Data 572: Project working title

Avalvir Kaur Sekhon, Zihuan Liu & Jordan Kaseram

February 6, 2026

Abstract

Introduction

hello

Methodology

Titanic Data Set

The Titanic dataset consists of passenger-level information from the sinking of the RMS *Titanic*. Each observation corresponds to an individual passenger, with the response variable indicating survival status (1 = survived, 0 = did not survive). The dataset contains a mixture of demographic, socioeconomic, and voyage-related features, including age, sex, passenger class, fare paid, family composition, and cabin information.

In total, the dataset contains 891 observations with a diverse set of categorical and numerical variables. As is typical of real-world data, the raw dataset presents several challenges, including missing values, highly sparse columns, and correlated features derived from similar underlying information.

These characteristics make the Titanic dataset well suited for evaluating classification methods while also requiring careful preprocessing and feature selection prior to model fitting.

Data Preprocessing

Data preprocessing was performed using Python and the pandas library, which provides flexible and efficient tools for data cleaning, transformation, and preparation prior to machine learning analysis. Identifier variables such as passenger identifiers, ticket numbers, and booking references were removed, as they do not contain predictive information and may introduce unnecessary noise into the models.

Missing data were handled using simple, distribution-preserving imputation techniques. Missing values in passenger age, a numerical variable, were imputed using the median age within each passenger class. This approach allows age estimates to reflect socioeconomic differences across classes while remaining robust to outliers. Missing values in the port of embarkation, a categorical variable, were imputed using the mode, preserving the empirical distribution of this feature.

Categorical predictors were converted into numerical form using one-hot encoding creating dummy variables. To reduce redundancy and limit multicollinearity, several highly correlated or derived features were evaluated and pruned. For example, the variables `SibSp`, `Parch`, and `is_alone` were replaced by a single variable `family_size`.

The resulting feature set provides a stable and interpretable foundation for training supervised machine learning models for binary classification.

Resampling and Validation Strategy

To ensure reliable model evaluation and reduce overfitting, the dataset was partitioned into training (75%) and testing (25%) sets using a stratified split to preserve the original survival class proportions. Stratification is particularly important for the Titanic dataset due to its class imbalance between survivors and non-survivors.

All model training, hyperparameter tuning, and feature selection were performed exclusively on the training set using stratified k-fold cross-validation with $k = 5$. This resampling strategy maintains class balance within each fold and provides stable performance estimates while preventing information leakage.

Final model performance was evaluated on the held-out test set, which remained untouched during model selection. This approach yields an unbiased estimate of generalization performance and ensures fair comparison across all classification methods.

Model Selection

Three classification methods were selected to evaluate the predictive performance of the features present in the Titanic data set. These methods include: Logistic Regression, Linear Discriminant Analysis (LDA) and K-Nearest Neighbours (KNN). Logistic Regression and LDA were considered for their interpretability due to linearity [2]. Moreover, the Titanic prediction task is a binary response with a mix of numeric and categorical variables which these models perform well on [1].

These parametric tests assume a structured relationship between the predictors and the response, enabling stable parameter estimation and favourable bias-variance trade-offs. In contrast, KNN was included as a non-parametric test to capture potential non-linear patterns without assuming strong distribution assumptions. KNN provides a useful comparison model by classifying passengers based on identifying the observations that are nearest it rather than a global decision boundary [1].

Hyperparameter Tuning Strategy

Hyperparameter tuning was conducted using grid search, which is a simple memoryless method that can be used to explore predefined hyperparameter values [4]. This was combined with stratified cross-validation to identify the optimal model configuration.

For Linear Discriminant Analysis, hyperparameter tuning focused on the choice of solver (`svd` versus `lsqr`) and the use of covariance shrinkage, as shrinkage regularization is known to stabilize covariance estimation in the presence of correlated predictors and limited sample sizes [2, 4].

For Logistic Regression, tuning was performed over the regularization strength parameter C and the choice of ℓ_1 and ℓ_2 penalties, allowing control over model complexity and coefficient sparsity, which directly affects bias–variance trade-offs and interpretability in high-dimensional settings [1, 5].

For K-Nearest Neighbours, hyperparameters including the number of neighbours, distance metric, and weighting scheme were tuned to balance local versus global decision behavior and to regulate model sensitivity to noise and class overlap [1, 5].

Experiment

Experimental Design

Results and Analysis

Discussion

Conclusion

References

[1] James, G., Witten, D., Hastie, T., Tibshirani, R., & Taylor, J. (2023). An introduction to statistical learning: Python edition.

[2] Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning.

[3] Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. Annals of Eugenics, 7(2), 179-188.

[4] Franceschi, L., Donini, M., Perrone, V., Klein, A., Archambeau, C., Seeger, M., ... & Frasconi, P. (2025). Hyperparameter optimization in machine learning. Foundations and Trends® in Machine Learning, 18(6), 975-1109.

[5] Kuhn, M., & Johnson, K. (2013). Applied predictive modeling (Vol. 26, p. 13). New York: Springer.