Titanic Survival Prediction Using Supervised Machine Learning

ZIHAO SHENG, The University of British Columbia, Canada

INAARA RAJWANI, The University of British Columbia, Canada

YUE WANG, The University of British Columbia, Canada

This project develops compact yet accurate classifiers for predicting passenger survival on the Titanic using an augmented Titanic dataset with engineered demographic, ticket, and cabin-related features. Following the project requirements, we adopted a fully reproducible workflow with a fixed random seed of 42 and a stratified 75/25 train/test split to preserve class balance. Key preprocessing steps addressed substantial missingness (notably Cabin) by retaining the provided numeric `cabin_score` and introducing explicit missingness indicators (e.g., `cabin_missing` and `age_missing`), while removing identifier-like variables that provide no predictive signal and may induce leakage.

We evaluated three complementary modeling approaches: an interpretable baseline using logistic regression with backward stepwise feature selection driven by cross-validated AIC, a nonparametric distance-based classifier (KNN) to capture local decision boundaries, and an ensemble tree-based model (Random Forest) to model nonlinear interactions and higher-order feature effects. To reduce systematic bias for young male passengers, we incorporated a *Master* indicator derived from

Authors' Contact Information: Zihao Sheng, The University of British Columbia, Kelowna, BC, Canada, zihao.sheng@student.ubc.ca; Inaara Rajwani, The University of British Columbia, Kelowna, BC, Canada, inaarari11@gmail.com; Yue Wang, The University of British Columbia, Kelowna, BC, Canada, yuewang2019@u.northwestern.edu.

passenger titles. Across models, we compared performance on a held-out test set using accuracy and AUC to assess both discrimination and generalization.

1 Introduction

Binary classification is a core task in supervised machine learning, with applications ranging from fraud detection and medical diagnosis to customer churn prediction. The objective is to learn a mapping from observed covariates to class labels that generalizes to unseen cases. In applied settings, predictive performance must be balanced with reproducibility, computational efficiency, and interpretability, particularly when model outputs are used to support real-world decision-making.

The Titanic survival prediction problem serves as a canonical benchmark for binary classification while exhibiting many challenges common in practical tabular datasets. The data contain heterogeneous predictors, including demographic attributes (e.g., sex and age), socioeconomic indicators (e.g., passenger class and fare), family structure variables (e.g., numbers of siblings, spouses, parents, and children), and travel-related identifiers (e.g., ticket and cabin). In addition to these original attributes, the augmented dataset used in this project includes engineered features derived from passenger names, tickets, and cabin information, such as title groups, family size, indicators of whether a passenger was traveling alone, group size by ticket, and deck-level cabin encodings. These features aim to capture social context, travel group structure, and cabin location, which plausibly influenced survival outcomes during evacuation.

From a data quality perspective, the dataset exhibits substantial missingness (most notably in the Cabin field), mixed data types (categorical and continuous), and identifier-like variables that provide little predictive signal and may induce data leakage if used

naively. These characteristics motivate careful preprocessing, explicit handling of missing values, and principled feature selection prior to model fitting. The modeling objective in this project is therefore not simply to maximize predictive accuracy, but to construct a small yet performant classifier that balances accuracy with simplicity and interpretability.

To ensure reproducibility and fair comparison, we adopt a fixed random seed of 42 and a stratified 75/25 train/test split, with all preprocessing and model selection performed on the training data and final evaluation conducted on a held-out test set. Within this unified experimental framework, we compare logistic regression as an interpretable baseline, k-nearest neighbors (KNN) as a nonparametric method capturing local decision boundaries, and random forests as an ensemble-based approach capable of modeling nonlinear interactions. This design enables a systematic examination of trade-offs between model simplicity, interpretability, and predictive performance on a realistic tabular classification task.

2 Methodology

This section presents the end-to-end modeling pipeline, including data preprocessing, feature engineering, and the supervised learning algorithms considered in this study. All modeling decisions, including feature selection and hyperparameter tuning, were performed exclusively on the training data to prevent information leakage, with the held-out test set reserved for final performance evaluation.

2.1 Data Preprocessing

The dataset contains passenger-level demographic and travel attributes, along with engineered variables intended to support modeling. A primary preprocessing challenge is

missing data, particularly for Cabin and Age. The Cabin field is missing for approximately 75% of passengers; however, the dataset provider has aggregated cabin-related information into a numeric cabin_score feature. We therefore retained cabin_score and introduced a binary cabin_missing indicator to explicitly encode whether the original cabin information was absent, while dropping the remaining raw cabin fields. Similarly, because a substantial fraction of passengers have missing Age values, we added an age_missing indicator to preserve the information conveyed by missingness itself.

To reduce the risk of information leakage and avoid spurious predictive signals, we removed identifier-like and high-cardinality fields that do not represent stable passenger characteristics. Specifically, we dropped PassengerId, Name, Ticket, booking_reference, service_id, and the raw cabin fields. In addition, original untransformed fare variables were removed when corresponding transformed versions were used. This step prevents the model from memorizing passenger identifiers or proxy keys and simplifies downstream modeling by restricting the feature set to variables with plausible behavioral or socioeconomic relationships to survival.

After preprocessing, the retained modeling variables include core demographic attributes (e.g., Pclass, Sex, Age), family structure variables (SibSp, Parch, family_size, is_alone), selected engineered features (e.g., name_length, title/title_group, name_word_count, ticket_group_size), explicit missingness indicators (age_missing, cabin_missing), and log-transformed fare features (log_Fare, log_fare_per_person, log_age_fare_ratio).

Finally, to satisfy reproducibility requirements, we created a stratified 75/25 train–test split with random_state = 42, keeping the test set untouched until final evaluation.

2.2 Random Forest

Random forest was adopted as a model because of its well know high level of prediction performance, relatively low level of computational demand and ease of implementation. Another benefit of Random forest is that users usually do not need to worry about manual predictor tuning as the model itself will randomly drop predictors. If after cross validation, the model performance suggested that there were indeed problems with some predictors (e.g. model construction errors, poor model performance), then manual predictor tuning will be considered.

After the data was cleaned, it was re imported into separate training and test variables and made sure that all variables had proper data types, including categorical data. The data were then further separated into data frame for response and data frame for all predictor data. After that, a random forest model was created with n=500 and all other attributes left as default to form a first impression of performance, and the model accuracy and importance graph was analyzed for anything out of the ordinary.

After the control model was completed, hyperparameter tuning was done using RandomSearchCV to find the optimal parameters in a wide range. A result was found, and a new random forest model formed with the parameters from the CV, and accuracy was once again calculated. An importance graph was once again generated, along with a confusion matrix graph, and the top of 2 random trees out of the forest was generated for a quick comparison.

Then, GridSearchCV was also tried, but due to computational restraints the range of parameters for GridSearchCV was not as wide as RandomSearchCV because of the way they both work. Results were once again used to construct another random forest model,

and accuracy was compared between all three models generated to determine which out of the no CV, RandomSearchCV, and GridSearchCV models performed the best.

2.3 K-Nearest Neighbors (KNN)

We adopt the K-Nearest Neighbors (KNN) algorithm as a nonparametric classifier to capture local structure in the feature space. Unlike linear models such as logistic regression, KNN does not impose a parametric form on the decision boundary. Instead, predictions are based on proximity in the feature space, under the assumption that passengers with similar demographic and travel profiles (e.g., class, age, and fare-related attributes) are likely to share similar survival outcomes. This property makes KNN a flexible baseline for assessing whether local neighborhood information provides predictive gains beyond linear effects.

Because KNN relies on distance computations, its performance is sensitive to feature scale. We therefore standardized all continuous input features using z-score normalization to ensure that no single variable dominated the Euclidean distance metric. Log-transformed features derived in the preprocessing stage were retained as inputs, but no additional distributional transformations were introduced specifically for KNN beyond the global preprocessing pipeline.

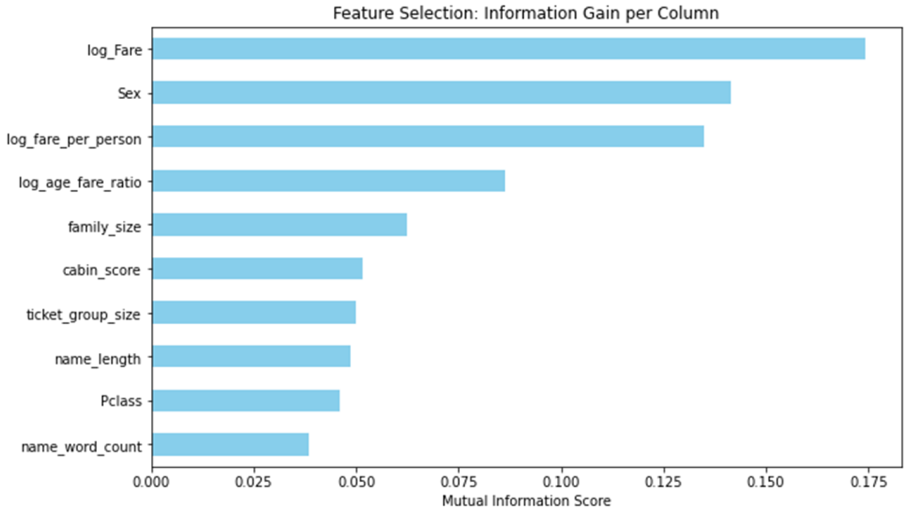

Fig. 1. Feature importance (information gain) used for KNN feature selection.

To reduce dimensionality and improve computational efficiency, we performed feature selection based on information gain, prioritizing predictors that exhibited strong marginal association with survival. This analysis consistently identified `Sex`, `Fare`-related variables, and `Pclass` as highly informative. The final KNN model was trained on a subset of 12 engineered features spanning demographic attributes, transformed numerical variables, and selected categorical indicators. This feature subset balances representational richness with neighborhood stability in the high-dimensional space.

2.4 Logistic Regression and Feature Selection

Logistic regression is adopted as an interpretable baseline model for binary survival prediction. Given an input feature vector $\mathbf{x}$, the model estimates the survival probability

via the logit link function:

$$\log \frac{P(Y = 1 \mid \mathbf{x})}{1 - P(Y = 1 \mid \mathbf{x})} = \eta(\mathbf{x}) = \beta_0 + \sum_{j=1}^{p} \beta_j x_j, \tag{1}$$

where $\eta(\mathbf{x})$ denotes the linear predictor and β_j are model coefficients. This formulation yields calibrated probabilities and enables direct interpretation of covariate effects through changes in the log-odds of survival.

A central challenge in this project is the presence of a large set of correlated engineered features (e.g., family-related variables, cabin-derived scores, and missingness indicators). Naïve stepwise selection based on single-sample AIC is known to be overly optimistic in such settings and may result in unstable feature subsets. To improve robustness and explicitly target generalization performance, we implement a custom backward stepwise selection procedure driven by cross-validated AIC (CV-AIC).

The procedure begins from a full candidate specification containing all admissible predictors after preprocessing. At each iteration, one predictor is removed at a time, and the reduced model is refitted using K-fold cross-validation on the training set. For each fold, the validation log-likelihood is computed and converted into an AIC score. These fold-level AIC values are then averaged to obtain a mean CV-AIC for the candidate model. The predictor whose removal yields the largest improvement in mean CV-AIC is eliminated. This process is repeated until no further CV-AIC improvement is observed, resulting in a parsimonious model selected to minimize estimated out-of-sample information loss.

This CV-AIC–driven backward elimination framework explicitly incorporates validation performance into the feature selection loop, mitigating overfitting risks associated with correlated engineered features and improving the stability of the selected model. The

final logistic regression specification is therefore determined jointly by interpretability considerations and cross-validated model fit, ensuring that coefficient estimates remain meaningful while maintaining competitive predictive performance.

3 Experiments

This section reports the experimental setup and empirical results for the proposed models. We describe the evaluation protocol and performance metrics, followed by method-specific results and analyses. All experiments were conducted under a unified preprocessing pipeline and data split to ensure fair comparison across models.

3.1 Experimental Setup and Evaluation Metrics

All models were trained under a fixed random seed (`random_state = 42`) using a stratified 75/25 train/test split based on the `Survived` outcome. The split was performed after data preprocessing, and the same training and test partitions were used across all methods to ensure a fair comparison. All model selection and hyperparameter tuning procedures were carried out exclusively on the training data, with the held-out test set reserved for final evaluation.

Model performance was evaluated using three complementary criteria: test accuracy, the area under the ROC curve (AUC), and the confusion matrix. Test accuracy provides an intuitive measure of overall classification performance at a fixed decision threshold. The ROC curve and its corresponding AUC quantify the model's ability to rank positive and negative instances across all possible thresholds and are therefore insensitive to the specific classification cutoff. The confusion matrix provides a class-wise breakdown of

prediction outcomes, enabling inspection of error types (false positives and false negatives) and potential asymmetries in misclassification behavior.

3.2 Random Forest Results

An initial random forest model was generated with mainly default parameters and n=500 as a baseline, with a prediction accuracy of 80.2%. An importance chart was also generated to see if there were any predictor values that might be causing a problem, and there were no high importance irregularities to note.

RandomSearchCV was then applied with a generous range of parameter values, which can be done as RandomSearchCV is very computationally efficient, and best parameters was then used to form a second random forest model, and summary and importance chart was once again generated.

```
ROC-AUC: 0.8605499915124767
Accuracy: 0.8071748878923767
[[118  19]
 [ 24  62]]
              precision    recall  f1-score   support

           0       0.83      0.86      0.85       137
           1       0.77      0.72      0.74        86

    accuracy                           0.81       223
   macro avg       0.80      0.79      0.79       223
weighted avg       0.81      0.81      0.81       223
```

Fig. 2. RandomSearchCV Parameter Random Forest Model Summary

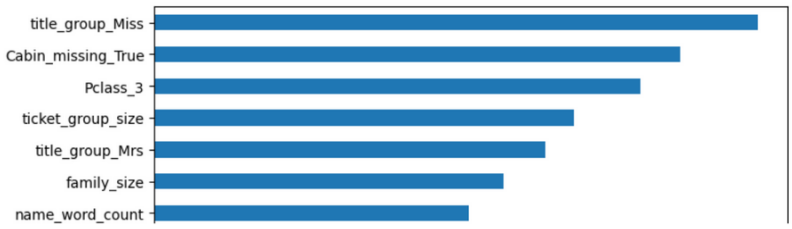

Fig. 3. RandomSearchCV Parameter Random Forest Model Importance

This time however, the importance did have a worrying parameter, mainly cabin_missing_true. Some quick research was done and it appears that cabin data was primarily missing for people of lower wealth classes, as well as people that did not board at the initial port of departure (paid a lower ticket cost compared to people that did board at the initial port of departure). As this data was correlated with the wealth class of passengers, the parameter is left in.

A third random forest model was then constructed using GridSearchCV to test if it would perform better than RandomSearchCV. The parameters for GridSearchCV was not as wide as previous because of the calculation power limit, as GridSearchCV tests every combination of every parameter passed to it, and the results summary were not as good as RandomSearchCV, and even worse than the base model.

The final result of the Random Forest models was that the model result from Random-SearchCV performed the best out of the three models tried. The best parameters for that random forest was the following.

```
Best hyperparameters: {'max_depth': 28, 'max_features': 'sqrt', 'min_samples_leaf': 3, 'min_samples_split': 8, 'n_estimators': 919}
```

Fig. 4. RandomSearchCV Hypertuned Parameter Result

3.3 K-Nearest Neighbors (KNN) Results

The K-Nearest Neighbors (KNN) model was tuned using grid search with 5-fold cross-validation on the training set to optimize generalization performance. The number of neighbors K was varied from 3 to 30, and both Euclidean and Manhattan distance metrics were evaluated. Small values of K (< 5) exhibited high variance and sensitivity to noise, while large values of K (> 25) led to oversmoothing and increased bias. The best-performing configuration was obtained at $K = 19$ with the Euclidean distance metric, which provided a favorable balance between capturing local structure and maintaining prediction stability.

```
Number of features used: 12
Final Test Accuracy: 0.8072
------------------------------
Final Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.85      0.84       137
           1       0.75      0.74      0.75        86

    accuracy                           0.81       223
   macro avg       0.80      0.80      0.80       223
weighted avg       0.81      0.81      0.81       223
```

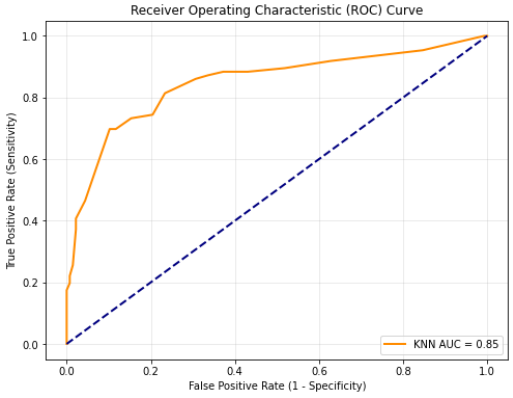

Fig. 5. Performance of the KNN model on the test set. Left: test accuracy. Right: ROC curve.

Using the optimized configuration ($K = 19$, Euclidean distance), the KNN classifier achieved a test accuracy of 80.72% on the held-out test set with AUC ≈ 0.85. Analysis of the confusion matrix indicates a conservative prediction behavior: the model maintains a relatively low false positive rate (rarely predicting survival for non-survivors) at the expense of a higher false negative rate, suggesting that strong neighborhood similarity is required before assigning the positive class. The ROC curve exhibits strong curvature

toward the top-left corner, confirming favorable discriminative ability across classification thresholds despite the model's cautious decision boundary in ambiguous regions of the feature space.

3.4 Logistic Regression Results

Starting from a full candidate specification after preprocessing and feature engineering, we applied backward stepwise feature selection driven by cross-validated AIC (CV-AIC) to obtain a parsimonious and more stable model. This approach mitigates the optimism and instability of single-sample AIC in the presence of correlated engineered predictors. During refinement, we identified pronounced multicollinearity among family-related variables; in particular, `SibSp` and `Parch` are highly correlated with the engineered `family_size` feature. To reduce redundancy and stabilize coefficient estimates, we removed `family_size` and retained the original family covariates.

Error analysis further revealed that a baseline specification tended to underestimate survival probabilities for young male passengers when `male` was treated as a homogeneous group. Consistent with historical evacuation practices prioritizing women and children, we introduced a binary `Master` indicator derived from passenger titles to distinguish boys from adult men. Incorporating this feature improved calibration for this subgroup without materially increasing model complexity.

The final selected logistic regression estimates the survival probability as

$$\Pr(\text{Survived} = 1 \mid \mathbf{x}) = \frac{1}{1 + e^{-\eta(\mathbf{x})}}, \tag{2}$$

with linear predictor

$$\eta(\mathbf{x}) = 3.9152 - 3.1837 \cdot \mathbb{I}(\text{Sex} = \text{male}) - 1.0695 \cdot \text{Pclass} - 0.4769 \cdot \text{SibSp} + 3.5465 \cdot \mathbb{I}(\text{Master}). \tag{3}$$

All retained coefficients are statistically significant. The negative coefficients for `male` and `Pclass` indicate substantially lower survival odds for males and passengers in lower classes, while the negative effect of `SibSp` suggests decreased survival probability with larger sibling/spouse groups, plausibly reflecting evacuation constraints. The strongly positive `Master` coefficient captures the survival advantage of young boys relative to adult males, partially offsetting the general male disadvantage and correcting a systematic miscalibration observed in earlier specifications.

```
Test Accuracy (threshold=0.5): 0.8296

Confusion Matrix:
        Pred 0  Pred 1
True 0     122      15
True 1      23      63

Classification Report:
              precision    recall  f1-score   support

           0     0.8414    0.8905    0.8652       137
           1     0.8077    0.7326    0.7683        86

    accuracy                         0.8296       223
   macro avg     0.8245    0.8115    0.8168       223
weighted avg     0.8284    0.8296    0.8279       223
```

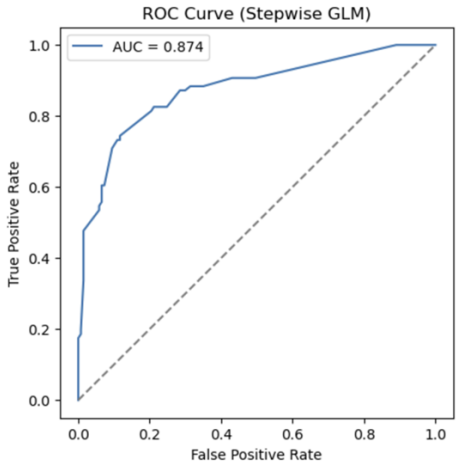

Fig. 6. Performance of the logistic regression model on the test set. Left: test accuracy. Right: ROC curve.

On the held-out test set, the final logistic regression achieved 82.96% accuracy with AUC $\approx$ 0.87, indicating strong discriminative ability for a compact linear model. Overall, these results show that careful feature selection, explicit control of multicollinearity, and

targeted error-driven feature engineering can substantially enhance the performance of an interpretable baseline classifier.

4 Conclusion

Table 1. Performance comparison of supervised learning models on the Titanic test set.

Model	Test Accuracy (%)	ROC–AUC	# Predictors
Random Forest	80.71	0.86	17
KNN	80.72	0.85	12
Logistic Regression (GLM)	82.96	0.874	4

In this project, we compared several supervised learning approaches for predicting passenger survival on the Titanic using an augmented and carefully cleaned dataset. A major focus was ensuring that performance differences reflected modeling choices rather than inconsistencies in preprocessing, which motivated the use of a unified data-cleaning pipeline, explicit handling of missing values, and a fixed train/test split for all experiments.

Although Random Forest and KNN models achieved solid predictive performance, the most notable result is that the logistic regression model ultimately performed best. After applying backward stepwise feature selection guided by cross-validated AIC, the final linear model achieved the highest test accuracy and AUC while relying on only four predictors. This result was somewhat surprising given the flexibility of the nonlinear models, but it highlights how effective feature selection and domain-aware modeling can be when working with structured tabular data.

Overall, the results demonstrate that increased model complexity does not automatically translate into better generalization. In this setting, a well-specified linear model with

carefully chosen predictors outperformed more flexible alternatives while offering clear interpretability. These findings reinforce the value of disciplined preprocessing, validation-driven feature selection, and simplicity when modeling real-world tabular datasets such as the Titanic survival data.

5 References

5.1 Random Forest

- https://www.datacamp.com/tutorial/random-forests-classifier-python
- https://www.geeksforgeeks.org/machine-learning/random-forest-hyperparameter-tuning-in-python/

5.2 K-Nearest Neighbors (KNN)

- Cover, T. and Hart, P. (1967). *Nearest Neighbor Pattern Classification.* IEEE Transactions on Information Theory, 13(1):21–27. https://ieeexplore.ieee.org/document/1053964
- Pedregosa, F., et al. (2011). *Scikit-learn: Machine Learning in Python.* Journal of Machine Learning Research, 12:2825–2830. https://scikit-learn.org/