Titanic Survival Prediction - Documentation

Overview

This project aims to develop a machine learning model that predicts whether a passenger survived the Titanic disaster. The dataset includes key features such as age, gender, ticket class, fare, and cabin information, which help in determining survival probabilities.

Steps Implemented in the Notebook

1. Importing Required Libraries

The first step is to import necessary Python libraries, including:

- **pandas** for data manipulation
- **numpy** for numerical computations
- **matplotlib & seaborn** for data visualization
- **sklearn** for machine learning model development

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
```

2. Loading the Dataset

The Titanic dataset is loaded into a Pandas DataFrame.

```
data = pd.read_csv("titanic.csv")
data.head()
```

3. Data Exploration

- Checking for missing values
- Understanding data distributions
- Identifying categorical and numerical variables

```
print(data.info())
print(data.describe())
```

4. Data Preprocessing

Handling Missing Values:

- 'Age' is filled with the median value.
- 'Embarked' is filled with the most frequent value.
- 'Cabin' is dropped due to excessive missing values.

```
data['Age'].fillna(data['Age'].median(), inplace=True)
data['Embarked'].fillna(data['Embarked'].mode()[0], inplace=True)
data.drop(columns=['Cabin'], inplace=True)
```

Encoding Categorical Variables:

- 'Sex' is converted to numerical values (Male = 0, Female = 1).
- 'Embarked' is transformed using one-hot encoding.

```
data['Sex'] = data['Sex'].map({'male': 0, 'female': 1})
data = pd.get_dummies(data, columns=['Embarked'], drop_first=True)
```

5. Feature Selection & Splitting Data

Important features are selected, and the dataset is split into training and testing sets.

```
features = ['Pclass', 'Age', 'Fare', 'Sex', 'Embarked_Q', 'Embarked_S']
X = data[features]
y = data['Survived']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

6. Model Training

A **Random Forest Classifier** is used for prediction.

```
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

7. Model Evaluation

The model's performance is evaluated using:

- **Accuracy Score**
- **Classification Report** (Precision, Recall, F1-score)

```
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

8. Conclusion

The final model provides a good survival prediction accuracy. Additional improvements can be made by:

- Hyperparameter tuning
- Trying different machine learning algorithms
- Using feature engineering techniques
