

BÁO CÁO CHALLENGE 1 - TITANIC

TỐI ƯU HÓA MÔ HÌNH DỰ ĐOÁN TỶ LỆ
SỐNG SÓT TITANIC THÔNG QUA KỸ THUẬT
FEATURE ENGINEERING LẮP LẠI

Trần Hồ Minh Hải, Trương Văn Thiện, Phan Đức Nhân, Võ Gia Kiệt

Agenda

1. Giới thiệu
2. Công việc liên quan
3. Phương pháp được đề xuất
4. Thực nghiệm và Thảo luận
5. Kết luận

1. Giới thiệu

Sơ lược bài toán

Thảm họa tàu Titanic năm 1912 là một trong những sự kiện hàng hải gây chấn động nhất trong lịch sử nhân loại, đồng thời cũng mở ra một hướng nghiên cứu đặc biệt trong lĩnh vực khoa học dữ liệu: dự đoán khả năng sống sót của hành khách dựa trên thông tin cá nhân và điều kiện đi tàu.



1. Giới thiệu

Bối cảnh bài toán:

- **Mục tiêu chính:** Xây dựng mô hình phân loại (Classification Model) để dự đoán khả năng sống sót (Survived) của hành khách trên tàu Titanic.
- **Bài toán dự đoán nhị phân (Binary Prediction):** đòi hỏi mô hình phải học được mối quan hệ phức tạp giữa các thuộc tính cá nhân và khả năng sống sót.
- **Dataset:** <https://www.kaggle.com/competitions/titanic/overview>

The screenshot shows the Kaggle competition page for 'Titanic - Machine Learning from Disaster'. At the top, there's a navigation bar with the Kaggle logo, the competition name, and links for 'Submit Prediction' and '...'. Below the navigation is a large image of the Titanic ship at night. The main title 'Titanic - Machine Learning from Disaster' is prominently displayed, followed by a subtitle 'Start here! Predict survival on the Titanic and get familiar with ML basics'. A horizontal menu bar below the title includes 'Overview', 'Data', 'Code', 'Models', 'Discussion', 'Leaderboard', 'Rules', 'Team', and 'Submissions', with 'Overview' being the active tab. On the left side, there's a sidebar with sections for 'Competition Host' (Kaggle), 'Prizes & Awards' (Knowledge, Does not award Points or Medals), and 'Participation'. A note at the bottom left states 'This competition runs indefinitely with a rolling leaderboard. Learn more'.

1. Giới thiệu

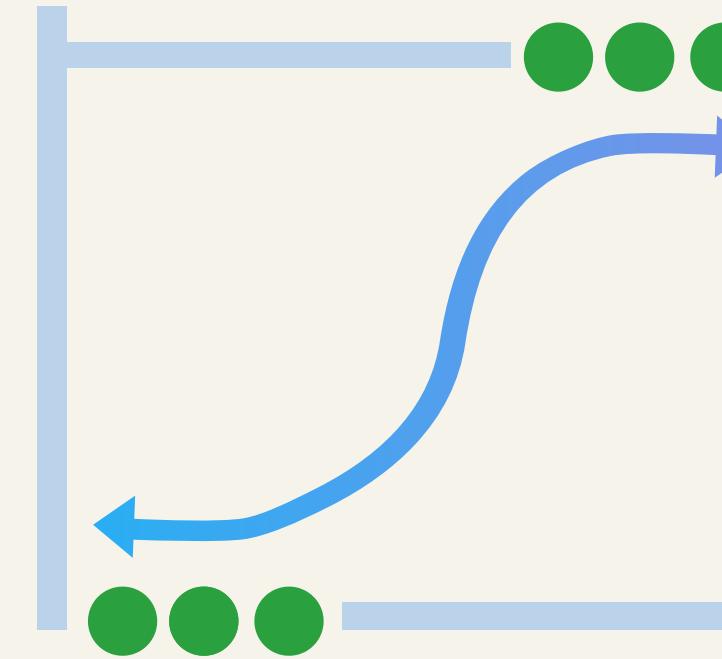
Dữ liệu bài toán:

- Bộ dữ liệu Titanic từ Kaggle (891 mẫu huấn luyện, 418 mẫu kiểm thử).

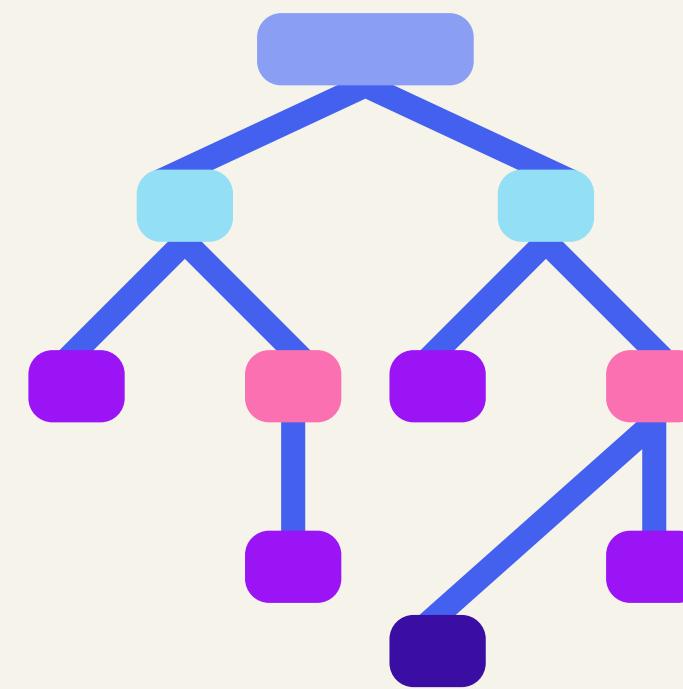
Tên cột (Feature)	Ý nghĩa (Tiếng Việt)	Giải thích / Ghi chú
PassengerId	Mã khách hàng	Mã định danh duy nhất của hành khách
Survived	Trạng thái sống	0 = Không sống sót, 1 = Sống sót
Pclass	Hạng vé	1 = Hạng nhất, 2 = Hạng hai, 3 = Hạng ba
Name	Họ tên khách hàng	Họ và tên đầy đủ của hành khách
Sex	Giới tính	Nam (male) hoặc Nữ (female)
Age	Độ tuổi	Tuổi của hành khách (có thể bị thiếu dữ liệu)
SibSp	Số lượng anh/chị/em hoặc vợ/chồng đi cùng	Siblings/Spouses Aboard
Parch	Số lượng cha/mẹ hoặc con đi cùng	Parents/Children Aboard
Ticket	Số vé	Mã số vé của hành khách
Fare	Giá vé khách trả	Tổng số tiền hành khách trả cho vé (đơn vị: bảng Anh)
Cabin	Số cabin	Mã số phòng / khoang trên tàu (thường bị thiếu nhiều)
Embarked	Nơi lên tàu	C = Cherbourg, Q = Queenstown, S = Southampton

2. Công việc liên quan

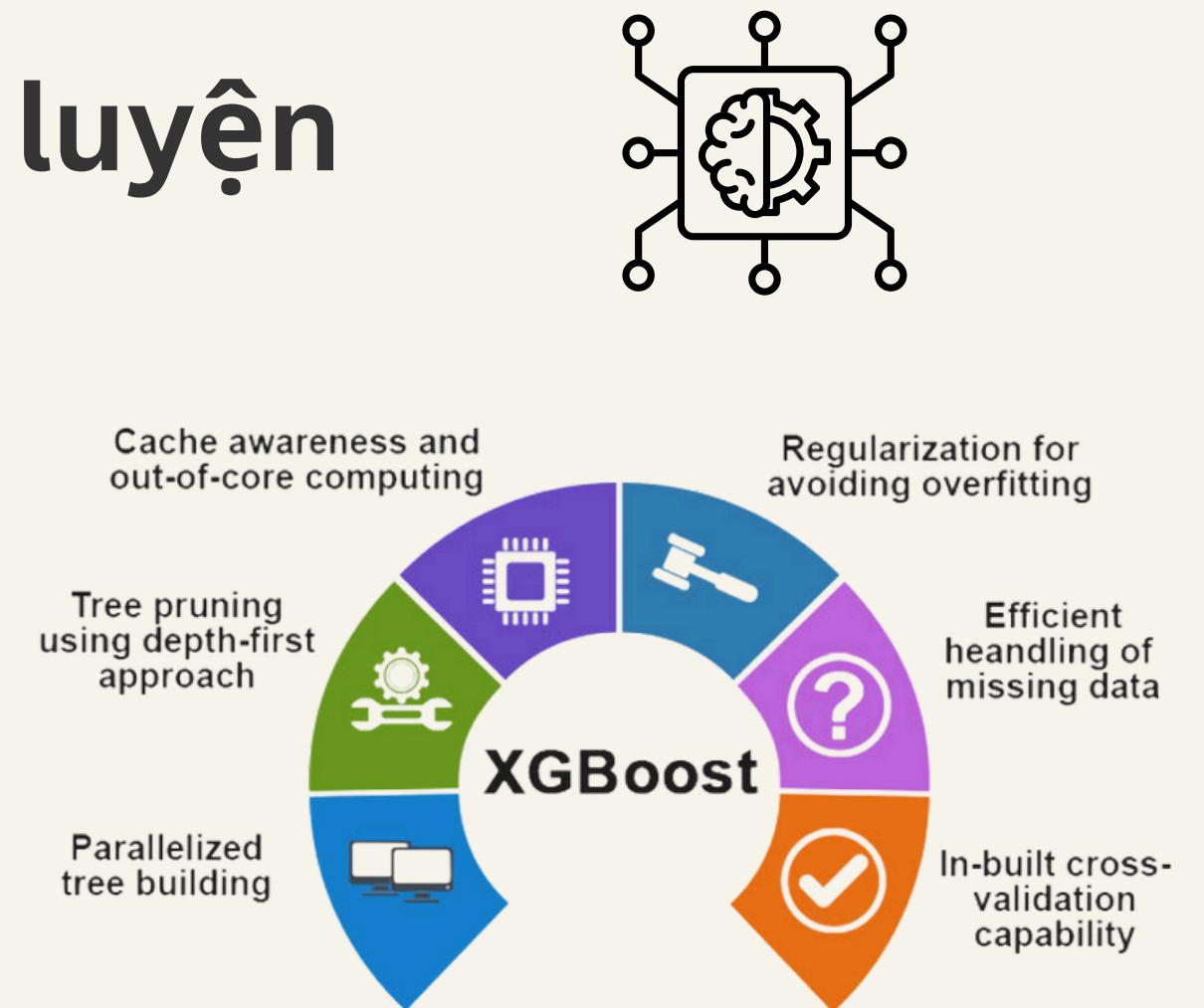
2.1 Các mô hình được huấn luyện



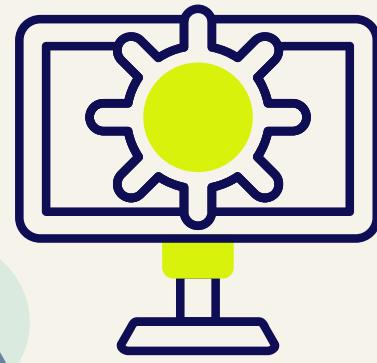
Logistic Regression



Random Forest

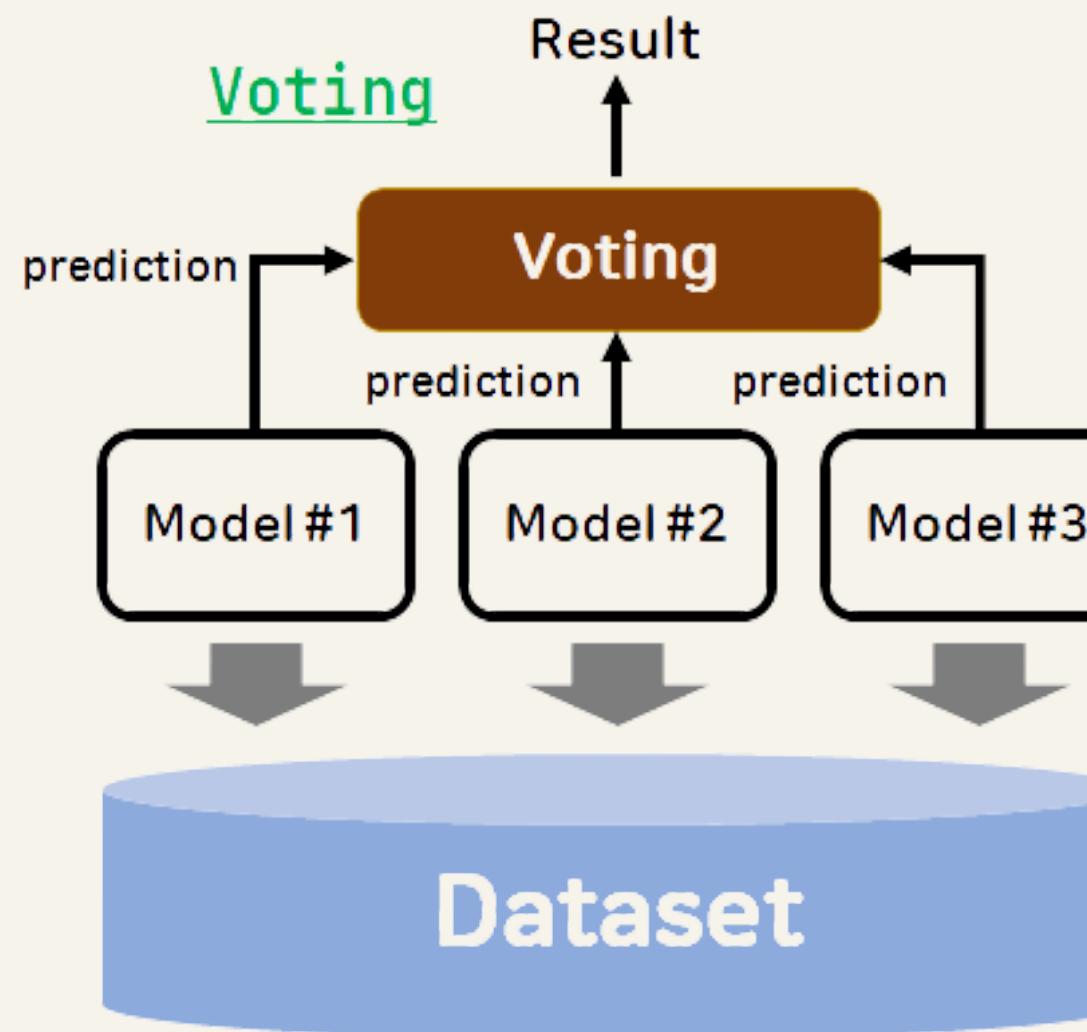
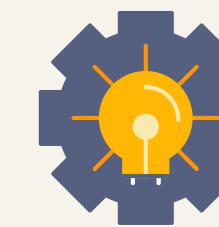


XGBoost

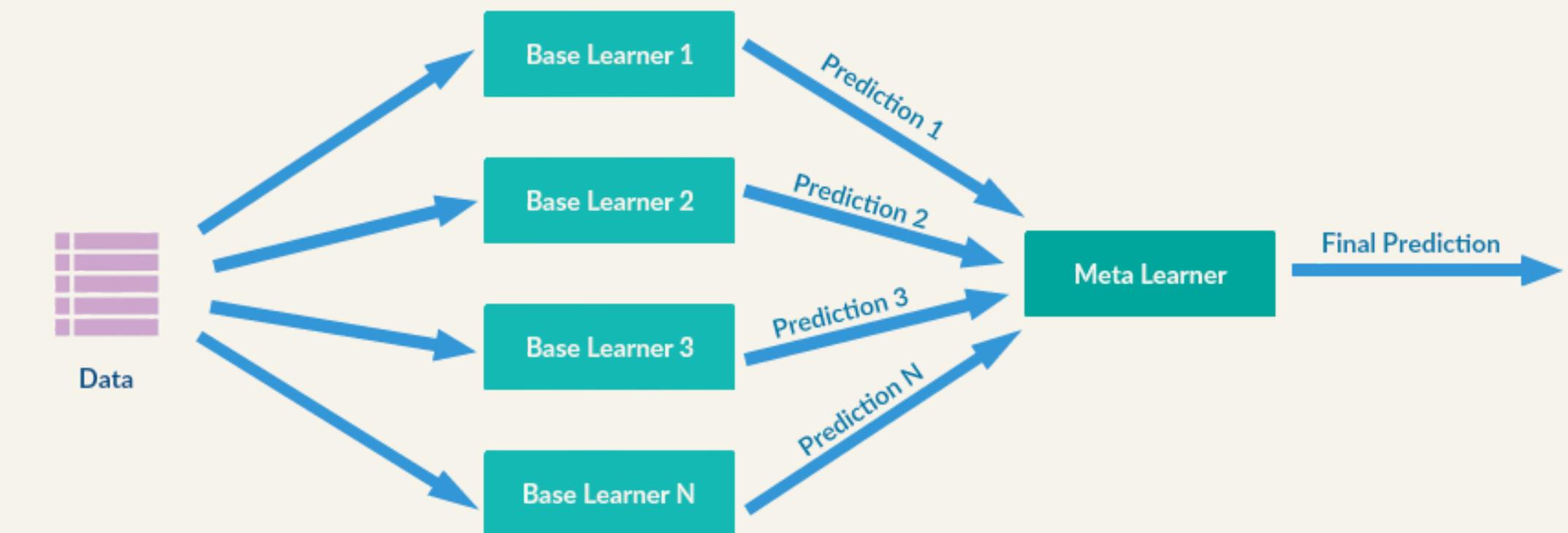


2. Công việc liên quan

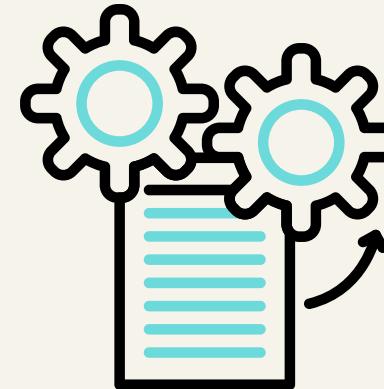
2.2 Ensemble mô hình



Voting

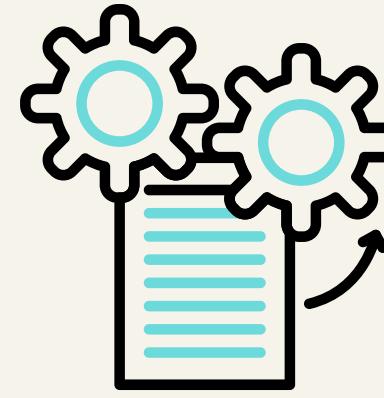
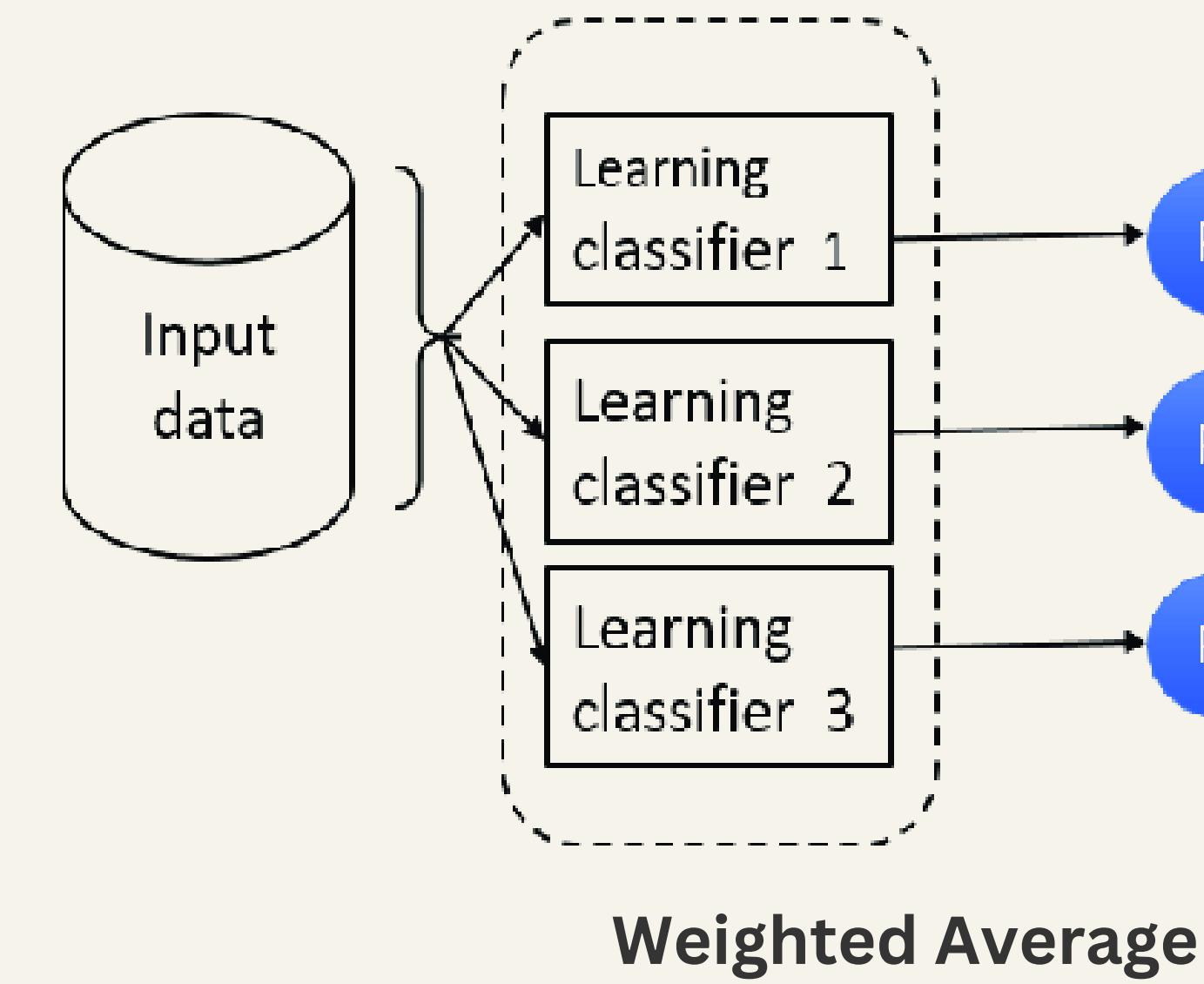
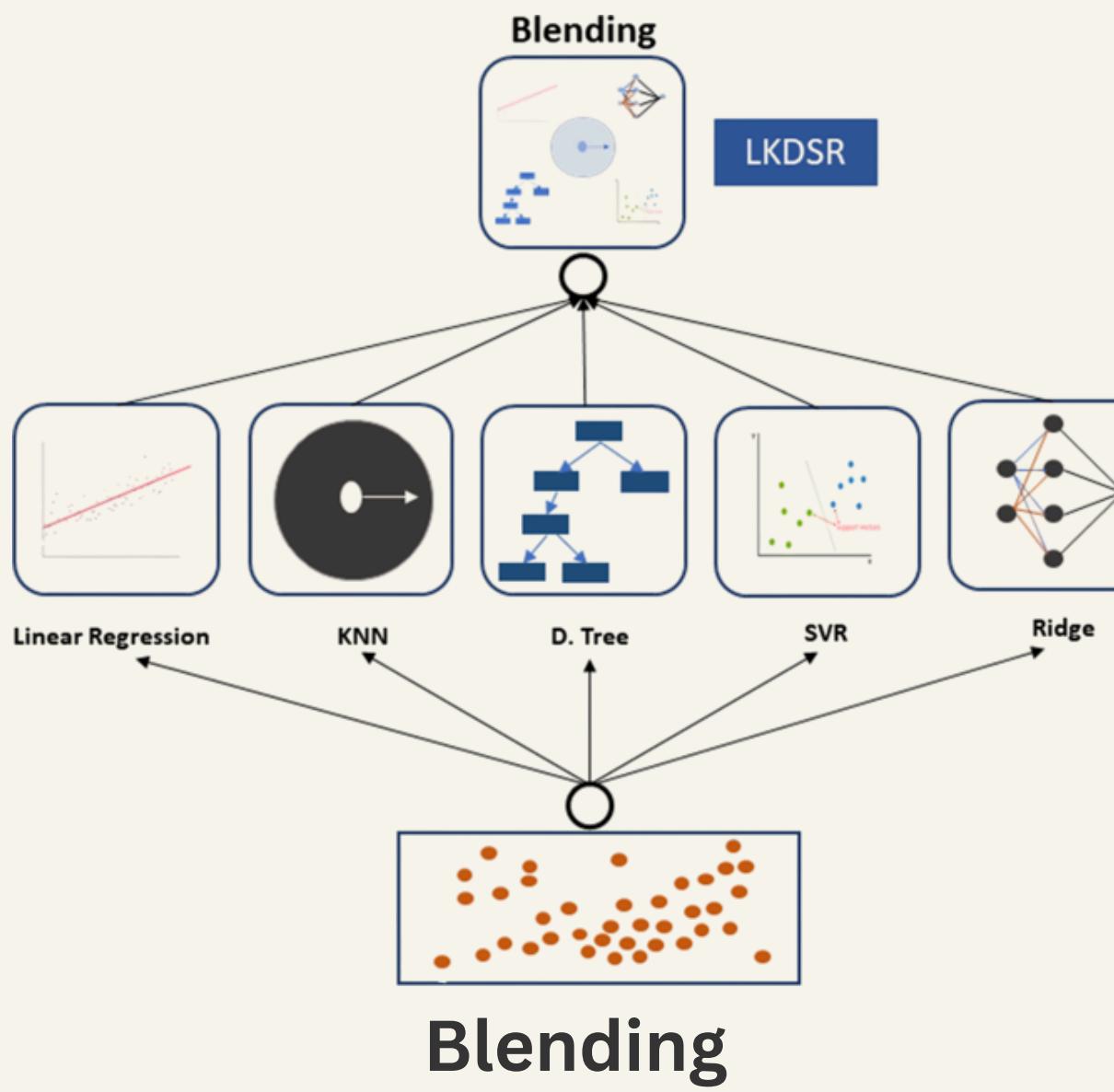
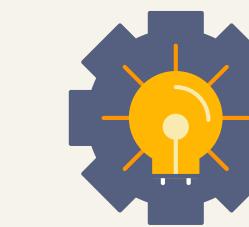


Stacking



2. Công việc liên quan

2.2 Ensemble mô hình



2. Công việc liên quan

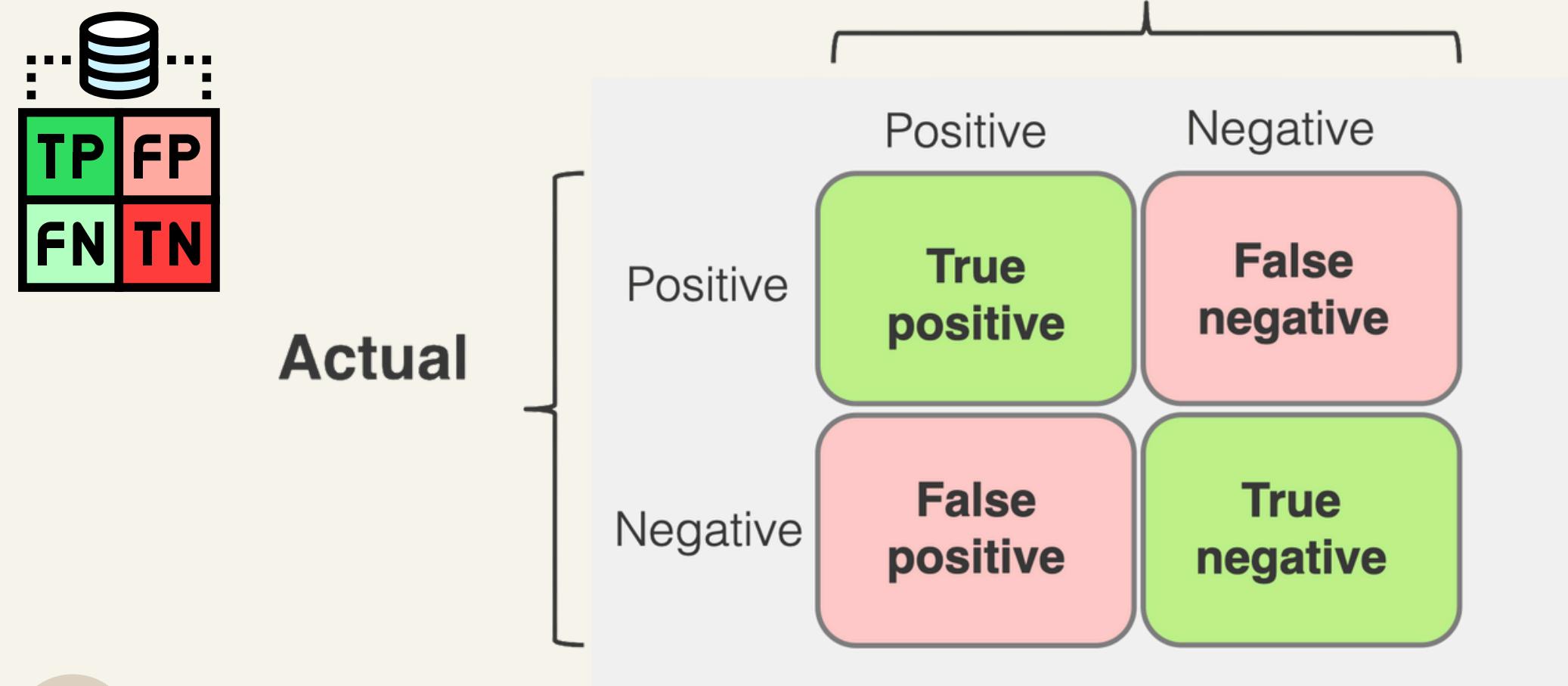
2.3 Các độ đo được sử dụng (Metrics)



Confusion Matrix

Confusion Matrix (ma trận nhầm lẫn) là công cụ dùng để **đánh giá hiệu quả mô hình phân loại**, thể hiện số lượng dự đoán đúng và sai cho từng lớp.

Nó giúp đo lường độ chính xác và mức độ sai lệch của mô hình trong quá trình dự đoán.



Giải thích:

- **TP:** Dự đoán đúng mẫu dương tính.
 - **TN:** Dự đoán đúng mẫu âm tính.
 - **FP:** Dự đoán sai – mô hình nhầm âm thành dương.
 - **FN:** Dự đoán sai – mô hình nhầm dương thành âm.

2. Công việc liên quan

2.3 Các độ đo được sử dụng (Metrics)



Accuracy

Accuracy là một trong những chỉ số phổ biến nhất được sử dụng để đo lường hiệu suất của các mô hình phân loại. Nó biểu thị **tỷ lệ phần trăm** các dự đoán đúng so với **tổng số mẫu dữ liệu được dự đoán**.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{số lượng mẫu}} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$



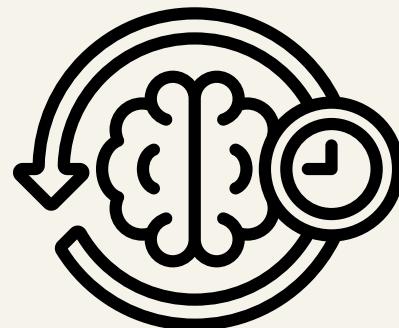
Precision

Precision (độ chính xác) là một chỉ số đánh giá hiệu suất của mô hình phân loại, tập trung vào độ tin cậy của các **dự đoán dương tính**.

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

2. Công việc liên quan

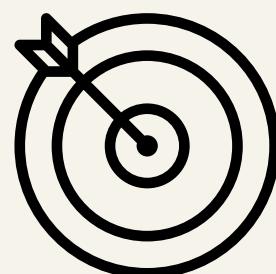
2.3 Các độ đo được sử dụng (Metrics)



Recall

Thể hiện khả năng mô hình nhận diện đúng mẫu dương tính, bằng tỷ lệ mẫu dương được dự đoán đúng trên tổng số mẫu dương thực tế.

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



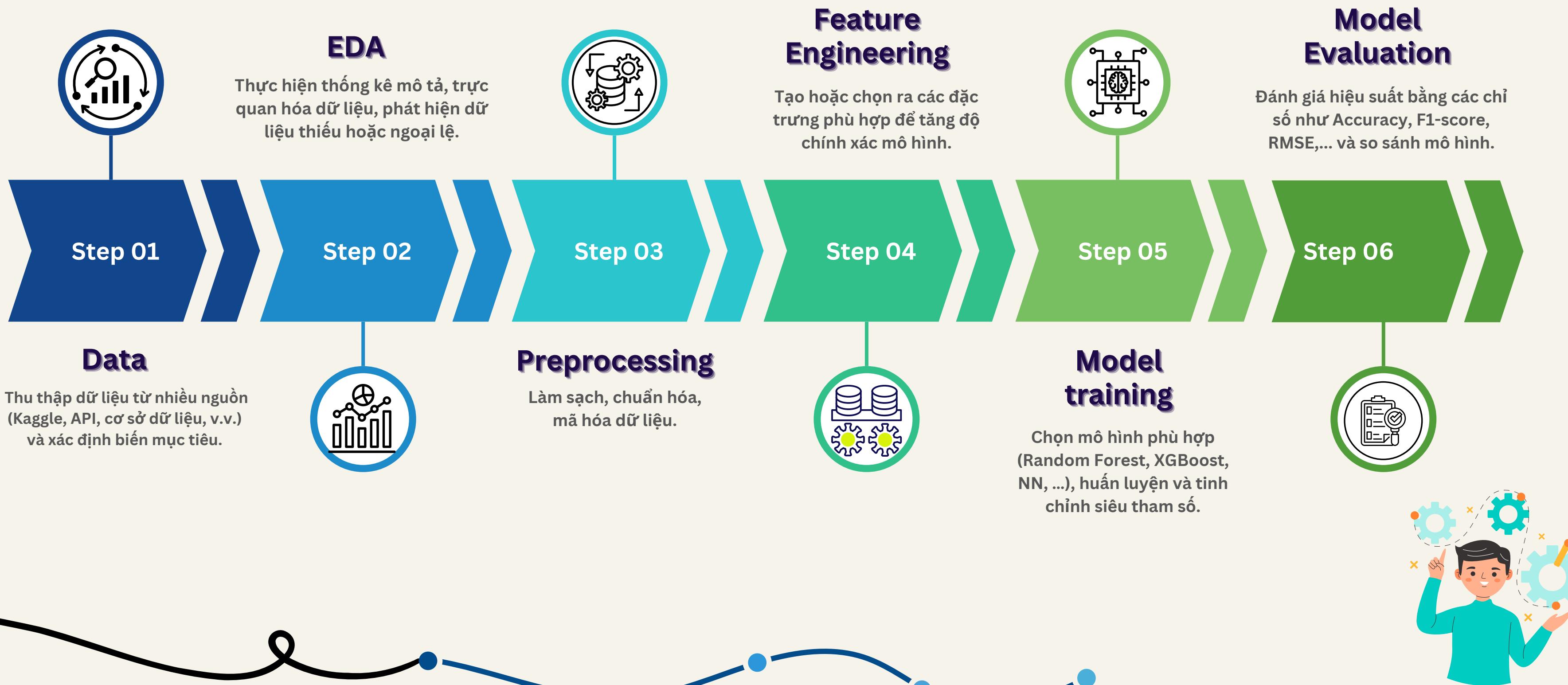
F1-score

Chỉ số kết hợp giữa Precision và Recall, phản ánh hiệu suất cân bằng của mô hình, đặc biệt hiệu quả với dữ liệu mất cân bằng.

$$F1 Score = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

3. Phương pháp được đề xuất

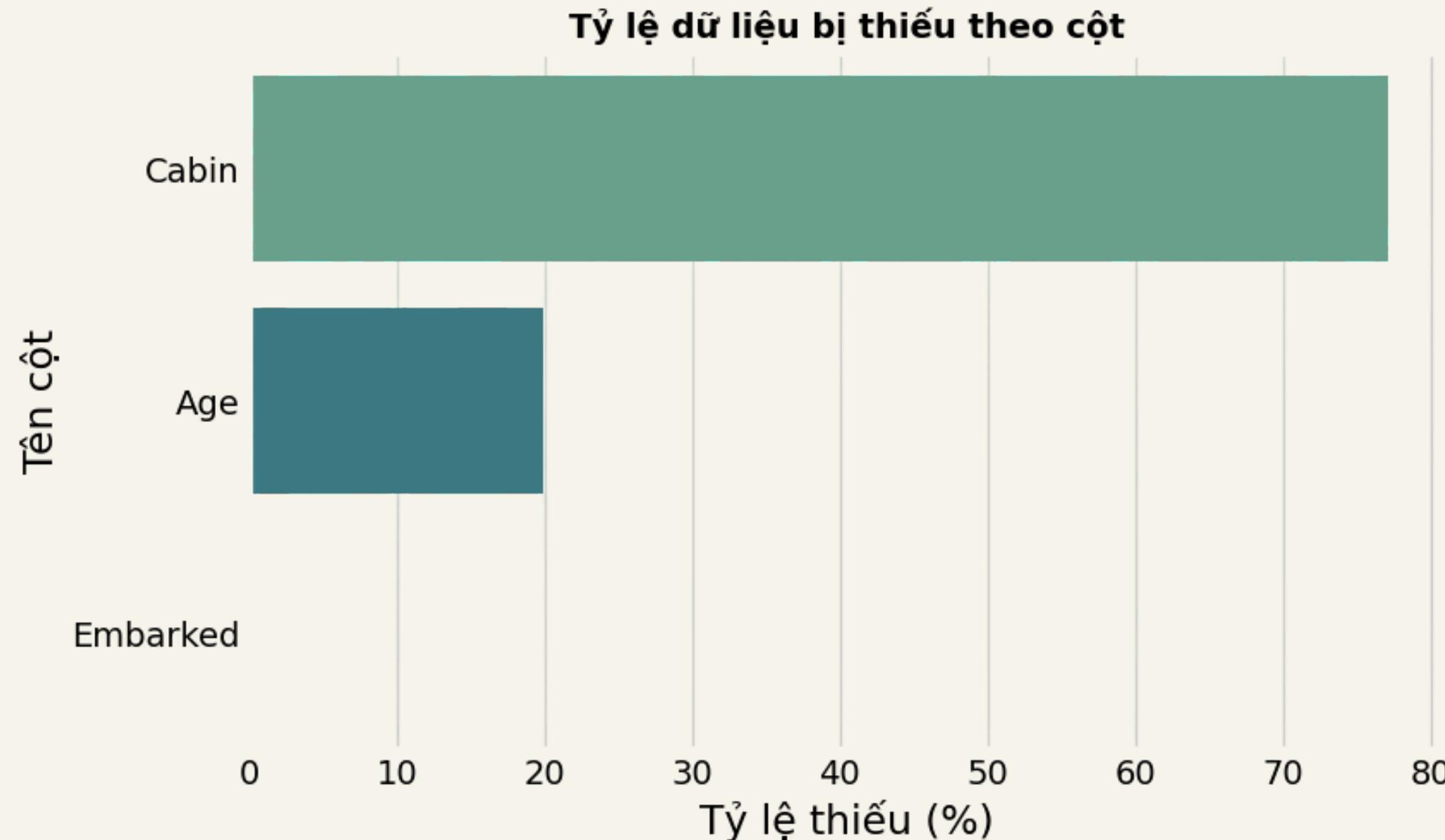
3.1 Quy trình



3. Phương pháp được đề xuất

3.2. Phân tích khám phá dữ liệu (EDA)

3.2.1 Phân tích dữ liệu thiếu



Kết quả 📋

Cột	Số lượng thiếu	Tỷ lệ (%)
Cabin	687	77.10%
Age	177	1.987%
Embarked	2	22%

Nhận xét🎯

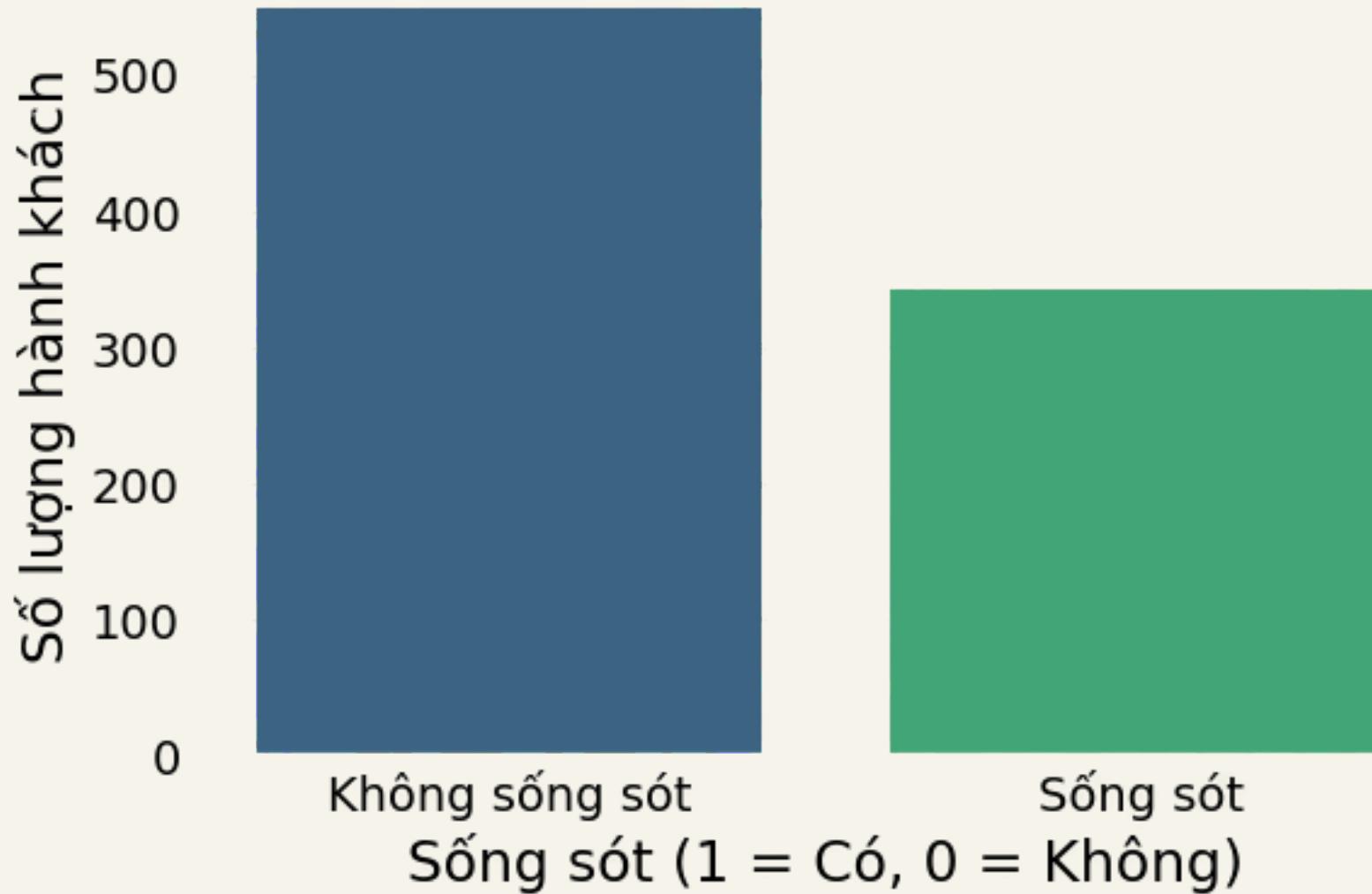
- Trong bộ dữ liệu Titanic, chỉ có 3 biến chứa dữ liệu thiếu.
- Cabin** bị thiếu nghiêm trọng nên cần loại bỏ hoặc rút trích phần thông tin còn lại, trong khi **Age** và **Embarked** có thể xử lý bằng các phương pháp điền giá trị phù hợp.

3. Phương pháp được đề xuất

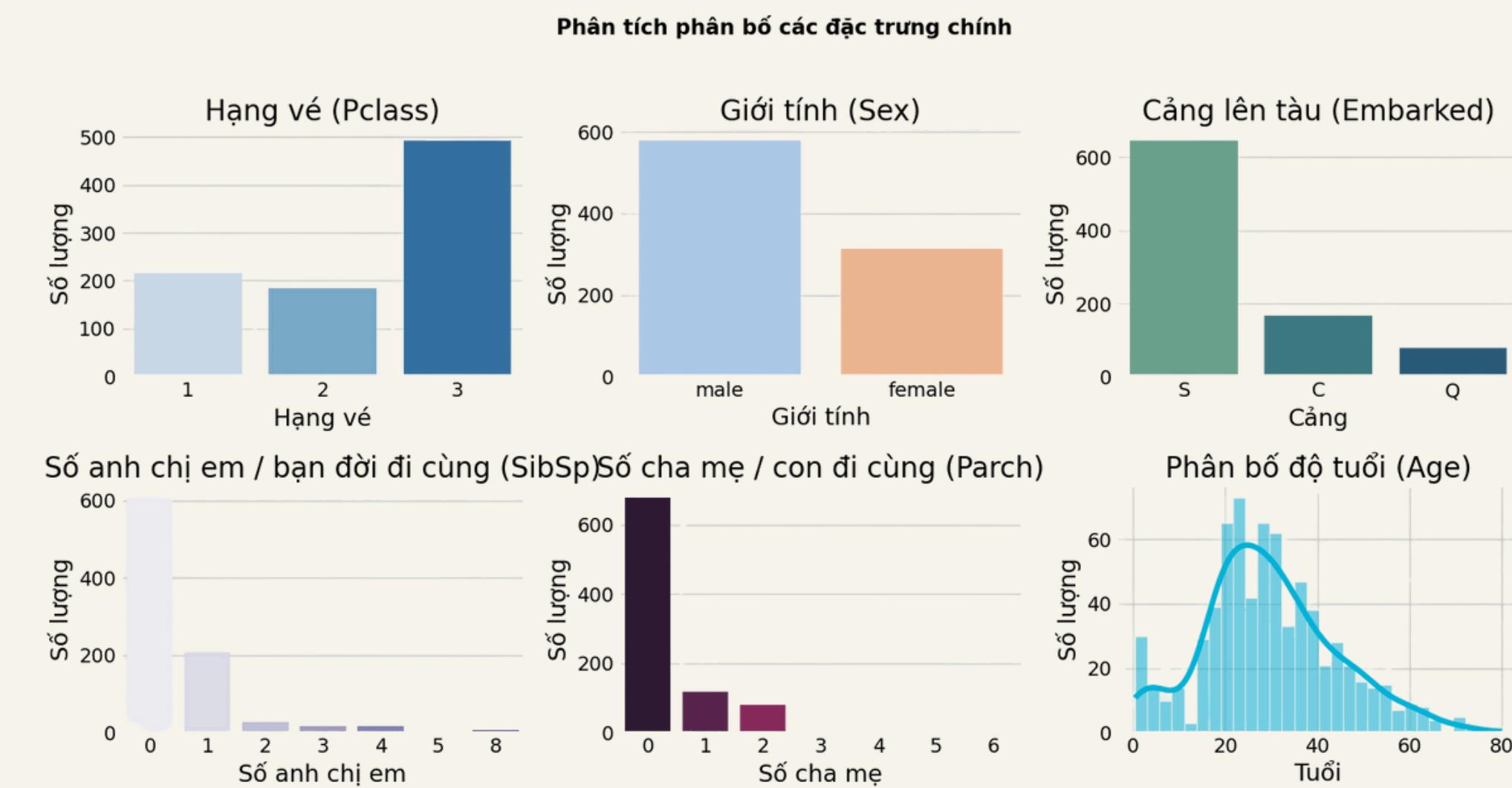
3.2. Phân tích khám phá dữ liệu (EDA)

3.2.2 Phân tích dữ liệu đơn biến

Tỷ lệ sống sót (Survival Distribution)



Phân bố biến mục tiêu — Survived



Phân bố các đặc trưng rời rạc & liên tục

3. Phương pháp được đề xuất

3.2. Phân tích khám phá dữ liệu (EDA)

3.2.2 Phân tích dữ liệu đơn biến

Kết quả 📋

- Khoảng 38% hành khách sống sót, 62% không sống sót.
- Bộ dữ liệu bất cân bằng nhẹ, nên khi huấn luyện mô hình cần chú ý đánh giá bằng F1-score hoặc ROC-AUC, không chỉ Accuracy.

Biến	Kiểu	Nhận xét chính
Pclass	Phân loại (Ordinal)	Hành khách hạng 3 chiếm đa số (~55%), hạng 1 ít hơn (~25%).
Sex	Nhị phân	Nam chiếm khoảng 65%, nữ ~35%.
Embarked	Phân loại (Categorical)	Cảng S (Southampton) là nơi khởi hành chủ yếu (~72%).
SibSp	Số lượng nguyên	Phần lớn hành khách đi một mình hoặc với 1 người.
Parch	Số lượng nguyên	Phần lớn không đi cùng cha mẹ/con, phân bố lệch trái.
Age	Liên tục (Continuous)	Phân bố gần chuẩn, tập trung trong khoảng 20-40 tuổi, có một số trẻ nhỏ.

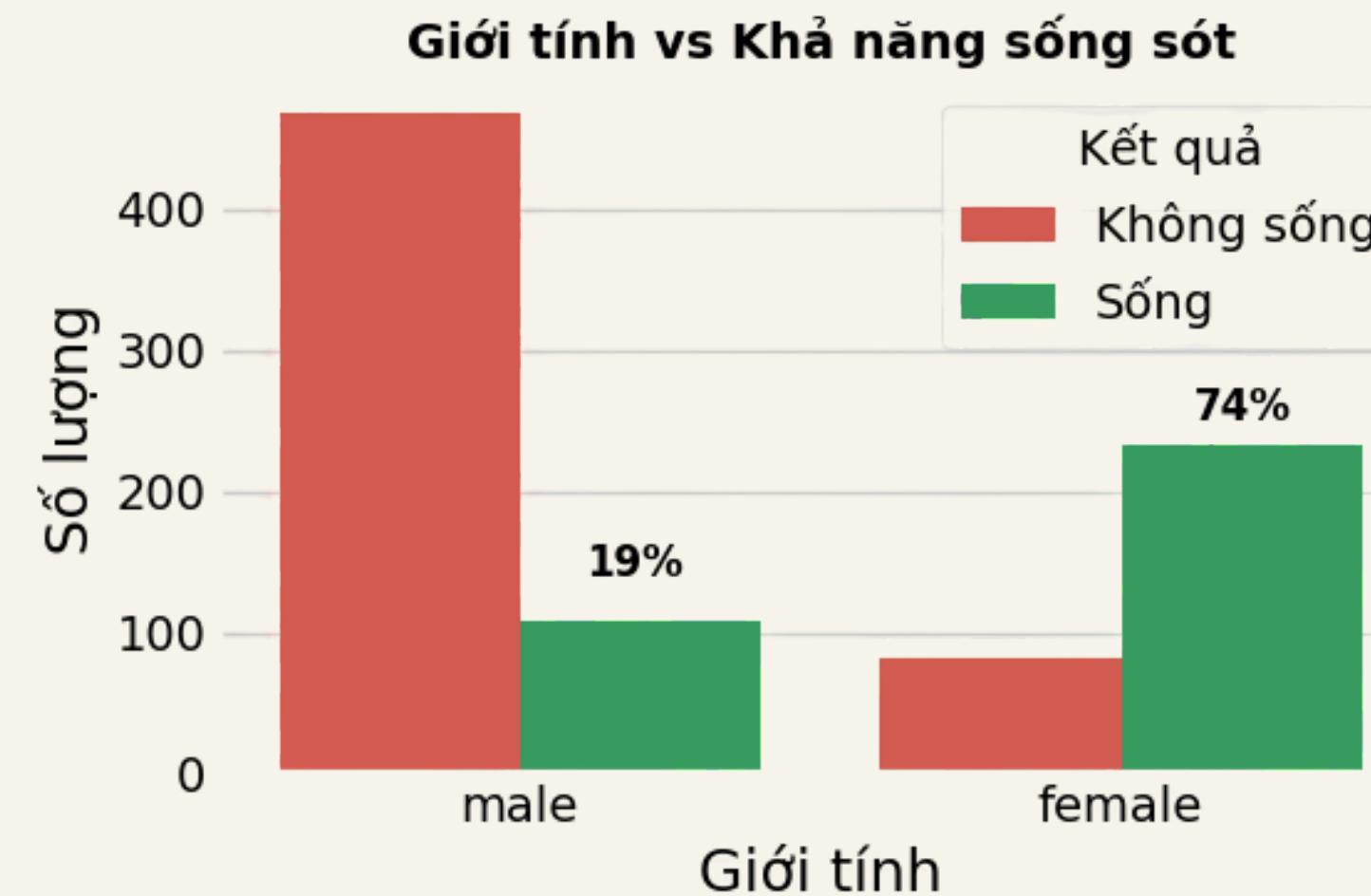
Nhận xét 🎯

- Bộ dữ liệu Titanic có phân bố lệch về giới tính (nhiều nam), hạng vé (nhiều người hạng 3), và độ tuổi chủ yếu ở người trưởng thành trẻ.
- Những yếu tố này — đặc biệt là **Pclass, Sex, và Age** — được kỳ vọng ảnh hưởng mạnh đến khả năng sống sót.

3. Phương pháp được đề xuất

3.2. Phân tích khám phá dữ liệu (EDA)

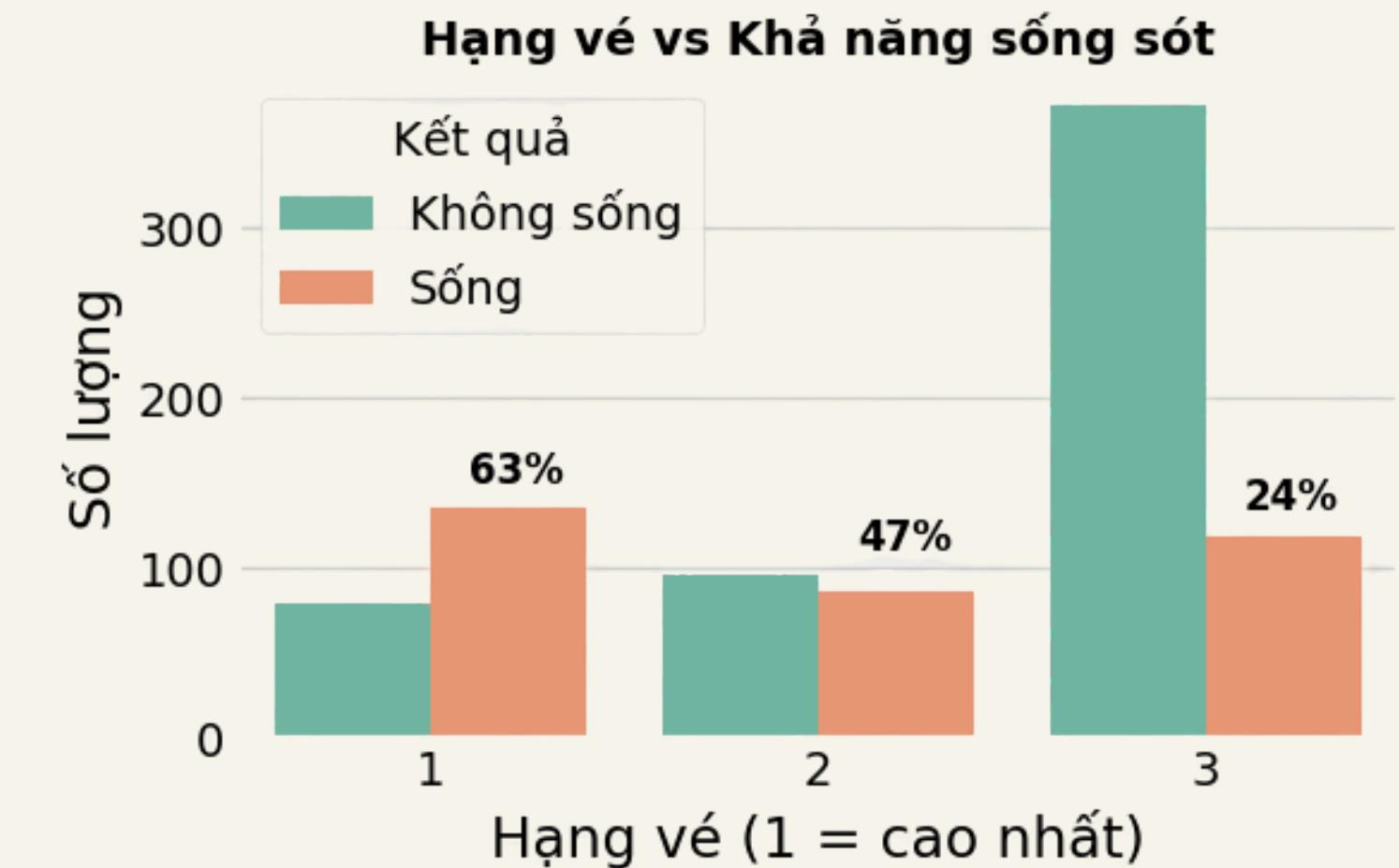
3.2.3 Phân tích dữ liệu nhị biến



Nhận xét

- Nữ có tỷ lệ sống sót ~74%
- Nam chỉ ~19%

➡ Rõ ràng phụ nữ được ưu tiên cứu trước.



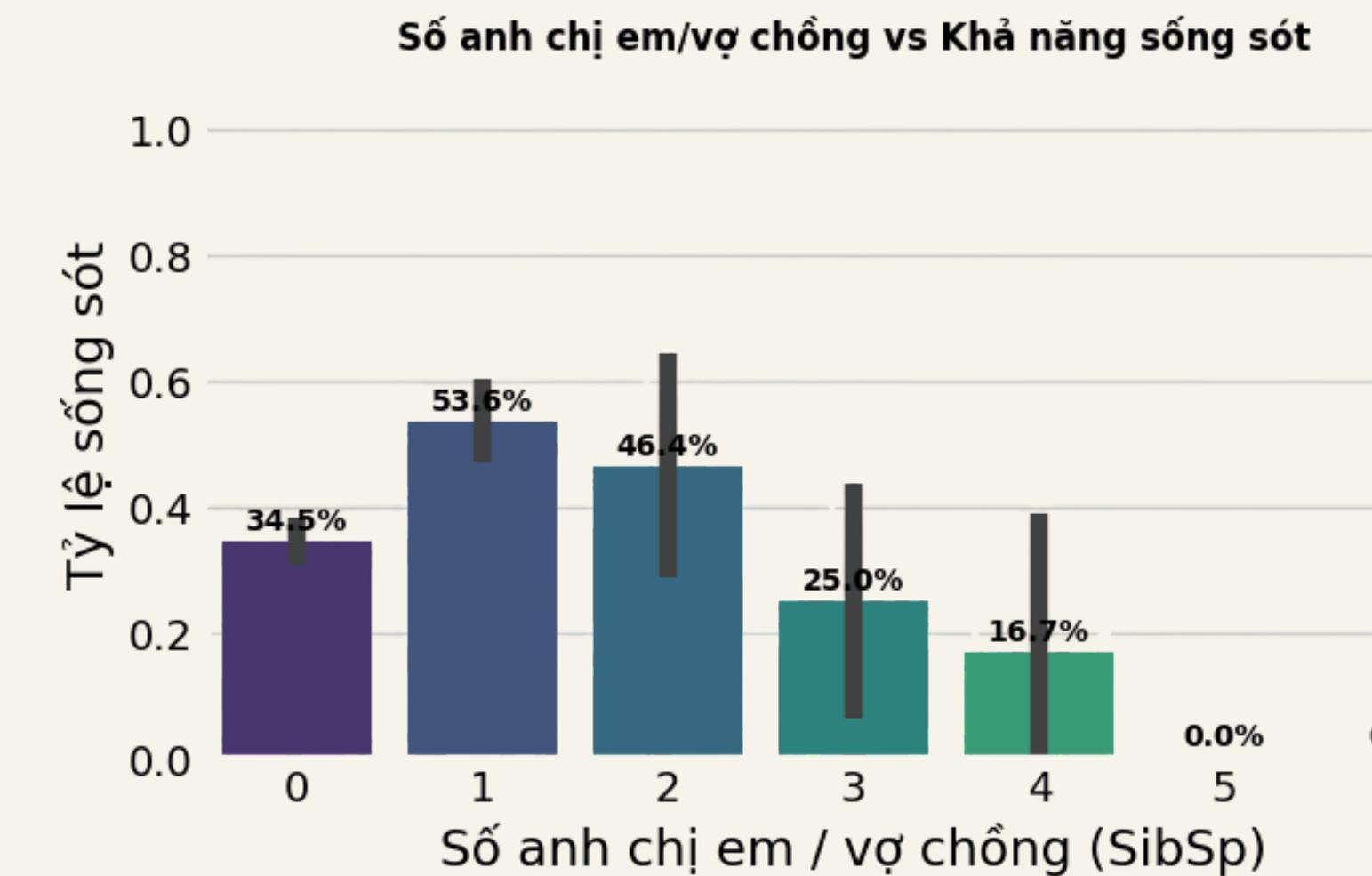
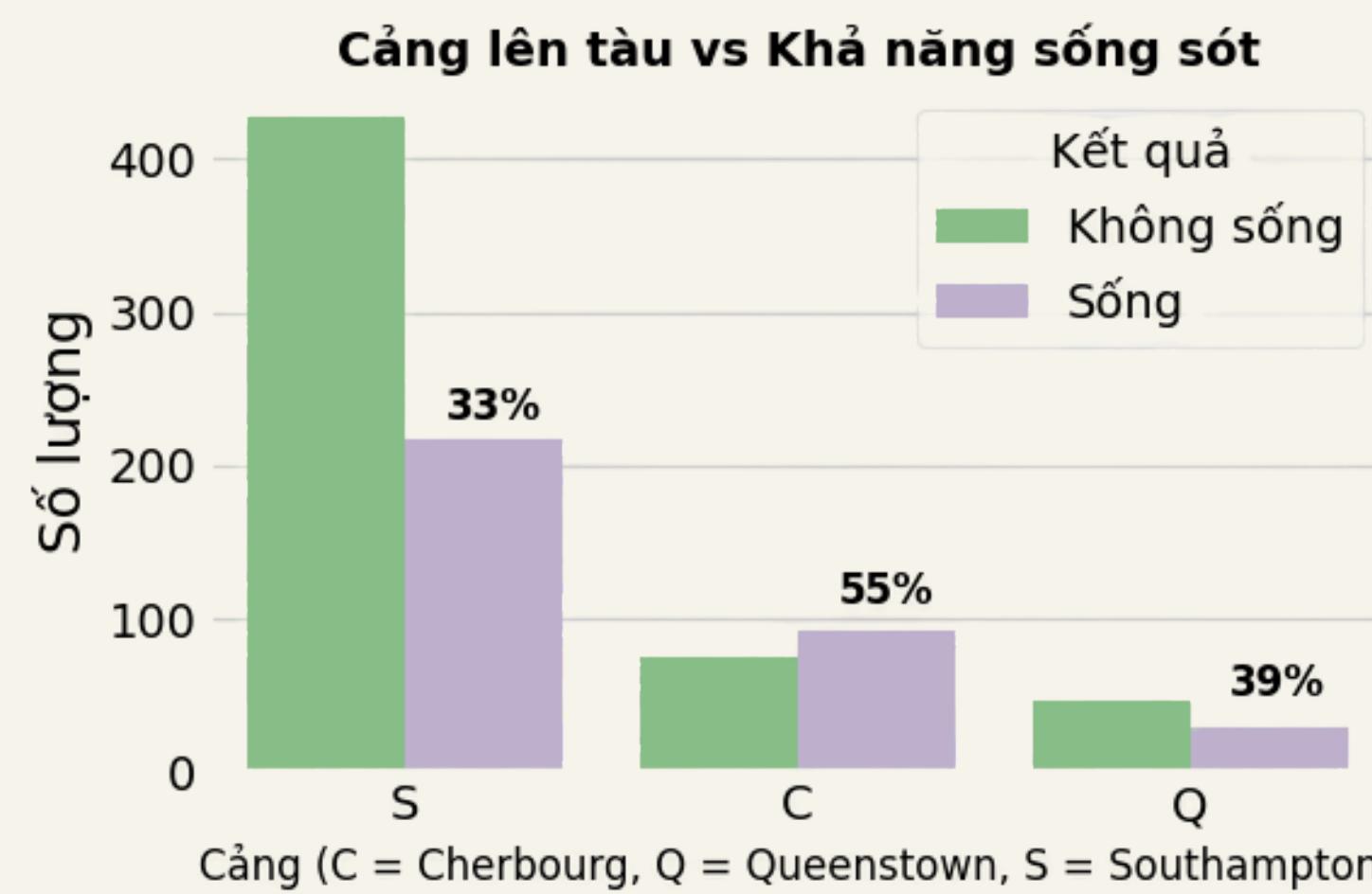
- Pclass 1: sống ~63%
- Pclass 2: sống ~47%
- Pclass 3: sống ~24%

➡ Hạng vé càng cao → khả năng sống càng lớn.

3. Phương pháp được đề xuất

3.2. Phân tích khám phá dữ liệu (EDA)

3.2.3 Phân tích dữ liệu nhị biến



Nhận xét

- Hành khách từ C (Cherbourg) có tỷ lệ sống cao nhất (~55%)
 - S (Southampton) thấp nhất (~33%)
- ➡ Có thể do phân bố Pclass khác nhau ở từng cảng.

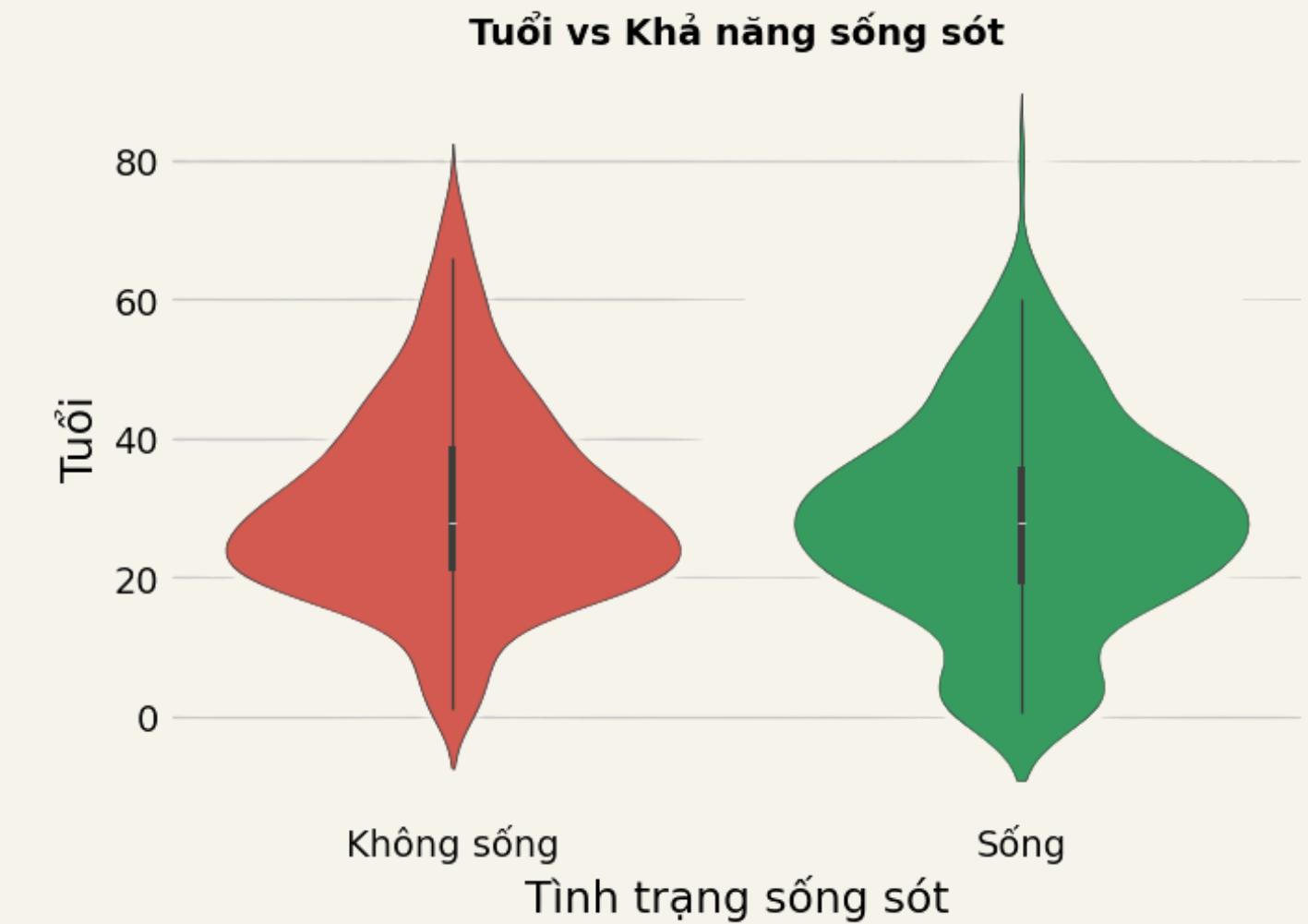
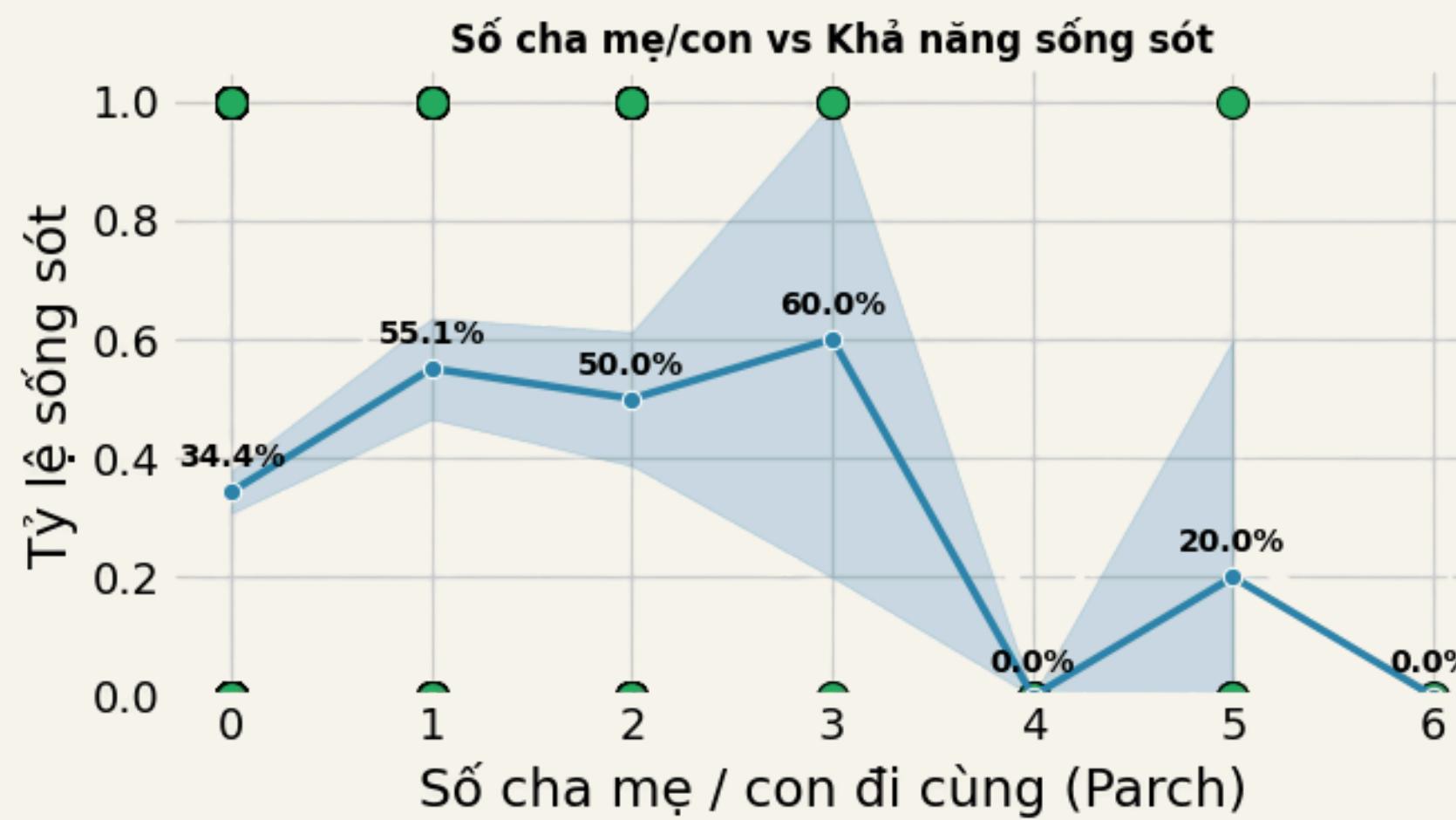
- Đi một mình (SibSp=0): sống ~34%
- Gia đình nhỏ (1–2): sống cao hơn (~50%)
- Gia đình lớn (>3): gần như không sống sót



3. Phương pháp được đề xuất

3.2. Phân tích khám phá dữ liệu (EDA)

3.2.3 Phân tích dữ liệu nhị biến



Nhận xét 🎯

- Người đi một mình (Parch=0): sống thấp (~34%)
- Đi cùng 1–3 người: tỷ lệ sống cao (~50%)
- Gia đình quá đông (>3): tỷ lệ sống thấp

➡ Gia đình nhỏ được hỗ trợ cứu hộ tốt hơn.

- Trẻ nhỏ (<10 tuổi) có khả năng sống cao hơn
 - Nhóm tuổi 30–40 tử vong nhiều nhất
- ➡ “Phụ nữ và trẻ em được cứu trước” thể hiện rất rõ.

3. Phương pháp được đề xuất

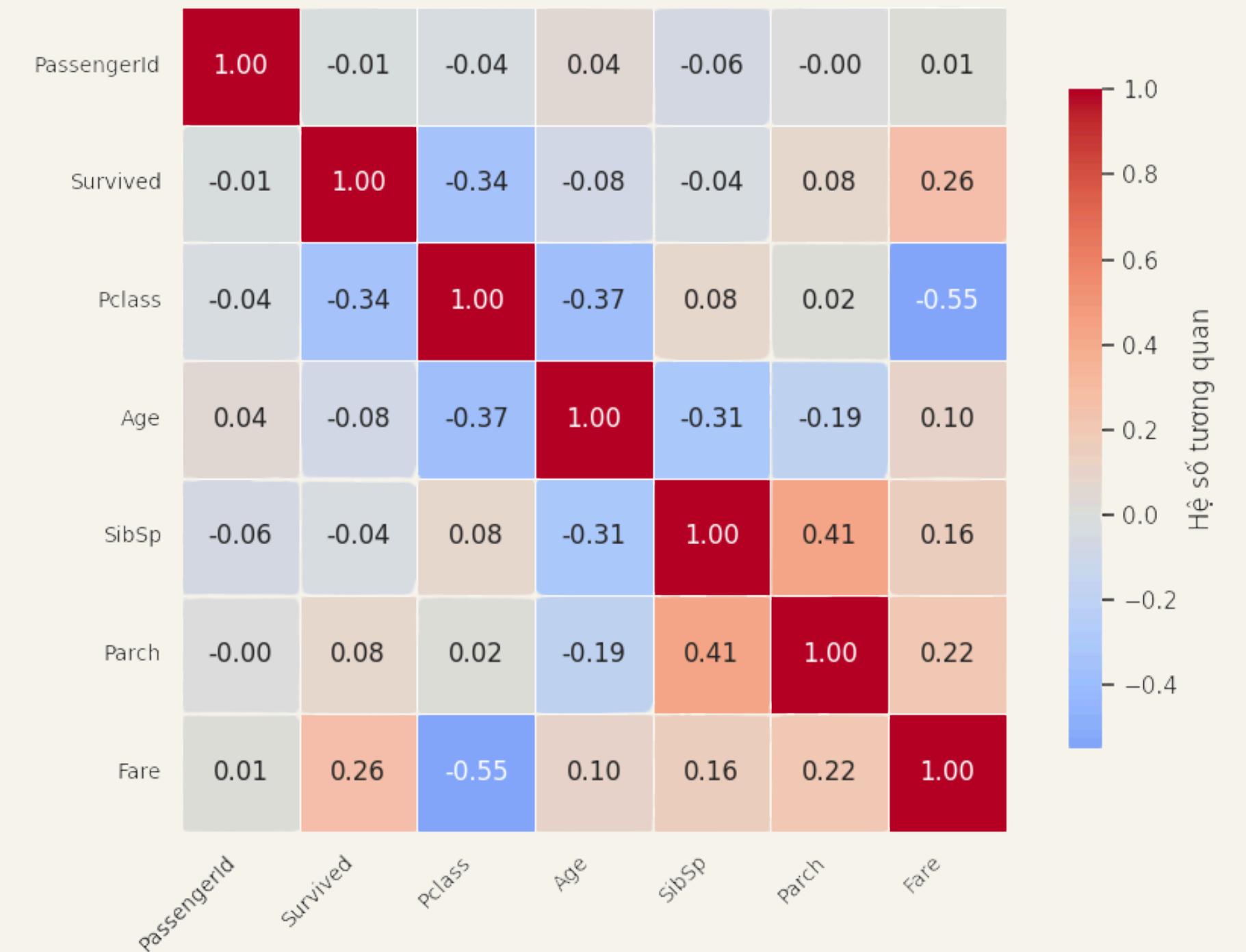
3.2. Phân tích khám phá dữ liệu (EDA)

3.2.4 Phân tích dữ liệu đa biến

Nhận xét

- Hạng vé (*Pclass*) và Giá vé (*Fare*) là hai yếu tố ảnh hưởng mạnh nhất đến khả năng sống sót.
- Các yếu tố như tuổi (*Age*) hay số người đi cùng (*SibSp, Parch*) chỉ có ảnh hưởng nhẹ.
- Hành khách giàu, ở hạng cao có nhiều cơ hội được cứu hơn — phù hợp với quy tắc “phụ nữ và trẻ em, quý tộc trước” thời đó.

Bản đồ tương quan giữa các đặc trưng



3. Phương pháp được đề xuất

3.2. Phân tích khám phá dữ liệu (EDA)

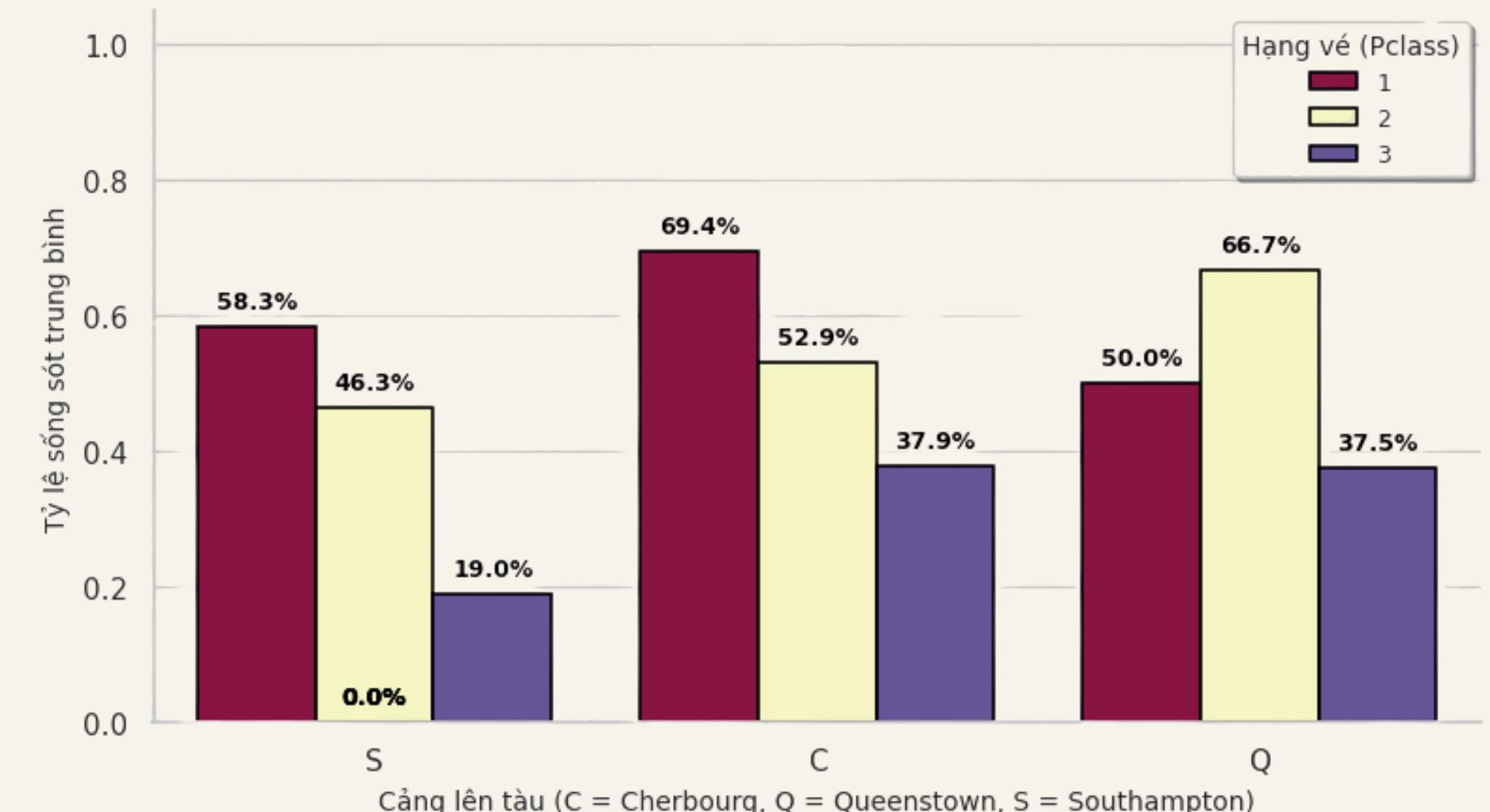
3.2.4 Phân tích dữ liệu đa biến

Nhận xét

- Cảng C (Cherbourg) có tỷ lệ sống cao nhất ở tất cả các hạng, đặc biệt là Pclass 1.
- Cảng S (Southampton) có nhiều người đi hạng 3 → tỷ lệ sống thấp.
- Cảng Q (Queenstown) ít người, tỷ lệ sống trung bình.

➡ Yếu tố cảng phản ánh phần nào điều kiện kinh tế – hành khách Cherbourg thường mua vé hạng cao hơn.

Tỷ lệ sống sót theo Cảng lên tàu và Hạng vé



3. Phương pháp được đề xuất

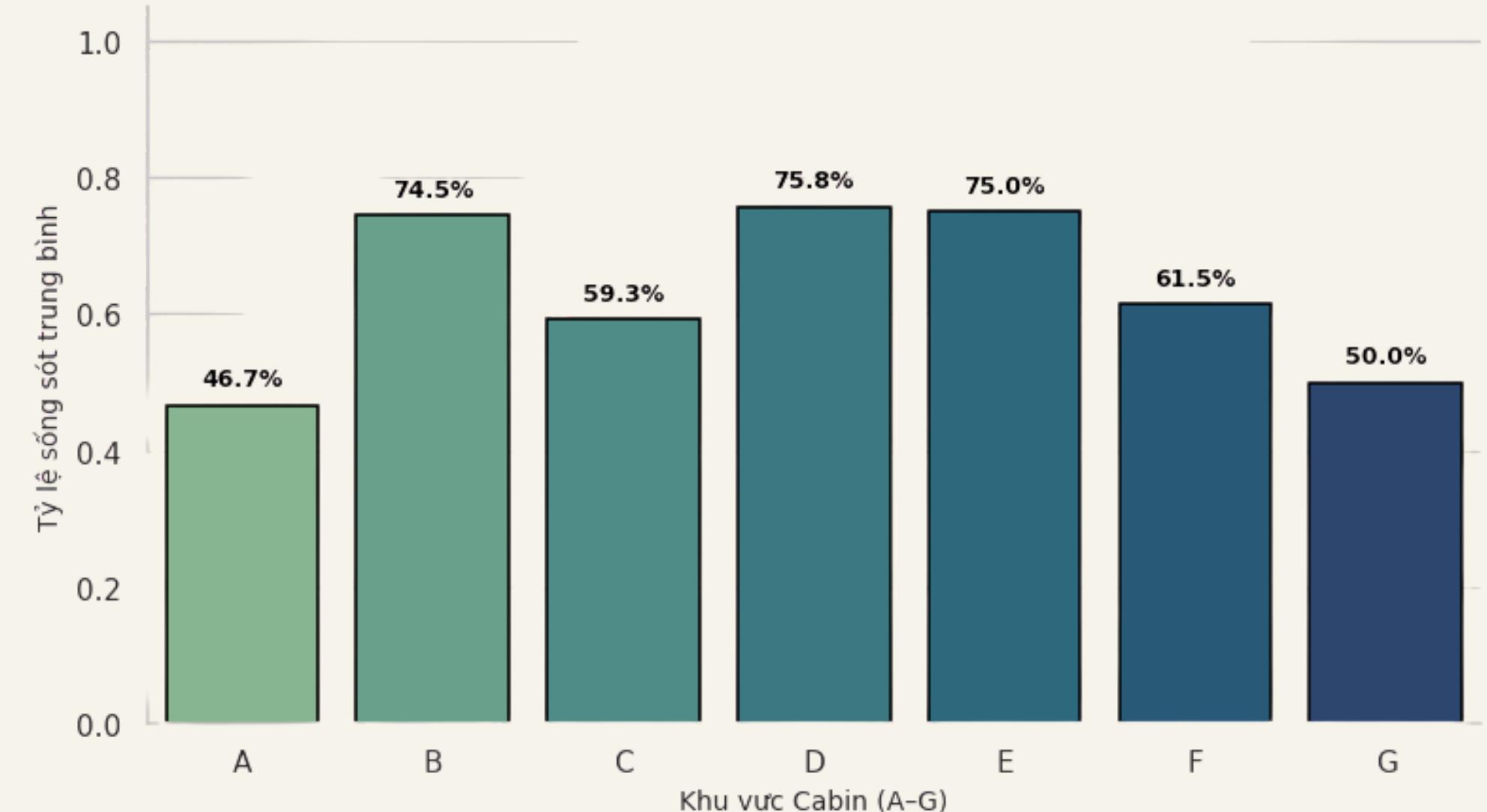
3.2. Phân tích khám phá dữ liệu (EDA)

3.2.4 Phân tích dữ liệu đa biến

Nhận xét 🎯

- Cabin A–C (gần boong trên, Pclass 1) có tỷ lệ sống rất cao.
- Cabin E–G (boong dưới, Pclass 3) tử vong nhiều.
➡ Cabin càng cao trên tàu → gần lối thoát → khả năng sống cao hơn.

Tỷ lệ sống sót theo Khu vực Cabin



3. Phương pháp được đề xuất

3.2. Phân tích khám phá dữ liệu (EDA)

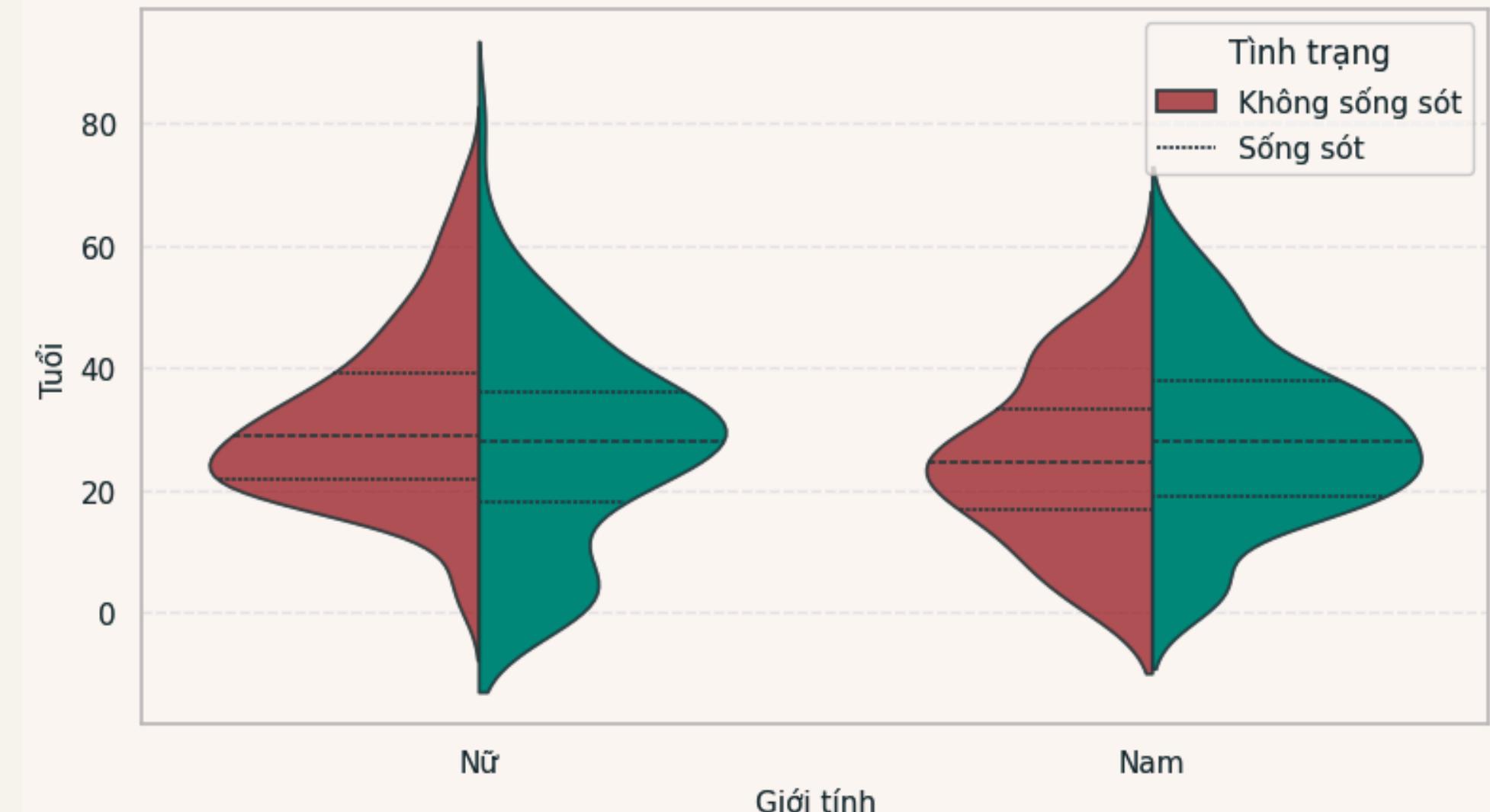
3.2.4 Phân tích dữ liệu đa biến

Nhận xét 🎯

- Phụ nữ ở mọi độ tuổi đều sống cao hơn nam.
- Trẻ em (đặc biệt bé gái) <10 tuổi sống cao.
- Nam giới tuổi 20–40 tử vong cao nhất.

➡ Thể hiện rõ chính sách “Women & Children First”
khi cứu hộ.

Phân bố Tuổi theo Giới tính và Khả năng sống sót



3. Phương pháp được đề xuất

3.2. Phân tích khám phá dữ liệu (EDA)

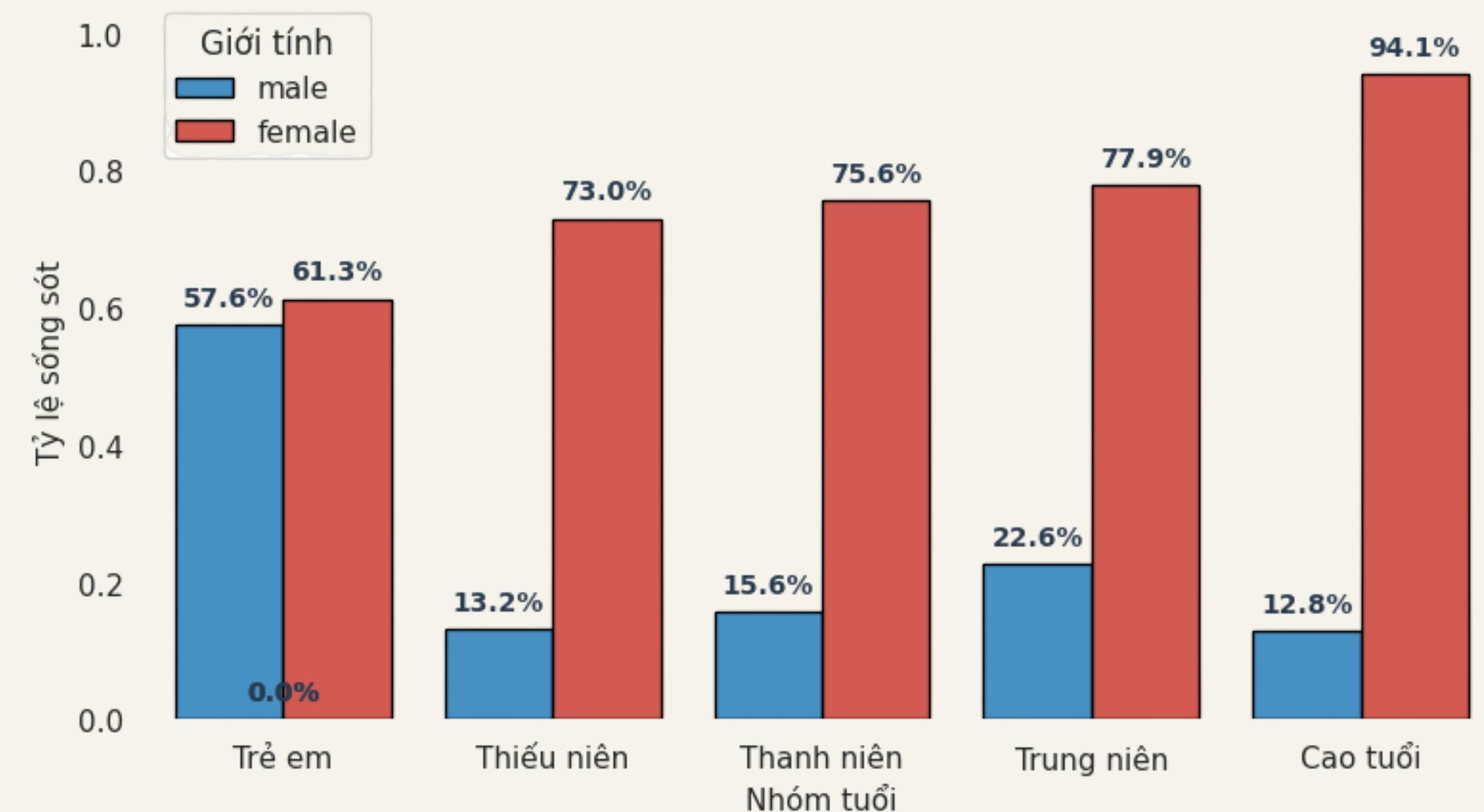
3.2.4 Phân tích dữ liệu đa biến

Nhận xét 🎯

- Trẻ em sống nhiều nhất, bất kể giới tính.
- Phụ nữ trung niên sống cao (~70%).
- Nam thanh niên (18–40) tử vong nhiều nhất.

➡ Cứu hộ ưu tiên trẻ em và phụ nữ trưởng thành, không phân biệt tầng lớp.

Tỷ lệ sống sót theo Nhóm tuổi và Giới tính



3. Phương pháp được đề xuất

3.2. Phân tích khám phá dữ liệu (EDA)

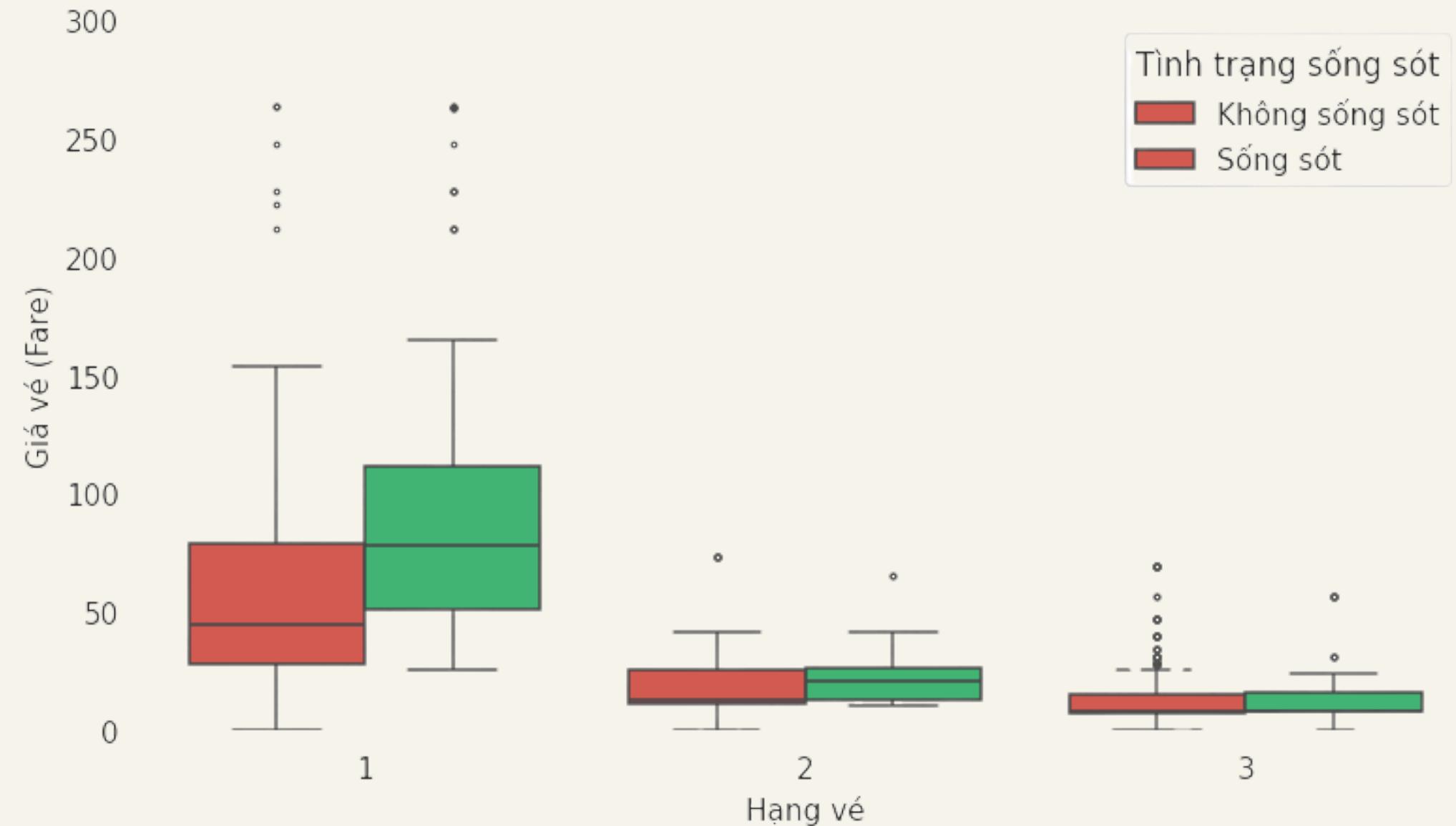
3.2.4 Phân tích dữ liệu đa biến

Nhận xét

- Người trả vé cao (Pclass 1) có cơ hội sống lớn.
- Pclass 3 giá thấp, sống thấp.
- Một số ít người Pclass 3 trả vé cao hơn mức trung bình vẫn sống (có thể vị trí cabin thuận lợi).

➡ Giá vé có tương quan dương mạnh với khả năng sống.

Giá vé - Hạng vé - Tình trạng sống sót



3. Phương pháp được đề xuất

3.3. Xử lý missing value

Cột	Số lượng thiếu	Tỷ lệ (%)
Cabin	687	77.10%
Age	177	1.987%
Embarked	2	22%

Cột	Số lượng thiếu	Tỷ lệ (%)
Cabin	327	78.23%
Age	86	20.57%
Fare	1	0.24%

Tập train

Nhận xét 

Cabin bị thiếu nhiều nhất, tiếp theo là Age và Embarked.

◆ **Hướng xử lý:**

- Cabin: loại bỏ do thiếu >70%, thông tin không đủ ý nghĩa.
 - Age: Dùng KNN Imputer với đặc trưng ['Age', 'Pclass']
 - Embarked: điền bằng mode (S) – cảng phổ biến nhất.
 - Fare: điền median theo Pclass.
- ◆ Sau khi xử lý, dữ liệu không còn giá trị khuyết, sẵn sàng cho phân tích đặc trưng.

Tập test

3. Phương pháp được đề xuất

3.3. Xử lý missing value

3.3.1. Xử lý cột Cabin

- Thiếu quá 77% dữ liệu → không thể suy luận chính xác.
- Dữ liệu Cabin (phòng) chỉ liên quan phần nhỏ hành khách, phân bố không đều.

 Giải pháp: Loại bỏ cột Cabin khỏi mô hình.

3.3.2. Xử lý cột Embarked

- Chỉ thiếu 2 giá trị.

 Điền bằng giá trị phổ biến nhất (mode): 'S'.

3.3.3. Xử lý cột Fare

- Chỉ thiếu 1 giá trị.

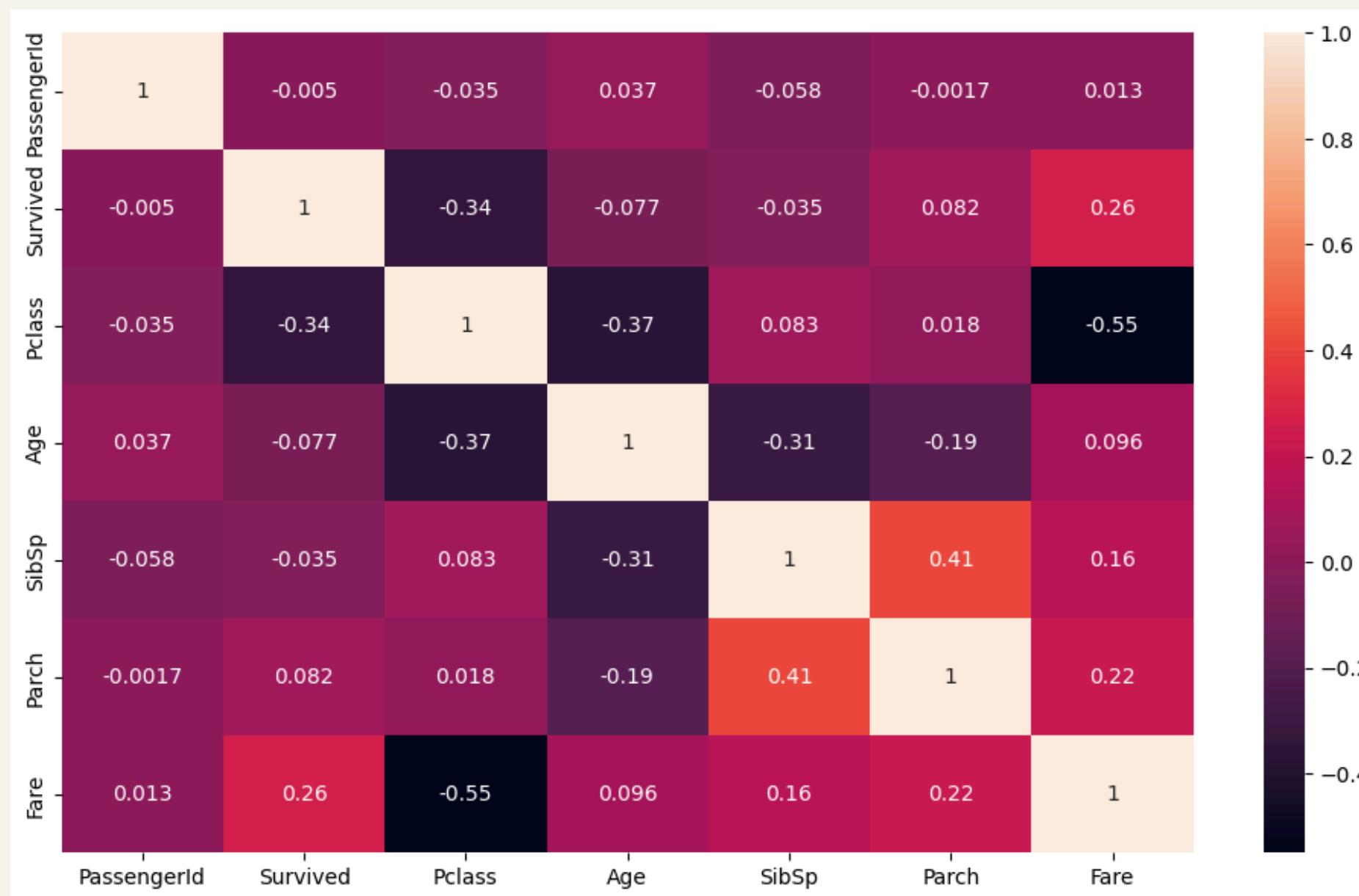
 Điền trung vị(median)



3. Phương pháp được đề xuất

3.3. Xử lý missing value

3.3.4. Xử lý cột Age

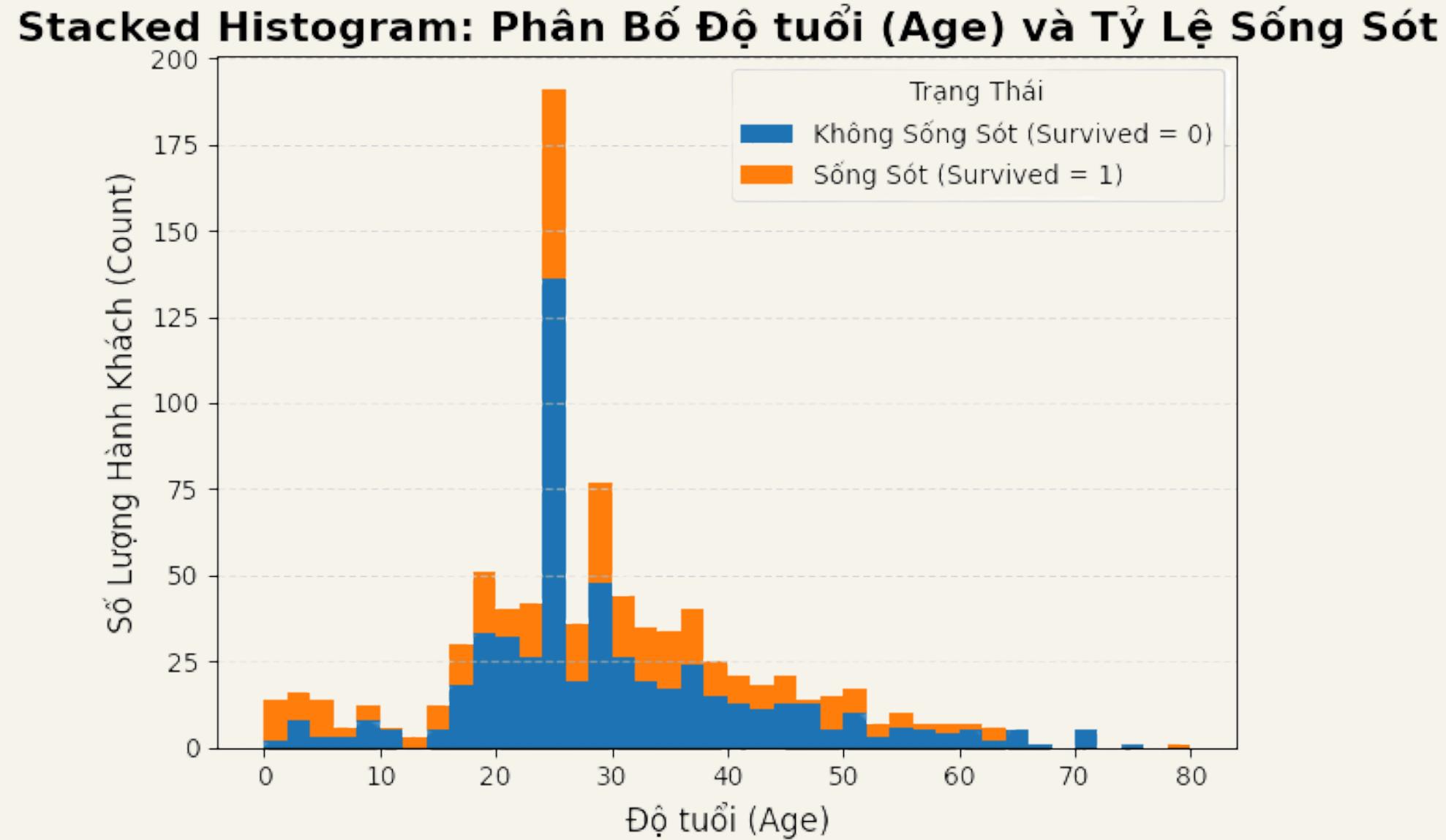


Nhận xét

Thấy 'Age' tương quan cao với 'Pclass' nên ta có thể xử lý missing value bằng các Imputer với biến được chọn là Pclass

3. Phương pháp được đề xuất

3.4. Feature Engineering

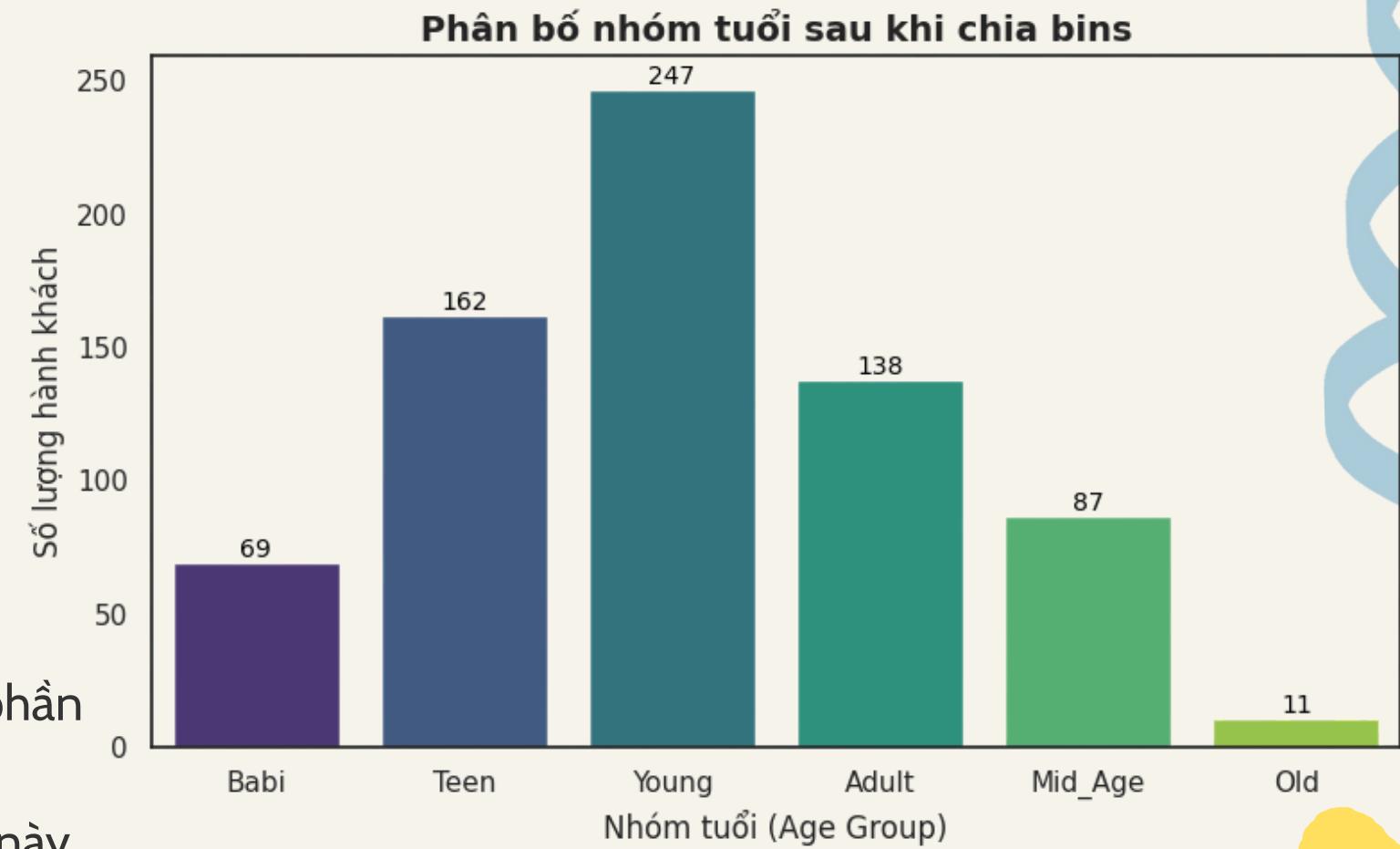


Nhận xét

- Ta có thể chia vé với bins=[-1,12,22,34,46,64,100] và gắn labels
- Nhóm Young (23–34) và Adult (35–46) chiếm tỷ lệ lớn nhất trong tập dữ liệu → phần lớn hành khách là người trưởng thành.
- Nhóm Old (65+) và Baby (0–12) có số lượng ít nhất → rất ít hành khách ở 2 nhóm này.

Lý do phân chia

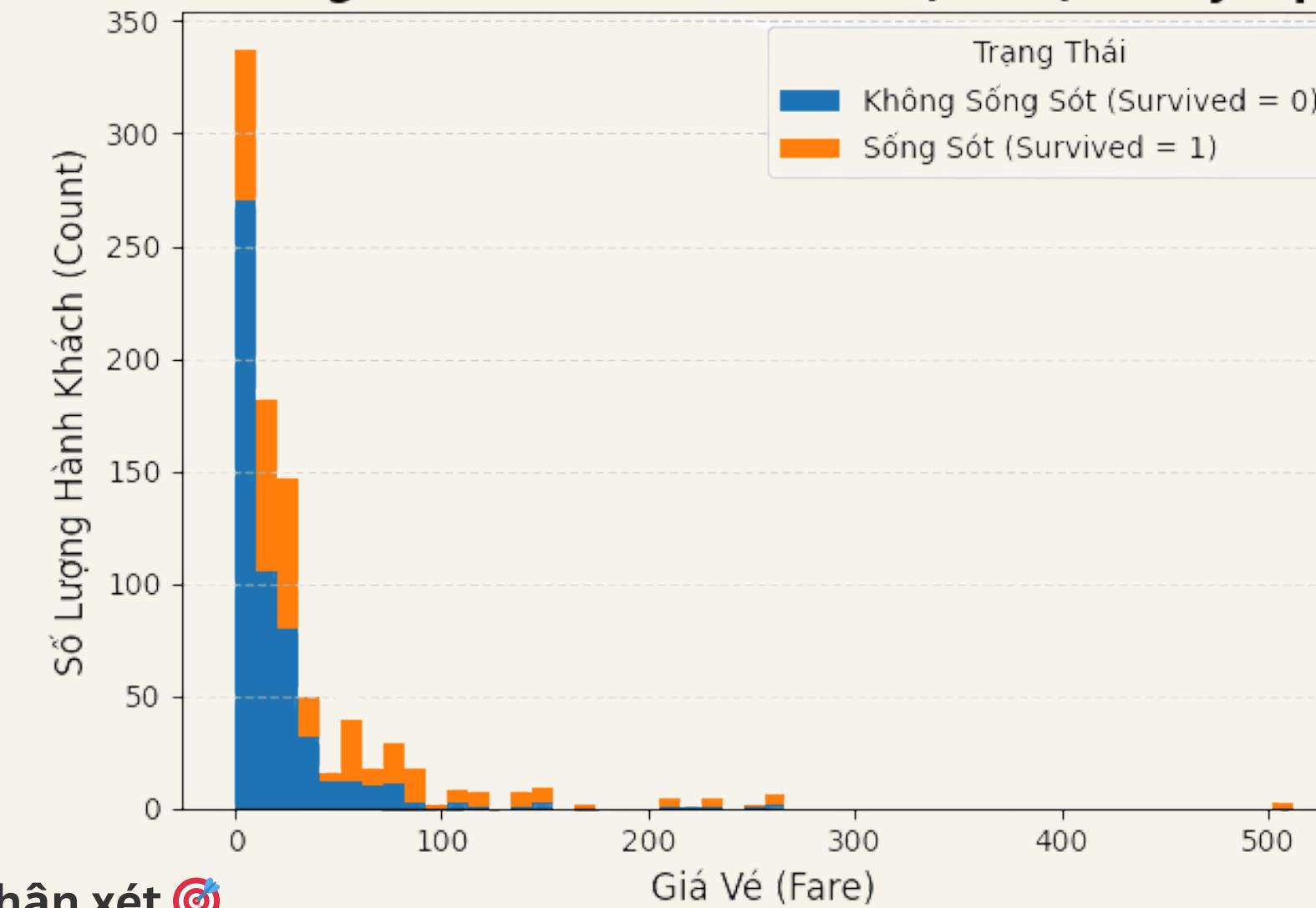
- Giúp chuẩn hóa biến liên tục (Age) thành biến phân loại (categorical) → mô hình học dễ hơn, giảm nhiễu.
- Một số thuật toán (như cây quyết định, logistic regression) hoạt động hiệu quả hơn với nhóm tuổi rời rạc thay vì giá trị tuổi thực.



3. Phương pháp được đề xuất

3.4. Feature Engineering

Stacked Histogram: Phân Bố Giá Vé (Fare) và Tỷ Lệ Sống Sót

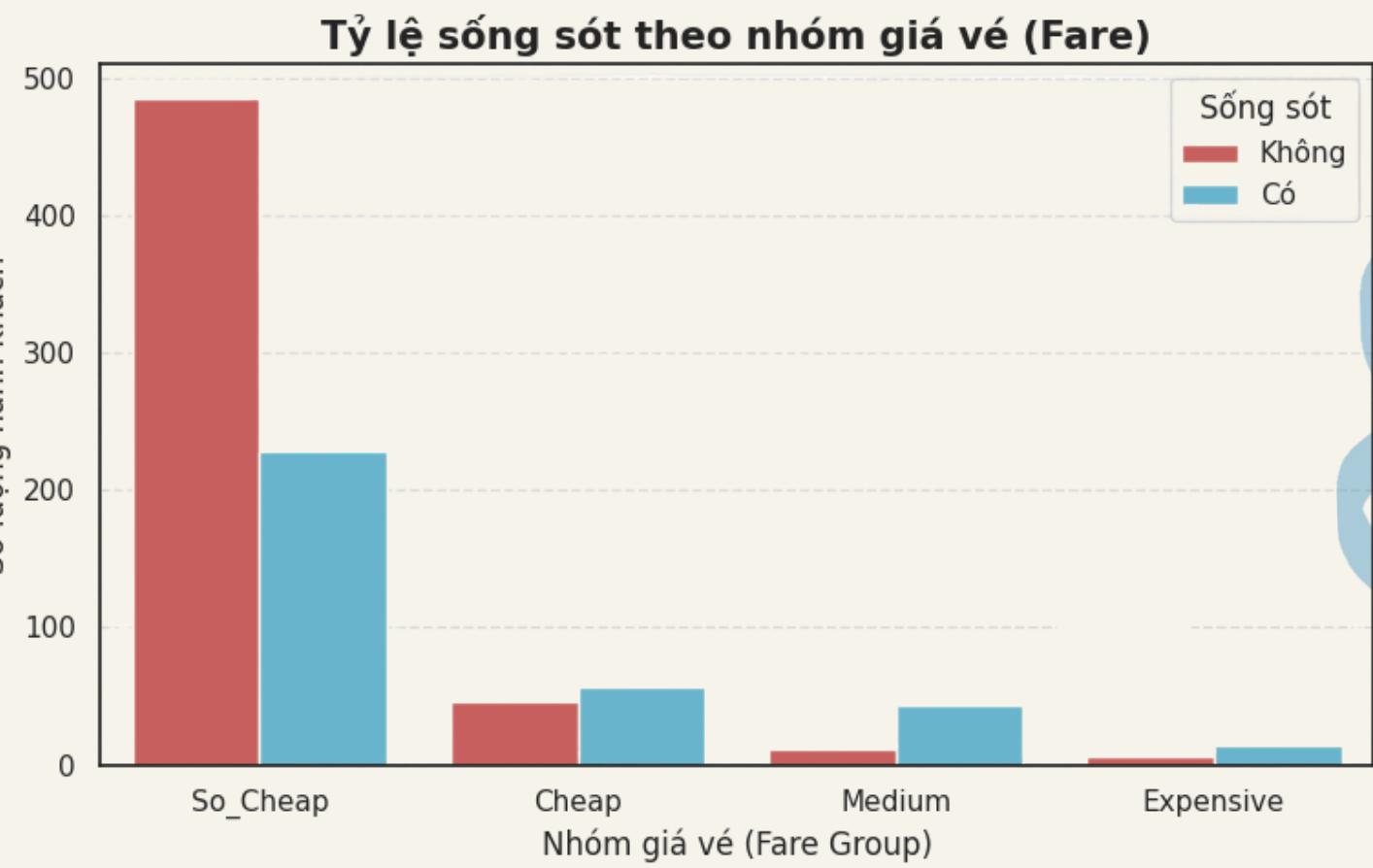


Nhận xét

- Ta có thể chia vé với bins=[-1,40,80,200,1000] và gắn labels
- Nhóm Expensive có tỷ lệ sống sót cao nhất (~80%) → hành khách giàu có, thường ở boong cao – dễ thoát hiểm hơn.
- Nhóm So_Cheap và Cheap chiếm đa số, nhưng tỷ lệ sống chỉ khoảng 20–30%, đặc biệt là hạng 3.
- Nhóm Medium nằm giữa, cho thấy khả năng sống sót trung bình (do gồm cả hạng 1 và 2).

Lý do phân chia

- ta chia thành 4 khoảng dễ hiểu, phản ánh phân tầng kinh tế
- Việc binning giúp mô hình học được sự khác biệt rõ ràng về “khả năng sống sót theo điều kiện kinh tế”.



3. Phương pháp được đề xuất

3.4. Feature Engineering

Pclass: Hạng

```
np.unique(df_train['Pclass'].values)  
  
array([1, 2, 3])
```

Nhận xét ⚡

- Cột Pclass là thuộc tính Category gồm các giá trị số 1 2 3
- ➡ Ta sẽ tạm thời giữ nguyên cột này

Embarked: Cảng

```
np.unique(df_train['Embarked'].dropna().values)  
  
array(['C', 'Q', 'S'], dtype=object)
```

Nhận xét ⚡

- Cột Embarked là thuộc tính Category gồm các giá trị C, Q, S
- ➡ Ta sẽ mã hóa cột Embarked dưới dạng số: "C": 1, "Q": 2, "S": 3



3. Phương pháp được đề xuất

3.4. Feature Engineering

Name: Tên hành khách

Nhận xét 

- Cột Name là thuộc tính text
- => Ta sẽ trích xuất đại từ nhân xưng của Name

```
np.unique(df_output['Name'])  
  
array(['Capt', 'Col', 'Don', 'Dr', 'Jonkheer', 'Lady', 'Major', 'Master',  
       'Miss', 'Mlle', 'Mme', 'Mr', 'Mrs', 'Ms', 'Rev', 'Sir',  
       'the Countess'], dtype=object)
```

Số lượng các đại từ nhân xưng 

```
df_output['Name'].value_counts()
```

Name	count
Mr	517
Miss	182
Mrs	125
Master	40
Dr	7
Rev	6
Col	2
Mlle	2
Major	2
Ms	1
Mme	1
Don	1
Lady	1
Sir	1
Capt	1
the Countess	1
Jonkheer	1

Top 5 đại từ phổ biến

```
top_5_title  
  
['Mr', 'Miss', 'Mrs', 'Master', 'Dr']
```

Gắn labels và đếm số lượng 

```
df_output['Name'].value_counts()
```

Name	count
1	517
2	182
3	125
4	40
6	20
5	7

Name: count, dtype: int64

3. Phương pháp được đề xuất

3.4. Feature Engineering

SibSp: Anh chị em

```
np.unique(df_train['SibSp'].dropna().values)  
  
array([0, 1, 2, 3, 4, 5, 8])
```

Nhận xét ⚪

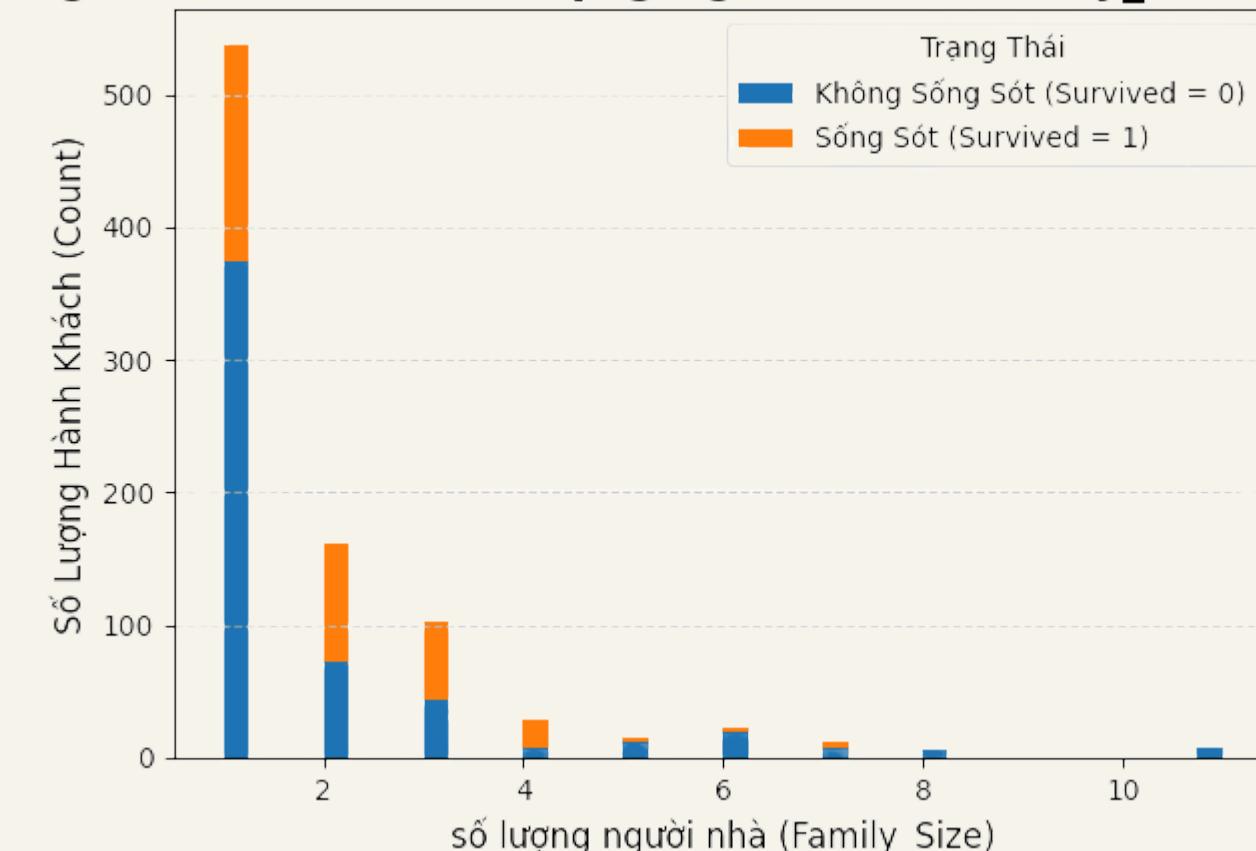
- Cột SibSp là thuộc tính Category gồm các giá trị số: 0, 1, 2, 3, 4, 5, 8

FamilySize: Số lượng thành viên trong gia đình

Cách xử lý

Tạo biến **Family_Size** = **SibSp** + **Parch** + 1 để biểu thị tổng số người trong gia đình đi cùng hành khách (bao gồm cả bản thân)

Stacked Histogram: Phân bố số lượng người nhà (Family_Size) và Tỷ lệ Sống Sót



Parch: Ba mẹ, con cái

```
np.unique(df_train['Parch'].dropna().values)  
  
array([0, 1, 2, 3, 4, 5, 6])
```

Nhận xét ⚪

- Cột Parch là thuộc tính Category gồm các giá trị số: 0, 1, 2, 3, 4, 5, 6

Kết quả 📈

- Family_Size = 1 chiếm ~537 hành khách (lớn nhất).
- Nhóm đông (≥ 5 người) hiếm gặp và có tỷ lệ sống sót thấp nhất.

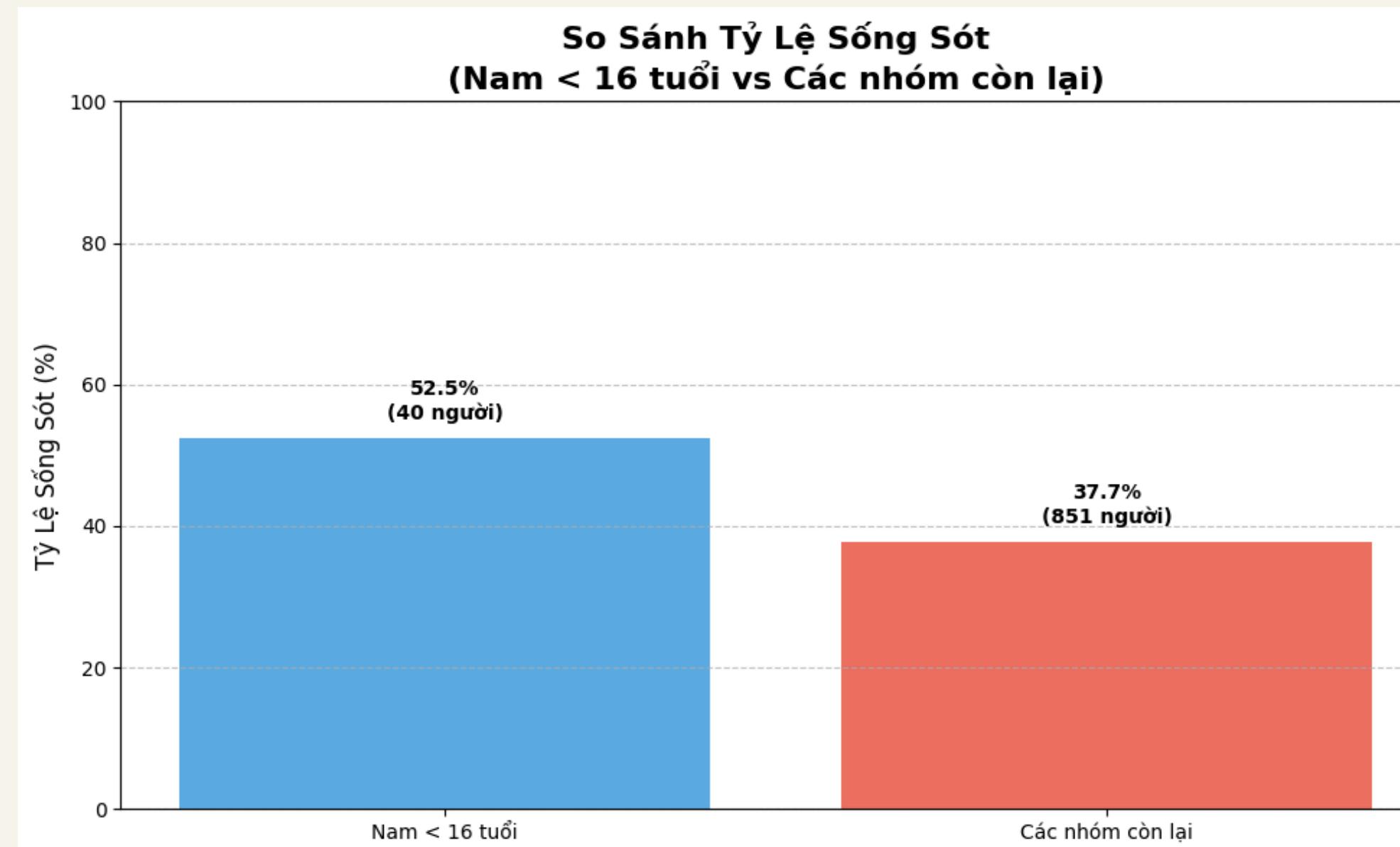
Nhận xét ⚪

-> Số lượng người nhà càng nhiều thì tỷ lệ sống sót càng thấp

3. Phương pháp được đề xuất

3.4. Feature Engineering

Boy: Name = 'male' & Age < 16



Kết quả 📈

Tỷ lệ sống sót Nam < 16 tuổi: 52.5%
Tỷ lệ sống sót các nhóm còn lại: 37.72%

Cách xử lý

Feature mới Boy mang giá trị:

- True (hoặc 1): hành khách là bé trai (<16 tuổi)
- False (hoặc 0): hành khách không phải bé trai

Nhận xét ⚪

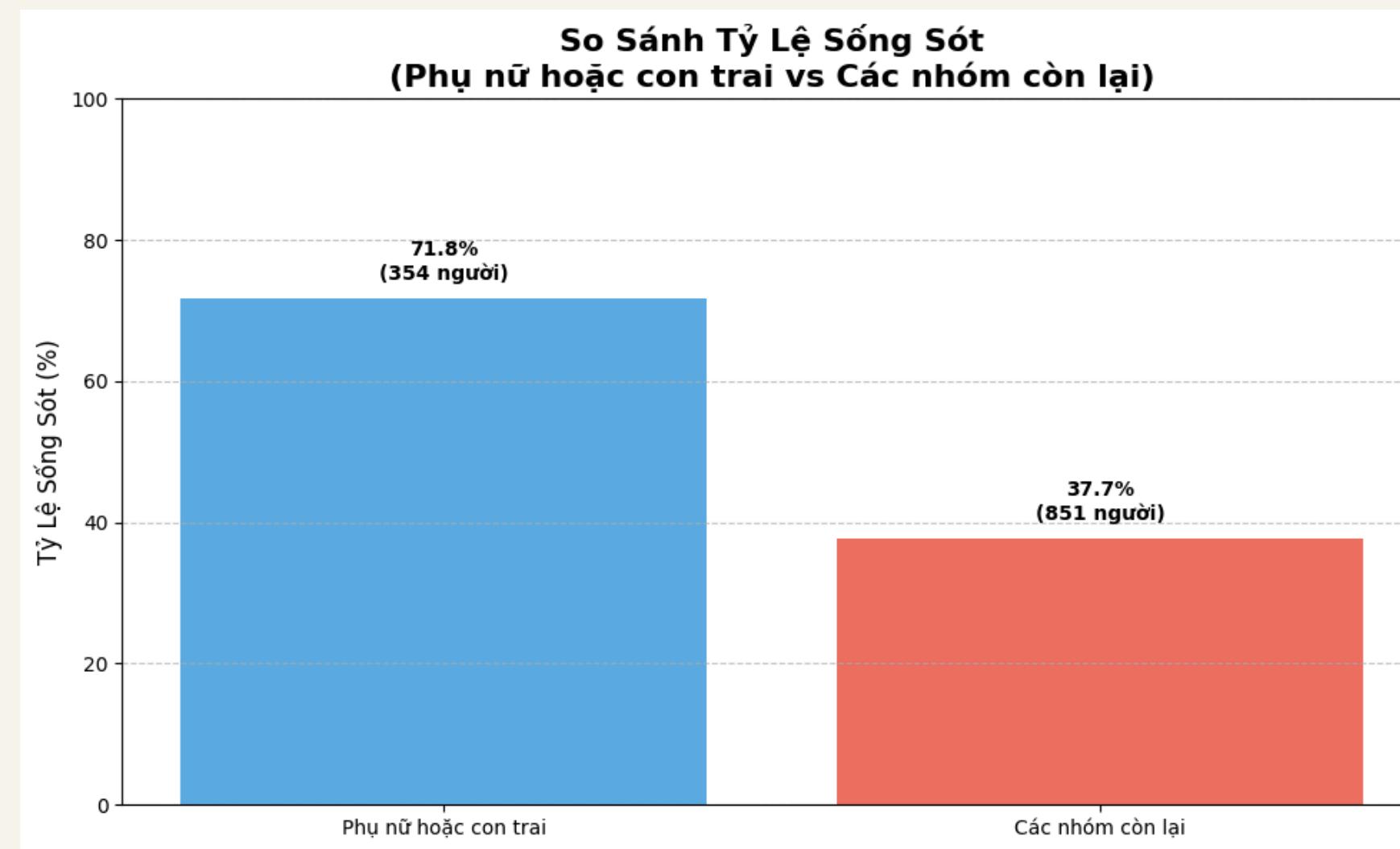
✓ Feature Boy thể hiện nhóm nam giới nhỏ tuổi (<16 tuổi), có tỷ lệ sống sót cao hơn trung bình (~52.5% so với 37.7%).

➡ Việc thêm đặc trưng này giúp mô hình hiểu rõ hơn mối tương quan giữa giới tính, độ tuổi và khả năng sống sót, có tiềm năng tăng độ chính xác dự đoán.

3. Phương pháp được đề xuất

3.4. Feature Engineering

WomanOrBoy



Kết quả

Tỷ lệ sống sót Phụ nữ hoặc con trai: 71.75%
Tỷ lệ sống sót các nhóm còn lại: 37.72%

Cách xử lý

Feature mới WomanOrBoy mang giá trị:

- True (hoặc 1): nếu hành khách là phụ nữ hoặc là bé trai (<16 tuổi)
- False (hoặc 0): nếu hành khách là nam trưởng thành (≥ 16 tuổi)

Nhận xét

Feature WomanOrBoy giúp mô hình:

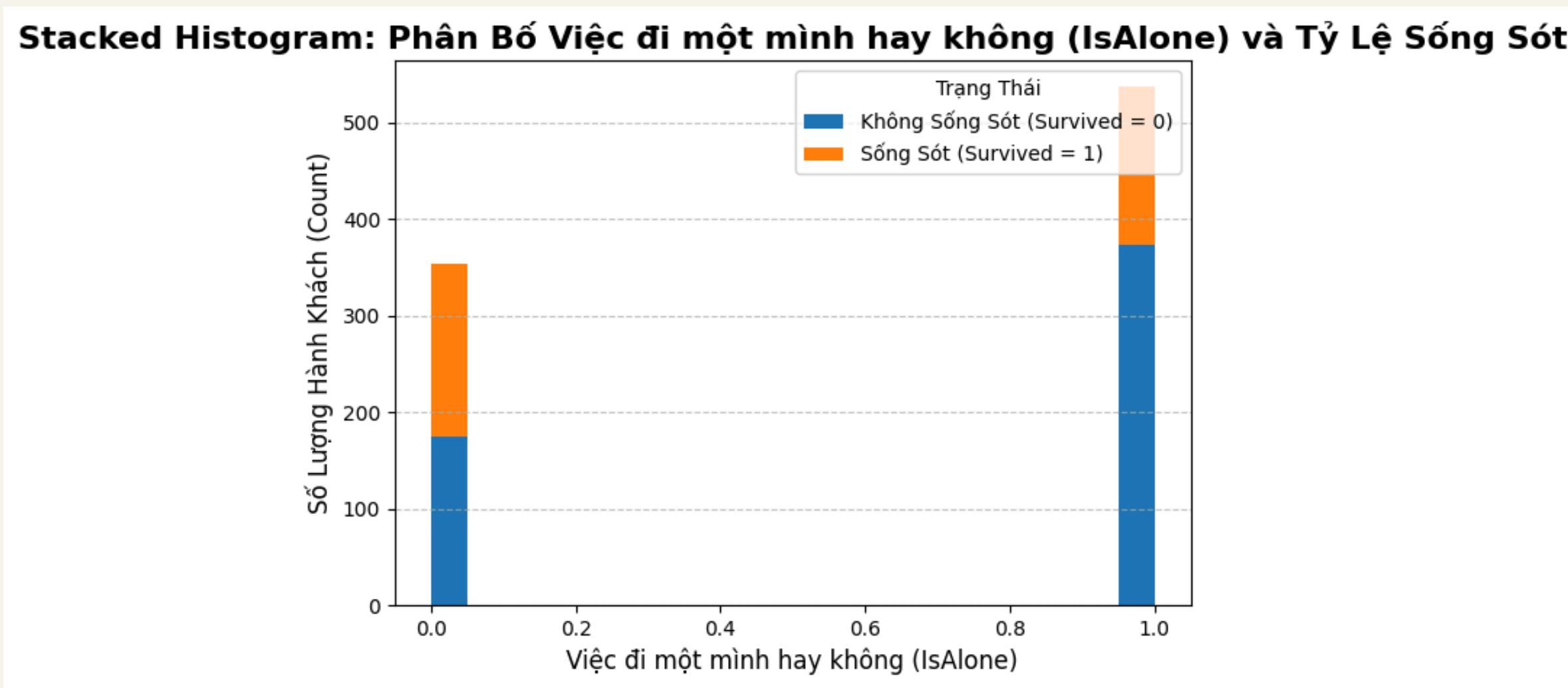
- Hiểu rõ mối quan hệ phi tuyến giữa giới tính, tuổi và khả năng sống sót.
- Giảm nhiễu do việc chỉ dùng Sex hoặc Age riêng lẻ.

➡ Tăng khả năng dự đoán đúng, vì đặc trưng này bao trùm một quy luật hành vi thực tế.

3. Phương pháp được đề xuất

3.4. Feature Engineering

IsAlone: Tính chất hành khách có đi một mình hay không



Kết quả 📈

- Biểu đồ cho thấy: Việc đi một mình có tỷ lệ sống sót cao hơn
- Nhóm đi cùng người thân ($\text{IsAlone} = 0$) lại có khả năng được cứu cao hơn — có thể vì được hỗ trợ, có tinh thần đồng đội, hoặc ưu tiên cứu hộ theo nhóm.

Cách xử lý

Tạo cột mới IsAlone:

1 → Hành khách đi một mình (không có anh chị em, vợ chồng, cha mẹ, con cái trên tàu).
0 → Hành khách đi cùng người thân.

👉 Dựa trên cột $\text{Family_Size} = \text{SibSp} + \text{Parch} + 1$, ta xác định tình trạng đi một mình.

Nhận xét 🎯

- Feature IsAlone giúp mô hình hiểu mối quan hệ giữa yếu tố xã hội và khả năng sống sót.
- Đây là đặc trưng đơn giản nhưng mạnh, vì nó chuyển hóa thông tin phức tạp (SibSp, Parch) thành một tín hiệu nhị phân dễ học.
- Các mô hình như Logistic Regression, Decision Tree, RandomForest... đều có thể khai thác đặc trưng này hiệu quả.

3. Phương pháp được đề xuất

3.4. Feature Engineering

Family_Survival: theo xu hướng người thân trong gia đình sẽ có cùng cảnh ngộ

Ý tưởng

- Các thành viên trong cùng gia đình thường có cùng số phận (cùng sống hoặc cùng mất) trên Titanic.
- Sử dụng các thông tin: **Last_Name + Fare** và **Ticket** để dự đoán “**Family_Survival**”.

Kết hợp 2 quy tắc:

```
def infer_family_survival(row):
    # Ưu tiên quy tắc theo Last_Name + Fare
    key1 = ("LF", row["Last_Name"], row["Fare"])
    if key1 in fam_rules:
        return fam_rules[key1]
    # Sau đó đến Ticket
    key2 = ("T", row["Ticket"])
    if key2 in fam_rules:
        return fam_rules[key2]
    return DEFAULT_SURV

df_output["Family_Survival"] = df_train.apply(infer_family_survival, axis=1)
```

Ưu tiên Last_Name + Fare → Ticket → DEFAULT_SURV (0.5)

Cách xử lý

Quy tắc 1: Có cùng last name và giá vé (Fare)

```
DEFAULT_SURV = 0.5
fam_rules = {}
tmp = df_train.copy()
tmp["Last_Name"] = tmp["Name"].str.split(",").str[0]
tmp["Fare"] = tmp["Fare"].fillna(tmp["Fare"].median())
for (lname, fare), grp in tmp.groupby(["Last_Name", "Fare"], dropna=False):
    if len(grp) <= 1:
        continue
    smax = grp["Survived"].max()
    smin = grp["Survived"].min()
    if smax == 1:
        fam_rules[("LF", lname, fare)] = 1.0
    elif smin == 0:
        fam_rules[("LF", lname, fare)] = 0.0
```

Nhóm theo họ và giá vé.
Nếu ít nhất 1 người sống → 1.0,
cả nhóm chết → 0.0.
Nếu xung đột → bỏ qua (trả về
mặc định 0.5).

Quy tắc 2: Có cùng Ticket

```
# Quy tắc theo Ticket trong TRAIN
for ticket, grp in tmp.groupby("Ticket", dropna=False):
    if len(grp) <= 1:
        continue
    smax = grp["Survived"].max()
    smin = grp["Survived"].min()
    if smax == 1:
        fam_rules[("T", ticket)] = 1.0
    elif smin == 0:
        fam_rules[("T", ticket)] = 0.0
```

Nhóm theo số vé, cùng
nguyên tắc như trên.

3. Phương pháp được đề xuất

3.4. Feature Engineering

Family_Survival: theo xu hướng người thân trong gia đình sẽ có cùng cảnh ngộ

Kết quả

- Số lượng trường hợp áp dụng được quy tắc
Last_Name + Fare: X người (ví dụ 150) → phần lớn là gia đình đi cùng vé giống nhau.
- Số lượng trường hợp áp dụng được quy tắc
Ticket: Y người (ví dụ 100) → bao gồm cả những hành khách không cùng họ nhưng đi chung ticket.
- Trường hợp không xác định (mặc định 0.5): Z người (ví dụ 542) → những người đi một mình hoặc không có thông tin trùng khớp với các quy tắc.

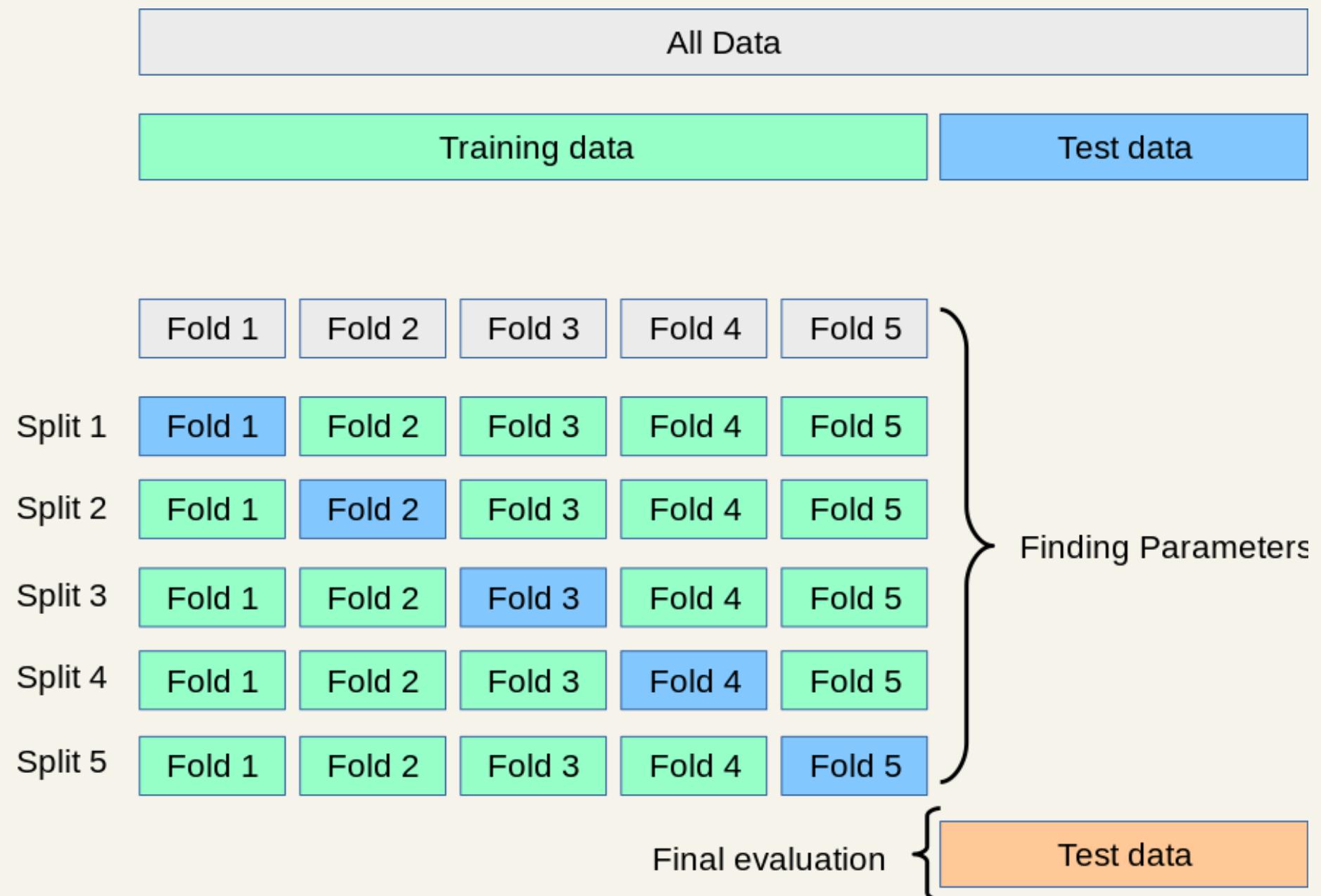
Nhận xét

Giúp mô hình nhận diện nhóm gia đình/nhóm vé, tăng độ chính xác cho những trường hợp đi cùng nhau.

Tác động đến mô hình: cải thiện khả năng dự đoán cho những trường hợp có nhiều người cùng họ hoặc cùng vé, nhưng ít ảnh hưởng đến các trường hợp đi một mình.

3. Phương pháp được đề xuất

3.5. K-Fold Cross-Validation Splitting



Concept:

“K-Fold Cross-Validation chia dữ liệu thành K phần bằng nhau, mỗi phần lần lượt được dùng làm Validation set, các phần còn lại dùng để Train.”

Application:

Trong bài toán Phân loại nhị phân (như dự đoán sống sót trên Titanic), việc sử dụng Stratified K-Fold là cần thiết

- Quá trình này **lặp lại K lần** (thường là 5 hoặc 10), mỗi lần chọn một fold khác nhau làm Validation set.
- Sau khi huấn luyện và đánh giá, **lấy trung bình kết quả** qua các vòng để có **độ chính xác ổn định hơn**.

Final metrics: Accuracy, F1-Score, ROC-AUC.

4. Thực nghiệm và thảo luận

4.1 Logistic Regression

Các Tham Số Huấn Luyện:

```
param_grid = {
    'logreg_c': [0.01, 0.1, 1, 10, 100],
    'logreg_penalty': ['l1', 'l2'],
    'logreg_solver': ['liblinear']}
}
```

Ghi chú:

- Model lưu bằng joblib
- Phiên bản: trainbase_23102025

Sau khi huấn luyện được bộ siêu tham số:

```
Best parameters: {'logreg_c': 10, 'logreg_penalty': 'l1', 'logreg_solver': 'liblinear'}
Best cross-validation accuracy: 0.8619
```

4. Thực nghiệm và thảo luận

4.2 Random Forest

Pipeline & Hyperparameters (trước train):

- StandardScaler + RandomForestClassifier (random_state=42)
- GridSearchCV:
 - n_estimators: [50, 100, 200]
 - max_depth: [3,5,7,None]
 - min_samples_split: [2,5,10]
 - min_samples_leaf: [1,2,4]
 - max_features: ['sqrt','log2',None]
 - StratifiedKFold 5 folds
 - Metrics: accuracy, F1, ROC AUC

Best parameters (sau train):

```
Best parameters: {'rf__max_depth': 5, 'rf__max_features': None, 'rf__min_samples_leaf': 4, 'rf__min_samples_split': 2, 'rf__n_estimators': 100}
Best cross-validation accuracy: 0.8664
```

Ghi chú:

- Model lưu bằng joblib
- Phiên bản: trainbase_23102025

4. Thực nghiệm và thảo luận

4.3 XGBoost

Pipeline & Hyperparameters (trước train):

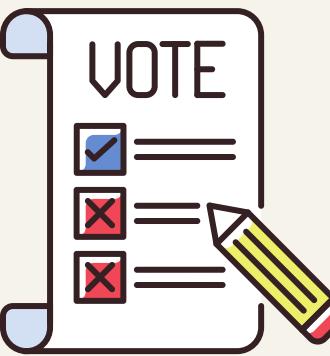
- StandardScaler + XGBClassifier (random_state=42, eval_metric='logloss')
- GridSearchCV:
 - n_estimators: [50,100,200]
 - max_depth: [3,5,7]
 - learning_rate: [0.01,0.1,0.2]
 - subsample: [0.8,0.9,1.0]
 - colsample_bytree: [0.8,0.9,1.0]
 - reg_alpha: [0,0.1,1]
 - reg_lambda: [1,10,100]
 - StratifiedKFold 5 folds
 - Metrics: accuracy, F1, ROC AUC

Best parameters (sau train):

```
Best parameters: {'xgb__colsample_bytree': 0.8, 'xgb__learning_rate': 0.01, 'xgb__max_depth': 5, 'xgb__n_estimators': 200,
'xgb__reg_alpha': 0, 'xgb__reg_lambda': 1, 'xgb__subsample': 1.0}
Best cross-validation accuracy: 0.8664
```

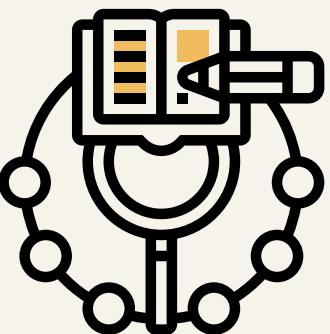
Ghi chú:

- Model lưu bằng joblib
- Phiên bản: trainbase_23102025



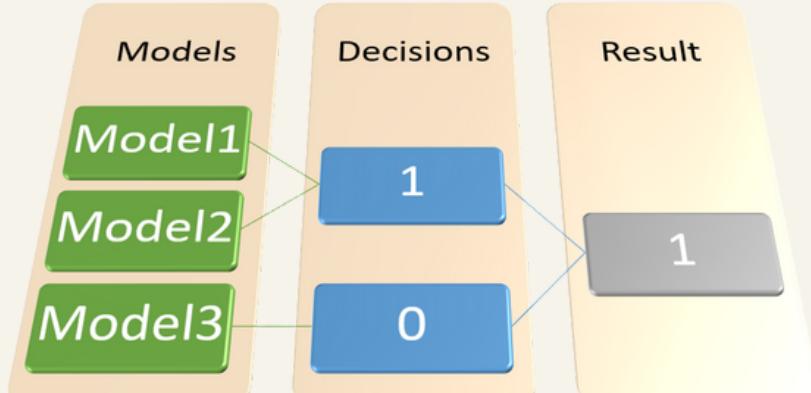
4. Thực nghiệm và thảo luận

4.4 Các phương pháp ensemble mô hình

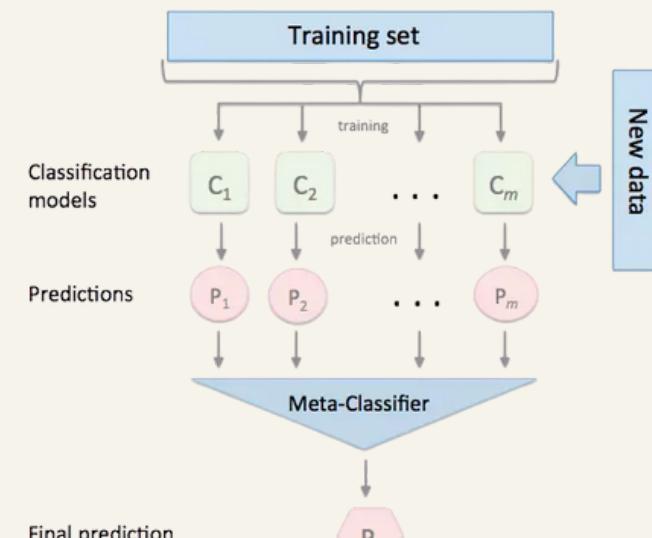


Tại sao dùng Ensemble (Motivation) ?

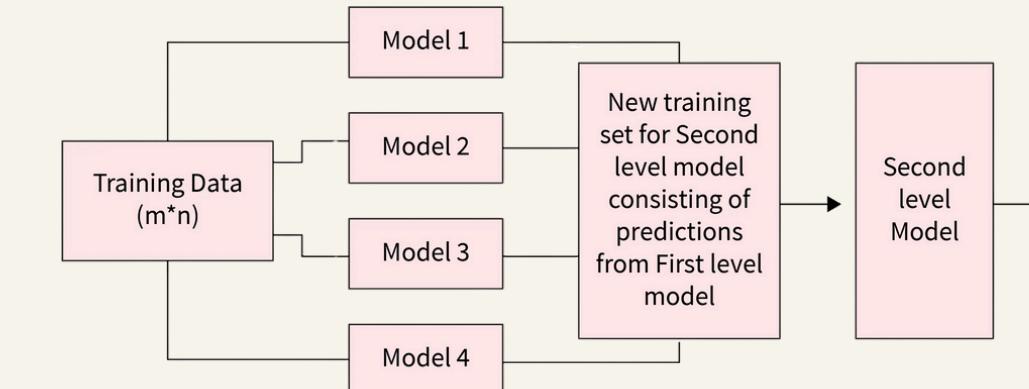
- Mỗi mô hình có điểm mạnh riêng → kết hợp giúp:
 - Giảm overfitting so với mô hình đơn lẻ.
 - Cải thiện độ chính xác dự đoán tổng thể.
 - Tăng tính “robust” của kết quả.
- Titanic có dữ liệu nhiều và ít feature → ensemble giúp cân bằng bias-variance.



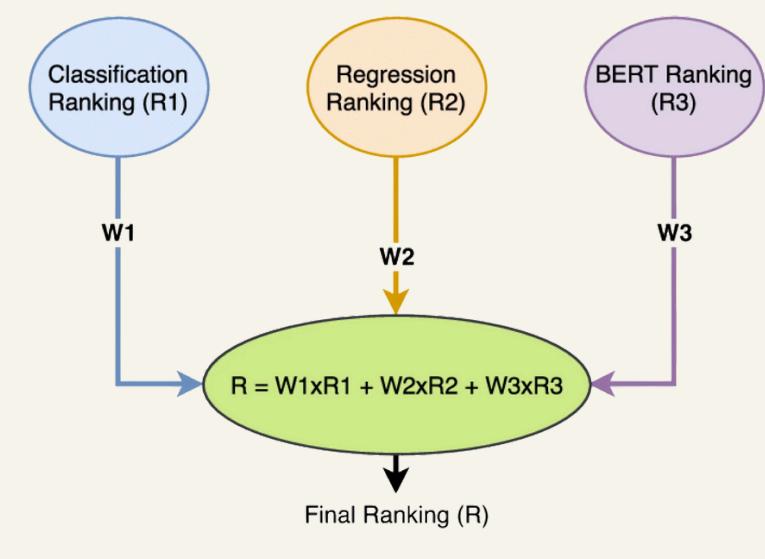
Voting



Stacking



Blending



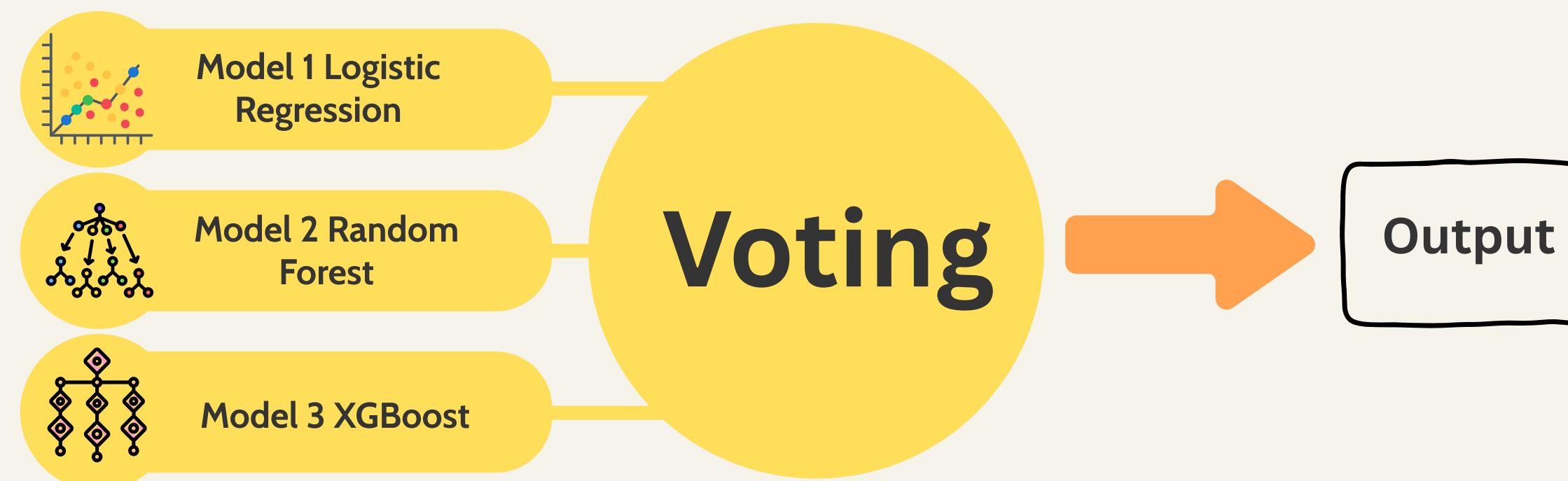
Weighted Average

4. Thực nghiệm và thảo luận

4.4 Các phương pháp ensemble mô hình

4.4.1 Voting Ensemble (Soft Voting)

Kết quả 

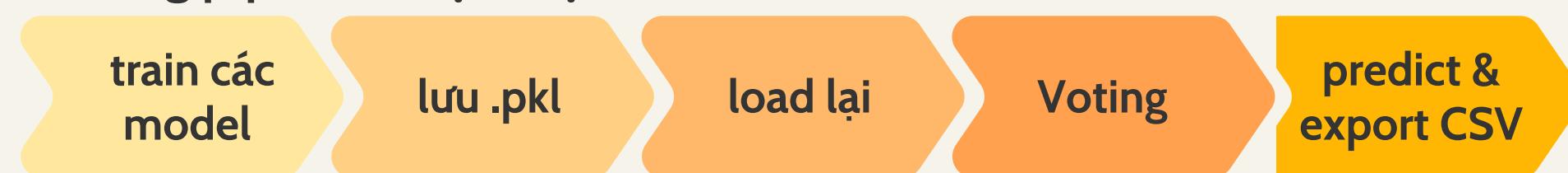


Prediction distribution:
Survived
0 260
1 158
Name: count, dtype: int64

Nhận xét 

→ Tỷ lệ dự đoán sống sót ≈ 37.8%, phù hợp với phân bố thật → mô hình hoạt động ổn định.

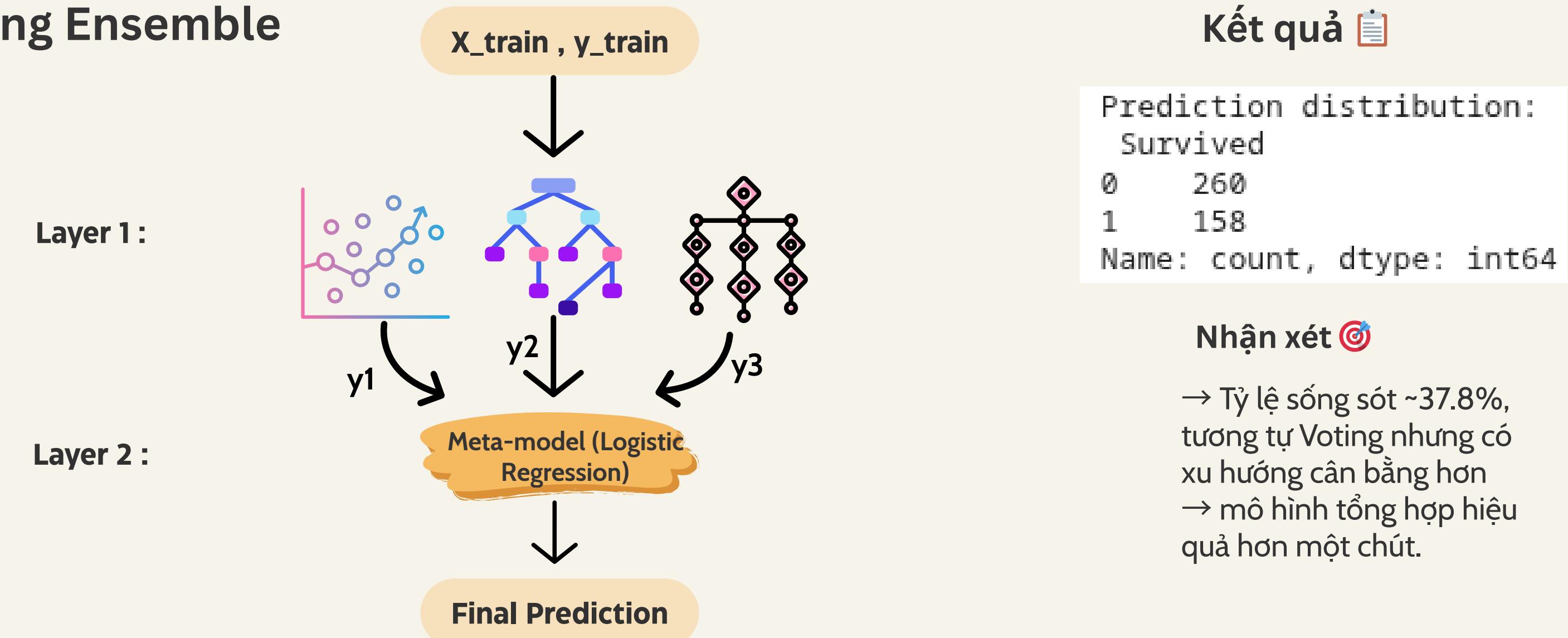
Đường pipeline thực hiện



4. Thực nghiệm và thảo luận

4.4 Các phương pháp ensemble mô hình

4.4.2 Stacking Ensemble



Đường pipeline thực hiện

train các
model

lưu .pkl

load lại

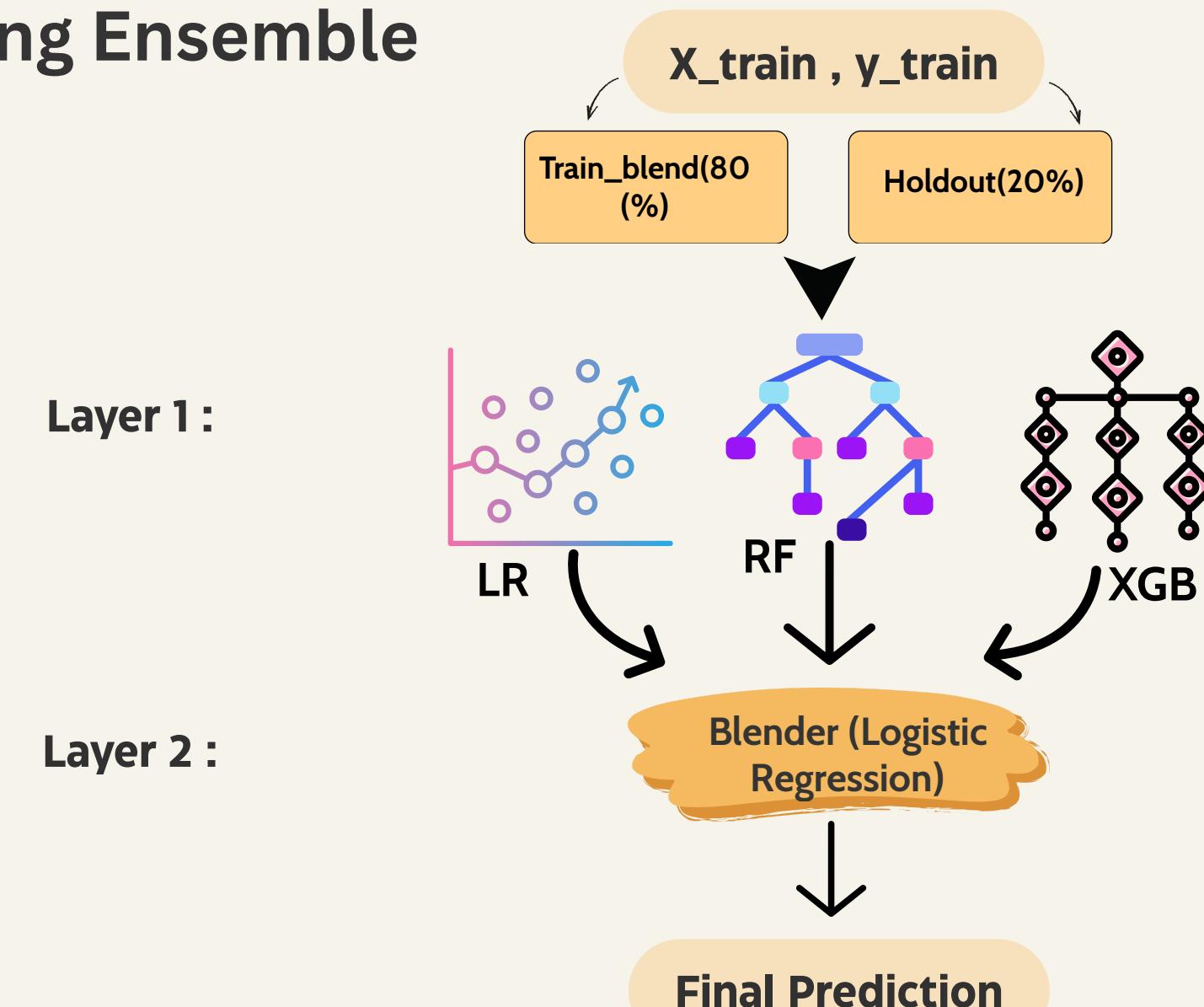
Stacking

predict &
export CSV

4. Thực nghiệm và thảo luận

4.4 Các phương pháp ensemble mô hình

4.4.3 Blending Ensemble



Đường pipeline thực hiện

train các
model

lưu .pkl

load lại

Blender

predict &
export CSV

Kết quả

Prediction distribution:

Survived

0 263

1 155

Name: count, dtype: int64

Nhận xét

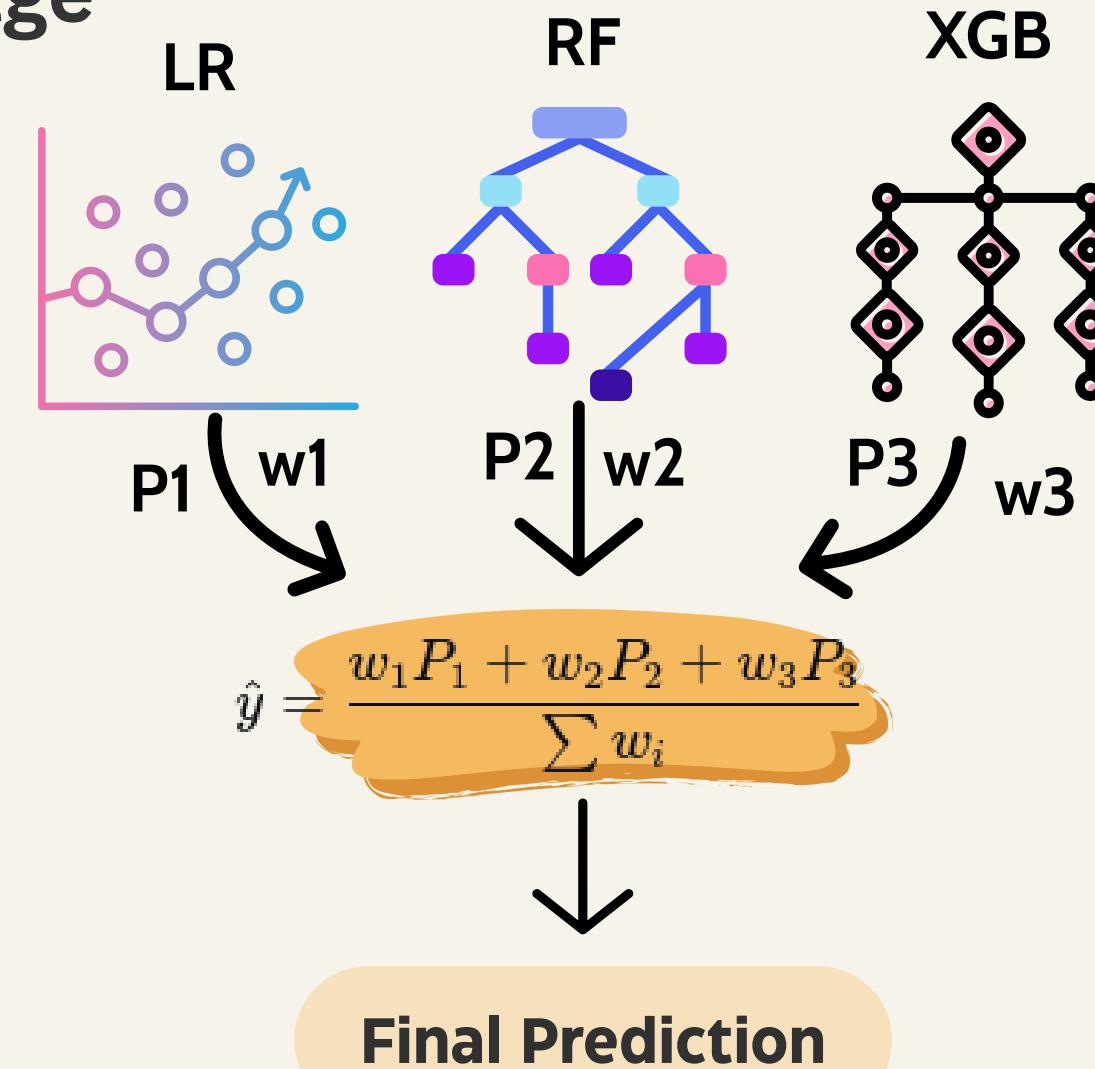
Blending dùng 1 tập holdout (validation riêng) để tạo meta features → đơn giản hơn, nhanh hơn nhưng tốn dữ liệu huấn luyện.

→ Tỷ lệ sống sót ~37.08%, tương tự Voting nhưng có xu hướng cân bằng hơn
→ mô hình tổng hợp hiệu quả hơn một chút.

4. Thực nghiệm và thảo luận

4.4 Các phương pháp ensemble mô hình

4.4.3 Weighted Average



Đường pipeline thực hiện

train các model

lưu .pkl

load lại

Weighted Average

predict & export CSV

Kết quả

Prediction distribution:

Survived	count
0	260
1	158

Name: count, dtype: int64

Nhận xét

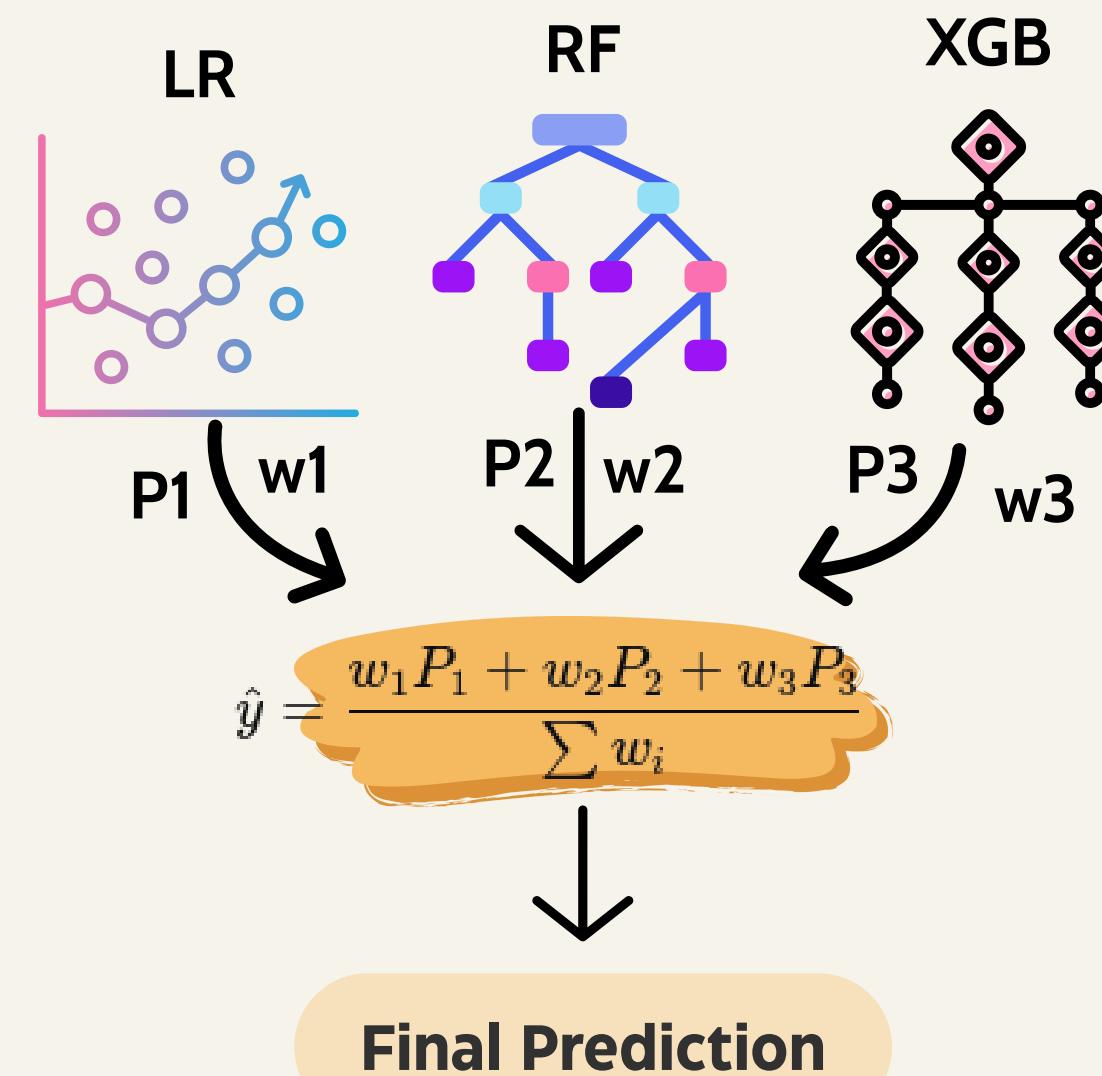
Weights: tỷ lệ với CV accuracy (5-fold)

- Ưu điểm: dễ hiểu, cân bằng giữa Voting và Stacking
- Nhược điểm: không học được quan hệ phi tuyến (meta-learning)

→ Tỷ lệ sống sót ~ 37.8%, gần giống Voting → mô hình ổn định, phản ánh độ tin cậy cao của ba model.

4. Thực nghiệm và thảo luận

4.4 Các phương pháp ensemble mô hình



Đường pipeline thực hiện

train các
model

lưu .pkl

load lại

Weighted
Average

predict &
export CSV

Kết quả

Prediction distribution:

Survived	count
0	260
1	158

Name: count, dtype: int64

Nhận xét

Weights: tỷ lệ với CV accuracy (5-fold)

- Ưu điểm: dễ hiểu, cân bằng giữa Voting và Stacking
- Nhược điểm: không học được quan hệ phi tuyến (meta-learning)

→ Tỷ lệ sống sót ~ 37.8%, gần giống Voting → mô hình ổn định, phản ánh độ tin cậy cao của ba model.

4. Thực nghiệm và thảo luận

4.5. Kết quả thực nghiệm

- Kết quả Đạt được

527	Haitran1207		0.80143
-----	-------------	--	---------

Nhận xét

- ◆ XGBoost đạt hiệu năng cao nhất nhờ khả năng học phi tuyến và regularization tốt.
- ◆ Các mô hình đều ổn định, phản ánh quy trình huấn luyện hiệu quả.
- ◆ Public score 0.80143 chứng minh tính tổng quát của mô hình trên dữ liệu thực tế.

- Kết quả độ đo thu được

Đánh giá các mô hình				
Mô hình	Accuracy	F1 score	ROC AUC	Public score
Logistic Regression	0,8698	0,8226	0,9208	0,79904
Random forest	0,8687	0,8157	0,9335	0,79425
XGBoost	0,8721	0,8202	0,9351	0,80143

5. Kết luận

Báo cáo khẳng định rằng Feature Engineering là động lực chính trong việc tối ưu hóa mô hình Titanic.

- **Xử lý missing values:** điền khuyết Age và Fare bằng median và loại bỏ cột Cabin vì khó xử lý.
- **Xử lý Feature Engineering:** FE tập trung vào khai thác Title và Family_Survival , sau đó chuẩn hóa (Scaling) và lựa chọn feature chiến lược để tinh chỉnh bộ feature.
- **Kết quả đạt được:** Mô hình Logistic Regression cho kết quả tốt nhất 0.80143 nhờ các feature được làm sạch và chuẩn hóa

Thanks For
Listening