Evaluating the Performance of Logistic Regression, SVM, Naive Bayes, and Random Forest on Kaggle's "Titanic - Machine Learning from Disaster" Dataset

Victor Jing
UBC - CPEN 355
victorjing03@gmail.com

Abstract

This project investigates the effectiveness of various classification algorithms taught in CPEN 355 on Kaggle's "Titanic: Machine Learning from Disaster" dataset. Using standard preprocessing techniques including imputation, encoding, and scaling, we trained and evaluated Logistic Regression, Support Vector Machine, Random Forest, and Naive Bayes models on a train-validation split and performed 5-fold cross-validation to assess robustness. Each model's performance was measured using accuracy and classification metrics to identify the most reliable approach for predicting passenger survival. Each model's predictions were also submitted to Kaggle's ongoing competition in order to evaluate their performance on the test set. The results highlight the comparative strengths and limitations of each classifier under constrained data conditions and inform best practices for model selection on these types of datasets.

1. Introduction

The Titanic dataset, made available through Kaggle's "Titanic: Machine Learning from Disaster" competition, is a widely-used benchmark in the field of supervised machine learning. It provides structured data on the passengers aboard the RMS Titanic and challenges researchers to develop predictive models for determining which passengers survived the disaster.

Predicting the survival of passengers aboard the RMS Titanic is a well-known classification task that has both practical and educational value in machine learning. In this project, we evaluate and compare the performance of four widely-used machine learning classification algorithms (Logistic Regression, Support Vector Machine (SVM), Random Forest, and Naive Bayes) on this dataset. The goal is to identify the model that best predicts passenger survival based on a subset of features such as age, sex, fare, and class. This machine learning challenge is an excellent gateway into the broader study of supervised learning techniques, and is the motivation behind this project.

This project includes data preprocessing, model training and evaluation, and a discussion on performance using both validation accuracy and 5-fold cross-validation. The results are also used to generate a final prediction file suitable for submission to Kaggle's website for verification against the test set.

2. Background

Classification tasks are central to supervised machine learning and involve predicting categorical outcomes from labeled data. The Titanic dataset is a historical passenger manifest compiled by Kaggle and includes two separate files: a training set of 891 entries and a test set of 418 entries. Each entry corresponds to a unique passenger and contains features such as Pclass (ticket class), Sex, Age, SibSp (number of siblings or spouses aboard), Parch (number of parents or children aboard), Fare, and Embarked (port of embarkation). The target variable in the training set is Survived, which is 1 if the passenger survived and 0 otherwise. The goal is to use the available features to build a model that generalizes well in predicting whether a passenger survived the disaster.

The most basic classifiers, such as Logistic Regression, are frequently used for this task due to their simplicity and interpretability. The model predicts the log-odds of an outcome (e.g., survival) as a linear function of the input features which is then transformed into a probability using the sigmoid function:

$$S(x) = \frac{1}{1 + e^{-x}}$$

However, logistic regression assumes a linear relationship between the input features and the log-odds of the outcome. If the actual decision boundary is nonlinear or if the data is not linearly separable, the model may perform poorly.

Support Vector Machines are supervised learning algorithms that aim to find an optimal separating hyperplane that maximizes the margin between different classes. In a linear setting, SVMs create a hyperplane that best divides the feature space into distinct regions, each corresponding to one class. Even for linearly inseparable data, SVMs can still perform well by utilizing non-linear kernels, such as the radial basis function (RBF). This flexibility allows SVMs to model intricate relationships between features, making them useful in complex classification tasks. However, SVMs can be computationally expensive and may struggle with noisy data since they focus on finding the optimal margin, which can be influenced by outliers or irrelevant features that can be especially common in real data like the Titanic set.

Random Forest builds multiple decision trees, where each tree is trained on a random subset of the data and a random subset of features. The final prediction is made by aggregating the predictions from all individual trees, typically through majority voting. Known for its high accuracy, robustness, and resistance to overfitting, RF models do well when handling noisy data and irrelevant features. However, since part of the data preprocessing in this project focuses on removing noise and handling missing values already, these advantages may be less impactful.

Naive Bayes, based on Bayes' theorem, assumes conditional independence among the input features, meaning that each feature contributes independently to the prediction. This assumption simplifies probability calculation, making Naive Bayes particularly computationally efficient. However, in real-world data, features are often correlated, and this strong assumption of independence rarely holds true. As a result, Naive Bayes tends to perform poorly when features are dependent. The oversimplification inherent in the model can thus limit its effectiveness, especially when dealing with complex datasets like the Titanic dataset.

Each of these models offers a unique approach to classification, making it possible to meaningfully compare their performance on our dataset of choice. The models studied in this project were chosen for their relevance to the course material, as well as their practical relevance to classification problems in general.

3. Methodology

3.1 Environment & Libraries

The analysis was conducted in Google Colaboratory, which provides free GPU and CPU access, integration with Google Drive, and support for a wide range of Python libraries. This made it preferable for development and experimentation.

The primary libraries used in the project include pandas for data loading and manipulation, numpy for numerical operations, and scikit-learn for implementing machine learning models and preprocessing tools. For visualizations, matplotlib and seaborn were employed to explore feature distributions and evaluate model performance.

3.2 Model Implementation

To compare performance across multiple classification algorithms, four models were trained using scikit-learn's implementations: `LogisticRegression`, `SVC`, `RandomForestClassifier`, and `GaussianNB`. Each model was trained on scaled training data and evaluated using `accuracy_score` and `classification_report`, cross-validation, and the test set via submission to Kaggle.

The Logistic Regression model was implemented with `max_iter=1000` to ensure convergence during training. All other parameters were left at their default values.

The SVM model was configured with `kernel='rbf'`, which enables the model to capture non-linear relationships in the feature space using a radial basis function. The `probability=True` flag was also enabled to allow probability estimates for evaluation. All other parameters were left at their default values.

The Random Forest model was trained with 100 decision trees by setting `n_estimators=100` and a fixed random seed (`random_state=42`) for reproducibility. Other parameters such as maximum depth and minimum samples per split were left at their default values, letting each tree grow based on data complexity.

The Gaussian Naive Bayes model was chosen as a baseline model due to its simplicity, light computational load, and ease of configuration. However, no effort was made to verify whether the input features followed a Gaussian distribution, so its performance was not expected to be optimal. All parameters were left at their default values.

3.3 Data Preprocessing

The Titanic dataset contains a variety of features, some of which are not immediately suitable for classification models due to missing values, irrelevant information, or unsuitable formats. Several preprocessing steps were applied to address these issues before training the models.

First, non-informative features were dropped. Features like Name and Ticket have no standardized meaning, containing inconsistent formatting, no patterns, and sometimes even duplicates. Others, like PassengerID, are arbitrary and only exist to better organize the dataset for human eyes.

Next, missing data in the dataset had to be addressed. Upon inspection, several features were found to have missing values: Fare and Embarked contained a few missing entries, while Age and Cabin had large portions of missing data. Numerical features like Age and Fare could be reasonably imputed using scikit-learn's `SimpleImputer` with the strategy set to `median`. For the Embarked feature, the missing values could also be reasonably handled using `SimpleImputer` with the strategy set to `most_frequent` (i.e., the mode). However, this type of imputation simply does not make sense with the Cabin feature. Although patterns could potentially exist within the full Cabin data, such as proximity to lifeboats or decks, the high volume of missing entries makes it difficult to model this feature reliably. Ultimately, Cabin also had to be dropped from the set.

The remaining features in the final dataset were all significant indicators of survival in some capacity. For instance, it is well known that passengers who embarked at Cherbourg were more likely to be in first class compared to those from ports like Southampton, and thus had a higher chance of survival. As a result, features such as Pclass and Fare, which are indicative of social class, are statistically significant predictors of survival. The inclusion of some features are apparent—due to the "women and children first" policy, Age and Sex are the most obvious indicators of survival chances. While the significance of SibSp and Parch may not be immediately clear, these can potentially speak to the human aspect of the tragedy. People with family aboard may have stayed together, clustered in the same area, or received more attention from staff or other passengers and staff which led to better odds of surviving. While these features may not be as important as Age or Sex, they are still potential indicators of survival and thus were kept.

Categorical variables were then encoded for compatibility with scikit-learn models. The Sex column was label-encoded, converting it from an entry containing "male" or "female" into a binary numerical format. The Embarked column, which contains multiple categories, was one-hot encoded to avoid implying any ordinal relationship between the categories. Of particular note is that drop_first=True is used for the Embarked feature to avoid multicollinearity, since if a passenger didn't embark at Cherbourg or Southampton, then they must have embarked at Queenstown.

Finally, the dataset was prepared for splitting and scaling. The chosen split is a standard 80% training to 20% validation split (which will later give us 5 iterations for cross-validation). Scaling is also applied to ensure consistency, which is particularly important for models like SVM that are sensitive to feature scales. Columns like Age, Fare, and the one-hot encoded Embarked vectors were standardized to have zero mean and unit variance using StandardScaler.

A correlation heatmap generated using seaborn's `heatmap` summarizes the final data set.

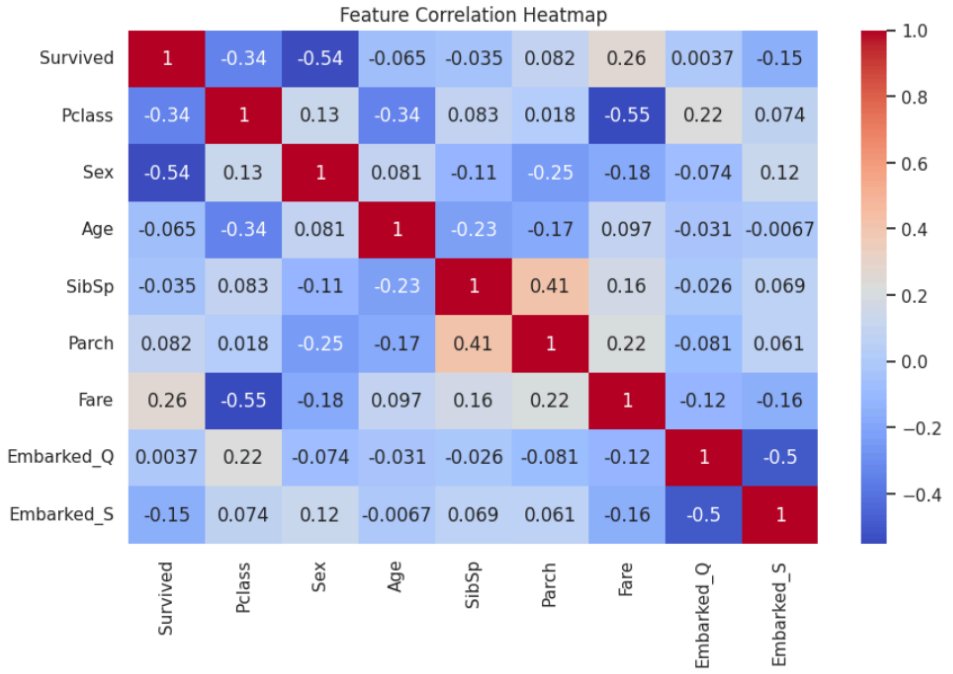


Figure 1. Feature correlation heatmap of dataset after preprocessing.

Most notably, the Sex feature has a strong negative correlation (-0.54) with Survival, indicating that females were more likely to survive. Similarly, lower Pclass values (i.e., higher-class passengers) were associated with higher survival rates, as shown by a correlation of -0.34.

4. Results

All four models were first evaluated on the held-out validation set after training on scaled training data. Performance was primarily measured using classification accuracy, precision, recall, and F1-score. To further assess generalizability, each model underwent 5-fold cross-validation, and the trained models were also submitted to the Kaggle competition for final testing on the hidden test set.

The accuracy score is the proportion of correct predictions, both true positives (TP) and true negatives (TN), out of all predictions. The precision score is interpreted as how often a model is correct when they identify that a data point belongs to a certain class. It is given by the formula below, where FP stands for false positive.

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}}$$

The recall, or sensitivity, score is how many of the actual instances of a particular class were correctly predicted. It is given by the formula below, where FN is false negative.

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}}$$

The F1-score is the harmonic mean of precision and recall. It provides a single metric that balances the trade-off between precision and recall. Since the F1-score takes the harmonic mean, large differences between the precision and recall scores of a model are punished severely, which is important when both false positives and false negatives are important.

$$\text{F}_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Finally, the macro-average and weighted-average is the generalization of the above metrics. The weighted-average takes the arithmetic mean of the precision, recall, and F1 scores while taking into account the size of the class support, whereas the macro-average only takes the arithmetic mean. This means that the weighted-average tends to provide a more accurate reflection of overall model performance, especially for binary classification tasks like the one studied in this project.

The raw data referenced subsequent sections can be found in the Appendix.

4.1 Logistic Regression

Logistic Regression achieved a validation accuracy of 0.8101. It demonstrated slightly better performance on predicting non-survivors (class 0), with a precision of 0.83 and recall of 0.86. In contrast, the model showed reduced sensitivity in detecting survivors, with a recall of 0.74. The weighted F1-score was 0.81, suggesting balanced but modest classification capability. The cross-validation accuracy averaged 0.7857 with a standard deviation of 0.0184, indicating consistent performance across splits. On the Kaggle test set, the model achieved a public leaderboard score of 0.76555.

4.2 Support Vector Machine

The SVM with an RBF kernel produced the highest validation accuracy among all models at 0.8212. It displayed the strongest performance in detecting survivors, achieving a precision of 0.84, although recall for this class was slightly lower at 0.70. Non-survivors were predicted with high precision and recall (0.81 and 0.90, respectively), contributing to a weighted F1-score of 0.82. Cross-validation yielded a mean accuracy of 0.8272 with a standard deviation of 0.0296, suggesting strong generalization. The SVM also performed best on the Kaggle test set, scoring 0.77751.

4.3 Random Forest

The Random Forest model matched the logistic regression model in validation accuracy, a score of 0.8101, and closely followed its classification metrics too. It favored non-survivors slightly less than Logistic Regression, achieving a recall of 0.85 for class 0

and 0.76 for class 1. The overall weighted F1-score was 0.81. In 5-fold cross-validation, the model achieved a mean accuracy of 0.8104 with a standard deviation of 0.0259. Interestingly enough it ranked lower than the logistic regression model in the Kaggle submission, having a score of 0.75598.

4.4 Naive Bayes

Naive Bayes achieved the lowest validation accuracy at 0.7709. Although its performance was relatively balanced (precision of 0.81 for class 0 and 0.72 for class 1), its predictive power was generally weaker across all metrics, with a weighted F1-score of 0.77. The model's mean cross-validation accuracy was 0.7779 with a higher standard deviation of 0.0346, suggesting more variability across folds. Its Kaggle submission yielded the lowest score among the models at 0.75358.

5. Discussion

Across all of the evaluation criteria, the Support Vector Machine consistently outperformed the other tested classification models across both validation and cross-validation metrics. Its high accuracy, strong F1-scores, and low variance suggest that it was the most reliable model for tackling Kaggle's Titanic dataset. The SVM's ability to handle complex, non-linear decision boundaries likely helped capture the structure of the Titanic dataset, even with limited feature interactions.

Logistic Regression and Random Forest followed closely behind, with near-identical performance in validation and similar average accuracy during cross-validation. While Random Forest tends to excel in capturing non-linear relationships and reducing overfitting through ensembling, its advantage over Logistic Regression was marginal in this case. This may be due to the relatively small feature set and the lack of deep hierarchical relationships between input variables. Logistic Regression's simplicity and interpretability still make it a valid choice, especially for beginners.

Naive Bayes underperformed relative to the other models, which aligns with expectations given its strong assumption of conditional independence between features. In the Titanic dataset, many features, such as Fare and Pclass, or SibSp and Parch, are correlated (see Figure 1). These violations likely contributed in some way to the lacklustre performance.

Cross-validation reinforced these findings. The SVM had the highest average score and lowest standard deviation, indicating both accuracy and consistency. Random Forest and Logistic Regression were close behind, while Naive Bayes exhibited the greatest variance and the lowest overall performance.

A key limitation across all models was that missing values were handled with imputation strategies, any errors introduced at this stage may have influenced the learned patterns, especially for models sensitive to input distributions.

6. Conclusion

This study evaluated four classification algorithms—Logistic Regression, Support Vector Machine (SVM), Random Forest, and Naive Bayes—on the Titanic survival prediction task. Among them, the SVM achieved the best overall performance across all evaluation settings, offering a strong balance between precision and recall, as well as the most consistent results during cross-validation and test-set verification. Logistic Regression and Random Forest also performed well, with nearly indistinguishable validation metrics. Naive Bayes lagged behind due to its assumptions about feature independence, which were not met in this dataset.

These findings reinforce the importance of matching model complexity and assumptions to the structure of the data. SVM's flexibility proved beneficial in capturing patterns that simpler models like Naive Bayes could not. However, the strong showing of Logistic Regression also highlights that simpler models can still perform competitively..

In practice, these insights could inform how classification models are selected for similar problems involving binary outcomes and tabular data. The interpretability of Logistic Regression might outweigh the marginal accuracy gains of SVM in some applications, whereas ensemble models like Random Forest could be more useful in noisier or higher-dimensional settings.

Future work could involve applying more advanced models, such as gradient boosting or neural networks, and implementing strategies to mitigate class imbalance. Additional improvements could be achieved through hyperparameter optimization, feature engineering, and exploring alternative preprocessing techniques. Testing these models across other passenger or demographic datasets would also help assess their generalizability beyond the Titanic scenario.

7. Declaration of Generative AI Usage

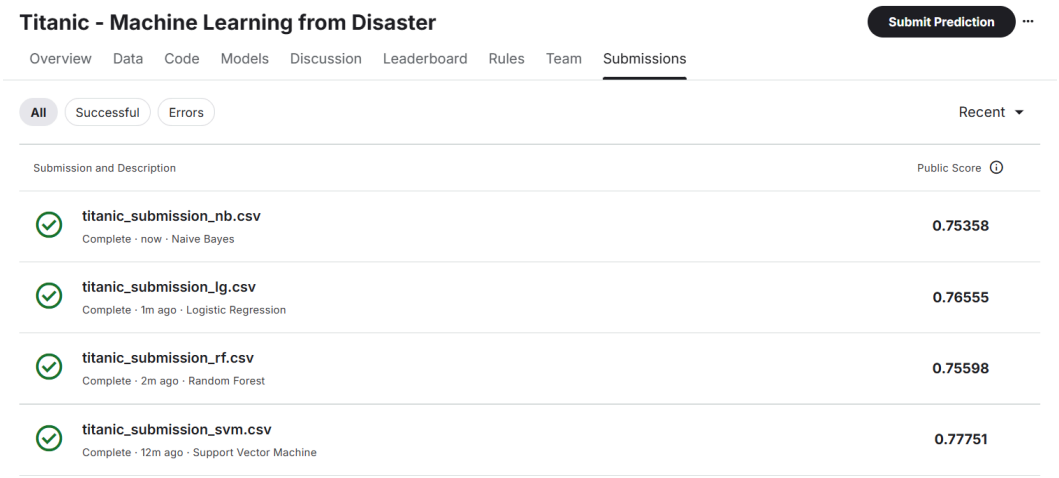
Text generation tools like ChatGPT were used to redraft and rephrase various sections of this report to help with organization and ensure ideas flowed smoothly in writing. The conclusions and analysis of the results produced by the experiments were done independently and with reference to reputable sources and common knowledge, not generated by AI.

ChatGPT was used moderately in the generation of the Python code in Google Colaboratory, mostly in realizing ideas and transforming pseudocode to actual code. ChatGPT also helped generate code used to create various evaluation metrics and graphics referenced and used in this report. However, this use of ChatGPT should not have retracted from the overall learning experience this assignment was meant to provide, as its purpose was to overcome an unfamiliarity with Python.

8. Appendices

Please see the Python file attached with the submission of this report. A more complete repository can also be found at github.com/victoryyjing/cpen355finalproject.

Appendix A. Kaggle Submissions



Appendix B. Accuracy Score & Classification Report

```
 Logistic Regression Accuracy: 0.8101
Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.86      0.84       105
           1       0.79      0.74      0.76        74

    accuracy                           0.81       179
   macro avg       0.81      0.80      0.80       179
weighted avg       0.81      0.81      0.81       179


 Random Forest Accuracy: 0.8101
Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.85      0.84       105
           1       0.78      0.76      0.77        74

    accuracy                           0.81       179
   macro avg       0.80      0.80      0.80       179
weighted avg       0.81      0.81      0.81       179


 SVM Accuracy: 0.8212
Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.90      0.86       105
           1       0.84      0.70      0.76        74

    accuracy                           0.82       179
   macro avg       0.83      0.80      0.81       179
weighted avg       0.82      0.82      0.82       179


 Naive Bayes Accuracy: 0.7709
Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.80      0.80       105
           1       0.72      0.73      0.72        74

    accuracy                           0.77       179
   macro avg       0.76      0.76      0.76       179
weighted avg       0.77      0.77      0.77       179
```