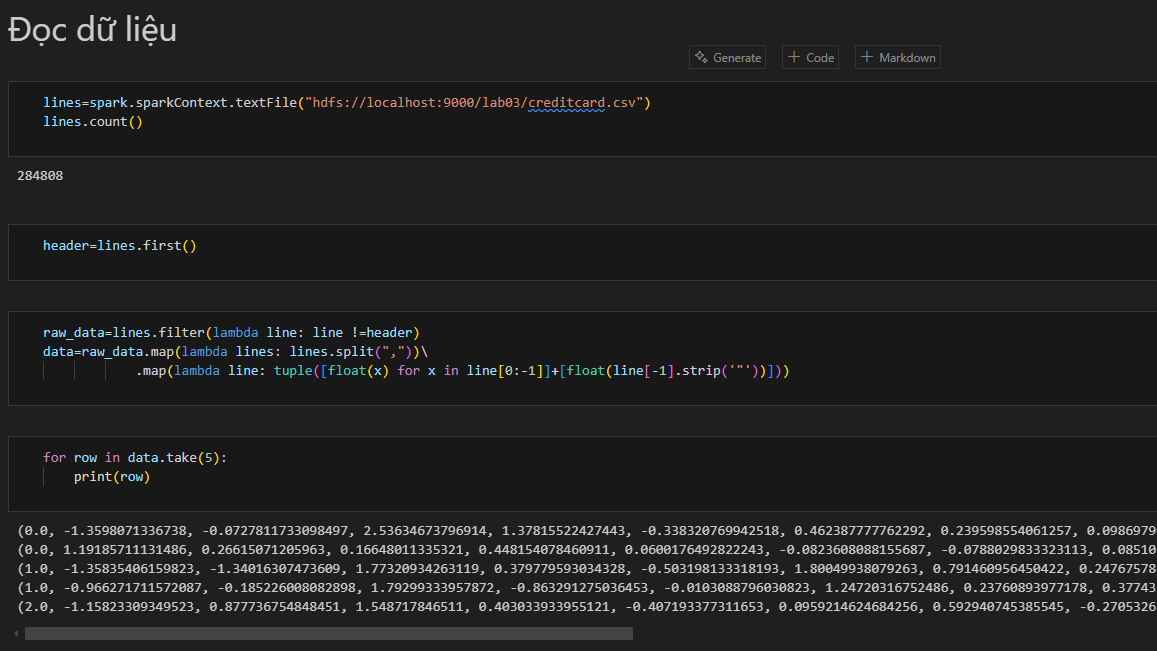
## MLlib RDD-Based Implementation

### **Ý tưởng thực hiện**

* + - Áp dụng các chức năng tích hợp sẵn của MLlib trong Spark trên dữ liệu RDD (spark.mllib) để thực hiện các tác vụ học máy, qua đó hiểu sâu hơn về cách Spark xử lý RDD.
    - Đối với bài toán này, mục đích của việc xây dựng mô hình học máy nhằm phát hiện ra các giao dịch gian lận bằng cách sử dụng thư viện Mllib. Bài làm gồm có quy trình như sau: Tiền xử lý dữ liệu cơ bản, phân chia tập dữ liệu và chuyển thành RDD LabeledPoint, huấn luyện mô hình và chọn tham số tốt nhất, đánh giá mô hình sau khi huấn luyện và sau khi dự đoán.
    - Như đã đề cập ở phần Structured API, vấn đề của dữ liệu này là bị mất cân bằng dữ liệu rất nặng.
    - Việc ghi nhận các giao dịch lừa đảo là rất quan trọng. Dự báo nhầm còn hơn bỏ xót do đó việc chọn tham số tốt nhất của mô hình sẽ dựa vào chỉ số đánh giá trên lớp “gian lận”.

### **Mô tả chi tiết**

* + - Khởi tạo Spark Session, đọc dữ liệu đầu vào và chuyển thành các dòng dữ liệu phù hợp trong việc xây dựng mô hình.



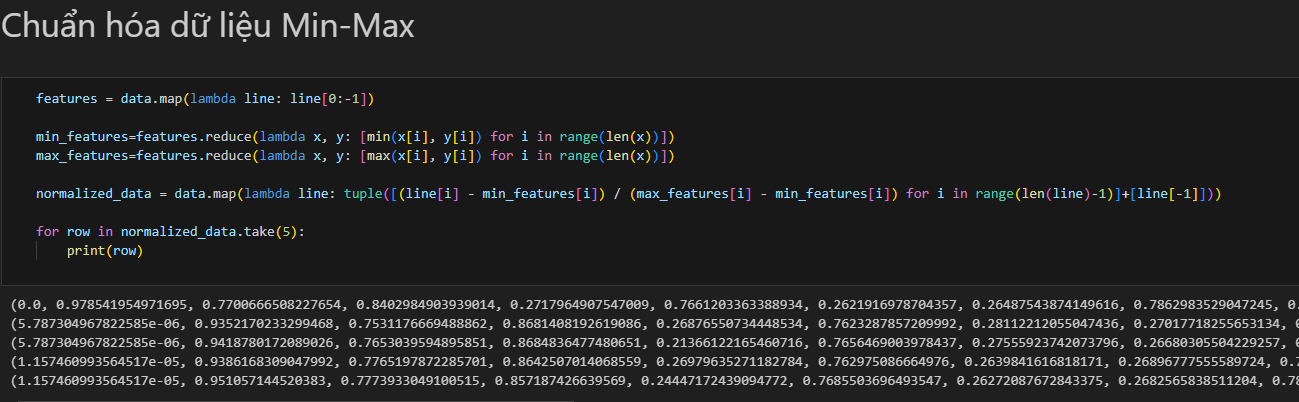
*Hình 13. Tập dữ liệu.*

* + - Tiền xử lý dữ liệu gồm các bước sau: Loại bỏ dữ liệu lặp và loại bỏ dữ liệu trống. Sau khi thực hai bước trên. Số dòng dữ liệu giảm xuống còn 283726 dòng. A screenshot of a computer

      AI-generated content may be incorrect.

*Hình 14. Tiền xử lý dữ liệu.*

* + - Chuẩn hóa dữ liệu theo phương pháp Min-Max: Đưa tất cả về cùng một thang đo, giúp mô hình hội tụ nhanh hơn và tránh việc đặc trưng có giá trị lớn lấn át đặc trưng có giá trị nhỏ.



*Hình 15. Chuẩn hóa dữ liệu Min-Max.*

* + - Chia dữ liệu train-validation-test theo tỉ lệ như sau: 0.7, 0.15, 0.15. Tỉ lệ này cân bằng giữa với việc huấn luyện, tối ưu và đánh giá. Và việc phân chia còn đảm bảo các tập dữ liệu đều có đủ cả hai lớp.

A black screen with colorful text

AI-generated content may be incorrect.

*Hình 16. Chia tập dữ liệu.*

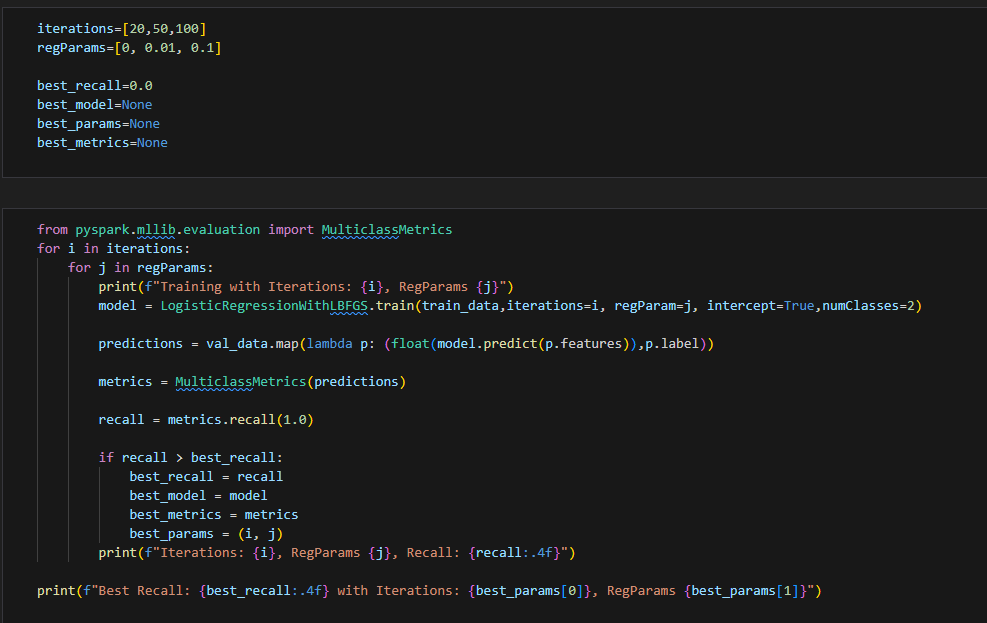
* + - Chuyển thành dạng RDD LabeledPoint: Tạo một tập dữ liệu huấn luyện thống nhất (đầu vào và đầu ra đi kèm nhau) phù hợp cho mô hình huấn luyện.

A screen shot of a computer code

AI-generated content may be incorrect.

*Hình 17. Chuyển thành RDD LabeledPoint.*

* + - Thực hiện huấn luyện mô hình: Mô hình sử dụng là mô hình LogisticsRegressionWithLBFGS với các tham số sử dụng là iterations (số vòng lặp tối đa mà thuật toán L-BFGS được phép chạy để hội tụ), regParam (Tham số điều chuẩn, thường là L2), intercept (có bias hay không) và numClassess (số lượng lớp phân loại). Về thang đo, nhóm sử dụng thư viện có sẵn MulticlassMetrics để đánh giá. Sau đó thực hiện Grid Search để tìm ra tham số tốt nhất với tiêu chí đánh giá là Recall của lớp 1 càng lớn càng tốt.



*Hình 18. Thực hiện huấn luyện mô hình.*

*A screenshot of a computer

AI-generated content may be incorrect.*

*Hình 19. Chọn được mô hình tốt nhất.*

* + - Đánh giá mô hình sau khi huấn luyện: Các chỉ số dùng để đánh giá là Accuracy, Precision, Recall và F1-Score. Mặc dù kết quả của lớp 0 rất tốt nhưng chỉ số của lớp 1 đặc biệt là Recall lại không cao.

A number on a black background

AI-generated content may be incorrect.

*Hình 20. Đánh giá sau khi huấn luyện.*

* + - Thực hiện dự đoán.

A screen shot of a computer

AI-generated content may be incorrect.

*Hình 21. Dự đoán.*

* + - Đánh giá mô hình sau khi dự đoán: Các chỉ số đánh giá như sau bước huấn luyện. Kết quả của Recall của lớp 1 lại có chút cải thiện so với sau khi huấn luyện. Tuy nhiên, muốn kết quả tốt hơn, chúng ta cần thực hiện như bổ sung thêm dữ liệu hoặc oversampling lớp 1, undersampling lớp 0.

A number on a black background

AI-generated content may be incorrect.

*Hình 22. Đánh giá sau khi dự đoán*

* + - So sánh với Structured API: Hiệu suất tương đương so với Structured API nếu không thực hiện các phương pháp cải thiện hiệu suất.

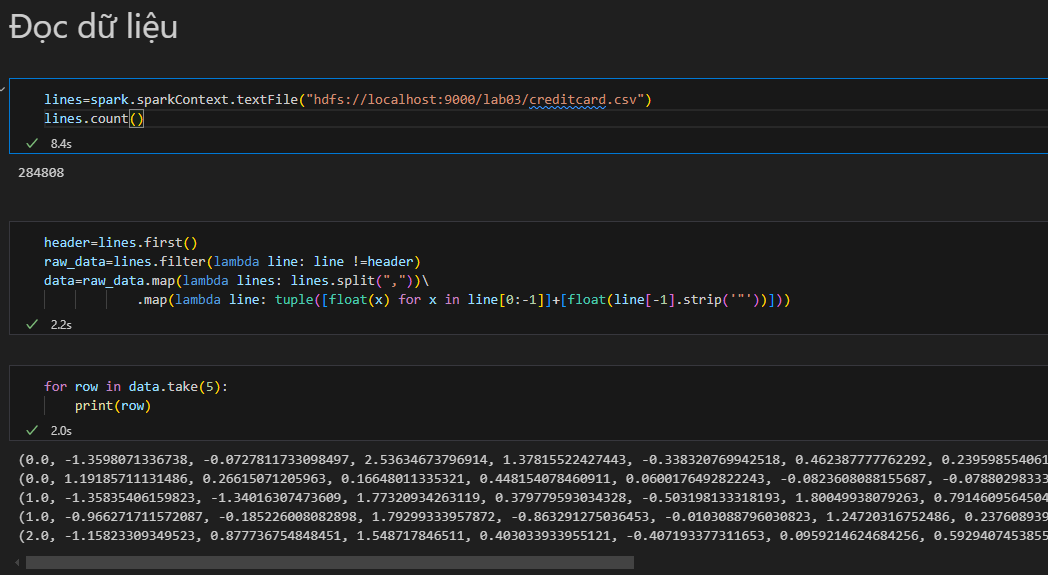
## Low-Level Operations

### **Ý tưởng thực hiện**

* + - Xây dựng thủ công mô hình Logistics Regression bằng các phép toán RDD cơ bản (mà không dựa vào MLlib), nhằm nâng cao tư duy song song và hiểu biết về tính toán phân tán.
    - Đối với bài toán này, mục đích của việc xây dựng mô hình học máy nhằm phát hiện các giao dịch gian lận bằng cách tự xây dựng thủ công mô hình theo RDD mà không sử dụng các thư viện nào khác.
    - Quy trình làm bao gồm: tiền xử lý dữ liệu cơ bản, chuẩn hóa dữ liệu, chuyển dữ liệu thành đầu vào thích hợp, phân chia dữ liệu, xây dựng mô hình, huấn luyện mô hình và đánh giá mô hình sau khi huấn luyện và sau khi dự đoán.
    - Như đã đề cập ở trên, dữ liệu bị mất cân bằng dữ liệu giữa lớp 0 và lớp 1 do đó việc xây dựng mô hình rất khó đạt được hiệu suất tốt.

### **Mô tả chi tiết**

* + - Khởi tạo Spark Session, đọc dữ liệu đầu vào và chuyển thành các dòng dữ liệu phù hợp trong việc xây dựng mô hình.



*Hình 23. Tập dữ liệu.*

* + - Tiền xử lý dữ liệu gồm các bước sau: Loại bỏ dữ liệu lặp và loại bỏ dữ liệu trống. Sau khi thực hai bước trên. Số dòng dữ liệu giảm xuống còn 283726 dòng.

A screenshot of a computer program

AI-generated content may be incorrect.

*Hình 24. Tiền xử lý dữ liệu.*

* + - Chuẩn hóa dữ liệu bằng Min-Max: Đưa tất cả về cùng một thang đo, giúp mô hình hội tụ nhanh hơn và tránh việc đặc trưng có giá trị lớn lấn át đặc trưng có giá trị nhỏ.

A screenshot of a computer

AI-generated content may be incorrect.

*Hình 25. Chuẩn hóa dữ liệu Min-Max.*

* + - Chuyển dữ liệu thành đầu vào phù hợp: chuyển dành dữ liệu gồm vector đặc trưng và nhãn. Ngoài ra bổ sung thêm bias vào vector đặc trưng.

A screenshot of a computer program

AI-generated content may be incorrect.

*Hình 26. Chuyển thành đầu vào hợp lý và thêm bias.*

* + - Chia dữ liệu thành train-test với tỉ lệ 0.8 và 0.2. Đây là một cách chia phổ biến với 80% dữ liệu huấn luyện cung cấp đủ mẫu để mô hình học được quy luật và mối quan hệ giữa đặc trưng và nhãn. Ngoài ra chia dữ liệu sao cho dữ liệu train và test đều có chứa 2 nhãn là 0 và 1.

A black background with colorful text

AI-generated content may be incorrect.

*Hình 27. Chia dữ liệu*

* + - Xây dựng các hàm có trong mô hình Logistics Regression. Đầu tiên là hàm sigmoid.

A black screen with white text and colorful numbers

AI-generated content may be incorrect.

*Hình 28. Hàm sigmoid*

* + - Hàm dot với chức năng nhân trọng số với một dòng dữ liệu.

A black screen with white text

AI-generated content may be incorrect.

*Hình 29. Hàm nhân trọng số với một dòng dữ liệu*

* + - Hàm tính gradient với một dòng dữ liệu.

A screen shot of a computer code

AI-generated content may be incorrect.

*Hình 30. Hàm tính gradient với một dòng dữ liệu.*

* + - Hàm train mô hình với tham số *num\_iterations*, *learning\_rate*: Khởi tạo bộ trọng số với giá trị 0. Thực hiện gradient descent cho tới khi đủ *num\_iterations* lần với tốc độ học *learning\_rate*. Kết quả trả về bộ trọng số *weights*.

A computer screen shot of text

AI-generated content may be incorrect.

*Hình 31. Hàm thực hiện huấn luyện.*

* + - Hàm dự đoán với tham số là weights (bộ trọng số) và features (1 dòng dữ liệu).

A screen shot of a computer code

AI-generated content may be incorrect.

*Hình 32. Hàm thực hiện dự đoán.*

* + - Hàm evaluate\_metric có chức năng tính các giá trị accuracy, precision, recall, f1\_score của một nhãn cụ thể.

A computer screen shot of a code

AI-generated content may be incorrect.

*Hình 33. Hàm thực hiện tính các metric.*

* + - Hàm evaluate\_model có chức năng đánh giá mô hình. Thực hiện bằng cách chạy hàm evaluate\_metric cho hai nhãn của mô hình và trả về kết quả cho từng nhãn.

A screen shot of a computer code

AI-generated content may be incorrect.

*Hình 34. Hàm thực hiện đánh giá mô hình.*

* + - Thực hiện huấn luyện mô hình với tham số *num\_iterations* bằng 100 và *learning\_rate* bằng 0.5. Sau khi thử với nhiều tham số khác nhau song vẫn không đạt được hiệu suất tốt. Đây là khó khăn trong việc huấn luyện. Không xác định được tham số phù hợp cho mô hình để có hiệu suất tốt.

A screenshot of a computer

AI-generated content may be incorrect.

*Hình 35. Thực hiện huấn luyện mô hình.*

* + - Đánh giá mô hình sau khi huấn luyện và đánh giá sau khi dự đoán. Như đã thấy, kết quả hoàn toàn nghiêng về bên lớp 0 và không hoàn toàn có lớp 1 nào được dự đoán.

A black screen with white numbers

AI-generated content may be incorrect.

*Hình 36. Đánh giá sau khi huấn luyện.*

A black screen with white numbers

AI-generated content may be incorrect.

*Hình 37. Đánh giá sau khi dự đoán.*

* + - So sánh kết quả với hai phương pháp Structured API và RDD-Based: Kết quả từ mô hình Logistic Regression triển khai bằng RDD cấp thấp cho thấy hiệu suất thấp hơn so với hai phương pháp trên, đặc biệt là với lớp gian lận (lớp 1). Hầu hết các điểm dữ liệu đều bị dự đoán về lớp 0, cho thấy mô hình không học được đủ thông tin phân biệt.
    - Các thử thách gặp phải trong quá trình làm: Khó xác định được learning rate và num iterations phù hợp. Mô hình không học được lớp thiểu số do chênh lệch nhãn, dẫn đến mất cân bằng nghiêm trọng trong kết quả dự đoán.

## So sánh các phương pháp giải quyết

|  |  |  |
| --- | --- | --- |
|  | Ưu điểm | Nhược điểm |
| Structured API-High Level | Hiệu suất cao nhờ Catalyst Optimizer & Tungsten Engine.  Dễ viết, dễ đọc khi cú pháp tương tự SQL.  Hỗ trợ nhiều API có sẵn, tránh xử lý thủ công.  Dễ dàng build các pipleline trong machine learning | Chỉ hỗ trợ xử lý dữ liệu có cấu trúc, có scheme rõ ràng. |
| MLLib RDD-Based | Kiểm soát tốt hơn trên RDD.  Mô hình tối ưu có sẵn.  Có đánh giá mô hình chi tiết. | Ít linh hoạt hơn Structured API. |
| Low-Level Operations | Giúp hiểu sâu hơn về phép toán RDD.  Kiểm soát toàn bộ quá trình tính toán.  Phù hợp để học và nghiên cứu. | Khó triển khai, dễ sai sót.  Hiệu suất kém nếu không tinh chỉnh kỹ. |

# PHẦN 2: REGRESSION WITH DECISION TREES

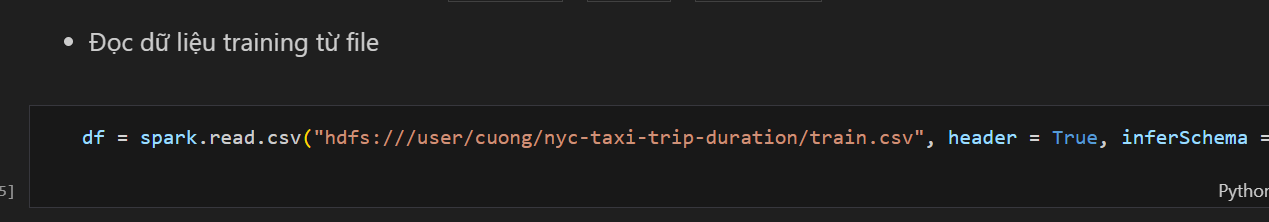
## Structured API Implementation (High Level)

### **Ý tưởng thực hiện**

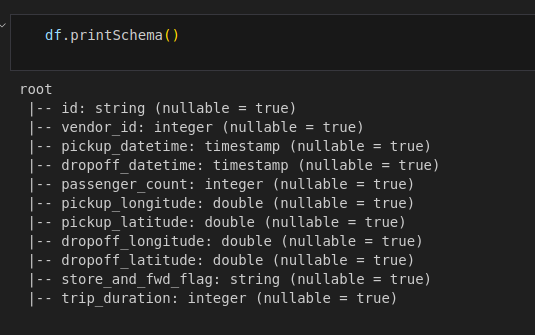
* Spark Structured API (spark.ml) cung cấp các công cụ cấp cao để xây dựng pipeline học máy một cách dễ dàng và có khả năng mở rộng tốt trên dữ liệu lớn. Việc sử dụng Structured API giúp đơn giản hóa quy trình tiền xử lý và huấn luyện mô hình.
* Bài toán yêu cầu dự đoán một biến liên tục (continuous variable), do đó đây là mộtbài toán hồi quy (regression). Việc sử dụng Decision Tree Regressor sẽ cho phép mô hình học được các mối quan hệ phi tuyến (non-linear) giữa các đặc trưng và đầu ra.
* Cây quyết định (Decision Tree) là một mô hình dễ hiểu, có thể trực quan hóa, và có khả năng xử lý tốt các đặc trưng dạng số và không yêu cầu chuẩn hóa dữ liệu đầu vào.
* Quy trình sẽ bao gồm các bước: tiền xử lý dữ liệu (bao gồm xử lý dữ liệu thiếu, kết hợp đặc trưng bằng VectorAssembler), huấn luyện mô hình hồi quy cây quyết định, trích xuất cấu trúc cây, và cuối cùng là đánh giá mô hình bằng các chỉ số như RMSE (Root Mean Squared Error) và R² (R-squared) trên tập kiểm tra.
* Việc triển khai bài toán hồi quy trên dữ liệu lớn bằng Decision Tree giúp thể hiện khả năng xử lý và huấn luyện mô hình song song của Spark, đồng thời cung cấp hiểu biết trực quan về cách các đặc trưng ảnh hưởng đến kết quả dự đoán.

### **Mô tả chi tiết**

* Khởi tạo Spark Session và đọc dữ liệu

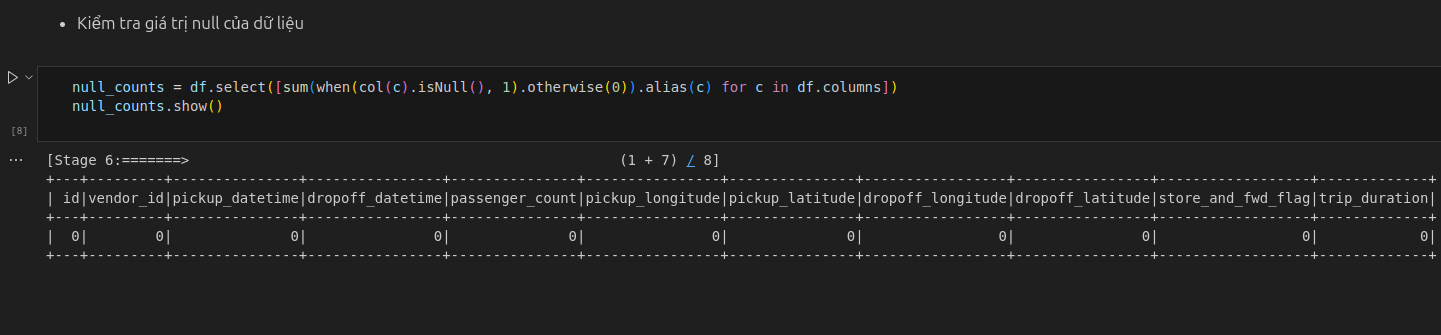


*Hình 38. Đọc file từ HDFS.*



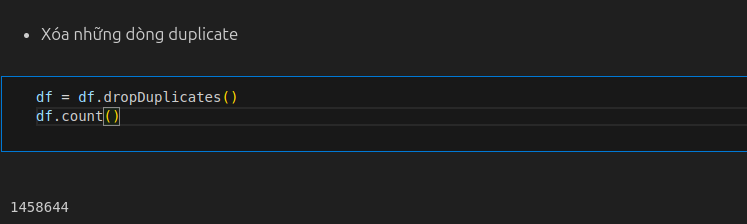
*Hình 39. Schema của dữ liệu train*

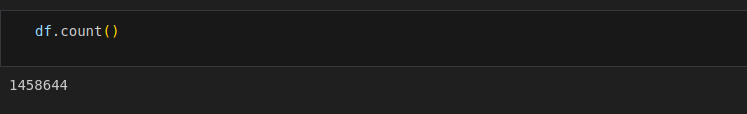
* Kiểm tra dữ liệu bị thiếu



*Hình 40. Xử lý dữ liệu thiếu*

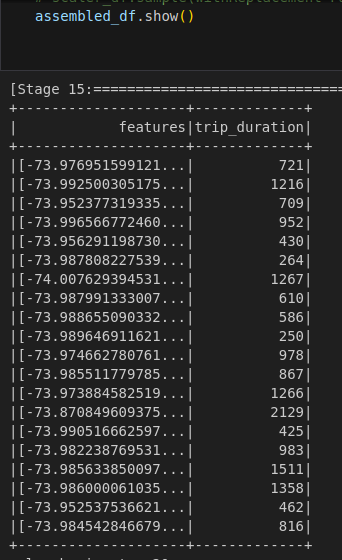
* Xử lý dữ liệu bị trùng





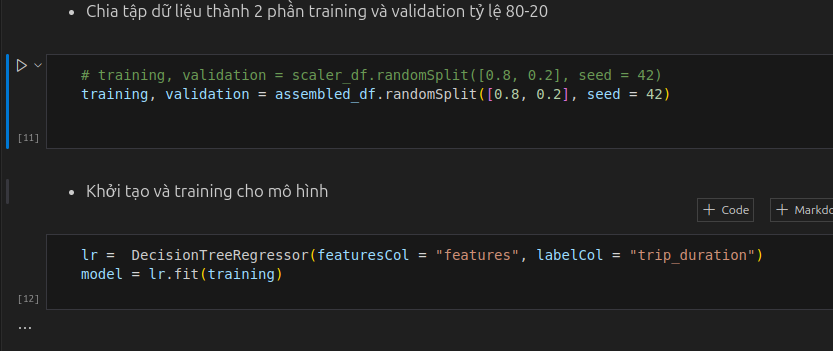
*Hình 41. Xóa các dòng trùng lặp*

* Tạo assembler các thuộc tính số để có thể đưa vào mô hình

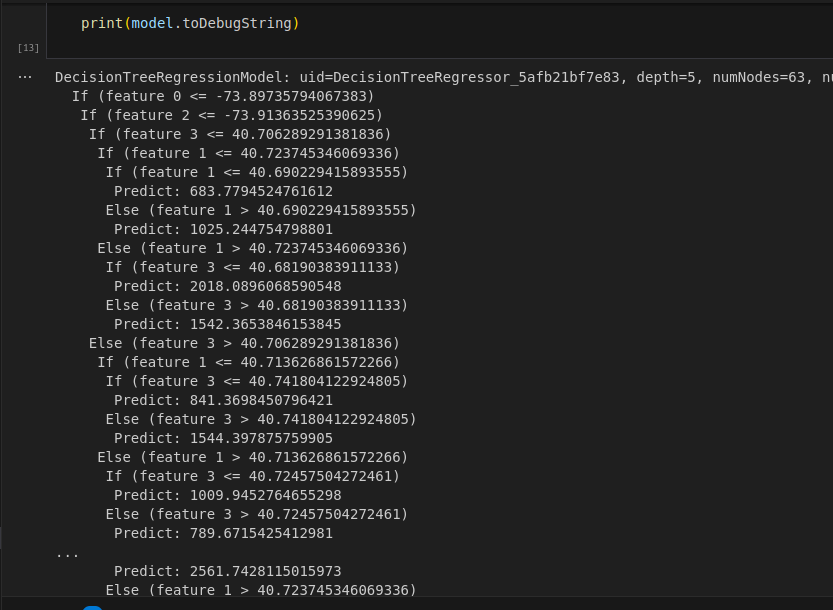


*Hình 42. Assembler dữ liệu đầu vào*

* Chia tập dữ liệu và training mô hình

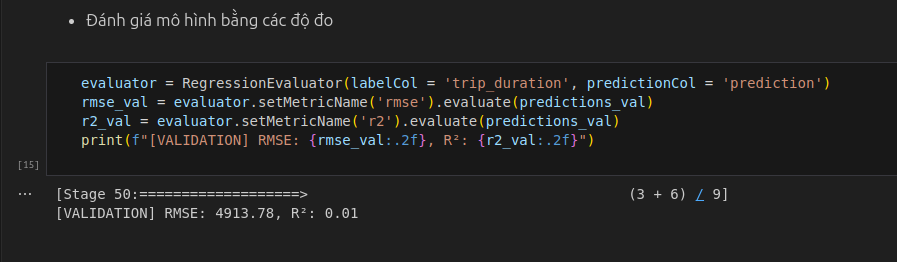


*Hình 43. Chia dữ liệu và huấn luyện*



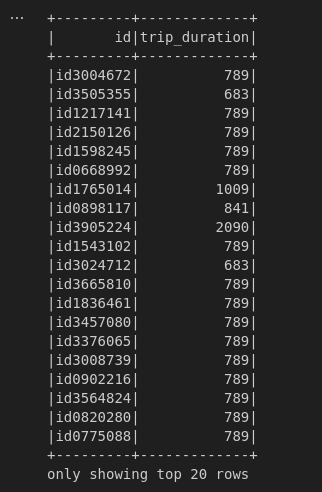
*Hình 44. Cây quyết định*

* Đánh giá



*Hình 45. Đánh giá mô hình*

* Dự đoán kết quả và ghi ra file



*Hình 46. Kết quả dự đoán*

## MLlib RDD-Based Implementation

### **Ý tưởng thực hiện**

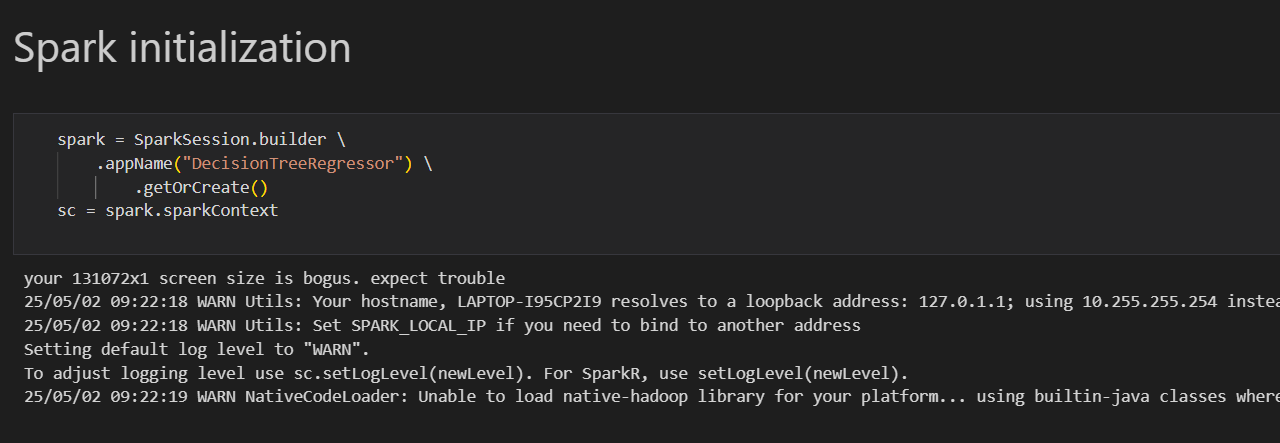
Mục tiêu bài toán:

* Dự đoán thời gian chuyến đi (trip\_duration) dựa trên các thông tin như:
* Vị trí đón và trả khách (longitude, latitude)
* Số lượng hành khách (passenger\_count)

Ý tưởng giải pháp: Áp dụng thuật toán Decision Tree Regression của MLlib (dựa trên RDD) để xây dựng mô hình dự đoán thời gian chuyến đi.

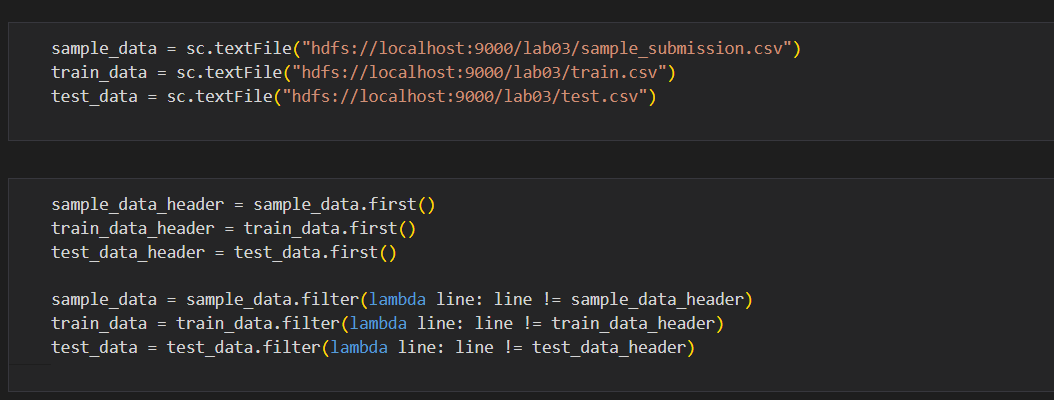
RDD-Based API:

* Cần kiểm soát chặt chẽ các thao tác xử lý dữ liệu.
* Tối ưu bộ nhớ và hiệu suất khi xử lý tệp dữ liệu lớn dạng text.
* Thích hợp để làm quen với nguyên lý hoạt động của hệ sinh thái Spark.
  1. **Mô tả chi tiết**
* Khởi tạo Spark Session, đọc dữ liệu đầu vào và chuyển thành các dòng dữ liệu phù hợp trong việc xây dựng mô hình.



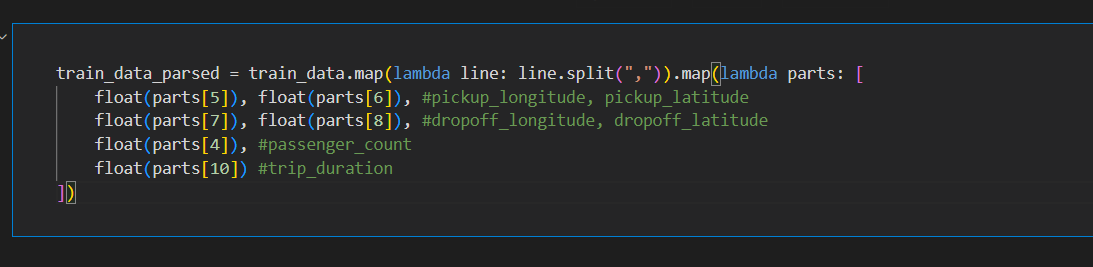
*Hình 47. Khởi tạo Spark*

* Dữ liệu từ các tệp sample\_submission.csv, train.csv, test.csv được đọc bằng textFile, tạo các RDD, mỗi dòng là một chuỗi văn bản (string).



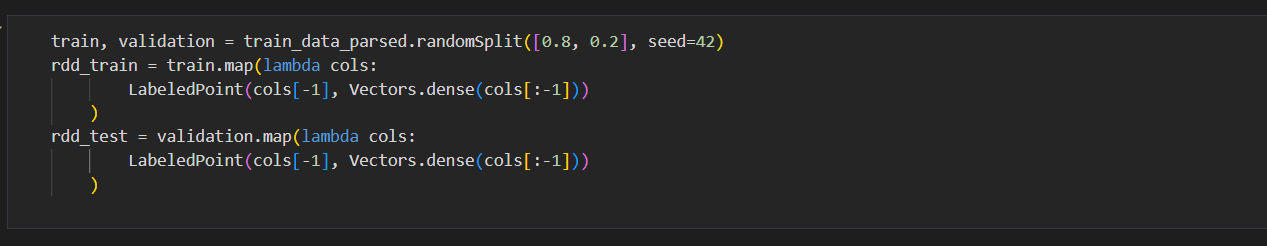
*Hình 48. Đọc dữ liệu từ HDFS*

* Parsing the data: Chọn ra các cột dạng numeric là những đặc trưng cần thiết để phục vụ huấn luyện mô hình từ tập dữ liệu `train.csv`. Sau đó ép kiểu `float`. `train\_data\_parsed` thu được là một RDD của danh sách số thực, mỗi phần tử đại diện cho một chuyến đi với 5 đặc trưng + 1 nhãn.



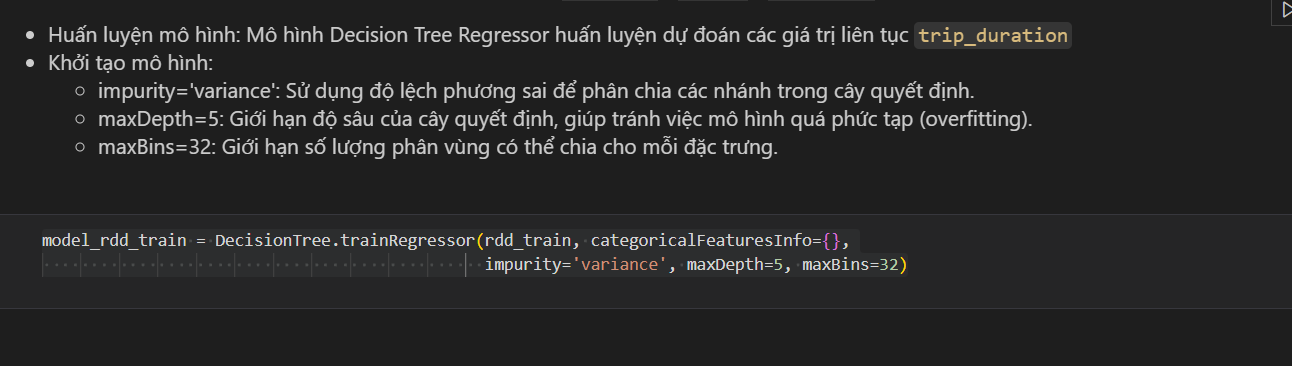
*Hình 49. Chọn lọc dữ liệu đầu vào*

* Tiền xử lý dữ liệu: Chia dữ liệu thành tập huấn luyện và kiểm tra: Dữ liệu được chia thành hai phần: (training set) và (validation set). Việc chia này đảm bảo rằng mô hình không "học thuộc" (overfitting) dữ liệu huấn luyện và có thể tổng quát tốt trên dữ liệu chưa thấy. Chuyển đổi thành `LabeledPoint`: đóng gói đặc trưng và nhãn thành cấu trúc `LabeledPoint` để phù hợp với các mô hình học máy trong Spark.



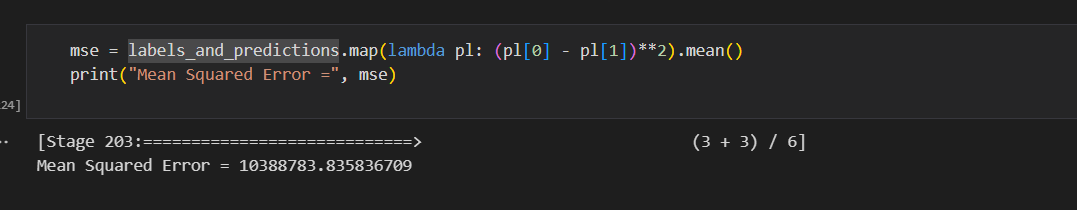
*Hình 50. Chia dữ liệu và chuẩn hóa*

* Huấn luyện mô hình: Huấn luyện mô hình: Mô hình Decision Tree Regressor huấn luyện dự đoán các giá trị liên tục `trip\_duration`



*Hình 51. Huấn luyện mô hình*

* Đánh giá mô hình: MSE đo lường trung bình bình phương khoảng cách giữa giá trị dự đoán và thực tế. Ở đây, giá trị này hơn 10 triệu là khá lớn. Điều này có nghĩa là mỗi lần dự đoán trung bình có thể lệch tới gần 54 phút so với thực tế.
* Cải thiện mô hình:
  + Thêm đặc trưng mới như thời gian trong ngày, ngày trong tuần.v.v.
  + Tăng độ sâu của cây (maxDepth) để mô hình học kỹ hơn.
  + Chuẩn hóa dữ liệu nếu có outliner.
  + Thử mô hình khác như Random Forest hoặc Gradient Boosted Trees.



*Hình 52. Đánh giá mô hình*

## Low-Level Operations

### **Ý tưởng thực hiện**

Mục tiêu bài toán

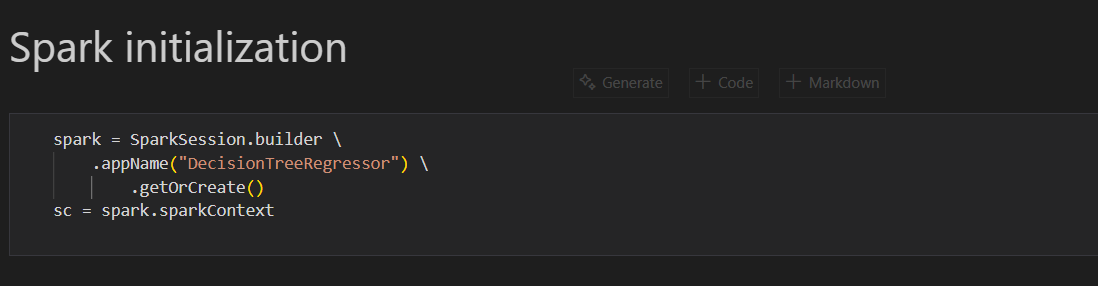
* Dự đoán trip\_duration (thời gian chuyến đi) từ các đặc trưng:
* Vị trí đón và trả khách (kinh độ, vĩ độ)
* Số hành khách (passenger\_count)

Ý tưởng giải pháp

* Thay vì sử dụng thư viện có sẵn như MLlib, mô hình được triển khai thủ công từ đầu (low-level) bằng cách:
* Xây dựng cây hồi quy (decision tree regressor) từ số liệu huấn luyện
* Áp dụng thuật toán chia nhánh (recursive splitting) theo feature và threshold tối ưu
* Dự đoán dữ liệu alidation thông qua việc traverse cây quyết định

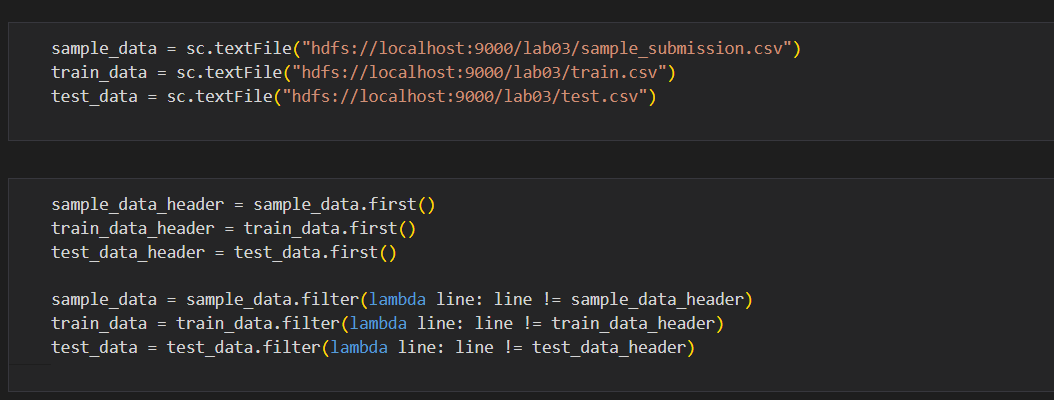
### **Mô tả chi tiết**

* Khởi tạo Spark Session, đọc dữ liệu đầu vào và chuyển thành các dòng dữ liệu phù hợp trong việc xây dựng mô hình.

**

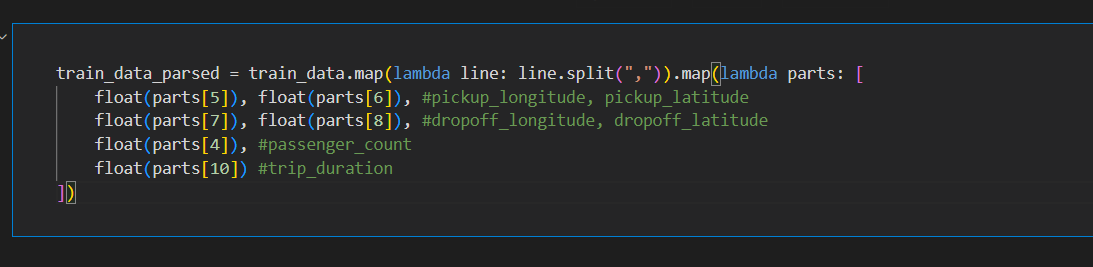
*Hình 53. Khởi tạo Spark*

* Dữ liệu từ các tệp sample\_submission.csv, train.csv, test.csv được đọc bằng textFile, tạo các RDD, mỗi dòng là một chuỗi văn bản (string).



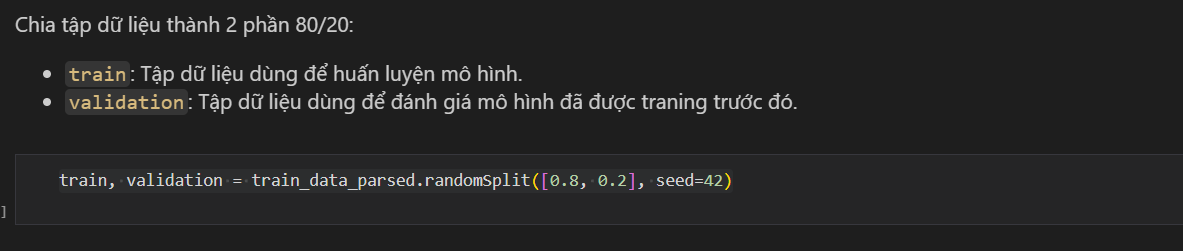
*Hình 54. Đọc file từ HDFS*

* Parsing the data: Chọn ra các cột dạng numeric là những đặc trưng cần thiết để phục vụ huấn luyện mô hình từ tập dữ liệu `train.csv`. Sau đó ép kiểu `float`. `train\_data\_parsed` thu được là một RDD của danh sách số thực, mỗi phần tử đại diện cho một chuyến đi với 5 đặc trưng + 1 nhãn.



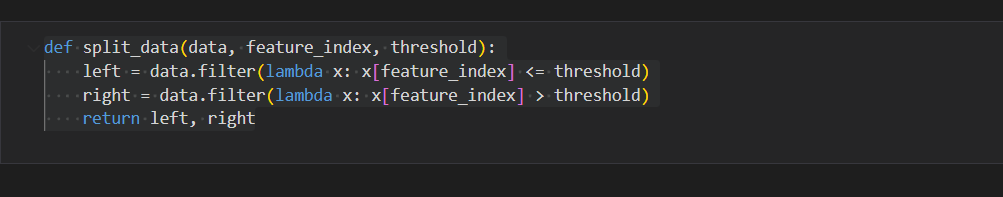
*Hình 55. Chọn lọc dữ liệu đầu vào*

* Tiền xử lý dữ liệu: Chia dữ liệu thành tập huấn luyện và kiểm tra: Dữ liệu được chia thành hai phần: (training set) và (validation set). Việc chia này đảm bảo rằng mô hình không "học thuộc" (overfitting) dữ liệu huấn luyện và có thể tổng quát tốt trên dữ liệu chưa thấy.



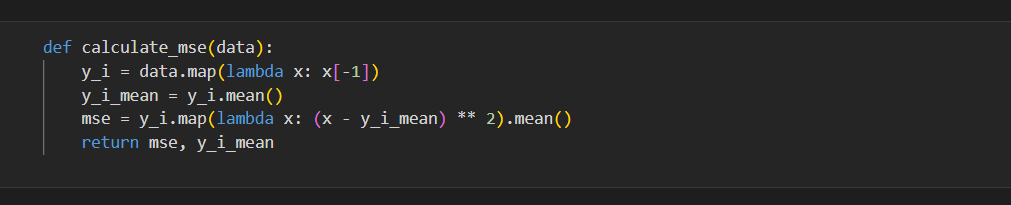
*Hình 56. Chia dữ liệu train, validation*

* Split data: Chia dữ liệu thành 2 nhánh (trái, phải) dựa trên `feature\_index` và `threshold`
  + Nhánh trái chứa các điểm có giá trị ≤ threshold
  + Nhánh phải chứa các điểm có giá trị > threshold



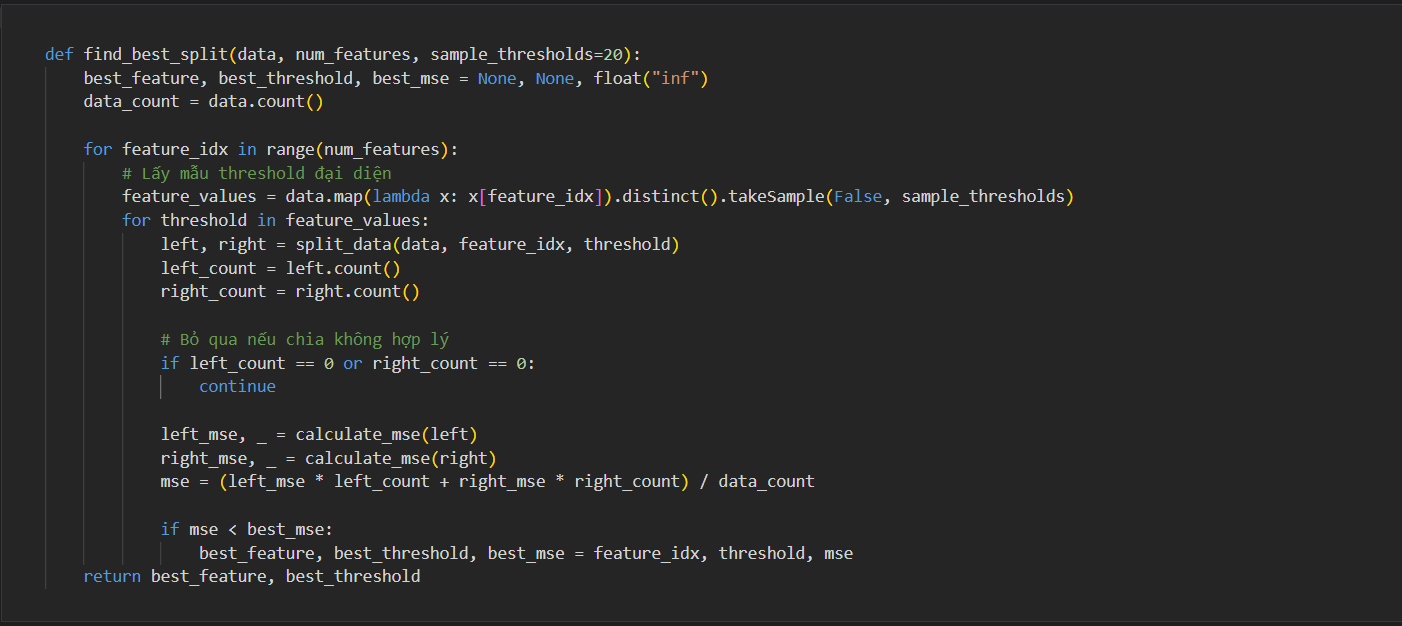
*Hình 57. Chia nhánh cây*

* Calculate mse: Mean Squared Error (MSE) của tập dữ liệu đầu vào và giá trị trung bình y. MSE được dùng để đo độ phân tán so với trung bình.



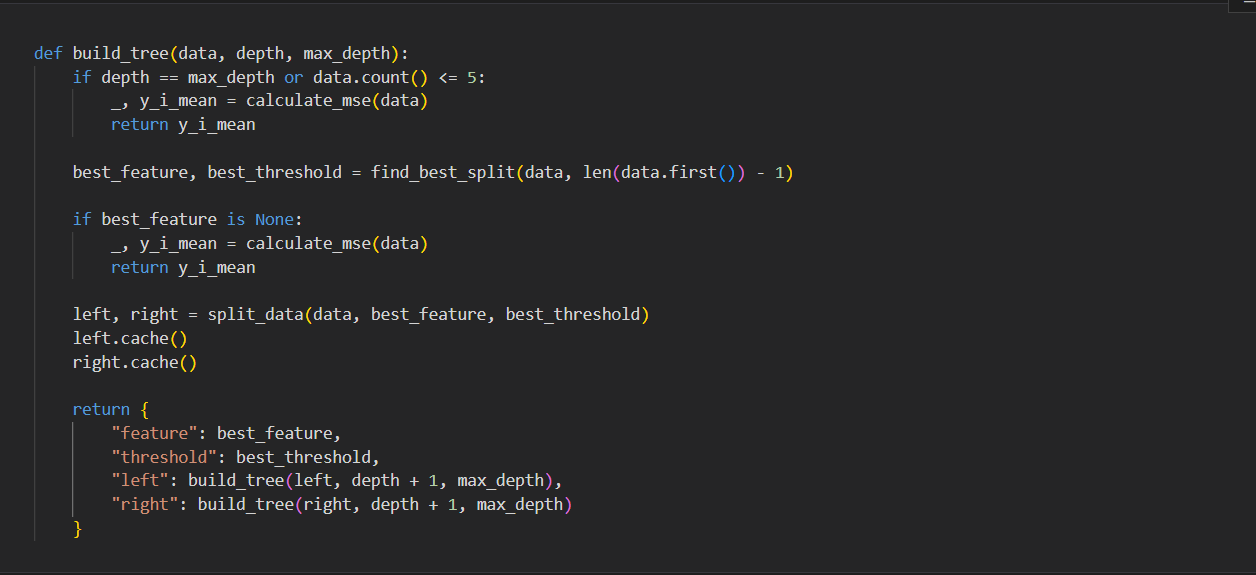
*Hình 58. Hàm tính MSE*

* Find best split: Tìm ra `feature` và ngưỡng `threshold` tốt nhất để chia dữ liệu, sao cho MSE của hai tập con (sau khi chia) là nhỏ nhất.



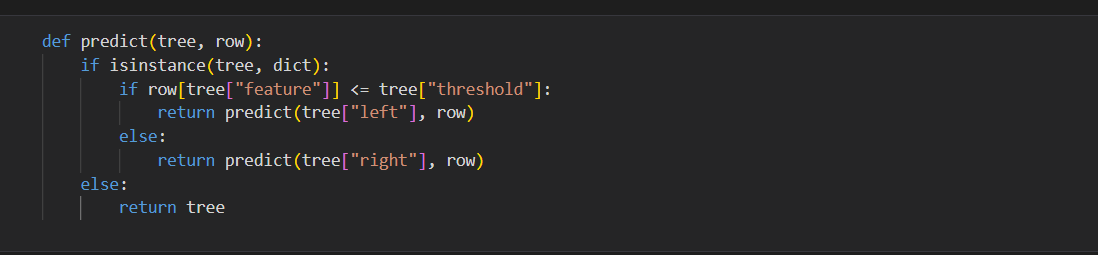
*Hình 59. Hàm chia dữ liệu tối ưu nhất*

* Build tree: Xây dựng cây theo phương pháp đệ quy:
  + Nếu đã đạt đến độ sâu tối đa hoặc dữ liệu quá ít -> trả về giá trị trung bình hàm lá (dự đoán)
  + Ngược lại:
    - Tìm cách chia tối ưu (find\_best\_split)
    - Đệ quy xây nhánh trái và phải



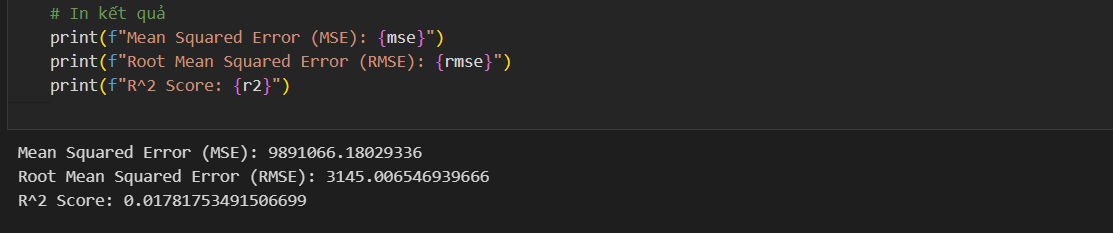
*Hình 60. Hàm xây dựng Decision tree*

* Predict
  + Kiểm tra nếu tree là một dictionary -> đây là nút không phải lá.
  + So sánh row[feature] với threshold:
    - Nếu nhỏ hơn hoặc bằng -> duyệt tiếp xuống nhánh trái (left)
    - Ngược lại -> duyệt xuống nhánh phải (right)
    - Nếu không phải dict (tức là giá trị thực tại lá) -> trả về giá trị đó (là dự đoán).



*Hình 61. Hàm dự đoán*

* Đánh giá mô hình: MSE đo lường trung bình bình phương khoảng cách giữa giá trị dự đoán và thực tế. Ở đây, giá trị này gần 10 triệu là khá lớn. Điều này có nghĩa là mỗi lần dự đoán trung bình có thể lệch tới gần 53 phút so với thực tế. Cho thấy mô hình train chưa được tối ưu. R^2 xấp xỉ 0 cho thấy, mô hình gần như không học được quy tắc nào cả.



*Hình 62. Đánh gia mô hình*

## So sánh các phương pháp giải quyết

|  |  |  |
| --- | --- | --- |
|  | Ưu điểm | Nhược điểm |
| Structured API-High Level | - Cú pháp ngắn gọn, dễ viết và dễ hiểu  - Tích hợp tốt với Spark SQL và DataFrame  - Tối ưu hiệu suất nhờ Catalyst Optimizer và Tungsten Engine | - Khó can thiệp sâu vào quá trình xử lý  - Ít linh hoạt nếu cần thực hiện các thao tác đặc thù ngoài mô hình hóa đơn giản |
| MLLib RDD-Based | - Dễ xây dựng mô hình học máy  - Có sẵn các thuật toán như Decision Tree, Random Forest, v.v.  - Có thể chạy phân tán trực tiếp trên RDD | - Mức độ trừu tượng vẫn còn cao  - Không tùy biến được cấu trúc thuật toán  - Đôi khi chậm nếu dữ liệu không được chuẩn hóa tốt |
| Low-Level Operations | - Linh hoạt tối đa: tự xây dựng mô hình, kiểm soát mọi bước  - Hiểu rõ cách thuật toán hoạt động  - Phù hợp với mục đích học thuật hoặc nghiên cứu | - Cần viết nhiều mã hơn  - Dễ lỗi logic  - Hiệu suất có thể kém hơn nếu không cache và chia dữ liệu hợp lý |

# PHẦN 3: BÁO CÁO NHÓM

* + Phương thức trao đổi và làm việc:
    - Vì khối lượng công việc lớn đồng thời để đảm bảo chất lượng, nhóm phân chia thành 2 đội nhóm chịu trách nhiệm giải quyết hai vấn đề lớn.
    - Nhóm tổ chức phiên họp nhóm sau một khoảng thời gian làm việc cá nhân để theo dõi sát tình hình giải quyết bài tập.
    - Phiên họp nhằm kiểm tra kết quả, ý tưởng giải quyết cho từng vấn đề.
    - Giao công việc cho từng thành viên trên workspace của nhóm sau khi thống nhất kết quả.
  + Bảng phân công công việc:

|  |  |  |
| --- | --- | --- |
| Thành viên | Công việc | Lưu ý |
| Trương Tiến  Anh | Thực hiện các phương thức giải quyết bài tập đối với vấn đề Regression with Decision Trees.  Khảo sát chất lượng các file, tổ chức thư mục và chú thích tại từng file. | Cả hai thành viên giải quyết vấn đề cùng nhau, chia sẻ các phương pháp cho từng cách tiếp cận và cân nhắc tính khả dụng ở các cách tiếp cận khác. |
| Đoàn Minh  Cường |
| Đinh Viết Lợi | Thực hiện các phương thức giải quyết bài tập đối với vấn đề Classification with Logistic Regression.  Viết báo cáo tổng hợp bài tập nhóm. |  |
| Nguyễn Trần  Lợi |