BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH

BÁO CÁO CUỐI KỲ

MÔN: PHÂN TÍCH DỮ LIỆU

ĐỀ TÀI: PHÂN TÍCH DỮ LIỆU ĐỂ DỰ ĐOÁN GIAO HÀNG TRỄ TRONG CHUỖI CUNG ỨNG CỦA DATACO

GVHD: NGUYỄN VĂN THÀNH

SINH VIÊN THỰC HIỆN(NHÓM 13):

1. Mai Hồng Hải	22133014
2. Vy Gia Nghi	22133037
3. Nguyễn Ngọc Hiếu Hảo	22133015

Thành phố Hồ Chí Minh, ngày 17 tháng 05 năm 2025

PHÂN CÔNG NHIỆM VỤ

Nhóm:13

Lớp:DAAN436277

Nhiệm vụ	Nguyễn Ngọc Hiếu Hảo	Mai Hồng hải	Vy Gia Nghi
Mở đầu, giới thiệu dự án		X	
Tìm kiếm dữ liệu	X		
Mô tả dữ liệu			X
Tiền xử lý dữ liệu	X		X
EDA: Phân tích doanh thu liên quan đến vấn đề giao hàng	X		
EDA: Phân tích vận hành/giao hàng	X		
Biến đổi đặc trưng	X		X
Xây dựng mô hình học máy		X	
Đánh giá mô hình học máy		X	
Dự đoán kết quả		X	
Phân tích kết quả		X	X
Kết luận & kiến nghị			X

NHẬN XÉT CỦA GIÁO VIÊN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

ĐIỂM CỦA GV

MỤC LỤC

1. MỞ ĐẦU

1.1. Giới thiệu dự án

Mục tiêu: Dự án này tập trung vào việc xây dựng mô hình dự đoán nguy cơ giao hàng trễ trong chuỗi cung ứng của DataCo Global. Mục tiêu chính là xác định các yếu tố then chốt dẫn đến tình trạng giao hàng không đúng hẹn, từ đó cung cấp một công cụ mạnh mẽ hỗ trợ DataCo chủ động đưa ra các biện pháp phòng ngừa và tối ưu hóa quy trình vận hành.

Giới thiệu bộ dữ liệu Supply Chain của DataCo Global: Dự án sử dụng bộ dữ liệu toàn diện về chuỗi cung ứng do DataCo Global cung cấp, bao gồm thông tin chi tiết về đơn hàng, khách hàng, sản phẩm, vận chuyển và các yếu tố liên quan khác. Đây là một nguồn tài nguyên quý giá, chứa đựng nhiều thông tin tiềm năng để hiểu sâu sắc về động lực và hiệu suất của chuỗi cung ứng. Việc phân tích bộ dữ liệu lớn này hứa hẹn sẽ khám phá ra những mối tương quan và phức tạp mà các phương pháp truyền thống khó có thể nhận diện.

Ứng dụng thực tiễn: Kết quả của dự án này có ý nghĩa ứng dụng thực tiễn to lớn đối với DataCo Global. Một mô hình dự đoán giao hàng trễ chính xác sẽ cho phép công ty:

- **Chủ động quản lý rủi ro:** Nhận diện sớm các đơn hàng có nguy cơ trễ hẹn, từ đó có biện pháp can thiệp kịp thời.
- **Nâng cao hiệu quả vận hành:** Tối ưu hóa các quy trình liên quan đến giao nhận, giảm thiểu các yếu tố gây ra sự chậm trễ.
- **Cải thiện trải nghiệm khách hàng:** Đảm bảo khách hàng nhận được hàng đúng thời gian cam kết, tăng cường sự hài lòng và lòng trung thành.
- **Giảm thiểu chi phí:** Tránh các chi phí phát sinh do giao hàng trễ, chẳng hạn như bồi thường hoặc xử lý các vấn đề phát sinh.

1.2. Bối cảnh và động lực thực hiện dự án

Sự phát triển mạnh mẽ của công nghệ và sự gia tăng theo cấp số nhân của dữ liệu đã mở ra những cơ hội mới trong việc phân tích và tối ưu hóa các quy trình phức tạp như chuỗi cung ứng. DataCo, với lượng dữ liệu hoạt động đồ sộ, đang đứng trước cơ hội tận dụng nguồn tài nguyên này để hiểu rõ hơn về các yếu tố như hiệu quả giao hàng, quản lý tồn kho, và hành vi khách hàng.

Động lực thực hiện dự án này xuất phát từ nhu cầu thực tế của các doanh nghiệp trong việc nâng cao hiệu quả chuỗi cung ứng, giảm thiểu chi phí, và cải thiện trải nghiệm khách hàng. Bằng cách phân tích sâu rộng tập dữ liệu của DataCo, dự án này kỳ vọng sẽ làm sáng tỏ các điểm nghẽn, xác định các yếu tố rủi ro tiềm ẩn (ví dụ như giao hàng trễ), và khám phá các cơ hội để tối ưu hóa toàn bộ quy trình, từ đó mang lại lợi ích thiết thực cho DataCo nói riêng và các doanh nghiệp hoạt động trong lĩnh vực chuỗi cung ứng nói chung.

1.3. Phương pháp và cấu trúc báo cáo

Để đạt được các mục tiêu đề ra, dự án này sẽ tiếp cận bài toán bằng cách kết hợp các phương pháp phân tích dữ liệu định lượng và định tính. Quy trình phân tích sẽ bao gồm các giai đoạn chính: thu thập và tiền xử lý dữ liệu, phân tích khám phá (EDA) để hiểu rõ đặc điểm của dữ liệu, biến đổi và mã hóa đặc trưng để chuẩn bị cho việc xây dựng mô hình, xây dựng và đánh giá các mô hình phân loại (với trọng tâm là dự đoán giao hàng trễ), và cuối cùng là diễn giải kết quả

và đưa ra các khuyến nghị.

Cấu trúc báo cáo được tổ chức một cách logic, đi từ tổng quan về dự án và dữ liệu (Mục 1 và 2) đến các bước phân tích chi tiết (Mục 3 đến 8), và kết thúc bằng những kết luận và kiến nghị quan trọng (Mục 9). Phần tài liệu tham khảo (Mục 10) sẽ liệt kê các nguồn thông tin được sử dụng trong quá trình thực hiện dự án.

2. MÔ TẢ DỮ LIỆU

2.1. Tổng quan dữ liệu

Bộ dữ liệu : DataCoSupplyChainDataset.csv

- Chứa thông tin chi tiết về các giao dịch và hoạt động trong chuỗi cung ứng của DataCo Global. Sau khi tải và kiểm tra, dữ liệu bao gồm **180.519 dòng** (entries) và **53 cột** (columns), cho thấy một quy mô dữ liệu lớn, phù hợp cho việc phân tích bằng các kỹ thuật dữ liệu lớn.

Nguồn dữ liệu: https://www.kaggle.com/datasets/shashwatwork/dataco-smart-supply-chain-for-big-data-analysis?select=DataCoSupplyChainDataset.csv

Kiểu dữ liệu:

- Số nguyên (int64): 14 cột, thường chứa các định danh (ID), số ngày, và các thuộc tính đếm.
- Số thực (float64): 15 cột, đại diện cho các giá trị đo lường như lợi nhuận, doanh số, vĩ độ, kinh độ, và các tỷ lệ.
- Đối tượng (object): 24 cột, chứa các biến kiểu chuỗi hoặc hỗn hợp, bao gồm thông tin về loại sản phẩm, trạng thái giao hàng, tên danh mục, địa điểm khách hàng và đơn hàng, email, mật khẩu, phân khúc khách hàng, thị trường, khu vực, trạng thái đơn hàng, hình ảnh sản phẩm, tên sản phẩm, và phương thức vận chuyển.

2.2. Giải thích các cột quan trọng

Để hiểu rõ hơn về dữ liệu và chuẩn bị cho các bước phân tích tiếp theo, việc xác định và giải thích ý nghĩa của các cột quan trọng là rất cần thiết. Dưới đây là mô tả sơ bộ về một số cột có vai trò quan trọng trong việc dự đoán nguy cơ giao hàng trễ và phân tích chuỗi cung ứng:

- **Type:** Loại đơn hàng (ví dụ: TRANSFER, PAYMENT).
- **Days for shipping (real):** Số ngày thực tế để giao hàng.
- **Days for shipment (scheduled):** Số ngày dự kiến để giao hàng.
- **Benefit per order:** Lợi nhuận trên mỗi đơn hàng.
- **Sales per customer:** Tổng doanh số cho mỗi khách hàng.
- **Delivery Status:** Trạng thái giao hàng cuối cùng (ví dụ: Late Delivery, Advance Shipping).
- **Late_delivery_risk:** Biến mục tiêu, cho biết đơn hàng có bị giao trễ hay không (1: Trễ, 0: Không trễ). Đây là cột quan trọng nhất cho mục tiêu dự đoán của dự án.
- **Category Name:** Tên danh mục sản phẩm.

- **Customer City, Customer Country, Customer State, Customer Segment:** Thông tin về địa lý và phân khúc khách hàng.
- **Department Name:** Tên bộ phận sản phẩm.
- **Market, Order Region, Order Country, Order State, Order City:** Thông tin về thị trường và địa điểm đặt hàng.
- **order date (DateOrders):** Ngày đặt hàng.
- **shipping date (DateOrders):** Ngày giao hàng (thực tế hoặc dự kiến).
- **Shipping Mode:** Phương thức vận chuyển.
- **Order Item Discount, Order Item Discount Rate:** Giá trị và tỷ lệ giảm giá trên sản phẩm trong đơn hàng.
- **Order Item Product Price:** Giá niêm yết của sản phẩm trong đơn hàng.
- **Order Item Quantity:** Số lượng sản phẩm trong đơn hàng.
- **Sales:** Tổng giá trị đơn hàng.
- **Product Name:** Tên sản phẩm.
- **Product Price:** Giá bán của sản phẩm.

Các cột khác cũng chứa những thông tin giá trị và sẽ được xem xét kỹ hơn trong quá trình phân tích khám phá dữ liệu (EDA) để hiểu rõ hơn về mối quan hệ giữa chúng và biến mục tiêu.

3. PHƯƠNG PHÁP & CÔNG CỤ

3.1. Quy trình phân tích dữ liệu

Dự án này sẽ tuân theo một quy trình phân tích dữ liệu khoa học và có hệ thống, bao gồm các giai đoạn chính sau:

1. **Thu thập dữ liệu:** Bước đầu tiên là thu thập bộ dữ liệu "DataCoSupplyChainDataset.csv" từ nguồn cung cấp (Kaggle).
2. **Tiền xử lý dữ liệu:** Giai đoạn này tập trung vào việc làm sạch và chuẩn bị dữ liệu cho các bước phân tích tiếp theo. Các hoạt động chính bao gồm xử lý dữ liệu trùng lặp, xử lý giá trị thiếu, loại bỏ các cột không liên quan hoặc chứa quá nhiều giá trị thiếu, chuyển đổi kiểu dữ liệu nếu cần, và chuẩn hóa hoặc chuẩn hóa dữ liệu để đảm bảo tính nhất quán và phù hợp cho các thuật toán học máy.
3. **Phân tích khám phá (EDA):** Mục tiêu của EDA là hiểu sâu hơn về cấu trúc, đặc điểm, và mối quan hệ giữa các biến trong dữ liệu. Các kỹ thuật trực quan hóa và thống kê mô tả sẽ được sử dụng để phát hiện các xu hướng, mẫu, và các vấn đề tiềm ẩn trong dữ liệu.
4. **Biến đổi và mã hóa đặc trưng (Feature Engineering):** Trong giai đoạn này, các biến hiện có có thể được biến đổi hoặc kết hợp để tạo ra các đặc trưng mới có ý nghĩa hơn và cải thiện hiệu suất của mô hình. Đối với các biến phân loại, chúng sẽ được mã hóa sang định dạng số mà các thuật toán học máy có thể xử lý. Các biến thời gian cũng có thể được xử lý để trích xuất các thông tin hữu ích.
5. **Xây dựng và đánh giá mô hình:** Giai đoạn này tập trung vào việc lựa chọn, huấn luyện, và đánh giá các mô hình học máy phù hợp cho bài toán dự đoán nguy cơ giao hàng trễ (bài toán phân loại). Một số thuật toán tiềm năng bao gồm Logistic Regression, Decision Trees, Random Forests, và các mô hình khác. Hiệu suất của các mô hình sẽ được đánh giá bằng nhiều phương pháp khác nhau.
6. **Dự đoán và diễn giải kết quả:** Mô hình tốt nhất sau quá trình đánh giá sẽ được sử dụng

để dự đoán nguy cơ giao hàng trễ trên dữ liệu mới hoặc dữ liệu kiểm tra. Quan trọng hơn, giai đoạn này cũng bao gồm việc phân tích và diễn giải ý nghĩa của các đặc trưng quan trọng đối với kết quả dự đoán, cung cấp những hiểu biết giá trị cho DataCo.

7. **Kết luận và kiến nghị:** Dựa trên kết quả phân tích và mô hình dự đoán, báo cáo sẽ đưa ra các kết luận chính và đề xuất các kiến nghị cụ thể nhằm cải thiện quy trình giao hàng và giảm thiểu nguy cơ giao hàng trễ cho DataCo.

3.2. Công cụ và thư viện sử dụng

```
# Thư viện phân tích dữ liệu
import pandas as pd
import numpy as np
import plotly.express as px

# Thư viện trực quan hóa
import seaborn as sns
import matplotlib.pyplot as plt

# Thư viện xử lý & xây dựng mô hình máy học
from sklearn.preprocessing import MinMaxScaler, StandardScaler,
LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeRegressor
from sklearn.naive_bayes import GaussianNB

from sklearn.pipeline import make_pipeline
from sklearn.model_selection import train_test_split, cross_val_score,
KFold, StratifiedKFold, ShuffleSplit
from sklearn.metrics import r2_score, confusion_matrix, accuracy_score,
roc_curve, auc, classification_report
```

4. TIỀN XỬ LÝ DỮ LIỆU

Giai đoạn tiền xử lý dữ liệu đóng vai trò quan trọng trong việc làm sạch và chuẩn bị dữ liệu thô cho các bước phân tích và xây dựng mô hình tiếp theo. Mục tiêu là giải quyết các vấn đề như giá trị thiếu, dữ liệu trùng lặp, các cột không cần thiết hoặc có phương sai thấp, và đảm bảo định dạng dữ liệu phù hợp.

4.1. Xử lý dữ liệu trùng lặp

Kiểm tra các giá trị trùng lặp có trong tập dữ liệu

```
df.duplicated().sum()
```

Output: np.int64(0)

Vì trong trường hợp này dữ liệu không có dữ liệu bị trùng lặp nên không cần phải xóa đi các giá trị trùng lặp này nếu như có thì phải thực hiện xóa các dữ liệu trùng lặp này đi.

4.2. Xử lý giá trị thiếu

```
dfna = df.isna().sum().sort_values(ascending=False)
dfna.head()
```

```
Order Zipcode              155679
Customer Lname                  8
Customer Zipcode                3
Type                            0
Days for shipping (real)        0
dtype: int64
```

Trong tập dữ liệu nếu như một cột có quá nhiều dữ liệu bị thiếu thì cách tốt nhất nên bỏ cột đó thay vì thay thế bằng giá trị phổ biến, trung bình, trung vị hoặc xóa các dòng có dữ liệu bị thiếu đó. Vì nếu thay thế giá trị thì sẽ mất đi sự đa dạng của dữ liệu và không tốt cho xây dựng mô hình còn xóa các dòng thì lại mất đi quá nhiều dữ liệu nên chọn cách xóa cột có dữ liệu bị thiếu nhiều như ở đây là Order Zipcode

- Bước 1: Loại bỏ các cột có số lượng giá trị thiếu quá nhiều (>50% dữ liệu):

```
df.drop(dfna.index[np.where(dfna > df.shape[0]/2)], axis=1,
inplace=True)
```

Cụ thể ở đây sẽ xóa các cột có trên 50% dữ liệu bị thiếu

- Điền thiếu cho số & phân loại

Bước 2: Với các biến số, điền giá trị trung bình:

```
num = df.select_dtypes('number')
df[num.columns] = num.fillna(num.mean())
```

Bước 3: Với biến phân loại, điền bằng chuỗi trống:

```
df = df.fillna('')
df.isna().sum().sum()   # Kiểm tra lại tổng số NA
```

```
np.int64(0)
```

Tiếp theo đến các biến là kiểu số thì sẽ điền bằng giá trị trung bình, còn các biến là kiểu khác kiểu số thì điền bằng một chuỗi rỗng.

4.3. Loại bỏ cột không có giá trị phân biệt

Các cột mà tất cả các giá trị đều giống nhau (hoặc có rất ít giá trị duy nhất) không mang lại thông tin hữu ích cho việc phân tích hoặc xây dựng mô hình. Chúng ta xác định và loại bỏ các cột này bằng cách kiểm tra số lượng giá trị duy nhất trong mỗi cột. Các biến có số lượng giá trị duy nhất nhỏ hơn 2 sẽ không mang thông tin phân loại hoặc dự báo.

```
dfunique= df.nunique().sort_values(ascending= True,
na_position='first')
dfunique.head()# Kiểm tra các biến có số lượng unique thấp
```

Kết quả kiểm tra cho thấy các cột sau có số lượng giá trị duy nhất rất thấp:

```
...   Product Description    0
      Customer Password      1
      Product Status         1
      Customer Email         1
      Late_delivery_risk     2
      dtype: int64
```

Dựa trên kết quả này, chúng ta sẽ loại bỏ các cột có số lượng giá trị duy nhất nhỏ hơn 2 (trong trường hợp này là 'Product Description', 'Customer Password', 'Product Status', 'Customer Email'). Lưu ý rằng cột 'Late_delivery_risk' có 2 giá trị duy nhất (0 và 1) nên sẽ không bị loại bỏ vì đây là biến mục tiêu của chúng ta.
Đoạn code sau được sử dụng để loại bỏ các cột này:

```
df.drop(dfunique.index[np.where(dfunique < 2)], axis=1, inplace=True)
```

Sau bước này, các cột không có giá trị phân biệt (hoặc chỉ có một giá trị duy nhất) đã được loại bỏ khỏi DataFrame.

4.4. Phát hiện và xử lý phương sai thấp

Tương tự, các thuộc tính có phương sai rất thấp, nghĩa là các giá trị của chúng ít thay đổi, có thể không đóng góp nhiều vào mô hình học máy. Chúng ta kiểm tra phương sai của các cột số.

```
df.select_dtypes(include='number').var().sort_values().head()
```

```
...   Order Item Discount Rate          0.004958
      Order Item Profit Ratio           0.217898
      Late_delivery_risk                0.247669
      Days for shipment (scheduled)     1.889111
      Order Item Quantity               2.112521
      dtype: float64
```

Trong trường hợp này, dựa trên kết quả kiểm tra, các cột 'Order Item Discount Rate' có phương sai rất thấp so với các cột khác. Tuy nhiên, trong code bạn cung cấp, không có bước loại bỏ cụ thể nào dựa trên ngưỡng phương sai thấp được thực hiện. Việc quyết định ngưỡng phương sai để loại bỏ các thuộc tính thường phụ thuộc vào đặc điểm cụ thể của bài toán và dữ liệu. Trong trường hợp này, chúng ta sẽ ghi nhận thông tin về phương sai thấp nhưng không tiến hành loại bỏ để theo sát quy trình bạn đã thực hiện. Nếu cần thiết, trong các bước phát triển tiếp theo, chúng ta có thể xem xét việc loại bỏ các thuộc tính có phương sai thấp hơn một ngưỡng nhất định.

4.5. Loại bỏ cột không cần thiết

Trong quá trình tiền xử lý dữ liệu, việc xác định và loại bỏ các thuộc tính không cần thiết cho mục tiêu phân tích và xây dựng mô hình học máy là một bước quan trọng. Các thuộc tính này có thể bao gồm các định danh (ID) chủ yếu được sử dụng cho việc liên kết các bảng trong cơ sở dữ liệu, thông tin mang tính chất mô tả không trực tiếp liên quan đến dự đoán (ví dụ: hình ảnh sản phẩm), hoặc các trường có độ biến động thấp hoặc trùng lặp thông tin. Việc loại bỏ các cột không cần thiết giúp giảm chiều dữ liệu, tăng tốc độ huấn luyện mô hình và có thể cải thiện hiệu suất của mô hình bằng cách loại bỏ nhiễu.

```
df.drop(["Product Image", "Order Item Cardprod Id", "Order Item Id",
"Product Card Id",
         "Category Id", "Customer Country", "Customer State", "Customer
Street",
         "Department Id", "Customer Fname", "Order Item Discount",
"Product Category Id"],
        axis=1, inplace=True)
```

Giải thích lý do loại bỏ một số cột:

- **Các cột ID (Order Item Cardprod Id, Order Item Id, Product Card Id, Category Id, Department Id):** Đây là các định danh duy nhất cho từng mục, đơn hàng hoặc sản phẩm. Chúng hữu ích trong việc quản lý cơ sở dữ liệu nhưng thường không mang lại thông tin dự đoán trực tiếp cho mô hình học máy.
- **Product Image:** Đường dẫn đến hình ảnh sản phẩm không chứa thông tin định lượng hoặc phân loại hữu ích cho việc dự đoán nguy cơ giao hàng trễ.
- **Thông tin khách hàng mang tính cá nhân (Customer Country, Customer State, Customer Street, Customer Fname):** Mặc dù thông tin về địa lý có thể liên quan, nhưng ở mức độ chi tiết này (tên đường phố, tên riêng), chúng có thể tạo ra quá nhiều chiều dữ liệu và có thể không khái quát hóa tốt. Các thông tin tổng quát hơn về khách hàng (ví dụ: phân khúc khách hàng) có thể hữu ích hơn.
- **Order Item Discount:** Giá trị giảm giá tuyệt đối có thể tương quan với tỷ lệ giảm giá ('Order Item Discount Rate') đã được giữ lại, do đó, việc giữ cả hai có thể gây ra đa cộng tuyến.

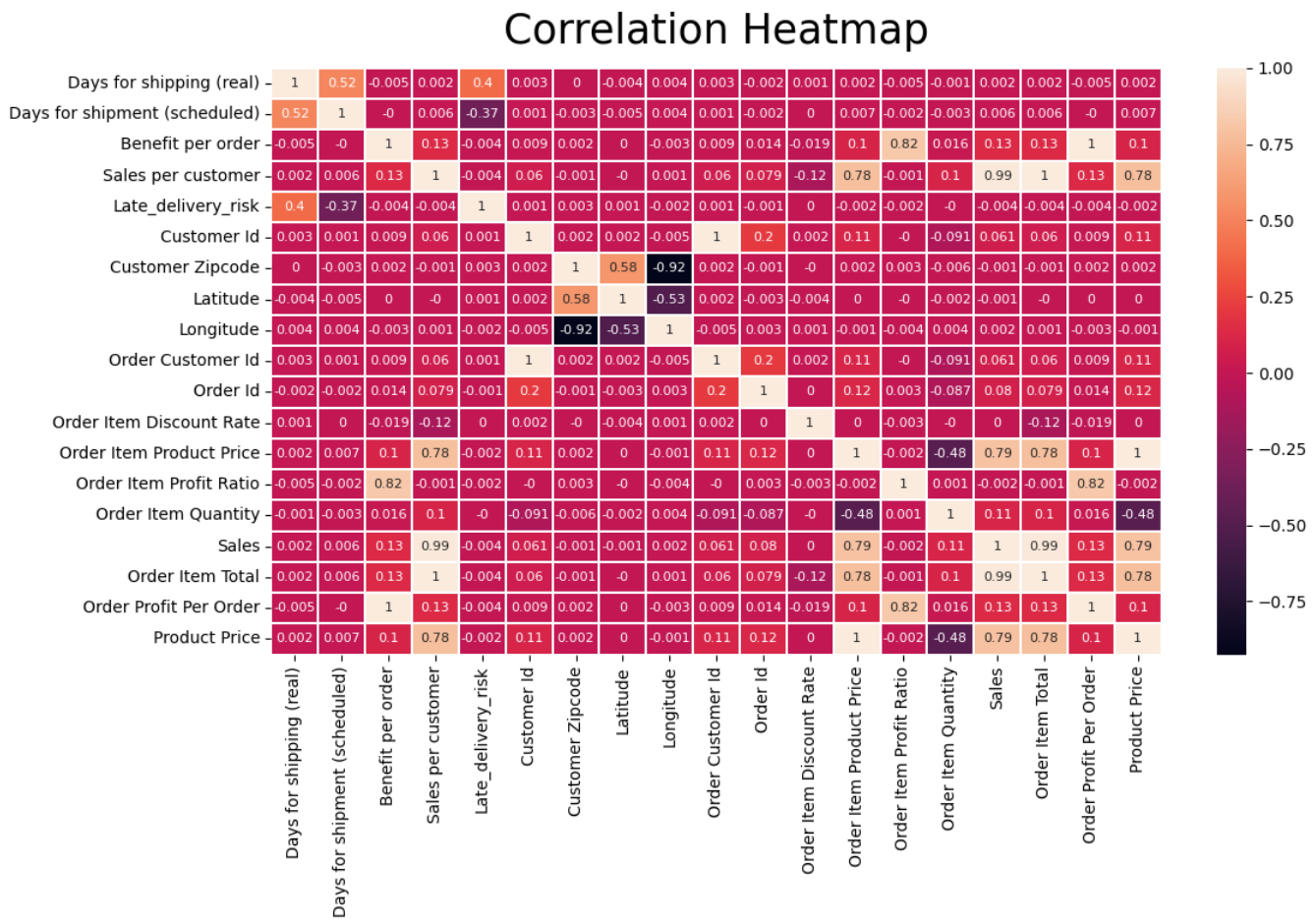
Việc loại bỏ các cột này giúp tập trung vào các đặc trưng có khả năng mang lại thông tin dự đoán giá trị hơn cho mô hình.

4.6. Phân tích tương quan để phát hiện đa cộng tuyến

Đa cộng tuyến xảy ra khi có sự tương quan cao giữa các biến độc lập, điều này có thể ảnh hưởng đến tính ổn định và khả năng diễn giải của mô hình. Chúng ta sử dụng ma trận tương quan để phát hiện các cặp biến có tương quan cao.

Đầu tiên, chúng ta tính toán ma trận tương quan giữa các biến số trong DataFrame và trực quan hóa nó bằng heatmap:

```
fig, ax = plt.subplots(figsize=(12, 8))
numeric_df = df.select_dtypes(include='number')
sns.heatmap(round(numeric_df.corr(), 3), annot=True, linewidths=0.1,
annot_kws={"size": 8}, ax=ax)
plt.title('Correlation Heatmap', size=25, pad=15)
plt.tight_layout()
# plt.savefig('Correlation Heatmap.png', dpi=200)
```



Nhận xét: Heatmap này hiển thị hệ số tương quan giữa các cặp biến số. Các ô có màu sắc đậm hơn (đỏ hoặc xanh đậm) cho thấy mức độ tương quan cao hơn (dương hoặc âm). Các giá trị số trong ô thể hiện hệ số tương quan cụ thể. Chúng ta sẽ tìm kiếm các cặp biến có hệ số tương quan tuyệt đối gần 1, đặc biệt là lớn hơn một ngưỡng nhất định (ví dụ: 0.85).

Phát hiện và xử lý đa cộng tuyến bằng ma trận tương quan:

Để tự động phát hiện và loại bỏ các biến có tương quan cao, chúng ta thực hiện các bước sau:

```
# Chọn các biến số để tính toán tương quan
numeric_df = df.select_dtypes(include='number')

# Tính toán ma trận tương quan
corr_matrix = numeric_df.corr()

# Giữ lại phần tam giác trên (upper triangle) của ma trận tương quan
upper_corr_mat = corr_matrix.where(np.triu(np.ones(corr_matrix.shape),
k=1).astype(bool))
```

```
# Chuyển đổi từ dạng ma trận sang dạng Series 1 chiều, bỏ giá trị NaN
unique_corr_pairs = upper_corr_mat.unstack().dropna()

# Sắp xếp các cặp tương quan theo giá trị giảm dần
unique_corr_pairs.sort_values(ascending=False).head(10)
```

```
...  Order Customer Id        Customer Id                 1.000000
     Product Price            Order Item Product Price    1.000000
     Order Item Total         Sales per customer          1.000000
     Order Profit Per Order   Benefit per order           1.000000
     Sales                    Sales per customer          0.989744
     Order Item Total         Sales                       0.989744
     Order Item Profit Ratio  Benefit per order           0.823689
     Order Profit Per Order   Order Item Profit Ratio     0.823689
     Sales                    Order Item Product Price    0.789948
     Product Price            Sales                       0.789948
     dtype: float64
```

Nhận xét: Bảng này hiển thị 10 cặp biến có hệ số tương quan cao nhất. Chúng ta thấy có những cặp biến có tương quan hoàn hảo (1.0) hoặc rất cao (gần 1.0), ví dụ như 'Order Customer Id' và 'Customer Id', 'Product Price' và 'Order Item Product Price', 'Order Item Total' và 'Sales per customer', 'Order Profit Per Order' và 'Benefit per order', 'Sales' và 'Sales per customer', 'Order Item Total' và 'Sales'.

- Loại bỏ các biến có tương quan cao > 0.85

Chúng ta sẽ loại bỏ một trong hai biến của mỗi cặp có hệ số tương quan tuyệt đối lớn hơn 0.85 để giảm thiểu đa cộng tuyến. Thông thường, chúng ta có thể giữ lại biến mà chúng ta cho là quan trọng hơn hoặc có ý nghĩa kinh tế hơn.

```
# Chuyển sang DataFrame để dễ xử lý
cortable = unique_corr_pairs.sort_values(ascending=False).reset_index()

# Xác định danh sách các cột cần loại bỏ do tương quan quá cao
drop_corr = cortable[cortable.iloc[:, 2] >
0.85]["level_1"].values.tolist()

drop_corr
```

```
...   ['Customer Id',
       'Order Item Product Price',
       'Sales per customer',
       'Benefit per order',
       'Sales per customer',
       'Sales']
```

Nhận xét: Dựa trên ngưỡng tương quan 0.85, danh sách các cột được xác định để loại bỏ là: ['Customer Id', 'Order Item Product Price', 'Sales per customer', 'Benefit per order', 'Sales per customer', 'Sales']. Lưu ý rằng 'Sales per customer' và 'Sales' xuất hiện hai lần do chúng có tương quan cao với nhiều biến khác.

```
# Xoá các cột có tương quan cao khỏi dataframe
df.drop(drop_corr, axis=1, inplace=True)
```

Kiểm tra lại kết quả sau khi loại bỏ

```
# Kích thước DataFrame sau khi xoá
print("Shape after dropping highly correlated columns:", df.shape)

# Danh sách các cột đã xoá
print("Dropped columns due to high correlation:", drop_corr)
```

```
Shape after dropping highly correlated columns: (180519, 31)
Dropped columns due to high correlation: ['Customer Id', 'Order Item Product Price', 'Sales per customer', 'Benefit per order', 'Sales per customer', 'Sales']
```

Nhận xét: Sau khi loại bỏ các cột có tương quan cao, kích thước của DataFrame đã thay đổi (số cột giảm). Danh sách các cột đã bị loại bỏ được in ra để chúng ta theo dõi.

4.7. Chuyển đổi và chuẩn hóa dữ liệu

Sau các bước làm sạch và loại bỏ dữ liệu, chúng ta tiến hành chuyển đổi kiểu dữ liệu của các cột cho phù hợp với mục đích phân tích và xây dựng mô hình.

Cập nhật kiểu dữ liệu cho biến phân loại

Để tối ưu hóa việc sử dụng bộ nhớ và chuẩn bị cho các bước mã hóa (encoding) sau này, chúng ta chuyển đổi kiểu dữ liệu của các cột chứa dữ liệu phân loại (categorical) sang kiểu 'category'. Kiểu dữ liệu 'category' hiệu quả hơn cho các biến có số lượng giá trị duy nhất hữu hạn và thường được sử dụng trong các thuật toán học máy.

```
col_types = {
    'Customer Lname':'category', 'Customer City':'category', 'Order
State':'category',
    'Type':'category', 'Delivery Status':'category', 'Category
```

```
Name':'category',
    'Customer Segment':'category', 'Department Name':'category',
'Market':'category',
    'Order Region':'category', 'Order Status':'category', 'Product
Name':'category',
    'Shipping Mode':'category', 'Order City':'category', 'Order
Country':'category'
}
df = df.astype(col_types)
```

Chuyển đổi kiểu dữ liệu ngày giờ

Các cột chứa thông tin về ngày đặt hàng ('order date (DateOrders)') và ngày giao hàng ('shipping date (DateOrders)') ban đầu có thể ở định dạng 'object' (chuỗi). Để có thể thực hiện các phép toán liên quan đến thời gian (ví dụ: tính khoảng thời gian giao hàng, trích xuất thông tin về ngày, tháng, năm), chúng ta cần chuyển đổi chúng sang kiểu dữ liệu 'datetime'.

```
# Chuyển đổi cột ngày mà không cần 'infer_datetime_format'
df["shipping date (DateOrders)"] = pd.to_datetime(df["shipping date
(DateOrders)"])
df["order date (DateOrders)"] = pd.to_datetime(df["order date
(DateOrders)"])
```

Cuối cùng, chúng ta kiểm tra lại kiểu dữ liệu của tất cả các cột trong DataFrame để đảm bảo các chuyển đổi đã được thực hiện thành công:

```
df.dtypes
```

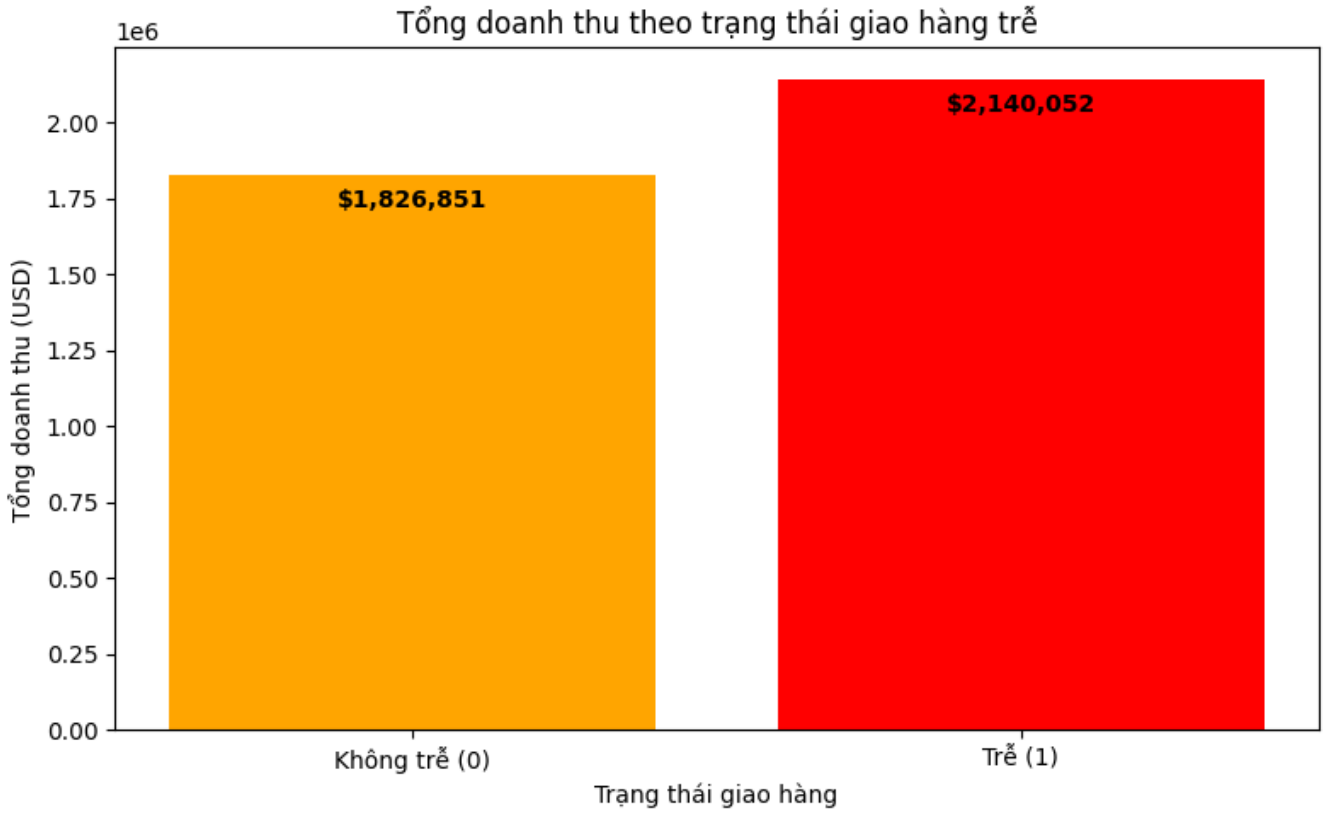
```
Type                                category
Days for shipping (real)               int64
Days for shipment (scheduled)          int64
Delivery Status                     category
Late_delivery_risk                     int64
Category Name                       category
Customer City                       category
Customer Lname                      category
Customer Segment                    category
Customer Zipcode                     float64
Department Name                     category
Latitude                             float64
Longitude                            float64
Market                              category
Order City                          category
Order Country                       category
Order Customer Id                      int64
order date (DateOrders)         datetime64[ns]
Order Id                               int64
Order Item Discount Rate             float64
Order Item Profit Ratio              float64
Order Item Quantity                    int64
Order Item Total                     float64
Order Profit Per Order               float64
Order Region                        category
Order State                         category
Order Status                        category
Product Name                        category
Product Price                        float64
shipping date (DateOrders)      datetime64[ns]
```

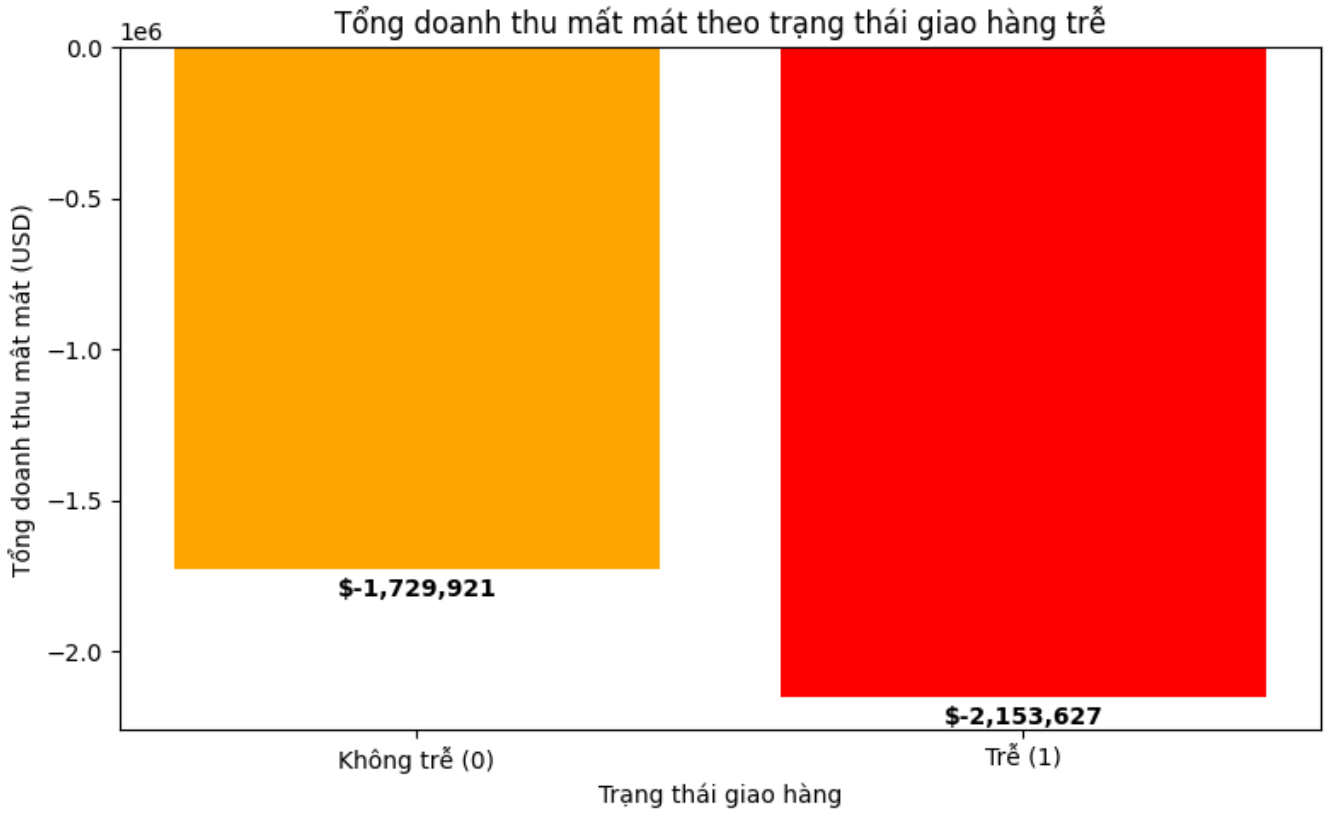
Nhận xét: Hình ảnh cho thấy kiểu dữ liệu của các cột đã được cập nhật. Các cột được chỉ định trong col_types đã chuyển sang kiểu 'category', và các cột 'order date (DateOrders)' và 'shipping date (DateOrders)' đã chuyển sang kiểu 'datetime64[ns]', cho thấy quá trình chuyển đổi kiểu dữ liệu đã thành công.

5. PHÂN TÍCH KHÁM PHÁ (EDA)

5.1. Phân tích doanh thu liên quan đến vấn đề giao hàng

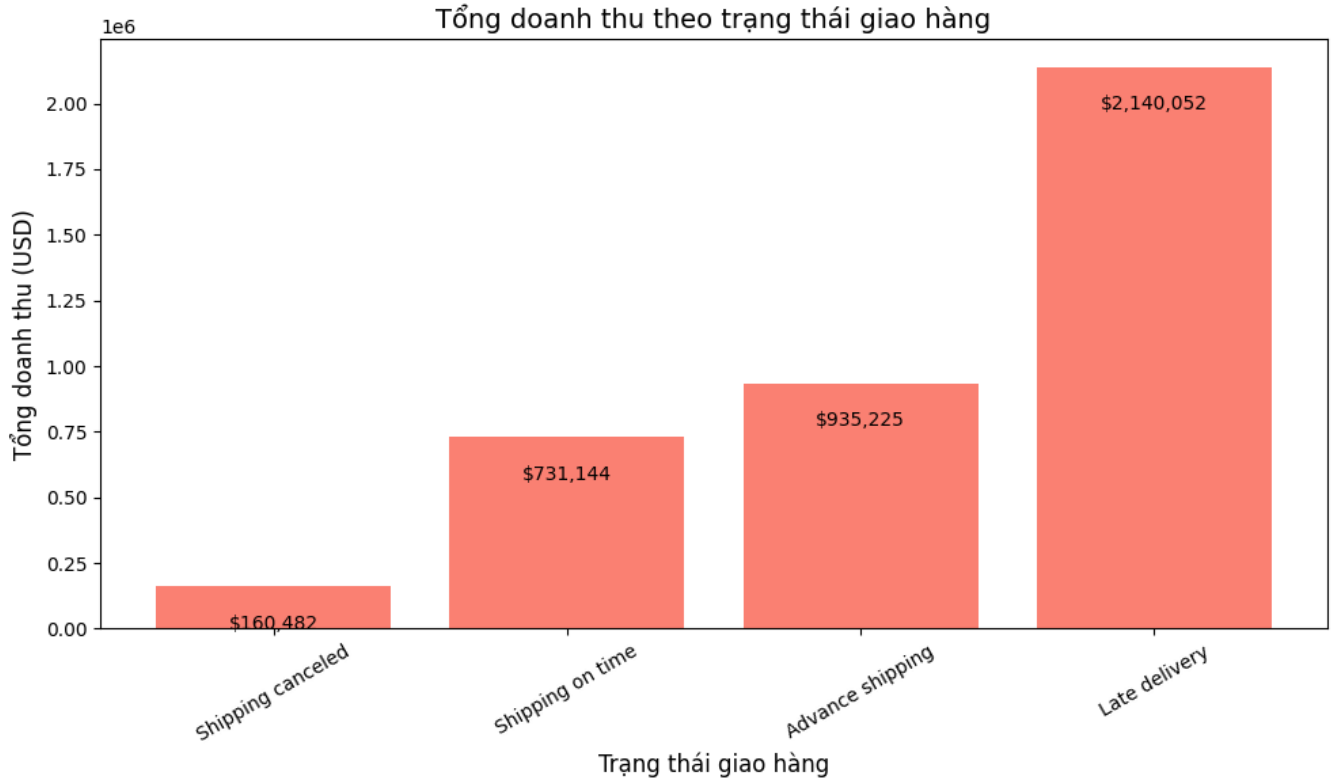
- So sánh doanh thu và doanh thu mất mát của đơn giao trễ và đúng hạn

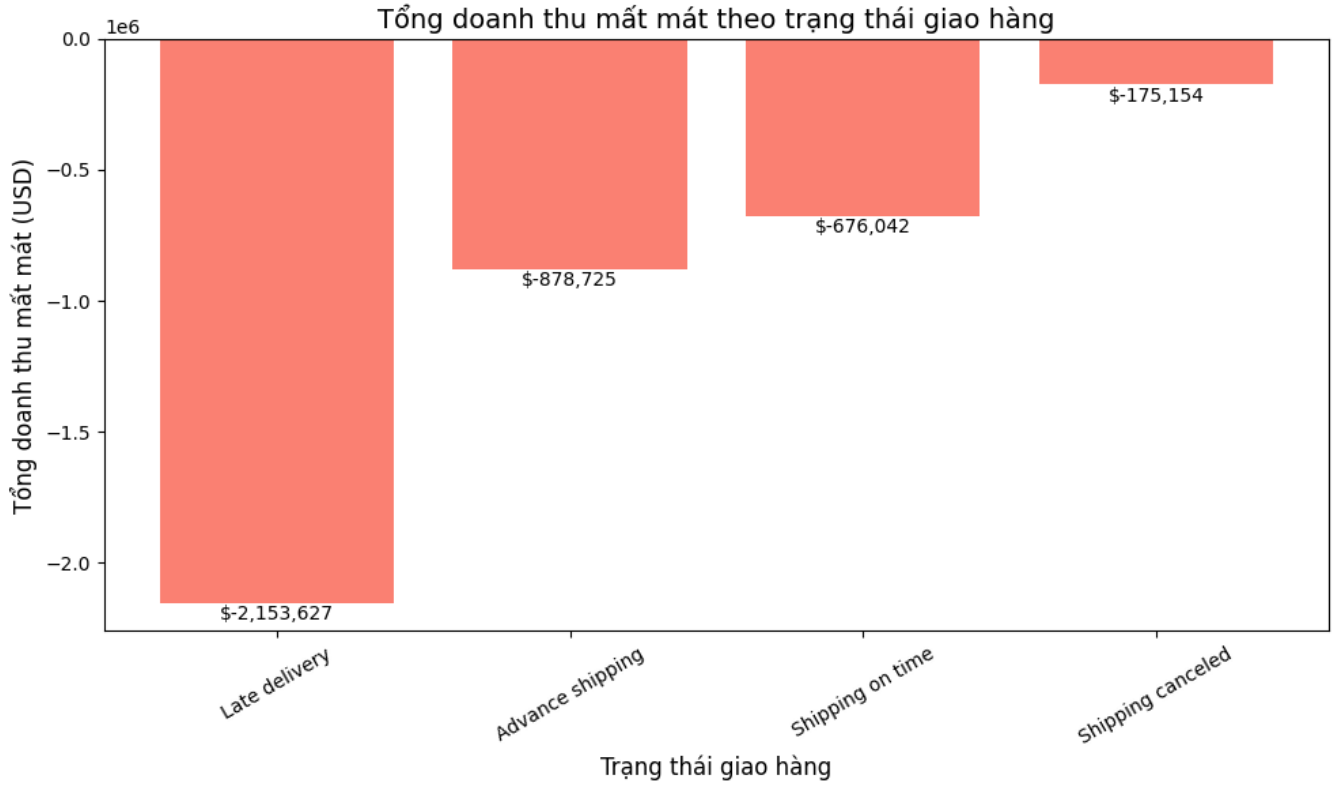




Hai biểu đồ cho thấy một nghịch lý thú vị: trong khi tổng doanh thu từ các đơn hàng trễ cao hơn, thì tổng doanh thu mất mát từ các đơn hàng trễ cũng cao hơn đáng kể.

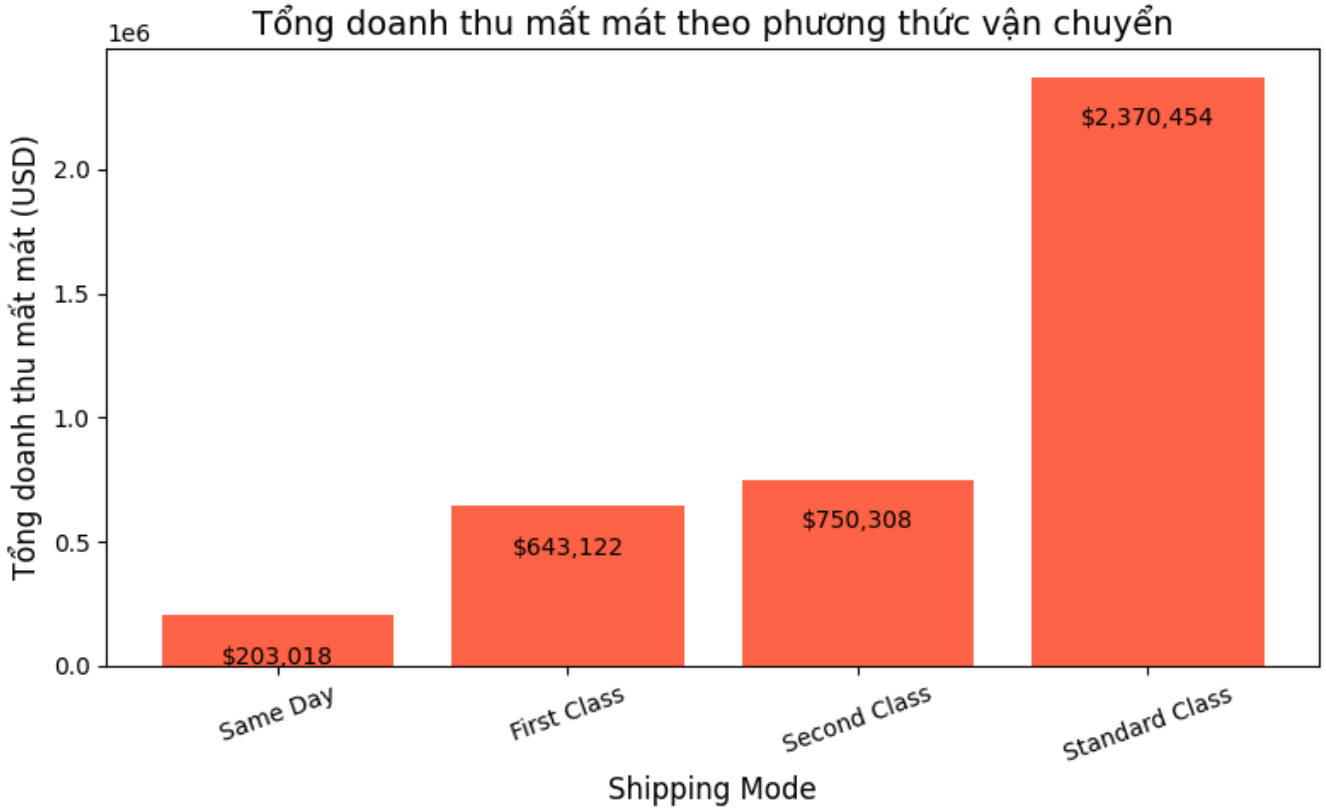
- So sánh doanh thu và doanh thu mất mát theo trạng thái đơn hàng

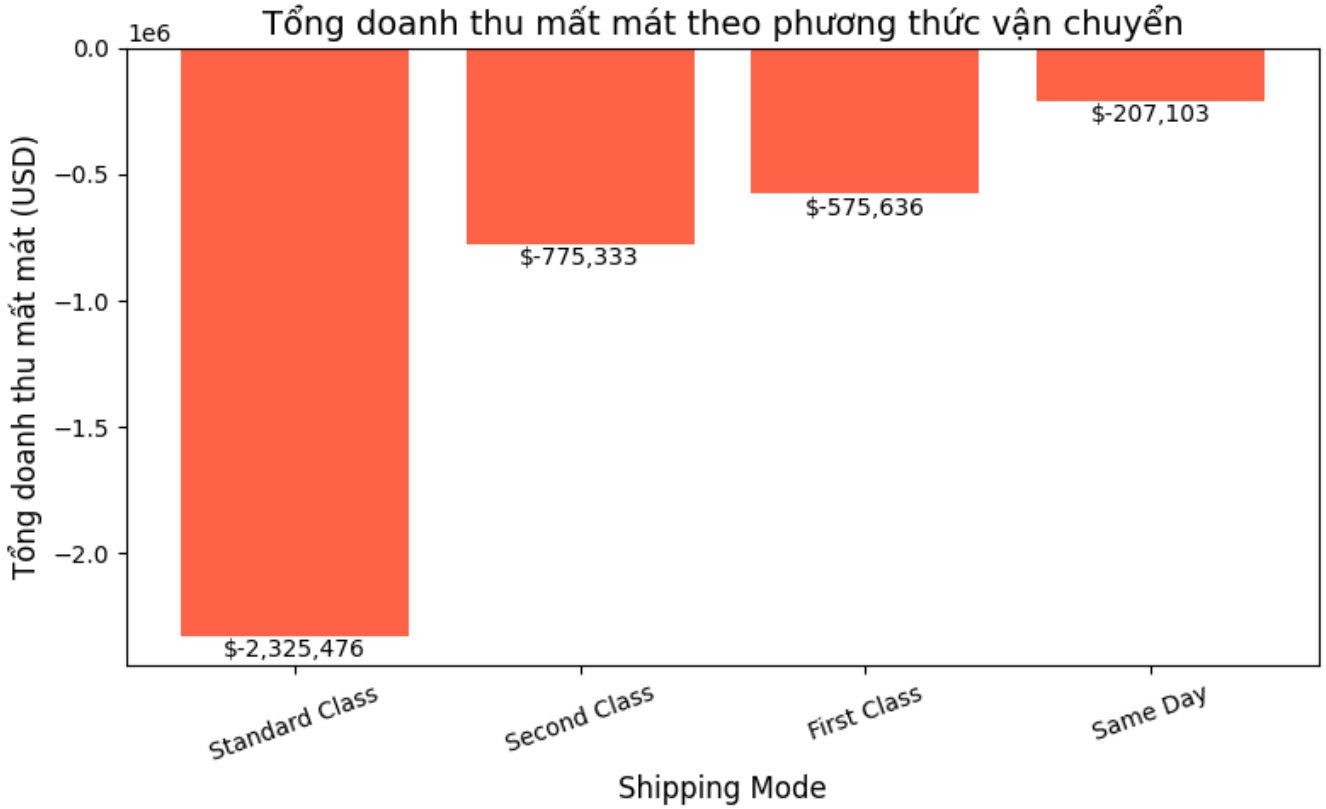




Tiếp theo doanh thu theo trạng thái đơn hàng sẽ làm rõ hơn trong giao đúng hạn sẽ gồm những thành phần: Advance shipping(giao hàng sớm) , Shipping on time (giao đúng ngày), Shipping canceled (bị hủy giao hàng).

- Doanh thu và doanh thu mất mát theo phương thức giao hàng

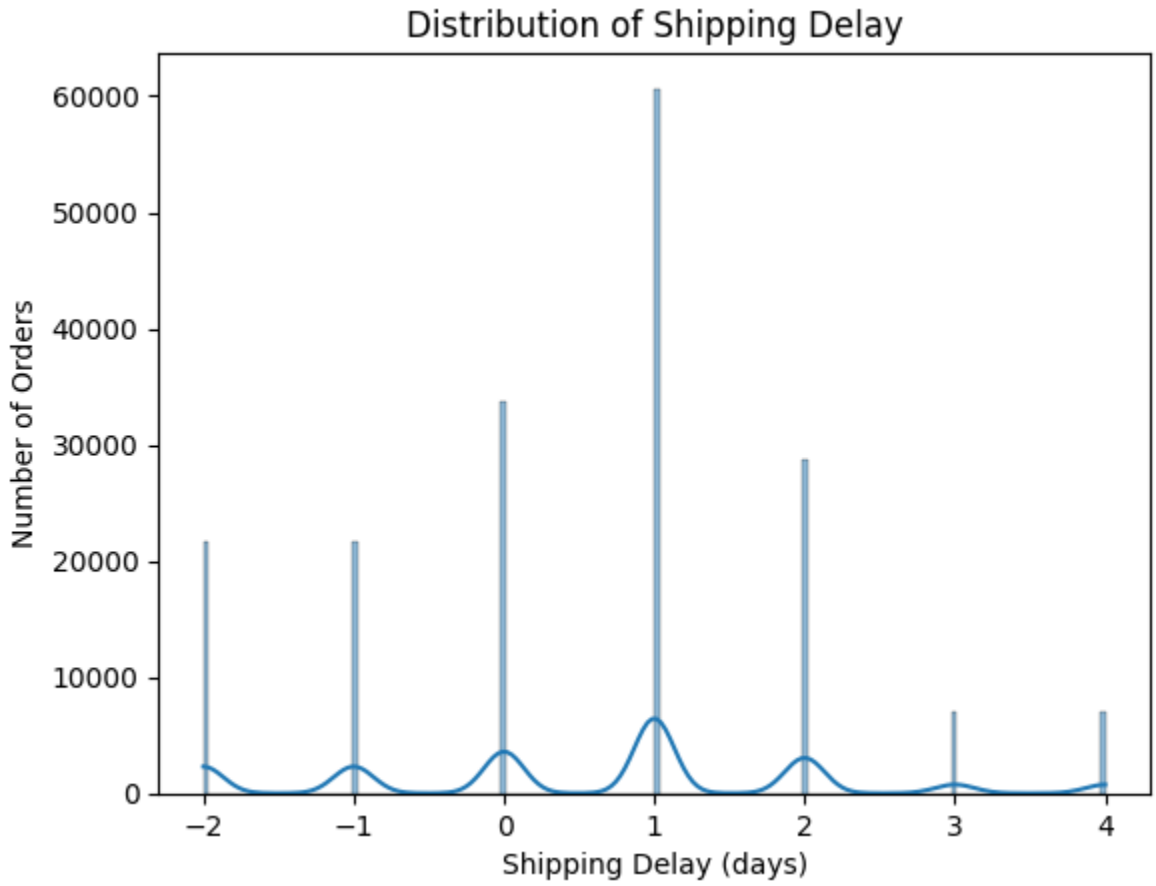




Hai biểu đồ trên biểu thị rõ ràng phương thức giao hàng Standard Class giao hàng thông thường có doanh thu cũng như mất mát doanh thu cao nhất có lẽ ở đây phương thức giao hàng này chưa được tối ưu tốt dẫn đến giao trễ gây hao hụt nhiều cho doanh thu

5.2. Phân tích vận hành/giao hàng

- Phân phối thời gian giao hàng thực tế so với lịch trình

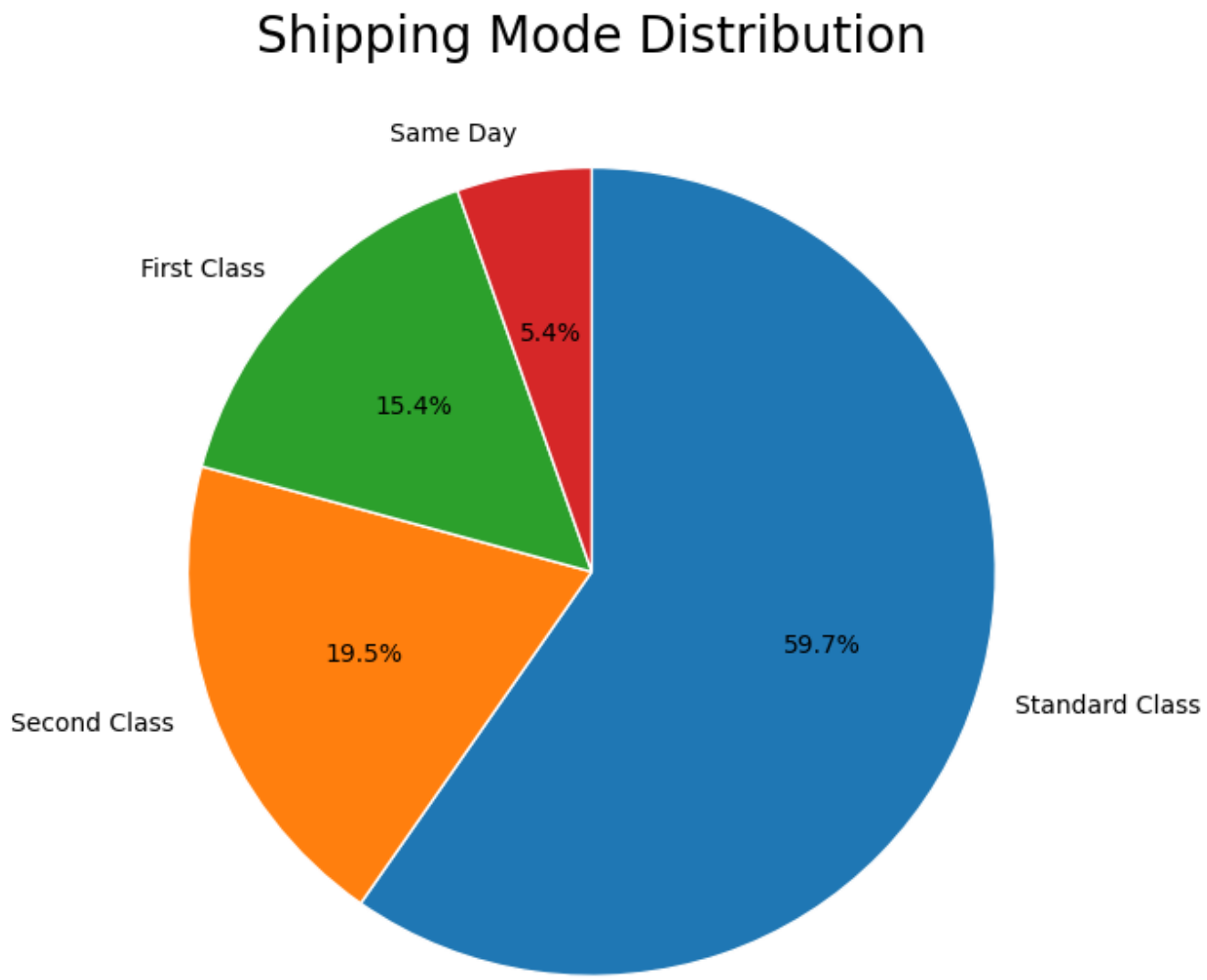


- Hệ thống giao hàng thường bị trễ 1 ngày là chủ yếu.

- Có một phần không nhỏ đơn hàng được giao sớm hoặc đúng hạn.

- Phân phối dữ liệu cho thấy đặc điểm vận hành không ổn định (có nhiều đỉnh khác nhau), có thể cần xem xét lại quy trình logistics để giảm độ trễ và biến động.

- Phân bố tỷ lệ các đơn hàng theo loại hình thức vận chuyển



Phương thức "Standard Class" chiếm tỷ trọng lớn nhất:

- Với **59.7%**, đây là phương thức giao hàng được sử dụng phổ biến nhất.
- Điều này có thể do chi phí thấp, phù hợp với đa số khách hàng hoặc chính sách mặc định của hệ thống.

"Second Class" và "First Class" chiếm tỷ trọng trung bình:

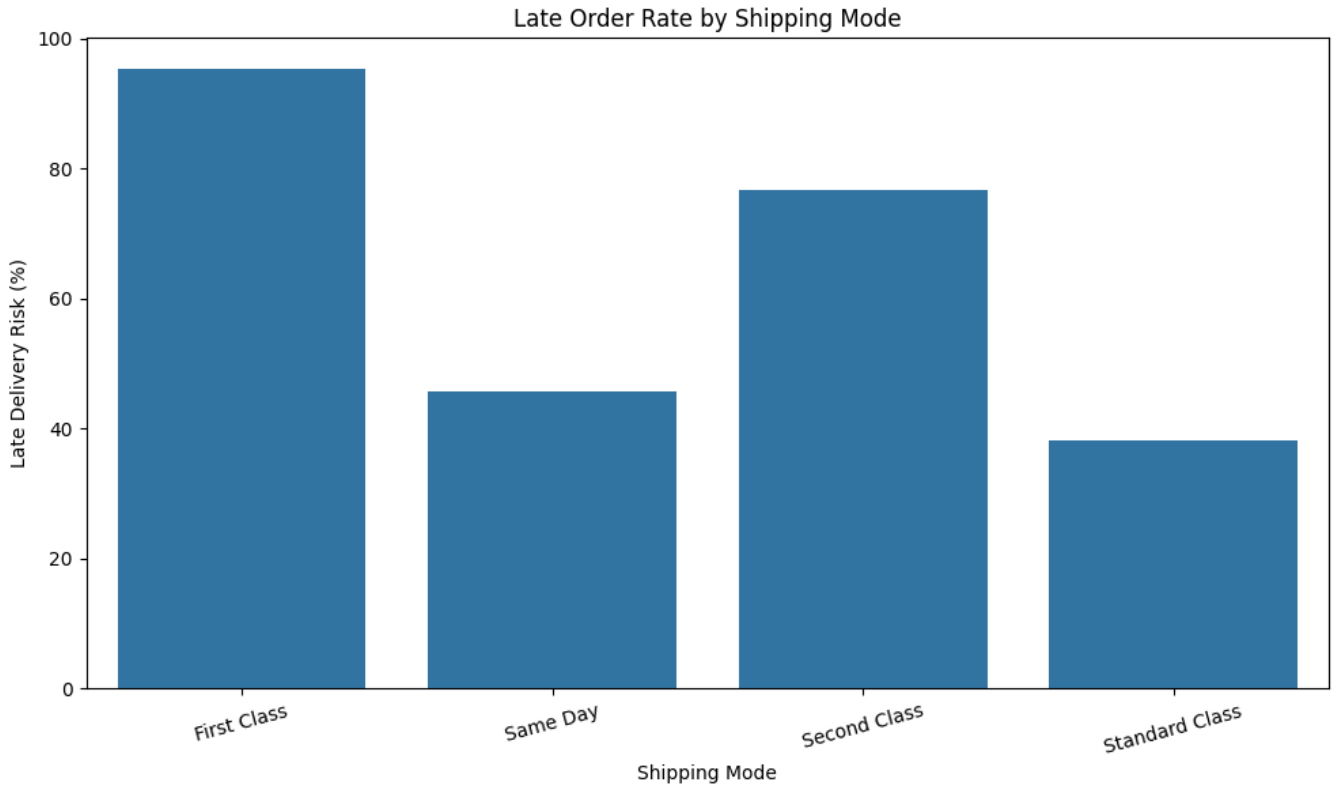
- "Second Class": **19.5%** – mức phổ biến thứ hai.
- "First Class": **15.4%** – ít phổ biến hơn, có thể do chi phí cao hơn.

"Same Day" là phương thức ít phổ biến nhất:

- Chỉ **5.4%**, điều này dễ hiểu vì đây là loại hình yêu cầu chi phí cao và điều kiện vận hành đặc biệt (ví dụ: khoảng cách gần, kho hàng tại chỗ...).

Có vẻ chiến lược giao hàng hiện tại thiên về tiết kiệm chi phí (ưu tiên Standard Class).

- Tỷ lệ đơn hàng giao trễ theo Shipping Mode



First Class có tỷ lệ giao trễ cao nhất (~95%)

- Điều này trái ngược với kỳ vọng rằng dịch vụ cao cấp sẽ giao đúng hạn.
- Có thể do kỳ vọng thời gian giao hàng quá ngắn nên dễ bị tính là "trễ".
- Cũng có thể là vấn đề vận hành với các đơn hàng ưu tiên.

Same Day có tỷ lệ giao trễ tương đối thấp (~46%)

- Điều này có thể gây bất ngờ vì đây là dịch vụ gấp.

- Tuy nhiên, do số lượng đơn hàng sử dụng phương thức này thấp (như đã thấy ở biểu đồ tròn trước đó), có thể hệ thống chỉ nhận giao trong điều kiện đảm bảo giao đúng — dẫn đến tỷ lệ trễ thấp hơn.

Second Class có tỷ lệ trễ cao (~77%)

- Là mức trung bình nhưng vẫn cao, có thể là do đây là loại dịch vụ không được ưu tiên trong xử lý hoặc vận chuyển.

Standard Class có tỷ lệ trễ thấp nhất (~38%)

- Là dịch vụ được sử dụng nhiều nhất (chiếm gần 60% trong biểu đồ tròn), nhưng lại có tỷ lệ trễ thấp nhất.
- Điều này cho thấy độ tin cậy của phương thức này trong vận hành hiện tại.

First Class không đáng tin cậy như kỳ vọng, dù là dịch vụ "cao cấp".

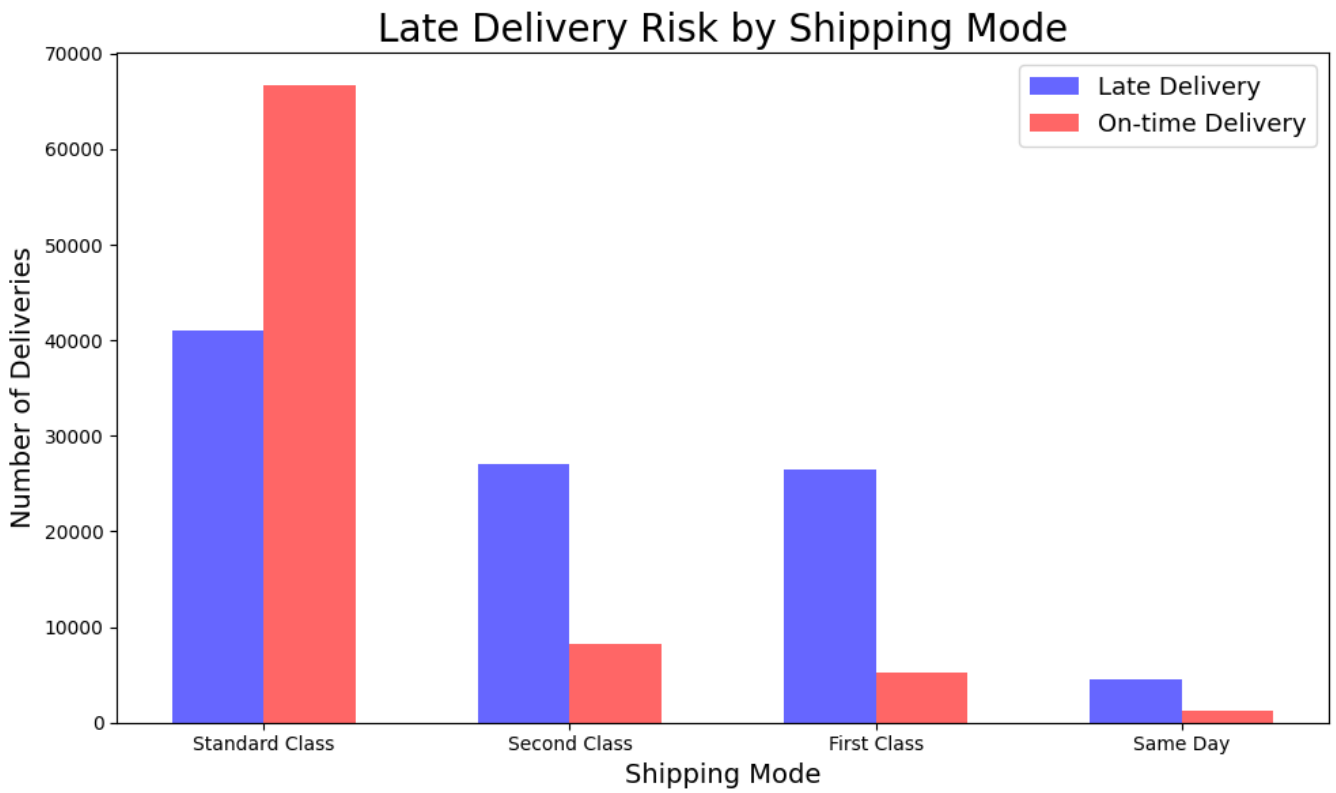
Standard Class có hiệu suất tốt nhất, vừa phổ biến vừa ít trễ.

Các doanh nghiệp có thể cần:

- Đánh giá lại kỳ vọng thời gian giao của First Class, hoặc
- Cải thiện quy trình vận hành, đặc biệt với các dịch vụ ưu tiên.

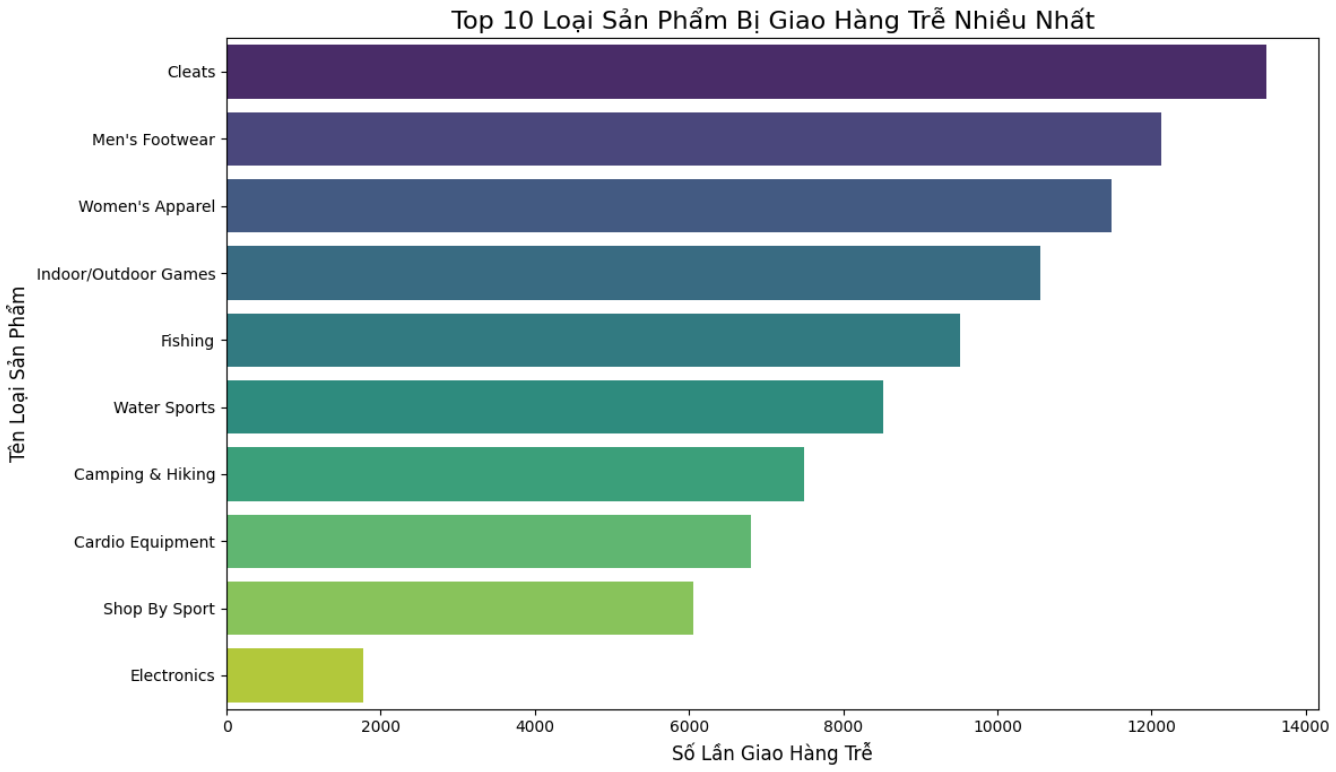
Nếu có thể, khách hàng nên được khuyến nghị sử dụng Standard Class nếu muốn đảm bảo giao đúng hạn.

- Số lượng đơn hàng giao trễ và đúng hạn theo Shipping Mode



Biểu đồ này giúp làm rõ số lượng cho phần biểu đồ phía trên

- Top loại sản phẩm giao trễ nhiều nhất



	Category Name	Top Late Shipping Date	Late Orders on That Date
0	Cleats	2015-11-24 03:13:00	5
1	Men's Footwear	2017-03-12 17:06:00	4
2	Women's Apparel	2017-03-13 03:26:00	6
3	Indoor/Outdoor Games	2017-04-07 08:07:00	5
4	Fishing	2016-11-10 19:43:00	4
5	Water Sports	2015-12-08 14:04:00	4
6	Camping & Hiking	2015-04-15 05:38:00	5
7	Cardio Equipment	2016-04-02 02:25:00	4
8	Shop By Sport	2015-01-07 15:24:00	4
9	Electronics	2017-04-09 10:45:00	3

Lý do vì sao các loại sản phẩm này bị trễ nhiều

1. Cleats (Giày đinh thể thao)

Ngày trễ cao nhất: 2015-11-24

Lý do khả thi: Đây là sản phẩm thể thao mùa vụ, nhu cầu có thể tăng đột biến trong một số sự kiện hoặc mùa giải thể thao, gây áp lực cho kho và logistics.

2. Men's Footwear & Women's Apparel (Giày nam và thời trang nữ)

Ngày trễ cao nhất: khoảng đầu năm 2017 (tháng 3)

Lý do: Sản phẩm thời trang thường có biến động mùa vụ và các chương trình khuyến mãi lớn (ví dụ, xuân/hè hoặc giảm giá đầu năm), dẫn đến đơn hàng tăng đột biến, khiến việc giao hàng chậm trễ.

3. Indoor/Outdoor Games (Đồ chơi trong nhà/ngoài trời)

Ngày trễ cao nhất: 2017-04-07

Lý do: Đây là mặt hàng có kích thước cồng kềnh và đa dạng, đôi khi phụ thuộc vào nhập khẩu hoặc sản xuất theo mùa, dẫn đến tồn kho không ổn định và chậm trễ vận chuyển.

4. Fishing, Water Sports, Camping & Hiking (Đồ câu, thể thao nước, cắm trại & leo núi)

Ngày trễ cao nhất: trải dài từ 2015 đến 2016

Lý do: Các sản phẩm này thường có kích thước lớn, yêu cầu đóng gói và vận chuyển cẩn thận, dễ bị ảnh hưởng bởi điều kiện vận chuyển hoặc phụ thuộc vào các đối tác logistics chuyên biệt.

5. Cardio Equipment (Thiết bị tập thể dục)

Ngày trễ cao nhất: 2016-04-02

Lý do: Thiết bị thể dục cồng kềnh, nặng, đòi hỏi phương tiện vận chuyển chuyên dụng; thường xuyên bị trễ do hạn chế phương tiện hoặc thiếu hụt nhân lực bốc xếp.

6. Shop By Sport (Mua sắm theo thể thao)

Ngày trễ cao nhất: 2015-01-07

Lý do: Có thể đây là nhóm sản phẩm đa dạng, chứa nhiều loại hàng hoá khác nhau, nên dễ bị ảnh hưởng chung bởi các vấn đề về kho bãi và vận chuyển.

7. Electronics (Thiết bị điện tử)

Ngày trễ cao nhất: 2017-04-09

Lý do: Thiết bị điện tử có thể gặp trục trặc về tồn kho hoặc hải quan (nếu nhập khẩu), cùng với sự gia tăng đơn hàng theo mùa lễ, dễ gây ra tắc nghẽn trong chuỗi cung ứng.

- Những nơi tương ứng với nơi kho vận chuyển có tỷ lệ trễ cao nhất

	Order Country	Lat_round	Long_round	count	Late Rate (%)
3694868	Estados Unidos	35.8	-77.5	32	87.500000
4077034	Francia	36.1	-115.2	45	86.666667
3698880	Estados Unidos	37.0	-122.0	35	85.714286
7828152	México	40.0	-75.2	35	85.714286
3204624	El Salvador	18.4	-66.1	41	85.365854
4067631	Francia	33.7	-118.0	34	85.294118
3719836	Estados Unidos	42.3	-83.4	36	83.333333
3666692	Estados Unidos	26.2	-80.2	59	83.050847
5319377	India	41.9	-87.7	34	82.352941
207184	Alemania	40.9	-73.9	48	81.250000

Các khu vực có tỷ lệ đơn hàng giao trễ rất cao đều nằm rải rác trên nhiều quốc gia và vùng địa lý khác nhau.

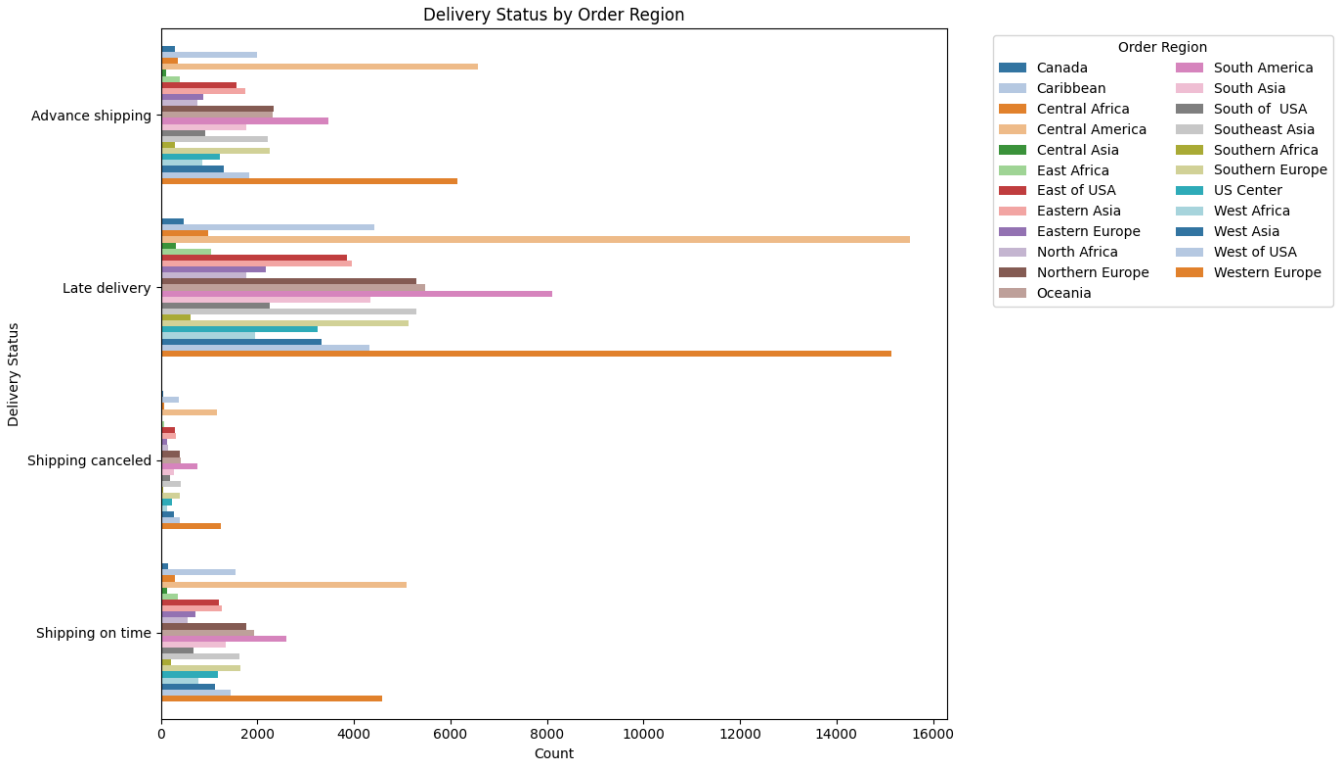
- Điều này có thể do các yếu tố như:

- Hạn chế về hạ tầng vận chuyển, đặc biệt trong các khu vực có địa hình khó hoặc vùng ngoại ô.

- Quản lý tồn kho và phân phối chưa hiệu quả, dẫn đến delay nhiều.

- Các yếu tố đặc thù vùng miền như điều kiện thời tiết, thủ tục hải quan phức tạp,...

- Trạng thái đơn hàng phân bố theo vùng



Việc có nhiều khu vực với số lượng đơn hàng trễ lớn cho thấy vấn đề trễ giao hàng là một thách thức trên phạm vi rộng, không chỉ giới hạn ở một vài địa điểm cụ thể.

Western Europe có số lượng đơn hàng giao trễ cao nhất, vượt trội hơn các khu vực khác. Đây có thể là điểm nghẽn nghiêm trọng trong chuỗi cung ứng.

Central America và South America cũng ghi nhận số lượng đơn trễ cao, cho thấy vấn đề giao hàng không chỉ tập trung ở một khu vực.

Dù có nhiều đơn hàng được giao sớm (Advance shipping), nhưng các khu vực như Western Europe và Central America vẫn có số đơn trễ cao. Điều này cho thấy việc giao sớm chưa đủ để đảm bảo giao đúng hạn.

Các khu vực như Canada, East of USA, Northern Europe có tỷ lệ đơn giao đúng hạn cao hơn và ít đơn bị trễ, cho thấy hệ thống giao hàng ở đây ổn định hơn.

Tỷ lệ đơn hàng bị hủy rất thấp ở tất cả khu vực, cho thấy phần lớn đơn hàng vẫn được xử lý, dù có thể bị trễ.

5.3. Tóm tắt phát hiện quan trọng

a. Phương thức giao hàng Standard Class dù có tỷ lệ giao trễ thấp nhất nhưng chiếm tỷ trọng lớn nhất trong tổng số đơn hàng, đóng vai trò then chốt trong vận hành hiện tại. Đây là phương thức vận chuyển có hiệu quả tương đối ổn định và đáng tin cậy.

b. First Class – dịch vụ cao cấp lại có tỷ lệ giao trễ cao bất ngờ, không đáp ứng được kỳ vọng về tốc độ và chất lượng giao hàng. Điều này có thể xuất phát từ kỳ vọng thời gian quá ngắn hoặc quy trình vận hành chưa hiệu quả.

c. Tình trạng giao hàng trễ phổ biến trên toàn cầu, phân bố rộng rãi ở nhiều quốc gia và vùng địa lý, đặc biệt là tại Western Europe và khu vực Mỹ Latin. Vấn đề này không chỉ là yếu tố địa phương mà là thách thức tổng thể cho chuỗi cung ứng.

d. Các loại sản phẩm có đặc điểm riêng như kích thước lớn, mùa vụ hoặc phụ thuộc vào logistics đặc thù thường gặp rủi ro giao hàng trễ cao, làm ảnh hưởng đến doanh thu và uy tín dịch vụ.

e. Khuyến nghị

- Ưu tiên tối ưu và duy trì chất lượng của phương thức Standard Class.

- Cần xem xét điều chỉnh kỳ vọng và cải thiện quy trình vận hành cho dịch vụ First Class.

- Đánh giá và nâng cao năng lực vận chuyển, kho bãi đặc biệt ở các khu vực có tỷ lệ giao trễ cao.

- Tập trung quản lý tồn kho và logistics cho các nhóm sản phẩm dễ bị trễ để giảm thiểu rủi ro.

- Phân tích thêm để cải thiện dự báo nhu cầu, giảm biến động đột ngột trong các mùa vụ.

6. BIẾN ĐỔI ĐẶC TRƯNG

Giai đoạn biến đổi đặc trưng đóng vai trò quan trọng trong việc chuyển đổi dữ liệu thô sang định dạng phù hợp hơn cho các thuật toán học máy. Các bước chính trong giai đoạn này bao gồm mã hóa các biến ngày tháng, khám phá đặc trưng số, mã hóa các biến phân loại và loại bỏ các thuộc tính có tương quan cao.

6.1. Mã hóa biến ngày tháng

Để mô hình có thể nhận biết và học được các xu hướng theo thời gian, chúng ta tiến hành tách các biến ngày tháng ('shipping date (DateOrders)' và 'order date (DateOrders)') thành các thành phần cơ bản như năm, tháng, ngày và giờ. Sau khi trích xuất các thành phần này, các cột ngày tháng ban đầu sẽ bị loại bỏ.

```
# Shipping date
df["ship_year"]= df["shipping date (DateOrders)"].dt.year
df["ship_month"]= df["shipping date (DateOrders)"].dt.month
df["ship_day"]= df["shipping date (DateOrders)"].dt.day
df["ship_hour"]= df["shipping date (DateOrders)"].dt.hour
df.drop(["shipping date (DateOrders)"], axis = 1, inplace= True)

# Order date
df["order_year"]= df["order date (DateOrders)"].dt.year
df["order_month"]= df["order date (DateOrders)"].dt.month
df["order_day"]= df["order date (DateOrders)"].dt.day
df["order_hour"]= df["order date (DateOrders)"].dt.hour
df.drop(["order date (DateOrders)"], axis = 1, inplace= True)
```

6.2. Khám phá đặc trưng số (histogram, boxplot)

Trước khi tiến hành các bước biến đổi tiếp theo, chúng ta trực quan hóa các đặc trưng dạng số để hiểu rõ hơn về phân phối của chúng và phát hiện các giá trị ngoại lai (outlier).

- Biểu đồ Histogram:

Biểu đồ histogram hiển thị tần suất phân phối của từng giá trị trong mỗi thuộc tính số. Hình dạng của histogram có thể cho thấy liệu dữ liệu có phân phối chuẩn, lệch trái, lệch phải hay có nhiều đỉnh.

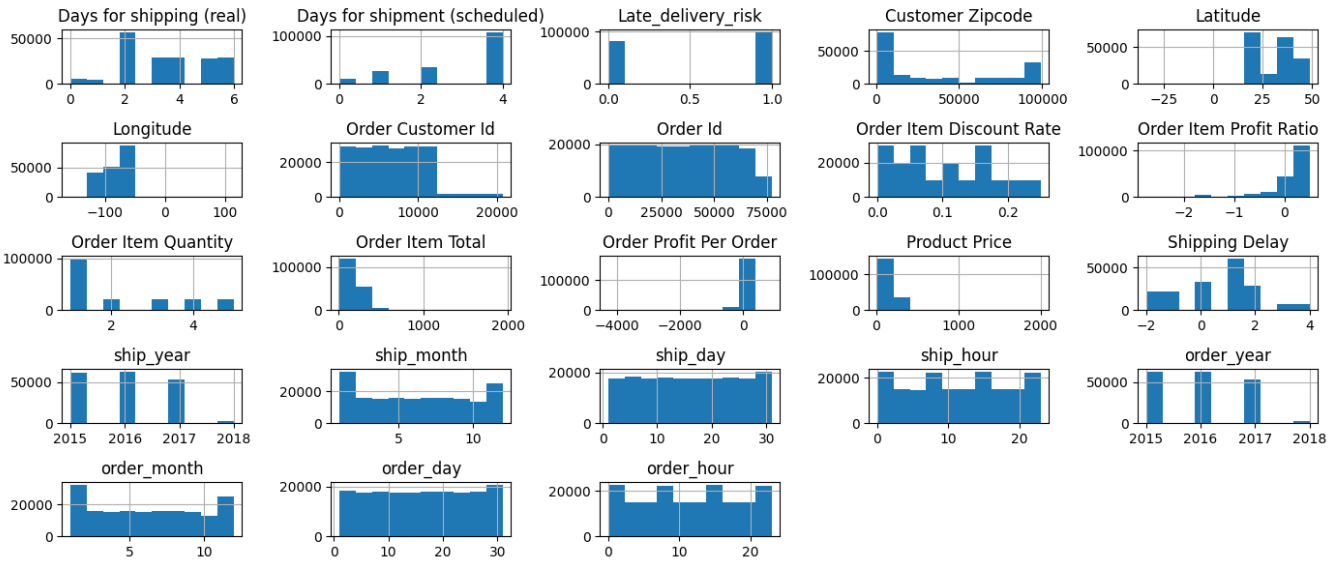
- Biểu đồ Boxplot:

Biểu đồ boxplot hiển thị các квартиль (quartiles), giá trị trung vị (median) và các giá trị ngoại lai (outliers) tiềm năng trong mỗi thuộc tính số. Các điểm nằm ngoài "râu" (whiskers) của boxplot thường được coi là outliers.

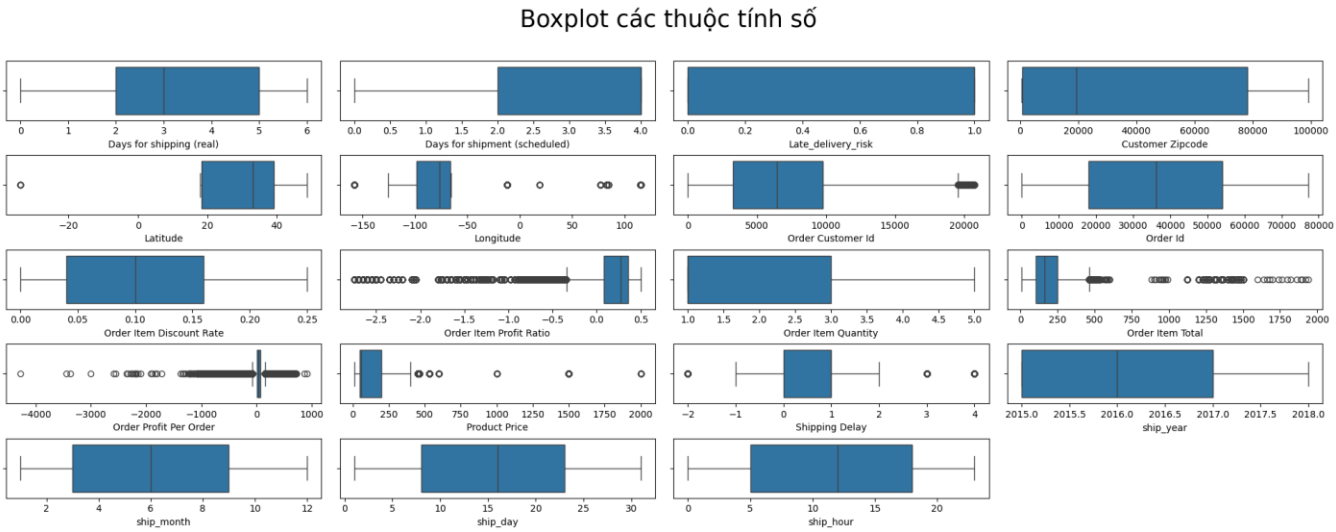
```
# Histogram
df.hist(figsize=(14,6))
```

```
plt.tight_layout()
plt.suptitle('Biểu đồ phân phối các thuộc tính số', size= 20, y= 1.1)
plt.show()

# Boxplot
fig, axes = plt.subplots(5,4,figsize=(20,8))
fig.delaxes(axes[4][3])  # Xóa ô trống
numdf= df.select_dtypes(exclude="category")
for attr, ax in zip(numdf.columns, axes.flatten()):
    sns.boxplot(x=attr, data=numdf, ax=ax)

plt.tight_layout()
fig.suptitle('Boxplot các thuộc tính số', size= 25)
fig.subplots_adjust(top=0.88)
plt.show()
```

Biểu đồ phân phối các thuộc tính số



Nhận xét: Các biểu đồ histogram cho thấy sự phân phối khác nhau giữa các thuộc tính số. Ví dụ, 'Days for shipping (real)' và 'Days for shipment (scheduled)' có vẻ phân phối rời rạc, trong khi 'Customer Zipcode' có phân phối trải rộng. 'Late_delivery_risk' là biến nhị phân (0 và 1). Các thuộc tính liên quan đến thời gian ('ship_year', 'ship_month', 'ship_day', 'ship_hour', 'order_year', 'order_month', 'order_day', 'order_hour') cho thấy sự phân phối theo thời gian trong dữ liệu.



Nhận xét: Các biểu đồ boxplot cho thấy sự hiện diện của các giá trị ngoại lai ở nhiều thuộc tính số, ví dụ như 'Customer Zipcode', 'Latitude', 'Longitude', 'Order Customer Id', 'Order Id', 'Order Item Discount Rate', 'Order Item Profit Ratio', 'Order Item Total', 'Order Profit Per Order', và 'Product Price'. Việc xử lý các giá trị ngoại lai này (nếu cần thiết) sẽ được xem xét trong các bước tiền xử lý hoặc biến đổi đặc trưng tiếp theo.

- Danh sách các thuộc tính phân loại:

```
# Categorical Attributes
df.select_dtypes(include="category").columns
```

```
Index(['Type', 'Delivery Status', 'Category Name', 'Customer City',
       'Customer Lname', 'Customer Segment', 'Department Name', 'Market',
       'Order City', 'Order Country', 'Order Region', 'Order State',
       'Order Status', 'Product Name', 'Shipping Mode'],
      dtype='object')
```

Nhận xét: Hình ảnh liệt kê các cột trong DataFrame có kiểu dữ liệu 'category'. Đây là các thuộc tính mà chúng ta sẽ cần mã hóa ở bước tiếp theo.

- Kiểu dữ liệu của các thuộc tính số:

```
## Numerical Attributes
df.select_dtypes(exclude="category").dtypes
```

```
Days for shipping (real)          int64
Days for shipment (scheduled)     int64
Late_delivery_risk                int64
Customer Zipcode                float64
Latitude                        float64
Longitude                       float64
Order Customer Id                 int64
Order Id                          int64
Order Item Discount Rate        float64
Order Item Profit Ratio         float64
Order Item Quantity               int64
Order Item Total                float64
Order Profit Per Order          float64
Product Price                   float64
Shipping Delay                    int64
ship_year                         int32
ship_month                        int32
ship_day                          int32
ship_hour                         int32
order_year                        int32
order_month                       int32
order_day                         int32
order_hour                        int32
dtype: object
```

Nhận xét: Hình ảnh hiển thị kiểu dữ liệu của các cột không phải là 'category'. Hầu hết là kiểu số nguyên ('int64', 'int32') hoặc số thực ('float64'), phù hợp cho việc phân tích và xây dựng mô hình.

6.3. Mã hóa biến phân loại

Các thuật toán học máy thường yêu cầu dữ liệu đầu vào là dạng số. Do đó, chúng ta cần mã hóa các thuộc tính phân loại sang dạng số. Tùy thuộc vào số lượng giá trị duy nhất trong mỗi thuộc tính phân loại, chúng ta sẽ áp dụng các phương pháp mã hóa khác nhau.

Đầu tiên, chúng ta xem xét số lượng giá trị duy nhất trong từng thuộc tính phân loại:

```
unique_categorical =
df.select_dtypes(include="category").nunique().sort_values()
unique_categorical
```

```
Customer Segment        3
Type                    4
Delivery Status         4
Shipping Mode           4
Market                  5
Order Status            9
Department Name        11
Order Region           23
Category Name          50
Product Name          118
Order Country         164
Customer City         563
Order State          1089
Customer Lname       1110
Order City           3597
dtype: int64
```

Nhận xét: Hình ảnh cho thấy số lượng giá trị duy nhất trong mỗi cột phân loại. Chúng ta sẽ sử dụng thông tin này để quyết định phương pháp mã hóa phù hợp cho từng cột. Các cột có ít giá trị duy nhất sẽ được mã hóa bằng One-Hot Encoding, trong khi các cột có nhiều giá trị duy nhất sẽ được mã hóa bằng Label Encoding.

- One-Hot Encoding:

Phương pháp này được sử dụng cho các biến có ít hơn 8 giá trị duy nhất (dựa trên kinh nghiệm và có thể điều chỉnh). One-hot encoding tạo ra các cột nhị phân (True/False hoặc 1/0) cho mỗi giá trị duy nhất trong thuộc tính ban đầu.

```
dummydf =
pd.get_dummies(df[list(unique_categorical.to_dict().keys())[:5]])
dummydf.head(3)
```

	Customer Segment_Consumer	Customer Segment_Corporate	Customer Segment_Home Office	Type_CASH	Type_DEBIT	Type_PAYMENT	Type_TRANSFER	Delivery Status_Advance shipping	Delivery Status_Late delivery	Delivery Status_Shipping canceled	Delivery Status_Shipping on time	Shipping Mode_First Class
0	True	False	False	False	True	False	False	True	False	False	False	False
1	True	False	False	False	False	False	True	False	True	False	False	False
2	True	False	False	True	False	False	False	False	False	False	True	False

Delivery Status_Advance shipping	Delivery Status_Late delivery	Delivery Status_Shipping canceled	Delivery Status_Shipping on time	Shipping Mode_First Class	Shipping Mode_Same Day	Shipping Mode_Second Class	Shipping Mode_Standard Class	Market_Africa	Market_Europe	Market_LATAM	Market_Pacific Asia	Market_USCA
True	False	False	False	False	False	False	True	False	False	False	True	False
False	True	False	False	False	False	False	True	False	False	False	True	False
False	False	False	True	False	False	False	True	False	False	False	True	False

```
dummydf.shape
```

```
···    (180519, 20)
```

```
df1= pd.concat([numdf, dummydf], axis =1)
```

Nhận xét: Hình ảnh cho thấy kết quả của One-Hot Encoding trên 5 thuộc tính phân loại đầu tiên (theo số lượng giá trị duy nhất tăng dần). Mỗi giá trị duy nhất trong các thuộc tính này đã được chuyển đổi thành một cột riêng biệt với giá trị True hoặc False (hoặc 1 hoặc 0) cho biết sự xuất hiện của giá trị đó trong từng hàng. Hình ảnh thứ tư cho thấy kích thước của DataFrame dummydf sau khi áp dụng One-Hot Encoding.

- Label Encoding:

Phương pháp này được áp dụng cho các thuộc tính có nhiều hơn 8 giá trị duy nhất. Label encoding gán một số thứ tự cho mỗi giá trị duy nhất trong thuộc tính.

```
le=LabelEncoder()
def Labelencoder_feature(x):
    le=LabelEncoder()
    x=le.fit_transform(x.astype(str))
    return x
```

```
abel= list(unique_categorical.to_dict().keys())[5:]
labeldf=df[label].apply(Labelencoder_feature)
```

```
labeldf.head(3)
```

	Order Status	Department Name	Order Region	Category Name	Product Name	Order Country	Customer City	Order State	Customer Lname	Order City
0	2	4	15	40	78	70	66	475	483	331
1	5	4	13	40	78	69	66	841	616	391
2	1	4	13	40	78	69	452	841	627	391

Nhận xét: Hình ảnh cho thấy kết quả của Label Encoding trên các thuộc tính phân loại có nhiều hơn 8 giá trị duy nhất. Mỗi giá trị duy nhất trong các thuộc tính này đã được thay thế bằng một số nguyên tương ứng.

```
df2= pd.concat([df1, labeldf], axis =1)
```

Sau khi mã hóa, DataFrame df2 sẽ chứa các đặc trưng đã được chuyển đổi sang dạng số.

6.4. Loại bỏ thuộc tính tương quan cao

Để tránh hiện tượng đa cộng tuyến (multicollinearity), chúng ta loại bỏ các thuộc tính số có mối tương quan quá cao với nhau (hệ số tương quan > 0.85). Đa cộng tuyến có thể làm cho mô hình kém ổn định và khó diễn giải.

- Bước 1: Tính ma trận tương quan

Chúng ta tính toán ma trận tương quan Pearson giữa tất cả các cặp thuộc tính số trong DataFrame df2 đã được mã hóa. Sau đó, chúng ta giữ lại phần tam giác trên của ma trận để tránh lặp lại các cặp tương quan

```
# Tính hệ số tương quan Pearson giữa các cặp thuộc tính
upper_corr_mat1 = df2.corr().where(np.triu(np.ones(df2.corr().shape),
k=1).astype(bool))
```

- Bước 2: Rút trích các cặp tương quan

Chúng ta chuyển đổi ma trận tương quan thành một Series các cặp thuộc tính và hệ số tương quan tương ứng, loại bỏ các giá trị NaN. Sau đó, chúng ta hiển thị 10 cặp có hệ số tương quan cao nhất.

```
# Lấy các cặp thuộc tính có giá trị tương quan khác NaN
unique_corr_pairs1 = upper_corr_mat1.unstack().dropna()

# Hiển thị 10 cặp thuộc tính tương quan cao nhất
unique_corr_pairs1.sort_values(ascending=False).head(10)
```

```
Delivery Status_Late delivery  Late_delivery_risk              1.000000
order_year                     ship_year                      0.994073
order_month                    ship_month                     0.952179
Shipping Mode_Standard Class   Days for shipment (scheduled)  0.945696
ship_year                      Order Id                       0.942353
order_year                     Order Id                       0.941952
order_hour                     ship_hour                      0.918932
Order Profit Per Order         Order Item Profit Ratio        0.823689
Product Price                  Order Item Total               0.781781
Delivery Status_Late delivery  Shipping Delay                 0.777644
dtype: float64
```

Nhận xét: Hình ảnh hiển thị 10 cặp thuộc tính có hệ số tương quan cao nhất. Chúng ta thấy có những cặp có tương quan rất cao, gần bằng 1, ví dụ như 'Delivery Status_Late delivery' và 'Late_delivery_risk' (tương quan 1.0), 'order_year' và 'ship_year' (tương quan 0.99), 'order_month' và 'ship_month' (tương quan 0.95).

- Bước 3: Lọc các thuộc tính có tương quan > 0.85

Chúng ta lọc ra danh sách các thuộc tính mà có ít nhất một cặp tương quan với hệ số lớn hơn 0.85. Chúng ta chỉ giữ lại một trong hai thuộc tính của mỗi cặp tương quan cao để loại bỏ đa cộng tuyến. Trong bước này, chúng ta chọn thuộc tính xuất hiện ở cột 'level_0' của DataFrame cortable1 (là một trong hai thuộc tính của cặp tương quan cao).

```
## Chuyển về bảng và lọc ra các thuộc tính có tương quan vượt ngưỡng 0.85
cortable1 =
unique_corr_pairs1.sort_values(ascending=False).reset_index()
drop_corr1 = cortable1[cortable1.iloc[:,2] >
0.85]["level_0"].values.tolist()
drop_corr1   # Danh sách thuộc tính cần loại bỏ
```

```
['Delivery Status_Late delivery',
 'order_year',
 'order_month',
 'Shipping Mode_Standard Class',
 'ship_year',
 'order_year',
 'order_hour']
```

Nhận xét: Hình ảnh hiển thị danh sách các thuộc tính được xác định để loại bỏ do có tương quan cao với các thuộc tính khác. Danh sách này bao gồm: 'Delivery Status_Late delivery', 'order_year', 'order_month', 'Shipping Mode_Standard Class', 'ship_year', 'order_year', 'order_hour'. Lưu ý rằng 'order_year' xuất hiện hai lần, nhưng khi loại bỏ, nó sẽ chỉ bị loại bỏ một lần.

- Bước 4: Loại bỏ các thuộc tính tương quan cao

Cuối cùng, chúng ta loại bỏ các thuộc tính đã xác định khỏi DataFrame df2.

```
# Xóa các thuộc tính tương quan cao khỏi tập dữ liệu
df2.drop(drop_corr1, axis=1, inplace=True)
df2.shape   # Kiểm tra kích thước dữ liệu sau khi loại bỏ
```

```
···   (180519, 47)
```

Nhận xét: Hình ảnh cho thấy kích thước mới của DataFrame df2 sau khi các thuộc tính có tương quan cao đã bị loại bỏ. Số lượng cột đã giảm, cho thấy quá trình loại bỏ đã thành công.

Sau bước này, DataFrame df2 đã được tiền xử lý và biến đổi, sẵn sàng cho việc xây dựng mô hình học máy.

6.5. Kết quả xử lý đặc trưng

Sau quá trình biến đổi đặc trưng, DataFrame df2 đã trải qua những thay đổi quan trọng sau:

- **Mã hóa biến ngày tháng:** Các cột 'shipping date (DateOrders)' và 'order date (DateOrders)' đã được tách thành các thành phần số (năm, tháng, ngày, giờ) và các cột ban đầu đã bị loại bỏ.
- **Mã hóa biến phân loại:** Các thuộc tính phân loại đã được mã hóa thành dạng số bằng phương pháp One-Hot Encoding (cho các thuộc tính có ít giá trị duy nhất) và Label Encoding (cho các thuộc tính có nhiều giá trị duy nhất). Điều này đảm bảo rằng tất cả các đặc trưng đầu vào đều ở định dạng số, phù hợp cho các thuật toán học máy.
- **Loại bỏ thuộc tính tương quan cao:** Các thuộc tính số có hệ số tương quan lớn hơn 0.85 đã được loại bỏ để giảm thiểu hiện tượng đa cộng tuyến, giúp mô hình học tốt hơn và tránh trùng lặp thông tin.

Kết quả cuối cùng là một DataFrame df2 đã được làm sạch, biến đổi và giảm chiều (loại bỏ các thuộc tính không cần thiết hoặc gây nhiễu). DataFrame này chứa các đặc trưng số đã được mã hóa và chọn lọc, sẵn sàng để được sử dụng làm đầu vào cho các mô hình học máy để dự đoán nguy cơ giao hàng trễ.

Bước tiếp theo sẽ là xây dựng và huấn luyện các mô hình học máy trên tập dữ liệu đã được chuẩn bị này.

7. XÂY DỰNG VÀ ĐÁNH GIÁ MÔ HÌNH

Trong giai đoạn này, chúng ta sẽ xây dựng và đánh giá các mô hình học máy để dự đoán biến mục tiêu 'Late_delivery_risk' (0: Không trễ, 1: Trễ). Chúng ta sẽ áp dụng một số thuật toán

phân loại phổ biến và đánh giá hiệu suất của chúng bằng nhiều phương pháp chia tập dữ liệu khác nhau để có cái nhìn toàn diện về khả năng dự đoán của từng mô hình.

7.1. Chia dữ liệu và chuẩn hóa

Để đánh giá khách quan hiệu suất của mô hình, chúng ta chia bộ dữ liệu đã tiền xử lý thành hai tập: tập huấn luyện (train set) để huấn luyện mô hình và tập kiểm tra (test set) để đánh giá khả năng dự đoán trên dữ liệu mới. Tỷ lệ chia thường được sử dụng là 75% cho tập huấn luyện và 25% cho tập kiểm tra, với random_state=42 để đảm bảo tính nhất quán trong quá trình chia.

```
# Bước 1: Tách biến mục tiêu (y) và tập thuộc tính đầu vào (X)
X = df2.drop(["Late_delivery_risk"], axis=1)

# Loại bỏ các cột bắt đầu bằng "Delivery" để tránh dữ liệu bị rò rỉ
(data leakage)
X = X.loc[:, ~X.columns.str.startswith('Delivery')]

# Gán biến mục tiêu
y = df2["Late_delivery_risk"]

# Bước 2: Chia dữ liệu thành tập huấn luyện và kiểm tra với tỷ lệ 75/25
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.25, random_state=42)

# Bước 3: Chuẩn hóa dữ liệu bằng StandardScaler (giúp mô hình học hiệu quả hơn)
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Sau khi chia dữ liệu, chúng ta tiến hành chuẩn hóa các đặc trưng số trong tập huấn luyện và áp dụng cùng phép biến đổi này lên tập kiểm tra bằng cách sử dụng StandardScaler. Chuẩn hóa giúp đưa các đặc trưng về cùng một thang đo, tạo điều kiện tốt hơn cho các thuật toán học máy hoạt động hiệu quả. Lưu ý rằng các cột bắt đầu bằng "Delivery" đã được loại bỏ khỏi tập thuộc tính đầu vào (X) để tránh rò rỉ dữ liệu, vì chúng có thể chứa thông tin trực tiếp liên quan đến việc giao hàng có bị trễ hay không và không nên được sử dụng để huấn luyện mô hình dự đoán.

7.2. Huấn luyện mô hình phân loại

Chúng ta sẽ huấn luyện và đánh giá hiệu suất của ba mô hình phân loại phổ biến:

- **K-Nearest Neighbors (KNN):** Một thuật toán dựa trên khoảng cách, phân loại một điểm dữ liệu dựa trên lớp của k điểm dữ liệu gần nhất trong không gian đặc trưng.
- **Logistic Regression:** Một mô hình tuyến tính sử dụng hàm sigmoid để ước tính xác suất của một sự kiện nhị phân (trong trường hợp này là giao hàng trễ).
- **Naive Bayes (GaussianNB):** Một thuật toán dựa trên định lý Bayes, giả định rằng các đặc trưng là độc lập với nhau trong mỗi lớp.

```
# Khởi tạo danh sách các mô hình phân loại
classifiers = [
    KNeighborsClassifier(),        # K-Nearest Neighbors
    LogisticRegression(),          # Hồi quy logistic
    GaussianNB()                  # Naive Bayes

]

# Danh sách tên tương ứng của các mô hình
model_names = ["KNN", "Logistic Regression", "Naive Bayes"]
```

7.3. Đánh giá với nhiều phương pháp

Để có một đánh giá toàn diện về hiệu suất của các mô hình, chúng ta sẽ sử dụng nhiều phương pháp chia tập dữ liệu và đánh giá khác nhau:

- **Train/Test Split cố định:** Đánh giá mô hình trên một lần chia tập dữ liệu huấn luyện và kiểm tra.

```
# 1. Đánh giá với train/test split cố định
accuracy_results_1 = []
for clf in classifiers:
    # Huấn luyện mô hình và dự đoán trên tập test
    y_pred = clf.fit(X_train, y_train).predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    accuracy_results_1.append(round(acc * 100, 2))
```

- **K-Fold Cross Validation (KFold):** Chia tập dữ liệu thành K phần (trong trường hợp này là 5), huấn luyện mô hình trên K-1 phần và đánh giá trên phần còn lại, lặp lại quá trình này K lần và lấy trung bình kết quả.

```
# 2. KFold Cross Validation (chia tập dữ liệu thành 5 phần đều nhau)
accuracy_results_2 = []
for clf in classifiers:
    pipe = make_pipeline(StandardScaler(), clf)
    acc = np.mean(cross_val_score(pipe, X, y, scoring="accuracy",
cv=KFold(), n_jobs=-1))
    accuracy_results_2.append(round(acc * 100, 2))
```

- **Stratified K-Fold Cross Validation (StratifiedKFold):** Tương tự như K-Fold, nhưng đảm bảo rằng tỷ lệ các lớp (giao hàng trễ và không trễ) trong mỗi fold là tương đương với tỷ lệ trong tập dữ liệu gốc. Điều này đặc biệt quan trọng khi dữ liệu mục tiêu không cân bằng.

```
# 3. StratifiedKFold giữ nguyên tỷ lệ các lớp trong mỗi fold
accuracy_results_3 = []
for clf in classifiers:
    pipe = make_pipeline(StandardScaler(), clf)
    acc = np.mean(cross_val_score(pipe, X, y, scoring="accuracy",
cv=StratifiedKFold(), n_jobs=-1))
    accuracy_results_3.append(round(acc * 100, 2))
```

- **ShuffleSplit Cross Validation (ShuffleSplit):** Chia ngẫu nhiên dữ liệu thành các tập huấn luyện và kiểm tra nhiều lần (trong trường hợp này là 5 lần), với tỷ lệ chỉ định (ví dụ: 70% huấn luyện, 30% kiểm tra).

```
# 4. ShuffleSplit chia ngẫu nhiên 5 lần, mỗi lần lấy 70% làm train, 30% test
accuracy_results_4 = []
for clf in classifiers:
    pipe = make_pipeline(StandardScaler(), clf)
    acc = np.mean(cross_val_score(pipe, X, y, scoring="accuracy",
cv=ShuffleSplit(n_splits=5, test_size=0.3), n_jobs=-1))
    accuracy_results_4.append(round(acc * 100, 2))
```

Sau khi đánh giá bằng các phương pháp khác nhau, chúng ta tổng hợp kết quả để so sánh hiệu suất của các mô hình.

```
# Tạo DataFrame để so sánh độ chính xác của các mô hình
eval_frame = pd.DataFrame({
    'Model': model_names,
    'Train/Test Split': accuracy_results_1,
    'KFold (5)': accuracy_results_2,
    'StratifiedKFold (5)': accuracy_results_3,
    'ShuffleSplit': accuracy_results_4
```

```
})

# Tính độ chính xác trung bình
eval_frame['Average Accuracy'] = eval_frame.iloc[:, 1:].mean(axis=1)

# Hiển thị bảng kết quả
eval_frame.sort_values('Average Accuracy', ascending=False)
```

	Model	Train/Test Split	KFold (5)	StratifiedKFold (5)	ShuffleSplit	Average Accuracy
1	Logistic Regression	97.50	97.55	97.37	97.58	97.5000
0	KNN	91.68	90.61	89.09	91.58	90.7400
2	Naive Bayes	90.29	91.02	90.89	90.31	90.6275

Bảng trên cho thấy **Logistic Regression** đạt được độ chính xác trung bình cao nhất (97.50%) trên tất cả các phương pháp đánh giá. Điều này cho thấy mô hình Logistic Regression có hiệu suất ổn định và khả năng dự đoán tốt nhất trên bộ dữ liệu này so với KNN và Naive Bayes.

Mô hình KNN có độ chính xác thấp hơn đáng kể so với Logistic Regression, trong khi Naive Bayes có hiệu suất tương đương với KNN. Sự khác biệt về hiệu suất giữa các phương pháp đánh giá cho từng mô hình không quá lớn, cho thấy các mô hình khá ổn định trên các cách chia dữ liệu khác nhau. Stratified K-Fold có xu hướng cho kết quả thấp hơn một chút so với K-Fold và ShuffleSplit đối với KNN, điều này có thể là do sự phân phối lớp trong các fold ảnh hưởng đến hiệu suất của thuật toán dựa trên khoảng cách này.

Dựa trên kết quả này, chúng ta sẽ tiếp tục phân tích sâu hơn về mô hình Logistic Regression, mô hình có hiệu suất tốt nhất.

7.4. Chọn mô hình tốt nhất: Logistic Regression

Dựa trên kết quả đánh giá ở phần trước, mô hình Logistic Regression cho thấy hiệu suất tổng thể tốt nhất với độ chính xác trung bình cao nhất trên các phương pháp đánh giá khác nhau. Do đó, chúng ta chọn Logistic Regression làm mô hình tiềm năng nhất để dự đoán nguy cơ giao hàng trễ.

Để đánh giá chi tiết hơn hiệu suất của mô hình Logistic Regression trên tập kiểm tra, chúng ta sử dụng hàm classification_report từ thư viện sklearn.metrics để hiển thị các chỉ số quan trọng như precision, recall, F1-score và support cho từng lớp (0: Không trễ, 1: Trễ).

```
# Dựa vào độ chính xác trung bình, ta chọn Logistic Regression làm mô
hình tốt nhất
best_classifier = LogisticRegression()
```

```
# Huấn luyện và dự đoán
y_pred = best_classifier.fit(X_train, y_train).predict(X_test)

# In báo cáo chi tiết các chỉ số: precision, recall, f1-score
print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.98	0.96	0.97	20396
1	0.97	0.99	0.98	24734
accuracy			0.98	45130
macro avg	0.98	0.97	0.97	45130
weighted avg	0.98	0.98	0.98	45130

Báo cáo phân loại cho thấy mô hình Logistic Regression đạt được độ chính xác (accuracy) tổng thể là **98%** trên tập kiểm tra. Điều này có nghĩa là mô hình dự đoán đúng khoảng 98% các trường hợp giao hàng trễ và không trễ.

Chi tiết hơn về từng lớp:

- **Lớp 0 (Không trễ):**
    - **Precision:** 0.98 (Khi mô hình dự đoán một đơn hàng không trễ, có 98% khả năng nó thực sự không trễ).
    - **Recall:** 0.96 (Trong số tất cả các đơn hàng thực tế không trễ, mô hình đã dự đoán đúng 96%).
    - **F1-score:** 0.97 (Điểm trung bình hài hòa giữa precision và recall).
    - **Support:** 20396 (Số lượng mẫu thực tế thuộc lớp 0 trong tập kiểm tra).
- **Lớp 1 (Trễ):**
    - **Precision:** 0.97 (Khi mô hình dự đoán một đơn hàng trễ, có 97% khả năng nó thực sự trễ).
    - **Recall:** 0.99 (Trong số tất cả các đơn hàng thực tế trễ, mô hình đã dự đoán đúng 99%).
    - **F1-score:** 0.98 (Điểm trung bình hài hòa giữa precision và recall).
    - **Support:** 24734 (Số lượng mẫu thực tế thuộc lớp 1 trong tập kiểm tra).

Các chỉ số precision, recall và F1-score đều rất cao cho cả hai lớp, cho thấy mô hình Logistic Regression có khả năng phân loại tốt và có sự cân bằng giữa việc dự đoán chính xác các

trường hợp giao hàng không trễ và các trường hợp giao hàng trễ.

7.5. Đánh giá chi tiết Logistic Regression

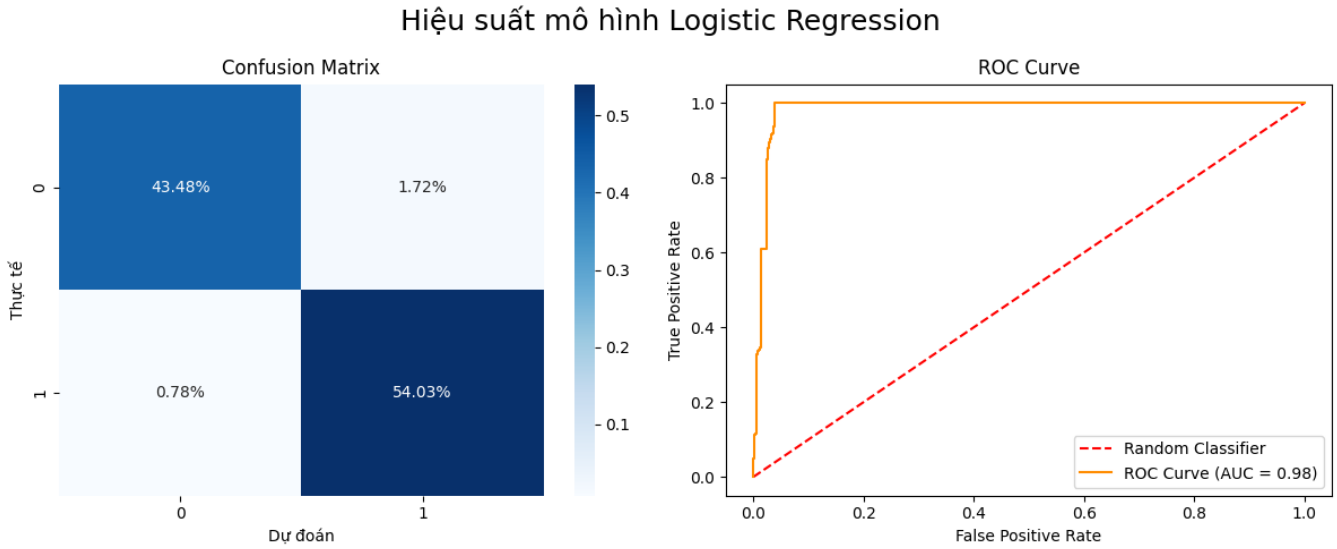
Để hiểu rõ hơn về hiệu suất của mô hình Logistic Regression, chúng ta sẽ trực quan hóa ma trận nhầm lẫn (Confusion Matrix) và đường cong ROC (Receiver Operating Characteristic) cùng với giá trị AUC (Area Under the Curve).

- **Confusion Matrix:** Hiển thị số lượng dự đoán đúng và sai cho từng lớp, giúp đánh giá loại lỗi mà mô hình mắc phải (ví dụ: dự đoán sai một đơn hàng không trễ thành trễ, hoặc ngược lại).

- **ROC Curve & AUC:** Đường cong ROC biểu diễn mối quan hệ giữa tỷ lệ dương tính thực (True Positive Rate - TPR) và tỷ lệ dương tính giả (False Positive Rate - FPR) ở các ngưỡng phân loại khác nhau. Giá trị AUC là diện tích dưới đường cong ROC, một giá trị AUC càng gần 1 thì mô hình càng tốt trong việc phân biệt giữa hai lớp.

```
# Confusion matrix: ma trận nhầm lẫn
cf_matrix = confusion_matrix(y_test, y_pred)

# Lấy xác suất dự đoán để vẽ đường ROC
y_probs = best_classifier.predict_proba(X_test)[:, 1]
fpr, tpr, _ = roc_curve(y_test, y_probs)
roc_auc = auc(fpr, tpr)

# Vẽ biểu đồ
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 5))
fig.suptitle("Hiệu suất mô hình Logistic Regression", fontsize=18)

# Biểu đồ Confusion Matrix
sns.heatmap(cf_matrix / np.sum(cf_matrix), annot=True, fmt='.2%',
cmap='Blues', ax=ax1)
ax1.set_title('Confusion Matrix')
ax1.set_xlabel('Dự đoán')
ax1.set_ylabel('Thực tế')

# Biểu đồ ROC Curve
ax2.plot([0, 1], [0, 1], 'r--', label='Random Classifier')
ax2.plot(fpr, tpr, color='darkorange', label=f'ROC Curve (AUC =
{roc_auc:.2f})')
ax2.set_xlabel('False Positive Rate')
ax2.set_ylabel('True Positive Rate')
```

```
ax2.set_title('ROC Curve')
ax2.legend(loc='lower right')

plt.tight_layout()
plt.show()
```



Confusion Matrix:

Hình ảnh ma trận nhầm lẫn cho thấy:

- Số lượng dự đoán đúng cho lớp 0 (giao hàng không trễ) là khoảng **43.48%** tổng số mẫu.
- Số lượng dự đoán sai khi dự đoán là giao hàng trễ nhưng thực tế không trễ là **1.72%**.
- Số lượng dự đoán sai khi dự đoán là không trễ nhưng thực tế là giao hàng trễ là **0.78%**.
- Số lượng dự đoán đúng cho lớp 1 (giao hàng trễ) là khoảng **54.03%** tổng số mẫu.

Ma trận nhầm lẫn cho thấy mô hình có khả năng dự đoán đúng cao cho cả hai lớp, với tỷ lệ lỗi tương đối thấp.

ROC Curve & AUC:

Đường cong ROC cho thấy hiệu suất phân loại của mô hình Logistic Regression. Giá trị AUC (Area Under the Curve) đạt được là **0.98**. Giá trị AUC gần 1 cho thấy mô hình có khả năng phân biệt rất tốt giữa các đơn hàng có nguy cơ giao hàng trễ và các đơn hàng không có nguy cơ. Đường cong ROC nằm xa đường chéo (đại diện cho một bộ phân loại ngẫu nhiên), cho thấy mô hình hoạt động tốt hơn nhiều so với việc dự đoán ngẫu nhiên.

8. DỰ ĐOÁN & PHÂN TÍCH KẾT QUẢ

Trong phần này, chúng ta sẽ sử dụng mô hình Logistic Regression đã được huấn luyện để dự đoán nguy cơ giao hàng trễ trên tập dữ liệu kiểm tra và phân tích mức độ ảnh hưởng của các đặc trưng đầu vào đến kết quả dự đoán.

8.1. Dự đoán giao hàng trễ

Mô hình Logistic Regression đã được huấn luyện sẽ được sử dụng để dự đoán nhãn (0 hoặc 1) cho biến mục tiêu 'Late_delivery_risk' trên tập dữ liệu kiểm tra (X_test). Nhãn 1 biểu thị đơn hàng có nguy cơ giao hàng trễ, và nhãn 0 biểu thị đơn hàng không có nguy cơ trễ.

```
# Dự đoán nhãn 0/1: Có rủi ro giao hàng trễ hay không
pred = best_classifier.predict(X_test)

# Chuyển kết quả dự đoán sang DataFrame để tiện hiển thị
prediction = pd.DataFrame(pred, columns=['Late Delivery Risk'])

# Xem trước 5 kết quả đầu tiên
prediction.head()
```

	Late Delivery Risk
0	1
1	1
2	0
3	1
4	0

Nhận xét: Hình ảnh này hiển thị 5 kết quả dự đoán đầu tiên của mô hình Logistic Regression trên tập dữ liệu kiểm tra. Cột 'Late Delivery Risk' cho biết dự đoán của mô hình: 1 nghĩa là mô hình dự đoán đơn hàng có nguy cơ giao hàng trễ, và 0 nghĩa là mô hình dự đoán đơn hàng không có nguy cơ giao hàng trễ. Ví dụ, hai đơn hàng đầu tiên được dự đoán là có nguy cơ giao hàng trễ (giá trị 1), trong khi đơn hàng thứ ba và thứ năm được dự đoán là không có nguy cơ (giá trị 0), và đơn hàng thứ tư được dự đoán là có nguy cơ (giá trị 1). Đây chỉ là một phần nhỏ của kết quả dự đoán trên toàn bộ tập kiểm tra.

8.2. Phân tích ảnh hưởng của đặc trưng

Một ưu điểm của mô hình Logistic Regression là khả năng cung cấp thông tin về tầm quan trọng của các đặc trưng thông qua các hệ số (coefficients) của chúng. Hệ số dương cho thấy đặc trưng đó có xu hướng làm tăng khả năng giao hàng trễ, trong khi hệ số âm cho thấy đặc trưng đó có xu hướng làm giảm khả năng giao hàng trễ. Độ lớn tuyệt đối của hệ số thể hiện mức độ ảnh hưởng của đặc trưng đó đến dự đoán.

```
# Trích xuất hệ số của các đặc trưng từ Logistic Regression
feature_importance = best_classifier.coef_[0]

# Tạo DataFrame hiển thị tên đặc trưng và độ quan trọng
feature_names = X.columns
importance_df = pd.DataFrame({
    'Feature': feature_names,
    'Importance': feature_importance
})

# Sắp xếp theo độ lớn tuyệt đối của hệ số (trọng số)
importance_df = importance_df.reindex(

importance_df['Importance'].abs().sort_values(ascending=False).index
)

# Lọc ra các đặc trưng có ảnh hưởng đáng kể (|hệ số| >= 0.05)
filtered_importance_df = importance_df[
    importance_df['Importance'].abs() >= 0.05
]

# Sắp xếp lại cho biểu đồ dễ nhìn
filtered_importance_df = filtered_importance_df.reindex(

filtered_importance_df['Importance'].abs().sort_values(ascending=True).
index
)

# Hiển thị bảng đặc trưng quan trọng
filtered_importance_df
```

	Feature	Importance
16	ship_hour	-0.133717
23	Type_PAYMENT	-0.227362
27	Shipping Mode_Second Class	0.300592
26	Shipping Mode_Same Day	0.559744
21	Type_CASH	0.674576
22	Type_DEBIT	0.751421
24	Type_TRANSFER	-1.073411
33	Order Status	1.252290
25	Shipping Mode_First Class	1.296709
1	Days for shipment (scheduled)	-1.563693
0	Days for shipping (real)	2.786603
13	Shipping Delay	4.470353

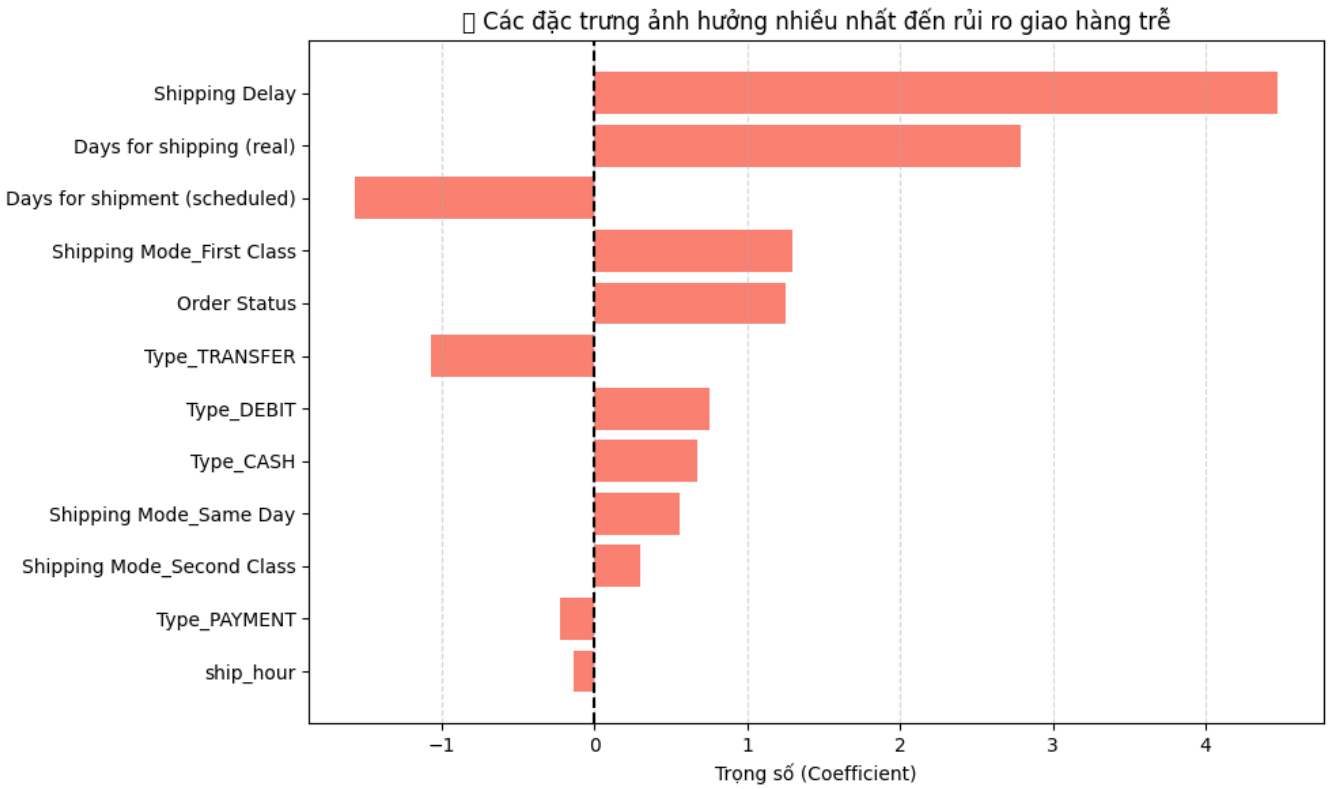
Nhận xét: Bảng này liệt kê các đặc trưng quan trọng nhất và hệ số tương ứng của chúng từ mô hình Logistic Regression. Cột 'Feature' hiển thị tên của đặc trưng, và cột 'Importance' hiển thị giá trị hệ số. Giá trị dương cho thấy tác động làm tăng nguy cơ giao hàng trễ, giá trị âm cho thấy tác động làm giảm nguy cơ. Độ lớn tuyệt đối của giá trị cho biết mức độ ảnh hưởng của đặc trưng đó. Ví dụ, 'Shipping Delay' có hệ số dương lớn nhất (4.47), cho thấy nó là yếu tố làm tăng nguy cơ trễ nhiều nhất. 'Days for shipment (scheduled)' có hệ số âm lớn nhất (-1.56), cho thấy nó là yếu tố làm giảm nguy cơ trễ nhiều nhất trong số các đặc trưng được liệt kê.

8.3. Trực quan hóa đặc trưng quan trọng

Để trực quan hóa mức độ ảnh hưởng của các đặc trưng quan trọng nhất, chúng ta sử dụng biểu đồ thanh ngang. Độ dài của thanh và dấu của giá trị (dương hoặc âm) thể hiện mức độ và hướng tác động của đặc trưng lên nguy cơ giao hàng trễ.

```
plt.figure(figsize=(10, 6))
plt.barh(
    filtered_importance_df['Feature'],
    filtered_importance_df['Importance'],
    color='salmon'
)
plt.axvline(x=0, color='black', linestyle='--')
plt.xlabel('Trọng số (Coefficient)')
plt.title('🎯 Các đặc trưng ảnh hưởng nhiều nhất đến rủi ro giao hàng trễ')
```

```
plt.tight_layout()
plt.grid(axis='x', linestyle='--', alpha=0.5)
plt.show()
```



Nhận xét: Biểu đồ thanh ngang này một cách trực quan hóa mức độ ảnh hưởng của các đặc trưng đến khả năng giao hàng bị trễ. Các thanh màu đỏ san hô thể hiện hệ số của từng đặc trưng. Những đặc trưng có thanh nằm về phía bên phải của đường kẻ dọc màu đen (tại giá trị 0) có hệ số dương, cho thấy chúng làm tăng nguy cơ giao hàng trễ. Ngược lại, những đặc trưng có thanh nằm về phía bên trái có hệ số âm, cho thấy chúng làm giảm nguy cơ này. Độ dài của mỗi thanh biểu thị mức độ ảnh hưởng của đặc trưng đó. Chúng ta có thể thấy rõ 'Shipping Delay' và 'Days for shipping (real)' có tác động tăng nguy cơ trễ lớn nhất, trong khi 'Days for shipment (scheduled)' và 'Type_TRANSFER' lại có tác động giảm nguy cơ trễ đáng kể.

8.4. Diễn giải kết quả và ý nghĩa

Dựa trên bảng hệ số và biểu đồ trực quan hóa, chúng ta có thể đưa ra những diễn giải quan trọng sau:

- **Shipping Delay:** Đây là đặc trưng có hệ số dương lớn nhất (4.47), cho thấy sự chậm trễ trong quá trình vận chuyển có tác động mạnh mẽ nhất đến việc giao hàng bị trễ. Điều này là một phát hiện logic và củng cố tầm quan trọng của việc theo dõi và giảm thiểu sự chậm trễ trong quá trình vận chuyển.
- **Days for shipping (real):** Số ngày vận chuyển thực tế cũng có hệ số dương lớn (2.79), cho thấy thời gian vận chuyển càng dài thì nguy cơ giao hàng trễ càng cao.

- **Days for shipment (scheduled):** Số ngày vận chuyển dự kiến có hệ số âm (-1.56), ngụ ý rằng khi thời gian giao hàng dự kiến dài hơn, có thể có nhiều "buffer" hơn, dẫn đến giảm nguy cơ giao hàng trễ so với lịch trình.
- **Shipping Mode_First Class:** Việc sử dụng hình thức vận chuyển First Class có hệ số dương (1.30), điều này có vẻ trái ngược với suy nghĩ vì người ta thường nghĩ First Class sẽ nhanh hơn. Tuy nhiên, có thể có những yếu tố khác liên quan đến các đơn hàng được vận chuyển bằng First Class (ví dụ: yêu cầu giao hàng gấp hơn, dẫn đến dễ bị trễ nếu có bất kỳ vấn đề nào).
- **Order Status:** Trạng thái đơn hàng cũng có hệ số dương (1.25), cho thấy một số trạng thái đơn hàng nhất định có thể liên quan đến nguy cơ giao hàng trễ cao hơn. Cần phân tích sâu hơn về các trạng thái cụ thể.
- **Type_TRANSFER:** Loại giao dịch TRANSFER có hệ số âm (-1.07), cho thấy các đơn hàng thuộc loại này có xu hướng ít bị giao trễ hơn.
- **Các hình thức thanh toán (Type_DEBIT, Type_CASH, Type_PAYMENT):** Các hình thức thanh toán khác nhau có hệ số khác nhau, cho thấy có thể có mối liên hệ giữa phương thức thanh toán và nguy cơ giao hàng trễ.
- **Shipping Mode_Same Day và Shipping Mode_Second Class:** So với các phương thức khác (có thể là Standard), việc sử dụng Same Day có hệ số dương (0.56) và Second Class cũng có hệ số dương (0.30), cho thấy chúng có thể liên quan đến nguy cơ giao hàng trễ cao hơn. Điều này có thể do áp lực thời gian hoặc các yếu tố khác liên quan đến các loại hình vận chuyển này.
- **ship_hour:** Giờ đặt hàng có hệ số âm (-0.13), cho thấy có thể có một số khung giờ đặt hàng nhất định liên quan đến việc giao hàng đúng hẹn hơn.

Những phân tích này cung cấp những hiểu biết giá trị về các yếu tố nào đang ảnh hưởng đến nguy cơ giao hàng trễ tại DataCo. Dựa trên những phát hiện này, DataCo có thể tập trung vào việc cải thiện các quy trình liên quan đến các đặc trưng có hệ số dương cao (ví dụ: giảm thiểu 'Shipping Delay', tối ưu hóa thời gian vận chuyển thực tế) và tận dụng những yếu tố liên quan đến hệ số âm để nâng cao hiệu quả giao hàng.

9. KẾT LUẬN & KIẾN NGHỊ

9.1. Tóm tắt phát hiện chính

Qua quá trình phân tích dữ liệu lớn chuỗi cung ứng của DataCo và xây dựng mô hình dự đoán nguy cơ giao hàng trễ, chúng tôi đã đạt được những phát hiện chính sau:

- **Tiền xử lý dữ liệu:** Dữ liệu đã được làm sạch và chuẩn hóa thông qua việc xử lý giá trị thiếu (loại bỏ cột 'Order Zipcode' và điền giá trị cho các cột còn lại), loại bỏ các cột không có giá trị phân biệt ('Product Description', 'Customer Password', 'Product Status', 'Customer Email'), xử lý phương sai thấp (ghi nhận thông tin), loại bỏ các cột không cần thiết (các ID, thông tin cá nhân chi tiết, 'Product Image', 'Order Item Discount', 'Product Category Id'), và xử lý đa cộng tuyến (loại bỏ các cột 'Customer Id', 'Order Item Product Price', 'Sales per customer', 'Benefit per order', 'Sales').
- **Biến đổi đặc trưng:** Các biến ngày tháng ('shipping date (DateOrders)' và 'order date (DateOrders)') đã được tách thành các thành phần năm, tháng, ngày, giờ. Các biến phân loại đã được mã hóa bằng One-Hot Encoding và Label Encoding tùy thuộc vào số lượng giá trị duy nhất.
- **Xây dựng và đánh giá mô hình:** Ba mô hình phân loại (KNN, Logistic Regression, Naive Bayes) đã được xây dựng và đánh giá bằng nhiều phương pháp chia tập dữ liệu (Train/Test Split, K-Fold, Stratified K-Fold, ShuffleSplit). **Logistic Regression** cho thấy hiệu suất tốt nhất với độ chính xác trung bình cao nhất (khoảng 97.5%).
- **Đánh giá chi tiết mô hình tốt nhất:** Báo cáo phân loại cho thấy Logistic Regression đạt độ chính xác 98% trên tập kiểm tra, với precision, recall và F1-score cao cho cả lớp giao hàng trễ và không trễ, cho thấy khả năng phân loại tốt và cân bằng.
- **Phân tích tầm quan trọng của đặc trưng:** Các đặc trưng có ảnh hưởng lớn nhất đến nguy cơ giao hàng trễ theo mô hình Logistic Regression là 'Shipping Delay' (tác động tăng nguy cơ trễ), 'Days for shipping (real)' (tác động tăng nguy cơ trễ), 'Days for shipment (scheduled)' (tác động giảm nguy cơ trễ), và các yếu tố liên quan đến phương thức vận chuyển và trạng thái đơn hàng.

9.2. Thảo luận ý nghĩa kết quả

Kết quả phân tích cho thấy mô hình Logistic Regression là một công cụ hiệu quả để dự đoán nguy cơ giao hàng trễ trong chuỗi cung ứng của DataCo. Độ chính xác cao của mô hình cho phép DataCo xác định sớm các đơn hàng có khả năng bị trễ, từ đó có những biện pháp can thiệp kịp thời.

Việc xác định được các đặc trưng quan trọng có ý nghĩa lớn trong việc tối ưu hóa quy trình giao hàng. 'Shipping Delay' là yếu tố có tác động mạnh nhất làm tăng nguy cơ trễ, điều này nhấn mạnh tầm quan trọng của việc giảm thiểu sự chậm trễ trong quá trình vận chuyển. 'Days for shipping (real)' cũng cho thấy thời gian vận chuyển càng dài thì nguy cơ trễ càng cao, gợi ý việc tối ưu hóa lộ trình và phương thức vận chuyển. Ngược lại, 'Days for shipment (scheduled)' có tác động làm giảm nguy cơ trễ, có thể do việc lên lịch giao hàng dài hơn tạo ra nhiều "buffer" hơn.

Các yếu tố liên quan đến phương thức vận chuyển ('Shipping Mode') và trạng thái đơn hàng ('Order Status') cũng đóng vai trò quan trọng và cần được xem xét kỹ lưỡng để hiểu rõ hơn về cách chúng ảnh hưởng đến hiệu suất giao hàng.

9.3. Gợi ý cải tiến quy trình giao hàng

Dựa trên những phát hiện trên, chúng tôi đề xuất DataCo xem xét các cải tiến sau trong quy trình giao hàng:

- **Tối ưu hóa quy trình vận chuyển:** Tập trung vào việc giảm thiểu 'Shipping Delay' bằng cách cải thiện quản lý kho, quy trình đóng gói và giao hàng, có thể thông qua việc lựa chọn đối tác vận chuyển đáng tin cậy và tối ưu hóa lộ trình giao hàng.
- **Quản lý thời gian giao hàng dự kiến:** Xem xét cách thiết lập 'Days for shipment (scheduled)' một cách hợp lý để giảm thiểu nguy cơ trễ thực tế, có thể bằng cách cung cấp ước tính thời gian giao hàng chính xác hơn dựa trên các yếu tố khác như địa điểm, phương thức vận chuyển.
- **Phân tích sâu hơn về phương thức vận chuyển:** Nghiên cứu kỹ lưỡng hơn về tác động của từng 'Shipping Mode' đến nguy cơ giao hàng trễ để đưa ra các quyết định lựa chọn phương thức vận chuyển hiệu quả hơn.
- **Theo dõi và quản lý trạng thái đơn hàng:** Phân tích các 'Order Status' khác nhau để xác định các trạng thái nào có liên quan đến nguy cơ giao hàng trễ cao hơn, từ đó có các biện pháp can thiệp sớm.
- **Đầu tư vào hệ thống theo dõi và cảnh báo sớm:** Xây dựng hoặc nâng cấp hệ thống theo dõi đơn hàng và cảnh báo sớm các đơn hàng có nguy cơ bị trễ để có thể chủ động giải quyết vấn đề trước khi nó xảy ra.

9.4. Đề xuất phát triển mở rộng trong tương lai

Để tiếp tục nâng cao khả năng dự đoán và tối ưu hóa chuỗi cung ứng, chúng tôi đề xuất các hướng phát triển mở rộng sau:

- **Sử dụng các mô hình học máy phức tạp hơn:** Thử nghiệm với các mô hình học máy tiên tiến hơn như Random Forest, Gradient Boosting (ví dụ: XGBoost, LightGBM) hoặc mạng nơ-ron để có thể nắm bắt các mối quan hệ phức tạp hơn trong dữ liệu và có khả năng cải thiện độ chính xác dự đoán.
- **Kết hợp dữ liệu bên ngoài:** Tích hợp thêm các nguồn dữ liệu bên ngoài có thể ảnh hưởng đến giao hàng như điều kiện thời tiết, tình hình giao thông, hoặc các sự kiện đặc biệt.
- **Phân tích nguyên nhân gốc rễ của giao hàng trễ:** Thực hiện phân tích sâu hơn về các trường hợp giao hàng trễ đã xảy ra để xác định nguyên nhân gốc rễ và đưa ra các giải pháp phòng ngừa hiệu quả hơn.
- **Xây dựng hệ thống dự đoán thời gian giao hàng:** Thay vì chỉ dự đoán nguy cơ trễ, phát triển mô hình dự đoán thời gian giao hàng cụ thể để cung cấp thông tin chính xác hơn cho khách hàng.
- **Triển khai hệ thống dự đoán thời gian thực:** Xây dựng hệ thống dự đoán và cảnh báo nguy cơ giao hàng trễ theo thời gian thực để có thể can thiệp ngay lập tức khi phát hiện vấn đề.

10. TÀI LIỆU THAM KHẢO

Các Notebook và Tài liệu Tham khảo trên Kaggle:

- Predict Late Delivery Risk: https://www.kaggle.com/code/thmaith/predict-late-delivery-risk
- Prediksi Resiko Late Delivery DataCo Supply Chain: https://www.kaggle.com/code/farreldr/prediksi-resiko-late-delivery-dataco-supply-chain
- Supply Chain Analysis: https://www.kaggle.com/code/adityabatsexemplary/supply-chain