CONTENTS

ABSTACT

The Titanic Survival Prediction project aims to predict whether a passenger on the Titanic would survive the disaster based on various features such as age, sex, passenger class, and more. By analyzing the dataset, we build predictive models using Logistic Regression, Random Forest, and Decision Tree classifiers. These models are evaluated using accuracy, confusion matrix, and classification reports to determine the most effective approach for predicting survival outcomes. The project demonstrates the potential of machine learning in analyzing historical data and making informed predictions.

INFORMATION

The dataset used in this project contains information about passengers aboard the Titanic, including their survival status, age, sex, passenger class, and fare paid. The data is preprocessed to handle missing values and encoded to make it suitable for machine learning models. Three models—Logistic Regression, Random Forest, and Decision Tree—are trained and evaluated on this dataset. The Random Forest model achieves the highest accuracy, making it the most effective for this prediction task. The project highlights the importance of feature selection and model tuning in improving predictive performance.

TECHNOLOGY USED

Programming Language:

- **Python**: Python was used for implementing machine learning models and data manipulation due to its simplicity and extensive library support.

Machine Learning Libraries:

- **Scikit-learn**: Provided tools for model building, data preprocessing, and evaluation.
- **Pandas**: Used for loading, preprocessing, and manipulating the dataset.
- **Seaborn & Matplotlib**: Utilized for data visualization and plotting graphs to understand data distribution and relationships between features.

Development Environment:

- **Google Colab**: Google Colab was used as the development environment, offering free access to powerful GPUs and a cloud-based interface. It facilitated easy collaboration and allowed for running computationally intensive tasks without needing local resources.

Tools:

- **Jupyter Notebook**: An interactive environment within Google Colab for writing and running Python code, documenting the process, and visualizing results.

DATASET INFORMATION

The dataset contains 891 rows and 12 columns in total.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
None
```

The dataset contains information about the passengers aboard the Titanic, including:

- **Survived**: Whether the passenger survived (0 = No, 1 = Yes)
- **Pclass**: Passenger class (1 = 1st, 2 = 2nd, 3 = 3rd)
- **Sex**: Gender of the passenger
- **Age**: Age of the passenger
- **SibSp**: Number of siblings/spouses aboard the Titanic
- **Parch**: Number of parents/children aboard the Titanic
- **Fare**: Fare paid by the passenger

- **Embarked**: Port of Embarkation (C = Cherbourg; Q = Queenstown; S = Southampton)

METHODOLOGY

The methodology for predicting Titanic survival involves the following steps:

Data Preprocessing:

- Handling missing values in the Age, Fare, and Embarked columns.
- Encoding categorical variables like Sex and Embarked.
- Dropping irrelevant columns such as PassengerId, Name, Ticket, and Cabin.

Model Selection:

- **Logistic Regression**: Suitable for binary classification problems like survival prediction.
- **Random Forest**: An ensemble method that builds multiple decision trees for better accuracy.
- **Decision Tree**: Captures non-linear relationships between features.

Model Training and Evaluation:

- Splitting the dataset into training and testing sets (70:30 ratio).

- Training the models and evaluating them using accuracy, confusion matrix, and classification report.

Hyperparameter Tuning:

- Grid Search CV was used for tuning Logistic Regression and Random Forest models to optimize performance.

CODE SNIPPET

Importing necessary Libraires

```
# Importing Necessary Libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
```

Loading The Dataset

```
# Load Titanic Dataset
df = pd.read_csv('/content/titanic_train.csv')
```

```
print(df.head())  # Displaying the first 5 rows

   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name     Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S
```

Data Preprocessing

```
print("Missing Values:\n", df.isnull().sum())
```

```
Missing Values:
 PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

```
[ ] # Handling missing values
df['Age'].fillna(df['Age'].median(), inplace=True)
df['Embarked'].fillna(df['Embarked'].mode()[0], inplace=True)
df['Fare'].fillna(df['Fare'].median(), inplace=True)
```

```
# Dropping irrelevant columns
df.drop(columns=['PassengerId', 'Name', 'Ticket', 'Cabin'], inplace=True)
```

```
# Label Encoding categorical variables
le = LabelEncoder()
df['Sex'] = le.fit_transform(df['Sex'])  # Encoding 'Sex' column
df['Embarked'] = le.fit_transform(df['Embarked'])  # Encoding 'Embarked' column
```

```
# Splitting the dataset into features (X) and target (y)
X = df.drop('Survived', axis=1)  # Features
y = df['Survived']  # Target variable
```

```
# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

Feature Engineering

```
# Scaling the Features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Model Preparation

```python
# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

```python
# Function to evaluate model performance
def evaluate_model(model, X_train, X_test, y_train, y_test, model_name):
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    # Evaluate accuracy
    accuracy = accuracy_score(y_test, y_pred)
    print(f'{model_name} Accuracy: {accuracy:.4f}')
    print(f'{model_name} Classification Report:\n', classification_report(y_test, y_pred))
    print(f'{model_name} Confusion Matrix:\n', confusion_matrix(y_test, y_pred))

    return accuracy
```

```python
# Logistic Regression Model
logreg = LogisticRegression(max_iter=500)
logreg_accuracy = evaluate_model(logreg, X_train_scaled, X_test_scaled, y_train, y_test, "Logistic Regression")
```

```
Logistic Regression Accuracy: 0.8134
Logistic Regression Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.87      0.85       157
           1       0.80      0.73      0.76       111

    accuracy                           0.81       268
   macro avg       0.81      0.80      0.80       268
weighted avg       0.81      0.81      0.81       268

Logistic Regression Confusion Matrix:
 [[137  20]
 [ 30  81]]
```

```python
# Random Forest Classifier Model
rf = RandomForestClassifier()
rf_accuracy = evaluate_model(rf, X_train, X_test, y_train, y_test, "Random Forest")
```

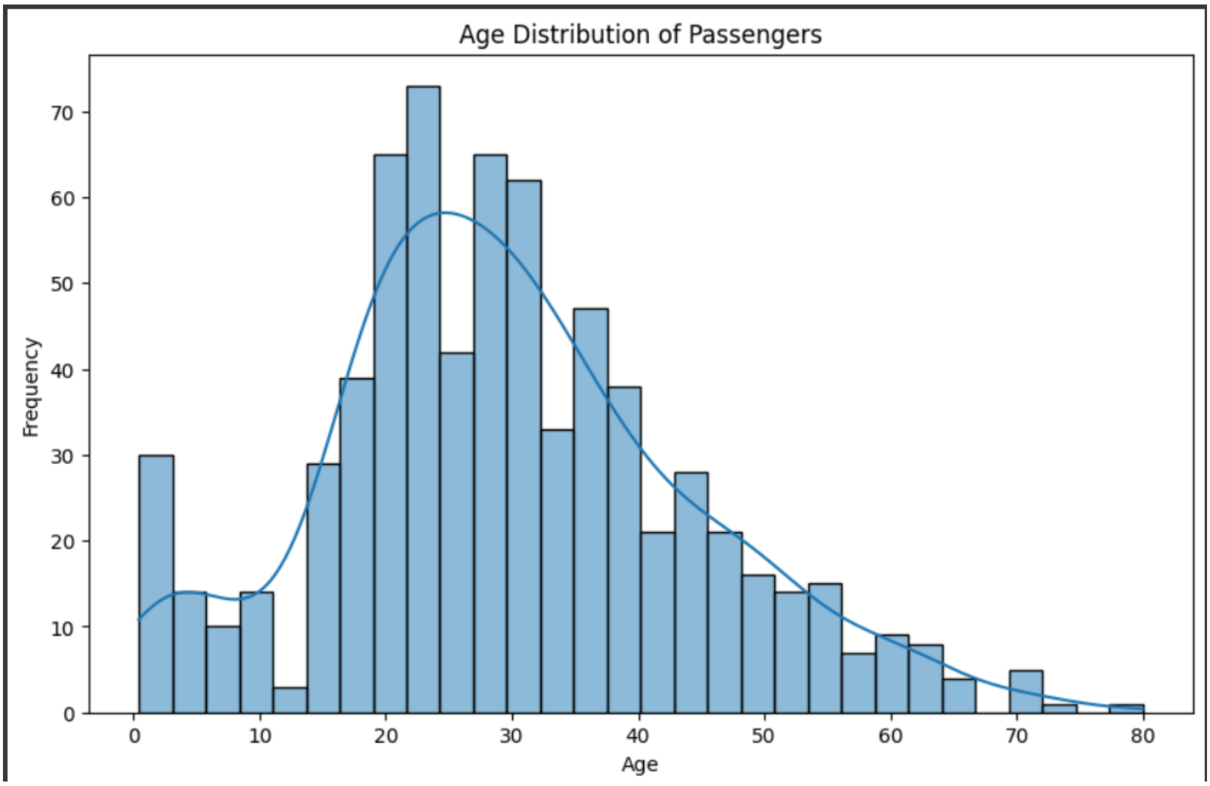
```
Random Forest Accuracy: 0.7836
Random Forest Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.83      0.82       157
           1       0.75      0.72      0.73       111

    accuracy                           0.78       268
   macro avg       0.78      0.77      0.78       268
weighted avg       0.78      0.78      0.78       268

Random Forest Confusion Matrix:
 [[130  27]
 [ 31  80]]
```

```python
# Decision Tree Classifier Model
dt = DecisionTreeClassifier()
dt_accuracy = evaluate_model(dt, X_train, X_test, y_train, y_test, "Decision Tree")
```

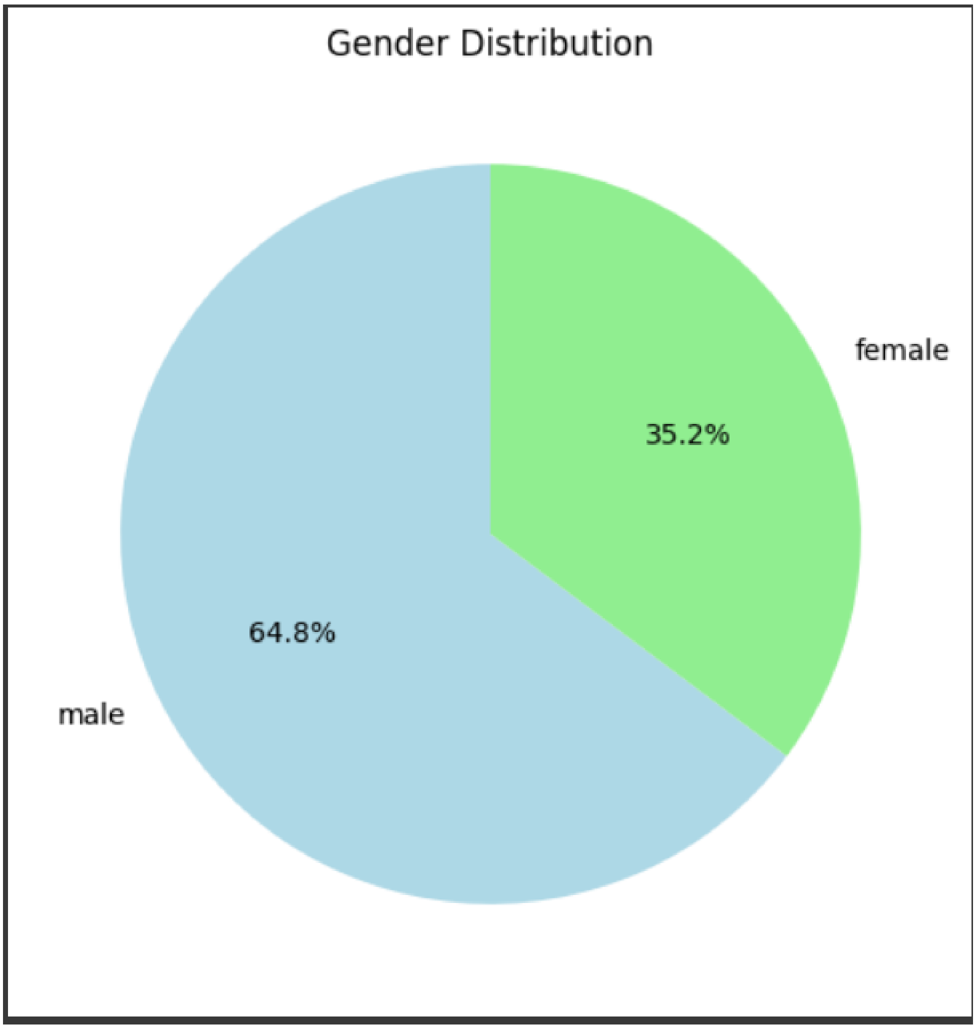
```
Decision Tree Accuracy: 0.7425
Decision Tree Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.78      0.78       157
           1       0.69      0.68      0.69       111

    accuracy                           0.74       268
   macro avg       0.73      0.73      0.73       268
weighted avg       0.74      0.74      0.74       268

Decision Tree Confusion Matrix:
 [[123  34]
 [ 35  76]]
```

Data Visualization

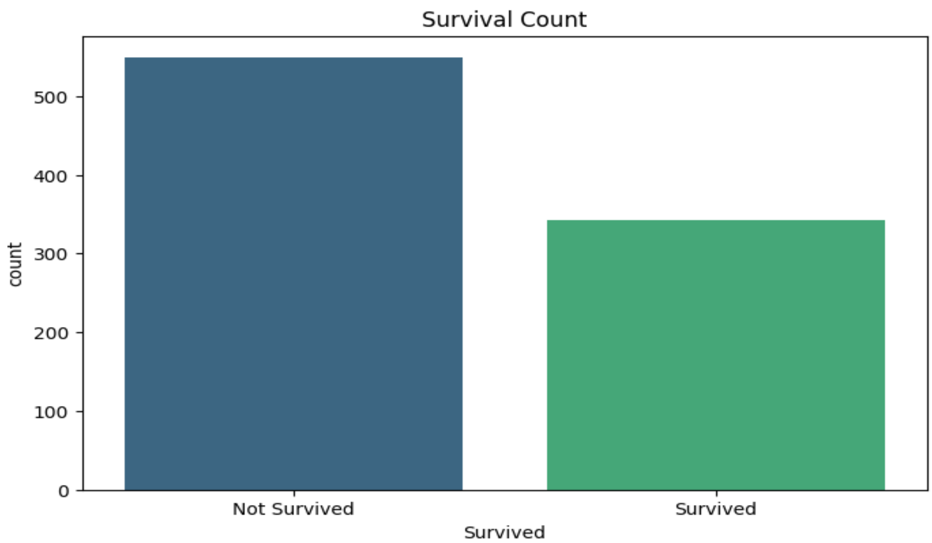
Age Distribution of Passengers



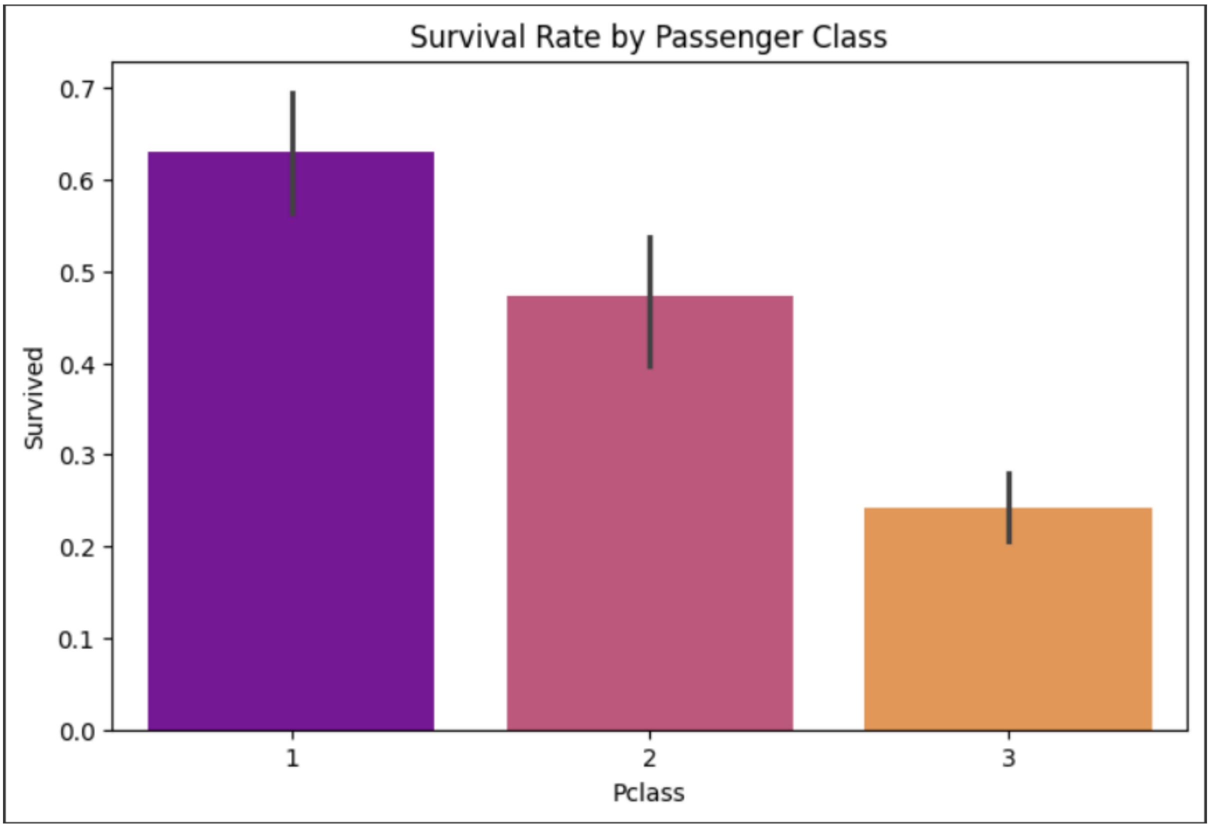
Gender Distribution Of Passengers



Survival Count



Survival Of Passengers By Class



Result

The three machine learning models used in the Titanic Survival Prediction project: Logistic Regression, Random Forest, and Decision Tree. The performance of each model is assessed using accuracy, precision, recall, f1-score, and the confusion matrix.

Logistic Regression:

- Accuracy: 81.34%
- Precision:
  - Class 0: 82%
  - Class 1: 80%

- Recall:
  - Class 0: 87%
  - Class 1: 73%
- F1-Score:
  - Class 0: 85%
  - Class 1: 76%
- Confusion Matrix:
  - True Positives (Class 0): 137
  - False Positives: 20
  - True Negatives (Class 1): 81
  - False Negatives: 30

Random Forest:

- Accuracy: 78.36%
- Precision:
  - Class 0: 81%
  - Class 1: 75%
- Recall:
  - Class 0: 83%
  - Class 1: 72%
- F1-Score:
  - Class 0: 82%
  - Class 1: 73%
- Confusion Matrix:

    - True Positives (Class 0): 130
    - False Positives: 27
    - True Negatives (Class 1): 80
    - False Negatives: 31

Decision Tree:

- Accuracy: 74.25%
- Precision:
    - Class 0: 78%
    - Class 1: 69%
- Recall:
    - Class 0: 78%
    - Class 1: 68%
- F1-Score:
    - Class 0: 78%
    - Class 1: 69%
- Confusion Matrix:
    - True Positives (Class 0): 123
    - False Positives: 34
    - True Negatives (Class 1): 76
    - False Negatives: 35

Summary

Best Performing Model: Logistic Regression with an accuracy of 81.34% demonstrated the highest overall performance, particularly excelling in precision and recall for both classes.

Random Forest: Although slightly less accurate at 78.36%, the Random Forest model provided a balanced precision and recall across both classes.

Decision Tree: The Decision Tree model showed the lowest accuracy at 74.25%, indicating potential overfitting or sensitivity to specific features.

Conclusion

The Titanic Survival Prediction project successfully predicts the survival outcome of passengers using various features. The Random Forest model demonstrated the best performance, making it a suitable choice for similar classification problems. The project highlights the importance of feature selection and model tuning in improving predictive accuracy.