ĐẠI HỌC QUỐC GIA TP. HCM

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA HỆ THỐNG THÔNG TIN

MÔN BIG DATA

Dự đoán độ trễ hạ cánh so với thời gian dự kiến

Giảng viên hướng dẫn: ThS. Nguyễn Hồ Duy Trí

Lớp: IS405.P11

Nhóm sinh viên thực hiện

Võ Hồng Kim Anh	21520597
Trần Xuân Bằng	21521847
Lê Văn Tuấn	21522751

TP. HỒ CHÍ MINH 12/2024

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

LỜI CẢM ƠN

Lời đầu tiên, chúng em xin gửi lời cảm ơn chân thành đến thầy Nguyễn Hồ Duy Trí, giảng viên môn Dữ liệu lớn. Cảm ơn thầy đã tận tâm giảng dạy, truyền đạt cho chúng em những kiến thức vô cùng sâu sắc và thực tiễn về xử lý và khai thác dữ liệu lớn. Những bài giảng và hướng dẫn của thầy đã giúp chúng em nắm bắt rõ hơn các khái niệm cốt lõi cũng như phương pháp áp dụng vào bài toán cụ thể trong nghiên cứu này.

Bên cạnh đó, chúng em cũng xin cảm ơn thầy đã luôn nhiệt tình hướng dẫn, giải đáp mọi thắc mắc và đưa ra những góp ý quý báu, giúp nhóm vượt qua khó khăn và hoàn thiện đồ án. Sự hỗ trợ và chia sẻ từ thầy là động lực lớn để chúng em nỗ lực hơn trong quá trình học tập và nghiên cứu.

Do thời gian và kiến thức còn hạn chế, chắc chắn sẽ không tránh khỏi những thiếu sót. Chúng em rất mong nhận được sự thông cảm và những ý kiến đóng góp từ thầy để nhóm có thể hoàn thiện đồ án tốt hơn trong tương lai.

Một lần nữa, chúng em xin chân thành cảm ơn thầy!

Nhóm sinh viên thực hiện

MỤC LỤC

DANH MỤC BẢNG

DANH MỤC HÌNH ẢNH

CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI

1.1 Lý do chọn đề tài

Độ trễ hạ cánh so với thời gian dự kiến là một vấn đề nhận được sự quan tâm lớn từ hành khách, các hãng hàng không và các nhà quản lý giao thông vận tải. Khác với thời gian bay, thời gian đến hoặc thời gian khởi hành có thể được dự đoán khá chính xác nhờ vào các công cụ hỗ trợ hiện có, việc dự đoán độ trễ hạ cánh phức tạp hơn nhiều do bị ảnh hưởng bởi hàng loạt yếu tố khó kiểm soát. Những yếu tố này bao gồm:

- **Điều kiện bên ngoài:** thời tiết, tắc nghẽn không lưu, và thay đổi lịch trình bay đột xuất.
- **Yếu tố nội tại:** sự cố kỹ thuật, trọng tải chuyến bay, và các vấn đề vận hành hoặc nhân sự.

Những yếu tố trên tạo ra sự không chắc chắn trong dự đoán độ trễ, gây ảnh hưởng đến trải nghiệm hành khách, hoạt động của sân bay và hiệu quả vận hành của hãng hàng không.

Đề tài này trở nên quan trọng không chỉ vì tính thực tiễn của nó trong ngành hàng không mà còn vì tính ứng dụng của các phương pháp phân tích và dự đoán dữ liệu lớn. Việc sử dụng dữ liệu lịch sử từ các chuyến bay và áp dụng các kỹ thuật hiện đại như phân tích thống kê, học máy (Machine Learning) và môi trường xử lý phân tán cho phép:

1. **Cải thiện dịch vụ hành khách:**
    - Dự đoán chính xác giúp hành khách chủ động hơn trong việc lên kế hoạch sau khi hạ cánh, như sắp xếp phương tiện di chuyển hoặc chuẩn bị cho các chuyến bay nối tiếp.
2. **Tối ưu hóa hoạt động của sân bay và hãng hàng không:**
    - Các nhà quản lý có thể lập kế hoạch điều phối nguồn lực, sắp xếp lịch trình chuyến bay, giảm thiểu chi phí vận hành và hạn chế tình trạng quá tải tại sân bay.
3. **Hỗ trợ chiến lược dài hạn:**

- Thông tin dự đoán độ trễ có thể được sử dụng để xây dựng các kế hoạch chiến lược, nâng cao hiệu quả khai thác và cải thiện trải nghiệm khách hàng.

Bằng cách tập trung vào việc dự đoán độ trễ hạ cánh, đề tài này không chỉ mang lại giá trị trong việc tối ưu hóa hoạt động hàng không mà còn mở ra cơ hội nghiên cứu và áp dụng các giải pháp công nghệ tiên tiến trong xử lý dữ liệu lớn.

1.2 Mục tiêu đề tài

Mục tiêu chính của đề tài là ứng dụng các kỹ thuật khai thác dữ liệu tiên tiến trên môi trường phân tán để xây dựng mô hình dự đoán độ trễ hạ cánh so với thời gian dự kiến. Cụ thể, nhóm hướng đến việc thực hiện các bước sau:

1. Thu thập và xử lý dữ liệu từ nguồn chính thống, đảm bảo tính toàn diện và chất lượng.
2. Áp dụng các kỹ thuật tiền xử lý như mã hóa dữ liệu (StringIndexer, One Hot Encoding), chuẩn hóa dữ liệu (Data Standardization), và ghép các đặc trưng (Vector Assembler) nhằm tạo ra tập dữ liệu sạch và sẵn sàng cho huấn luyện.
3. Xây dựng các mô hình dự đoán bằng các thuật toán máy học như Softmax Regression và KNN (K-Nearest Neighbors), tận dụng khả năng xử lý song song và phân tán của Apache Spark.
4. Đánh giá hiệu suất của các mô hình bằng cách sử dụng các chỉ số: Precision, Recall, Accuracy và F1-score, nhằm chọn ra mô hình tối ưu nhất cho bài toán.
5. Đưa ra những hạn chế và hướng phát triển trong tương lai nhằm cải thiện độ chính xác và tính ứng dụng thực tế của mô hình.

1.3 Mô tả Dữ liệu

Nguồn dữ liệu: Link

Cục Thống kê Giao thông Vận tải (BTS), một đơn vị thuộc Bộ Giao thông Vận tải Hoa Kỳ (USDOT), có nhiệm vụ thu thập, phân tích và công bố dữ liệu liên quan đến hệ thống giao thông vận tải tại Hoa Kỳ. BTS đóng vai trò quan trọng trong việc cung cấp thông tin về các

xu hướng giao thông, hỗ trợ các nhà hoạch định chính sách xây dựng chiến lược quản lý giao thông, lập kế hoạch vận tải và cải thiện an toàn.

Bộ dữ liệu được lấy từ dữ liệu tháng 7 của nguồn trên. Dữ liệu gồm 616282 dòng và 11 thuộc tính được mô tả như bảng dưới

Tên thuộc tính	**Kiểu dữ liệu**	**Mô tả**
DAY OF MONTH	Số	Ngày trong tháng (1-31)
DAY OF WEEK	Số	Ngày trong tuần (1-7)
OP_UNIQUE_CARRIER	Chuỗi	Mã nhà cung cấp dịch vụ
OP_CARRIER_FL_NUM	Số	Số hiệu chuyến bay
ORIGIN	Chuỗi	Sân bay xuất phát
DEST	Chuỗi	Sân bay đến
DEP_TIME	Số	Thời gian khởi hành thực tế
TAXI_OUT	Số	Thời gian taxi_out
WHEELS_OFF	Số	Thời gian tắt bánh xe
DISTANCE	Số	Khoảng cách giữa các sân bay (dặm)
LABEL	Số	0: Tới đúng giờ hoặc tới sớm 1: Trễ không quá 20 phút. 2: Trễ hơn 20 phút hoặc hủy chuyến.

Bảng 1: Mô tả thuộc tính

1.4 Mô tả bài toán

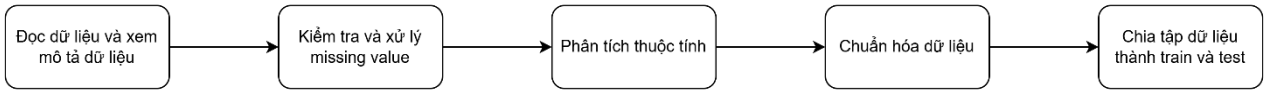
Đề tài tập trung vào việc dự đoán độ trễ hạ cánh của các chuyến bay dựa trên bộ dữ liệu từ Cục Thống kê Giao thông Vận tải (BTS), Hoa Kỳ. Dữ liệu bao gồm nhiều thuộc tính liên quan như ngày tháng, mã nhà cung cấp dịch vụ, sân bay xuất phát, sân bay đến, thời gian khởi hành, và khoảng cách giữa các sân bay.

Quy trình thực hiện bài toán được mô tả như sau:

1. **Tiền xử lý dữ liệu:** Dữ liệu được làm sạch và chuẩn hóa thông qua các bước mã hóa nhãn (StringIndexer), chuyển đổi dữ liệu phân loại sang dạng vector nhị phân (One Hot Encoding), và tổng hợp các đặc trưng thành một cột vector duy nhất (Vector Assembler). Đồng thời, việc chuẩn hóa dữ liệu được thực hiện để đảm bảo tính đồng nhất và cải thiện hiệu quả huấn luyện.
2. **Xây dựng mô hình:** Hai thuật toán máy học, Softmax Regression và KNN, được triển khai trên môi trường phân tán. Trong đó:
    - **Softmax Regression**: Giải quyết bài toán phân loại đa lớp, dự đoán xác suất và nhãn của mỗi chuyến bay dựa trên đặc trưng đầu vào.
    - **KNN**: Sử dụng khoảng cách giữa các điểm dữ liệu để xác định nhãn cho chuyến bay dựa trên các điểm gần nhất trong tập huấn luyện.
3. **Đánh giá và so sánh:** Hiệu suất của các mô hình được đo lường bằng các chỉ số Accuracy, Precision, Recall, và F1-score. Kết quả này giúp xác định mô hình phù hợp nhất với bài toán.

Chương 2: Tiền xử lý dữ liệu

2.1 Quy trình tiền xử lý dữ liệu



Hình 1: Quy trình tiền xử lý

2.2 Đọc và mô tả dữ liệu

Đầu tiên dữ liệu được đọc từ file bằng hàm spark.read.csv sau đó dùng printSchema để xem cấu trúc.

```
root
 |-- DAY_OF_MONTH: integer (nullable = true)
 |-- DAY_OF_WEEK: integer (nullable = true)
 |-- OP_UNIQUE_CARRIER: string (nullable = true)
 |-- OP_CARRIER_FL_NUM: integer (nullable = true)
 |-- ORIGIN: string (nullable = true)
 |-- DEST: string (nullable = true)
 |-- DEP_TIME: integer (nullable = true)
 |-- TAXI_OUT: double (nullable = true)
 |-- WHEELS_OFF: integer (nullable = true)
 |-- DISTANCE: double (nullable = true)
 |-- LABEL: integer (nullable = false)
```

Hình 2: Kiểu dữ liệu của từng thuộc tính

DAY_OF_MONTH	DAY_OF_WEEK	OP_UNIQUE_CARRIER	OP_CARRIER_FL_NUM	ORIGIN	DEST	DEP_TIME	TAXI_OUT	WHEELS_OFF	DISTANCE	LABEL
1	1	9E	4800	CHS	JFK	656	14.0	710	636.0	0
1	1	9E	4801	ATL	TRI	2056	46.0	2142	227.0	1
1	1	9E	4802	FSD	MSP	1113	25.0	1138	196.0	1
1	1	9E	4802	MSP	FSD	902	26.0	928	196.0	0
1	1	9E	4803	TRI	ATL	555	21.0	616	227.0	0
1	1	9E	4804	ABE	ATL	1756	12.0	1808	692.0	0
1	1	9E	4805	ATL	ABE	2051	14.0	2105	692.0	0
1	1	9E	4806	ABE	ATL	601	19.0	620	692.0	0
1	1	9E	4807	TYS	DTW	730	12.0	742	443.0	0
1	1	9E	4810	LGA	SAV	2025	35.0	2100	722.0	1

Hình 3: 10 dòng đầu của bộ dữ liệu

Sử dụng hàm describe.show và count để quan sát thống kê của bảng dữ liệu đồng thời xem số lượng

summary	DAY_OF_MONTH	DAY_OF_WEEK	OP_UNIQUE_CARRIER	OP_CARRIER_FL_NUM	ORIGIN	DEST	DEP_TIME	TAXI_OUT	WHEELS_OFF	DISTANCE	LABEL
count	616282	616282	616282	616282	616282	616282	616282	616282	616282	616282	616282
mean	16.093249518889 08	3.7845547979658662	NULL	2542.2255607011075	NULL	NULL	1338.7989702765942	18.170827640593107	1357.9834247957915	846.9052154695415	0.7168341765620284
stddev	8.960251397098915	2.0120375133737527	NULL	1658.9357326388813	NULL	NULL	529.8060956537737	10.640305468364367	535.0333131091577	607.1517952533083	0.848260768026537
min	1	1	9E	1	ABE	ABE	1	1.0	1	11.0	0
max	31	7	YX	8800	YUM	YUM	2400	214.0	2400	5095.0	2

616282

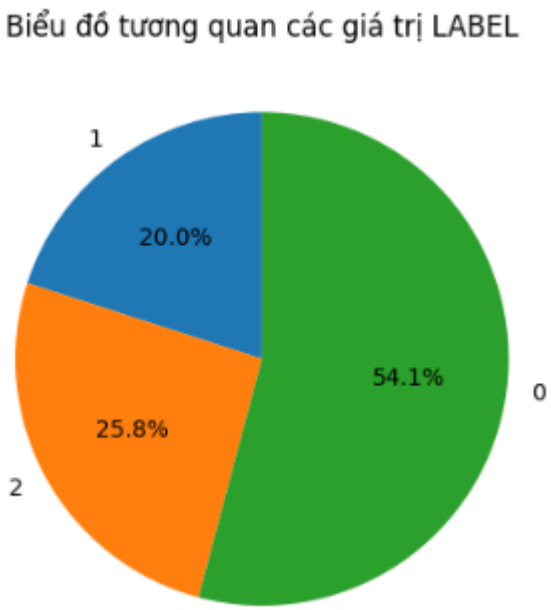
Hình 4: Mô tả dữ liệu

2.3 Kiểm tra và xử lý missing value

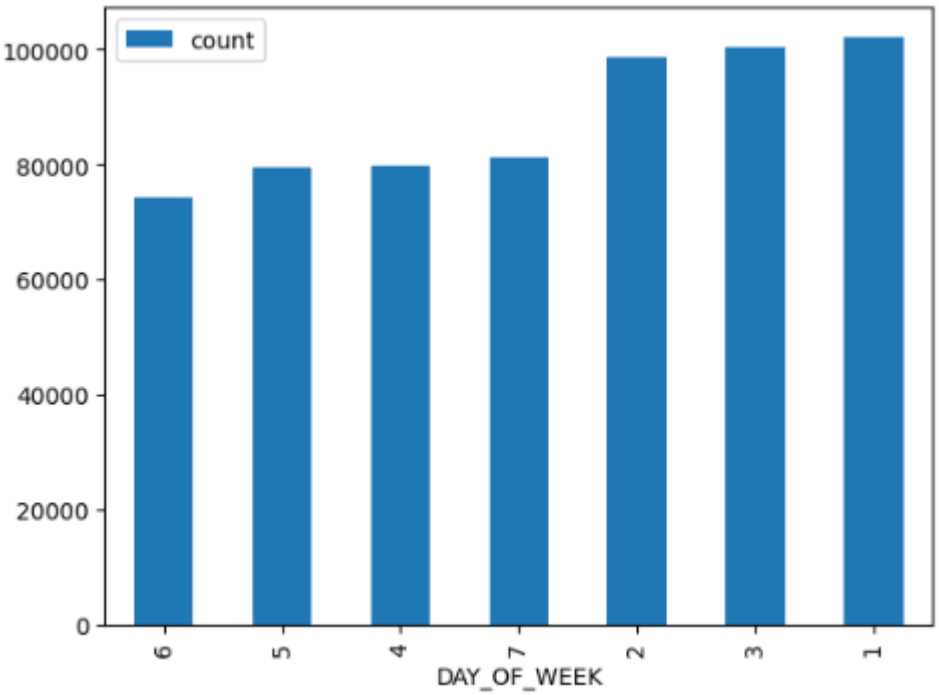
Sử dụng hàm dropna để xóa hết những dòng trống

2.4 Phân tích thuộc tính

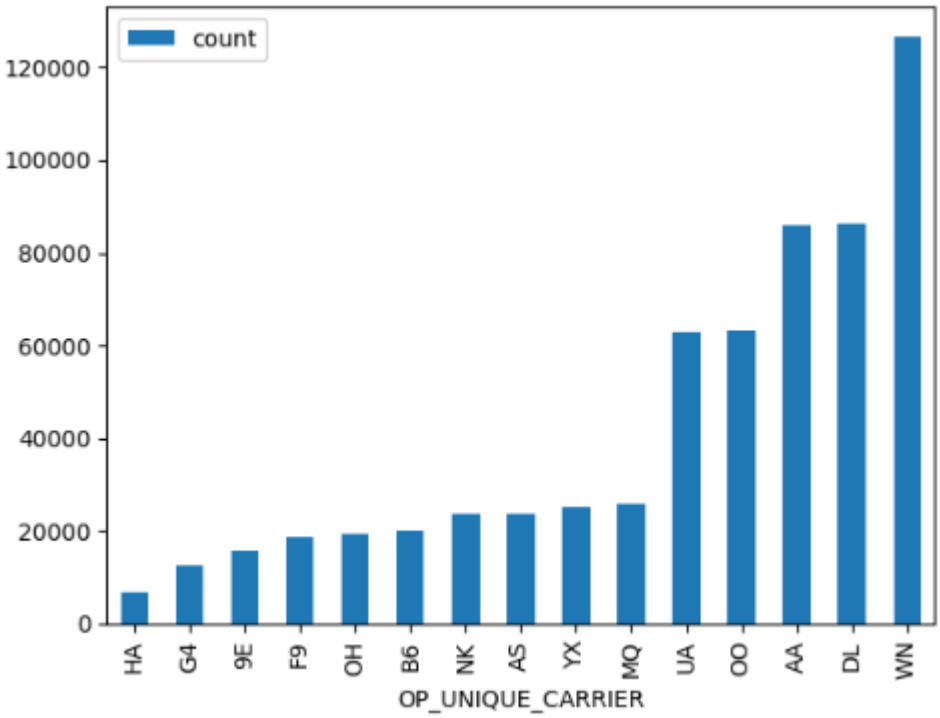
Trước khi phân tích thuộc tính thì em sử dụng hàm distinct để thống kê số thuộc tính khác nhau của từng cột để tìm biểu đồ phù hợp.



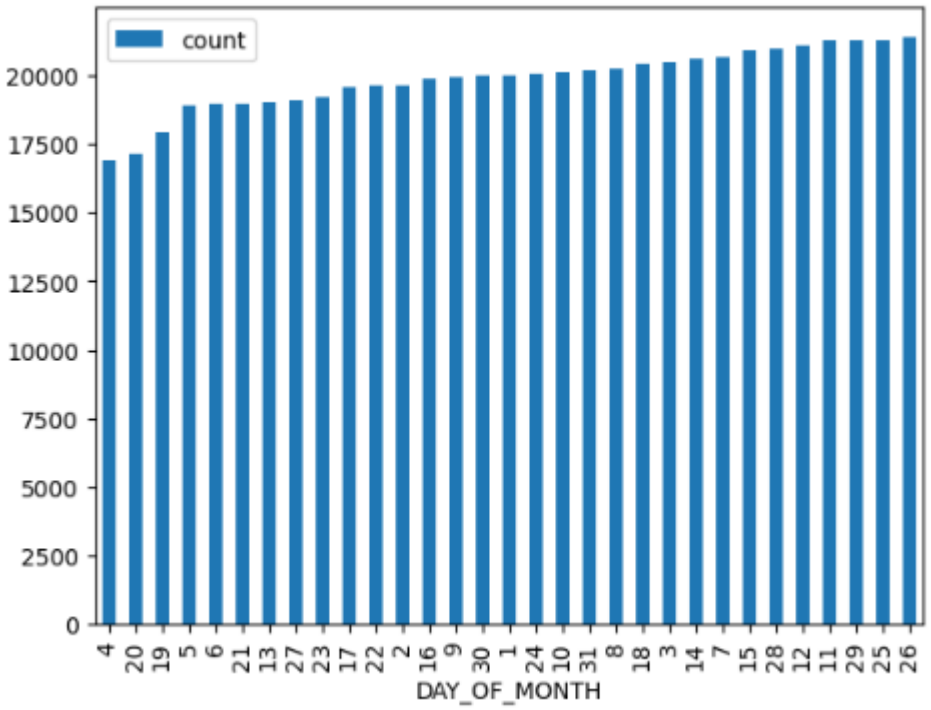
Hình 5: Thống kê tỉ lệ dữ liệu ở cột label



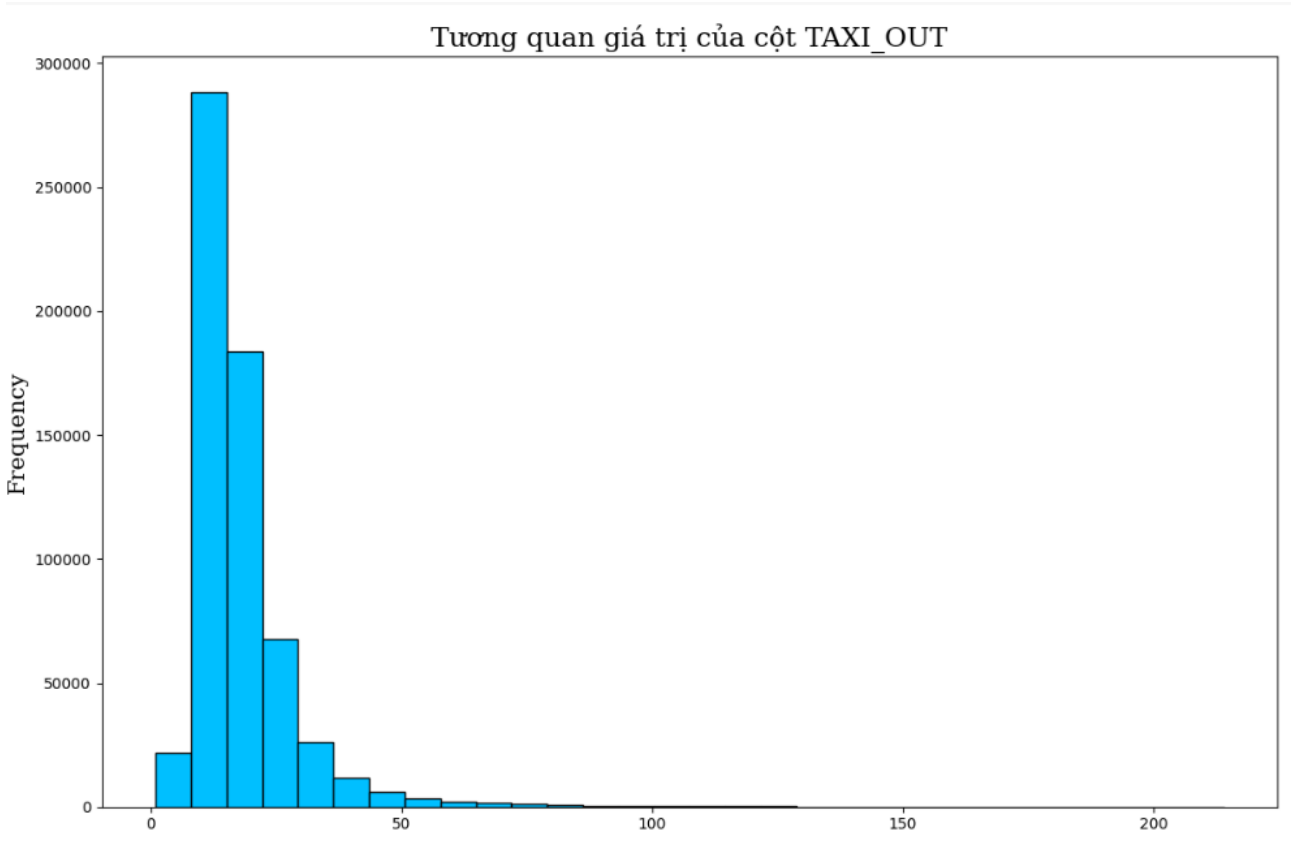
Hình 6: Thống kê tỉ lệ dữ liệu ở cột DAY_OF_WEEK



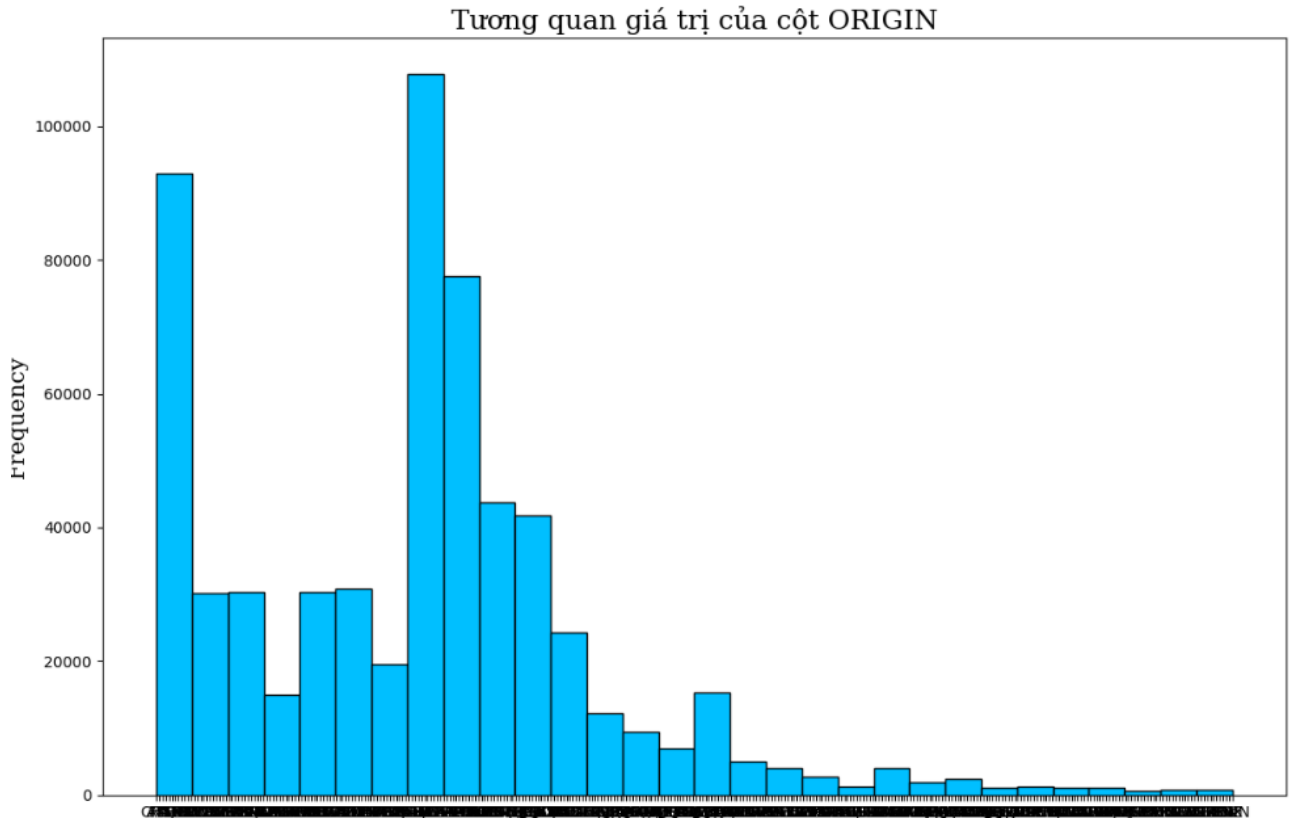
Hình 7: Thống kê tỉ lệ dữ liệu ở cột OP_UNIQUE_CARRIER



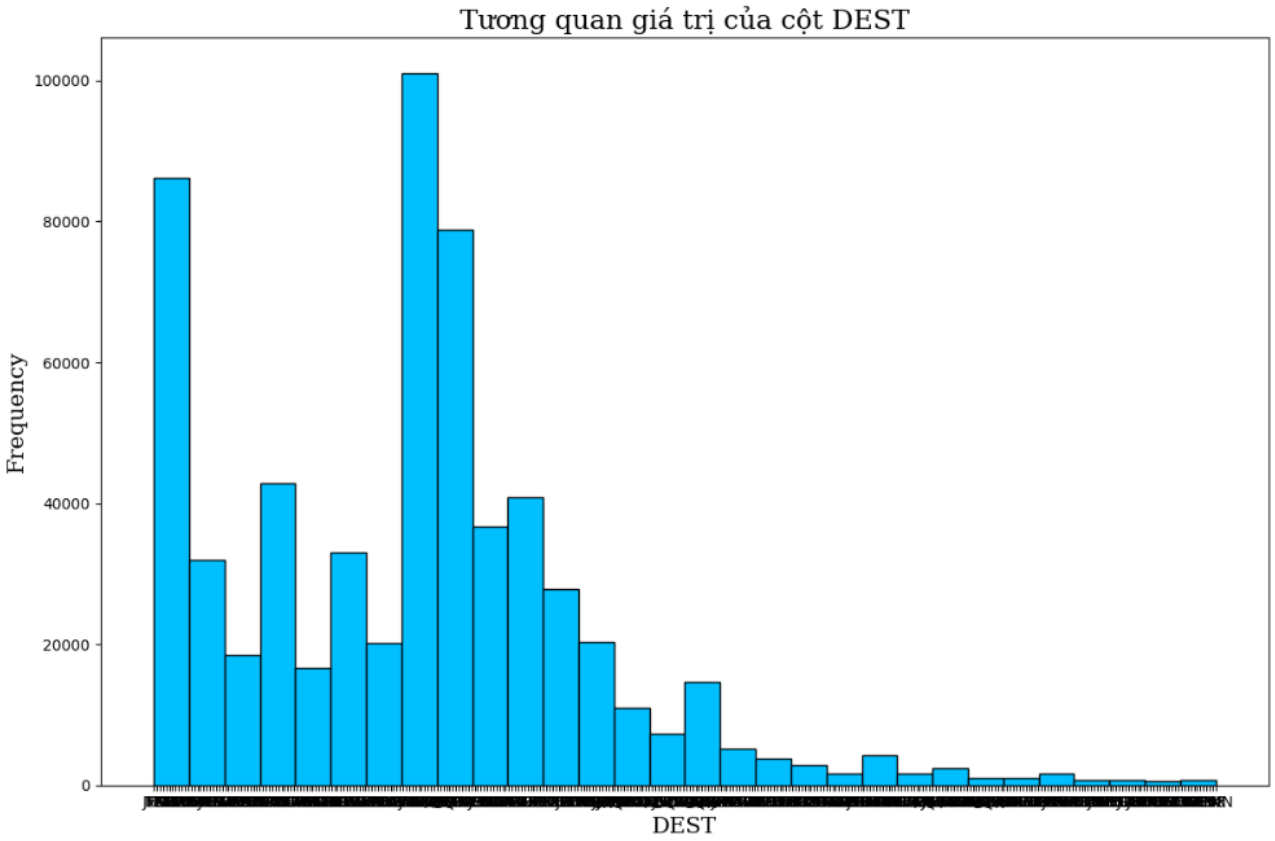
Hình 8: Thống kê tỉ lệ dữ liệu ở cột DAY OF MONTH



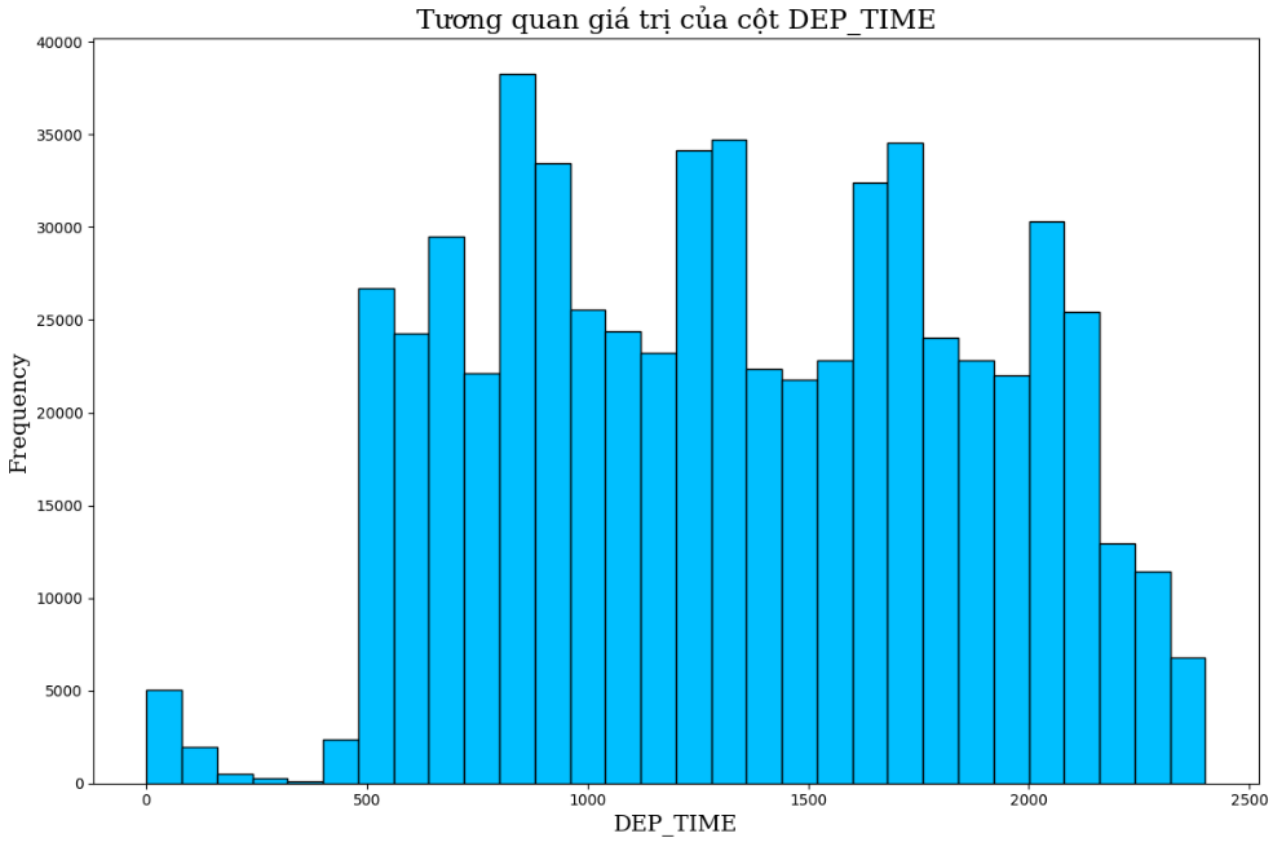
Hình 9: Thống kê tỉ lệ dữ liệu ở cột TAXI_OUT



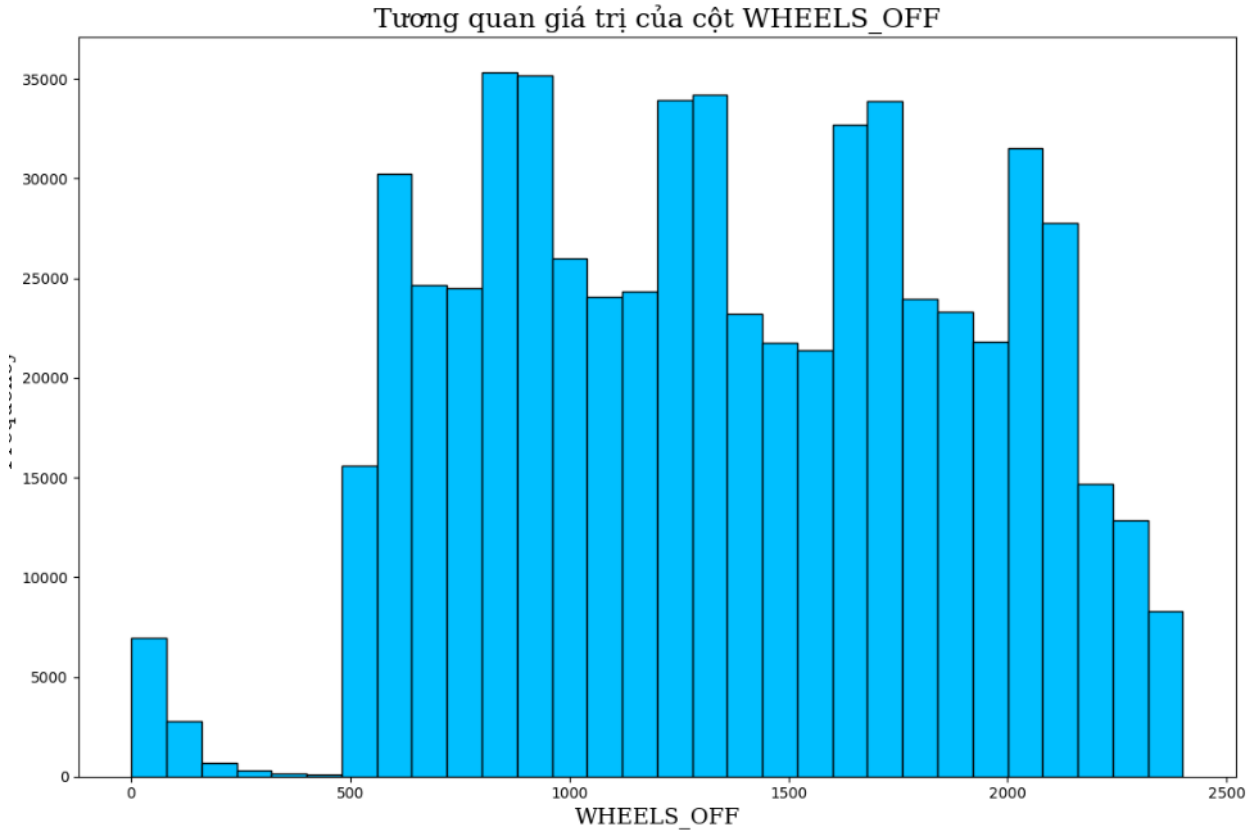
Hình 10: Thống kê tỉ lệ dữ liệu ở cột ORIGIN



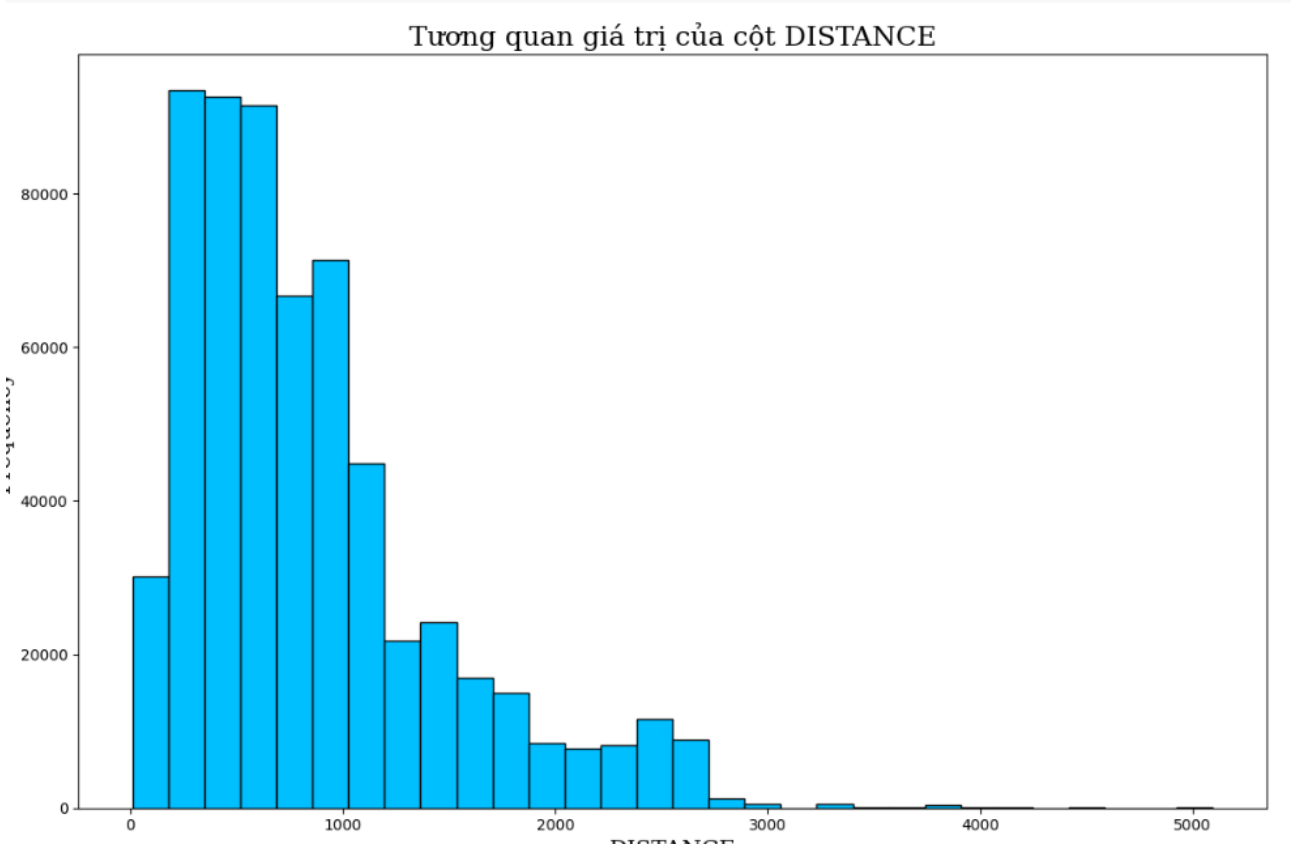
Hình 11: Thống kê tỉ lệ dữ liệu ở cột DEST



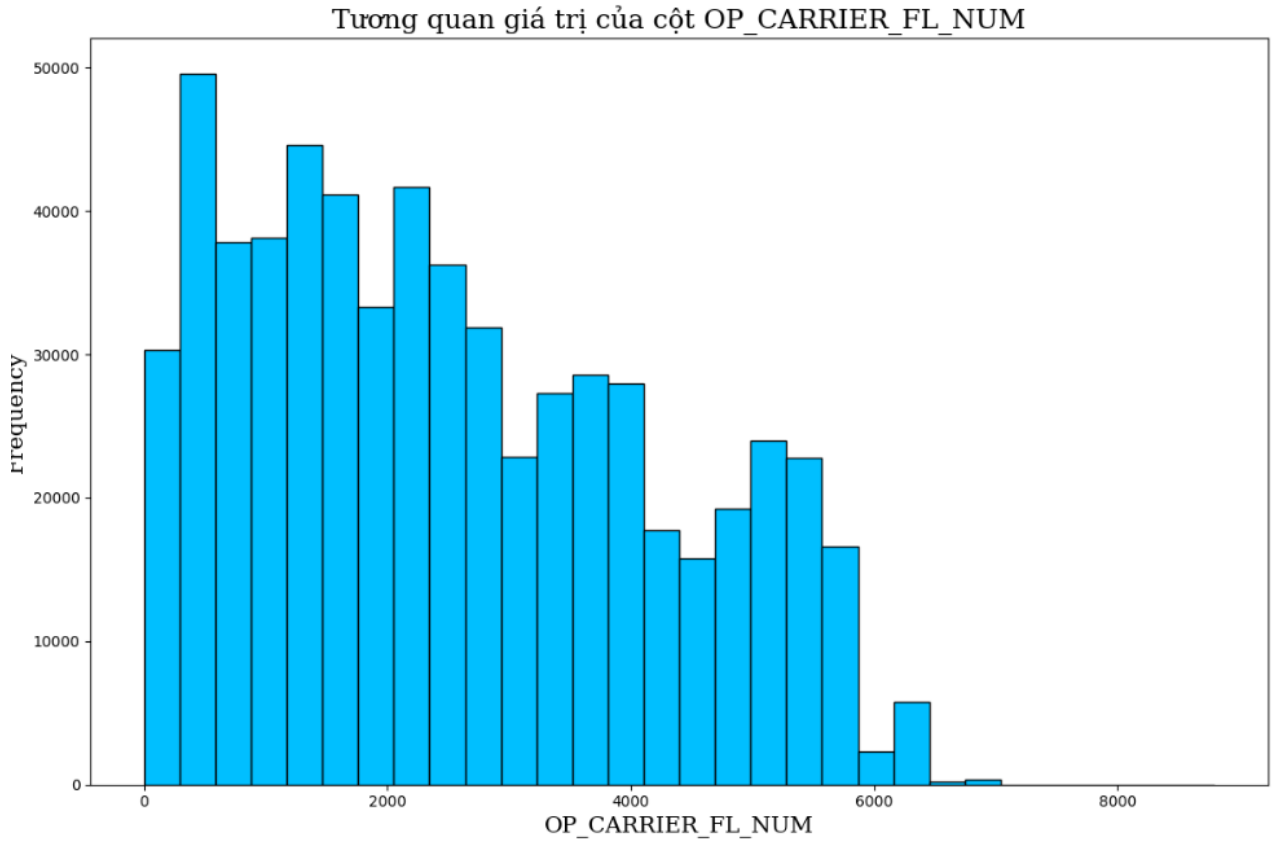
Hình 12: Thống kê tỉ lệ dữ liệu ở cột DEP_TIME



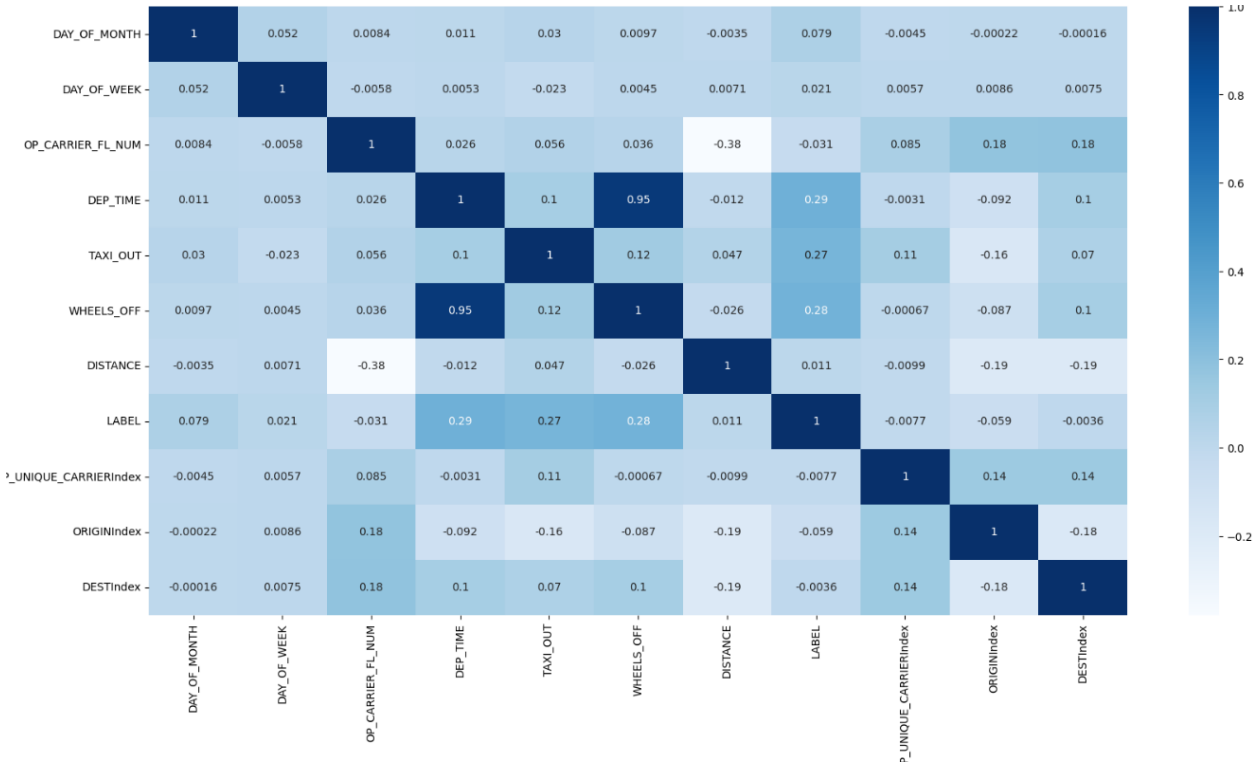
Hình 13: Thống kê tỉ lệ dữ liệu ở cột WHEELS_OFF



Hình 14: Thống kê tỉ lệ dữ liệu ở cột DISTANCE



Hình 15: Thống kê tỉ lệ dữ liệu ở cột OP_CARRIER_FL_NUM



Hình 16: Biểu đồ ma trận hệ số tương quan pearson

2.6 Chuẩn hóa dữ liệu

2.6.1 String Indexer

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

OP_UNIQUE_CARRIER	OP_UNIQUE_CARRIERIndex
OO	4.0
UA	3.0
HA	14.0
WN	0.0
B6	7.0
AS	9.0
9E	10.0
G4	13.0
DL	1.0
F9	12.0
AA	2.0
OH	11.0
YX	5.0
MQ	8.0
NK	6.0

Hình 17: String Indexer ở cột OP_UNIQUE_CARRIER

Chúng em thực hiện String Indexer ở 3 cột mang kiểu dữ liệu object là 'OP_UNIQUE_CARRIER', 'ORIGIN', 'DEST' thành các cột 'OP_UNIQUE_CARRIERIndex', 'ORIGINIndex', 'DESTIndex'

2.6.2 One Hot Encoding

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.

OP_UNIQUE_CARRIERIndex	OP_UNIQUE_CARRIERVec
2.0	(14,[2],[1.0])
5.0	(14,[5],[1.0])
0.0	(14,[0],[1.0])
6.0	(14,[6],[1.0])
13.0	(14,[13],[1.0])
14.0	(14,[],[])
10.0	(14,[10],[1.0])
3.0	(14,[3],[1.0])
8.0	(14,[8],[1.0])
7.0	(14,[7],[1.0])
1.0	(14,[1],[1.0])
4.0	(14,[4],[1.0])
9.0	(14,[9],[1.0])
11.0	(14,[11],[1.0])
12.0	(14,[12],[1.0])

Hình 18: One Hot Encoding ở cột OP_UNIQUE_CARRIERIndex

Chúng em thực hiện One Hot Encoding ở 3 cột 'OP_UNIQUE_CARRIERIndex', 'ORIGINIndex', 'DESTIndex' thành các cột 'OP_UNIQUE_CARRIERVec', 'ORIGINVec','DESTVec'

2.6.3 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.

DAY_OF_MONTH	DAY_OF_WEEK	OP_CARRIER_FL_NUM	DEP_TIME	TAXI_OUT	WHEELS_OFF	DISTANCE	LABEL	OP_UNIQUE_CARRIERIndex	ORIGINIndex	DESTIndex	OP_UNIQUE_CARRIER_OneHot	ORIGIN_OneHot	DEST_OneHot	features
1	1	4800	656	14.0	710	636.0	0	12.0	53.0	17.0	(14,[12],[1.0])	(341,[53],[1.0])	(341,[17],[1.0])	(702,[0,1,2,3,4,17,72,377,701],[1.0,14.0,656.0,4800.0,1.0,1.0,1.0,1.0,636.0])
1	1	4801	2056	46.0	2142	227.0	1	12.0	1.0	198.0	(14,[12],[1.0])	(341,[1],[1.0])	(341,[198],[1.0])	(702,[0,1,2,3,4,17,20,558,701],[1.0,46.0,2056.0,4801.0,1.0,1.0,1.0,1.0,227.0])
1	1	4802	1113	25.0	1138	196.0	1	12.0	127.0	14.0	(14,[12],[1.0])	(341,[127],[1.0])	(341,[14],[1.0])	(702,[0,1,2,3,4,17,146,374,701],[1.0,25.0,1113.0,4802.0,1.0,1.0,1.0,1.0,196.0])
1	1	4802	902	26.0	928	196.0	0	12.0	14.0	127.0	(14,[12],[1.0])	(341,[14],[1.0])	(341,[127],[1.0])	(702,[0,1,2,3,4,17,33,487,701],[1.0,26.0,902.0,4802.0,1.0,1.0,1.0,1.0,196.0])
1	1	4803	555	21.0	616	227.0	0	12.0	198.0	1.0	(14,[12],[1.0])	(341,[198],[1.0])	(341,[1],[1.0])	(702,[0,1,2,3,4,17,217,361,701],[1.0,21.0,555.0,4803.0,1.0,1.0,1.0,1.0,227.0])
1	1	4804	1756	12.0	1808	692.0	0	12.0	156.0	1.0	(14,[12],[1.0])	(341,[156],[1.0])	(341,[1],[1.0])	(702,[0,1,2,3,4,17,175,361,701],[1.0,12.0,1756.0,4804.0,1.0,1.0,1.0,1.0,692.0])
1	1	4805	2051	14.0	2105	692.0	0	12.0	1.0	155.0	(14,[12],[1.0])	(341,[1],[1.0])	(341,[155],[1.0])	(702,[0,1,2,3,4,17,20,515,701],[1.0,14.0,2051.0,4805.0,1.0,1.0,1.0,1.0,692.0])
1	1	4806	601	19.0	620	692.0	0	12.0	156.0	1.0	(14,[12],[1.0])	(341,[156],[1.0])	(341,[1],[1.0])	(702,[0,1,2,3,4,17,175,361,701],[1.0,19.0,601.0,4806.0,1.0,1.0,1.0,1.0,692.0])
1	1	4807	730	12.0	742	443.0	0	12.0	76.0	15.0	(14,[12],[1.0])	(341,[76],[1.0])	(341,[15],[1.0])	(702,[0,1,2,3,4,17,95,375,701],[1.0,12.0,730.0,4807.0,1.0,1.0,1.0,1.0,443.0])
1	1	4810	2025	35.0	2100	722.0	1	12.0	9.0	69.0	(14,[12],[1.0])	(341,[9],[1.0])	(341,[69],[1.0])	(702,[0,1,2,3,4,17,28,429,701],[1.0,35.0,2025.0,4810.0,1.0,1.0,1.0,1.0,722.0])

Hình 19: Vector Assembler các thuộc tính đọc lập

2.7 Chia tập dữ liệu

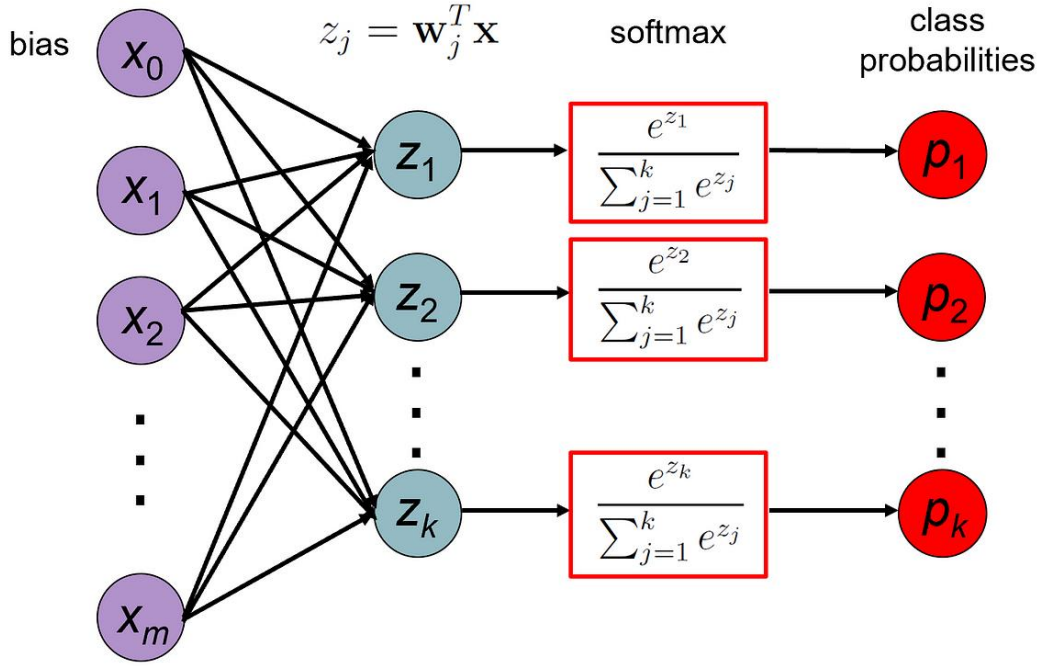
Chúng em sẽ chia ngâu nhiên tập dữ liệu thành train và test theo tỷ lệ 8:2. Sau khi chia và thông kê Label nhận thấy rằng cả trên tập train và test đang mất cân bằng dữ liệu vì số lượng dữ liệu số lượng dữ liệu hủy chuyến thấp hơn.

CHƯƠNG 4: ÁP DỤNG CÁC GIẢI THUẬT KHAI THÁC DỮ LIỆU

4.1 Thuật toán Softmax Regression

4.1.1 Tổng quan về Softmax Regression

Hồi quy Softmax là một phương pháp mở rộng từ hồi quy logistic, dùng để giải quyết các bài toán **phân loại nhiều lớp** (multi-class classification). Trong khi hồi quy logistic chỉ xử lý các bài toán **phân loại nhị phân**, **Softmax Regression** mở rộng khả năng phân loại thành **nhiều lớp rời rạc**. Đây là kỹ thuật được áp dụng rộng rãi trong các hệ thống học máy hiện đại, đặc biệt là các bài toán yêu cầu dự đoán phân phối xác suất của nhiều lớp.



Hình 20: Tổng quan Softmax Regression

Phương pháp Softmax Regression xử lý dữ liệu dưới dạng:

- Tập dữ liệu đầu vào có **m** đặc trưng (**features**) và **n** quan sát (**observations**).
- Nhãn dữ liệu thuộc **k** lớp rời rạc (**classes**) mà mỗi quan sát sẽ được gán vào một trong số đó.

Chuẩn hóa dữ liệu là bước quan trọng để đảm bảo dữ liệu đầu vào có tính đồng nhất, giúp mô hình dễ dàng học và xử lý hiệu quả.

4.1.1 Ưu điểm của thuật toán

- Phân phối xác suất rõ ràng: Hàm Softmax cho phép mô hình tính toán xác suất cho từng lớp, đảm bảo tổng các xác suất bằng 1.
- Ý nghĩa xác suất: Giá trị đầu ra từ hàm Softmax có thể được diễn giải như một phân phối xác suất, rất phù hợp cho các bài toán phân loại trong học máy.
- Khả năng chuẩn hóa: Softmax hạn chế ảnh hưởng của các giá trị cực trị và dữ liệu ngoại lệ mà không cần can thiệp vào dữ liệu gốc.

4.1.2 Hạn chế của thuật toán

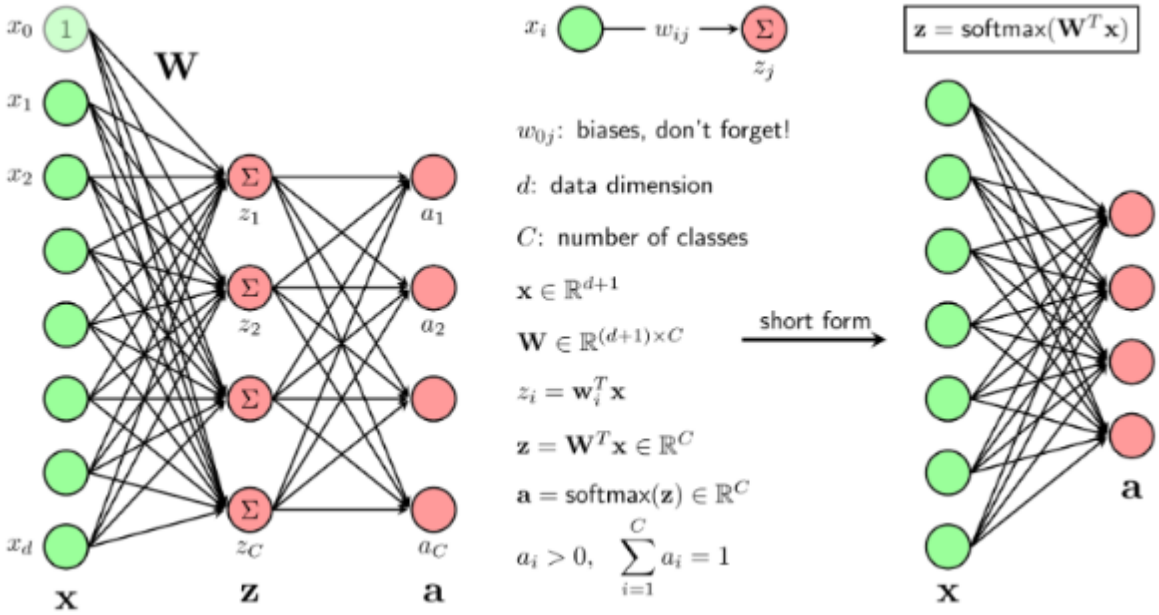
- Độ phức tạp tính toán cao: Khi số lượng lớp hoặc số chiều của dữ liệu tăng, việc tính toán hàm Softmax sẽ tiêu tốn nhiều tài nguyên và thời gian hơn.
- Dễ bị ảnh hưởng bởi dữ liệu không cân bằng: Nếu một lớp có nhiều dữ liệu hơn các lớp còn lại, mô hình có thể bị lệch và kém chính xác.

4.1.3 Phương pháp

Chúng ta cần một mô hình xác suất sao cho với mỗi input x, a_i thể hiện xác suất để input đó rơi vào class i. Vậy điều kiện cần là các a_i phải dương và tổng của chúng bằng 1. Để có thể thỏa mãn điều kiện này, chúng ta cần nhìn vào mọi giá trị z_i và dựa trên quan hệ giữa các z_i này để tính toán giá trị của a_i . Ngoài các điều kiện a_i lớn hơn 0 và có tổng bằng 1, chúng ta sẽ thêm một điều kiện cũng rất tự nhiên nữa, giá trị $z_i = W_i^T X$ càng lớn thì xác suất dữ liệu rơi vào class i càng cao. Điều kiện cuối này chỉ ra rằng chúng ta cần một hàm đồng biến ở đây.

Chú ý rằng z_i có thể nhận giá trị cả âm và dương. Một hàm số mượt đơn giản có thể chắc chắn biến z_i thành một giá trị dương, và hơn nữa, đồng biến, là hàm $\exp(z_i) = e^{z_i}$. Điều kiện cuối cùng, tổng các a_i bằng 1 có thể được đảm bảo nếu: $a_i = \frac{\exp(z_i)}{\sum_{j=1}^{C} \exp(z_i)}, \forall j = 1,2, \dots, C$

Hàm số này, tính tất cả các a_i dựa vào tất cả các a_i , thỏa mãn tất cả các điều kiện đã xét: dương, tổng bằng 1, giữ được thứ tự của z_i. Hàm số này được gọi là softmax function.

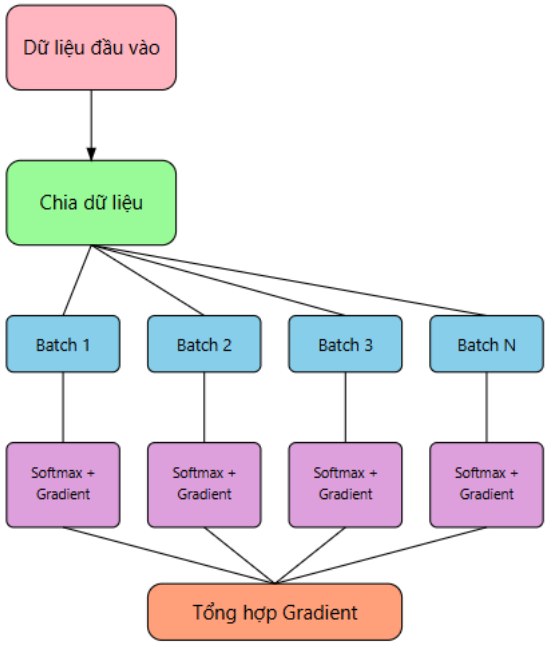


Hình 21: Hàm Softmax Regression

4.1.4 Lý do chọn thuật toán

- **Phù hợp với bài toán đa lớp:** Bài toán có 3 nhãn (không trễ, trễ ít, trễ nhiều), và Softmax Regression được thiết kế để giải quyết các bài toán phân loại nhiều lớp.
- **Khả năng dự đoán xác suất:** Softmax Regression trả về xác suất cho từng nhãn, giúp xác định nhãn có xác suất cao nhất và cung cấp độ tin cậy cho dự đoán.
- **Tổng quát tốt:** Với dữ liệu vừa đủ và mối quan hệ đặc trưng - nhãn không quá phức tạp, mô hình có thể học tốt mà tránh quá khớp (overfitting).

4.1.5 Song song hóa giải thuật



Hình 22: Quá trình song song hóa giải thuật Softmax Regression

1. Đầu vào: Dữ liệu huấn luyện (x,y) - x là đặc trưng, y là nhãn
2. Chia dữ liệu:

- Tách thành N batch nhỏ có kích thước bằng nhau
- Mỗi batch độc lập, có thể xử lý song song

3. Xử lý song song trên từng batch:

- Forward: Tính softmax + cross entropy loss
- Backward: Tính gradient của loss với W,b

4. Tổng hợp kết quả:

- Gom gradient từ các batch
- Tính trung bình gradient
- Cập nhật tham số W,b

4.1.6 Chạy Thuật toán

Hàm softmax

```
def softmax(z):
    exp_z = np.exp(z - np.max(z))
    return exp_z / exp_z.sum(axis=0)
```

Tạo model softmax_regression cho rdd

```
def execute_softmax_regression(rdd, Weights_init, eta, tolerance=1e-4, max_count=100):
    spark_ctx = rdd.context
    Weights = Weights_init
    C = Weights_init.shape[1] #Số lớp (num_classes)
    N = rdd.count() #Số mẫu dữ liệu trong tập huấn luyện (RDD)
    input_size = len(rdd.first()[0]) #Số đặc trưng (num_features)

    count = 0
    check_weights_after = 20
    Weights_broadcast = spark_ctx.broadcast(Weights) #Chia sẻ W đến các worker node mà k cần truyền nhiều lần

    def compute_gradient(record):
        xi, label = record
        xi = np.array(xi).reshape(input_size, 1)
        yi = np.zeros((C, 1)) #vector cột y kích thước (C,1) slg lớp
        yi[label] = 1
        ai = softmax(np.dot(Weights_broadcast.value.T, xi))
        gradient = xi.dot((yi - ai).T)
        return gradient

    while count < max_count:
        gradients = rdd.map(compute_gradient).reduce(lambda g1, g2: g1 + g2)
        Weights_new = Weights + eta * gradients
```

```
        count += 1

        if count % (check_weights_after * N) == 0:
            if np.linalg.norm(Weights_new - Weights) < tolerance: #Kiểm tra điều kiện hội tụ
                return Weights_new

        Weights = Weights_new
        Weights_broadcast = spark_ctx.broadcast(Weights)

    return Weights
```

Khởi tạo tham số

```
# Khởi tạo tham số
num_features = len(dataRDD.first()[0])
num_classes = dataRDD.map(lambda x: x[1]).distinct().count() #lấy nhãn từ dataRDD khác nhau
Weights_init = np.random.randn(num_features, num_classes)
eta = 0.01
tolerance = 1e-4
max_count = 1000
```

Tính toán ma trận dựa vào thuật toán Softmax Regression

```
Weights_final = execute_softmax_regression(dataRDD, Weights_init, eta, tolerance, max_count)
```

```
print("Final weights:", Weights_final)
```

Đưa ra ma trận

Final weights: [[6.26321143e+03 -4.22380276e+02 -5.84395837e+03]

[-5.56118676e+04 1.18685146e+04 4.37450048e+04]

[-2.14080551e+04 -2.75053271e+04 4.89149102e+04]

...

[5.43640690e-03 2.58197189e-03 3.88709027e-01]

[-6.39608746e-01 1.51941750e+00 4.77937039e-01]

[5.97548577e+04 -2.92422393e+04 -3.05100524e+04]]

Tạo hàm predit label để chọn lớp có xác suất cao nhất

```
# Hàm predict label
def predict(x, Weights):
    scores = np.dot(Weights.T, x) # tính W^tX
    return np.argmax(softmax(scores), axis=0) #Dự đoán nhãn bằng cách chọn lớp có xác suất cao nhất
```

Tính toán các độ đo

```python
def calculate_metrics(confusion_matrix):
    num_classes = confusion_matrix.shape[0]

    precision = np.zeros(num_classes)
    recall = np.zeros(num_classes)
    f1 = np.zeros(num_classes)

    for i in range(num_classes):
        tp = confusion_matrix[i, i]
        fp = np.sum(confusion_matrix[:, i]) - tp
        fn = np.sum(confusion_matrix[i, :]) - tp
        tn = np.sum(confusion_matrix) - (tp + fp + fn)

        precision[i] = tp / (tp + fp) if (tp + fp) > 0 else 0
        recall[i] = tp / (tp + fn) if (tp + fn) > 0 else 0
        f1[i] = 2 * precision[i] * recall[i] / (precision[i] + recall[i]) if (precision[i] + recall[i]) > 0 else 0

    accuracy = np.trace(confusion_matrix) / np.sum(confusion_matrix)
    macro_precision = np.mean(precision)
    macro_recall = np.mean(recall)
    macro_f1 = np.mean(f1)

    return accuracy, macro_precision, macro_recall, macro_f1
```

Hàm đánh giá mô hình trên tập dữ liệu test bằng cách tính ma trận nhầm lẫn và các chỉ số hiệu suất.

```python
def evaluate_model(test_rdd, Weights):
    spark_ctx = test_rdd.context

    def predict_label(record):
        xi, label = record
        xi = np.array(xi).reshape(len(xi), 1)
        predicted_label = predict(xi, Weights)
        return (predicted_label, label) #trả về cặp dự đoán và nhãn thật.

    predictions_and_labels = test_rdd.map(predict_label).collect()

    predictions = [pred for pred, label in predictions_and_labels]
    true_labels = [label for pred, label in predictions_and_labels]

    num_classes = len(set(true_labels))
    confusion_matrix = np.zeros((num_classes, num_classes), dtype=int)

    for true, pred in zip(true_labels, predictions):
        confusion_matrix[true, pred] += 1

    accuracy, precision, recall, f1 = calculate_metrics(confusion_matrix)

    return accuracy, precision, recall, f1
```

In ra kết quả của các độ đo

```
accuracy, precision, recall, f1 = evaluate_model(datatestRDD, Weights_final)
```

```
print(f"Accuracy: {accuracy}")
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1-score: {f1}")
```

```
Accuracy: 0.5394996708360764
Precision: 0.28519371845025243
Recall: 0.3384095738905177
F1-score: 0.2597318531835001
```

4.2 Thuật toán KNN (K-Nearest Neighbors)

4.2.1 Khái niệm về KNN (K-Nearest Neighbors)

K-Nearest Neighbor (KNN) là một thuật toán học có giám sát (supervised learning) thuộc loại đơn giản nhất, nhưng lại hiệu quả trong một số trường hợp. Đây là một thuật toán lazy learning, vì trong giai đoạn huấn luyện, nó không thực hiện quá trình học hỏi nào từ dữ liệu đào tạo. Thay vào đó, mọi tính toán được thực hiện khi cần dự đoán kết quả cho dữ liệu mới. KNN có thể được sử dụng cho cả hai loại bài toán: Phân loại (Classification) và Hồi quy (Regression).

- Trong bài toán Phân loại, nhãn của một điểm dữ liệu mới được xác định dựa trên K điểm lân cận gần nhất trong tập dữ liệu đào tạo. Quyết định về nhãn có thể được đưa ra thông qua bầu chọn đa số (dựa trên số lần xuất hiện của từng nhãn) hoặc bằng cách áp dụng các trọng số khác nhau cho các điểm lân cận, tùy thuộc vào khoảng cách của chúng đến điểm cần dự đoán.
- Trong bài toán Hồi quy, giá trị đầu ra của một điểm dữ liệu mới có thể được xác định là giá trị của điểm gần nhất (nếu K = 1) hoặc là trung bình có trọng số của các điểm lân cận. Nói ngắn gọn, KNN dự đoán đầu ra của một điểm dữ liệu mới chỉ bằng thông tin từ K điểm gần nhất trong tập dữ liệu đào tạo, bất kể một số điểm trong số đó có thể là nhiễu.

Lý do lựa chọn thuật toán KNN:

- **Đơn giản và hiệu quả**:
    - KNN là một trong những thuật toán học có giám sát đơn giản nhất, áp dụng hiệu quả cho cả bài toán phân loại và hồi quy. Thuật toán này hoàn toàn đáp ứng yêu cầu khai thác dữ liệu trong đồ án.
- **Không yêu cầu thời gian huấn luyện**:
    - KNN được gọi là Lazy Learning vì không thực hiện bất kỳ quá trình huấn luyện nào trên dữ liệu. Nó chỉ lưu trữ tập dữ liệu đào tạo trong bộ nhớ và thực hiện các tính toán khi có dữ liệu mới cần dự đoán.
- **Dự đoán đơn giản và linh hoạt**:
    - Việc dự đoán kết quả của dữ liệu mới rất đơn giản, do không cần quá trình đào tạo trước. Dữ liệu mới có thể được thêm vào tập dữ liệu mà không làm giảm độ chính xác của thuật toán.
- **Không có tham số phức tạp**:
    - KNN là một thuật toán phi tham số (nonparametric), với tính toán được thực hiện độc lập cho từng điểm dữ liệu. Nó yêu cầu rất ít tham số, chủ yếu chỉ cần:
        - K (số điểm lân cận cần xem xét).
        - Một phương pháp đo khoảng cách.
    - So với nhiều thuật toán Machine Learning khác, KNN có yêu cầu tham số thấp hơn đáng kể.

4.2.2 Ưu điểm của thuật toán

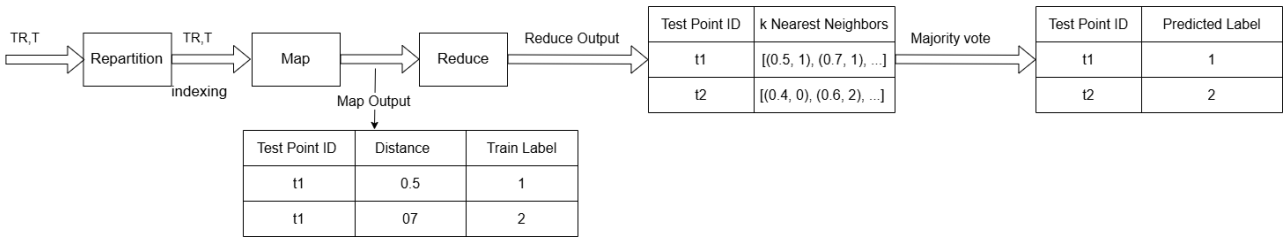
- Dễ hiểu và đơn giản: KNN là một thuật toán dễ tiếp cận và đơn giản, không yêu cầu người dùng phải có kiến thức chuyên sâu về lý thuyết thuật toán.

- Tính linh hoạt: KNN có thể áp dụng hiệu quả cho cả nhiệm vụ phân loại và hồi quy. Ngoài ra, nó có thể được mở rộng để phân loại nhiều lớp chỉ với một số điều chỉnh nhỏ.
- Không cần giai đoạn huấn luyện: KNN không đòi hỏi quá trình huấn luyện trước. Mô hình hoạt động bằng cách lưu trữ dữ liệu đào tạo trong bộ nhớ và thực hiện phân loại hoặc dự đoán cho dữ liệu mới ngay lập tức, theo thời gian thực.
- Khả năng xử lý dữ liệu nhiễu: KNN hoạt động tốt ngay cả khi dữ liệu có nhiễu hoặc ngoại lệ, nhờ vào việc không yêu cầu giả định về phân phối dữ liệu. Thuật toán này giảm thiểu tác động của nhiễu bằng cách tính trung bình giá trị từ các điểm lân cận gần nhất.

4.2.3 Nhược điểm của thuật toán

- Lựa chọn giá trị K: Việc chọn giá trị K phù hợp đóng vai trò quan trọng và có thể ảnh hưởng đáng kể đến hiệu suất của thuật toán. K quá nhỏ có thể gây ra hiện tượng overfitting, trong khi K quá lớn có thể dẫn đến underfitting.
- Chi phí tính toán cao: KNN yêu cầu tính toán khoảng cách giữa điểm dữ liệu mới và tất cả các điểm trong tập dữ liệu huấn luyện, dẫn đến việc tiêu tốn nhiều thời gian và tài nguyên, đặc biệt khi xử lý các tập dữ liệu lớn.

4.2.4 Song song hóa giải thuật



Hình 23:Quá trình song song hóa giải thuật KNN.

Trong môi trường dữ liệu lớn, việc KNN phải tính toán khoảng cách giữa từng điểm kiểm tra và tất cả các điểm huấn luyện trở nên rất tốn kém về mặt tính toán, đặc biệt khi kích thước tập dữ liệu lớn. Để giải quyết vấn đề này, các khung tính toán phân tán như MapReduce được sử dụng để song song hóa quá trình tính toán, giúp thuật toán trở nên khả thi và mở rộng hơn.

Bằng cách chia dữ liệu thành các phân vùng nhỏ, việc tính toán khoảng cách được xử lý độc lập trên nhiều nút trong giai đoạn Map. Kết quả sau đó được tổng hợp lại trong giai đoạn Reduce, nơi k-lân cận gần nhất cho từng điểm kiểm tra được xác định và tiến hành bỏ phiếu đa số để phân loại hoặc hồi quy

4.2.4.1 Giai đoạn Map

Quy trình bắt đầu với giai đoạn Input, trong đó tập dữ liệu huấn luyện (TR) và tập dữ liệu kiểm tra (T) được cung cấp. Tập dữ liệu huấn luyện được phân chia lại (repartition) thành các tập con nhỏ hơn để xử lý song song, đồng thời các chỉ số (indexing) được sử dụng để đảm bảo phân phối dữ liệu hiệu quả cho các tác vụ tiếp theo. Trong giai đoạn Map, khoảng cách giữa mỗi điểm kiểm tra trong tập dữ liệu kiểm tra (T) và tất cả các điểm huấn luyện trong tập dữ liệu huấn luyện (TR) được tính toán. Mỗi tác vụ Map tạo ra đầu ra trung gian bao gồm ID của điểm kiểm tra, khoảng cách đã tính toán, và nhãn của điểm huấn luyện. Ví dụ, đối với điểm kiểm tra t1, khoảng cách như 0.5 (nhãn 1) và 0.7 (nhãn 2) được tính toán. Các đầu ra này đại diện cho tất cả các lân cận tiêm năng của điểm kiểm tra.

Ví dụ Map output:

ID Điểm Kiểm Tra	Khoảng Cách	Nhãn Huấn Luyện
t1	0.5	1
t1	0.7	2

4.2.4.2 Giai đoạn Reduce

Trong giai đoạn Reduce, các đầu ra từ giai đoạn Map được tổng hợp. Đối với mỗi điểm kiểm tra, reducer xác định k lân cận gần nhất bằng cách lựa chọn k khoảng cách nhỏ nhất và nhãn tương ứng. Kết quả được biểu diễn dưới dạng các bộ đôi (khoảng cách, nhãn) đối với từng điểm kiểm tra. Ví dụ, điểm kiểm tra t1 có các lân cận gần nhất là [(0.5, 1), (0.7, 1), ...].

Trong bước cuối, cơ chế bỏ phiếu đa số được áp dụng vào các lân cận gần nhất để xác định nhãn dự đoán cho mỗi điểm kiểm tra. Nhãn xuất hiện nhiều nhất (bỏ phiếu đa số) sẽ được gán là nhãn dự đoán cuối cùng.

Ví dụ kết quả dự đoán:

ID Điểm Kiểm Tra	Nhãn Dự Đoán
t1	1
t2	2

Bảng 2: Ví dụ kết quả dự đoán KNN

4.3 Chạy thuật toán

Hàm tính khoảng cách

```
def euclidean_distance(point1, point2):
    return np.sqrt(np.sum((np.array(point1) - np.array(point2)) ** 2))
```

Gán ID phân vùng cho train và test RDDS

```
def knn_distributed(train_rdd, test_rdd, k):
    train_rdd_with_id = train_rdd.zipWithIndex().map(lambda x: (x[1] % 10, x[0]))  # (partition_id, train_point)
    test_rdd_with_id = test_rdd.zipWithIndex().flatMap(
        lambda x: [(i, (x[1], x[0])) for i in range(10)]  # Broadcast các test point và các phân vùng
    )
```

Kết giữa tập test và train theo từng phân vùng

```
    joined_rdd = test_rdd_with_id.join(train_rdd_with_id)  # (partition_id, (test_point, train_point))
```

Tính khoảng cách giữa điểm test và các điểm train và kết lại

```
    distances_rdd = joined_rdd.map(
        lambda x: (
            x[1][0][0],  # test_idx
            (euclidean_distance(x[1][0][1][0], x[1][1][0]), x[1][1][1]),  # (distance, train_label)
        )
    )
```

Group by theo ID của điểm test và tìm K láng giềng gần nhất

```
    neighbors_rdd = distances_rdd.groupByKey().mapValues(
        lambda neighbors: heapq.nsmallest(k, neighbors, key=lambda x: x[0])
    )
```

Thực hiện bỏ phiếu đa số để tìm nhãn dự đoán

```
    predictions = neighbors_rdd.map(
        lambda test_idx_neighbors: (
            test_idx_neighbors[0],
            max(
                set([label for _, label in test_idx_neighbors[1]]),
                key=lambda label: [n[1] for n in test_idx_neighbors[1]].count(label),
            ),
        )
    )
    return predictions
```

Thực hiện đánh giá độ chính xác và ma trận nhầm lẫn của mô hình Map Reduce KNN.

```
accuracy = np.trace(conf_matrix) / np.sum(conf_matrix)
num_classes = conf_matrix.shape[0]
precision = []
recall = []
f1_scores = []
for i in range(num_classes):
    tp = conf_matrix[i, i]
    fp = np.sum(conf_matrix[:, i]) - tp
    fn = np.sum(conf_matrix[i, :]) - tp
    precision_i = tp / (tp + fp) if (tp + fp) > 0 else 0
    recall_i = tp / (tp + fn) if (tp + fn) > 0 else 0
    f1_i = 2 * precision_i * recall_i / (precision_i + recall_i) if (precision_i + recall_i) > 0 else 0

    precision.append(precision_i)
    recall.append(recall_i)
    f1_scores.append(f1_i)
macro_precision = np.mean(precision)
macro_recall = np.mean(recall)
macro_f1 = np.mean(f1_scores)
# Print the results
print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {macro_precision:.4f}")
print(f"Recall: {macro_recall:.4f}")
print(f"F1-Score: {macro_f1:.4f}")
```

Kết quả thu được: Accuracy: 0.4496, Precision: 0.3239, Recall: 0.3275, F1-Score: 0.3135

4.3 Đánh giá mô hình

Trong bài toán dự đoán nhãn, sử dụng các độ đo như Accuracy, Precision, Recall, và F1-score để đo lường hiệu suất của mô hình. Accuracy đo tỷ lệ dự đoán đúng, Precision tập trung

vào việc giảm thiểu dự đoán sai dương tính, Recall tối đa hóa việc phát hiện các trường hợp thực sự dương tính, và F1-score cung cấp cái nhìn cân bằng giữa Precision và Recall. Các độ đo này cung cấp cái nhìn toàn diện về khả năng phân loại của mô hình, hỗ trợ quyết định và điều chỉnh trong quá trình phát triển mô hình.

4.3.1 Accuracy

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. Tuy nhiên điều này không đúng với các bài toán có bộ dữ liệu bị mất cân bằng, xem Công thức.

$$Accuracy = \frac{Số\ dự\ đoán\ đúng}{Tổng\ số\ dự\ đoán} = \frac{TP + TN}{TP + TN + FP + FN}$$

4.3.2 Precision & Recall

Precision & Recall Hai 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. Recall đo lường số lượng các trường hợp đúng được truy xuất chia cho tất cả các trường hợp đúng.

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

4.3.3 F1–score

F–score được định nghĩa là giá trị trung bình có trọng số precision và recall tùy thuộc vào hàm trọng số β.

$$F1\ score\ =\ 2\ x\ \frac{P\ x\ R}{P\ +\ R}$$

4.3.4 Đánh giá các mô hình đã thực nghiệm

Thuật toán	Accuracy	Precision	Recall	F1-Score

Softmax Regression	0.5395	0.2852	0.3384	0.2597
KNN	0.4496	0.3239	0.3275	0.3135

Bảng 3: Kết quả các mô hình thực nghiệm

- Nếu ưu tiên Accuracy (tỉ lệ dự đoán đúng toàn bộ), thì Softmax Regression tốt hơn.
- Nếu muốn cân bằng giữa Precision và Recall, hoặc cần tập trung vào F1-Score, thì KNN là lựa chọn tốt hơn

Chương 5: KẾT LUẬN

5.1 Kết quả đạt được

Trong quá trình thực hiện đề tài, nhóm đã đạt được các kết quả sau:

1. **Hiểu rõ bài toán và tập dữ liệu:**
    - Phân tích, làm sạch, và chuẩn hóa dữ liệu từ nguồn cung cấp, đảm bảo chất lượng và tính chính xác trước khi huấn luyện mô hình.
    - Hiểu rõ mối quan hệ giữa các thuộc tính và cách chúng ảnh hưởng đến nhãn đầu ra.
2. **Xây dựng và triển khai mô hình dự đoán:**
    - Sử dụng các thuật toán như Softmax Regression và KNN trên môi trường phân tán Apache Spark để dự đoán độ trễ của chuyến bay.
    - Tối ưu hóa quá trình tiền xử lý dữ liệu với các kỹ thuật mã hóa nhãn, chuẩn hóa, và ghép vector đặc trưng.
3. **Đánh giá và so sánh mô hình:**
    - Đánh giá hiệu suất dự đoán của các mô hình qua các chỉ số: Accuracy, Precision, Recall và F1-score.
    - Kết quả cho thấy Softmax Regression đạt độ chính xác tổng thể cao hơn, trong khi KNN mang lại sự cân bằng tốt giữa Precision và Recall.
4. **Xác định hướng cải thiện:**
    - Nhóm đã ghi nhận những hạn chế của mô hình như vấn đề mất cân bằng dữ liệu và chi phí tính toán cao, qua đó đề xuất các phương pháp khắc phục tiềm năng.

Kết quả của đề tài không chỉ thể hiện khả năng ứng dụng các thuật toán máy học mà còn góp phần đưa ra giải pháp thực tế nhằm nâng cao hiệu quả quản lý và dự đoán độ trễ hạ cánh trong ngành hàng không.

5.2 Hướng phát triển

Giải quyết mất cân bằng dữ liệu:

- Áp dụng các kỹ thuật như oversampling lớp thiểu số hoặc undersampling lớp đa số để cân bằng lại tập dữ liệu.

- Thử nghiệm các phương pháp như SMOTE (Synthetic Minority Over-sampling Technique) để tạo thêm các mẫu dữ liệu lớp thiểu số.

Ứng dụng mô hình học sâu (Deep Learning):

- Sử dụng các kiến trúc mạng thần kinh như Artificial Neural Networks (ANN) để khai thác các mối quan hệ phức tạp trong dữ liệu.
- Kết hợp với các framework như BigDL để tận dụng tối đa sức mạnh tính toán song song trên Spark.

Tối ưu hóa mô hình:

- Tinh chỉnh các tham số của thuật toán, áp dụng kỹ thuật Grid Search hoặc Random Search để tìm cấu hình tốt nhất.
- Sử dụng Cross-validation để đảm bảo mô hình hoạt động ổn định và tổng quát hơn trên dữ liệu mới.

Khả năng triển khai thực tế:

- Đưa mô hình vào thử nghiệm trong hệ thống thực tế của các sân bay hoặc hãng hàng không để kiểm tra tính ứng dụng.
- Tích hợp các nguồn dữ liệu bổ sung, chẳng hạn dữ liệu thời tiết hoặc thông tin không lưu, để tăng độ chính xác của dự đoán.

BẢNG PHÂN CHIA CÔNG VIỆC

MSSV	Họ tên	Phân công	Đánh giá
21520597	Võ Hồng Kim Anh	- Tìm hiểu và Thực hiện thuật toán Softmax Regression - Nghiên cứu bài toán - Đánh giá kết quả bài toán - Làm Slide, Word	Hoàn thành tốt, đầy đủ, đúng hạn (10/10)
21521847	Trần Xuân Bằng	- Nghiên cứu bài toán - Tìm hiểu và Thực hiện thuật toán KNN - Đánh giá kết quả bài toán - Làm Slide, Word	Hoàn thành tốt, đầy đủ, đúng hạn (10/10)
21522751	Lê Văn Tuấn	- Nghiên cứu bài toán - Tìm kiếm, nghiên cứu bộ dữ liệu - Thực hiện tiền xử lý dữ liệu - Làm Slide, Word	Hoàn thành tốt, đầy đủ, đúng hạn (10/10)

Bảng 4: Bảng phân chia công việc

TÀI LIỆU THAM KHẢO

[1]. Hướng dẫn từng bước về thuật toán KNN: Hiểu các nguyên tắc cơ bản và ứng dụng, ichi.pro. [Truy cập: 2-Jun-2024]. [Trực tuyến]. Có sẵn: https://ichi.pro/vi/huong-dan-tung-buoc-ve-thuat-toan-knn-hieu-cac-nguyen-tac-co-ban-va-ung-dung-54135645769301.

[2]. Machine Learning cơ bản, machinelearningcoban.com. [Truy cập: 2-Jun-2024]. [Trực tuyến]. Có sẵn: https://machinelearningcoban.com.

[3]. "What Is k-Nearest Neighbors (k-NN)?" IBM. [Truy cập: 2-Jun-2024]. [Trực tuyến]. Có sẵn: https://www.ibm.com/topics/knn#:~:text=The%20k%2Dnearest%20neighbors%20(KNN,used%20in%20machine%20learning%20today.

[4]. L. E. Sucar, "Introduction to Machine Learning and Data Mining," in Neural Computing & Applications, vol. 27, no. 8, pp. 2401-2402, Oct. 2016. [Trực tuyến]. Có sẵn: https://link.springer.com/article/10.1007/s00521-016-2401-x.

[5]. Phan Tiến Ngọc, Hồ Quốc Thư, Đặng Quang Anh Tuấn, “Mô hình Softmax Regression”, MMLabUIT, 2024, [Trực tuyến]. Có sẵn: https://mmlab.uit.edu.vn/tutorials/ml/gradient-based-model/softmax_regression