Titanic Survival Predictor Project Report

1. Introduction

The Titanic Survival Predictor is a classic machine learning classification task where the goal is to predict whether a passenger survived the Titanic disaster based on their personal details and ticket information.

- Problem Type: Binary Classification (Survived = 1, Not Survived = 0)
- Dataset: Kaggle Titanic dataset, containing features such as age, sex, passenger class, fare, and more.

2. Dataset Description

The dataset includes two files:

- train.csv: Contains features and the target label Survived.
- test.csv: Contains only features (used for final prediction).

Key features used:

- Pclass: Passenger class (1 = 1st, 2 = 2nd, 3 = 3rd)
- Sex: Gender (male or female)
- Age: Age of passenger
- SibSp: Number of siblings/spouses aboard
- Parch: Number of parents/children aboard
- Fare: Ticket fare
- Embarked: Port of embarkation (C = Cherbourg, Q = Queenstown, S = Southampton)

3. Data Preprocessing

Machine learning models require clean, numeric data without missing values. The preprocessing steps included:

- Handling Missing Values:
  - Filled missing Age values with the median age.
  - Filled missing Fare values (in test set) with median fare.
  - Filled missing Embarked values with the mode (most common port).

- Encoding Categorical Variables:
  - Converted Sex to numeric (male = 0, female = 1).
  - One-hot encoded Embarked to numeric columns.
- Dropping Irrelevant Columns:
  - Dropped columns unlikely to improve prediction: PassengerId, Name, Ticket, Cabin.
- Splitting Dataset:
  - Split the training data into training and validation sets (80%/20%).
- Scaling Features:
  - Used StandardScaler to standardize feature values for better Logistic Regression performance.

4. Model Building

Logistic Regression

- A simple, interpretable linear model used as a baseline.
- Trained on scaled features.
- Evaluated using accuracy, precision, recall, F1-score, and confusion matrix.

Handling Missing Data During Model Building

- Missing values were filled after splitting the data to avoid data leakage.
- Ensured no NaN values existed before training.

5. Evaluation

- The model's performance was evaluated on the validation set.
- Metrics included accuracy and detailed classification reports.
- Confusion matrix visualized the true vs predicted classifications.

6. Challenges and Solutions

- Missing Data: Caused errors during model training.
  Solution: Carefully filled missing values after splitting data and before scaling.
- Feature Encoding: Needed to convert categorical data to numeric for model input.
- Data Alignment: Ensured train and test datasets had consistent columns after encoding.

7. Bonus & Next Steps

- Tried a Random Forest Classifier to improve performance using a non-linear model.
- Possible future improvements:

- Feature engineering (e.g., family size, titles extracted from names).
- Hyperparameter tuning (GridSearchCV).
- Advanced models like Gradient Boosting or XGBoost.
- Cross-validation for more robust evaluation.

8. Conclusion

This project demonstrates the full pipeline of a machine learning classification task:

- From data exploration and preprocessing
- To model training and evaluation
- Handling real-world issues like missing data and categorical encoding.

It is a foundational exercise that builds the skills necessary for tackling more complex machine learning problems.