**Survival Prediction Model: The Titanic Dataset**

**Abstract**

**Background:** This report explores a supervised classification approach to predicting the survival of Titanic passengers. The target variable “Survived” is binary, and explanatory features cover a wide range of demographic, socio-economic, and travel-related details.

Methods: Two primary models—Logistic Regression and Random Forest—were developed. Data preprocessing included handling missing values, feature engineering (e.g., extracting passenger titles, simplifying cabin info), and encoding categorical variables. Each model was evaluated using accuracy, precision, recall, F1-score, and AUC-ROC, with cross-validation and grid search employed for hyperparameter tuning.

**Results:** Logistic Regression provided quick interpretability but plateaued at approximately 85% accuracy (AUC of 0.87). Random Forest required more computational overhead yet showed stronger performance, reaching about 86% accuracy with an AUC of 0.91. Feature importance analyses identified passenger sex, title, and fare-related attributes as key predictors of survival.

**Conclusions:** Random Forest emerged as the more accurate model, reflecting its ability to capture complex patterns in the data. Nevertheless, Logistic Regression remains an attractive option for its simplicity and transparency. Future improvements may involve more extensive hyperparameter searches, advanced ensemble techniques, and deeper feature engineering to further enhance predictive power.

*Keywords*: Titanic Survival Prediction, Supervised Classification, Logistic Regression, Random Forest, Data Preprocessing, Feature Engineering, Model Evaluation Metrics, AUC-ROC, Hyperparameter Tuning, Feature Importance

**Introduction**

The goal of this machine learning report is to predict the survival status of passengers abroad the Titanic using a supervised classification approach. The target variable (**Survived**) is a binary feature representing whether a passenger survived (1) or not (0). The datasets have a diverse of features such as **demographics** (Age, Sex, Name), **socioeconomic** information (Pclass, Fare, Salary), and **travel-related details** (Embarked, Ticket, Cabin)

**Data Mining Theory**

**Supervised Learning Overview**

Predictive data mining typically uses supervised learning algorithms, where models learn patterns from labelled training data and then apply those patterns to predict outcomes on new data (Sarker, 2021). In classification tasks, the output is categorical—like whether someone survived (1) or did not survive (0). By contrast, regression tasks predict continuous values. For the Titanic dataset, because the Survived variable is binary, we use a classification strategy (Han et al., 2022).

**Logistic Regression**

Logistic Regression models the log-odds of a binary target as a linear function of input predictors (Acito, 2023). Instead of predicting a raw numeric output like linear regression does, it applies the logistic (sigmoid) function so that predictions stay between 0 and 1—interpreted as probabilities. Model parameters are generally estimated via maximum likelihood estimation, looking for coefficients that best capture relationships between predictors and the likelihood of survival (Suresh et al., 2022). Logistic regression is appropriate for the Titanic prediction task because:

* **Simplicity and Interpretability**: It’s easy to gauge how each variable influences the odds of survival.
* **Direct Binary Classification**: It natively fits the two-category nature of the Titanic problem (survived vs. not survived), generating probability estimates of belonging to each class.

**Random Forest**

Random Forest is an ensemble learning model that combines multiple Decision Trees to enhance predictive accuracy. Each Decision Tree is fitted on a bootstrapped sample from the training set, often using a random subset of features at each split, which introduces diversity among trees. A final prediction emerges by aggregating the outputs (e.g., majority vote) (Hatwell et al., 2020). It’s well-suited to the Titanic data because:

* **Ability to Capture Non-linear Relationships**: It naturally accommodates more complex interactions among features.
* **Robustness to Overfitting**: An ensemble of diversified trees is less prone to fitting noise than a single tree.
* **High Accuracy Potential**: Typically, Random Forests outperform simpler models, serving as a strong baseline in classification tasks (Hatwell et al., 2020).

**Performance Assessment**

Several evaluation metrics and validation techniques were used to gauge how effectively each model predicts survival:

* **Accuracy:** The fraction of correct predictions out of all predictions, providing a quick initial measure of model performance

Accuracy alone may be insufficient if class imbalance exists or if different misclassification errors carry different costs. Therefore, more evaluation metrics and techniques were used (Kumar, 2021):

* **Precision, Recall, and F1-Score**: Offer a more nuanced view, especially in imbalanced scenarios.
* **AUC-ROC**: Measures how well a model can differentiate between classes across different classification thresholds.
* **Cross-Validation**: Model evaluation should also incorporate techniques like cross-validation, Cross-validation splits the training data repeatedly into folds, training on a portion and validating on the remainder to obtain a more reliable estimate of general performance.

\

**Data Exploration and Preparation**

**Overview of the Dataset**

The Titanic data provides comprehensive passenger information, such as:

* **Demographics:** Name, Sex, Age
* **Socio-Economic Data:** Pclass, Fare, Salary
* **Travel Details:** Ticket, Embarked, Cabin
* **Family Structure:** SibSp (siblings/spouses), Parch (parents/children)

The binary target Survived indicates whether a passenger lived (1) or died (0) in the Titanic disaster.

**Initial Data Exploration**

I began by merging the two datasets on **PassengerId** to combine each passenger’s information. Using commands like head(), info(), and shape() gave an early look into the dataset’s dimensions, variable formats, and potential issues (like missing data). Basic statistical summaries and counts of null and unique values helped identify where data cleaning was necessary.

**Data Cleaning and Handling Missing Values**

Certain columns (e.g., **Embarked** and **Fare**) had missing entries. Because **Embarked** and **Fare** each had only three missing rows total, I dropped those rows to retain the rest of the dataset’s integrity. **Age** had more extensive missingness, so rather than removing many rows, I substituted temporary placeholders and later imputed these missing ages based on the mode of Age within groups defined by passenger titles extracted from names. This grouping leverages domain knowledge: certain titles (e.g., “Miss” or “Master”) are associated with typical age ranges. Such context-driven imputation can yield more realistic and consistent age values than generic methods.

**Feature Extraction and Engineering**

Feature engineering played an important role in creating more meaningful features from the raw data:

* **Titles**: I extracted and standardized titles from the **Name** field (e.g., “Mr.,” “Mrs.,” “Miss.”). Uncommon titles were placed into broader groups like “Rare” or “Royal” for practicality. This step enhances the model’s capacity to interpret socio-economic cues embedded in titles.
* **Cabins**: I created simpler categories from the **Cabin** feature (e.g., “HasCabin,” “Deck”) to capture key location information while reducing sparsity.
* **Age Binning**: Grouping continuous **Age** into categories (e.g., “Baby,” “Child,” “Teenager,” “Adult”) made the column more interpretable.
* **Family Features**: Combining **SibSp** and **Parch** into a single **FamilySize** variable, then binning that into categories, helped capture survival patterns related to traveling in groups.

**Encoding Categorical Variables and Normalizing Numeric Features**

* **Categorical Encoding:** Binary encoding was applied to Job due to its high-cardinality, while one-hot encoding was used for nominal categories (e.g., Embarked, Sex, Title) to make them suitable for machine learning models.
* **Log Transformation:** Variables like Fare and Salary were log-transformed to manage skewness and stabilize variance.
* **Clipping and Scaling:** Outliers in Age, SibSp, and Parch were clipped to limit their influence, and MinMax scaling was applied so all numeric features align on a similar scale, mitigating bias towards attributes with larger numeric ranges.

**Feature Selection for Different Models**

After completing feature engineering and transformations, subsets of attributes were chosen based on their suitability to each model’s characteristics.

* **Logistic Regression:** Focused on a more interpretable set of predictors (demographics, essential socio-economic indicators, etc.).
* **Random Forest:** Leveraged a broader range of engineered features (e.g., title encodings, cabin groupings) to exploit the model’s ability to handle higher-dimensional inputs.

**Figure 1**  
Feature Sets for Logistic Regression and Random Forest Models

A computer screen with text and images

Description automatically generated

This methodology guaranteed that each model was given features tailored to its predictive strengths while maintaining a focus on interpretability and relevance.

**Experimental Setup**

The goal of the experimental setup was to ensure that the predictive models selected were both tuned and robustly evaluated. This required careful partitioning of data, selection of appropriate evaluation measures, and the use of methods designed to prevent overfitting:

**Data Partitioning and Train/Test Split**First step was to divide the dataset into training and test portions. By reserving part of the data for testing, ensuring that the model performance can be evaluated on truly unseen data. This setup also helps spot any overfitting that might happen if I trained and assessed the model on the exact same data. In my workflow, I carried out two separate splits:

**Figure 2**  
Train-Test Split for Logistic Regression and Random Forest Models

A computer code with white text

Description automatically generated

Using *test\_size=0.2* means 20% of the data is held back for testing, while *stratify=y* preserves the original class distribution in both the training and test sets. Setting *random\_state=42* ensures that the split is consistent every time the code is run. This allows for reproducible results, making it easier to compare models and validate their performance.

**Performance Measures**

Accuracy was the metric chosen for initial model comparison due to its simplicity and interpretability. I also used cross-validation and additional metrics (precision, recall, F1-score, and AUC-ROC) to get a broader perspective on performance.  
**Figure 3**  
Baseline Accuracy Evaluation for Logistic Regression and Random Forest Models

A screen shot of a computer program

Description automatically generated

**Cross-Validation**

To reduce variance in performance estimates, I utilized **k-fold cross-validation** (with k=5) on the training sets. This procedure ensures each model is trained and validated on different splits of the data, producing more dependable metrics than a single train/validation partition (Kumar, 2021).

**Figure 4**  
*Cross-Validation for Logistic Regression and Random Forest Models*

A screenshot of a computer program

Description automatically generated

**Hyperparameter Optimization**

Both logistic regression and random forest models can be fine-tuned by adjusting various hyperparameters to boost their accuracy and generalization capabilities. I used a grid search paired with cross-validation (i.e., **GridSearchCV**) to methodically search for the best hyperparameters. For logistic regression, I tested different values of the regularization strength C, while for the random forest I explored a range of **n\_estimators**, **max\_depth**, and **min\_samples\_split**. By evaluating each parameter combination via cross-validation, **GridSearchCV** identifies the hyperparameter set that produces the highest average validation accuracy, offering a solid basis for selecting an optimally tuned model.

**Figure 5**  
Hyperparameter Tuning Using GridSearchCV for Logistic Regression and Random Forest

A screenshot of a computer

Description automatically generated

**Preventing Overfitting**

To reduce the risk of overfitting, I used cross-validation as part of the hyperparameter tuning process, preventing the model from tailoring itself too closely to just one split of the data. Logistic regression relies on **regularization** to keep model complexity in check, while random forest naturally spreads out variance by combining diverse trees. Evaluating performance on a completely separate test set—one that was never used in training or hyperparameter selection—adds another layer of assurance that these models will perform well on data they haven’t seen before.

**Figure 6**  
Final Test Accuracy Evaluation for Tuned Logistic Regression and Random Forest Models

A screenshot of a computer

Description automatically generated

**Model Selection**

After determining the best hyperparameters for each algorithm based on cross-validation accuracy, I retrained both models with those optimized settings. Testing these final models on the held-out test set confirmed that the improvements carried over to unseen data. In the end, this approach to hyperparameter tuning and careful validation produced accurate and generalizable predictive models.

**Results and Discussion**

**Presenting and Comparing Model Results**

The goal of this project was to predict which Titanic passengers survived using a supervised classification approach. We focused on two main models—Logistic Regression and Random Forest—to see which method could best handle the binary target variable (Survived: 1 or 0). Below, I summarize the final outcomes of each step, compare their performance in both graphical and tabular formats, and discuss the key insights gained from the models. This includes identifying the most important features, whether any unusual predictions occurred, and determining which algorithm performed best. I also note each model’s advantages and disadvantages.

**Logistic Regression**

* **Baseline:** Started with default settings, achieving an accuracy of about **0.85** on the test set.
* **Cross-Validation:** Around **0.86**, suggesting that the model generalizes reasonably across different folds.
* **Hyperparameter Tuning:** Tweaking the regularization parameter CC didn’t dramatically change performance—final CV accuracy remained at **0.86.**
* **Test Accuracy and AUC:** Held steady at **0.85** accuracy, with an **AUC** **of 0.87.**

**Random Forest**

* **Baseline:** Default settings gave a test accuracy of about **0.82.**
* **Cross-Validation:** Increased to about **0.87**—a noticeable jump compared to the baseline.
* **Hyperparameter Tuning:** Adjusting n\_estimators, max\_depth, and min\_samples\_split improved both cross-validation and test results.
* **Test Accuracy and AUC:** Reached **0.86** accuracy with an **AUC of 0.91**, indicating stronger class separation overall.

**Table 1**  
Model Comparison Table: Baseline Accuracy, CV Accuracy, Test Accuracy, and AUC-ROC for Logistic Regression and Random Forest

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Baseline Accuracy | CV Accuracy | Test Accuracy | AUC-ROC |
| Logistic Reg. | 0.85 | 0.86 | 0.85 | 0.87 |
| Random Forest | 0.82 | 0.87 | 0.86 | 0.91 |

Graphical plots (ROC curves and feature importances) also highlighted how each model performed and which variables mattered most.

**Figure 7**

ROC Curves for Logistic Regression

A screenshot of a computer

Description automatically generated

**Figure 8**

*ROC Curves for Random Forest*

A screenshot of a computer

Description automatically generated

**Figure 9**

*Feature Importance for Logistic Regression*

A screenshot of a computer

Description automatically generated

**Figure 10**

*Feature Importance for Random Forest*

A screenshot of a computer

Description automatically generated

**Key Insights and Feature Importance**

***Logistic Regression:***

* The largest negative coefficient came from being male (Sex\_male), indicating it greatly reduces survival odds.
* Embarked locations (Embarked\_S, Embarked\_Q) and FamilySizeEncoded also played noticeable roles.
* Because Logistic Regression is linear, it’s straightforward to interpret coefficients and see how each predictor shifts the odds of survival (Acito, 2023; Suresh et al., 2022).

***Random Forest:***

* Consistently highlighted Sex\_male and Title\_2, Title\_3 (e.g., “Mr.,” “Mrs.,” “Miss.”) as top predictors, but also found interactions with Salary\_log and FamilySizeEncoded.
* It’s more flexible than Logistic Regression, uncovering more complex relationships. However, it’s less transparent in explaining how it arrives at a prediction (Hatwell et al., 2020).

***Outliers in Predictions***

Both models occasionally misclassified passengers with unusual attribute combinations (e.g., male travelers with first-class fares but large families). The Random Forest handled these trickier cases better, possibly due to its ability to model non-linear patterns.

**Performance Details: Precision, Recall, and F1-Score**

To get a deeper look at how each class was treated (0 = did not survive, 1 = survived), I checked precision, recall, and F1-scores:

**Table 2**

*Performance Metrics for Logistic Regression and Random Forest: Precision, Recall, and F1-Score*

|  |  |  |
| --- | --- | --- |
| Metric | Logistic Reg. | Random Forest |
| Precision (0) | 0.85 | 0.86 |
| Recall (0) | 0.93 | 0.94 |
| F1-Score (0) | 0.89 | 0.90 |
| Precision (1) | 0.86 | 0.88 |
| Recall (1) | 0.73 | 0.73 |
| F1-Score (1) | 0.79 | 0.80 |
| Accuracy | 0.85 | 0.86 |
| Macro Avg F1 | 0.84 | 0.85 |
| Weighted F1 | 0.85 | 0.86 |

Random Forest held a slight edge across most metrics, particularly in AUC-ROC **(0.91 vs. 0.87)**, meaning it differentiated between survivors and non-survivors more consistently at various decision thresholds.

**Table 3**

*Advantages and Disadvantages of Logistic Regression and Random Forest Models*

|  |  |  |
| --- | --- | --- |
| Aspect | Logistic Regression | Random Forest |
| Pros | - Very interpretable  - Quick to train and predict | - Models complex data well  - Typically high accuracy |
| Cons | - Assumes linear relationships  - May miss nuances | - More resource-intensive  - Often described as a “black-box” |

**Which Model is Best and Why?**

Although Logistic Regression is simpler to interpret and quicker to train, Random Forest outperformed it on key metrics, including a test accuracy of 0.86 (compared to 0.85) and an AUC-ROC of 0.91 (compared to 0.87). These results highlight Random Forest's superior ability to model non-linearities and complex interactions, making it the stronger performer for this Titanic dataset. Additionally, its slightly higher recall and precision for the "survived" class underscore its practical advantage in accurately identifying survivors (Han et al., 2022).

**Conclusion and Reflections**

In this project, I aimed to predict Titanic passenger survival through two classification techniques: Logistic Regression and Random Forest. The main findings revealed that Random Forest slightly outperformed Logistic Regression in terms of both test accuracy and AUC-ROC, suggesting it better captures the complex relationships in the dataset. In contrast, Logistic Regression offered easier interpretation and quicker training but sometimes struggled with more intricate feature interactions. These results underscore the value of ensemble methods when data includes non-linearities or important interactions among variables (Hatwell et al., 2020).

Looking back, there are several ways the models could be improved. First, a broader hyperparameter search space—especially for Logistic Regression—might uncover settings that further enhance performance (Suresh et al., 2022). Second, incorporating additional features or engineering more advanced indicators (such as deeper analyses of family relationships or cabin groupings) may help highlight nuances that neither model fully captured. Third, employing more sophisticated ensemble methods like gradient boosting or stacking might yield incremental gains in accuracy, though interpretability might become a challenge (Géron, 2019).

Overall, the choice of algorithm depends on the balance between interpretability and predictive power. Logistic Regression remains appealing for its clarity and simplicity, whereas Random Forest offers better performance at the cost of complexity. Future work could combine both approaches—using a transparent model for initial exploration and a more flexible ensemble for final predictions (Han et al., 2022). In line with best practices, continuing to refine feature engineering and hyperparameter tuning would likely lead to even more robust outcomes.

**References**

**Sarker, I. H. (2021).** Machine learning: Algorithms, real-world applications, and research directions. SN Computer Science, 2(3), 160.

**Han, X., Zheng, H., & Zhou, M. (2022).** Card: Classification and regression diffusion models. Advances in Neural Information Processing Systems, 35, 18100–18115.

**Acito, F. (2023).** Logistic regression. In Predictive analytics with KNIME: Analytics for citizen data scientists (pp. 125–167). Cham, Switzerland: Springer Nature Switzerland.

**Suresh, K., Severn, C., & Ghosh, D. (2022).** Survival prediction models: An introduction to discrete-time modeling. BMC Medical Research Methodology, 22(1), 207.

**Hatwell, J., Gaber, M. M., & Azad, R. M. A. (2020).** CHIRPS: Explaining random forest classification. Artificial Intelligence Review, 53, 5747–5788.

**Kumar, S. (2021, July 5).** Metrics to evaluate your classification model to take the right decisions. Analytics Vidhya. <https://www.analyticsvidhya.com/blog/2021/07/metrics-to-evaluate-your-classification-model-to-take-the-right-decisions/>

**Appendix A**

**Code Block Images**

**Figure 1**  
Feature Sets for Logistic Regression and Random Forest Models

A computer screen with text and images

Description automatically generated

**Figure 2**  
Train-Test Split for Logistic Regression and Random Forest Models

A computer code with white text

Description automatically generated

**Figure 3**  
Baseline Accuracy Evaluation for Logistic Regression and Random Forest Models

A screen shot of a computer program

Description automatically generated

**Figure 4**  
*Cross-Validation for Logistic Regression and Random Forest Models*

A screenshot of a computer program

Description automatically generated

**Figure 5**  
Hyperparameter Tuning Using GridSearchCV for Logistic Regression and Random Forest

A screenshot of a computer

Description automatically generated

**Figure 6**  
Final Test Accuracy Evaluation for Tuned Logistic Regression and Random Forest Models

A screenshot of a computer

Description automatically generated

**Appendix B**

**Tables and Graphs**

**Table 1**  
Model Comparison Table: Baseline Accuracy, CV Accuracy, Test Accuracy, and AUC-ROC for Logistic Regression and Random Forest

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Baseline Accuracy | CV Accuracy | Test Accuracy | AUC-ROC |
| Logistic Reg. | 0.85 | 0.86 | 0.85 | 0.87 |
| Random Forest | 0.82 | 0.87 | 0.86 | 0.91 |

**Figure 7**

ROC Curves for Logistic Regression

A screenshot of a computer

Description automatically generated

**Figure 8**

*ROC Curves for Random Forest*

A screenshot of a computer

Description automatically generated

**Figure 9**

*Feature Importance for Logistic Regression*

A screenshot of a computer

Description automatically generated

**Figure 10**

*Feature Importance for Random Forest*

A screenshot of a computer

Description automatically generated

**Table 2**

*Performance Metrics for Logistic Regression and Random Forest: Precision, Recall, and F1-Score*

|  |  |  |
| --- | --- | --- |
| Metric | Logistic Reg. | Random Forest |
| Precision (0) | 0.85 | 0.86 |
| Recall (0) | 0.93 | 0.94 |
| F1-Score (0) | 0.89 | 0.90 |
| Precision (1) | 0.86 | 0.88 |
| Recall (1) | 0.73 | 0.73 |
| F1-Score (1) | 0.79 | 0.80 |
| Accuracy | 0.85 | 0.86 |
| Macro Avg F1 | 0.84 | 0.85 |
| Weighted F1 | 0.85 | 0.86 |

**Table 3**

*Advantages and Disadvantages of Logistic Regression and Random Forest Models*

|  |  |  |
| --- | --- | --- |
| Aspect | Logistic Regression | Random Forest |
| Pros | - Very interpretable  - Quick to train and predict | - Models complex data well  - Typically high accuracy |
| Cons | - Assumes linear relationships  - May miss nuances | - More resource-intensive  - Often described as a “black-box” |

**Appendix C**

**Titanic Dataset**

titanic\_ticket\_data.csv consists of the following variables:  
PassengerId: the identifier  
Survived: the value to predict  
Ticket: the Ticket Number  
Fare: the passenger fare  
Cabin: Cabin number  
Embarked: Port of embarkation. C = Cherbourg, Q = Queenstown, S = Southampton  
titanic\_personal\_data.csv consists of the following variables:  
PassengerId – the identifier  
Name: the name of the passenger  
Sex: male or female  
Age: Age is fractional if less than 1. If the age is estimated, is it in the form of xx.5  
SibSp: number of siblings/spouses where family relations are defined as follows:  
Sibling = brother, sister, stepbrother, stepsister  
Spouse = husband, wife  
Parch: number of parent/children where family relations are defined as follows:  
Parent = mother, father;  
Child = daughter, son, stepdaughter, stepson.  
Some children travelled only with a nanny, therefore parch=0 for them  
Salary: in dollars  
Job: job title