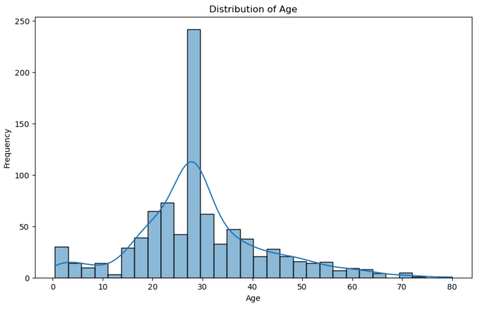
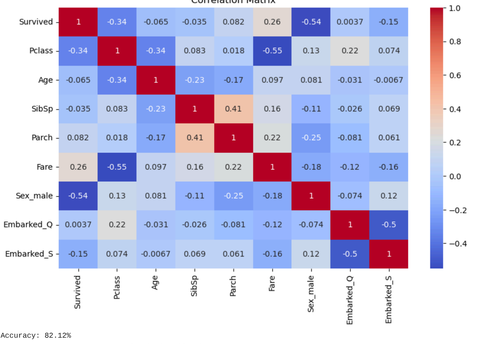
Use the Titanic dataset to build a model that predicts whether a

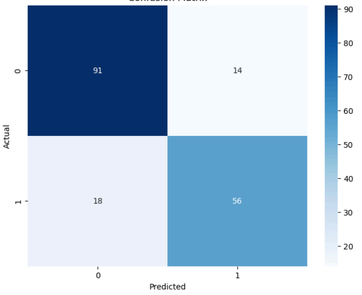
passenger on the Titanic survived or not. This is a classic beginner project with readily available data. The dataset typically used for this project contains information about individual passengers, such as their age, gender, ticket class, fare, cabin, and whether or not they survived.

In [1]:
```
# Importing required libraries
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 StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix

# Function to load and preprocess the Titanic dataset
def load_and_preprocess_data(file_path):
    """
    Load and preprocess the Titanic dataset.

    Parameters:
    file_path (str): File path of the dataset

    Returns:
    DataFrame: Preprocessed dataset
    """
    # Load the dataset
    data = pd.read_csv(file_path)

    # Drop unnecessary columns
    data.drop(['PassengerId', 'Name', 'Ticket', 'Cabin'], axis=1, inplace=True)

    # Handle missing values
    data['Age'].fillna(data['Age'].median(), inplace=True)
    data['Embarked'].fillna(data['Embarked'].mode()[0], inplace=True)

    # Encode categorical variables
    data = pd.get_dummies(data, columns=['Sex', 'Embarked'], drop_first=True)

    return data

# Function to visualize the data
def visualize_data(data):
    """
    Visualize the distribution of 'Age' and correlation matrix.

    Parameters:
    data (DataFrame): Preprocessed dataset
    """
    # Plot distribution of 'Age'
    plt.figure(figsize=(10, 6))
    sns.histplot(data['Age'], kde=True)
    plt.title('Distribution of Age')
    plt.xlabel('Age')
    plt.ylabel('Frequency')
    plt.show()

    # Plot correlation matrix
    plt.figure(figsize=(10, 6))
    correlation_matrix = data.corr()
    sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
    plt.title('Correlation Matrix')
    plt.show()

# Function to train and evaluate the model
def train_and_evaluate_model(X_train, X_test, y_train, y_test):
    """
    Train and evaluate the Random Forest classifier.

    Parameters:
    X_train (DataFrame): Training features
    X_test (DataFrame): Testing features
    y_train (Series): Training target
    y_test (Series): Testing target
    """
    # Feature scaling
    scaler = StandardScaler()
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test)

    # Initialize Random Forest classifier
    clf = RandomForestClassifier(random_state=42)

    # Train the model
    clf.fit(X_train, y_train)

    # Make predictions
    y_pred = clf.predict(X_test)

    # Evaluate the model
    accuracy = accuracy_score(y_test, y_pred)
    print(f'Accuracy: {accuracy * 100:.2f}%')

    # Plot confusion matrix
    conf_matrix = confusion_matrix(y_test, y_pred)
    plt.figure(figsize=(8, 6))
    sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
    plt.title('Confusion Matrix')
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.show()

# Main function
if __name__ == '__main__':
    # Corrected file path with quotes
    file_path = 'Titanic-Dataset[1].csv'  # Replace with your actual file path

    # Load and preprocess data
    data = load_and_preprocess_data(file_path)

    # Visualize data
    visualize_data(data)

    # Split dataset
    X = data.drop('Survived', axis=1)
    y = data['Survived']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    # Train and evaluate model
    train_and_evaluate_model(X_train, X_test, y_train, y_test)
```






```
Accuracy: 82.12%
```



DOCUMENATION:

Importing Required Libraries

The code initiates by importing crucial libraries essential for data manipulation, visualization, and machine learning modeling. These encompass:

pandas: For data manipulation, analysis, and DataFrame operations numpy: For numerical computations, array operations, and mathematical calculations matplotlib.pyplot: For basic data visualization, plotting graphs, and visual representation seaborn: For advanced data visualization, statistical graphics, and enhanced plotting sklearn: For machine learning tasks including model selection, preprocessing (StandardScaler), evaluation (accuracy_score, confusion_matrix), and ensemble methods (RandomForestClassifier) Function Definitions

1. load_and_preprocess_data(file_path) This function meticulously handles the Titanic dataset by performing the following preprocessing steps:

Eliminating redundant columns (PassengerId, Name, Ticket, Cabin) Managing missing values in Age and Embarked columns using median and mode imputation techniques Encoding categorical variables (Sex and Embarked) into numerical representations using one-hot encoding Parameters:

file_path (str): Absolute or relative file path pointing to the dataset file Returns:

DataFrame: Cleaned and preprocessed dataset ready for analysis and modeling

1. visualize_data(data) This function offers insightful visualizations to understand the dataset's characteristics:

Visualizing the distribution of Age using a histogram with KDE (Kernel Density Estimation) for smooth representation Displaying the correlation matrix heatmap to identify relationships between features Parameters:

data (DataFrame): Preprocessed dataset obtained after data cleaning and transformation

1. train_and_evaluate_model(X_train, X_test, y_train, y_test) This function leverages a Random Forest classifier for predictive modeling and evaluates its performance:

Feature scaling of training and testing datasets using StandardScaler to standardize features Initializing a Random Forest classifier with default hyperparameters Training the classifier on the training dataset Making predictions on the testing dataset Evaluating the model's accuracy using accuracy_score Displaying the confusion matrix to visualize true positives, true negatives, false positives, and false negatives Parameters:

X_train (DataFrame): Training features dataset X_test (DataFrame): Testing features dataset y_train (Series): Training target variable y_test (Series): Testing target variable Main Execution The primary execution flow of the code encompasses:

Defining the file path to the Titanic dataset stored locally or remotely Invoking the load_and_preprocess_data() function to ingest and preprocess the dataset Utilizing the visualize_data() function to generate insightful visualizations for data exploration Partitioning the dataset into training and testing subsets using train_test_split with a default test size of 0.2 Engaging the train_and_evaluate_model() function to execute the end-to-end machine learning pipeline, encompassing model training, prediction, evaluation, and visualization of results

In [ ]:
```
```