Titanic Survival Prediction

DATA 572 Group Project - Group 13

Zaed Khan, Manpreet Singh, Yihang Wang

Abstract

This project explores the application of supervised machine learning techniques to predict passenger survival on the Titanic. Using an augmented dataset, we implemented and evaluated three classification algorithms: Logistic Regression, Random Forest, and K-Nearest Neighbors (KNN), along with a bonus Transformer model. The primary objective was to construct efficient models that maximize predictive performance while minimizing feature complexity. Our methodology involved rigorous data preprocessing, including median imputation for missing values, one-hot encoding for categorical factors, and stratified resampling to maintain class distribution. Through extensive hyperparameter tuning using GridSearchCV and feature selection using feature importance threshold, we optimized each model's performance. The experimental results indicate that the tuned K-Nearest Neighbors model achieved the highest classification accuracy (81%) on the test set, effectively capturing the local structure of the passenger data. Feature importance analysis across models consistently highlighted gender and socio-economic status as the most important factors of survival.

1 Introduction

Supervised learning is a branch of machine learning that relies on labeled datasets to train algorithms to identify underlying patterns and relationships. Within this framework, classification is a specific task where the model learns to assign input data into discrete, categorical labels - such as 0 vs. 1 - based on ground truth examples provided during training. The algorithm processes these training pairs to infer a mapping function that can accurately predict the correct output for new, unseen data. This process involves an iterative optimization of an objective function to minimize errors and enhance the model's predictive precision over time.

Despite its name, Logistic Regression [2] is a fundamental supervised learning algorithm used for binary classification rather than traditional regression. In a modern machine learning context, it serves as a linear classifier within the Generalized Linear Model (GLM) family. Unlike linear regression, which predicts continuous numerical values, logistic regression uses the sigmoid function to map input features to a probability value between 0 and 1. A decision threshold (typically 0.5) is then applied to these probabilities to categorize the input into one of two classes.

Random Forest [3] is another powerful ensemble (supervised) learning method that addresses the tendency of individual decision trees to overfit their training data. The algorithm builds a forest of multiple, uncorrelated decision trees. It operates on two key principles: bagging (bootstrap aggregating), where each tree is trained on a random subset of the data, and feature randomness, where only a random subset of features is considered at each split. For classification tasks, the Random Forest aggregates the predictions of all individual trees and outputs the class that receives the majority vote, significantly boosting the model's robustness and accuracy.

The K-Nearest Neighbors (KNN) [4] algorithm is one of the simplest and most intuitive supervised learning techniques. As a non-parametric model, KNN does not learn a fixed mathematical function during a training phase; instead, it memorizes the entire training dataset and performs computations only when a query is made. The algorithm functions on the principle of proximity, assuming that similar things exist in close proximity. To classify a new data point, KNN identifies the 'k' most similar training examples (neighbors) based on a distance metric and assigns the label that is most frequent among those neighbors.

Supervised machine learning has become a cornerstone of modern data analysis, providing powerful tools for classification tasks across diverse domains. In this project, we apply these methods to the classic Titanic dataset to predict passenger survival. The binary classification problem - determining survival (1) or non-survival (0) - serves as an excellent benchmark for comparing different algorithmic approaches.

The ability to accurately classify outcomes based on demographic and structural features is critical to make future predictions on new data. By analyzing the Titanic dataset [1], which includes attributes such as age, sex, passenger class, and fare, we aim to understand not just *who* survived, but *why* specific features hold predictive power. Our study focuses on building interpretable and performant models, emphasizing the trade-off between model complexity and accuracy.

2 Methodology

This section describes the full workflow we used to build, tune, and evaluate the models. The overall goal is to keep the pipeline simple and reproducible, and to compare models fairly under the same train/test split and evaluation metrics.

2.1 Data preprocessing

We used the provided augmented Titanic dataset, where the target variable is `Survived` (1 = survived, 0 = not survived). Before modelling, we removed columns that are not useful for prediction or do not generalize well, such as identifiers and high-cardinality text fields (e.g., passenger name and ticket-related fields). Keeping these columns usually leads to noise and makes the model harder to interpret.

For missing values, we applied straightforward imputations:

- **Age:** imputed with the median from the training set, since age has outliers and median is more robust.
- **Embarked (if missing):** imputed with the most frequent category (mode).

We did not use complex imputation methods because the dataset is not very large and a simple approach is easier to explain and reproduce.

2.2 Encoding and scaling

Several features are categorical. We encoded them in a standard way:

- **Sex:** encoded as binary (0/1).
- **Embarked and Pclass (if treated as categorical):** one-hot encoded to avoid forcing an artificial ordering.

For numerical features, we applied standardization (mean 0, standard deviation 1) when needed. This is especially important for **KNN**, because distance-based methods are sensitive to feature scale. Logistic Regression can also benefit from scaling when regularization is used, while Random Forest is generally less sensitive to scaling.

2.3 Train/test split and evaluation setup

We split the dataset into training (75%) and test (25%). To keep the class distribution consistent across splits, we used **stratified splitting** based on `Survived`. We fixed a random seed so that results are reproducible and all models are compared on the same test set.

For evaluation, we report:

- **Accuracy** on the test set
- **Weighted F1-score** on the test set
- **Confusion matrix** for error analysis

We used weighted F1 because the class distribution is not perfectly balanced, and we want a metric that reflects performance on both classes.

2.4 Model selection

The project requires three classical supervised learning methods. We trained and evaluated:

- **Logistic Regression (LR):** used as a simple and interpretable baseline.

- **Random Forest (RF):** used to capture non-linear relationships and feature interactions.
- **K-Nearest Neighbors (KNN):** used to capture local structure in the feature space.

As an additional (bonus) model, we also evaluated a **Transformer** [5] model to compare deep learning performance on tabular data, but it is not used as the final selected model.

2.5 Hyperparameter tuning and feature reduction

To improve performance, we tuned model hyperparameters using **GridSearchCV** with **5-fold stratified cross-validation**. This provides a more reliable estimate than a single validation split.

We also compared two feature settings:

- **Full feature set:** all selected processed features.
- **Reduced feature set:** a smaller subset based on feature importance / coefficient strength.

The reduced setting is included because the project goal is not only high accuracy, but also efficiency (fewer predictors) when performance is similar. In practice, feature reduction can remove redundant information and reduce overfitting risk.

Overall, this pipeline allows us to: (1) build comparable models, (2) tune them fairly using the same CV strategy, and (3) evaluate them on the same held-out test set with consistent metrics.

3 Experiment

This section presents the experimental design, results, and discussion. All classical models were evaluated on the same test set, and we report both performance metrics and confusion matrices to better understand the types of errors each model makes.

3.1 Experimental design

We trained three classical models (Logistic Regression, Random Forest, and KNN) and compared:

- baseline vs tuned versions
- full features vs reduced features (when applicable)

Hyperparameter tuning was done using GridSearchCV with 5-fold stratified cross-validation on the training set. After tuning, we refit the best estimator on the full training set and evaluated once on the held-out test set. This avoids leaking test information into training.

The main metrics reported are test Accuracy and weighted F1-score. We also generated classification reports and confusion matrices to inspect precision/recall trade-offs. For example, in this dataset, predicting survival (class 1) is typically harder than predicting non-survival (class 0), so looking at the confusion matrix helps explain the model behavior beyond one number.

3.2 Logistic Regression results

Logistic Regression performed strongly as a baseline model and remained competitive after tuning. Since LR is linear, it tends to generalize well on smaller datasets and is less likely to overfit compared to more flexible models.

After tuning the regularization strength (parameter C) and solver settings, the tuned LR achieved stable test performance. In our confusion matrices, LR usually made fewer extreme mistakes and produced a consistent balance between false positives and false negatives. Another benefit of LR is interpretability: coefficients help confirm intuitive patterns, such as survival being strongly related to gender and socio-economic status.

We also tested a reduced-feature version. In our results, reduced LR was very close to the full-feature tuned LR, which suggests that many predictors are redundant once the strongest signals (e.g., sex and class-related features) are included.

3.3 Random Forest results

Random Forest provides a non-linear alternative that can model feature interactions. However, in tabular datasets with limited size, RF can overfit if not controlled.

We observed that the untuned RF was not the strongest model on the test set. After tuning parameters such as number of trees, maximum depth, and minimum samples per leaf, the RF model became more stable. The tuned RF generally improved compared to the untuned version, showing that tuning helps reduce overfitting.

Feature reduction was also applied using feature importance. The reduced RF kept similar performance while using fewer predictors. This suggests that feature importance can help remove less useful features without hurting overall performance. In addition, reduced RF can be faster to train and easier to communicate in the report.

3.4 KNN results and model comparison

KNN achieved the strongest test-set performance among the classical models in our comparison table, with the highest test Accuracy and weighted F1-score. This suggests that survival patterns in this dataset have some local structure: passengers with similar feature profiles tend to have similar outcomes, and a nearest-neighbor approach can capture this effectively.

KNN is sensitive to scaling and to the choice of k. After tuning the number of neighbors and distance settings, the tuned KNN improved and became the best-performing classical model in our results. The confusion matrix for KNN also shows fewer total mistakes compared to the other classical models in our final comparison.

At the same time, KNN is less interpretable than Logistic Regression, and its performance can be more sensitive to small dataset changes. For this project, we still select KNN as the final classical model because the project focuses on test-set performance, and KNN leads on both Accuracy and weighted F1.

3.5 Test-set results (Accuracy and weighted F1)

Table 1 matches the comparison table from `project.ipynb`. Among the models, tuned KNN ranks highest on both Accuracy and weighted F1.

Table 1: Model comparison on the test set.

Model	**Accuracy (%)**	**Weighted F1-score**
KNN (Tuned)	80.72	0.8049
Logistic Regression (Baseline)	79.82	0.7975
Logistic Regression (Tuned, Full)	78.03	0.7777
Logistic Regression (Tuned, Reduced)	78.03	0.7777
Random Forest (Tuned, Full)	76.68	0.7644
Random Forest (Tuned, Reduced)	74.89	0.7455
Transformer (Bonus)	80.27	0.7946

3.6 Bonus: Transformer evaluation

We also evaluated a Transformer model as a bonus. The goal is not to replace classical models, but to provide a comparison point.

In our experiments, the Transformer achieved competitive accuracy, but this dataset is relatively small for deep learning. Transformers typically benefit from large datasets and longer training, and they are also harder to interpret. In addition, the project rubric focuses mainly on classical supervised learning methods. Therefore, we report Transformer results as a bonus only and do not treat it as the final selected model.

3.7 Final discussion and selected model

Based on the test-set comparison table and visualizations, **KNN (tuned)** is selected as the final classical model because it achieved the best overall test performance (highest Accuracy and weighted F1-score among classical models).

We also observed two useful practical points:

- **Tuning matters:** hyperparameter tuning improves performance for RF and KNN, and can also stabilize LR.

- **Feature reduction can work:** using feature importance to reduce predictors often keeps similar performance while simplifying the model.

Overall, the experiments show that classical methods can perform very well on this Titanic dataset, and the final choice is driven by test-set results and a fair comparison under the same evaluation setup.

4 Conclusion

This project successfully completed comprehensive supervised learning experiments to predict Titanic survival. By rigorously preprocessing data, handling missing values with median imputation, and employing stratified resampling, we ensured a robust evaluation framework.

Our key finding is that K-Nearest Neighbors serves as the most effective classifier for this specific task, outperforming both linear and ensemble tree-based methods. This may be due to the small size of the data, which prevented better algorithms like random forests from learn the feature relationships well. The analysis justified the importance of feature selection, showing that a compact set of features centered on demographics and class can yield high-performance models. The experiments also highlighted that state-of-the-art deep learning methods like Transformers are not automatically superior for all data types and scales.

5 Contributions

Zaed Khan

- **Data Preprocessing:** Conducted exploratory data analysis, comprehensive data cleaning, including median imputation for missing values and handling of null entries.

- **Feature Engineering:** Managed categorical feature encoding, feature selection, and data standardization to improve model efficiency.

- **Bonus Modeling:** Developed, implemented, and evaluated the Transformer model to benchmark against classical models.

Manpreet Singh

- **Model Development:** Designed and implemented the core classifiers, including Logistic Regression, Random Forest, and K-Nearest Neighbors (KNN).
- **Model Training & Evaluation:** Managed the end-to-end training process and performed performance evaluation on the training datasets.
- **Optimization:** Conducted hyperparameter tuning and refined the feature set through feature importance analysis to maximize predictive accuracy.

Yihang Wang

- Ran test-set predictions for Logistic Regression, Random Forest, KNN, and the bonus Transformer model.
- Reported evaluation results using accuracy and weighted F1-score, and generated classification reports.
- Created visual summaries (confusion matrices and model-comparison charts) and wrote the final model comparison and selection discussion.

References

[1] Data Science Dojo. (n.d.). Titanic Dataset. GitHub Repository.

[2] Chung, M. K. (2020). Introduction to logistic regression. ArXiv. https://arxiv.org/abs/2008.13567

[3] Louppe, G. (2014). Understanding Random Forests: From Theory to Practice. ArXiv. https://arxiv.org/abs/1407.7502

[4] Cunningham, P., Delany, S. J. (2020). K-Nearest Neighbour Classifiers: 2nd Edition (with Python examples). ArXiv. https://doi.org/10.1145/3459665

[5] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., Polosukhin, I. (2017). Attention Is All You Need. ArXiv. https://arxiv.org/abs/1706.03762