Titanic Survival Prediction Using Machine Learning

A Research-Based Technical Report

Author: Satyam Gajjar

Abstract
This research paper explores the use of machine learning to predict passenger survival in the Titanic disaster. The study involves data preprocessing, feature engineering, logistic regression modeling, and evaluation. This work demonstrates how classical ML algorithms can extract meaningful insights from structured historical datasets.

1. Introduction

The Titanic dataset is one of the most well-known datasets in machine learning. It contains detailed passenger information such as age, sex, class, and fare. The goal is to predict whether a passenger survived the disaster using ML techniques.

2. Dataset Description

The dataset consists of multiple features, including: - Passenger Class (Pclass) - Sex (Gender) - Age - Fare - Cabin and Ticket details The target variable *Survived* is binary: 1 = survived, 0 = did not survive.

3. Methodology

The machine learning pipeline includes: 1. Data Loading and Exploration 2. Handling missing values 3. Dropping irrelevant columns 4. Encoding categorical variables 5. Train-test split 6. Logistic Regression model training 7. Model evaluation

4. Data Preprocessing

Data preprocessing included: - Dropping columns like Name, Ticket, Cabin, PassengerId - Filling missing Age values - Converting Sex and Embarked categorical values into numeric format These steps ensure the dataset is ready for ML modeling.

5. Feature Scaling

Feature scaling improves numerical stability and model accuracy for algorithms such as logistic regression. Standardization may be applied depending on the feature distribution.

6. Model Training

A Logistic Regression classifier was used due to its simplicity, interpretability, and effectiveness for binary classification tasks. The model was trained on preprocessed data.

7. Model Evaluation

Evaluation was done using accuracy score, confusion matrix, and prediction testing. The model successfully identifies key survival factors based on input features.

8. Prediction Pipeline

Prediction on new passenger data involves: - Applying the same preprocessing steps - Structuring input in the required ML format - Using the trained model to produce a binary prediction

9. Conclusion

The Titanic survival prediction model demonstrates that logistic regression can effectively classify survival outcomes. Further improvements may include hyperparameter tuning, feature engineering, and advanced algorithms like Random Forests or Gradient Boosting.

Appendix: Notebook Code and Explanations

```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

Data Collection and Preprocessing

```
titanic_dataset = pd.read_csv('/content/drive/MyDrive/Data
Science/Projects/15 Titanic Survival Prediction/train.csv')
titanic_dataset.head()
```

```
#check the rows and cols
titanic_dataset.shape
```

```
#get some info for the data
titanic_dataset.info()
```

```
#check null values
titanic_dataset.isnull().sum()
```

Handling the missing values

Drop the Cabin column

```
titanic_dataset = titanic_dataset.drop(['Cabin'], axis=1)
titanic_dataset.head()
```

```
#replace the missing values in age columnwith mean values
titanic_dataset['Age'].fillna(titanic_dataset['Age'].mean(), inplace=True)
titanic_dataset.isnull().sum()
```

```
titanic_dataset['Embarked'].value_counts()
```

```
#Findind the mode value of the Embarked column
titanic_dataset['Embarked'].mode()
```

```
titanic_dataset['Embarked'].fillna(titanic_dataset['Embarked'].mode()[0],
inplace=True)
titanic_dataset.isnull().sum()
```

Data Analysis

```
#getting some statistical measures
titanic_dataset.describe()

#Find no. of people survived
titanic_dataset['Survived'].value_counts()
```

Data Visualization

```
sns.set()
#making count plot for survived column
sns.countplot(data=titanic_dataset, x='Survived')

#Checking for the Sex column
titanic_dataset['Sex'].value_counts()

sns.set()
#making count plot for survived column
sns.countplot(data=titanic_dataset, x='Sex')

#Number of survivors Gender vise
sns.countplot(data=titanic_dataset, x='Sex', hue='Survived')

#for Pclass
titanic_dataset['Pclass'].value_counts()

#ticket class wise
sns.countplot(data=titanic_dataset, x='Pclass', hue='Survived')
```

Replacing - Encoding the categorical columns

```
titanic_dataset['Sex'].value_counts()

titanic_dataset['Embarked'].value_counts()

#Replace the sex with 0 and 1
titanic_dataset.replace({
'Sex':{'male':0, 'female':1},
'Embarked':{'S':0, 'C':1, 'Q':2}
}, inplace=True)
titanic_dataset.head()
```

Splitting the data and features

```
x = titanic_dataset.drop(['Name', 'Ticket', 'PassengerId', 'Survived'],
axis=1)
```

```
y = titanic_dataset['Survived']
```

Now, splitting the data into training and test data

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,
random_state=2)
x.shape, x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

Model Training

```
model = LogisticRegression()
model.fit(x_train, y_train)
```

Model Evaluation: Accuracy

```
x_train_prediction = model.predict(x_train)
x_train_accuracy = accuracy_score(x_train_prediction, y_train)
x_train_accuracy*100
```

```
x_test_prediction = model.predict(x_test)
x_test_accuracy = accuracy_score(x_test_prediction, y_test)
x_test_accuracy*100
```

Make predictive system

```
input_data = x_test.iloc[2]
predictions = model.predict([input_data])
predictions
```

```
if predictions[0] == 0:
print('Dead')
else:
print('Survived')
```

```
y_test
```

Author: Satyam Gajjar