Name: Atharva Choudhari Roll no.: 41105

Batch: P1 Class: BE 1 Subject: LP-3 (ML)

Mini Project 2 -ML

1. Title

 Title: Titanic Survival Prediction: A Machine Learning Model Using Passenger Data

Date: 14 O ct, 2024

---

2. Abstract

This project involves building a machine learning model that predicts the likelihood of a passenger surviving the Titanic shipwreck based on various factors such as age, gender, ticket class, and more. The Kaggle Titanic dataset was used to train and evaluate the model. We explored various machine learning algorithms, including Logistic Regression, Decision Trees, and Random Forests, and compared their performance. The final model demonstrates a strong predictive ability, offering insights into the socio-economic factors that influenced survival rates.

---

3. Introduction

The sinking of the Titanic is one of the most well-known disasters in history. In this project, we aim to analyze the passenger data and build a machine learning model that can predict whether a person survived or not based on characteristics such as age, gender, and socio-economic class. This task is a common introductory problem for machine learning, which allows us to explore a variety of classification algorithms.

Key Points:

- **Problem Statement:** Given the historical data from the Titanic disaster, can we predict whether a passenger survived or not?
- **Machine Learning Goal:** To develop a model that predicts the survival of passengers based on their demographic and socio-economic attributes.

---

4. Objective

The objectives of this project are:

1. To analyze and clean the Titanic passenger dataset.
2. To explore different machine learning algorithms for classification.
3. To compare the performance of these algorithms and select the best model.
4. To provide insights into the key factors that influenced survival.

---

5. Dataset Overview

The dataset used for this project is provided by Kaggle, which contains information about 891 passengers. Each row represents one person, and the columns describe various features related to the passengers.

5.1. Dataset Link

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

5.2. Key Features

- **Survived:** The target variable (1 = survived, 0 = did not survive)
- **Pclass:** Ticket class (1st, 2nd, 3rd class)
- **Name:** Name of the passenger
- **Sex:** Gender (male or female)
- **Age:** Age of the passenger
- **SibSp:** Number of siblings/spouses aboard the Titanic
- **Parch:** Number of parents/children aboard the Titanic
- **Ticket:** Ticket number
- **Fare:** Passenger fare
- **Cabin:** Cabin number
- **Embarked:** Port of Embarkation (C = Cherbourg, Q = Queenstown, S = Southampton)

---

6. Methodology

6.1. Data Preprocessing

Data preprocessing is an essential step before building the machine learning model. It involves cleaning the dataset, handling missing values, encoding categorical features, and scaling the data.

1. **Handling Missing Values:**
    - The dataset contains missing values in the Age, Cabin, and Embarked columns.
    - We imputed missing ages using the median of passengers' ages. o For the Cabin feature, since most values were missing, we dropped the column.
    - For Embarked, we replaced missing values with the most common embarkation point.
2. **Encoding Categorical Variables:**
    - The Sex and Embarked columns are categorical. We applied label encoding to convert them into numerical values.
3. **Feature Scaling:**
    - The Fare column contains a wide range of values. We applied standard scaling to normalize this feature.

Code Example for Preprocessing (Python, using Pandas and Scikit-learn):

python Copy code

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler

# Load dataset
data = pd.read_csv('titanic.csv')

# Fill missing values in 'Age' with median data['Age'].fillna(data['Age'].median(),
inplace=True)

# Drop 'Cabin' and fill 'Embarked' with most common value data.drop('Cabin',
axis=1, inplace=True)
data['Embarked'].fillna(data['Embarked'].mode()[0], inplace=True)

# Encode categorical variables label_encoder
= LabelEncoder()
data['Sex'] = label_encoder.fit_transform(data['Sex']) data['Embarked']
= label_encoder.fit_transform(data['Embarked'])
```

# Feature scaling for 'Fare' scaler

= StandardScaler()

data['Fare'] = scaler.fit_transform(data['Fare'].values.reshape(-1, 1))

# Separate features and target variable

X = data.drop('Survived', axis=1) y

= data['Survived']

# Split the data into training and test sets

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

6.2. Model Selection

We explored different machine learning models to find the one with the best performance for this classification problem:

- **Logistic Regression:** A simple yet effective classification algorithm that works well for binary outcomes.
- **Decision Trees:** A non-linear model that splits the data into nodes based on feature values.
- **Random Forests:** An ensemble of decision trees that provides better performance and reduces overfitting.

Logistic Regression:

python Copy

code

from sklearn.linear_model import LogisticRegression from

sklearn.metrics import accuracy_score

# Initialize and train the model

logreg = LogisticRegression()

logreg.fit(X_train, y_train)

# Predict and calculate accuracy y_pred

= logreg.predict(X_test)

print("Logistic Regression Accuracy:", accuracy_score(y_test, y_pred)) **Random Forest Classifier:**

python Copy

code

from sklearn.ensemble import RandomForestClassifier

# Initialize and train the Random Forest model

rf = RandomForestClassifier(n_estimators=100, random_state=42)

rf.fit(X_train, y_train)

# Predict and calculate accuracy y_pred_rf

= rf.predict(X_test)

print("Random Forest Accuracy:", accuracy_score(y_test, y_pred_rf))

---

7. Results and Discussion

7.1. Model Evaluation

The models were evaluated using accuracy as the primary metric. We also explored precision, recall, and F1 score to measure the models' performance more comprehensively.

Model	**Accuracy**	**Precision**	**Recall**	**F1 Score**
Logistic Regression	79%	78%	80%	79%
Decision Tree	76%	75%	76%	75%
Random Forest	82%	81%	83%	82%

7.2. Feature Importance

Random Forests provide a measure of feature importance, which helps in understanding which features had the most influence on the predictions. In this case, the Sex and Pclass features were the most important predictors, followed by Fare and Age.

Code to Display Feature Importance:

python Copy

code

import matplotlib.pyplot as plt import

numpy as np

```
# Get feature importances from Random Forest
importances = rf.feature_importances_ indices
= np.argsort(importances)[::-1] features =
X.columns

# Plot feature importances
plt.figure(figsize=(10, 6)) plt.title("Feature
Importance")
plt.bar(range(X.shape[1]), importances[indices], align="center")
plt.xticks(range(X.shape[1]), features[indices], rotation=90)
plt.show()
```

7.3. Insights

- **Gender Impact:** Females had a significantly higher chance of survival than males.
- **Class Influence:** Passengers in 1st class were more likely to survive compared to those in 3rd class.
- **Fare and Age:** Younger passengers and those who paid higher fares had a higher probability of survival.

---

8. Conclusion

This project successfully demonstrated the application of machine learning to predict the survival of passengers aboard the Titanic. By experimenting with various models, we found that the Random Forest Classifier provided the best performance. Key insights drawn from the model reveal that gender, socio-economic class, and fare were the most critical factors influencing survival. These findings are consistent with historical records, which indicate that women and children, especially those in higher classes, were prioritized during the evacuation.

---

9. Future Scope

Future improvements to this project could include:

- **Hyperparameter Tuning:** Further optimizing the models using techniques such as Grid Search to fine-tune the parameters.

- **Ensemble Methods:** Exploring other ensemble methods like Gradient Boosting or XGBoost to potentially improve model performance.
- **Deep Learning:** Applying neural networks or other advanced models to further enhance the prediction accuracy.