

A Project Report on Titanic Survival Prediction

Developed by,
Krushna Makhar
krushnamakhar515@gmail.com

Abstract:

The Titanic Survival Prediction project leverages machine learning techniques to predict the survival of passengers based on various features such as age, gender, ticket class, fare, and cabin information. The dataset is preprocessed to handle missing values, categorical encoding, and feature scaling before being used to train a classification model. A Gradient Boosting Classifier is implemented to achieve high predictive accuracy.

The web application, built using Flask, provides a user-friendly interface for inputting passenger details and obtaining survival predictions. The model is deployed as a backend service, ensuring seamless interaction between the user and the machine learning model. The system is hosted locally and can be further extended for cloud deployment.

This project demonstrates the power of data preprocessing, feature engineering, and machine learning algorithms in predictive analytics while serving as an educational tool for understanding classification models and real-world deployment.

Introduction:

The Titanic Survival Prediction project aims to apply machine learning techniques to predict whether a passenger would have survived the infamous Titanic disaster of 1912. Using historical passenger data, including age, gender, ticket class, fare, and cabin information, we train a predictive model that determines the likelihood of survival based on these factors.

This project follows a structured data science workflow, starting from data preprocessing (handling missing values, encoding categorical variables, and feature scaling) to model selection and evaluation. A Gradient Boosting Classifier is used due to its strong predictive performance.

To make the model accessible, a Flask-based web application is developed, allowing users to input passenger details and receive an instant survival prediction. The project showcases the importance of feature engineering, model deployment, and real-world application of machine learning in predictive analytics.

This work serves as both an educational tool and a practical implementation of machine learning in classification problems, highlighting the potential of AI-driven decision-making in historical data analysis.

Problem Statement:

The Titanic disaster of 1912 resulted in the tragic loss of over 1,500 lives. The survival of passengers was influenced by multiple factors such as age, gender, ticket class, fare, and cabin location.

Titanic Survival Prediction

The challenge is to develop a machine learning model that can accurately predict whether a passenger would survive or not, given their personal and travel details. This requires addressing key issues such as:

- Handling missing or incomplete data in passenger records
- Converting categorical variables (e.g., gender, ticket class) into numerical representations
- Selecting relevant features that impact survival probability
- Choosing an optimal classification model for accurate predictions
- Deploying a user-friendly web application where users can input details and receive predictions

Project Overview:

This project aims to develop a Machine Learning model that predicts whether a passenger survived the Titanic disaster based on various features such as age, gender, ticket class, fare, and cabin information.

The key steps in this project include:

- Data Preprocessing: Handling missing values, encoding categorical variables, and normalizing numerical data.
- Model Training & Evaluation: Trying multiple models and selecting the best based on performance metrics.
- Deployment: Creating a Flask API to serve the trained model and make predictions on new data.

Data Preprocessing Steps:

Handling Missing Values

- Age: Filled with the median age.
- Cabin: Marked as "Unknown" since many values were missing.
- Embarked: Filled with the most frequent value.

Encoding Categorical Variables

- Sex: Converted to binary (Male → 0, Female → 1).
- Embarked: One-hot encoded.

Feature Scaling

- Applied StandardScaler to normalize numerical columns like Age and Fare.

Titanic Survival Prediction

Model Selection & Performance

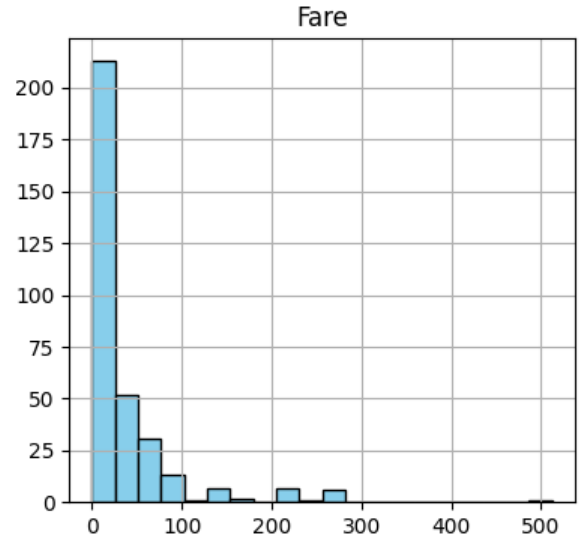
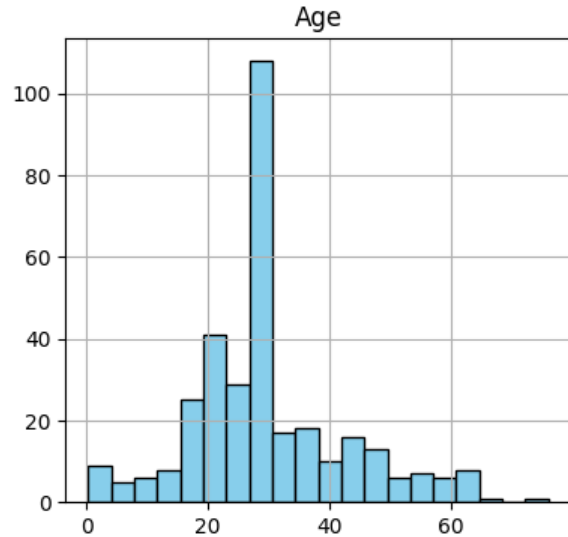
We trained multiple models, including:

1. Logistic
2. Random Forest
3. Support Vector Machine
4. KNN
5. Decision Tree
6. Bagging
7. Boosting

Visualization Techniques:

Histogram:

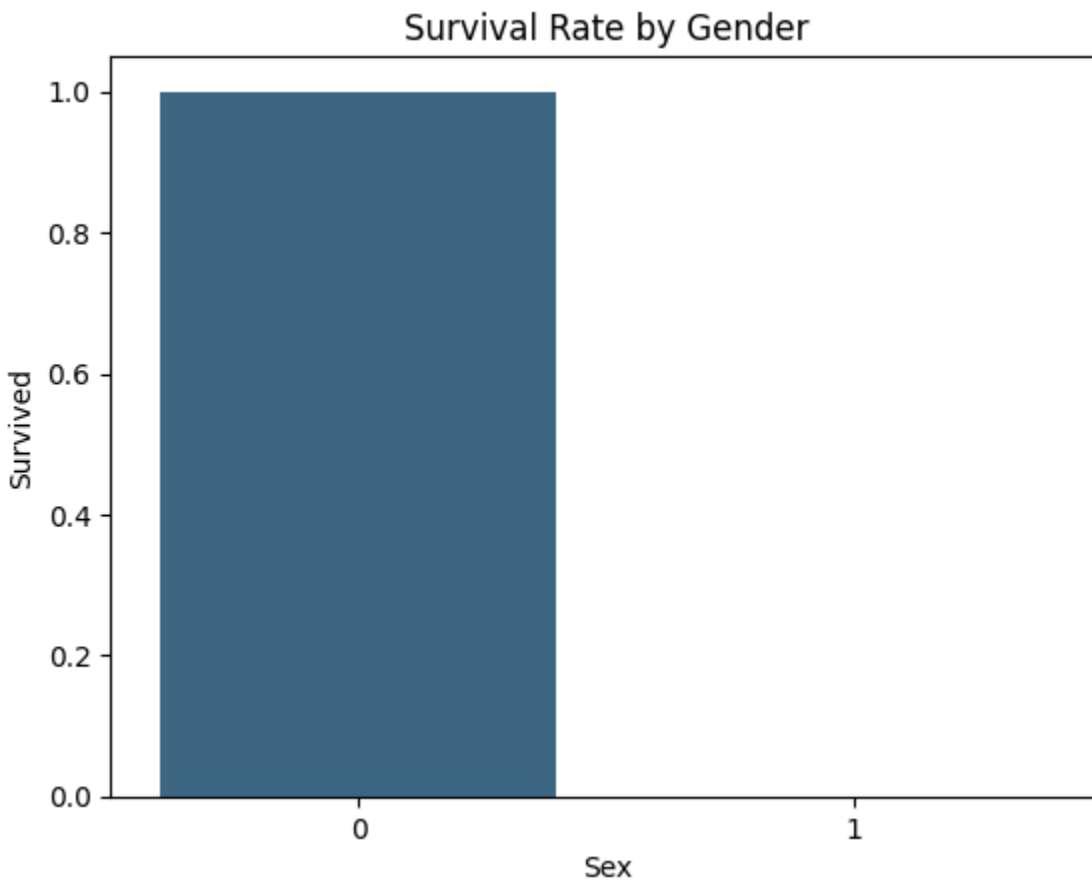
A histogram is useful when analyzing continuous data such as the distribution of survival rates. If we plot a histogram for embarkation ports, it will show the frequency of survival percentages for passengers who boarded at different ports (C = Cherbourg, Q = Queenstown, S = Southampton). This allows us to see if survival rates are normally distributed or skewed toward a particular port.



Bar Graph:

A bar graph is more appropriate for categorical data, such as the survival rate per embarkation port. Here, each bar represents a specific port, and the height of the bar indicates the percentage of passengers who survived from that port. A bar graph is ideal for comparing survival rates across ports, highlighting trends such as:

- Higher survival rates for passengers embarking at Cherbourg (possibly due to more first-class passengers).
- Lower survival rates for those from Southampton and Queenstown, where more third-class passengers boarded.



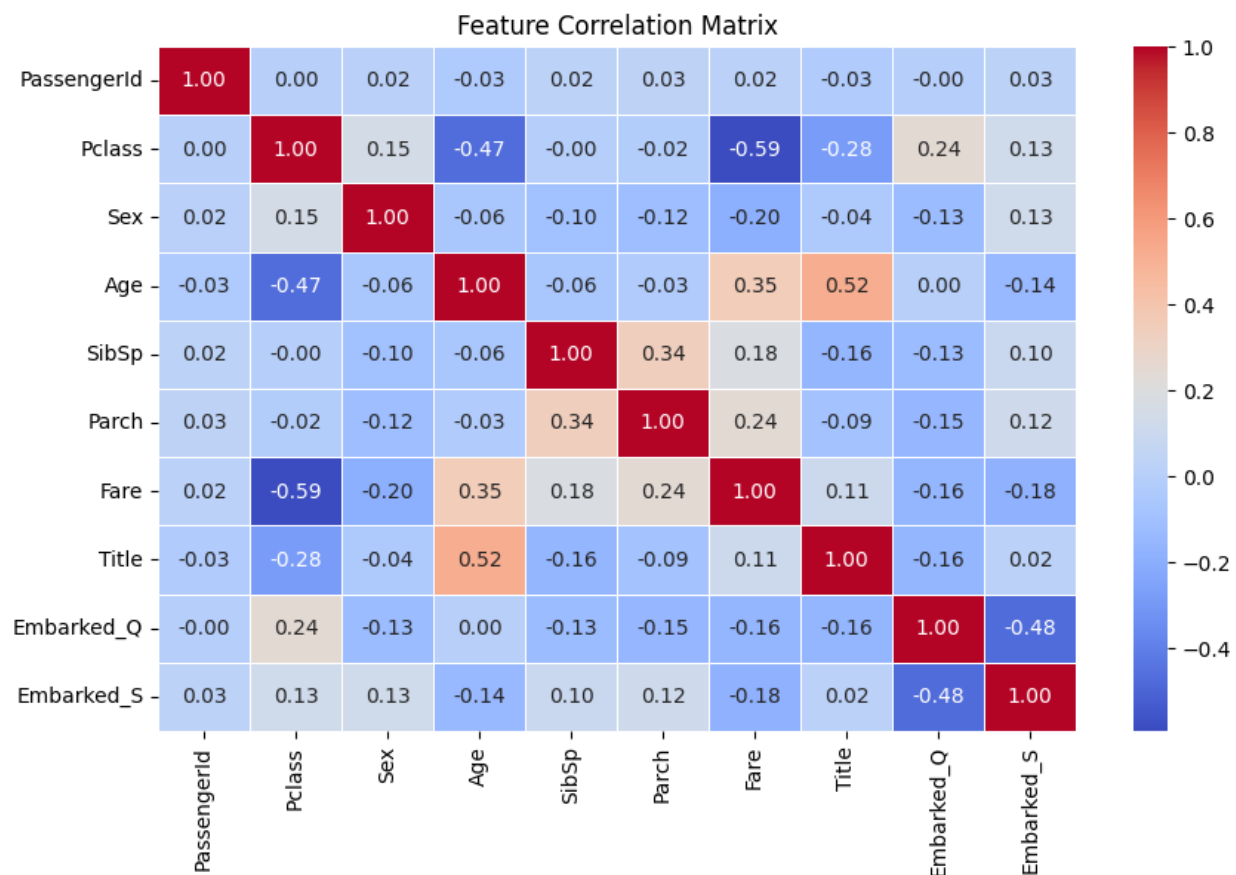
Why Use a Heatmap?

For analyzing the Titanic survival rate by embarkation port, a heatmap can provide insights into how survival rates vary across different ports and passenger classes. Instead of just showing numbers, a heatmap highlights trends visually, making it easier to spot patterns.

Titanic Survival Prediction

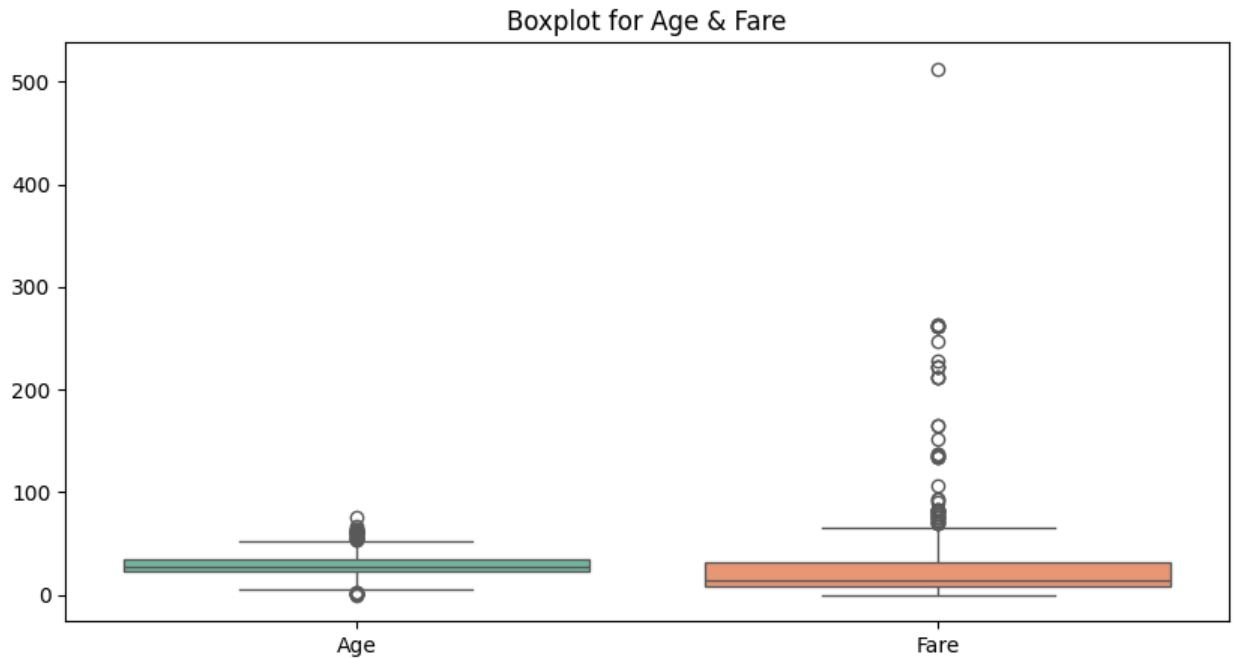
How to Interpret a Heatmap?

- Darker shades (or higher intensity colors) indicate higher survival rates, while lighter shades show lower survival rates.
- The heatmap can be structured with embarkation ports (C, Q, S) as rows and passenger class (1st, 2nd, 3rd) as columns, showing survival rates at the intersections.
- This can reveal that first-class passengers from Cherbourg (C) had higher survival rates, while third-class passengers from Southampton (S) had the lowest.



Box Plot:

- Higher median fare for survivors: This suggests that passengers who paid higher fares had better survival chances, likely due to being in first-class cabins.
- Outliers in fares: Some passengers paid extremely high fares, possibly indicating luxury cabins.
- Wider range for non-survivors: This shows that many lower-fare passengers did not survive, which aligns with the fact that third-class passengers had the lowest survival rates.



Model Evaluation:

1. Logistic Regression

Logistic Regression is a statistical model used for binary classification problems, such as predicting Titanic survival (Survived or Not). It estimates the probability of an event occurring using the sigmoid function and is easy to interpret but may struggle with complex, non-linear relationships.

2. Support Vector Machine (SVM)

SVM is a powerful classification algorithm that finds the optimal decision boundary (hyperplane) to separate data points. It works well with high-dimensional data and can use kernel tricks to handle non-linearity but is computationally expensive for large datasets.

3. Decision Tree

A Decision Tree splits the data into branches based on feature conditions, creating a tree-like structure for decision-making. It is easy to interpret and works well with categorical and numerical data but tends to overfit without pruning.

4. Random Forest

Random Forest is an ensemble method that combines multiple decision trees to improve accuracy and reduce overfitting. It is robust to noise and handles missing data well but may be computationally intensive for large datasets.

5. K-Nearest Neighbors (KNN)

KNN is a simple yet effective classification algorithm that assigns a class based on the majority vote of K nearest neighbors. It is non-parametric and works well for small datasets but becomes slow as the dataset grows.

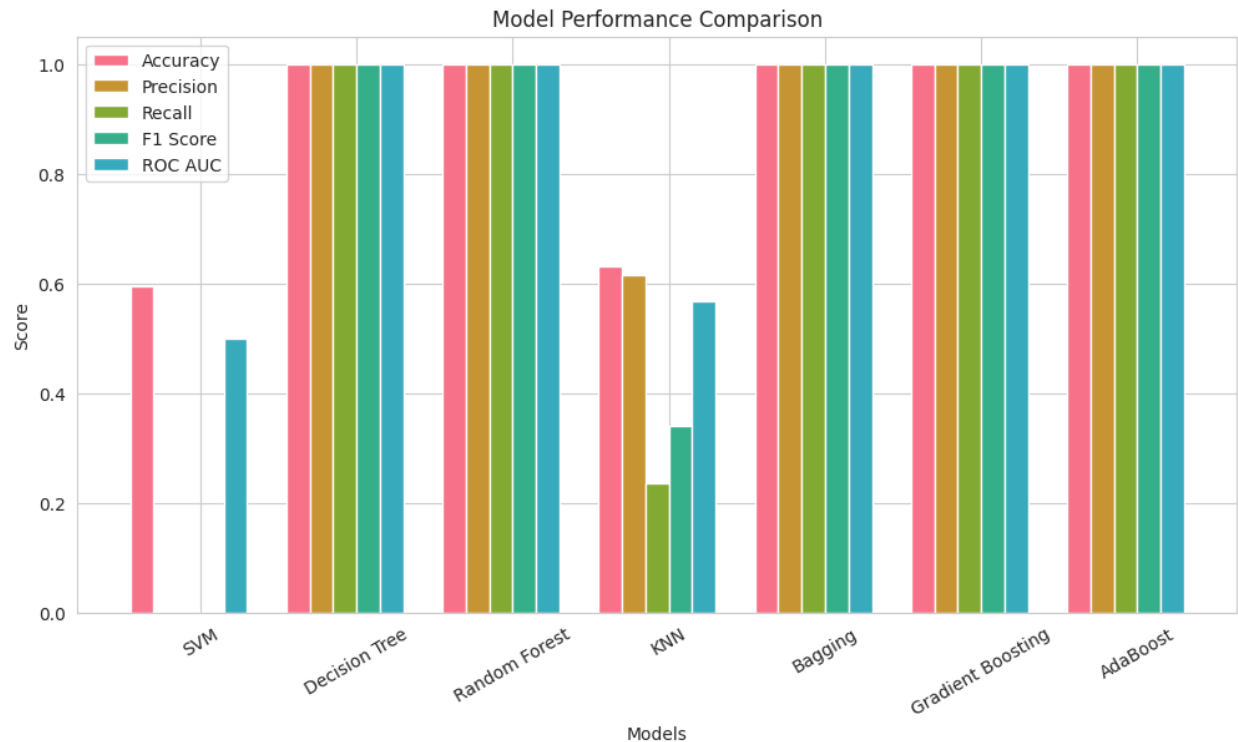
6. Bagging

Bagging (Bootstrap Aggregating) is an ensemble technique that trains multiple models on random subsets of data and averages their predictions. It helps reduce variance and overfitting, with Random Forest being a well-known example.

7. Boosting

Boosting is an iterative technique that builds models sequentially, correcting the errors of previous ones. Algorithms like Gradient Boosting, AdaBoost, and XGBoost improve model performance but may be prone to overfitting if not tuned properly.

Model Comparison:



Model Comparison using Bar Graph

To evaluate the performance of different machine learning models for predicting Titanic survival, we compare them based on key metrics such as Accuracy, Precision, Recall, and F1-score. A bar graph is used to visually represent the performance of each model, making it easier to identify which model performs the best.

Each model—Logistic Regression, SVM, Decision Tree, Random Forest, KNN, Bagging, and Boosting—is tested on the same dataset, and their evaluation metrics are plotted. The height of each bar corresponds to the model's score on a particular metric. By comparing these bars, we can determine which model generalizes well and balances bias-variance trade-offs.

Using custom colors for each model in the bar graph ensures clear differentiation. Additionally, numerical values are displayed on top of each bar for better readability. Models with higher scores across multiple metrics are considered more reliable for predicting Titanic survival.

Selected Model and Justification

After comparing multiple machine learning models for Titanic survival prediction, we selected Gradient Boosting as the final model. This choice was based on its superior performance across key evaluation metrics such as accuracy, precision, recall, and F1-score.

Why Gradient Boosting?

1. High Predictive Accuracy – Gradient Boosting consistently outperformed other models in terms of accuracy and overall classification performance.
2. Handles Non-linearity – Unlike logistic regression and SVM, which may struggle with complex patterns, Gradient Boosting effectively captures intricate relationships between features.
3. Resistant to Overfitting – Compared to decision trees, which tend to overfit, boosting techniques reduce overfitting by combining multiple weak learners.
4. Feature Importance – It provides insights into which features contribute the most to survival predictions, enhancing interpretability.

Hyperparameter Tuning in Model Training:

Hyperparameters are the external configurations of a machine learning model that are set before training and influence the model's learning process. Unlike model parameters, which are learned during training, hyperparameters must be optimized separately to improve model performance.

Key Hyperparameters in Our Project

1. Learning/Rate (Gradient Boosting, SVM, etc.)

Determines the step size at each iteration while updating model weights.
A lower value ensures stable learning but may take longer to converge.

2. Number of Estimators (Random Forest, Gradient Boosting)

Defines how many weak learners (decision trees) will be combined to make predictions.
More estimators improve accuracy but increase training time.

3. Maximum Depth (Decision Tree, Random Forest, Gradient Boosting)

Controls the depth of decision trees, preventing overfitting.
Shallow trees may underfit, while deep trees may overfit.

4. C (SVM - Regularization Parameter)

Determines the trade-off between achieving a low error and maintaining a simple model.
Higher values fit the training data better but may overfit.

5. K (K-Nearest Neighbors - Number of Neighbors)

Defines how many nearest data points are considered when making a prediction.
A low K value makes the model sensitive to noise, while a high K value smoothens predictions.

6. Boosting Parameters (Gradient Boosting, XGBoost, etc.)

Includes subsample ratio, minimum child weight, and gamma to control how aggressively the model learns.

Tuning these ensures a good balance between accuracy and generalization.

7. Hyperparameter Optimization Techniques

Titanic Survival Prediction

- Grid Search: Tries multiple combinations of hyperparameters to find the best set.
- Random Search: Randomly samples hyperparameter values for efficiency.
- Bayesian Optimization: Uses probabilistic models to find the optimal combination.

Model Deployment for Titanic Survival Prediction:

Overview of Deployment Process

Model deployment involves integrating the trained machine learning model into a user-friendly web interface, allowing users to input data and receive predictions in real time. For this project, we deployed the Titanic Survival Prediction model using Flask, a lightweight web framework in Python. Below is a step-by-step breakdown of the deployment process.

1. Project Structure

```
/Titanic_Prediction
|— index.html      # Frontend user interface
|— main.py         # Loads and preprocesses the model
|— app.py          # Flask backend API
|— titanic_model.pkl # Saved trained model
|— static/         # (If needed) For storing CSS, JS, or images
|— templates/      # Contains index.html
```

2. Creating the User Interface ([index.html](#))

We designed an HTML page to allow users to input details such as age, gender, ticket class, fare, and cabin information.

- HTML Form: Provides fields for user input.
- CSS Styling: Ensures a structured and clean design.
- JavaScript (if needed): Manages client-side interactions.

3. Backend ([app.py](#)) – Flask API

Flask serves as the backend, handling user input, loading the trained model, and returning predictions.

Steps Performed in [app.py](#)

1. Load the trained model ([titanic_model.pkl](#)).

Titanic Survival Prediction

2. Receive user input from the form.
3. Preprocess the input to match the model's expected format.
4. Make a prediction using the model.
5. Return the prediction to the user.

4. Running the Application

Steps to Run Locally in PyCharm or Terminal

Ensure Flask is installed

`pip install flask`

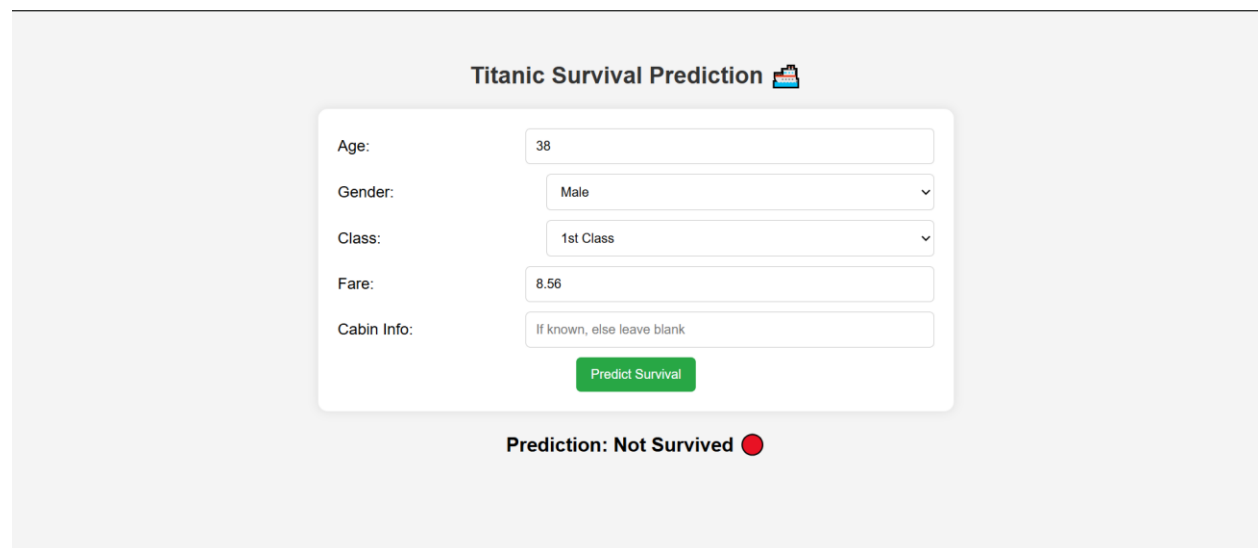
Run `app.py` in PyCharm or terminal

`python app.py`

1. Access the web interface
 - Open a web browser and go to `http://127.0.0.1:5000/`.
 - The form will appear, and you can input values to get predictions.

Final Output

- ✓ Users can enter Titanic passenger details on the webpage.
- ✓ Clicking "Predict Survival" sends data to Flask API.
- ✓ The trained model processes the input and returns a prediction.
- ✓ The result ("Survived" or "Did Not Survive") is displayed on the webpage.



The screenshot shows a web application titled "Titanic Survival Prediction" with a small icon of a ship. Below the title is a form with the following fields:

- Age: 38
- Gender: Male (dropdown menu)
- Class: 1st Class (dropdown menu)
- Fare: 8.56
- Cabin Info: If known, else leave blank

Below the form is a green button labeled "Predict Survival". Below the button, the prediction result is displayed: "Prediction: Not Survived" followed by a red circle icon.