

Tóm Tắt Kết Quả EDA Và Mô Hình

Khám phá dữ liệu (EDA):

1. Thông tin dữ liệu:

- Thông tin của dữ liệu của tập **train**:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column   Non-Null Count Dtype  
 --- 
 0   PassengerId 891 non-null    int64   
 1   Survived    891 non-null    int64   
 2   Pclass      891 non-null    int64   
 3   Name        891 non-null    object  
 4   Sex         891 non-null    object  
 5   Age         714 non-null    float64 
 6   SibSp       891 non-null    int64   
 7   Parch       891 non-null    int64   
 8   Ticket      891 non-null    object  
 9   Fare         891 non-null    float64 
 10  Cabin        204 non-null    object  
 11  Embarked     889 non-null    object  
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

Hiển thị số dòng (891), số cột (12), kiểu dữ liệu từng cột, và phát hiện giá trị thiếu ở cột Age, Cabin, Embarked.

- Thông tin của dữ liệu của tập **test**:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
 #   Column   Non-Null Count Dtype  
 --- 
 0   PassengerId 418 non-null    int64
```

```
1 Pclass    418 non-null      int64
2 Name     418 non-null      object
3 Sex      418 non-null      object
4 Age      332 non-null      float64
5 SibSp    418 non-null      int64
6 Parch    418 non-null      int64
7 Ticket   418 non-null      object
8 Fare     417 non-null      float64
9 Cabin    91 non-null       object
10 Embarked 418 non-null     object
dtypes: float64(2), int64(4), object(5)
memory usage: 36.1+ KB
```

Ta có thể thấy số dòng (418), số cột (11), kiểu dữ liệu, và phát hiện giá trị thiếu ở cột Age, Fare, Cabin.

2. Giá trị bị thiếu:

- Số lượng giá trị bị thiếu của tập **train**:

```
PassengerId 0
Survived     0
Pclass       0
Name         0
Sex          0
Age          177
SibSp        0
Parch        0
Ticket       0
Fare          0
Cabin        687
Embarked     2
dtype: int64
```

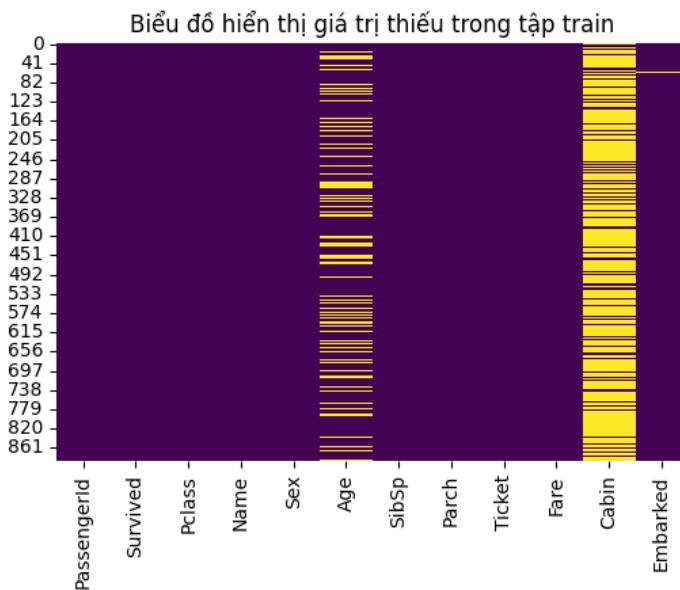
Xác nhận số lượng thiếu: **Age (177)**, **Cabin (687)**, **Embarked (2)**.

- Số lượng giá trị bị thiếu trong tập **test**:

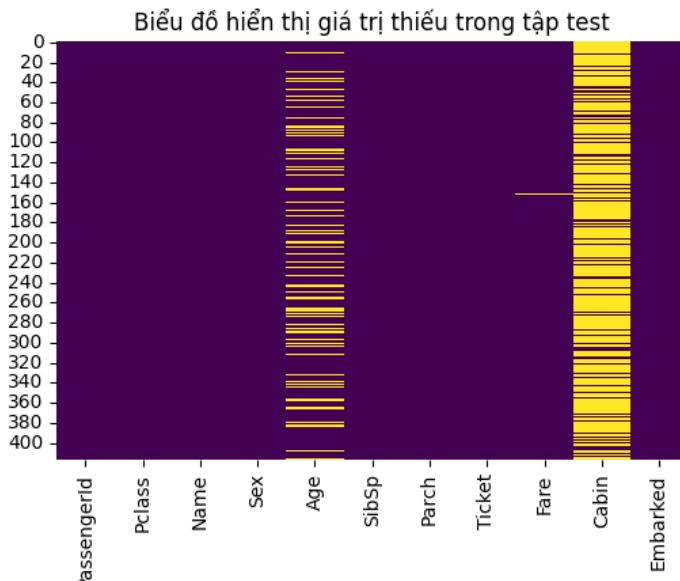
```
PassengerId    0  
Pclass         0  
Name          0  
Sex           0  
Age          86  
SibSp         0  
Parch         0  
Ticket        0  
Fare          1  
Cabin        327  
Embarked      0  
dtype: int64
```

Xác nhận số lượng thiếu: **Age (86), Cabin (327), Fare (1)**.

3. Xử lý dữ liệu bị thiếu:



Biểu đồ nhiệt cho ta thấy cột Cabin thiếu rất nhiều, Age thiếu rải rác, Embarked thiếu rất ít trong tập train.



Biểu đồ nhiệt cho ta thấy cột Cabin vẫn thiếu rất nhiều, cột Age vẫn rải rác và cột Fare thiếu cực ít.

Ta tiến hành xử lí dữ liệu bằng cách:

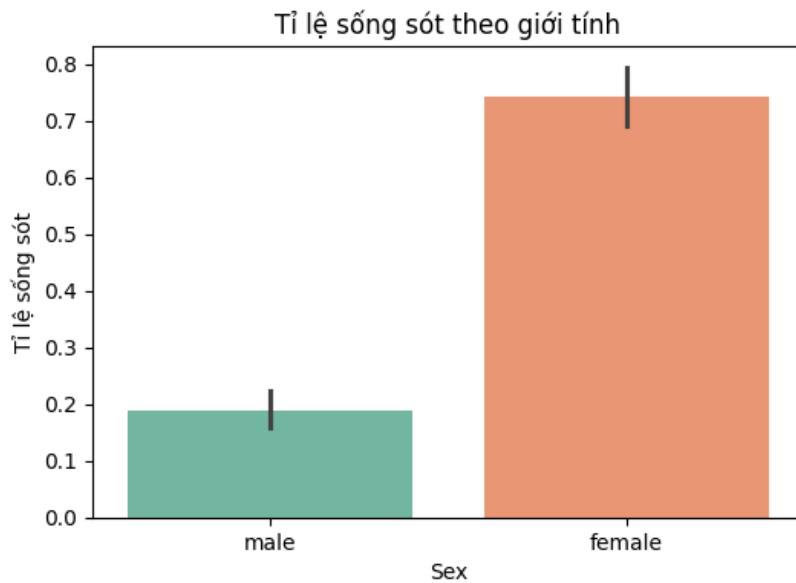
- + Điền các giá trị bị thiếu của Age bằng các giá trị trung bình (median).
- + Điền các giá trị bị thiếu của Embarked (của tập train) bằng các giá trị phổ biến (mode).
- + Điền các giá trị bị thiếu của Fare (của tập test) bằng các giá trị trung bình (median).
- + Nhận thấy cột cabin thiếu quá nhiều nên ta sẽ tiến hành xóa cột cabin.

Ta tiến hành kiểm tra lại dữ liệu của 2 tập train và test:

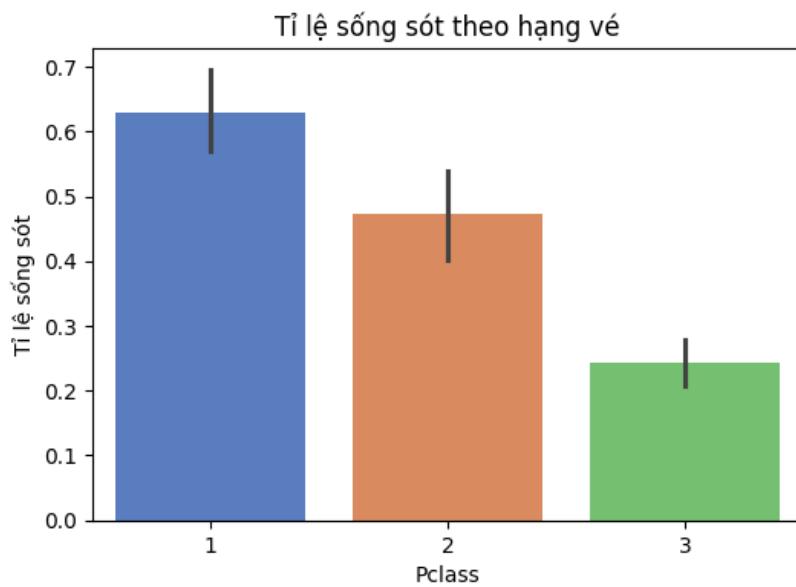
Số lượng giá trị bị thiếu trong tập train :		Số lượng giá trị bị thiếu trong tập test :	
PassengerId	0	PassengerId	0
Survived	0	Pclass	0
Pclass	0	Name	0
Name	0	Sex	0
Sex	0	Age	0
Age	0	SibSp	0
SibSp	0	Parch	0
Parch	0	Ticket	0
Ticket	0	Fare	0
Fare	0	Embarked	0
Embarked	0	dtype:	int64
dtype: int64			

Có thể thấy rằng không còn giá trị bị thiếu nào và đã xóa cột Cabin.

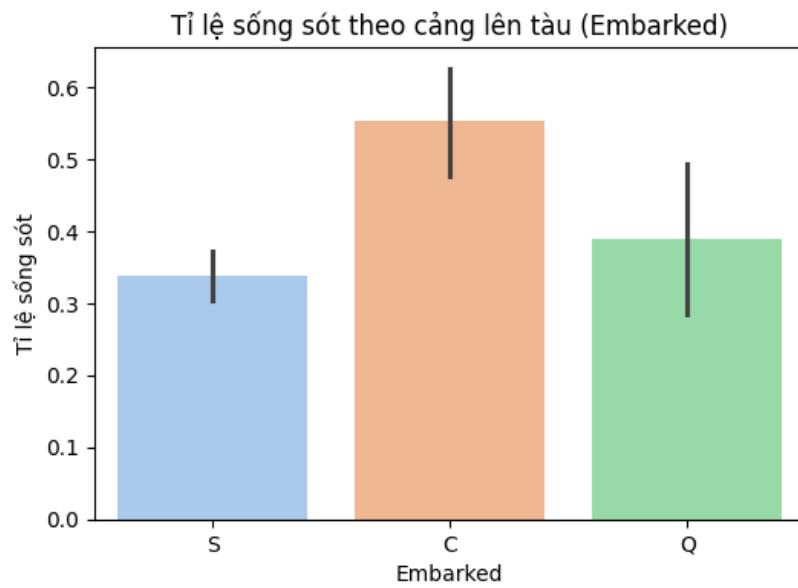
4. Trực quan hóa dữ liệu:



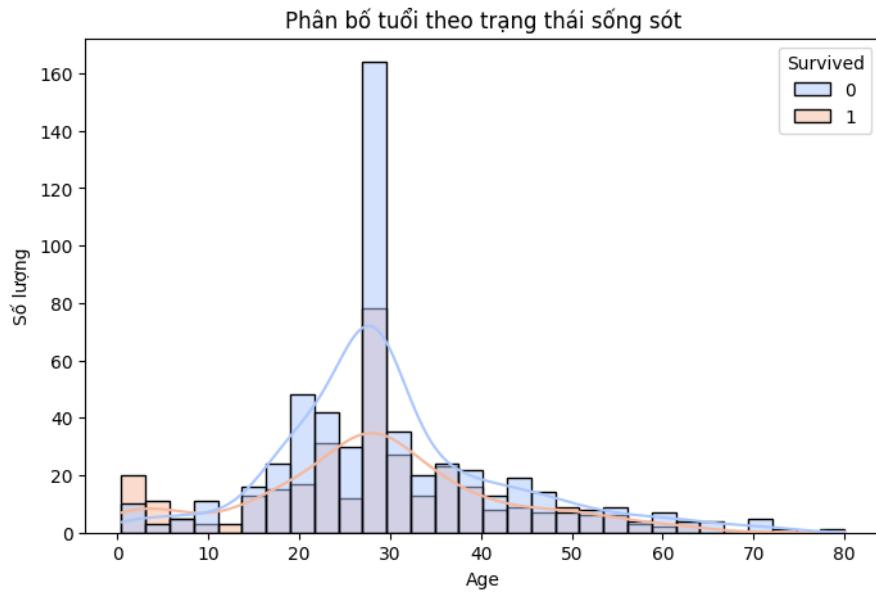
- Biểu đồ này cho thấy một sự khác biệt rất lớn về tỉ lệ sống sót giữa nam và nữ. Phụ nữ có tỉ lệ sống sót cao hơn nhiều (khoảng 74%), trong khi nam giới có tỉ lệ sống sót rất thấp (khoảng 19%).
- Điều này khẳng định mạnh mẽ quy tắc xã hội "phụ nữ và trẻ em trước" đã được áp dụng trong quá trình sơ tán, khiến giới tính trở thành một yếu tố dự đoán khả năng sống sót cực kỳ quan trọng.



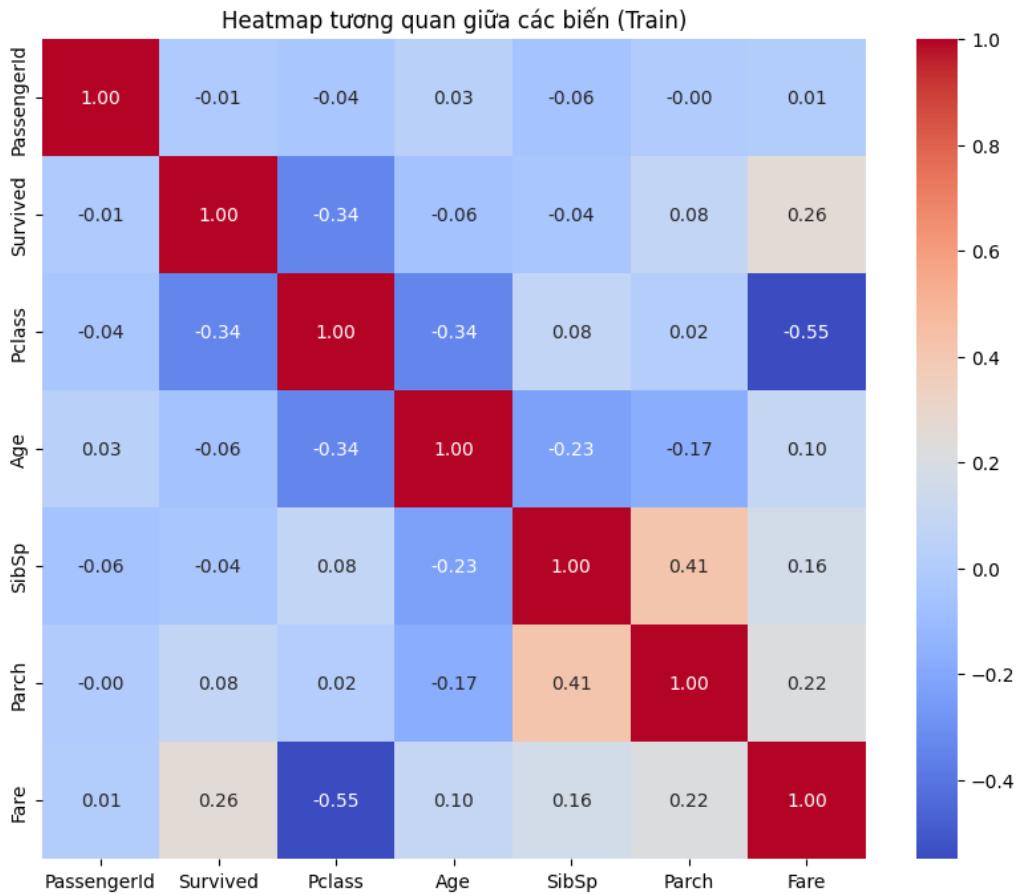
- Biểu đồ thể hiện rõ xu hướng tỉ lệ sống sót giảm dần theo hạng vé. Hành khách hạng 1 có tỉ lệ sống sót cao nhất (~63%), tiếp đến là hạng 2 (~47%), và hạng 3 có tỉ lệ thấp nhất (~24%).
- Hạng vé phản ánh địa vị kinh tế-xã hội và có thể liên quan đến vị trí cabin trên tàu (gần boong, gần thuyền cứu sinh hơn). Hành khách hạng cao hơn rõ ràng có cơ hội sống sót tốt hơn.



- Hành khách lên tàu từ Cherbourg (C) có tỉ lệ sống sót cao hơn (~55%) so với Queenstown (Q, ~39%) và Southampton (S, ~34%).
- Mỗi liên hệ này có thể không trực tiếp bằng giới tính hay hạng vé. Tỉ lệ sống sót cao hơn ở cảng C có thể là do thành phần hành khách lên từ cảng này khác biệt (ví dụ: tỉ lệ hành khách hạng 1 hoặc phụ nữ cao hơn).

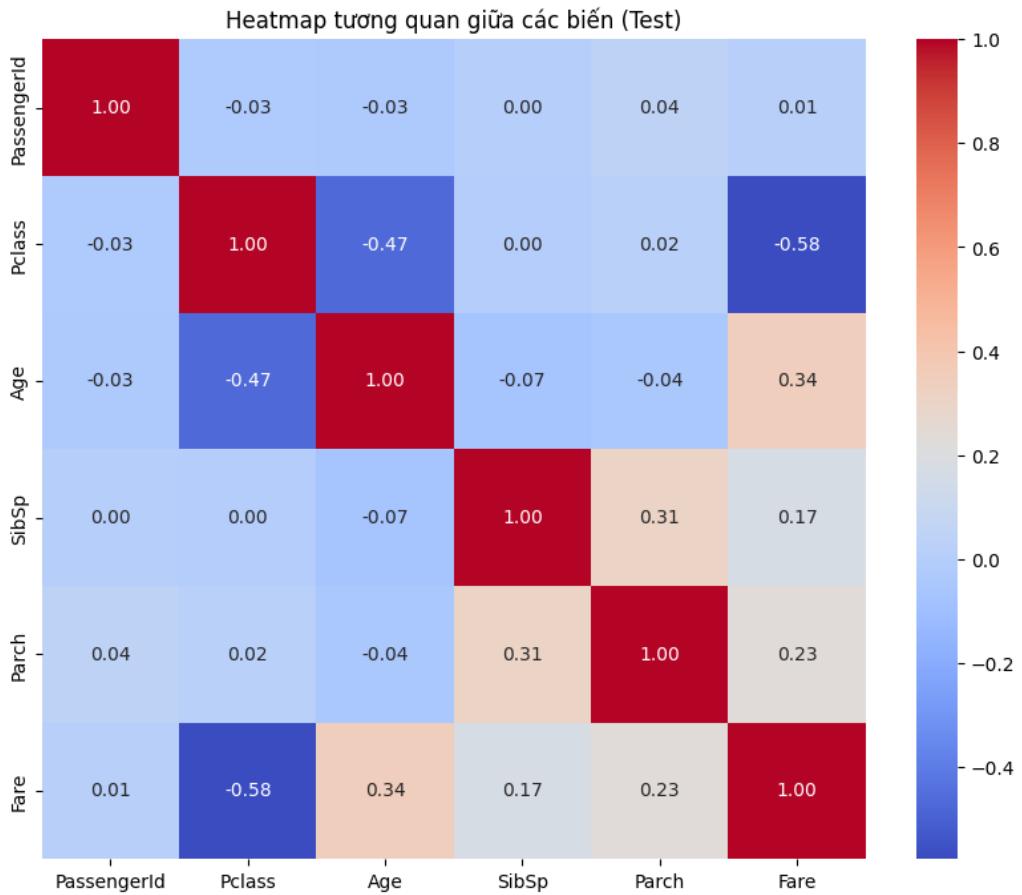


- Biểu đồ histogram cho thấy rõ ràng một đỉnh cao về số lượng người sống sót ở nhóm tuổi trẻ em (dưới 10 tuổi). Ngược lại, nhóm không sống sót có số lượng đông nhất ở độ tuổi thanh niên và trung niên (khoảng 20-40).
- Điều này củng cố giả thuyết trẻ em được ưu tiên cứu hộ hơn.



- **Survived (biến mục tiêu)** có tương quan âm đáng kể với Pclass (-0.34). Điều này có nghĩa là khi hạng vé tăng lên (từ 1 -> 2 -> 3), khả năng sống sót giảm xuống.
- Survived có tương quan dương vừa phải với Fare (0.26). Hành khách trả giá vé cao hơn có xu hướng sống sót cao hơn.
- Pclass và Fare có tương quan âm mạnh (-0.55). Hạng vé càng thấp (số Pclass càng lớn) thì giá vé (Fare) càng thấp, điều này rất hợp lý.
- Age có tương quan âm với Pclass (-0.34), cho thấy hành khách ở hạng cao hơn (Pclass nhỏ) có xu hướng lớn tuổi hơn một chút.
- SibSp (số anh chị em/vợ chồng) và Parch (số bố mẹ/con cái) có tương quan dương (0.41), phản ánh việc hành khách thường đi cùng nhiều loại người thân (theo nhóm gia đình).

Heatmap này giúp nhanh chóng xác định các mối quan hệ tuyến tính giữa các biến số. Nó làm nổi bật Pclass và Fare là các yếu tố số có liên quan rõ ràng đến khả năng sống sót. Các mối quan hệ giữa các biến độc lập như Pclass/Fare hay SibSp/Parch cũng được thể hiện rõ.



- Mỗi tương quan âm mạnh giữa Pclass và Fare (-0.58) vẫn rất rõ ràng, tương tự như trong tập train.
- Tương quan âm giữa Age và Pclass (-0.47) có vẻ mạnh hơn một chút so với tập train.
- Tương quan dương giữa SibSp và Parch (0.31) vẫn tồn tại nhưng yếu hơn so với tập train.
- Các mối tương quan khác giữa các biến độc lập nhìn chung khá yếu và tương đối giống với tập train (ví dụ: PassengerId hầu như không tương quan với các biến khác).

Heatmap này chủ yếu dùng để kiểm tra xem cấu trúc tương quan giữa các biến độc lập có nhất quán giữa tập train và tập test hay không. Kết quả cho thấy sự tương đồng khá cao (đặc biệt là mối quan hệ Pclass/Fare), điều này là tốt, cho thấy dữ liệu trong hai tập có cấu trúc tương tự nhau, giúp mô hình học được từ tập train có thể áp dụng tốt trên tập test.

Chuẩn bị và xử lý đặc trưng:

1. Tạo đặc trưng mới:

- Tính toán **FamilySize** (kích thước gia đình) và đặc trưng nhị phân **IsAlone** (đi một mình) từ các cột SibSp và Parch, giúp mô hình đánh giá tầm quan trọng của việc đi theo nhóm.
- Trích xuất Title (danh xưng) từ cột Name bằng regex và chuẩn hóa các danh xưng ít phổ biến (như Mlle, Mme) để giảm sự phân loại và tăng cường ý nghĩa.

2. Chọn các cột cần thiết:

- Chọn 8 cột quan trọng (Pclass, Sex, Age, Fare, Embarked, FamilySize, IsAlone, Title) để đưa vào huấn luyện mô hình. Xác định biến mục tiêu (**\$y\$**): Cột Survived được chọn làm biến mục tiêu, đại diện cho kết quả cần dự đoán (0 hoặc 1).

3. Định nghĩa Pipeline Tiền xử lý:

- Định nghĩa num_pipe để điền khuyết bằng trung vị (median) và chuẩn hóa (StandardScaler) cho các cột định lượng (Age, Fare, FamilySize).
- Định nghĩa cat_pipe để điền khuyết bằng giá trị thường xuyên nhất (most_frequent) và mã hóa One-Hot cho các cột danh mục.
- Sử dụng ColumnTransformer để áp dụng hai pipeline xử lý khác nhau cho hai nhóm cột tương ứng.

4. Huấn luyện và Lưu Pipeline:

- Lệnh `preprocessor.fit(X)` tính toán các thông số cần thiết (như median, mode, mean, std dev, danh mục duy nhất) từ dữ liệu huấn luyện X.
- Sử dụng `joblib.dump()` để lưu toàn bộ đối tượng preprocessor đã fit vào tệp `preprocessor.pkl`.
- Đảm bảo cùng một phép biến đổi chính xác sẽ được áp dụng cho dữ liệu kiểm tra (test data) hoặc dữ liệu mới sau này, duy trì tính nhất quán.

Huấn luyện mô hình và báo cáo:

1. Xác định cột nhãn (thay “target” bằng tên cột thực tế):

- Tách dữ liệu thành biến độc lập (features) và biến mục tiêu (target variable) để chuẩn bị cho quá trình huấn luyện mô hình:

`target_col = 'Survived'`:

- Xác định cột đích (biến mục tiêu) là “Survived”, biểu thị hành vi cần dự đoán (0 = không sống sót, 1 = sống sót).

`X = data.drop(columns=[target_col])`:

- Lưu toàn bộ các đặc trưng đầu vào (features) — tức là những cột dùng để dự đoán kết quả (như Age, Sex, Fare,...).

y = data[target_col]:

- Lưu biến mục tiêu (label) mà mô hình sẽ học để dự đoán.

Mục tiêu của đoạn code này là tách dữ liệu thành hai phần:

+ Biến đầu vào (X) – chứa các đặc trưng (features) dùng để dự đoán.

+ Biến đầu ra (y) – chứa giá trị mục tiêu (target) mà mô hình cần dự đoán.

2. Nếu nhãn là chữ, ta mã hóa lại:

Ta tiến hành chuyển đổi các nhãn (label) trong biến mục tiêu y từ dạng chuỗi (categorical) sang dạng số (numeric) để các mô hình học máy có thể xử lý được.

3. Bỏ cột không cần thiết:

Ta tiến hành loại bỏ các cột không cần thiết hoặc không có giá trị dự đoán rõ ràng trong quá trình huấn luyện mô hình.

Những cột này thường mang tính định danh, mô tả hoặc chứa nhiều giá trị thiếu, nên không giúp ích cho việc học của mô hình.

4. Đullen giá trị bị thiếu:

Xử lý các giá trị bị thiếu (NaN) trong hai cột "Age" và "Embarked" bằng cách thay thế chúng bằng giá trị phù hợp nhất, giúp mô hình có thể huấn luyện trơn tru mà không gặp lỗi.

5. Mã hóa cột chữ:

Ta thực hiện chuyển đổi các giá trị dạng chuỗi (categorical variables) trong hai cột "Sex" và "Embarked" thành giá trị số (numeric) để mô hình học máy có thể xử lý được.

6. Chia dữ liệu:

- **Tập huấn luyện (training set):** dùng để “dạy” mô hình học cách dự đoán.

- **Tập kiểm tra (test set):** dùng để đánh giá độ chính xác của mô hình trên dữ liệu chưa thấy trước đó.

7. Chuẩn hóa:

Chuẩn hóa (scale) các đặc trưng đầu vào sao cho chúng có trung bình bằng 0 và độ lệch chuẩn bằng 1, giúp các mô hình học máy hoạt động hiệu quả và ổn định hơn.

8. Các mô hình:

Ta tiến hành khởi tạo và lưu trữ nhiều mô hình học máy khác nhau trong một từ điển (dictionary) để so sánh hiệu năng của chúng trên cùng một tập dữ liệu.

9. Huấn luyện và đánh giá:

- Huấn luyện các mô hình đã khởi tạo, sau đó đánh giá hiệu suất của từng mô hình bằng cách đo độ chính xác (Accuracy), ma trận nhầm lẫn (Confusion Matrix) và báo cáo phân loại (Classification Report).

- Kết quả của từng mô hình được lưu vào dictionary results để dễ so sánh.

- ◆ Logistic Regression

Độ chính xác: 0.8045

Confusion Matrix:

[[90 15]

[20 54]]

Báo cáo phân loại:

	precision	recall	f1-score	support
--	-----------	--------	----------	---------

0	0.82	0.86	0.84	105
1	0.78	0.73	0.76	74

accuracy		0.80	179	
----------	--	------	-----	--

macro avg	0.80	0.79	0.80	179
-----------	------	------	------	-----

weighted avg	0.80	0.80	0.80	179
--------------	------	------	------	-----

=====

- ◆ Decision Tree

Độ chính xác: 0.7821

Confusion Matrix:

[[83 22]

[17 57]]

Báo cáo phân loại:

	precision	recall	f1-score	support
--	-----------	--------	----------	---------

...

accuracy		0.80	179	
----------	--	------	-----	--

macro avg	0.80	0.79	0.80	179
-----------	------	------	------	-----

weighted avg	0.80	0.80	0.80	179
--------------	------	------	------	-----

10. Tổng kết:

Tổng hợp và xuất kết quả đánh giá các mô hình học máy ra một file CSV (experiments_summary.csv) để dễ dàng lưu trữ, so sánh và trình bày trong báo cáo.

TỔNG HỢP KẾT QUẢ:

Logistic Regression	: 0.8045
Decision Tree	: 0.7821
Random Forest	: 0.8212
SVM	: 0.8156
KNN	: 0.8045

Đã tạo file experiments_summary.csv thành công!

	Model	Accuracy
0	Logistic Regression	0.804469
1	Decision Tree	0.782123
2	Random Forest	0.821229
3	SVM	0.815642
4	KNN	0.804469

Kết hợp mô hình và đánh giá cuối:

Hyperparameter Tuning:

Tối ưu siêu tham số cho các mô hình baseline bằng GridSearchCV và RandomizedSearchCV.

1. Tiền xử lý dữ liệu:

- Thực hiện tiền xử lý dữ liệu cơ bản, bao gồm điền khuyết (impute) giá trị thiếu của Age bằng trung vị và Embarked bằng mode, sau đó mã hóa (LabelEncoder) các cột phân loại.
- Tách tập dữ liệu huấn luyện (train) thành tập huấn luyện (X_train) và kiểm tra/thẩm định (X_val) với tỷ lệ 80/20.
- Sử dụng StandardScaler để chuẩn hóa các đặc trưng số, đảm bảo dữ liệu được chuẩn bị sẵn sàng cho việc huấn luyện mô hình.

2. Định nghĩa mô hình và tham số tuning:

- Xác định một danh sách 4 mô hình học máy cơ bản để so sánh: **Logistic Regression**, **Random Forest**, **Support Vector Machine (SVM)**, và **XGBoost**.
- Thiết lập khoảng tìm kiếm siêu tham số (params) cho từng mô hình, tập trung vào các tham số quan trọng như C (đối với LogReg/SVM), n_estimators, và max_depth (đối với RF/XGB).

3. Tuning mô hình bằng RandomizedSearchCV:

Tuning LogReg ...

```
C:\Users\thanh\AppData\Roaming\Python\Python311\site-packages\sklearn\model_selection\_search.py:317: UserWarning: The total space of parameters 3 is smaller than n_iter=5. Running 3 iterations. For exhaustive searches, use GridSearchCV.
```

```
warnings.warn(
```

```
→ LogReg accuracy: 0.7989
```

Tuning RF ...

```
→ RF accuracy: 0.8045
```

Tuning SVM ...

```
→ SVM accuracy: 0.8156
```

Tuning XGB ...

```
→ XGB accuracy: 0.8156
```



Tổng hợp kết quả:

LogReg: 0.7989

RF: 0.8045

SVM: 0.8156

XGB: 0.8156



Best model: SVC → 0.8156

```
C:\Users\thanh\AppData\Roaming\Python\Python311\site-packages\xgboost\training.py:199: UserWarning: [21:21:34] WARNING: C:\actions-runner\_work\xgboost\xgboost\src\learner.cc:790:  
Parameters: { "use_label_encoder" } are not used.
```

```
bst.update(dtrain, iteration=i, fobj=obj)
```

- Sử dụng RandomizedSearchCV với 5 lần lặp và 3-fold Cross-Validation để tìm kiếm hiệu quả các siêu tham số tối ưu cho từng mô hình.
- Đánh giá hiệu suất của mô hình đã tinh chỉnh bằng cách dự đoán trên tập kiểm tra/thẩm định (X_{val}) và tính toán độ chính xác (accuracy_score).
- Xác định và lưu trữ mô hình có độ chính xác cao nhất trên tập kiểm tra/thẩm định làm best_model.

4. Lưu mô hình tốt nhất:

Saved best model → best_model.pkl

- Mô hình tốt nhất được chọn (**best_model**) sau quá trình tìm kiếm siêu tham số sẽ được lưu vào đĩa dưới dạng tệp best_model.pkl bằng thư viện pickle.

- Việc này giúp tái sử dụng mô hình này sau này, không cần chạy lại quá trình tinh chỉnh, đảm bảo tính nhất quán và hiệu quả.

Ensemble & Evaluation:

1. Tải dữ liệu và chuẩn bị mô hình:

Thực hiện Voting, Stacking và tạo file `submission.csv`

- Tiền xử lý dữ liệu train và test theo cách thức đơn giản tương tự như notebook trước (điền khuyết và LabelEncoder).

- Dữ liệu kiểm tra (test data) được chuẩn hóa (StandardScaler) bằng cách áp dụng các thông số đã học từ dữ liệu huấn luyện.

- Tải mô hình tốt nhất đã được lưu từ bước trước (best_model.pkl) để sử dụng và so sánh.

2. Ensemble models (Voting + Stacking):

- Định nghĩa các mô hình cơ sở (base estimators) với các siêu tham số đã được chọn/tinh chỉnh (ví dụ: RandomForestClassifier, XGBClassifier, SVC, LogisticRegression).

- Xây dựng mô hình tổ hợp VotingClassifier (sử dụng voting='soft', dựa trên xác suất) để kết hợp kết quả từ các mô hình cơ sở.

- Xây dựng mô hình tổ hợp StackingClassifier, trong đó kết quả dự đoán của các mô hình cơ sở được sử dụng làm đầu vào cho mô hình cuối cùng (final_estimator), ở đây là LogisticRegression.

3. Huấn luyện và đánh giá mô hình:

bst.update(dtrain, iteration=i, fobj=obj)

Stacking: 0.8101

Best ensemble model: Voting → 0.8268

- Huấn luyện các mô hình tổ hợp (VotingClassifier, StackingClassifier) cùng với Best_Model đã tải trên tập huấn luyện (X_train).

- Đánh giá độ chính xác của cả ba mô hình trên tập kiểm tra/thẩm định (X_val) để xác định mô hình tổ hợp (hoặc mô hình đơn) nào mang lại kết quả tốt nhất.

- Xác định mô hình chiến thắng (best_ens_model) dựa trên độ chính xác cao nhất trên tập kiểm tra/thẩm định.

4. Dự đoán Cuối cùng và Tạo Tập Submission:

Saved submission.csv

	PassengerId	Survived
0	892	0
1	893	0
2	894	0
3	895	0
4	896	1

- Sử dụng mô hình có hiệu suất tốt nhất (final_model) để thực hiện dự đoán trên dữ liệu kiểm tra (test data) hoàn toàn chưa thấy (X_test).
- Tạo DataFrame kết quả với cột PassengerId và Survived (dự đoán).