SRI LANKA INSTITUTE OF INFORMATION TECHNNOLOGY

MSc in Information Technology

Titanic Survival Prediction Using Machine Learning Algorithms

Module: Machine Learning – IT6080 (2024)

Supervisor: Dr. Dharshana Kasthurirathna

MS23466876 - Wijegunasinghe Y.S.D

Table of Contents

1. Introduction

The RMS Titanic sinking in 1912 is still regarded as one of the most terrible and talked-about maritime catastrophes in history. During its inaugural trip, the ship struck an iceberg and perished in the North Atlantic Ocean, killing over 1,500 of its 2,200 passengers and crew. For more than a century, the public has been interested in this tragedy due to the societal dynamics at play throughout the rescue operations as well as the human cost. Class, gender, and age all had a substantial impact on survival chances; first-class passengers, women, and children had a higher chance than others.

Because of the complexity and unpredictability of the circumstances involved, predicting which passengers survived the catastrophe is an intriguing topic for data science and machine learning. Machine learning models can be trained to forecast survival outcomes by examining passenger data, including age, sex, class, and fare. In addition to being helpful for historical research, these forecasts provide a real-world illustration of how machine learning may be applied to datasets where a variety of factors affect the result.

The objective of this research is to use two supervised learning algorithms—Random Forest and Logistic Regression—to determine which passengers survived the Titanic accident. We seek to comprehend how various machine learning algorithms can be used to binary classification tasks such as these by assessing these models' performance using accuracy and other measures. The dataset utilized for this study, which includes demographic and ticket statistics for Titanic passengers, is openly accessible on Kaggle.

2. Dataset Description

The Titanic passenger dataset, which is openly accessible on Kaggle, served as the basis for this project. It provides thorough information about the passengers on board the RMS Titanic during its tragic maiden voyage in 1912. The dataset is frequently used in machine learning applications to forecast a passenger's survival based on a variety of characteristics.

The following variables make up the dataset:

- **PassengerId**: A unique identifier assigned to each passenger.
- **Pclass**: Ticket class (1 = First class, 2 = Second class, 3 = Third class). This variable is a proxy for socioeconomic status.
- **Name**: The name of the passenger.
- **Sex**: Gender of the passenger (male or female).
- **Age**: The passenger's age in years.
- **SibSp**: Number of spouses or siblings that were on board the Titanic.
- **Parch**: How many parents or kids were on board the Titanic?
- **Ticket**: The passenger's ticket number.
- **Fare**: The ticket was paid for by the fare.
- **Cabin**: The cabin number.
- **Embarked**: The port of embarkation (C = Cherbourg, Q = Queenstown, S = Southampton).

The target variable for this dataset is:

- **Survived**: If the traveler lived (1 = Yes, 0 = No).

Based on the other features, which record details about each passenger's demographics, ticket class, family structure, and other details, the goal is to forecast the value of the Survived column. We must use a variety of input features to predict a binary outcome (survival) in this dataset's classification challenge. This is a great example of data preprocessing and machine learning model construction because it has missing values and categorical data.

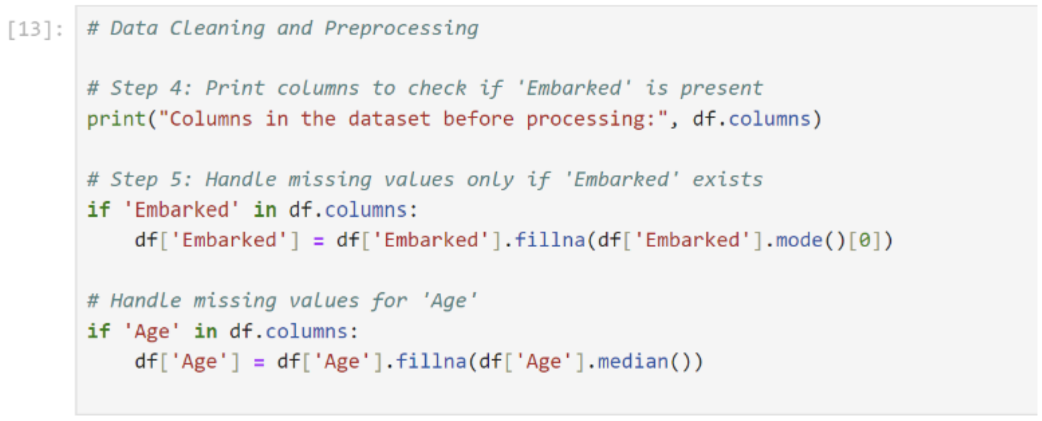
3. Data Preprocessing

An essential part of any machine learning effort is data preprocessing. Some attributes in the Titanic dataset are categorical and need to be converted into numerical format, while some columns have missing values. Furthermore, unnecessary columns are removed because not all features are pertinent to the prediction task. The main procedures for cleaning and getting the data ready for model training are listed below:

3.1 Handling Missing Values

The Age and Embarked columns of the dataset include details that are missing. Since many machine learning algorithms do not tolerate partial data, missing values must be handled.

- **Age**: The passengers' median age is used to fill in the gaps in the Age column. This strategy was used to prevent the age distribution from being skewed.
- **Embarked**: A few data points are missing from the Embarked column, which represents the port of embarkation. The column's mode, or most frequent value, is used to fill up these missing values.

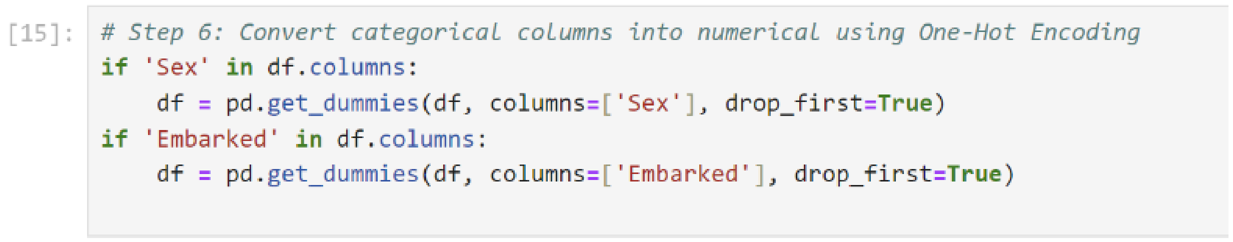


```
[13]: # Data Cleaning and Preprocessing

      # Step 4: Print columns to check if 'Embarked' is present
      print("Columns in the dataset before processing:", df.columns)

      # Step 5: Handle missing values only if 'Embarked' exists
      if 'Embarked' in df.columns:
          df['Embarked'] = df['Embarked'].fillna(df['Embarked'].mode()[0])

      # Handle missing values for 'Age'
      if 'Age' in df.columns:
          df['Age'] = df['Age'].fillna(df['Age'].median())
```

3.2 Converting Categorical Variables

The dataset contains several categorical characteristics, including Embarked and Sex. These category variables must be transformed into numerical representations since machine learning algorithms need numerical inputs. One-Hot Encoding is used for this, generating new binary columns for every category.

- **Sex**: Transformed into two columns that are binary, with 0 denoting female and 1 denoting male.
- **Embarked**: Transformed into three binary columns that stand for Southampton, Queenstown, and Cherbourg, the three embarkation ports. To prevent multicollinearity, one of these columns is removed.

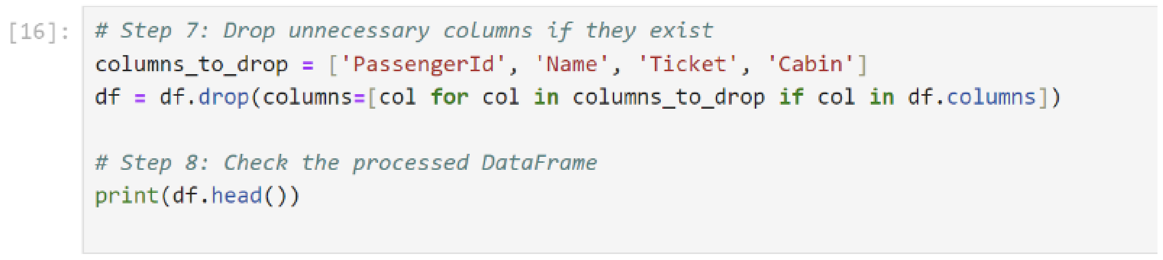


```
[15]: # Step 6: Convert categorical columns into numerical using One-Hot Encoding
      if 'Sex' in df.columns:
          df = pd.get_dummies(df, columns=['Sex'], drop_first=True)
      if 'Embarked' in df.columns:
          df = pd.get_dummies(df, columns=['Embarked'], drop_first=True)
```

3.3 Dropping Irrelevant Columns

PassengerId, Name, Ticket, and Cabin are among the dataset’s columns that don’t have any bearing on survival prediction and don’t give the model any useful information. These columns are therefore removed.

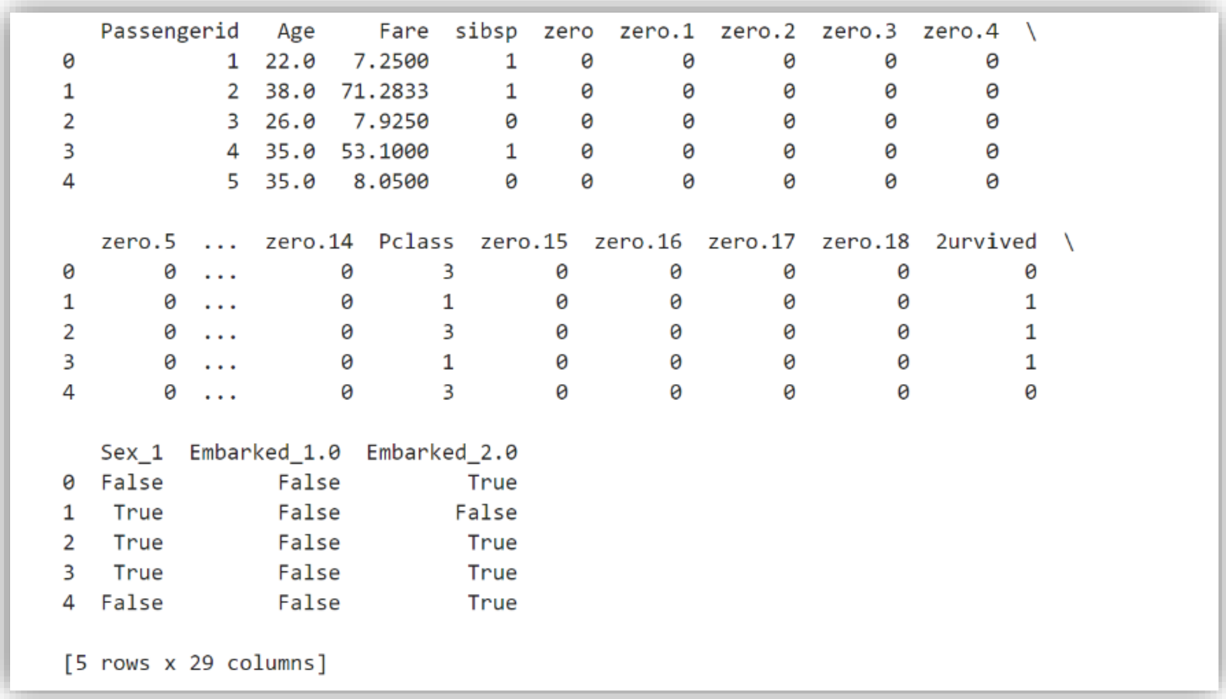
- **PassengerId**: Merely an identifier and has no effect on survival.
- **Name**: Names are not useful for the prediction task.
- **Ticket**: Ticket numbers are unique and don't provide any predictive value.
- **Cabin**: Many missing values, and even when present, the cabin information may not be highly relevant for prediction.



```
[16]: # Step 7: Drop unnecessary columns if they exist
      columns_to_drop = ['PassengerId', 'Name', 'Ticket', 'Cabin']
      df = df.drop(columns=[col for col in columns_to_drop if col in df.columns])

      # Step 8: Check the processed DataFrame
      print(df.head())
```

3.4 Final Preprocessed Data

The dataset is now clean and prepared for usage in the machine learning models following the completion of the procedures. Irrelevant characteristics have been eliminated, missing values have been filled in, and categorical variables have been transformed into numerical format. The features that are most important for forecasting Titanic passengers' survival are included in the final dataset.



```
   Passengerid   Age     Fare  sibsp  zero  zero.1  zero.2  zero.3  zero.4  \
0            1  22.0   7.2500      1     0       0       0       0       0
1            2  38.0  71.2833      1     0       0       0       0       0
2            3  26.0   7.9250      0     0       0       0       0       0
3            4  35.0  53.1000      1     0       0       0       0       0
4            5  35.0   8.0500      0     0       0       0       0       0

   zero.5  ...  zero.14  Pclass  zero.15  zero.16  zero.17  zero.18  2urvived  \
0       0  ...        0       3        0        0        0        0         0
1       0  ...        0       1        0        0        0        0         1
2       0  ...        0       3        0        0        0        0         1
3       0  ...        0       1        0        0        0        0         1
4       0  ...        0       3        0        0        0        0         0

   Sex_1  Embarked_1.0  Embarked_2.0
0  False         False          True
1   True         False         False
2   True         False          True
3   True         False          True
4  False         False          True

[5 rows x 29 columns]
```

4. Methodology

To determine if a passenger survived the Titanic accident, we used two supervised learning algorithms: Random Forest Classifier and Logistic Regression. Both techniques were selected to compare their performance on this dataset since they are often employed for binary classification tasks.

4.1. Algorithms Used

a. Logistic Regression

Logistic Regression is a well-liked and simple method for binary classification issues. It forecasts the probability that a given input will fall into one of two categories using the logistic function. The method performs well on datasets where the relationship between the input qualities and the output is approximately linear. Here, we apply it to forecast if a passenger will survive (1) or not (0).

b. Random Forest Classifier

The Random Forest ensemble learning technique is used to construct several decision trees during training. The average of each tree’s predictions determines the class that emerges. For a range of classification tasks, it often offers acceptable accuracy and is resistant to overfitting. Random Forest performs well on complicated datasets like Titanic and can capture non-linear correlations.

4.2. Steps Involved

The following actions were taken to train and assess the models:

a. **Define X (features) and y (target)**:
Following data preprocessing, the remaining columns are utilized as features, with the target variable being Survived. We choose y as the goal vector and X as the feature matrix.

b. **Split Data into Training and Testing Sets**:
The training and testing sets of the dataset are separated. The model is trained using 80% of the data, and its performance is tested using 20%.

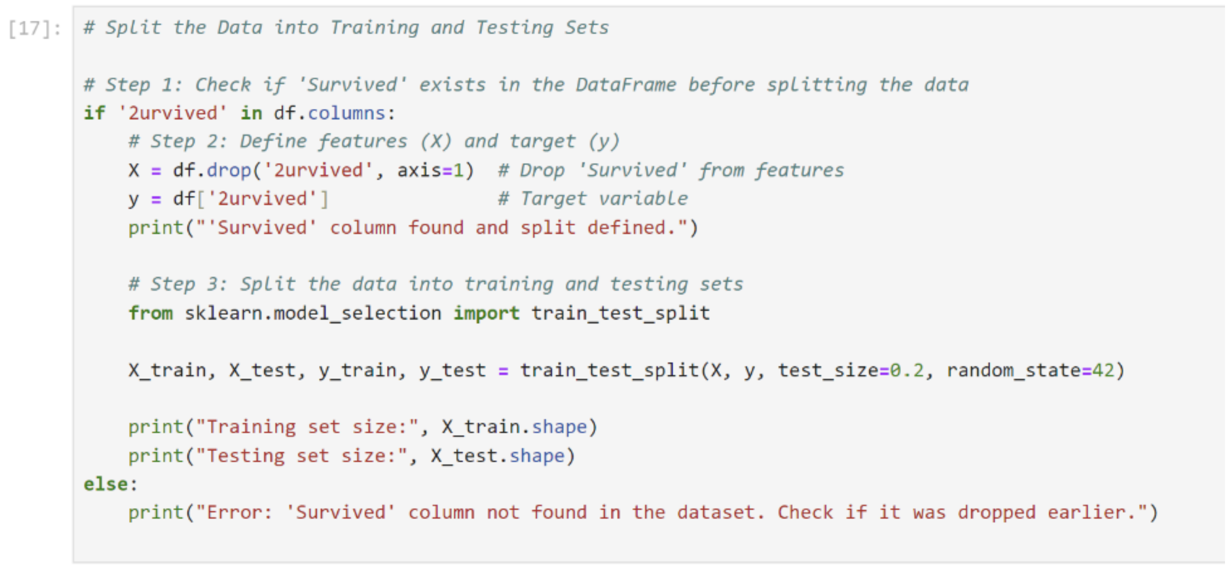
c. **Train the Models**:
We train the Random Forest and Logistic Regression models with the training data. To ascertain the connection between the input features and the target variable, the models need to be fitted to the training data.

d. Make Predictions and Evaluate:

We utilize the test data to make predictions once the models have been trained, and we assess their performance using classification metrics such as accuracy.

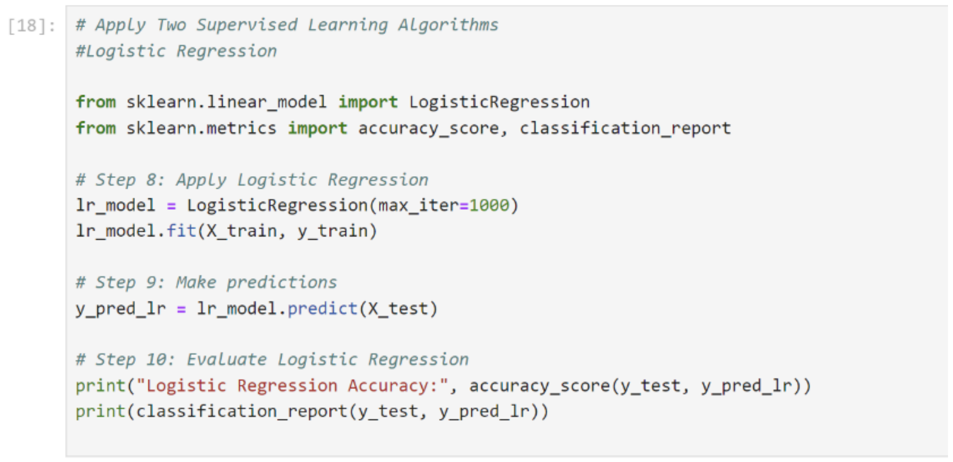
4.3 Code Snippets

The following Python code was used to divide the data, train the models, and generate predictions:



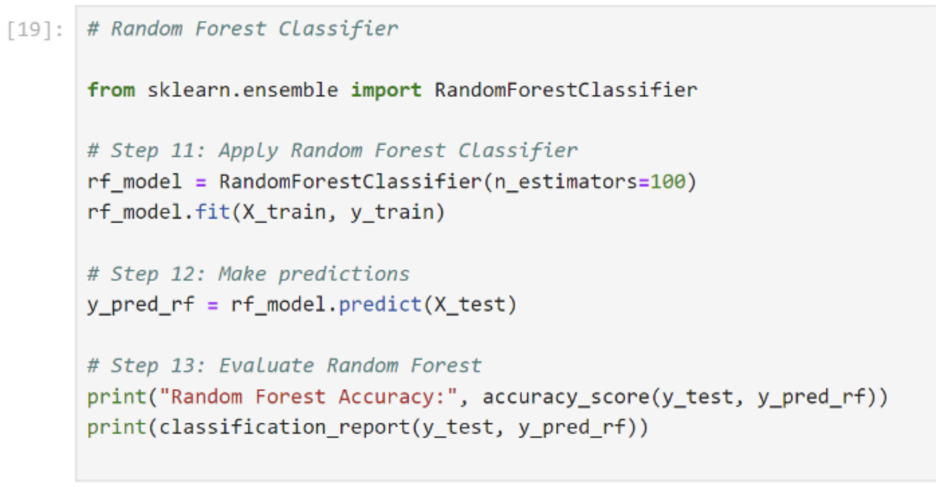
```
[17]: # Split the Data into Training and Testing Sets

      # Step 1: Check if 'Survived' exists in the DataFrame before splitting the data
      if '2urvived' in df.columns:
          # Step 2: Define features (X) and target (y)
          X = df.drop('2urvived', axis=1)  # Drop 'Survived' from features
          y = df['2urvived']               # Target variable
          print("'Survived' column found and split defined.")

          # Step 3: Split the data into training and testing sets
          from sklearn.model_selection import train_test_split

          X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

          print("Training set size:", X_train.shape)
          print("Testing set size:", X_test.shape)
      else:
          print("Error: 'Survived' column not found in the dataset. Check if it was dropped earlier.")
```

Logistic Regression:



```
[18]: # Apply Two Supervised Learning Algorithms
      #Logistic Regression

      from sklearn.linear_model import LogisticRegression
      from sklearn.metrics import accuracy_score, classification_report

      # Step 8: Apply Logistic Regression
      lr_model = LogisticRegression(max_iter=1000)
      lr_model.fit(X_train, y_train)

      # Step 9: Make predictions
      y_pred_lr = lr_model.predict(X_test)

      # Step 10: Evaluate Logistic Regression
      print("Logistic Regression Accuracy:", accuracy_score(y_test, y_pred_lr))
      print(classification_report(y_test, y_pred_lr))
```

Random Forest Classifier:



```
[19]: # Random Forest Classifier

      from sklearn.ensemble import RandomForestClassifier

      # Step 11: Apply Random Forest Classifier
      rf_model = RandomForestClassifier(n_estimators=100)
      rf_model.fit(X_train, y_train)

      # Step 12: Make predictions
      y_pred_rf = rf_model.predict(X_test)

      # Step 13: Evaluate Random Forest
      print("Random Forest Accuracy:", accuracy_score(y_test, y_pred_rf))
      print(classification_report(y_test, y_pred_rf))
```

4.4 Evaluation

After training the model, predictions are made on the test set. We evaluate the models' performance using F1-score, recall, accuracy, and precision. This enables us to evaluate how well Random Forest and Logistic Regression predict passenger survivability.

We can determine whether approach is more appropriate for this kind of classification problem by comparing the performance of these two methods.

5. Results and Evaluation

We employed several evaluation metrics, including accuracy, precision, recall, and F1-score, to evaluate the two supervised learning algorithms' performance. These measurements shed light on how well the models were able to forecast whether Titanic passengers would survive.

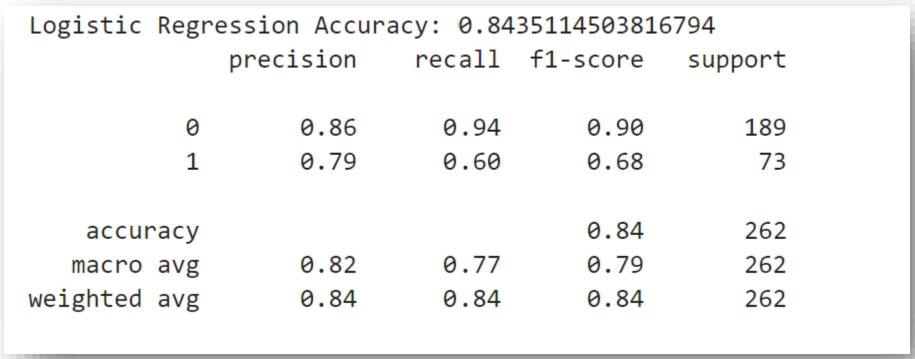
5.1. Evaluation Metrics

- **Accuracy**: The proportion of observations that were accurately predicted to all observations. It gauges the model's overall efficacy, but if the courses are unbalanced, it could be deceptive.
- **Precision**: The proportion of all optimistic forecasts to those that came true. It shows the percentage of anticipated survivors that survived.
- **Recall**: The proportion of true positives to correctly predicted positives. The percentage of actual survivors that the model correctly identified is displayed.
- **F1-Score**: The harmonic methods of recall and accuracy. In situations when recall and precision must be balanced, this statistic may be useful.

5.2.Model Performance

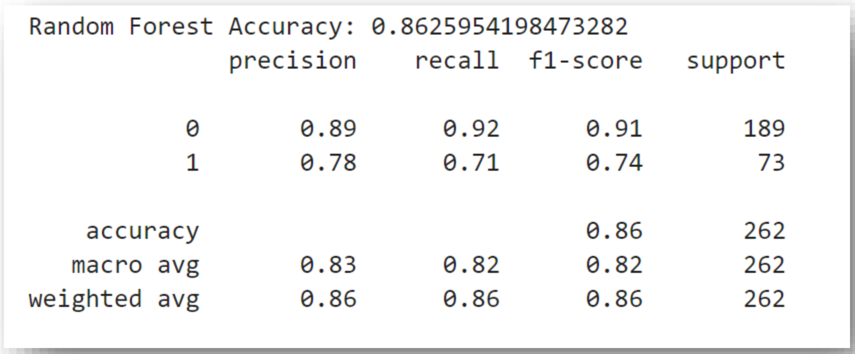
We used the preprocessed Titanic dataset to train and evaluate the Random Forest Classifier and Logistic Regression models. We calculated the evaluation metrics for each model after generating predictions for the test set.

Logistic Regression Performance



```
Logistic Regression Accuracy: 0.8435114503816794
              precision    recall  f1-score   support

           0       0.86      0.94      0.90       189
           1       0.79      0.60      0.68        73

    accuracy                           0.84       262
   macro avg       0.82      0.77      0.79       262
weighted avg       0.84      0.84      0.84       262
```

Random Forest Performance


```
Random Forest Accuracy: 0.8625954198473282
              precision    recall  f1-score   support

           0       0.89      0.92      0.91       189
           1       0.78      0.71      0.74        73

    accuracy                           0.86       262
   macro avg       0.83      0.82      0.82       262
weighted avg       0.86      0.86      0.86       262
```

5.3. Comparison Results

The following table compiles the Random Forest and Logistic Regression classifiers' performance using a range of evaluation metrics:

Metric	**Logistic Regression**	**Random Forest**
Accuracy	84.35%	86.26%
Precision (0)	0.86	0.89
Precision (1)	0.79	0.78
Recall (0)	0.94	0.92
Recall (1)	0.60	0.71
F1-Score (0)	0.90	0.91
F1-Score (1)	0.68	0.74
Macro Avg Precision	0.82	0.83
Macro Avg Recall	0.77	0.82
Macro Avg F1-Score	0.79	0.82
Weighted Avg Precision	0.84	0.86
Weighted Avg Recall	0.84	0.86
Weighted Avg F1-Score	0.84	0.86

In most tests, the Random Forest classifier performs somewhat better than Logistic Regression; however, it significantly improves in class 1 (survived passengers) recall and F1-score. This implies that Random Forest outperforms both algorithms in terms of passenger survival prediction.

5.4. Analysis of Results

Logistic Regression Analysis:

- With an accuracy of 84.35%, logistic regression was able to accurately predict travelers' survival status in almost 84% of the cases.
- With a high recall of 0.94 and precision of 0.86, it does well in classifying non-survivors (class 0); but it has more difficulty with survivors (class 1), whose recall is only 0.60. Because of its inferior memory, a sizable portion of real survivors are missed.
- This difficulty in accurately classifying survivors is reflected in the class 1 overall F1-score of 0.68.

Random Forest Analysis:

- With an accuracy of 86.26%, the Random Forest model fared better than Logistic Regression.
- With a recall of 0.71 as opposed to 0.60 in Logistic Regression, it demonstrated superior performance in predicting survivors (class 1), indicating that it detects more real survivors.
- Class 1's F1-score increased to 0.74, indicating a more equitable trade-off between survivors' recall and precision.
- Furthermore, comparable to logistic regression, the precision and recall for non-survivors (class 0) stayed high at 0.89 and 0.92, respectively.

All things considered, Random Forest outperformed, especially when it came to forecasting the survival of passengers (class 1). Although both models demonstrated high accuracy and good precision/recall for class 0, Random Forest's capacity to train in an ensemble probably helped it identify more intricate patterns in the data, which resulted in better performance. For this Titanic dataset, Random Forest would be the recommended model.

6. Discussion

6.1. Model Performance

Here, we compared two supervised learning methods: Random Forest and Logistic Regression. Both models performed well, according to the data, with the Random Forest classifier surpassing Logistic Regression by a little margin on a number of important parameters.

- Overall, Random Forest performed better than Logistic Regression, with an accuracy of 86.26 percent versus 84.35%. Additionally, it outperformed Logistic Regression (recall of 0.60 and F1-score of 0.68) in predicting survival (class 1), with stronger recall (0.71) and F1-score (0.74). This implies that the more intricate patterns connected to survival were better captured by Random Forest.

- With an accuracy of 84.35%, logistic regression fared well despite being simpler. The lower recall and F1-score for class 1 indicate that it had difficulty identifying survivors, while it demonstrated good precision and recall for predicting non-survivors (class 0).

6.2 Limitations

Both models have their strengths and limitations:

- A more straightforward model that assumes that characteristics and the target variable have a linear relationship is called logistic regression. This presumption might make it more difficult to capture intricate feature interactions, which could account for its poor performance in comparison to Random Forest.

- These interactions were better captured by Random Forest as an ensemble approach. Overfitting is a potential drawback, though, particularly when dealing with tiny datasets. Random Forest's tendency to perform exceptionally well on training data can occasionally lead to overfitting, which may impair the model's capacity to generalize to new data. Separating the data into training and testing sets helped to mitigate this in our case, but more may be done to prevent potential overfitting by using cross-validation techniques or changing hyper parameters.

6.3 Feature Impact on Survival Prediction

A few characteristics in the Titanic dataset significantly influenced survivor prediction, chief among them being:

- **Sex**: The survival rate for women was significantly higher than that of men. The significance of this characteristic in predicting survival can be explained by the "women and children first" approach that was prevalent during the crisis.

- **Pclass (Passenger Class)**: Pclass was an essential component of both models since passengers in higher classes (first-class) had better access to lifeboats and survival supplies.

- **Age**: Age was another significant indicator because younger passengers—especially children—were also given preference. However, during data preprocessing, it was required to handle missing values for Age.

6.4 Future Work

Several suggestions for future research can be taken into consideration to further enhance the models' performance:

- **Feature Engineering:** The prediction capability of the models may be increased by adding new features or combining preexisting ones. For example, a "Family Size" feature might be created by adding the number of parents/children and siblings/spouses on board to get more information.

- **Using More Advanced Algorithms:** To find out if they perform even better, other machine learning techniques like Support Vector Machines (SVM) and Gradient Boosting Machines (GBM) should be investigated.

- **Cross-Validation and Hyperparameter Tuning:** Cross-validation can be used to more accurately evaluate model performance and reduce overfitting, while hyperparameter adjustment (e.g., varying the depth of trees in Random Forest) can assist in determining the best model settings.

- **Handling Imbalanced Classes:** Techniques like oversampling, under sampling, or using various loss functions could help to increase performance because the dataset contains fewer survivors, especially when it comes to detecting survivors (class 1).

- **Incorporating More Data:** Additional helpful elements for survival prediction might be obtained by integrating additional datasets or external data, such as comprehensive ship or weather information.

7. Conclusion

To predict the Titanic passenger survival, we used two supervised learning algorithms: Random Forest and Logistic Regression. With a greater accuracy of 86.26% compared to 84.35%, as well as superior recall and F1-scores for predicting survivors, Random Forest surpassed Logistic Regression, according to our data, proving its ability to capture intricate patterns. Nevertheless, both models had drawbacks: Random Forest may be prone to overfitting, particularly with small datasets, while Logistic Regression's linear structure may have limited its performance. Further fine-tuning via cross-validation and hyperparameter optimization might be necessary to address these problems. In line with historical reports, key characteristics such as sex, Pclass, and age were important in predicting survival since younger people, women, and first-class passengers had greater survival rates. To further improve predictions, future developments might entail investigating sophisticated algorithms like Support Vector Machines or Gradient Boosting Machines, improving feature engineering, and resolving class imbalance. Although additional optimization may result in even higher performance in subsequent applications, Random Forest ultimately proved to be the most successful model for this dataset.

8. Appendix

```
.. code:: ipython3

    # Load the dataset
    # Step 1: Import necessary libraries
    import pandas as pd

    # Step 2: Load the dataset
    df = pd.read_csv('train.csv')

    # Step 3: Check the data structure
    print(df.head())
    print(df.info())



.. code:: ipython3

    # Data Cleaning and Preprocessing

    # Step 4: Print columns to check if 'Embarked' is present
    print("Columns in the dataset before processing:", df.columns)

    # Step 5: Handle missing values only if 'Embarked' exists
    if 'Embarked' in df.columns:
        df['Embarked'] = df['Embarked'].fillna(df['Embarked'].mode()[0])

    # Handle missing values for 'Age'
    if 'Age' in df.columns:
        df['Age'] = df['Age'].fillna(df['Age'].median())


.. code:: ipython3

    # Step 6: Convert categorical columns into numerical using One-Hot Encoding
    if 'Sex' in df.columns:
        df = pd.get_dummies(df, columns=['Sex'], drop_first=True)
    if 'Embarked' in df.columns:
        df = pd.get_dummies(df, columns=['Embarked'], drop_first=True)


.. code:: ipython3

    # Step 7: Drop unnecessary columns if they exist
    columns_to_drop = ['PassengerId', 'Name', 'Ticket', 'Cabin']
    df = df.drop(columns=[col for col in columns_to_drop if col in df.columns])

    # Step 8: Check the processed DataFrame
    print(df.head())
```

```
.. code:: ipython3

    # Split the Data into Training and Testing Sets

    # Step 1: Check if 'Survived' exists in the DataFrame before splitting the data
    if '2urvived' in df.columns:
        # Step 2: Define features (X) and target (y)
        X = df.drop('2urvived', axis=1)  # Drop 'Survived' from features
        y = df['2urvived']               # Target variable
        print("'Survived' column found and split defined.")

        # Step 3: Split the data into training and testing sets
        from sklearn.model_selection import train_test_split

        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

        print("Training set size:", X_train.shape)
        print("Testing set size:", X_test.shape)
    else:
        print("Error: 'Survived' column not found in the dataset. Check if it was dropped earlier.")


.. code:: ipython3

    # Apply Two Supervised Learning Algorithms
    #Logistic Regression

    from sklearn.linear_model import LogisticRegression
    from sklearn.metrics import accuracy_score, classification_report

    # Step 8: Apply Logistic Regression
    lr_model = LogisticRegression(max_iter=1000)
    lr_model.fit(X_train, y_train)

    # Step 9: Make predictions
    y_pred_lr = lr_model.predict(X_test)

    # Step 10: Evaluate Logistic Regression
    print("Logistic Regression Accuracy:", accuracy_score(y_test, y_pred_lr))
    print(classification_report(y_test, y_pred_lr))
```

.. code:: ipython3

```
    # Random Forest Classifier

    from sklearn.ensemble import RandomForestClassifier

    # Step 11: Apply Random Forest Classifier
    rf_model = RandomForestClassifier(n_estimators=100)
    rf_model.fit(X_train, y_train)

    # Step 12: Make predictions
    y_pred_rf = rf_model.predict(X_test)

    # Step 13: Evaluate Random Forest
    print("Random Forest Accuracy:", accuracy_score(y_test, y_pred_rf))
    print(classification_report(y_test, y_pred_rf))
```

.. code:: ipython3

```
    # Compare and Contrast the Algorithms

    # Comparison of Logistic Regression and Random Forest results
    print("Comparison of the Models:")
    print(f"Logistic Regression Accuracy: {accuracy_score(y_test, y_pred_lr)}")
    print(f"Random Forest Accuracy: {accuracy_score(y_test, y_pred_rf)}")
```