Titanic Survival Prediction using Machine Learning

Project Abstract | Built-to-order software project | projectech.in

1. Introduction / Background

The Titanic dataset - 891 training passengers from the 1912 disaster, 38.4% survived - is the canonical first classification project in machine learning, released as the Kaggle competition 'Titanic: Machine Learning from Disaster'. Most student builds waste it: they one-hot encode the raw columns, fit a model, and report an accuracy number nobody can learn from. This project takes the opposite approach, treating the dataset as a feature-engineering exercise first and a modelling exercise second, and ships an explainable web demo alongside the training pipeline.

2. Problem Statement

Survival was driven by interactions the raw columns hide: class crossed with sex ('women and children first' played out as 74.2% female vs 18.9% male survival), family size (small families beat both solo travellers and very large ones), and title extracted from names. A model trained on raw columns alone cannot see these patterns cleanly. The project addresses this by engineering Title, FamilySize, IsAlone, Deck and LogFare with documented justification, handling missing values explicitly instead of silently dropping rows, and tuning a gradient-boosting classifier with stratified cross-validation.

3. Objectives

1. Engineer 8 meaningful features from the 12 raw Titanic columns, each justified with data. 2. Handle missing values explicitly: Age by title-median, Embarked by mode, Cabin via deck/missing flag. 3. Train a gradient-boosting classifier tuned with stratified 5-fold cross-validation. 4. Evaluate once on a held-out test set with accuracy, F1, ROC-AUC and confusion matrix, all framed as design targets. 5. Ship an explainable web demo showing per-factor contributions to every prediction.

4. Proposed Methodology

Title is parsed from the Name string with regex; FamilySize = SibSp + Parch + 1; IsAlone flags solo travellers; Deck is the first letter of Cabin with a missing indicator; Fare is log-transformed. Age (177 missing) is imputed by the median of the passenger's title group. A GradientBoostingClassifier is tuned over depth and learning rate with stratified 5-fold CV, then evaluated once on a held-out 20% split. The demo embeds a simplified scoring function over the same engineered features so predictions are explainable factor-by-factor.

5. System Architecture / Working

Training pipeline: loading -> feature engineering -> missing-value handling -> gradient boosting -> 5-fold CV tuning -> held-out evaluation -> metric plots. Demo app: three hash-routed views - Survival Predictor (segmented controls and sliders building a passenger profile, live probability gauge, per-factor contribution list), Dataset (documented statistics, survival-by-class and survival-by-sex charts, feature-engineering table), Model and Metrics (pipeline steps, design-target confusion matrix, importance bars). Changing any profile input re-scores instantly and re-ranks the factor contributions.

6. Hardware / Software Requirements

Software: Python 3 with scikit-learn, NumPy, pandas, Matplotlib and Jupyter for training; any modern browser for the demo. Hardware: any laptop or desktop; training completes in approximately 1-3 minutes on CPU. Data: Titanic competition CSVs from Kaggle (public, approx. 60 KB). No GPU required.

7. Expected Outcomes

A trained gradient-boosting pipeline with design targets of approximately 82.1% accuracy, 0.74 F1 and 0.87 ROC-AUC on the held-out test set - recorded as actuals in the report after the training run. An explainable single-file demo, a full project report with feature-engineering rationale, a presentation and a viva Q&A; set covering boosting, cross-validation, leakage and imputation strategy.

8. Applications

Teaching feature engineering, cross-validation and explainability in ML courses; a template pipeline reusable for other tabular classification datasets; demonstration of honest small-data ML reporting for student portfolios; and a starting point for extensions such as model comparison or calibration analysis.

9. Future Scope

Model comparison (random forest, logistic regression, XGBoost) with statistical testing; probability calibration curves; SHAP values in the demo for per-prediction explanations; submission pipeline to the Kaggle leaderboard using the 418-row test set; and fairness-style analysis of how the model's reliance on class and sex reflects historical inequity - as analysis, not as a product claim.

10. Conclusion

The project shows that on a small, well-understood dataset, the engineering around the model matters more than the model choice: careful features, honest missing-value handling and

explainable outputs. That discipline is the real deliverable, and it is what makes the work defensible in a viva and honest as a published listing.

Keywords: titanic survival prediction, kaggle, gradient boosting, feature engineering, classification, final year project