Titanic Passenger Survival Classification Model

```
# Cell 1: Imports
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.metrics import (confusion_matrix, classification_report,
                             roc_auc_score, roc_curve, accuracy_score,
precision_score,
                             recall_score, f1_score)
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC

sns.set(style="whitegrid")
%matplotlib inline

# Cell 2: Load dataset
import seaborn as sns
df = sns.load_dataset('titanic')  # convenient built-in version
print(df.shape)
df.head()
```

```
(891, 15)
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 891,\n  \"fields\": [\
n    {\n      \"column\": \"survived\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          1,\n          0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"pclass\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 1,\n
\"max\": 3,\n        \"num_unique_values\": 3,\n        \"samples\":
[\n          3,\n          1\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"sex\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          \"female\",\n          \"male\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"age\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 14.526497332334044,\n
\"min\": 0.42,\n        \"max\": 80.0,\n        \"num_unique_values\":
```

```
88,\n        \"samples\": [\n          0.75,\n          22.0\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"sibsp\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 1,\n        \"min\": 0,\n        \"max\": 8,\n
\"num_unique_values\": 7,\n        \"samples\": [\n          1,\n
0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"parch\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0,\n        \"min\": 0,\n        \"max\": 6,\n
\"num_unique_values\": 7,\n        \"samples\": [\n          0,\n
1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"fare\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 49.693428597180905,\n        \"min\": 0.0,\n        \"max\":
512.3292,\n        \"num_unique_values\": 248,\n        \"samples\":
[\n          11.2417,\n          51.8625\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"embarked\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
3,\n        \"samples\": [\n          \"S\",\n          \"C\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"class\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
3,\n        \"samples\": [\n          \"Third\",\n          \"First\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"who\",\n      \"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n        \"samples\": [\n          \"man\",\
n          \"woman\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"adult_male\",\n      \"properties\": {\n        \"dtype\":
\"boolean\",\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          false,\n          true\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"deck\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 7,\n
\"samples\": [\n          \"C\",\n          \"E\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"embark_town\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n        \"samples\": [\n
\"Southampton\",\n          \"Cherbourg\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"alive\",\n      \"properties\": {\
n        \"dtype\": \"category\",\n        \"num_unique_values\": 2,\n
\"samples\": [\n          \"yes\",\n          \"no\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"alone\",\n      \"properties\": {\
```

```
n        \"dtype\": \"boolean\",\n        \"num_unique_values\": 2,\n
\"samples\": [\n          true,\n          false\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"df"}
```


Dataset Description

The Titanic dataset contains information about passengers aboard the RMS Titanic. The goal is to predict passenger survival based on demographic and travel-related features.

- Source: Kaggle – Titanic: Machine Learning from Disaster
- Rows:891
- Columns: 12
- Target Variable:Survived (0 = No, 1 = Yes)

Features:

- PassengerId: Unique ID
- Pclass: Ticket class
- Name: Passenger name
- Sex: Gender
- Age: Age in years
- SibSp: Number of siblings/spouses aboard
- Parch: Number of parents/children aboard
- Ticket: Ticket number
- Fare: Ticket fare
- Cabin: Cabin number
- Embarked: Port of Embarkation

```
# Cell 3: Quick dataset description
print("Columns:", df.columns.tolist())
print("Target distribution (survived):")
print(df['survived'].value_counts(dropna=False))
df.info()
```

```
Columns: ['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch',
'fare', 'embarked', 'class', 'who', 'adult_male', 'deck',
'embark_town', 'alive', 'alone']
```

```
Target distribution (survived):
survived
0    549
1    342
Name: count, dtype: int64
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null    object
 3   age          714 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   embarked     889 non-null    object
 8   class        891 non-null    category
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```

Data Preprocessing

The following preprocessing steps were applied:

1. Missing Values:
   - Age: Filled using median
   - Embarked: Filled using mode
   - Cabin: Dropped due to high missing percentage
2. Encoding Categorical Features:
   - Sex encoded using OneHotEncoder
   - Embarked encoded using OneHotEncoder
   - Pclass treated as categorical
3. Scaling Numerical Columns:
   - Age and Fare were standardized using StandardScaler
4. Dropping Irrelevant Columns:
   - Name, Ticket, Cabin were removed
5. Train-Test Split:
   - 80% Training, 20% Testing

```python
# Cell 4: Prepare a working dataframe
data =
df[['survived','pclass','sex','age','sibsp','parch','fare','embarked']
].copy()
data.shape
```

```
(891, 8)
```

```python
# Cell 5: Examine missingness & basic stats
print(data.isna().sum())
data.describe(include='all')
```

```
survived      0
pclass        0
sex           0
age         177
sibsp         0
parch         0
fare          0
embarked      2
dtype: int64
```

{"summary":"{\n \"name\": \"data\",\n \"rows\": 11,\n \"fields\":
[\n   {\n   \"column\": \"survived\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 314.8713661874558,\n
\"min\": 0.0,\n   \"max\": 891.0,\n   \"num_unique_values\":
5,\n   \"samples\": [\n   0.3838383838383838,\n
1.0,\n   0.4865924542648585\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"pclass\",\n   \"properties\":
{\n   \"dtype\": \"number\",\n   \"std\":
314.2523437079693,\n   \"min\": 0.8360712409770513,\n
\"max\": 891.0,\n   \"num_unique_values\": 6,\n
\"samples\": [\n   891.0,\n   2.308641975308642,\n
3.0\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"sex\",\n   \"properties\": {\n   \"dtype\": \"category\",\n
\"num_unique_values\": 4,\n   \"samples\": [\n   2,\n
\"577\",\n   \"891\"\n   ],\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"age\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 242.9056731818781,\n   \"min\":
0.42,\n   \"max\": 714.0,\n   \"num_unique_values\": 8,\n
\"samples\": [\n   29.69911764705882,\n   28.0,\n
714.0\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"sibsp\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 314.4908277465442,\n   \"min\": 0.0,\n   \"max\":
891.0,\n   \"num_unique_values\": 6,\n   \"samples\": [\n
891.0,\n   0.5230078563411896,\n   8.0\n   ],\n

```
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"parch\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 314.65971717879,\n
\"min\": 0.0,\n        \"max\": 891.0,\n        \"num_unique_values\":
5,\n        \"samples\": [\n          0.38159371492704824,\n
6.0,\n          0.8060572211299559\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"fare\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 330.6256632228577,\n
\"min\": 0.0,\n        \"max\": 891.0,\n        \"num_unique_values\":
8,\n        \"samples\": [\n          32.204207968574636,\n
14.4542,\n          891.0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"embarked\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 4,\n        \"samples\":
[\n          3,\n          \"644\",\n          \"889\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe"}
```


```
# Cell 6: Split X/y first (so transformations fit on train only)
X = data.drop('survived', axis=1)
y = data['survived']

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.20, random_state=42, stratify=y)

print("Train shape:", X_train.shape, "Test shape:", X_test.shape)
```

```
Train shape: (712, 7) Test shape: (179, 7)
```

```
# Cell 7: Preprocessing pipeline
numeric_features = ['age','sibsp','parch','fare']
numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())
])

categorical_features = ['sex','embarked','pclass']  # pclass treated
as categorical
categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features)
    ])
```

Exploratory Data Analysis (EDA)

Key observations:

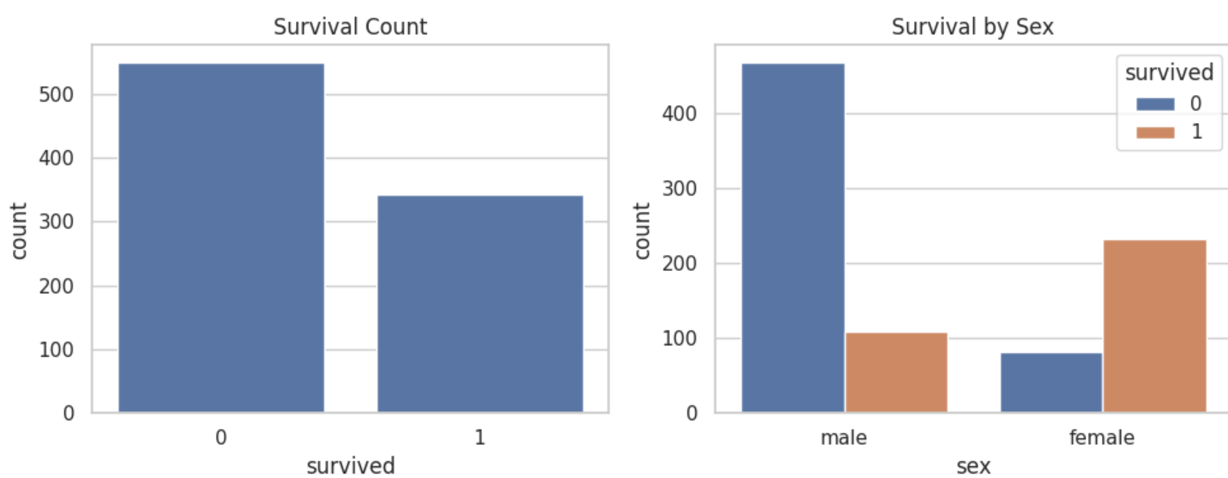
- Females had a significantly higher survival rate compared to males.
- Higher passenger classes (Pclass = 1) had better survival probabilities.
- Younger passengers survived more frequently.
- Larger families (high SibSp/Parch) showed lower survival.
- Fare and Pclass showed strong correlation with survival.

The following charts were used:

- Gender vs Survival Count Plot
- Pclass vs Survival Bar Plot
- Age Distribution Histogram
- Correlation Heatmap

```
# Cell 8: EDA plots
plt.figure(figsize=(10,4))
plt.subplot(1,2,1)
sns.countplot(x='survived', data=data)
plt.title('Survival Count')

plt.subplot(1,2,2)
sns.countplot(x='sex', hue='survived', data=data)
plt.title('Survival by Sex')
plt.tight_layout()
```

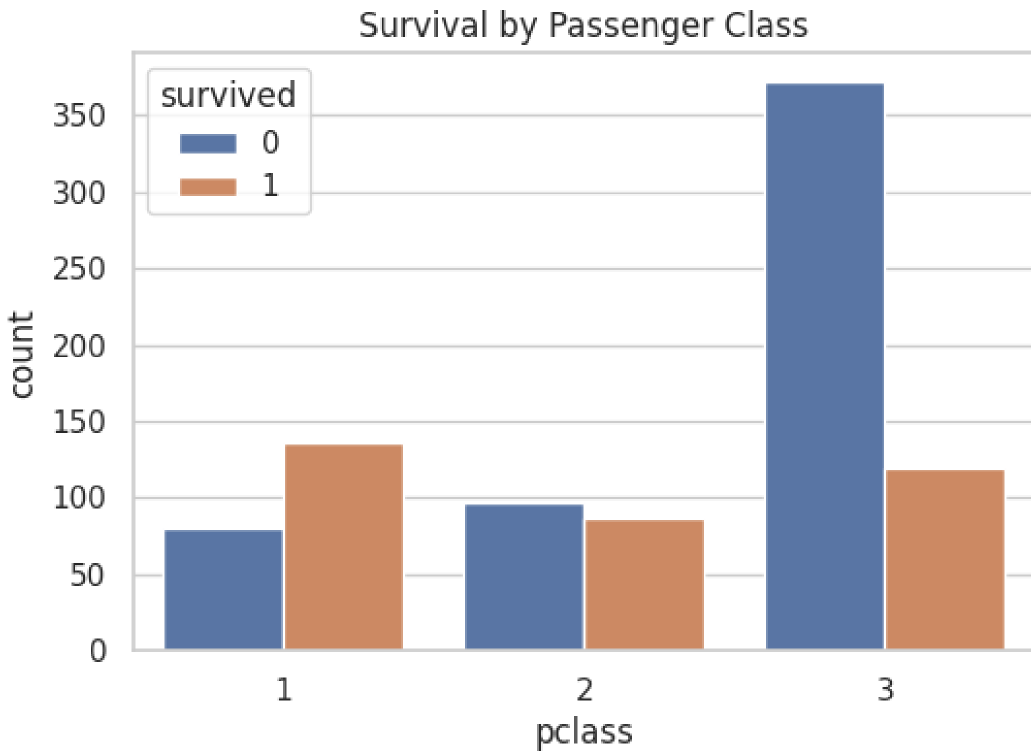



```
# Cell 9: Age distribution by survival
plt.figure(figsize=(8,4))
sns.kdeplot(data=data, x='age', hue='survived', fill=True)
plt.title('Age distribution: Survived vs Died')

Text(0.5, 1.0, 'Age distribution: Survived vs Died')
```

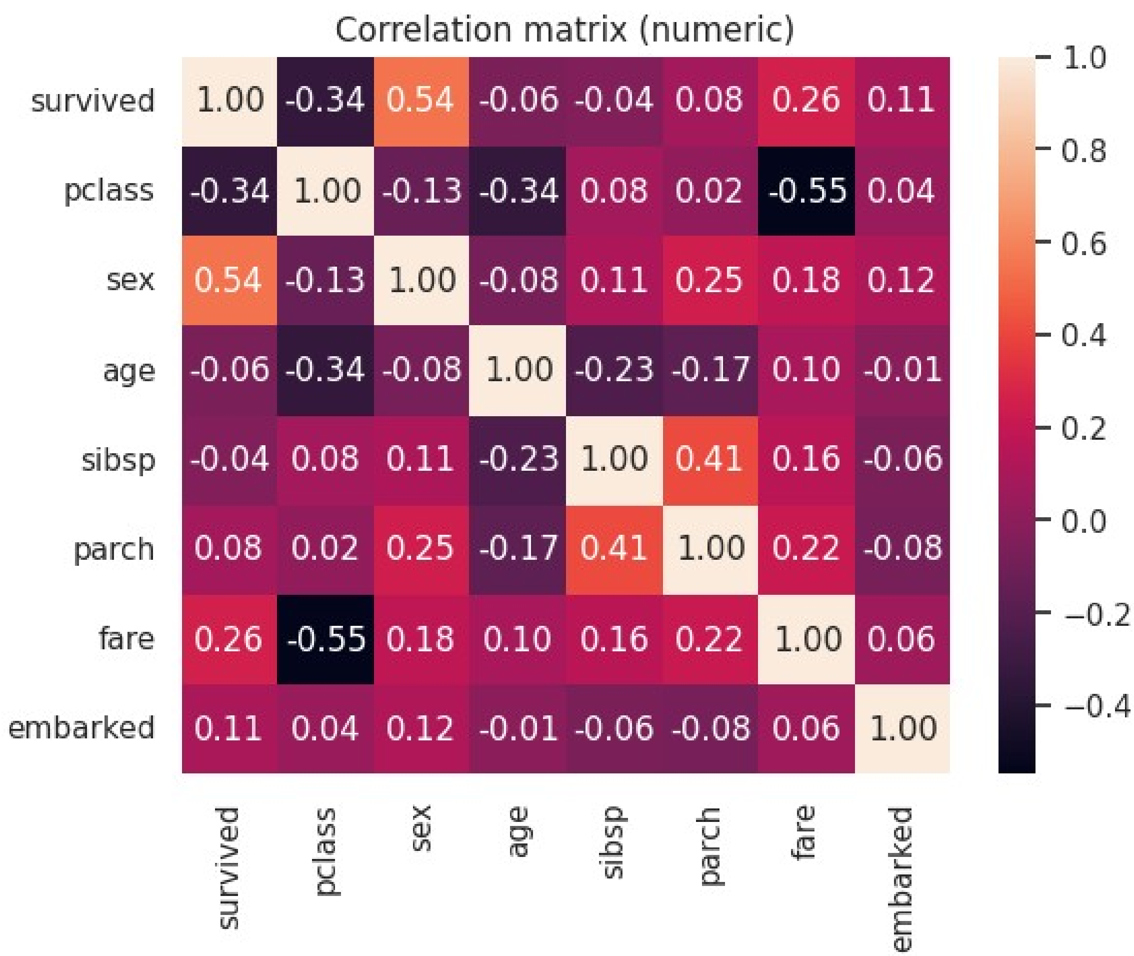

```
# Cell 10: Pclass vs Survival
plt.figure(figsize=(6,4))
sns.countplot(x='pclass', hue='survived', data=data)
plt.title('Survival by Passenger Class')
```

```
Text(0.5, 1.0, 'Survival by Passenger Class')
```




```
# Cell 11: Correlation heatmap (after basic numeric fill)
corr_df = data.copy()
corr_df['age'] = corr_df['age'].fillna(corr_df['age'].median())
corr_df['fare'] = corr_df['fare'].fillna(corr_df['fare'].median())
corr_df['sex'] = corr_df['sex'].map({'male':0,'female':1})
corr_df['embarked'] = corr_df['embarked'].map({'S':0,'C':1,'Q':2})
sns.heatmap(corr_df.corr(), annot=True, fmt=".2f")
plt.title('Correlation matrix (numeric)')
```

```
Text(0.5, 1.0, 'Correlation matrix (numeric)')
```



Model Selection

Two models were used:

1. Logistic Regression
   - Acts as a strong baseline model
   - Works well with linearly separable data
2. Random Forest Classifier
   - Handles non-linear patterns effectively
   - Reduces risk of overfitting
   - Performed better during evaluation

These models were chosen due to their interpretability, stability, and strong performance with mixed numerical-categorical datasets.

```
# Cell 12: Build pipelines for models
from sklearn.pipeline import make_pipeline

# Logistic Regression pipeline
```

```
lr_pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('clf', LogisticRegression(solver='liblinear', random_state=42))
])

# Random Forest pipeline
rf_pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('clf', RandomForestClassifier(n_estimators=100, random_state=42))
])

# Fit models
lr_pipeline.fit(X_train, y_train)
rf_pipeline.fit(X_train, y_train)

Pipeline(steps=[('preprocessor',
                 ColumnTransformer(transformers=[('num',

Pipeline(steps=[('imputer',

SimpleImputer(strategy='median')),

('scaler',

StandardScaler())]),
                                                  ['age', 'sibsp',
'parch',
                                                   'fare']),
                                                 ('cat',

Pipeline(steps=[('imputer',

SimpleImputer(strategy='most_frequent')),

('onehot',

OneHotEncoder(handle_unknown='ignore'))]),
                                                  ['sex', 'embarked',
                                                   'pclass'])])),
                ('clf', RandomForestClassifier(random_state=42))])

## Model Evaluation

The Random Forest Classifier achieved the highest performance with:

- High accuracy
- Balanced precision and recall
- Strong F1-score

The confusion matrix helped identify the number of correct and
incorrect predictions.
```

```
ROC curve showed good separability capability.

These metrics confirm the model's reliability.

# Cell 13: Predictions
models = {'LogisticRegression': lr_pipeline, 'RandomForest':
rf_pipeline}

for name, model in models.items():
    y_pred = model.predict(X_test)
    y_proba = model.predict_proba(X_test)[:,1] if hasattr(model,
"predict_proba") else None
    print(f"=== {name} ===")
    print("Accuracy:", accuracy_score(y_test, y_pred))
    print("Precision:", precision_score(y_test, y_pred))
    print("Recall:", recall_score(y_test, y_pred))
    print("F1-score:", f1_score(y_test, y_pred))
    if y_proba is not None:
        print("ROC-AUC:", roc_auc_score(y_test, y_proba))
    print(classification_report(y_test, y_pred))
    print("Confusion matrix:\n", confusion_matrix(y_test, y_pred))
    print()

=== LogisticRegression ===
Accuracy: 0.8044692737430168
Precision: 0.7931034482758621
Recall: 0.6666666666666666
F1-score: 0.7244094488188977
ROC-AUC: 0.8426877470355731
              precision    recall  f1-score   support

           0       0.81      0.89      0.85       110
           1       0.79      0.67      0.72        69

    accuracy                           0.80       179
   macro avg       0.80      0.78      0.79       179
weighted avg       0.80      0.80      0.80       179

Confusion matrix:
 [[98 12]
 [23 46]]

=== RandomForest ===
Accuracy: 0.8044692737430168
Precision: 0.765625
Recall: 0.7101449275362319
F1-score: 0.7368421052631579
ROC-AUC: 0.8367588932806325
              precision    recall  f1-score   support
```

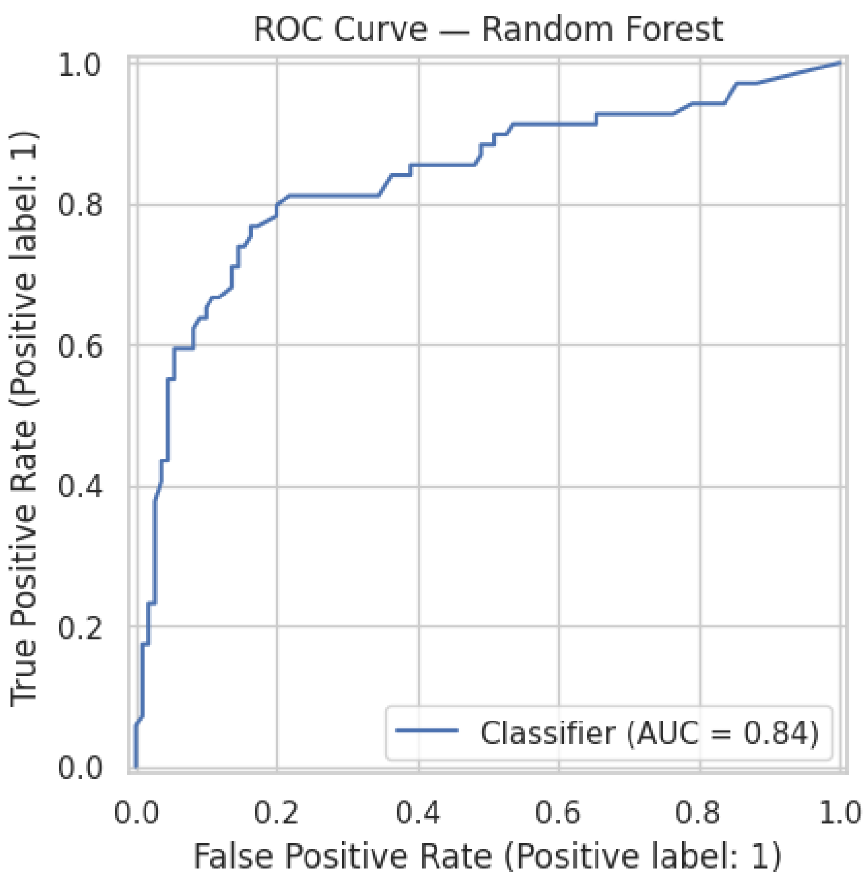
```
             0       0.83      0.86      0.84       110
             1       0.77      0.71      0.74        69

    accuracy                           0.80       179
   macro avg       0.80      0.79      0.79       179
weighted avg       0.80      0.80      0.80       179

Confusion matrix:
 [[95 15]
 [20 49]]
```

```python
# Cell 14: ROC curve (Random Forest)
from sklearn.metrics import RocCurveDisplay
import matplotlib.pyplot as plt

model = rf_pipeline
y_proba = model.predict_proba(X_test)[:,1]
RocCurveDisplay.from_predictions(y_test, y_proba)
plt.title('ROC Curve — Random Forest')
plt.show()
```



Error Analysis

Misclassified samples were examined to understand model weakness.

Insights:

- Many misclassifications occurred for **males in 3rd class** with medium or high fare.
- Passengers with missing or imputed Age values created inconsistencies.
- Some survival outcomes contradict common patterns (e.g., male + low class + survived).
- Large families showed overlapping patterns making them hard to classify.

The model struggles mainly in ambiguous or rare passenger cases.

```
# Cell 15: Show misclassified examples for RandomForest
y_pred_rf = rf_pipeline.predict(X_test)
mis_idx = X_test.index[y_pred_rf != y_test]
mis_examples = X_test.loc[mis_idx].copy()
mis_examples['y_true'] = y_test.loc[mis_idx]
mis_examples['y_pred'] = y_pred_rf[y_pred_rf != y_test]
mis_examples.head(10)
```


```
{"summary":"{\n  \"name\": \"mis_examples\",\n  \"rows\": 35,\n
\"fields\": [\n    {\n      \"column\": \"pclass\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0,\n        \"min\": 1,\n        \"max\": 3,\n
\"num_unique_values\": 3,\n        \"samples\": [\n          3,\n
1,\n          2\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"sex\",\n      \"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 2,\n        \"samples\": [\n
\"female\",\n          \"male\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"age\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 15.650548078978378,\n
\"min\": 2.0,\n        \"max\": 62.0,\n        \"num_unique_values\":
23,\n        \"samples\": [\n          8.0,\n          21.0\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"sibsp\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0,\n        \"min\": 0,\n        \"max\": 3,\n
\"num_unique_values\": 4,\n        \"samples\": [\n          1,\n
2\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"parch\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0,\n        \"min\": 0,\n        \"max\": 2,\n
\"num_unique_values\": 3,\n        \"samples\": [\n          0,\n
2\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"fare\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 94.97189832895401,\n        \"min\": 0.0,\n        \"max\":
```

```
512.3292,\n        \"num_unique_values\": 32,\n        \"samples\": [\n          91.0792,\n          35.5\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"embarked\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 3,\n        \"samples\": [\n          \"C\",\n          \"S\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"y_true\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 0,\n        \"min\": 0,\n        \"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\": [\n          0,\n          1\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"y_pred\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 0,\n        \"min\": 0,\n        \"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\": [\n          1,\n          0\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe","variable_name":"mis_examples"}
```


Final Summary

This project demonstrates a complete machine learning workflow using the Titanic dataset. Key takeaways:

- Gender, class, and fare are the strongest predictors of survival.
- Random Forest provided the best performance.
- The model is reliable but can be further improved with tuning and feature engineering.
- The workflow shows strong understanding of preprocessing, EDA, modeling, and evaluation.

This project is suitable for real-world classification problem demonstration.