Titanic Survival Prediction Project

Objective

The goal of this project is to build a basic machine learning model that can predict whether a passenger survived the Titanic disaster. This involves understanding the dataset, preprocessing the data, training a model, and evaluating its performance. The project is structured to be understandable for both technical and non-technical audiences.

Project Steps and Rationale

Step 1: Import Libraries

Action: Imported Python libraries such as pandas, numpy, matplotlib, seaborn, and scikit-learn.

Reason: These libraries provide essential functions to handle data, create machine learning models, and visualize results efficiently.

Step 2: Load the Dataset

Action: Loaded the Titanic dataset (CSV file) into a pandas DataFrame.

Reason: Data must be brought into the working environment (Python) before we can explore, clean, and model it.

Step 3: Exploratory Data Analysis (EDA)

Action: Examined the dataset's structure, missing values, and overall patterns using summary statistics and graphs.

Reason: EDA helps us understand the dataset, identify potential issues like missing values, and observe important trends (e.g., survival rates by gender).

Step 4: Data Preprocessing

Action:

- Filled missing 'Age' values with the median age.
- Filled missing 'Embarked' values with the most frequent embarkation port.
- Dropped non-essential columns: 'Cabin', 'Ticket', 'Name', and 'PassengerId'.
- Converted categorical variables ('Sex', 'Embarked') into numeric form using one-hot encoding.

Reason:

- Machine learning models require complete and numerical data.
- Removing irrelevant columns reduces noise.
- Encoding allows models to interpret categorical information correctly.

Step 5: Define Features and Target

Action:

- Defined features (X) as all columns except 'Survived'.
- Defined target (y) as the 'Survived' column.

Reason: Clearly separating input variables (features) and the output variable (target) is critical for supervised machine learning.

Step 6: Split the Data

Action: Split the dataset into a training set (80%) and a testing set (20%).

Reason: Training data is used to build the model. Testing data is used to evaluate how well the model performs on unseen data.

Step 7: Train the Machine Learning Model

Action: Used Logistic Regression to train the model on the training data.

Reason: Logistic Regression is simple, effective, and well-suited for binary classification problems like survival prediction (yes/no outcome).

Step 8: Make Predictions

Action: Used the trained model to predict survival on the test data.

Reason: Predictions allow us to assess how well the model has learned from the training data.

Step 9: Evaluate the Model

Action: Calculated performance metrics:

- Accuracy
- Precision
- Recall
- F1-Score

Reason: Evaluation metrics help measure the quality of predictions. They give insight into different aspects of the model's performance (overall correctness, handling of false positives/negatives, etc.).

Step 10: Confusion Matrix

Action: Created and visualized a confusion matrix.

Reason: The confusion matrix provides a detailed breakdown of correct and incorrect predictions, offering a clear understanding of the model's strengths and weaknesses.

Step 11: ROC Curve and AUC Score

Action: Plotted the ROC curve and calculated the ROC AUC score.

Reason: The ROC curve helps visualize the model's ability to distinguish between the two classes (survived vs not survived). A higher AUC score indicates better model performance.

Final Summary

This project involved:

- Understanding and preparing the Titanic dataset.
- Building and evaluating a simple but powerful Logistic Regression model.
- Interpreting model performance with professional metrics and visualizations.

Additional Important Point:

Why was normalization/standardization not done?

- For Logistic Regression, if your features are already fairly clean (like Age, Fare), normalization is not strictly necessary.

- Since the Titanic data doesn't have extreme feature differences, we safely skipped scaling.
  If needed in future (e.g., for Random Forest or SVM models), you can easily add it.