Heart Disease Dataset

the "Heart Disease" dataset from the UCI Machine Learning Repository, which is a bit more complex and involves data cleaning, EDA, and classification. We will use this dataset to train and evaluate several classification models.

Steps:

1. **Load the dataset and perform data cleaning.**
2. **Conduct Exploratory Data Analysis (EDA).**
3. **Train several classification models.**
4. **Evaluate the models.**

Complete Code with Explanations

```
# Import necessary 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.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, classification_report,
roc_curve, auc

# Load the dataset
url =
"https://archive.ics.uci.edu/ml/machine-learning-databases/heart-disea
se/processed.cleveland.data"
column_names = ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs',
'restecg', 'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal',
'target']
data = pd.read_csv(url, header=None, names=column_names)

# Display the first few rows of the dataset
print("First few rows of the dataset:")
```

```
print(data.head())

# Data Cleaning
# Replace '?' with NaN and convert the columns to numeric
data = data.replace('?', np.nan)
data = data.apply(pd.to_numeric)

# Check for missing values
print("\nMissing values in the dataset:")
print(data.isnull().sum())

# Fill missing values with the median value of each column
data = data.fillna(data.median())

# Verify that there are no more missing values
print("\nMissing values after filling:")
print(data.isnull().sum())

# EDA
print("\nStatistical summary of the dataset:")
print(data.describe())

# Plotting distributions of numerical features
plt.figure(figsize=(12, 8))
for i, column in enumerate(['age', 'trestbps', 'chol', 'thalach',
'oldpeak'], 1):
    plt.subplot(2, 3, i)
    sns.histplot(data[column], kde=True)
    plt.title(f'Distribution of {column}')
plt.tight_layout()
plt.show()

# Correlation matrix
plt.figure(figsize=(12, 8))
sns.heatmap(data.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```

```
# Data Preprocessing
# Split the data into features and target variable
X = data.drop('target', axis=1)
y = data['target']

# Standardize the features
scaler = StandardScaler()
X = scaler.fit_transform(X)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Model Training and Evaluation
models = {
    'Logistic Regression': LogisticRegression(),
    'Decision Tree': DecisionTreeClassifier(),
    'Random Forest': RandomForestClassifier(n_estimators=100)
}

results = []

for model_name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    # Evaluation metrics
    cm = confusion_matrix(y_test, y_pred)
    cr = classification_report(y_test, y_pred)
    fpr, tpr, _ = roc_curve(y_test, model.predict_proba(X_test)[:, 1])
    roc_auc = auc(fpr, tpr)

    results.append((model_name, cm, cr, fpr, tpr, roc_auc))

# Visualize the results
for model_name, cm, cr, fpr, tpr, roc_auc in results:
    print(f"\nModel: {model_name}")
    print("Confusion Matrix:")
```

```
print(cm)
print("\nClassification Report:")
print(cr)

plt.figure()
plt.plot(fpr, tpr, label=f'{model_name} (area = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title(f'ROC Curve - {model_name}')
plt.legend(loc="lower right")
plt.show()
```

Explanation

1. **Loading the Dataset**:
    - We load the Heart Disease dataset and specify the column names.
    - Display the first few rows to understand the structure.
2. **Data Cleaning**:
    - Replace missing values represented by '?' with NaN.
    - Convert columns to numeric types.
    - Fill missing values with the median of each column.
3. **Exploratory Data Analysis (EDA)**:
    - Display statistical summary.
    - Plot distributions of numerical features.
    - Display a correlation matrix to understand relationships between features.
4. **Data Preprocessing**:
    - Split data into features (X) and target variable (y).
    - Standardize features using `StandardScaler`.
    - Split data into training and testing sets.
5. **Model Training and Evaluation**:
    - Train Logistic Regression, Decision Tree, and Random Forest classifiers.
    - Evaluate each model using confusion matrix, classification report, and ROC curve.
    - Display evaluation metrics and ROC curves.