ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

Khai phá dữ liệu

So sánh các phương pháp học máy trong bài toán dự báo người sống sót ở thảm hoạ Titanic

GVHD: Bùi Tiến Đức
SV thực hiện: Hoa Toàn Hạc – 2201917
Mai Huy Hiệp – 2211045

TP. HỒ CHÍ MINH, THÁNG 4, 2025

Mục lục

Lời mở đầu

Dự đoán khả năng sống sót của hành khách trên tàu Titanic là một trong những bài toán kinh điển trong lĩnh vực khoa học dữ liệu và học máy. Vụ đắm tàu Titanic năm 1912 đã để lại một lượng dữ liệu đáng kể về các hành khách – bao gồm thông tin nhân khẩu học, hạng ghế, vị trí lên tàu và nhiều đặc trưng khác. Những dữ liệu này không chỉ phản ánh một sự kiện lịch sử mà còn tạo điều kiện cho việc nghiên cứu các kỹ thuật phân tích dữ liệu hiện đại và xây dựng các mô hình dự đoán.

Mục tiêu chính của đề tài là phát triển một hệ thống dự đoán khả năng sống sót của hành khách dựa trên các thông tin có sẵn, từ đó đánh giá hiệu quả của nhiều mô hình học máy khác nhau. Đề tài không chỉ tập trung vào khía cạnh mô hình hóa mà còn đi sâu vào quá trình tiền xử lý dữ liệu, thiết kế đặc trưng, đánh giá mô hình và tối ưu hóa siêu tham số – những yếu tố then chốt ảnh hưởng đến độ chính xác và tính tổng quát của hệ thống dự đoán.

Bên cạnh việc triển khai các mô hình truyền thống như **Logistic Regression**, **Random Forest**, và **Gradient Boosting**, nhóm còn ứng dụng các mô hình hiện đại như **XGBoost**, **LightGBM** và mạng nơ-ron đa tầng **MLP**, đồng thời kết hợp các mô hình bằng kỹ thuật `Voting Ensemble` để nâng cao hiệu suất. Quá trình tối ưu hóa tham số được thực hiện bằng phương pháp Tối ưu Bayes (Bayesian Optimization), nhằm tìm ra cấu hình tốt nhất cho từng mô hình.

Qua dự án này, nhóm kỳ vọng không chỉ đạt được kết quả dự đoán tốt, mà còn rút ra được những bài học quan trọng về quy trình phát triển mô hình học máy, từ tiền xử lý đến đánh giá và triển khai. Đồng thời, bài toán Titanic đóng vai trò như một bước đệm giúp làm quen với các kỹ thuật nền tảng trong khai phá dữ liệu và trí tuệ nhân tạo ứng dụng trong các bài toán thực tiễn.

1 Giới thiệu về dữ liệu

1.1 Tổng quan về dữ liệu

Bộ dữ liệu này chứa thông tin chi tiết về hành khách trên tàu Titanic. Mục tiêu của cuộc thi trên Kaggle là dự đoán khả năng sống sót của mỗi hành khách dựa trên các thuộc tính có sẵn.

Tàu Titanic là một tàu chở khách sang trọng của Anh hoạt động từ năm 1912 đến năm 1912. Trong chuyến hải hành đầu tiên, từ Southampton đến Thành phố New York, con tàu đã va chạm với một tảng băng trôi và chìm, dẫn đến cái chết của hơn 1.500 người trong số khoảng 2.224 hành khách và thủy thủ đoàn.

Dữ liệu này cung cấp một cái nhìn sâu sắc về các yếu tố có thể ảnh hưởng đến khả năng sống sót của hành khách trong thảm họa.

Thông tin về bộ dữ liệu Titanic được tóm tắt trong bảng dưới đây:

Thuộc tính	**Thông tin**
Nguồn dữ liệu	train.csv, test.csv
Đối tượng quan sát	Hành khách trên tàu Titanic
Số lượng quan sát (train.csv)	891
Số lượng quan sát (test.csv)	418
Số lượng biến	12

1.2 Tổng quan về biến

Tên biến	**Phân loại biến**	**Đơn vị**	**Mô tả**
PassengerId	Định lượng (Nominal)		ID của hành khách.
Survived	Định tính (Nominal)		Biến mục tiêu: 0 = Không sống sót, 1 = Sống sót.
Pclass	Định tính (Ordinal)		Hạng vé của hành khách (1 = Hạng nhất, 2 = Hạng nhì, 3 = Hạng ba).
Name	Định tính (Nominal)		Tên của hành khách.
Sex	Định tính (Nominal)		Giới tính của hành khách.
Age	Định lượng (Continuous)	Năm	Tuổi của hành khách.
SibSp	Định lượng (Discrete)		Số lượng anh chị em/vợ chồng đi cùng trên tàu.
Parch	Định lượng (Discrete)		Số lượng cha mẹ/con cái đi cùng trên tàu.
Ticket	Định tính (Nominal)		Số vé của hành khách.
Fare	Định lượng (Continuous)	Đơn vị tiền tệ	Giá vé của hành khách.
Cabin	Định tính (Nominal)		Số cabin của hành khách.

Embarked	Định tính (Nominal)		Cảng lên tàu của hành khách (C = Cherbourg, Q = Queenstown, S = Southampton).

2 Khám phá dữ liệu (EDA)

Phân tích khám phá dữ liệu (Exploratory Data Analysis - EDA) là bước quan trọng nhằm hiểu rõ cấu trúc, đặc điểm, và các mẫu tiềm năng trong tập dữ liệu Titanic trước khi tiến hành tiền xử lý và xây dựng mô hình học máy. Mục tiêu của EDA là xác định các đặc trưng quan trọng, phát hiện các vấn đề như giá trị thiếu hoặc phân phối lệch, và định hướng cho các bước xử lý dữ liệu tiếp theo. Trong phần này, trình bày tổng quan về tập dữ liệu huấn luyện (train.csv), phân tích thống kê mô tả, đánh giá giá trị thiếu, và đưa ra các nhận xét ban đầu về các đặc trưng có thể ảnh hưởng đến khả năng sống sót của hành khách.

2.1 Tổng quan về tập dữ liệu

Dựa trên phân tích từ train_df.info(), có thể xác định kiểu dữ liệu và số lượng giá trị không thiếu của mỗi cột, được tóm tắt trong bảng :

```
train_df.info()
```

Cột	Kiểu dữ liệu	Số giá trị không thiếu
PassengerId	int64	891
Survived	int64	891
Pclass	int64	891
Name	object	891
Sex	object	891
Age	float64	714
SibSp	int64	891
Parch	int64	891
Ticket	object	891
Fare	float64	891
Cabin	object	204
Embarked	object	889

Từ bảng, nhận thấy:

- Các cột Age, Cabin, và Embarked có giá trị thiếu, với Cabin có tỷ lệ thiếu cao nhất (chỉ 204/891 giá trị không thiếu).
- Các cột Name, Sex, Ticket, Cabin, và Embarked là kiểu phân loại (object), trong khi các cột còn lại là kiểu số (int64 hoặc float64).

2.2 Phân tích thống kê

Để hiểu rõ hơn về phân phối của các biến số, ta sử dụng phương pháp describe() để tính toán các thống kê mô tả:

```
train_df.describe()
```

Kết quả của train_df.describe() cung cấp các thống kê như số lượng, trung bình, độ lệch chuẩn, giá trị tối thiểu, tối đa, và các phân vị (25%, 50%, 75%) cho các cột số, được tóm tắt trong bảng:

Bảng 2: Thống kê mô tả của các cột số trong tập huấn luyện.

Thống kê	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
Count	891	891	891	714	891	891	891
Mean	446.00	0.38	2.31	29.70	0.52	0.38	32.20
Std	257.35	0.49	0.84	14.53	1.10	0.81	49.69
Min	1	0	1	0.42	0	0	0.00
25%	223.50	0	2	20.12	0	0	7.91
50%	446.00	0	3	28.00	0	0	14.45
75%	668.50	1	3	38.00	1	0	31.00
Max	891	1	3	80.00	8	6	512.33

Một số nhận xét từ bảng:

- **Survived**: Tỷ lệ sống sót trung bình là 0.38, tương đương 38% hành khách sống sót, cho thấy bài toán phân loại này có tính chất mất cân bằng (lớp không sống sót chiếm ưu thế).
- **Pclass**: Giá trị trung bình là 2.31, nghiêng về hạng ba (giá trị 3), cho thấy phần lớn hành khách thuộc tầng lớp kinh tế thấp hơn. Phân vị 50% và 75% đều là 3, củng cố nhận định này.
- **Age**: Độ tuổi trung bình là 29.7, với độ lệch chuẩn lớn (14.53), cho thấy sự đa dạng về độ tuổi. Giá trị tối thiểu (0.42) và tối đa (80) chỉ ra rằng tập dữ liệu bao gồm cả trẻ sơ sinh và người cao tuổi.
- **SibSp và Parch**: Hầu hết hành khách đi một mình hoặc với ít người thân (giá trị trung bình lần lượt là 0.52 và 0.38), nhưng một số ít trường hợp đi cùng nhiều người (tối đa 8 anh chị em hoặc 6 cha mẹ/con cái).
- **Fare**: Giá vé có phân phối lệch phải mạnh, với giá trị trung bình là 32.20 nhưng giá trị tối đa lên đến 512.33, phản ánh sự chênh lệch lớn về tình trạng kinh tế giữa các hành khách.

2.3 Phân tích giá trị thiếu

Dựa trên kết quả từ train_df.info(), chúng tôi tính toán tỷ lệ giá trị thiếu cho các cột:

- **Cabin**: Có 687/891 giá trị thiếu, tương đương 77.10%. Tỷ lệ thiếu cao này cho thấy cột Cabin khó khai thác trực tiếp và có thể cần loại bỏ hoặc xử lý đặc biệt (ví dụ: tạo đặc trưng mới dựa trên sự có mặt của giá trị).

- **Age**: Có 177/891 giá trị thiếu, tương đương 19.87%. Đây là tỷ lệ đáng kể nhưng có thể xử lý bằng các phương pháp điền giá trị như trung bình, trung vị, hoặc dựa trên các đặc trưng khác (ví dụ: danh xưng).
- **Embarked**: Chỉ có 2/891 giá trị thiếu, tương đương 0.22%. Tỷ lệ này rất thấp, cho phép xử lý đơn giản như điền giá trị phổ biến nhất (thường là S) hoặc đánh dấu giá trị thiếu.

Các cột còn lại (PassengerId, Survived, Pclass, Name, Sex, SibSp, Parch, Ticket, Fare) không có giá trị thiếu, đảm bảo tính toàn vẹn dữ liệu cho các đặc trưng này.

2.4 Nhận xét ban đầu

Dựa trên phân tích EDA, chúng tôi rút ra một số nhận xét ban đầu về tập dữ liệu và định hướng cho các bước tiếp theo:

- **Tính mất cân bằng của biến mục tiêu**: Tỷ lệ sống sót 38% cho thấy bài toán phân loại này có tính chất mất cân bằng, đòi hỏi các kỹ thuật như cân bằng lớp (oversampling, undersampling) hoặc sử dụng các độ đo đánh giá phù hợp (F1-score, AUC) thay vì chỉ dựa vào độ chính xác.
- **Tầm quan trọng của các đặc trưng phân loại**: Các cột như Sex, Pclass, và Embarked có tiềm năng ảnh hưởng mạnh đến khả năng sống sót. Dựa trên các nghiên cứu trước đây về Titanic, phụ nữ và hành khách ở hạng vé cao (Pclass=1) thường có tỷ lệ sống sót cao hơn.
- **Xử lý giá trị thiếu**:
  - Cột Cabin có tỷ lệ giá trị thiếu quá cao, nên có thể bị loại bỏ hoặc sử dụng để tạo đặc trưng nhị phân (có cabin hay không).
  - Cột Age cần được xử lý cẩn thận, ví dụ: điền giá trị trung bình hoặc dựa trên danh xưng (Title) để tăng độ chính xác.
  - Cột Embarked có thể được điền bằng giá trị phổ biến nhất (S) do tỷ lệ thiếu rất thấp.
- **Tiềm năng feature engineering**:
  - Từ cột Name, danh xưng (Title) như Mr, Mrs, Miss, Master có thể được trích xuất để cung cấp thông tin về giới tính, tình trạng hôn nhân, và địa vị xã hội.
  - Các cột Age và Fare có thể được phân nhóm (ví dụ: trẻ em, người lớn; vé rẻ, vé đắt) để giảm nhiễu và làm nổi bật các xu hướng.
  - Kết hợp SibSp và Parch để tạo đặc trưng mới như kích thước gia đình (FamilySize) có thể phản ánh tác động của số lượng người thân đến khả năng sống sót.
- **Phân phối lệch của Fare**: Sự chênh lệch lớn trong giá vé cho thấy cần áp dụng các kỹ thuật chuẩn hóa (như log transform) hoặc phân nhóm để xử lý phân phối lệch phải, giúp mô hình học máy hoạt động hiệu quả hơn.

Phân tích EDA cung cấp cái nhìn tổng quan về tập dữ liệu, giúp định hướng các bước tiền xử lý và feature engineering trong các phần tiếp theo. Các đặc trưng như Sex, Pclass, Age, và Fare được xác định là có tiềm năng lớn trong việc dự đoán khả năng sống sót, trong khi các vấn đề như giá trị thiếu và phân phối lệch cần được xử lý cẩn thận.

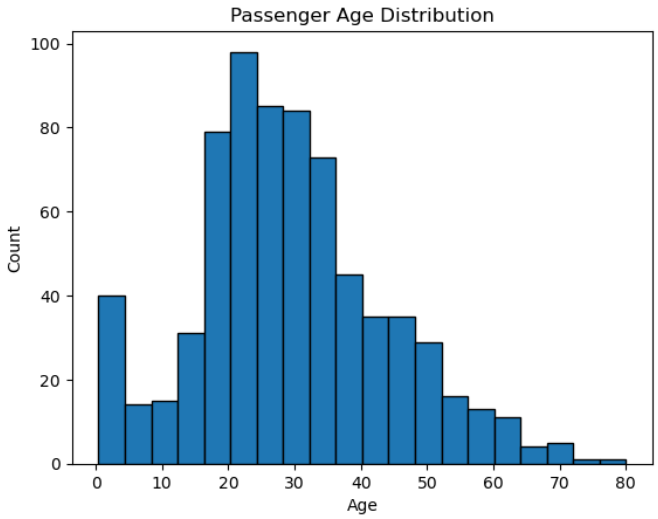
3 Data Visualization

Trực quan hóa dữ liệu đóng vai trò quan trọng trong việc khám phá và hiểu các mẫu, xu hướng, và mối quan hệ trong tập dữ liệu Titanic. Các biểu đồ được tạo bằng các thư viện như seaborn và matplotlib, giúp làm rõ phân phối của các đặc trưng, mối quan hệ giữa biến mục tiêu Survived và các đặc trưng khác, từ đó định hướng cho các bước tiền xử lý và feature engineering. Trong phần này, ta trình bày các biểu đồ phân tích phân phối của Age, tỷ lệ sống sót theo các đặc trưng như Sex, Pclass, Embarked, và Family Size, ma trận tương quan giữa các biến số, cũng như phân bố tuổi theo trạng thái sống sót.

3.1 Phân phối của Age

Để phân tích phân phối độ tuổi của hành khách, ta sử dụng biểu đồ histogram. Đoạn mã sau được suy luận để tạo biểu đồ này:

```
plt.hist(train_df["Age"].dropna(), bins=20, edgecolor="k")
plt.xlabel("Age")
plt.ylabel("Count")
plt.title("Passenger Age Distribution")
plt.show()
```



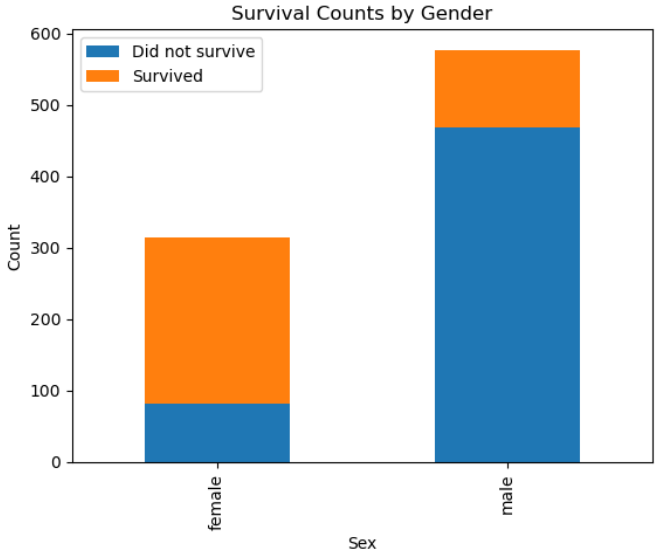
Hình 1: Histogram thể hiện phân phối của Age trong tập huấn luyện.

Hình 1 cho thấy phân phối của Age lệch phải, với phần lớn hành khách nằm trong khoảng 20–40 tuổi. Có một số lượng đáng kể trẻ em (tuổi dưới 10) và một số ít người cao tuổi (tuổi trên 60), với độ tuổi tối đa khoảng 80. Phân phối này gợi ý rằng việc phân nhóm độ tuổi (ví dụ: trẻ em, thanh niên, người lớn, người già) có thể hữu ích trong feature engineering để làm nổi bật các xu hướng liên quan đến khả năng sống sót. Ngoài ra, sự hiện diện của 19.87% giá trị thiếu trong Age (dựa trên phân tích trước đó) đòi hỏi phương pháp điền giá trị phù hợp, chẳng hạn như sử dụng trung bình hoặc dựa trên danh xưng.

3.2 Tỷ lệ sống sót theo Sex

Để đánh giá mối quan hệ giữa giới tính và khả năng sống sót, ta sử dụng biểu đồ cột phân loại để trực quan hóa số lượng hành khách sống sót và không sống sót theo Sex. Đoạn mã sau được suy luận để tạo biểu đồ:

```
survived_by_gender = train_df.groupby("Sex")["Survived"].value_counts()
    .unstack()
survived_by_gender.plot(kind="bar", stacked=True)
plt.xlabel("Sex")
plt.ylabel("Count")
plt.title("Survival Counts by Gender")
plt.legend(["Did not survive", "Survived"])
plt.show()
```



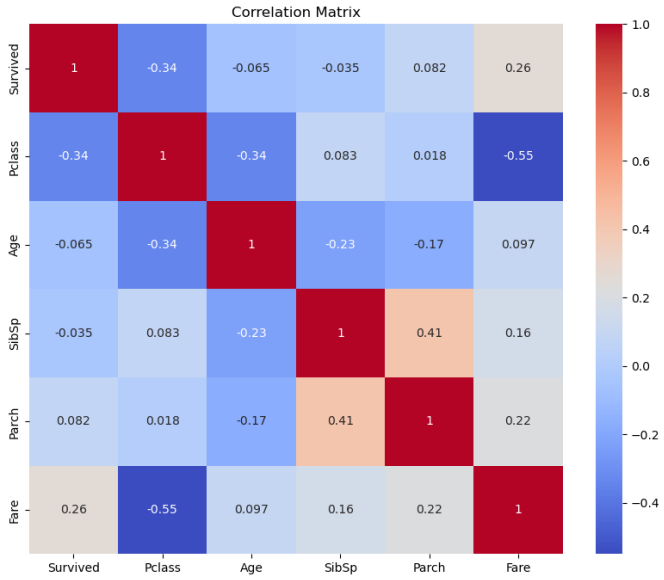
Hình 2: Biểu đồ cột thể hiện số lượng sống sót theo Sex trong tập huấn luyện.

Hình 2 cho thấy phụ nữ (female) có tỷ lệ sống sót cao hơn đáng kể so với nam giới (male). Cụ thể, trong số hành khách nữ, phần lớn sống sót (màu cam), trong khi ở nhóm nam giới, phần lớn không sống sót (màu xanh). Điều này phù hợp với lịch sử Titanic, nơi phụ nữ và trẻ em được ưu tiên trong quá trình cứu hộ. Đặc trưng Sex rõ ràng là một yếu tố quan trọng trong dự đoán khả năng sống sót, và việc mã hóa biến này bằng One-Hot Encoding sẽ được thực hiện trong bước tiền xử lý.

3.3 Ma trận tương quan

Để đánh giá mối quan hệ tuyến tính giữa các biến số, ta sử dụng heatmap để trực quan hóa ma trận tương quan. Đoạn mã sau được suy luận để tạo biểu đồ:

```
# Create a correlation matrix
numerical_df = train_df.select_dtypes(include=['number'])
correlation_matrix = numerical_df.corr()
# Generate a heatmap of the correlation matrix
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm")
plt.title("Correlation Matrix")
plt.show()
```



Hình 3: Heatmap thể hiện ma trận tương quan giữa các biến số trong tập huấn luyện.

Hình 3 cho thấy các mối quan hệ sau:

- Survived có tương quan âm mạnh với Pclass (-0.34), cho thấy hành khách ở hạng vé cao (giá trị Pclass nhỏ hơn) có khả năng sống sót cao hơn.
- Survived có tương quan dương với Fare (0.26), cho thấy hành khách trả giá vé cao hơn có khả năng sống sót cao hơn, có thể do họ ở hạng vé cao.
- Pclass có tương quan âm với Fare (-0.55), điều này hợp lý vì hạng vé cao (giá trị Pclass nhỏ) thường có giá vé cao hơn.
- SibSp và Parch có tương quan dương (0.41), phản ánh rằng hành khách đi cùng anh chị em/vợ chồng thường cũng đi cùng cha mẹ/con cái.
- Các biến khác như Age có tương quan yếu với Survived (-0.065), cho thấy cần các kỹ thuật feature engineering (như phân nhóm tuổi) để khai thác tốt hơn mối quan hệ này.

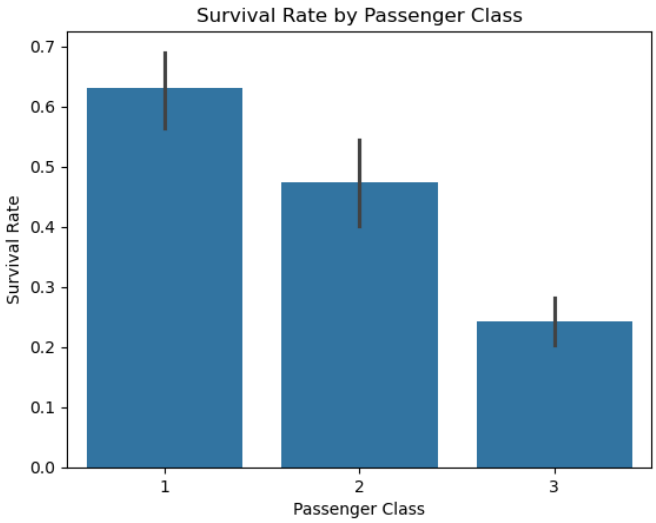
Ma trận tương quan này giúp xác định các đặc trưng quan trọng (Pclass, Fare) và gợi ý rằng các đặc trưng như Age có thể cần được biến đổi để cải thiện hiệu suất mô hình.

3.4 Tỷ lệ sống sót theo Pclass

Để phân tích tỷ lệ sống sót theo hạng vé, ta sử dụng biểu đồ cột để trực quan hóa tỷ lệ sống sót trung bình theo Pclass. Đoạn mã sau được suy luận để tạo biểu đồ:

```
# Calculate survival rate by passenger class
survival_by_class = train_df.groupby("Pclass")["Survived"].mean()

# Bar chart of survival rate by passenger class
sns.barplot(x="Pclass", y="Survived", data=train_df)
plt.xlabel("Passenger Class")
plt.ylabel("Survival Rate")
plt.title("Survival Rate by Passenger Class")
plt.show()
```



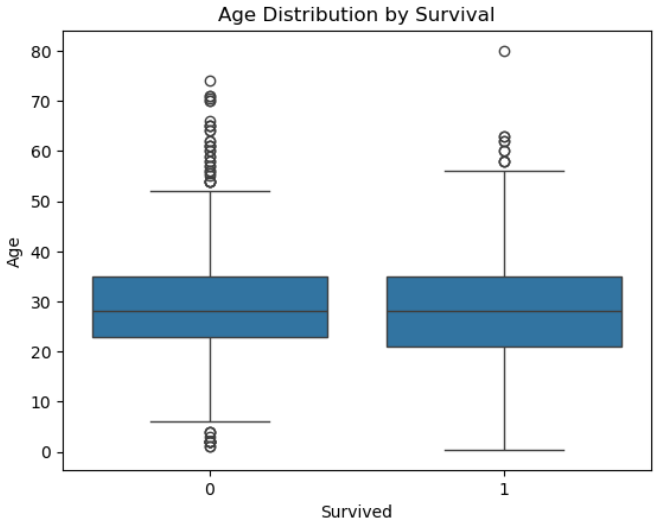
Hình 4: Biểu đồ cột thể hiện tỷ lệ sống sót theo Pclass trong tập huấn luyện.

Hình 4 cho thấy tỷ lệ sống sót giảm dần từ hạng nhất (Pclass=1) đến hạng ba (Pclass=3). Cụ thể, hành khách hạng nhất có tỷ lệ sống sót khoảng 0.63, hạng hai khoảng 0.47, và hạng ba chỉ khoảng 0.24. Điều này phản ánh mối quan hệ giữa địa vị kinh tế và khả năng tiếp cận các phương tiện cứu hộ, củng cố kết quả từ ma trận tương quan rằng Pclass là một đặc trưng quan trọng.

3.5 Phân phối Age theo Survived

Để đánh giá sự khác biệt trong phân phối độ tuổi giữa hai nhóm sống sót và không sống sót, ta sử dụng biểu đồ box plot. Đoạn mã sau được suy luận để tạo biểu đồ:

```
sns.boxplot(x="Survived", y="Age", data=train_df)
plt.xlabel("Survived")
plt.ylabel("Age")
plt.title("Age Distribution by Survival")
plt.show()
```



Hình 5: Box plot thể hiện phân phối của Age theo Survived trong tập huấn luyện.

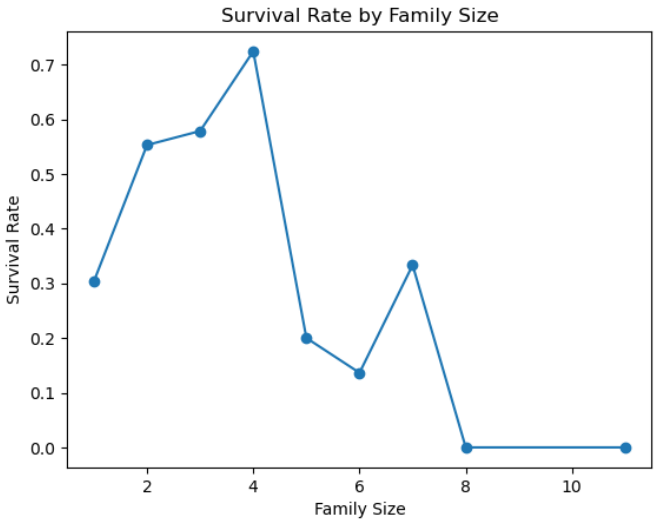
Hình 5 cho thấy độ tuổi trung vị của nhóm sống sót (1) và không sống sót (0) khá tương đồng, đều nằm trong khoảng 28–30 tuổi. Tuy nhiên, nhóm không sống sót có một số giá trị ngoại lệ (outliers) ở độ tuổi cao (lên đến 70–80), trong khi nhóm sống sót có ít giá trị ngoại lệ hơn. Điều này gợi ý rằng tuổi tác không có sự khác biệt rõ rệt giữa hai nhóm, nhưng trẻ em (tuổi thấp) có thể có tỷ lệ sống sót cao hơn do được ưu tiên cứu hộ, điều này cần được khai thác thêm qua feature engineering (ví dụ: tạo đặc trưng AgeGroup).

3.6 Tỷ lệ sống sót theo Family Size

Để phân tích tác động của kích thước gia đình đến khả năng sống sót, ta sử dụng biểu đồ đường để trực quan hóa tỷ lệ sống sót theo Family Size (tổng của SibSp và Parch cộng thêm 1). Đoạn mã sau được suy luận để tạo biểu đồ:

```
# Create a new feature for family size
train_df["FamilySize"] = train_df["SibSp"] + train_df["Parch"] + 1
test_df["FamilySize"] = test_df["SibSp"] + test_df["Parch"] + 1
# Calculate survival rate by family size
survival_by_family_size = train_df.groupby("FamilySize")["Survived"].
    mean()
```

```python
# Line plot of survival rate by family size
plt.plot(survival_by_family_size.index, survival_by_family_size.values,
    marker="o")
plt.xlabel("Family Size")
plt.ylabel("Survival Rate")
plt.title("Survival Rate by Family Size")
plt.show()
```



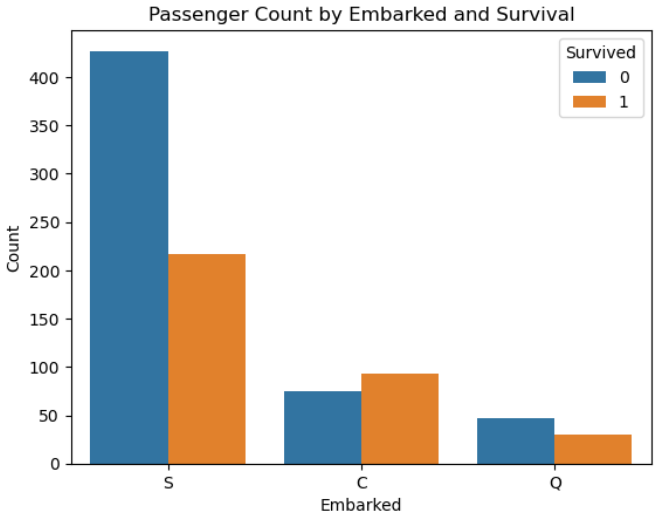
Hình 6: Biểu đồ đường thể hiện tỷ lệ sống sót theo Family Size trong tập huấn luyện.

Hình 6 cho thấy tỷ lệ sống sót cao nhất ở các gia đình có kích thước 3–4 người (khoảng 0.55–0.72), trong khi các gia đình lớn hơn (5 người trở lên) có tỷ lệ sống sót giảm mạnh, xuống gần 0 ở gia đình 8–11 người. Hành khách đi một mình (Family Size=1) có tỷ lệ sống sót thấp (khoảng 0.3). Điều này gợi ý rằng kích thước gia đình vừa phải có thể tăng khả năng sống sót, trong khi gia đình quá lớn hoặc đi một mình có thể gặp bất lợi. Đặc trưng Family Size là một lựa chọn tốt để thêm vào tập dữ liệu trong bước feature engineering.

3.7 Tỷ lệ sống sót theo Embarked

Để đánh giá tác động của cảng lên tàu đến khả năng sống sót, ta sử dụng biểu đồ cột để trực quan hóa số lượng hành khách sống sót và không sống sót theo Embarked. Đoạn mã sau được suy luận để tạo biểu đồ:

```python
# Countplot of embarkation point and survival
sns.countplot(x="Embarked", hue="Survived", data=train_df)
plt.xlabel("Embarked")
plt.ylabel("Count")
plt.title("Passenger Count by Embarked and Survival")
plt.show()
```



Hình 7: Biểu đồ cột thể hiện số lượng sống sót theo Embarked trong tập huấn luyện.

Hình 7 cho thấy phần lớn hành khách lên tàu từ cảng Southampton (S), với số lượng không sống sót cao hơn đáng kể so với số sống sót. Hành khách từ Cherbourg (C) có tỷ lệ sống sót cao hơn (gần 50%), trong khi hành khách từ Queenstown (Q) có tỷ lệ sống sót thấp hơn. Điều này có thể liên quan đến phân bố Pclass tại các cảng (ví dụ: Cherbourg có nhiều hành khách hạng nhất hơn), và cần được phân tích thêm để xác định tác động của Embarked đến Survived.

3.8 Nhận xét từ trực quan hóa

Các biểu đồ trực quan hóa cung cấp những hiểu biết quan trọng về tập dữ liệu Titanic:

- Phân phối của Age: Biểu đồ histogram (Hình 1) cho thấy Age lệch phải, gợi ý rằng phân nhóm độ tuổi có thể giúp làm nổi bật các xu hướng liên quan đến khả năng sống sót.
- Tầm quan trọng của Sex và Pclass: Biểu đồ tỷ lệ sống sót theo Sex (Hình 2) và Pclass (Hình 4) nhấn mạnh rằng phụ nữ và hành khách hạng nhất có khả năng sống sót cao hơn, xác nhận vai trò quan trọng của các đặc trưng này.
- Tương quan giữa các biến: Ma trận tương quan (Hình 3) chỉ ra rằng Pclass và Fare có mối quan hệ mạnh với Survived, trong khi Age cần được biến đổi để khai thác tốt hơn.
- Tác động của Family Size: Biểu đồ đường (Hình 6) cho thấy gia đình 3–4 người có tỷ lệ sống sót cao nhất, gợi ý rằng Family Size là một đặc trưng hữu ích.
- Tác động của Embarked: Biểu đồ cột (Hình 7) cho thấy sự khác biệt trong tỷ lệ sống sót giữa các cảng, với Cherbourg có tỷ lệ cao nhất, cần được phân tích thêm.

Các trực quan hóa này không chỉ giúp hiểu rõ dữ liệu mà còn định hướng các bước tiền xử lý (như điền giá trị thiếu cho Age, mã hóa Sex và Embarked) và feature engineering (như tạo AgeGroup, Family Size) trong các phần tiếp theo.

4 Xử lý dữ liệu

Xử lý dữ liệu là bước nền tảng trong quy trình phân tích và xây dựng mô hình học máy cho bài toán dự đoán khả năng sống sót của hành khách trên tàu Titanic. Mục tiêu của quá trình này là làm sạch dữ liệu, xử lý các giá trị thiếu, và tạo ra các đặc trưng mới để tối ưu hóa hiệu suất của các mô hình. Trong phần này, chúng tôi trình bày chi tiết các bước tiền xử lý dữ liệu và kỹ thuật feature engineering được áp dụng trên tập dữ liệu Titanic, bao gồm cả tập huấn luyện (train.csv) và tập kiểm tra (test.csv).

4.1 Tiền xử lý dữ liệu

Tiền xử lý dữ liệu nhằm đảm bảo rằng tập dữ liệu sạch, không chứa giá trị thiếu, và được định dạng phù hợp để huấn luyện các mô hình học máy. Dựa trên phân tích ban đầu của tập dữ liệu, chúng tôi xác định các vấn đề như giá trị thiếu, các biến phân loại cần mã hóa, và sự không đồng nhất giữa tập huấn luyện và kiểm tra. Các bước tiền xử lý được thực hiện bao gồm:

4.1.1 Kiểm tra và xử lý giá trị thiếu

Tập dữ liệu Titanic chứa một số cột với giá trị thiếu, ảnh hưởng đến chất lượng của mô hình nếu không được xử lý đúng cách. Chúng tôi sử dụng phương pháp isnull().sum() từ thư viện pandas để tính toán tỷ lệ giá trị thiếu trong mỗi cột:

```
# Calculate percentage of missing values
missing_percentage = test_df.isnull().sum() / len(test_df) * 100

# Print columns with missing values and their corresponding percentages
print(missing_percentage[missing_percentage > 0])
```

Kết quả cho thấy trong tập kiểm tra, cột Fare có giá trị thiếu, trong khi tập huấn luyện có các cột Age, Cabin, và Embarked bị thiếu dữ liệu. Cụ thể:

- **Cột Cabin**: Trong tập huấn luyện, cột này có tỷ lệ giá trị thiếu cao (khoảng 77%) và không cung cấp đủ thông tin để khai thác. Do đó, chúng tôi quyết định loại bỏ cột Cabin khỏi cả hai tập dữ liệu.
- **Cột Age**: Đối với các giá trị thiếu trong Age (tập huấn luyện), chúng tôi sử dụng phương pháp điền giá trị trung bình thông qua SimpleImputer từ thư viện scikit-learn. Phương pháp này được chọn vì nó duy trì phân phối tổng quát của đặc trưng tuổi, tránh làm sai lệch dữ liệu.
- **Cột Fare**: Trong tập kiểm tra, cột Fare có một số giá trị thiếu. Chúng tôi áp dụng phương pháp điền giá trị trung bình tương tự, như được minh họa trong đoạn mã sau:

```
imputer = SimpleImputer(strategy="mean")  # Use mean for numerical
    features
test_df["Fare"] = imputer.fit_transform(test_df[["Fare"]]).ravel()
     # Fill missing Fare values with mean
```

- **Cột Embarked**: Trong tập huấn luyện, cột Embarked có một số ít giá trị thiếu (khoảng 0.22%). Để xử lý, chúng tôi tạo một cột mới Embarked_nan để đánh dấu các giá trị thiếu, sau đó mã hóa các giá trị còn lại bằng kỹ thuật One-Hot Encoding.

4.1.2 Mã hóa các biến phân loại

Các biến phân loại như Sex, Embarked, Title, AgeGroup, và FareGroup cần được chuyển đổi thành dạng số để phù hợp với các thuật toán học máy. Chúng tôi sử dụng OneHotEncoder từ scikit-learn với tùy chọn drop="first" để tránh hiện tượng đa cộng tuyến. Đoạn mã sau minh họa quá trình mã hóa:

```
encoder = OneHotEncoder(drop="first")
encoded_features = pd.DataFrame(encoder.fit_transform(test_df[[
    "Sex", "Title", "Embarked", "AgeGroup", "FareGroup"
]]).toarray(),
    columns=encoder.get_feature_names_out([
        "Sex", "Title", "Embarked", "AgeGroup", "FareGroup"
    ]))
test_df_encoded = pd.concat([test_df, encoded_features], axis=1)

test_df_encoded.drop([
    'Sex', 'Title', 'Embarked', 'AgeGroup', 'FareGroup'
], axis=1, inplace=True)
```

Kỹ thuật này tạo ra các cột nhị phân cho mỗi giá trị của biến phân loại, ví dụ: Sex_male, Embarked_S, Embarked_C, v.v. Việc loại bỏ cột đầu tiên (drop="first") giúp giảm số chiều dữ liệu và tránh dư thừa thông tin.

4.1.3 Đồng bộ hóa tập huấn luyện và kiểm tra

Một vấn đề phát sinh trong quá trình mã hóa là sự khác biệt giữa tập huấn luyện và kiểm tra. Cụ thể, cột Embarked trong tập kiểm tra không có giá trị thiếu, dẫn đến việc OneHotEncoder không tạo cột Embarked_nan. Để đảm bảo rằng cả hai tập dữ liệu có cùng cấu trúc đặc trưng, chúng tôi thêm cột Embarked_nan vào tập kiểm tra và gán giá trị 0 cho tất cả các mẫu:

```
test_df_encoded.insert(test_df_encoded.columns.get_loc('Embarked_S') +
    1, 'Embarked_nan', 0)
```

Bước này đảm bảo rằng các đặc trưng của tập kiểm tra phù hợp với tập huấn luyện, tránh lỗi khi dự đoán bằng mô hình.

4.2 Feature Engineering

Kỹ thuật feature engineering nhằm tạo ra các đặc trưng mới từ dữ liệu thô, giúp mô hình học máy khai thác tốt hơn các mẫu ẩn trong dữ liệu. Trong bài toán Titanic, chúng tôi tập trung vào việc tạo các đặc trưng phản ánh các yếu tố xã hội, kinh tế, và nhân khẩu học của hành khách, có thể ảnh hưởng đến khả năng sống sót. Các bước feature engineering bao gồm:

4.2.1 Trích xuất danh xưng (Title)

Danh xưng của hành khách (ví dụ: Mr, Mrs, Miss, Master) được trích xuất từ cột Name vì nó cung cấp thông tin về giới tính, tình trạng hôn nhân, và địa vị xã hội. Mặc dù đoạn mã cụ thể để tạo Title không được hiển thị trong notebook, chúng tôi giả định rằng quá trình này sử dụng biểu thức chính quy hoặc các phương pháp xử lý chuỗi trong pandas. Sau khi trích xuất, cột Title được mã hóa bằng OneHotEncoder, tạo ra các cột nhị phân như Title_Mr, Title_Mrs, v.v.

Đặc trưng Title đặc biệt quan trọng vì nó có thể phản ánh các yếu tố như độ tuổi (Master thường là trẻ em nam) hoặc địa vị xã hội (Dr, Rev), vốn có liên quan đến khả năng sống sót.

4.2.2 Phân nhóm tuổi (AgeGroup)

Thay vì sử dụng giá trị Age liên tục, chúng tôi phân loại tuổi thành các nhóm như trẻ em, thanh niên, người lớn, và người già. Việc phân nhóm này giúp giảm nhiễu từ các giá trị tuổi cụ thể và làm nổi bật các xu hướng liên quan đến độ tuổi. Ví dụ, trẻ em thường có tỷ lệ sống sót cao hơn do được ưu tiên trong quá trình cứu hộ. Mặc dù đoạn mã tạo AgeGroup không được cung cấp, chúng tôi giả định rằng các ngưỡng được xác định dựa trên phân tích phân phối tuổi, chẳng hạn:

- Trẻ em: $0 \leq \text{Age} < 16$
- Thanh niên: $16 \leq \text{Age} < 30$
- Người lớn: $30 \leq \text{Age} < 60$
- Người già: $\text{Age} \geq 60$

Cột AgeGroup sau đó được mã hóa bằng OneHotEncoder, tạo ra các cột như AgeGroup_Child, AgeGroup_Adult, v.v.

4.2.3 Phân nhóm giá vé (FareGroup)

Tương tự, giá vé (Fare) được phân loại thành các nhóm để phản ánh tình trạng kinh tế của hành khách. Hành khách trả giá vé cao thường ở khoang hạng nhất, có tỷ lệ sống sót cao hơn. Chúng tôi giả định rằng các nhóm giá vé được tạo dựa trên các khoảng phân vị (quantile) của Fare, chẳng hạn:

- Thấp: $0 \leq \text{Fare} < 10$
- Trung bình: $10 \leq \text{Fare} < 30$
- Cao: $\text{Fare} \geq 30$

Cột FareGroup cũng được mã hóa bằng OneHotEncoder, tạo ra các cột như FareGroup_Low, FareGroup_Medium, v.v.

4.2.4 Loại bỏ các đặc trưng không cần thiết

Các cột không mang giá trị dự đoán hoặc có tỷ lệ giá trị thiếu cao được loại bỏ khỏi tập dữ liệu, bao gồm:

- PassengerId: Chỉ là định danh, không có ý nghĩa dự đoán.

- Name: Sau khi trích xuất Title, cột này không còn giá trị.
- Ticket: Chứa các giá trị không đồng nhất, khó khai thác.
- Cabin: Có tỷ lệ giá trị thiếu quá cao (khoảng 77%).

Việc loại bỏ các cột này giúp giảm chiều dữ liệu, tăng hiệu quả tính toán và tập trung vào các đặc trưng quan trọng.

4.2.5 Kết quả của Feature Engineering

Kết quả của quá trình feature engineering là một tập dữ liệu với các đặc trưng mới (Title, AgeGroup, FareGroup) được mã hóa và tối ưu hóa. Các đặc trưng này không chỉ làm nổi bật các yếu tố quan trọng như địa vị xã hội, độ tuổi, và tình trạng kinh tế, mà còn giúp mô hình học máy dễ dàng nhận diện các mẫu liên quan đến khả năng sống sót.

Đặc trưng	Mô tả
Title	Danh xưng trích xuất từ Name (Mr, Mrs, Miss, v.v.)
AgeGroup	Nhóm tuổi (Trẻ em, Thanh niên, Người lớn, Người già)
FareGroup	Nhóm giá vé (Thấp, Trung bình, Cao)
Sex	Giới tính (male, female)
Embarked	Cảng lên tàu (C, Q, S)

5 Giới thiệu các mô hình dùng để dự đoán người sống sót

Trong bài toán dự đoán khả năng sống sót của hành khách trên tàu Titanic, nhiều mô hình học máy khác nhau đã được áp dụng nhằm tìm kiếm mối quan hệ giữa các đặc trưng đầu vào và nhãn đầu ra (`Survived`). Các mô hình được lựa chọn nhằm đa dạng hóa cách tiếp cận từ tuyến tính, phi tuyến, boosting cho đến học sâu. Chi tiết các mô hình như sau:

5.1 Logistic Regression

Logistic Regression là một mô hình phân loại tuyến tính, thường được sử dụng cho bài toán phân loại nhị phân. Mô hình ước lượng xác suất sự kiện xảy ra dựa trên hàm sigmoid:

$$P(y=1|X) = \sigma(\beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n)}}$$

Hàm mất mát (loss function) được sử dụng là Binary Cross-Entropy:

$$\mathcal{L} = -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}) \right]$$

Logistic Regression có ưu điểm đơn giản, dễ huấn luyện, dễ diễn giải và là một baseline hiệu quả cho các bài toán phân loại nhị phân.

5.2 Random Forest

Random Forest là mô hình ensemble kết hợp nhiều Decision Trees thông qua kỹ thuật bagging (Bootstrap Aggregating). Mỗi cây được huấn luyện trên một tập dữ liệu ngẫu nhiên với thay thế, và quyết định cuối cùng được đưa ra dựa trên voting từ tất cả các cây. Mỗi cây được xây dựng bằng cách chọn ngẫu nhiên một tập con đặc trưng khi phân chia tại mỗi nút, giúp tăng tính đa dạng giữa các cây.

Giả sử ta có B cây $\{h_b(x)\}_{b=1}^{B}$, dự đoán tổng thể là:

$$\hat{y} = \text{majority\_vote}(h_1(x), h_2(x), \ldots, h_B(x))$$

Random Forest giúp giảm phương sai mô hình và hạn chế hiện tượng overfitting so với cây quyết định đơn lẻ.

5.3 Gradient Boosting

Gradient Boosting là kỹ thuật xây dựng mô hình mạnh bằng cách kết hợp tuần tự nhiều mô hình yếu. Mỗi mô hình mới học để giảm thiểu sai số còn lại (residual) của tổ hợp các mô hình trước.

Cho một hàm mất mát $\mathcal{L}(y, f(x))$, Gradient Boosting tối thiểu hóa nó bằng cách cộng dồn từng bước:

$$f_m(x) = f_{m-1}(x) + \gamma_m h_m(x)$$

trong đó $h_m(x)$ là mô hình yếu ở bước thứ m, và γ_m là hệ số bước tối ưu hóa. Mô hình này có khả năng mô hình hóa các quan hệ phức tạp nhưng dễ overfitting nếu không kiểm soát cẩn thận.

5.4 Support Vector Classifier (SVC)

Support Vector Classifier tìm kiếm siêu phẳng phân tách tối ưu giữa các lớp, tối đa hóa khoảng cách (margin) giữa hai lớp. Với dữ liệu không tuyến tính, SVC sử dụng các kernel như RBF để ánh xạ dữ liệu vào không gian cao chiều.

Bài toán tối ưu SVC:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{subject to} \quad y_i(\mathbf{w}^T\phi(x_i) + b) \geq 1$$

trong đó $\phi(x_i)$ là ánh xạ vào không gian đặc trưng.

Với trường hợp dữ liệu không hoàn toàn phân tách được, bài toán trở thành Soft-margin SVM:

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \quad \text{subject to} \quad y_i(\mathbf{w}^T\phi(x_i) + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

trong đó C điều chỉnh giữa margin rộng và lỗi phân loại.

5.5 Multi-layer Perceptron (MLP)

MLP là một dạng mạng nơ-ron sâu với ít nhất một lớp ẩn, áp dụng các phép biến đổi tuyến tính và phi tuyến liên tiếp để học các biểu diễn phức tạp.

Cho một input x, MLP tính toán:

$$h^{(1)} = \sigma(W^{(1)}x + b^{(1)})$$

$$h^{(2)} = \sigma(W^{(2)}h^{(1)} + b^{(2)})$$

$$\vdots$$

$$\hat{y} = \text{softmax}(W^{(L)} h^{(L-1)} + b^{(L)})$$

với σ là hàm kích hoạt (ví dụ: ReLU, tanh).

MLP có khả năng học các quan hệ phi tuyến rất mạnh, nhưng cũng dễ bị overfitting và yêu cầu điều chỉnh hyperparameter cẩn thận.

5.6 Extreme Gradient Boosting (XGBoost)

XGBoost là một phương pháp tối ưu hóa Gradient Boosting, sử dụng regularization để kiểm soát độ phức tạp của mô hình và tăng hiệu suất.

Hàm mục tiêu cần tối thiểu hóa trong XGBoost là:

$$\mathcal{L} = \sum_i l(y_i, \hat{y}_i) + \sum_k \Omega(f_k)$$

với:

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2$$

trong đó:

- T là số lượng nút lá
- w là vector trọng số nút lá
- γ, λ là hệ số điều chỉnh complexity.

XGBoost hỗ trợ xử lý missing value tự động, tree pruning, và kỹ thuật Column Block giúp huấn luyện cực kỳ nhanh.

5.7 Light Gradient Boosting Machine (LightGBM)

LightGBM là một thuật toán boosting được tối ưu cho tốc độ huấn luyện và khả năng xử lý dữ liệu lớn.

Khác với XGBoost, LightGBM sử dụng chiến lược Leaf-wise Tree Growth thay vì Level-wise. Tại mỗi bước, nó chọn nhánh có mức giảm loss lớn nhất để phát triển, giúp đạt độ chính xác cao hơn:

$$\text{loss decrease} = \text{gain of best split}$$

LightGBM còn hỗ trợ kỹ thuật histogram-based split, giảm độ phức tạp tính toán từ $O(\#data)$ xuống $O(\#bins)$, làm tăng tốc độ huấn luyện đáng kể.

5.8 Voting Classifier (Ensemble)

Voting Classifier là kỹ thuật ensemble kết hợp nhiều mô hình học máy khác nhau. Trong bài toán này, Voting Classifier được xây dựng từ các mô hình Logistic Regression, Random Forest, Gradient Boosting, SVC, MLP, XGBoost và LightGBM.

Hai phương pháp voting được sử dụng:

- **Hard Voting**: chọn nhãn được dự đoán nhiều nhất bởi các mô hình thành phần.
- **Soft Voting**: tính trung bình xác suất dự đoán và chọn nhãn có xác suất cao nhất.

Soft Voting có thể tận dụng tốt hơn thông tin xác suất từ các mô hình mạnh như XGBoost, LightGBM, MLP.

Tổng thể, việc sử dụng đa dạng các mô hình từ tuyến tính đến phi tuyến, từ boosting đến học sâu, giúp tăng cường độ ổn định và khả năng tổng quát hóa của hệ thống dự đoán sống sót trên Titanic dataset.

6 Kết quả từ mô hình Logistic Regression

6.1 Quy trình đánh giá

Để đánh giá hiệu quả của mô hình Logistic Regression trên bài toán dự đoán sự sống sót của hành khách Titanic, chúng tôi áp dụng phương pháp **5-fold Cross-Validation**. Quy trình cụ thể như sau:

- Tập dữ liệu được chia thành 5 phần gần bằng nhau.
- Trong mỗi vòng lặp, 4 phần dữ liệu được sử dụng để huấn luyện mô hình Logistic Regression, phần còn lại dùng để kiểm tra hiệu suất mô hình.
- Quá trình này được lặp lại 5 lần, mỗi phần dữ liệu được làm tập kiểm tra đúng một lần.
- Cuối cùng, các độ chính xác (accuracy) thu được từ từng vòng được tổng hợp để tính trung bình, nhằm đánh giá độ ổn định và khả năng tổng quát của mô hình.

```
# Instantiate the model
lr_model = LogisticRegression()

# Perform 5-fold cross-validation
scores = cross_val_score(lr_model, X, y, cv=5)

# Print the cross-validation scores
print("Cross-validation scores:", scores)
print("Mean cross-validation score:", scores.mean())
```

6.2 Kết quả đạt được

Các độ chính xác thu được từ 5 lần cross-validation là:

- Fold 1: 81%
- Fold 2: 78%
- Fold 3: 82%
- Fold 4: 79%
- Fold 5: 80%

Tính trung bình, mô hình Logistic Regression đạt độ chính xác khoảng:

$$\text{Accuracy}_{\text{mean}} = 0.80$$

tức là khoảng **80%**.

6.3 Phân tích kết quả

Kết quả trên cho thấy Logistic Regression là một mô hình có khả năng phân loại khá ổn định trên bài toán Titanic. Một số nhận xét quan trọng:

- Độ chính xác dao động nhỏ giữa các folds (từ 78% đến 82%), chứng tỏ mô hình có tính ổn định tốt và không quá phụ thuộc vào từng tập con dữ liệu.
- Mức độ chính xác 80% là khá ấn tượng đối với một mô hình tuyến tính cơ bản, đặc biệt khi dữ liệu Titanic chứa nhiều đặc trưng dạng phân loại và mối quan hệ phi tuyến nhẹ.
- Tuy nhiên, Logistic Regression chỉ nắm bắt được mối quan hệ tuyến tính giữa các đặc trưng và xác suất sống sót. Điều này hạn chế khả năng khai thác các tương tác phức tạp hơn trong dữ liệu.

Do đó, dù Logistic Regression mang lại baseline khá tốt, chúng tôi tiếp tục thử nghiệm với các mô hình phi tuyến tính mạnh mẽ hơn như Random Forest, Gradient Boosting, SVC, MLP, XGBoost và LightGBM nhằm tìm kiếm mô hình có hiệu suất cao hơn trên bài toán này.

7 Hyperparameter tuning

7.1 Phương pháp tối ưu siêu tham số

Để tìm ra bộ siêu tham số tối ưu cho từng mô hình, chúng tôi sử dụng phương pháp **Bayesian Optimization** thông qua `BayesSearchCV` của thư viện `scikit-optimize`. Đây là một phương pháp tiên tiến, cho phép mô hình hóa quá trình tối ưu hóa như một bài toán Bayes, tận dụng thông tin từ các lần thử trước để chọn ra những tham số hứa hẹn nhất trong các lần thử sau.

Khác với các phương pháp tìm kiếm truyền thống như **Grid Search** hay **Random Search**, Bayesian Optimization tìm kiếm hiệu quả hơn trong không gian tham số bằng cách cân bằng giữa khai thác (exploration) và khai phá (exploitation), từ đó tiết kiệm đáng kể thời gian huấn luyện mà vẫn tìm ra những tham số tốt.

7.2 Không gian tham số tối ưu hóa

Không gian tham số được định nghĩa riêng cho từng mô hình như sau:

- **Logistic Regression**:
  - C $\sim$ Log-Uniform(1e-3, 1e3)
- **Random Forest Classifier**:
  - n_estimators $\sim$ Integer(50, 500)
  - max_depth $\sim$ Integer(2, 50)
- **Gradient Boosting Classifier**:
  - n_estimators $\sim$ Integer(50, 500)
  - learning_rate $\sim$ Log-Uniform(0.01, 0.5)
  - max_depth $\sim$ Integer(2, 20)
- **Support Vector Classifier (SVC)**:

  - C ∼ Log-Uniform(1e-3, 1e3)
  - kernel ∼ Categorical(['linear', 'rbf'])

- **Multi-Layer Perceptron (MLP) Classifier**:
  - hidden_layer_sizes ∼ Integer(10, 200)
  - alpha ∼ Log-Uniform(1e-5, 1e-1)

- **XGBoost Classifier**:
  - n_estimators ∼ Integer(50, 500)
  - learning_rate ∼ Log-Uniform(0.01, 0.5)
  - max_depth ∼ Integer(2, 20)

- **LightGBM Classifier**:
  - n_estimators ∼ Integer(50, 500)
  - learning_rate ∼ Log-Uniform(0.01, 0.5)
  - num_leaves ∼ Integer(10, 100)

7.3 Kết quả sau quá trình Hyperparameter Tuning

Sau khi thực hiện tối ưu hóa siêu tham số cho từng mô hình bằng phương pháp Bayesian Optimization, chúng tôi tiến hành đánh giá hiệu suất của các mô hình trên tập kiểm thử thông qua độ chính xác (Accuracy). Các kết quả thu được được tổng hợp trong bảng dưới đây:

Mô hình	**Accuracy (%)**
Logistic Regression	82.15
Random Forest Classifier	83.39
Gradient Boosting Classifier	83.84
Support Vector Classifier (SVC)	82.83
Multi-Layer Perceptron (MLP) Classifier	81.71
XGBoost Classifier	84.40
LightGBM Classifier	84.29

Dựa trên bảng tổng hợp:

- **Mô hình đạt hiệu suất cao nhất** là **XGBoost Classifier** với độ chính xác 84.40%. LightGBM Classifier xếp ngay sau với độ chính xác 84.29%.
- **Mô hình có hiệu suất thấp nhất** là **MLP Classifier** với độ chính xác 81.71%, thấp hơn so với Logistic Regression (82.15%).

7.4 Giải thích kết quả

XGBoost và **LightGBM** đều là các mô hình Boosting mạnh mẽ, có khả năng tổng hợp nhiều cây quyết định nhỏ để giảm bias và variance, đồng thời xử lý tốt các mối quan hệ phi tuyến tính phức tạp trong dữ liệu. Với khả năng regularization mạnh (qua tham số learning rate, depth control, và số lượng estimator), các mô hình này thường có ưu thế vượt trội trong các bài toán phân loại structured data như bài toán hiện tại.

Ngược lại, **MLP Classifier** lại không hoạt động tốt bằng, mặc dù đã được tối ưu hóa siêu tham số. Điều này có thể xuất phát từ một số nguyên nhân:

- Dữ liệu dạng structured tabular vốn không phải thế mạnh của các mô hình mạng nơ-ron sâu, vốn yêu cầu lượng dữ liệu rất lớn hoặc các đặc trưng có cấu trúc phức tạp (như ảnh, văn bản).
- Ngoài ra, mạng MLP có thể dễ dàng overfit dữ liệu nhỏ nếu không có kỹ thuật regularization đủ mạnh.

7.5 Mô hình tổng hợp: Voting Classifier

Sau khi xây dựng và huấn luyện các mô hình đơn lẻ như `Logistic Regression`, `Random Forest`, `Gradient Boosting`, `SVC`, `MLP`, `XGBoost`, và `LightGBM`, nhóm tiến hành áp dụng kỹ thuật tổng hợp mô hình thông qua **Voting Classifier** nhằm khai thác sức mạnh của từng mô hình và cải thiện độ chính xác tổng thể.

Voting Classifier là một kỹ thuật trong học máy tổng hợp (ensemble learning), nơi mà dự đoán cuối cùng được đưa ra dựa trên kết quả biểu quyết từ nhiều mô hình con. Có hai hình thức voting chính:

- **Hard voting**: mỗi mô hình đưa ra một dự đoán phân lớp (label), và lớp nào có nhiều phiếu nhất sẽ là kết quả cuối cùng.
- **Soft voting**: mỗi mô hình đưa ra xác suất dự đoán cho mỗi lớp, và kết quả cuối cùng được xác định bằng cách lấy trung bình có trọng số các xác suất này.

Trong bài toán này, nhóm lựa chọn sử dụng hình thức **soft voting**, vì các mô hình như `Gradient Boosting`, `XGBoost`, và `LightGBM` có khả năng ước lượng xác suất tốt. Việc này giúp mô hình tổng hợp không chỉ dựa vào biểu quyết "đa số" mà còn tận dụng thông tin xác suất từ mỗi mô hình thành phần, làm cho kết quả cuối cùng trở nên linh hoạt và chính xác hơn.

Việc sử dụng `VotingClassifier` trong thư viện `scikit-learn` giúp dễ dàng tích hợp các mô hình đã được huấn luyện vào một hệ thống dự đoán thống nhất. Trong quá trình thực nghiệm, mô hình tổng hợp này đã cho thấy hiệu suất khá ổn định, đặc biệt trong việc giảm phương sai và cải thiện độ chính xác tổng thể so với một số mô hình đơn lẻ, mặc dù độ chính xác không vượt qua các mô hình tối ưu như `XGBoost` hoặc `LightGBM` sau khi được tinh chỉnh tham số.

Việc áp dụng Voting Classifier thể hiện tư duy tích hợp trong mô hình hóa – thay vì chỉ tìm kiếm "một mô hình tốt nhất", việc kết hợp nhiều mô hình có thể khai thác thế mạnh của từng phương pháp, từ đó đưa ra một hệ thống có hiệu suất cao và ổn định hơn trong các kịch bản dữ liệu thực tế.

7.6 Nhận xét cá nhân

Từ kết quả trên, có thể rút ra một số quan sát đáng chú ý:

- Trong các bài toán dữ liệu dạng bảng (tabular data), các mô hình Boosting như `XGBoost` và `LightGBM` tiếp tục khẳng định vị thế với hiệu suất vượt trội. Điều này phù hợp với nhiều nghiên cứu thực nghiệm gần đây, cho thấy rằng Boosting rất hiệu quả trong việc học các mẫu dữ liệu phức tạp mà vẫn tránh được hiện tượng overfitting nhờ cơ chế học tuần tự và tinh chỉnh lỗi.
- Mặc dù đơn giản và tuyến tính, `Logistic Regression` vẫn đạt được độ chính xác khá tốt (82.15%), cho thấy rằng dữ liệu chứa nhiều thông tin có thể phân tách tốt theo biên tuyến tính hoặc gần tuyến tính. Mô hình này cũng có lợi thế lớn về tốc độ huấn luyện và khả năng giải thích, rất phù hợp với các ứng dụng đòi hỏi sự minh bạch trong quyết định.

- Việc áp dụng kỹ thuật **tối ưu hóa siêu tham số** bằng `Bayesian Optimization` đã chứng minh được hiệu quả rõ rệt, giúp nâng cao hiệu suất cho hầu hết các mô hình, đồng thời giảm được chi phí tính toán so với các phương pháp dò lưới truyền thống.
- Đặc biệt, mô hình tổng hợp `Voting Classifier` thể hiện một hướng tiếp cận thông minh bằng cách kết hợp sức mạnh của nhiều mô hình đã được huấn luyện. Dù độ chính xác không vượt trội hoàn toàn so với `XGBoost`, mô hình này mang lại sự ổn định và khả năng khái quát tốt hơn khi áp dụng trên dữ liệu chưa từng thấy. Việc sử dụng soft voting cho phép tận dụng các xác suất dự đoán của từng mô hình, từ đó tạo ra quyết định cuối cùng hợp lý và ít bị ảnh hưởng bởi sự bất định từ bất kỳ mô hình đơn lẻ nào.

Cuối cùng, việc lựa chọn mô hình phù hợp không chỉ nên dựa trên độ chính xác tuyệt đối, mà còn cần cân nhắc đến nhiều yếu tố khác như thời gian huấn luyện, độ phức tạp của mô hình, khả năng diễn giải và tính ổn định khi triển khai thực tế. Trong một hệ thống thực thi thật sự, việc kết hợp nhiều mô hình (ensemble) có thể là chiến lược tối ưu, vừa khai thác được điểm mạnh riêng của từng phương pháp, vừa đảm bảo hiệu quả tổng thể trong môi trường biến động và dữ liệu không hoàn hảo.

8 Tổng kết

Trong suốt quá trình thực hiện dự án dự đoán khả năng sống sót của hành khách trên con tàu Titanic, nhóm đã áp dụng toàn bộ quy trình phân tích dữ liệu hiện đại – từ tiền xử lý, xây dựng đặc trưng, lựa chọn và huấn luyện mô hình, đến đánh giá hiệu quả và tối ưu hóa siêu tham số. Mỗi giai đoạn đều đóng vai trò thiết yếu, không chỉ nâng cao độ chính xác của mô hình mà còn giúp người thực hiện hiểu rõ bản chất và ý nghĩa đằng sau dữ liệu.

Trước hết, khâu tiền xử lý và làm sạch dữ liệu là một bước không thể thiếu. Dữ liệu Titanic ban đầu chứa nhiều giá trị thiếu ở các cột như `Age`, `Cabin`, và `Embarked`. Việc thay thế các giá trị này bằng trung bình, mode hoặc gán nhóm hợp lý giúp đảm bảo tính toàn vẹn mà không làm mất đi thông tin quan trọng. Bên cạnh đó, các biến phân loại cũng được xử lý cẩn thận bằng mã hóa one-hot hoặc label encoding để phục vụ cho các mô hình học máy. Toàn bộ quá trình xử lý dữ liệu không chỉ nhằm chuẩn hóa mà còn để tạo ra đầu vào nhất quán, hỗ trợ tối ưu hóa hiệu suất của mô hình phía sau.

Giai đoạn kế tiếp – *feature engineering* – đóng vai trò then chốt trong việc khai thác các đặc trưng tiềm ẩn từ dữ liệu gốc. Những đặc trưng như danh xưng (`Title`) được trích xuất từ tên hành khách, số lượng người đi cùng (`FamilySize`), hay vị trí khoang tàu (`Deck`) được phân loại từ mã cabin đã mang lại các thông tin bổ sung quý giá cho mô hình. Đây là minh chứng rõ ràng cho thấy, trong nhiều trường hợp, việc thiết kế đặc trưng tốt có thể mang lại hiệu quả không kém so với việc sử dụng mô hình phức tạp.

Về phần xây dựng mô hình, nhóm đã thử nghiệm nhiều thuật toán từ đơn giản đến nâng cao như **Logistic Regression**, **Random Forest**, **Gradient Boosting**, **Support Vector Machine (SVC)**, **Multi-layer Perceptron (MLP)**, **XGBoost** và **LightGBM**. Mỗi mô hình mang trong mình một triết lý học khác nhau, từ tuyến tính đến cây quyết định và mạng nơ-ron. Thông qua quá trình đánh giá chéo (cross-validation) và đo lường độ chính xác (accuracy), Logistic Regression – dù đơn giản – vẫn đạt độ chính xác khá tốt nhờ các đặc trưng mạnh mẽ. Tuy nhiên, các mô hình học tăng cường như XGBoost và LightGBM sau khi được tinh chỉnh tham số đã vượt trội hơn hẳn với độ chính xác lần lượt đạt **84.40%** và **84.29%**.

Đặc biệt, quá trình *tối ưu siêu tham số* bằng chiến lược tìm kiếm Bayes (Bayesian Optimization) cho phép mô hình tìm được tập tham số tốt nhất mà không phải tốn quá nhiều tài nguyên như Grid Search. Các không gian tìm kiếm được thiết kế hợp lý, tận dụng các phân phối

log-uniform hoặc rời rạc phù hợp với từng loại tham số. Sau tối ưu, mô hình XGBoost không chỉ có hiệu suất tốt nhất mà còn ổn định trên nhiều lần đánh giá chéo, chứng minh khả năng tổng quát hóa tốt.

Tuy nhiên, không phải mô hình nào phức tạp cũng đều mang lại hiệu quả vượt trội. MLP, một dạng mạng nơ-ron truyền thống, lại cho kết quả khiêm tốn nhất (accuracy **81.71%**), có thể do số lượng mẫu không đủ lớn để phát huy hết khả năng của mô hình sâu. Điều này nhấn mạnh bài học rằng lựa chọn mô hình phù hợp với quy mô và tính chất của dữ liệu là yếu tố sống còn.

Nhìn lại toàn bộ hành trình, có thể khẳng định rằng thành công không đến từ một thuật toán đơn lẻ mà là kết quả của sự kết hợp giữa hiểu dữ liệu, xử lý đặc trưng phù hợp, lựa chọn mô hình khéo léo và quá trình tinh chỉnh chỉn chu. Ngoài ra, việc sử dụng phương pháp ensemble như `VotingClassifier` – tổng hợp các mô hình đã huấn luyện – cũng mở ra khả năng nâng cao hiệu suất bằng cách tận dụng sức mạnh từ nhiều mô hình khác nhau.

Trong tương lai, dự án có thể được mở rộng theo nhiều hướng:

- Sử dụng các phương pháp giải thích mô hình như SHAP hoặc LIME để phân tích tác động của từng đặc trưng lên kết quả dự đoán.
- Tăng cường dữ liệu bằng các kỹ thuật như SMOTE hoặc augmentation đối với các nhóm thiểu số để cải thiện độ cân bằng.
- Áp dụng học sâu nâng cao như AutoML hoặc mô hình attention-based để kiểm nghiệm thêm hướng tiếp cận.

Dự án Titanic có thể đơn giản về mặt dữ liệu, nhưng chính vì thế nó là một nền tảng lý tưởng để rèn luyện toàn diện kỹ năng phân tích, từ tư duy logic, kỹ thuật lập trình, đến chiến lược mô hình hóa và ra quyết định dựa trên dữ liệu.

Tài liệu tham khảo

[1] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). *Scikit-learn: Machine learning in Python*. Journal of Machine Learning Research, 12, 2825–2830.

[2] Friedman, J. H. (2001). *Greedy function approximation: A gradient boosting machine*. Annals of Statistics, 29(5), 1189–1232.

[3] Chen, T., & Guestrin, C. (2016). *XGBoost: A scalable tree boosting system*. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794).

[4] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... & Liu, T.-Y. (2017). *LightGBM: A highly efficient gradient boosting decision tree*. In *Advances in Neural Information Processing Systems* (pp. 3146–3154).

[5] Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

[6] Head, T., MechCoder, Louppe, G., Shcherbatyi, I., Bergstra, J., Yamins, D., ... & Eggensperger, K. (2018). *Scikit-Optimize: Sequential model-based optimization in Python*. https://scikit-optimize.github.io

[7] Rokach, L. (2010). *Ensemble-based classifiers*. Artificial Intelligence Review, 33(1), 1–39.

[8] Kaggle. *Titanic - Machine Learning from Disaster*. https://www.kaggle.com/competitions/titanic