Introduction

The sinking of the Titanic in 1912 is one of the most tragic events in maritime history. The Titanic dataset from **Kaggl** contains detailed information about the passengers, including their age, gender, class, and whether they survived. By analyzing this data using machine learning techniques, we can build models that predict the likelihood of survival based on specific features.

we explore two models: **Logistic Regression** (a linear baseline) and **Random Forest** (an ensemble method). The goal is to identify key survival factors and evaluate model performance, emphasizing accuracy, interpretability, and robustness.

Models like this can help us in the future to handle such disasters more efficiently by providing insights into which factors are most important during emergencies. These insights can support better planning, resource allocation, and passenger management in similar situations.

Methodology

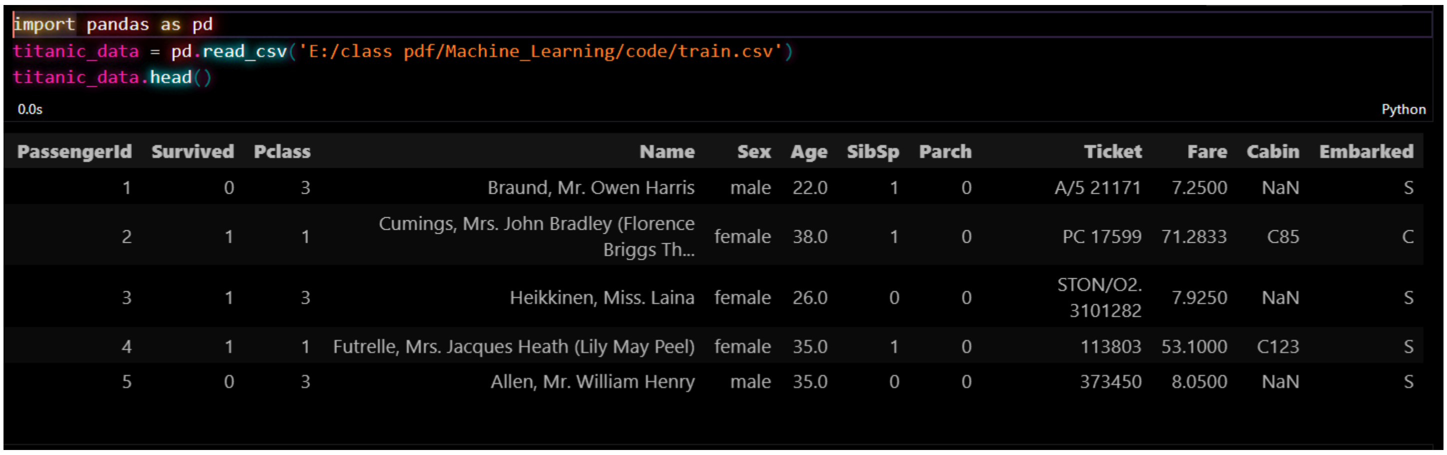
We divided our methodology into six main sections. This approach helped us track the project progress more effectively and allowed room for future updates or modifications if needed. The six sections are:

- Data
- pre-processing data
- Analyses
- test train split
- Model taring
- Evolution

Below, we explain each of these sections clearly and precisely.

Data:

Data set is collected is from **Kaggle.** The data set contains total of **891** passengers training data. Here an overview of the data set.



```python
import pandas as pd
titanic_data = pd.read_csv('E:/class pdf/Machine_Learning/code/train.csv')
titanic_data.head()
```

0.0s Python

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

fig: Default data set

Data Dictionary,

- Sex: male (m), female (f)
- Age: In years
- Sibsp: Number of siblings/ spouses
- Parch: Number of parents/ children
- Ticket: Ticket number
- fare: Passenger fare in dollars
- Cabin: Cabin number
- Embarked: Cherbourg(C), Queenstown(Q), Southampton(S)
- Pclass: upper (1), middle (2), lower (3)
- Survived: 0 = No, 1 = Yes

Pre-processing

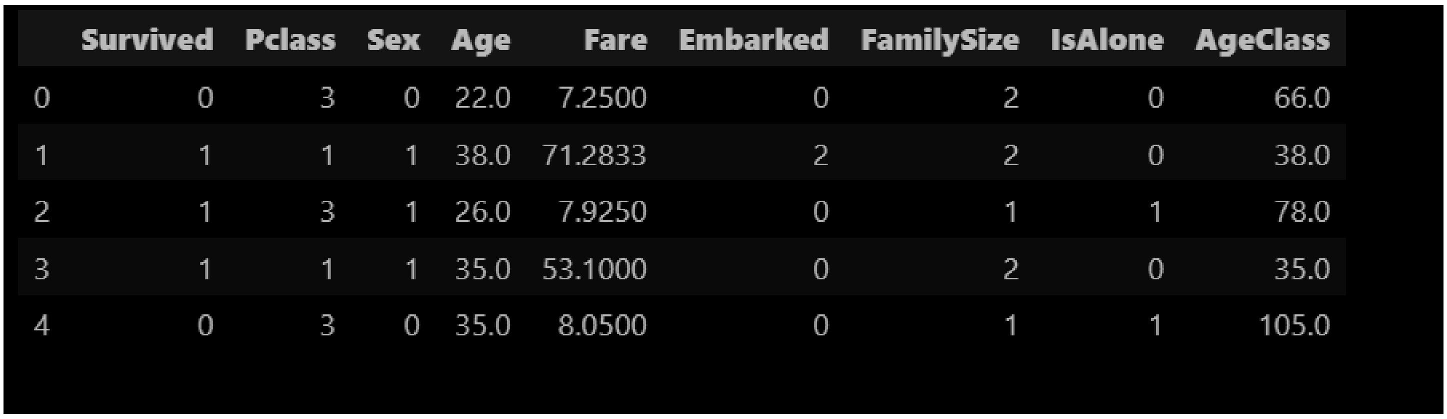
This is one of the most important sections of the project. The model's accuracy depends heavily on how well we handle and prepare the data. Initially, the model was trained without proper preprocessing, which led to biased predictions. The model showed higher accuracy (75–80%) for predicting 'Not Survived' but struggled to correctly predict 'Survived' cases (only 60%). This imbalance highlighted the importance of cleaning and transforming the dataset before training. Therefore, comprehensive preprocessing steps were applied to improve overall model performance.

After understanding the dataset, we performed several preprocessing steps to clean and prepare the data for training. These steps were essential to improve model performance and handle inconsistencies in the dataset. Below are the key preprocessing tasks we applied:

- **Handled Missing Values:**
  - Filled missing values in the **Age** column with the **mean age**.
  - Filled missing values in the **Embarked** column using the **most frequent value** (mode).
- **Encoded Categorical Features:**
  - Converted the **Embarked** column: S → 0, C → 1, Q → 2.
  - Converted the **Sex** column: Male → 0, Female → 1.
- **Dropped Unnecessary Columns:**

  - Removed columns like Cabin and others that contained too many missing values or were not useful for training.
- **Feature Engineering:**
  - Created a new feature **FamilySize** by combining SibSp and Parch and adding 1 (to include the person themselves).
  - Created a binary feature **IsAlone** to indicate whether the passenger was alone (1 = Yes, 0 = No).
  - Created **AgeClass** by multiplying Age and Pclass to capture the interaction between age and class.

These preprocessing steps helped clean the dataset, create meaningful features, and significantly boosted the accuracy of the models.



	Survived	Pclass	Sex	Age	Fare	Embarked	FamilySize	IsAlone	AgeClass
0	0	3	0	22.0	7.2500	0	2	0	66.0
1	1	1	1	38.0	71.2833	2	2	0	38.0
2	1	3	1	26.0	7.9250	0	1	1	78.0
3	1	1	1	35.0	53.1000	0	2	0	35.0
4	0	3	0	35.0	8.0500	0	1	1	105.0

fig: Modified versions

Analysis:

In this section, we aimed to understand which features had the most impact on a passenger's survival. We used various plots to compare features with the **Survived** column and visually analyze their relationships. This helped us identify which factors played a significant role in determining survival. Some of the key graphs and insights are shown below.

Total passenger 891:

- Did Not Survive (0): 549 passengers (~61.6%)
- Survived (1): 342 passengers (~38.4%)

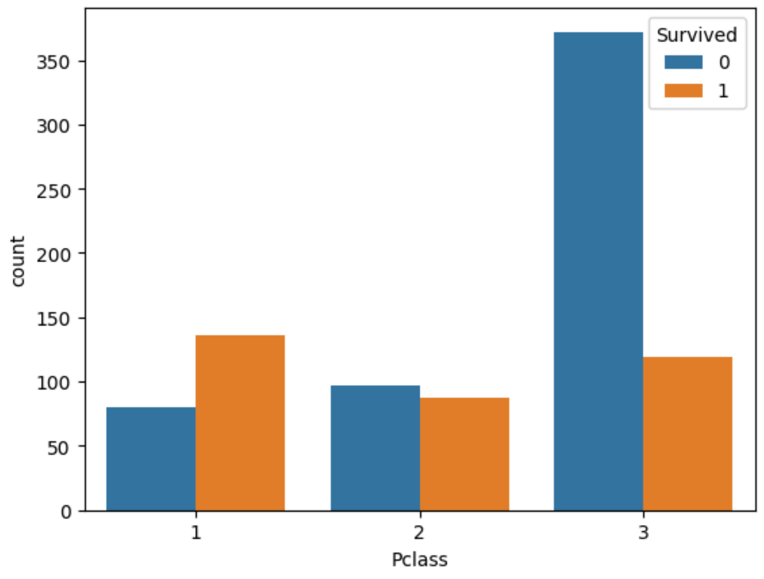


fig: Pclass vs Survived

This figure shows how a passenger’s class affected their chances of survival. First-class passengers had the highest survival rate, around 60–70%, while third-class passengers had the lowest, only 20–30%. This difference is mainly due to socioeconomic status, as higher-class passengers had better access to lifeboats and safety resources.

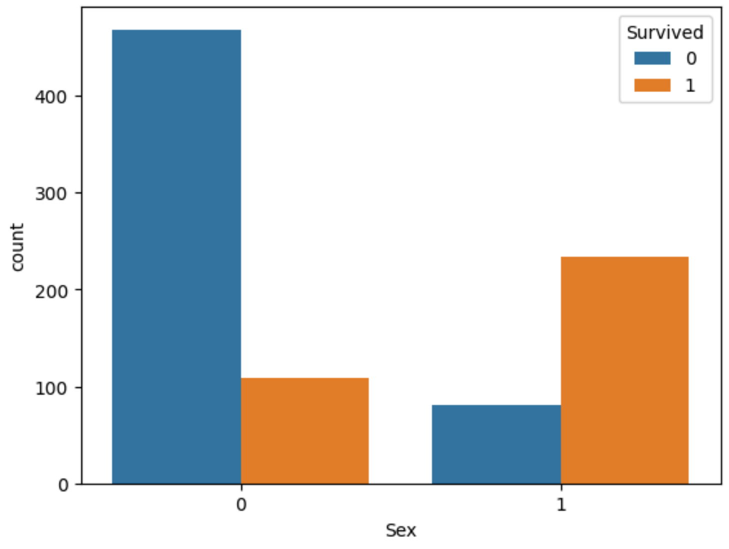


fig: Sex vs Survived

This figure highlights the gender-based survival difference during the disaster. Although there were more males on board, a significantly higher percentage of females survived — about 70–75% of women survived compared to only 15–20% of men. This was likely due to the "women and children first" policy during evacuation.

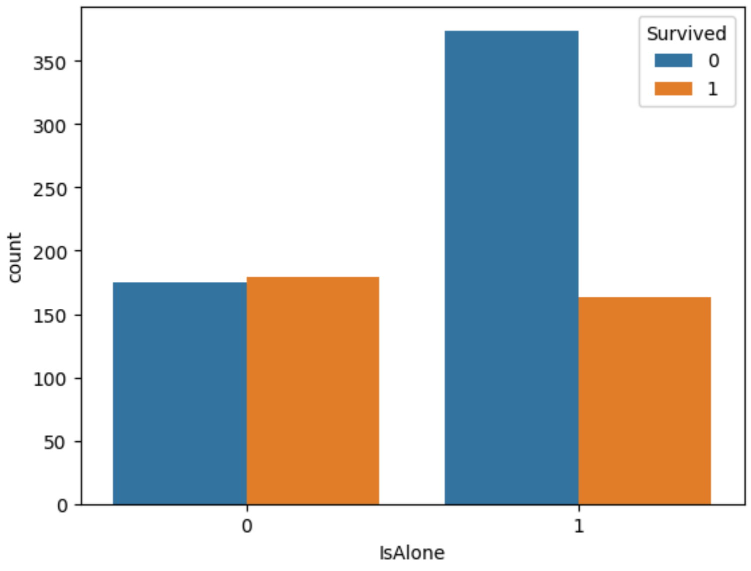


fig: isAlone vs Survived

This figure compares the survival rates of passengers traveling alone (IsAlone = 1) versus those traveling with family (IsAlone = 0). Passengers with family members had a higher survival rate, likely due to group support and cooperation during the evacuation process.

Summary of Findings:

Feature	**Survival Impact**
Pclass	**1st class > 2nd class > 3rd class**
Sex	**Females survived at ~3× the rate of males**
IsAlone	**Passengers with family had better survival odds**

Test-train split:

To build a reliable machine learning model, it is important to split the dataset into training and testing sets. This allows us to train the model on one portion of the data and evaluate its performance on unseen data to avoid overfitting.

In our project, we first separated the features (independent variables) and the target label (dependent variable). The features used were: **Pclass, Sex, Age, IsAlone, AgeClass, Fare, Embarked,** and **FamilySize**, while the target label was **Survived**, indicating whether a passenger lived or not.

We then used the train_test_split function from the **scikit-learn** library to divide the data:

- 80% of the data was used for training the model.
- 20% of the data was used for testing the model’s performance.
- A random_state was set to ensure consistent results every time the code is run.

After the split, we had:

- **712 samples** in the training set
- **179 samples** in the testing set

This separation helped us validate the model's performance accurately and ensured that the evaluation was done on data the model had never seen before.

Model training:

In this project, we implemented and compared two popular machine learning classification models: **Logistic Regression** and **Random Forest Classifier**. Initially, the model was trained using only Logistic Regression. However, we observed that the accuracy was **imbalanced and biased**—the model performed well when predicting passengers who did **not survive** (with 75–80% accuracy), but its performance dropped significantly when predicting those who **did survive**, achieving only about **60% accuracy**.

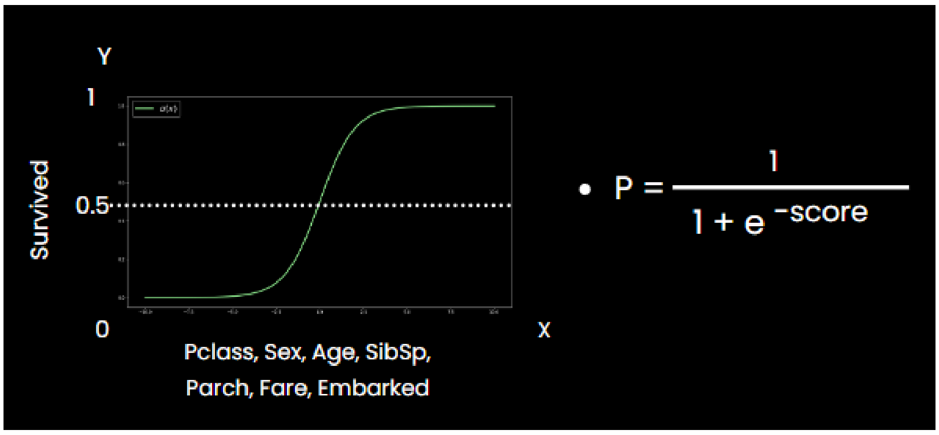
After analyzing the results, we identified two main reasons for this imbalance:

1. **Lack of proper data preprocessing**
2. **Insufficient model tuning**

The first issue has been addressed in the **Data Preprocessing** section. To further improve performance and reduce bias, we also introduced a second model—**Random Forest Classifier** which allows us to compare results and achieve a more balanced and accurate prediction outcome. Below we explain how we fix the 2nd problem step by step process. Before that here's the overview of the model we used.

Logistic Regression:

Logistic Regression is a statistical method used for binary classification problems. It estimates the probability that a given input point belongs to a certain class (in this case, survived or not). The model uses a sigmoid function to map predictions to probabilities between 0 and 1.



$$P = \frac{1}{1 + e^{-\text{score}}}$$

fig: sigmoid function

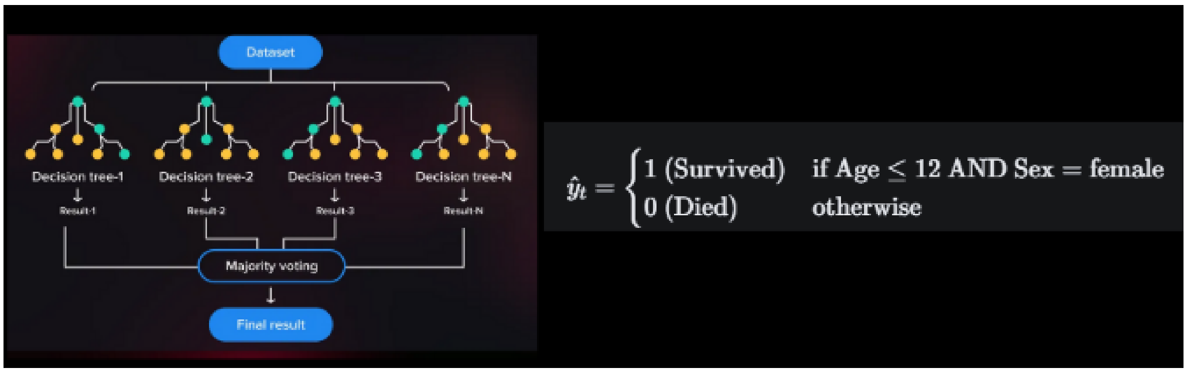
Where: score = β0+β1 · Pclass+β2 · Sex+β3 · Age+...

This model is simple, interpretable, and works well when the relationship between features and outcome is approximately lineal.

Random Forest:

Random Forest is an ensemble learning method that combines multiple decision trees to improve prediction accuracy. Each tree is trained on a subset of the data and makes its own prediction. The final output is decided by majority voting from all trees.

This model is especially powerful in capturing complex relationships between features, and it reduces the risk of overfitting compared to individual decision trees.



$$\hat{y}_t = \begin{cases} 1 \text{ (Survived)} & \text{if Age} \leq 12 \text{ AND Sex} = \text{female} \\ 0 \text{ (Died)} & \text{otherwise} \end{cases}$$

fig: decision trees

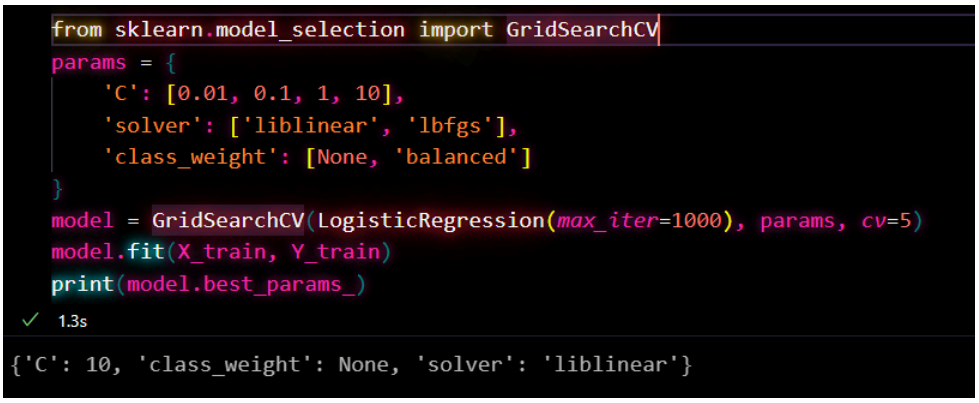
In our dataset, decision trees were primarily split based on key features like **Age, Sex,** and **Pclass**. For example, the model may predict a passenger's survival if they are female and under the age of 12, as shown in the figure.

Model Training and Hyperparameter Tuning:

To improve the accuracy and robustness of the machine learning models we applied **GridSearchCV** for systematic hyperparameter tuning. This method helps identify the best combination of parameters by performing cross-validation over a specified range of values. And help us solve our 2nd problem we mentioned previously.

1. Logistic Regression:

- **C**: Regularization strength; smaller values specify stronger regularization. I tested values like 0.01, 0.1, 1, and 10.
- **solver**: Algorithm used for optimization.
  a. 'liblinear' is suitable for smaller datasets.
  b. 'lbfgs' is generally used for larger datasets.
- **class_weight**: This helps handle class imbalance (i.e., more non-survivors than survivors solving the problem mentioned previously) by giving different weights to different classes ('None' and 'balanced' were tested).



```
from sklearn.model_selection import GridSearchCV
params = {
    'C': [0.01, 0.1, 1, 10],
    'solver': ['liblinear', 'lbfgs'],
    'class_weight': [None, 'balanced']
}
model = GridSearchCV(LogisticRegression(max_iter=1000), params, cv=5)
model.fit(X_train, Y_train)
print(model.best_params_)
```

```
{'C': 10, 'class_weight': None, 'solver': 'liblinear'}
```

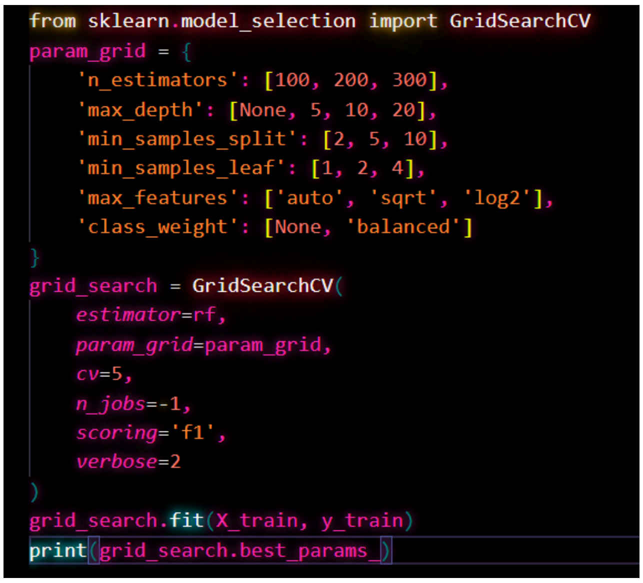
fig: using GridSearchCV for Logistic Regression

The best parameters found were:
{'C': 10, 'class_weight': None, 'solver': 'liblinear'}

2. Random Forest:

- **n_estimators**: Number of trees in the forest ([100, 200, 300])
- **max_depth**: Maximum depth of each tree ([None, 5, 10, 20])
- **min_samples_split**: Minimum number of samples required to split an internal node ([2, 5, 10])
- **min_samples_leaf**: Minimum number of samples required at a leaf node ([1, 2, 4])
- **max_features**: Number of features considered when looking for the best split (['auto', 'sqrt', 'log2'])
- **class_weight**: Balanced weight to handle class imbalance ([None, 'balanced'])

Here's the code snippet for hyperparameter tuning with Random Forest:



```python
from sklearn.model_selection import GridSearchCV
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 5, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['auto', 'sqrt', 'log2'],
    'class_weight': [None, 'balanced']
}
grid_search = GridSearchCV(
    estimator=rf,
    param_grid=param_grid,
    cv=5,
    n_jobs=-1,
    scoring='f1',
    verbose=2
)
grid_search.fit(X_train, y_train)
print(grid_search.best_params_)
```

fig: using GridSearchCV for Random Forest

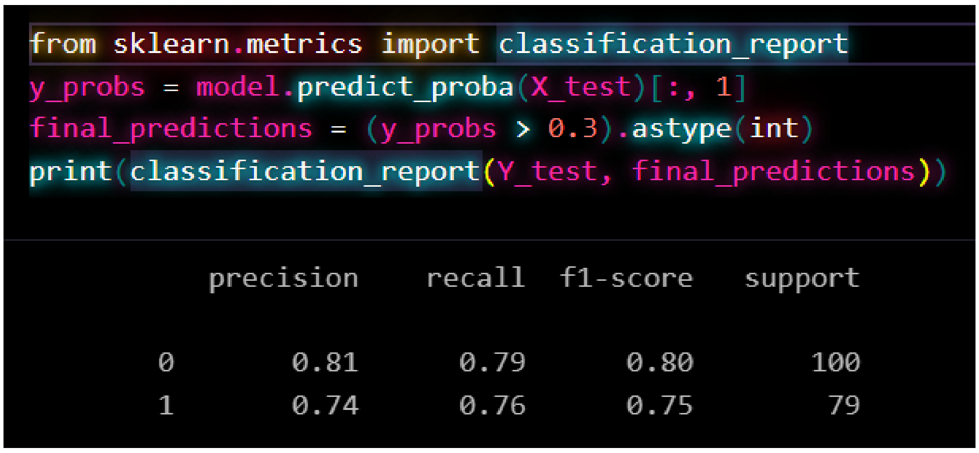
The best combination of parameters was:
{'class_weight': 'balanced', 'max_depth': 5, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100}

These hyperparameter tuning steps ensured that the models were optimally trained for the Titanic dataset, minimizing overfitting and improving generalization on unseen data.

Model Evaluation:

To evaluate the performance of both Logistic Regression and Random Forest classifiers, we used key classification metrics such as **precision**, **recall**, and **F1-score** for both classes: **Not Survived (0)** and **Survived (1)**.

The classification_report from the sklearn.metrics module was used to generate these metrics. For Logistic Regression, probabilities were used and a threshold of 0.3 was applied to make final predictions. For Random Forest, predictions were obtained directly from the trained model.

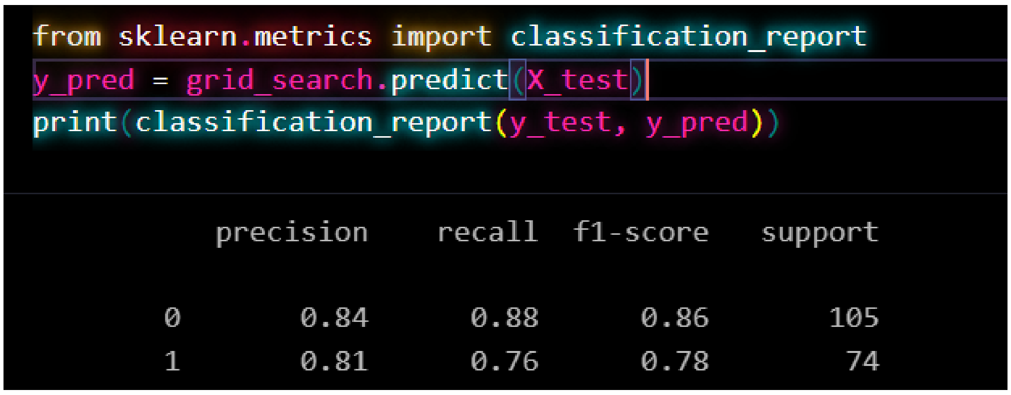


```python
from sklearn.metrics import classification_report
y_probs = model.predict_proba(X_test)[:, 1]
final_predictions = (y_probs > 0.3).astype(int)
print(classification_report(Y_test, final_predictions))
```

```
              precision    recall  f1-score   support

           0       0.81      0.79      0.80       100
           1       0.74      0.76      0.75        79
```

fig: classification_report for Logistic Regression



```
from sklearn.metrics import classification_report
y_pred = grid_search.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.84      0.88      0.86       105
           1       0.81      0.76      0.78        74
```

fig: classification_report for Random Forest

To better understand and communicate the model's performance, I visualized the classification results using the following graphs:

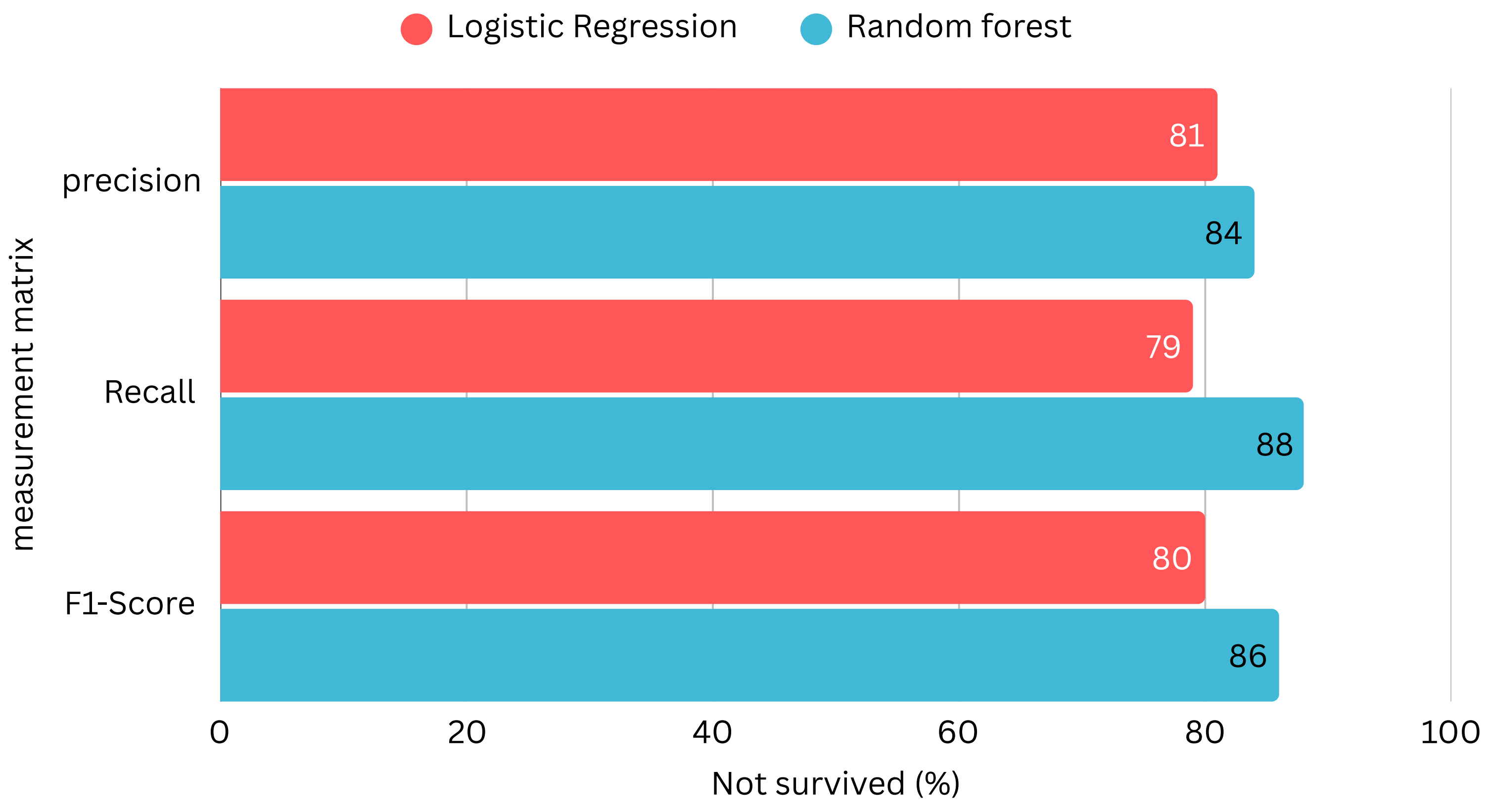
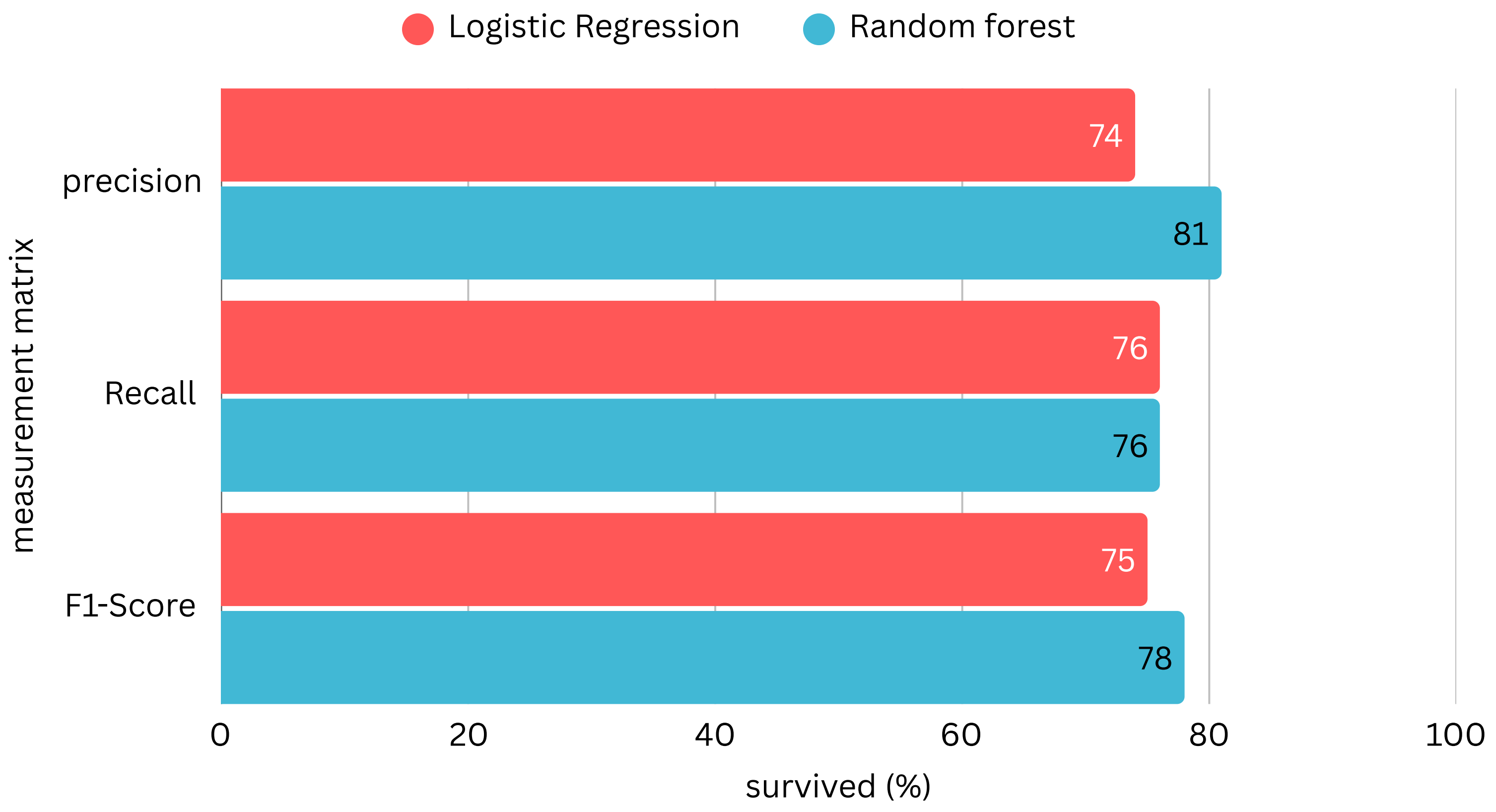


fig: Logistics Regression Vs Random Forest (Not survived)



From the comparison presented in the graphs, it is evident that **Random Forest consistently outperforms Logistic Regression** across all evaluation metrics—**precision**, **recall**, and **F1-score** —for both classes (survived and not survived). The improvement is especially notable for **Class 0 (Not Survived)**, where Random Forest achieves a recall of **0.88** compared to Logistic Regression's **0.79**, which means it is significantly better at correctly identifying passengers who did not survive. This is critical in imbalanced classification problems like this one, where accurately identifying the majority class (non-survivors) often dominates model performance.

The overall trend indicates that Random Forest, being an ensemble method, benefits from the aggregation of multiple decision trees, allowing it to model complex non-linear relationships and interactions in the data. On the other hand, Logistic Regression, while simpler and more interpretable, may not capture these complexities as effectively.

Therefore, based on the observed performance metrics, **Random Forest emerges as the more robust and reliable model** for predicting passenger survival on the Titanic dataset. It not only improves classification accuracy but also handles class imbalance more effectively, making it a more suitable choice for deployment or further analysis in this context.

Conclusion

In this project, we explored the use of machine learning techniques to predict passenger survival on the Titanic. One of the most important takeaways from our work is the **critical role of data preprocessing**. Initially, without proper preprocessing, our Logistic Regression model showed **biased performance**—predicting non-survivors with high accuracy (up to 80%) but significantly underperforming when predicting survivors (only about 60%).

After applying appropriate preprocessing techniques—such as handling missing values, encoding categorical variables, and feature engineering—the model's overall performance improved noticeably. This step not only boosted accuracy but also helped the model generalize better to unseen data, highlighting how essential **clean and well-structured data** is for any machine learning task.

Additionally, we implemented and compared **Random Forest Classifier** with Logistic Regression. We also conducted **hyperparameter tuning**, particularly with the Random Forest model, using grid search to find the optimal combination of parameters such as the number of estimators, maximum depth, and minimum samples split. This tuning process significantly enhanced the model's ability to capture meaningful patterns without overfitting. It helped maximize key performance metrics like recall and F1-score, especially for the minority class (survivors), where small improvements can make a big difference. As shown in the results, Random Forest outperformed Logistic Regression in almost every metric:

- For class **0** (did not survive), Random Forest achieved **0.86 F1-Score** compared to **0.80** with Logistic Regression.
- For class **1** (survived), it achieved **0.78 F1-Score** versus **0.75**.

This improvement can be credited to Random Forest's ability to handle complex patterns and relationships in the data through an ensemble of decision trees. It reduces overfitting and provides more **balanced predictions**, especially when the data has nonlinear dependencies or interactions.

In conclusion, this project highlights how **important proper data preprocessing and model tuning are**. Even a simple model like Logistic Regression can perform reasonably well when prepared and tuned properly, but more sophisticated models like Random Forest can bring additional gains. These practices not only boost accuracy but also ensure more **balanced and fair predictions**—which is critical in real-world applications such as disaster response and management systems.

Related work

1. **Kaggle Titanic: Machine Learning from Disaster**

Summary: This Kaggle competition serves as a benchmark for binary classification tasks. Participants employ various machine learning models, including Logistic Regression and Random Forest, often enhancing performance through feature engineering techniques like extracting titles from names and creating family size indicators.

link: https://www.kaggle.com/competitions/titanic

2.**A-Comparative-Study-on-Machine-Learning-Techniques-using-Titanic-Dataset**

Summary: This project compares machine learning algorithms (e.g., logistic regression, decision trees, random forests, SVM, KNN) on the Titanic dataset to predict passenger survival. It evaluates performance using accuracy, precision, recall, and F1-score.

Link:
GitHub - Muhammadisrar47/A-Comparative-Study-on-Machine-Learning-Techniques-usin...

3. **Classification of Titanic Passenger Data and Chances of Surviving the Disaster**

Summary: This study utilizes the Weka data mining tool to analyze passenger data, identifying key factors like cabin class, age, and point of departure that significantly influence survival chances.
link: https://arxiv.org/abs/1810.09851

4. **Survival Prediction and Comparison of the Titanic Based on Machine Learning Classifiers**

Summary: The paper compares Logistic Regression, Decision Tree, and Random Forest classifiers, concluding that Random Forest, especially with 45 or 75 trees, outperforms others in precision and recall metrics.
link: https://wepub.org/index.php/TCSISR/article/view/2428

5. **The Titanic Survival Prediction Using Predictive Modeling Algorithms**

Summary: This research applies multiple classifiers, including Random Forest, Logistic Regression, Naive Bayes, KNN, and Decision Trees, to predict survival. It emphasizes the importance of feature selection and model evaluation using metrics like accuracy and F1-score.
link: https://publications.isods.org/index.php/jdsai/article/view/15

6. **Titanic Disaster Prediction Based on Machine Learning Algorithms**

Summary: The study compares Decision Tree and Random Forest algorithms, finding that Decision Tree slightly outperforms Random Forest in accuracy. It highlights age, sex, and ticket class as the most significant features influencing survival.
link:
https://www.researchgate.net/publication/370569058_Titanic_Disaster_Prediction_Based_on_Machine_Learning_AlgorithmsResearchGate+1Academia+1

7. **Research on Titanic Survival Prediction Based on Machine Learning Method**

Summary: This paper employs a hard voting ensemble method combining Logistic Regression, Random Forest, and Decision Tree classifiers. The model achieves an accuracy of 87.64%, demonstrating the effectiveness of ensemble techniques.
link: https://www.ewadirect.com/proceedings/aemps/article/view/19451EWA Direct

8. **Machine Learning from Disaster: Predicting the Titanic Survival Rate, a Random Forest Approach**

Summary: Utilizing the Weka tool, this study applies the Random Forest algorithm to predict survival, achieving internal accuracy rates of up to 82%. It underscores the model's robustness in handling complex datasets.
link:
https://www.researchgate.net/publication/330909610_Machine_Learning_from_Disaster_Predicting_the_Titanic_Survival_Rate_a_Random_Forest_approach

9. **Titanic Survival Prediction Based on Machine Learning Algorithms**

Summary: This report demonstrates the application of Logistic Regression and Random Forest models on the Titanic dataset. It emphasizes data preprocessing and feature engineering, such as analyzing name length, to enhance model performance.
link: https://drpress.org/ojs/index.php/HSET/article/view/23123

10. **Titanic Survival Prediction Using Machine Learning**

Summary: A comprehensive project that involves data collection, preprocessing, exploratory data analysis, feature engineering, and model training. It highlights the significance of demographic features like age, sex, and ticket class in predicting survival.
link:
https://www.academia.edu/122009451/Titanic_Survival_Prediction_Using_Machine_Learning
Academia+1ResearchGate+1