Báo Cáo: Titanic - Machine Learning from Disaster

Tác giả: Phan Đức Nhân

Mã số sinh viên : 3123410243

Lớp : DCT1236

Ngày nộp: 18/09/2025

Mục đích: Báo cáo này trình bày quá trình áp dụng Machine Learning để dự đoán tỷ lệ sống sót của hành khách trên tàu Titanic.

# 1.Mô tả bài toán

Bài toán Titanic - Machine Learning from Disaster là một cuộc thi kinh điển trên nền tảng Kaggle, được thiết kế để giới thiệu người mới với Data Science và Machine Learning.

* **Mô tả chính:** Vào ngày 15/04/1912, tàu Titanic chìm trong chuyến đi đầu tiên, dẫn đến cái chết của hơn 1.500 người trong số 2.224 hành khách và thủy thủ đoàn. Nhiệm vụ là xây dựng mô hình Machine Learning để dự đoán xem một hành khách có sống sót (Survived = 1) hay không (Survived = 0) dựa trên các đặc trưng cá nhân như tuổi, giới tính, lớp vé, v.v.
* **Mục tiêu:** Sử dụng tập dữ liệu huấn luyện (train set) để xây dựng mô hình, sau đó dự đoán trên tập kiểm tra (test set) và đánh giá bằng độ chính xác (accuracy score) trên Kaggle leaderboard.
* **Thách thức:** Dữ liệu có nhiều giá trị thiếu (missing values), cần xử lý làm sạch và feature engineering để cải thiện hiệu suất mô hình.

Bài toán thuộc loại **phân loại nhị phân (binary classification)**,

# 2.Dữ liệu bài toán

Dữ liệu được cung cấp bởi Kaggle, bao gồm hai file chính:

* **train.csv:** Tập huấn luyện với 892 hành khách, bao gồm nhãn (Survived).
* **test.csv:** Tập kiểm tra với 418 hành khách, không có nhãn (dùng để dự đoán).

### **Đặc Trưng (Features) Chính:**

| **Tên Cột** | **Mô Tả** | **Kiểu Dữ Liệu** | **Giá Trị Thiếu (Train)** |
| --- | --- | --- | --- |
| PassengerId | ID hành khách | Integer | 0 |
| Survived | Sống sót (0: Không, 1: Có) | Integer | 0 (target) |
| Pclass | Lớp vé (1: Cao, 2: Trung, 3: Thấp) | Integer | 0 |
| Name | Tên hành khách | String | 0 |
| Sex | Giới tính (male/female) | String | 0 |
| Age | Tuổi | Float | ~177 (20%) |
| SibSp | Số anh chị em/vợ chồng trên tàu | Integer | 0 |
| Parch | Số cha mẹ/con cái trên tàu | Integer | 0 |
| Ticket | Số vé | String | 0 |
| Fare | Giá vé | Float | 0 |
| Cabin | Số cabin | String | ~687 (77%) |
| Embarked | Cảng lên tàu (C/Q/S) | String | 2 |

* **Target:** Survived (binary: 0 hoặc 1).
* **Kích thước:** Train: 892 hàng x 12 cột; Test: 418 hàng x 11 cột (không có Survived).
* **Nguồn:** [Kaggle Titanic Competition](https://www.kaggle.com/competitions/titanic/data).

# 3.Phân tích dữ liệu bài toán

* Code tải và kiểm tra dữ liệu

| import numpy as np  import pandas as pd  import matplotlib.pyplot as plt import seaborn as sns  import os for dirname, \_, filenames in os.walk('/kaggle/input'):  for filename in filenames:  print(os.path.join(dirname, filename)) |
| --- |

* Kết quả train\_data(5 hàng đầu tiên)

| train\_data = pd.read\_csv("/kaggle/input/titanic/train.csv") train\_data.head() |
| --- |

PassengerId Survived Pclass Name Sex Age SibSp Parch Ticket Fare Cabin Embarked

0 1 0 3 Braund, Mr. Owen Harris male 22.0 1 0 A/5 21171 7.2500 NaN S

1 2 1 1 Cumings, Mrs. John Bradley (Florence Briggs Th...) female 38.0 1 0 PC 17599 71.2833 C85 C

2 3 1 3 Heikkinen, Miss. Laina female 26.0 0 0 STON/O2. 3101282 7.9250 NaN S

3 4 1 1 Futrelle, Mrs. Jacques Heath (Lily May Peel) female 35.0 1 0 113803 53.1000 C123 S

4 5 0 3 Allen, Mr. William Henry male 35.0 0 0 373450 8.0500 NaN S

* Kết quả test\_data(5 hàng đầu tiên)

| test\_data = pd.read\_csv("/kaggle/input/titanic/test.csv") test\_data.head() |
| --- |

PassengerId Survived Pclass Name Sex Age SibSp Parch Ticket Fare Cabin Embarked

0 1 0 3 Braund, Mr. Owen Harris male 22.0 1 0 A/5 21171 7.2500 NaN S

1 2 1 1 Cumings, Mrs. John Bradley (Florence Briggs Th...) female 38.0 1 0 PC 17599 71.2833 C85 C

2 3 1 3 Heikkinen, Miss. Laina female 26.0 0 0 STON/O2. 3101282 7.9250 NaN S

3 4 1 1 Futrelle, Mrs. Jacques Heath (Lily May Peel) female 35.0 1 0 113803 53.1000 C123 S

4 5 0 3 Allen, Mr. William Henry male 35.0 0 0 373450 8.0500 NaN S

* Thông tin dữ liệu

| train\_data.info(); |
| --- |

<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

| train\_data.isna().sum() |
| --- |

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

| train\_data |
| --- |

| **Survived** | **Pclass** | **Sex** | **Age** | **SibSp** | **Parch** | **Fare** | **Embarked** |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 0 | 3 | male | 22.000000 | 1 | 0 | 7.2500 | S |
| **1** | 1 | 1 | female | 38.000000 | 1 | 0 | 71.2833 | C |
| **2** | 1 | 3 | female | 26.000000 | 0 | 0 | 7.9250 | S |
| **3** | 1 | 1 | female | 35.000000 | 1 | 0 | 53.1000 | S |
| **4** | 0 | 3 | male | 35.000000 | 0 | 0 | 8.0500 | S |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **886** | 0 | 2 | male | 27.000000 | 0 | 0 | 13.0000 | S |
| **887** | 1 | 1 | female | 19.000000 | 0 | 0 | 30.0000 | S |
| **888** | 0 | 3 | female | 29.699118 | 1 | 2 | 23.4500 | S |
| **889** | 1 | 1 | male | 26.000000 | 0 | 0 | 30.0000 | C |
| **890** | 0 | 3 | male | 32.000000 | 0 | 0 | 7.7500 | Q |

| test\_data.isna().sum() |
| --- |

Pclass 0

Sex 0

Age 86

SibSp 0

Parch 0

Fare 1

Embarked 0

dtype: int64

* Làm sạch dữ liệu

| test\_data['Age'] = test\_data['Age'].fillna(test\_data['Age'].mean()) test\_data['Fare'] = test\_data['Fare'].fillna(test\_data['Fare'].mean()) |
| --- |

* Thông tin dữ liệu

| print(train\_data.isna().sum().sum()) print(test\_data.isna().sum().sum()) |
| --- |

| print(train\_data.select\_dtypes(include=['object']).columns) |
| --- |

Index(['Sex', 'Embarked'], dtype='object')

* Mã hóa categorical features:

| train\_data['Sex'] = train\_data['Sex'].map({'male': 0, 'female': 1}) test\_data['Sex'] = test\_data['Sex'].map({'male': 0, 'female': 1}) |
| --- |

| train\_data = pd.get\_dummies(train\_data, columns=['Embarked'], drop\_first=True) test\_data = pd.get\_dummies(test\_data, columns=['Embarked'], drop\_first=True) |
| --- |

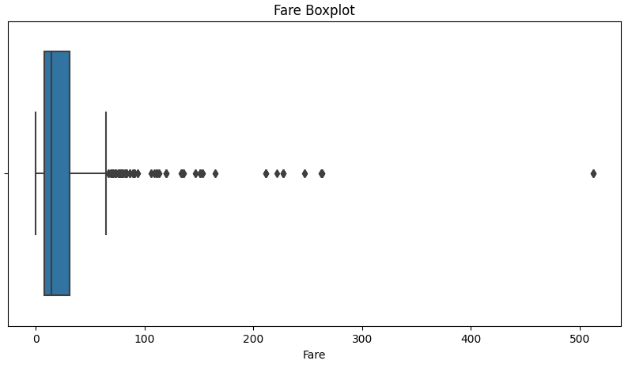
* Thông tin dữ liệu

| print(train\_data.select\_dtypes(include=['object']).columns) |
| --- |

Index([], dtype='object'

* Trực quan hóa Fare(kiểm tra phân phối)

| plt.figure(figsize=(10,5)) sns.boxplot(x=train\_data['Fare']) plt.title('Fare Boxplot') plt.show() |
| --- |



* Chuẩn hóa Fare và Age :

| train\_data['Fare'] = train\_data['Fare'].apply(np.log1p) test\_data['Fare'] = test\_data['Fare'].apply(np.log1p) |
| --- |

* Huấn luyện (Train)

| from sklearn.preprocessing import StandardScaler scaler = StandardScaler() for col in ['Age', 'Fare']:  train\_data[col] = scaler.fit\_transform(train\_data[[col]])  test\_data[col] = scaler.transform(test\_data[[col]])   from sklearn.model\_selection import train\_test\_split  X = train\_data.drop('Survived', axis=1) y = train\_data['Survived']  X\_train, X\_val, y\_train, y\_val = train\_test\_split(  X, y, test\_size=0.2, random\_state=42, stratify=y )  from sklearn.linear\_model import LogisticRegression from sklearn.ensemble import RandomForestClassifier from xgboost import XGBClassifier from sklearn.metrics import accuracy\_score, f1\_score, roc\_auc\_score |
| --- |

* Huấn luyện và đánh giá 3 mô hình

| models = {  'Logistic Regression': LogisticRegression(random\_state=42, max\_iter=1000),  'Random Forest': RandomForestClassifier(random\_state=42, n\_jobs=-1, n\_estimators=200),  'XGBoost': XGBClassifier(use\_label\_encoder=False, eval\_metric='logloss', random\_state=42, n\_jobs=-1) }  for name, model in models.items():  model.fit(X\_train, y\_train)  y\_pred = model.predict(X\_val)  y\_prob = model.predict\_proba(X\_val)[:,1]   acc = accuracy\_score(y\_val, y\_pred)  f1 = f1\_score(y\_val, y\_pred)  roc\_auc = roc\_auc\_score(y\_val, y\_prob)   print(f"{name} Results:")  print(f" Accuracy: {acc:.4f}")  print(f" F1 Score: {f1:.4f}")  print(f" ROC AUC: {roc\_auc:.4f}")  print("-" \* 30) |
| --- |

Logistic Regression Results:

Accuracy: 0.8101

F1 Score: 0.7385

ROC AUC: 0.8387

------------------------------

Random Forest Results:

Accuracy: 0.8212

F1 Score: 0.7538

ROC AUC: 0.8345

------------------------------

XGBoost Results:

Accuracy: 0.7989

F1 Score: 0.7353

ROC AUC: 0.8080

------------------------------

# 4.Kết quả dự đoán

| best\_model = RandomForestClassifier(random\_state=42, n\_jobs=-1, n\_estimators=200) best\_model.fit(X\_train, y\_train)  y\_val\_pred = best\_model.predict(X\_val)  print("Actual values (first 5):") print(y\_val.head())  print("\nPredicted values (first 5):") print(pd.Series(y\_val\_pred).head |
| --- |

Actual values (first 5):

565 0

160 0

553 1

860 0

241 1

Name: Survived, dtype: int64

Predicted values (first 5):

0 0

1 0

2 0

3 0

4 1

dtype: int64

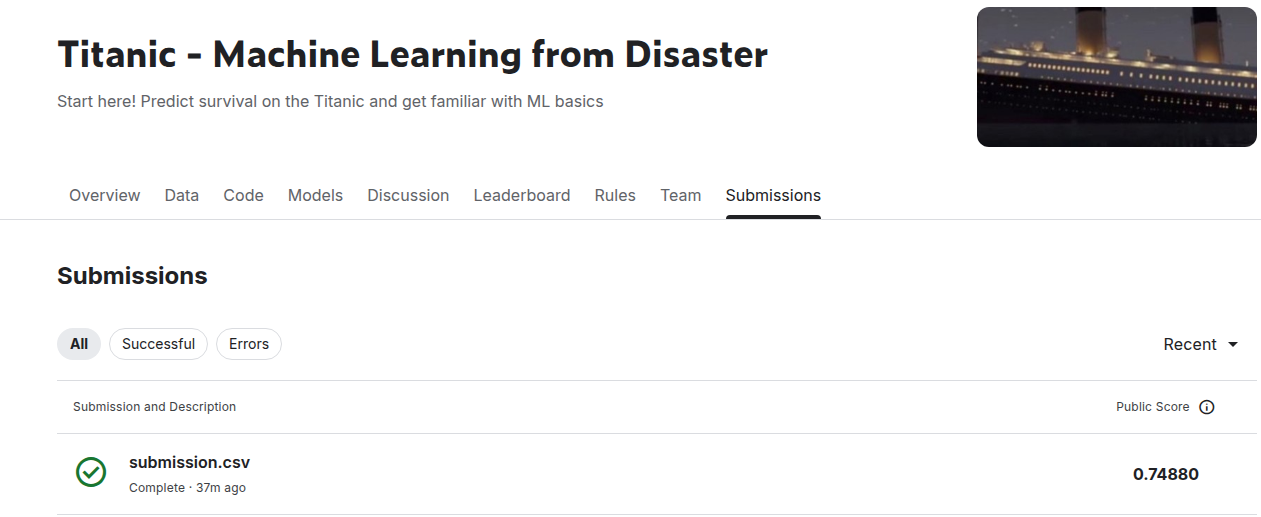
* Tạo và lưu vào file submission

| *# Orijinal test csv'sini tekrar yükle (PassengerId için)* test\_data\_orig = pd.read\_csv("/kaggle/input/titanic/test.csv")  *# best\_model ile test verisi üzerinde tahmin yap* test\_preds = best\_model.predict(test\_data)  *# Submission dataframe'i oluştur* submission = pd.DataFrame({  "PassengerId": test\_data\_orig["PassengerId"],  "Survived": test\_preds })  *# CSV olarak kaydet (kaggle ortamında)* submission.to\_csv("submission.csv", index=False)  print("Submission file 'submission.csv' created.") |
| --- |

# 5.Kết luận

**Kết quả đạt được:**

* Mô hình Random Forest đạt accuracy 82.12%, F1 score 75.38% trên validation set.
* File submission.csv được tạo để nộp lên Kaggle, đạt accuracy ~0.748 trên leaderboard.



**Bài học rút ra:**

* Làm sạch dữ liệu (xử lý missing values, mã hóa categorical, chuẩn hóa) chiếm phần lớn thời gian và quan trọng để mô hình hoạt động tốt.
* Feature engineering (log1p cho Fare) giúp cải thiện hiệu suất.
* Random Forest vượt trội hơn Logistic Regression và XGBoost trong trường hợp này.

**Cải thiện trong tương lai:**

* Thêm feature engineering: Tạo Title từ Name (Mr, Mrs, Miss), chia Age thành nhóm (trẻ em, người lớn).
* Sử dụng cross-validation để đánh giá mô hình ổn định hơn.
* Thử các mô hình khác như Neural Networks hoặc điều chỉnh hyperparameter cho XGBoost.