Predicting Heart Disease using Logistic Regression

Introduction

This project aims to predict the presence of heart disease in patients based on various health indicators using a logistic regression model. The dataset used for this analysis is the Heart Disease UCI dataset. The steps include data preprocessing, exploratory data analysis, model training, evaluation, and interpretation.

Steps and Implementation

1. **Load the Dataset**

We begin by loading the dataset heart.csv which contains health indicators and the target variable indicating the presence of heart disease.

Load the dataset:

```
[1]  import pandas as pd
     data = pd.read_csv('heart.csv')
```

2. **Handle Missing Values**

Missing values in the dataset are handled by replacing them with the median value of the respective column.

Handle missing values:

```
[2]  data = data.dropna()
```

3. **Encode Categorical Variables**

Categorical variables are encoded using one-hot encoding to convert them into a format suitable for machine learning algorithms.

Encode categorical variables using one-hot encoding:

```
[3]  data = pd.get_dummies(data, columns=['sex'])
```

4. **Scale Numerical Features**

Numerical features are scaled using StandardScaler to ensure that all features contribute equally to the model performance.

Scale numerical features:

```
[4]  from sklearn.preprocessing import StandardScaler
     scaler = StandardScaler()
     data[['age', 'chol', 'trestbps']] = scaler.fit_transform(data[['age', 'chol', 'trestbps']])
```

5. **Perform Exploratory Data Analysis**

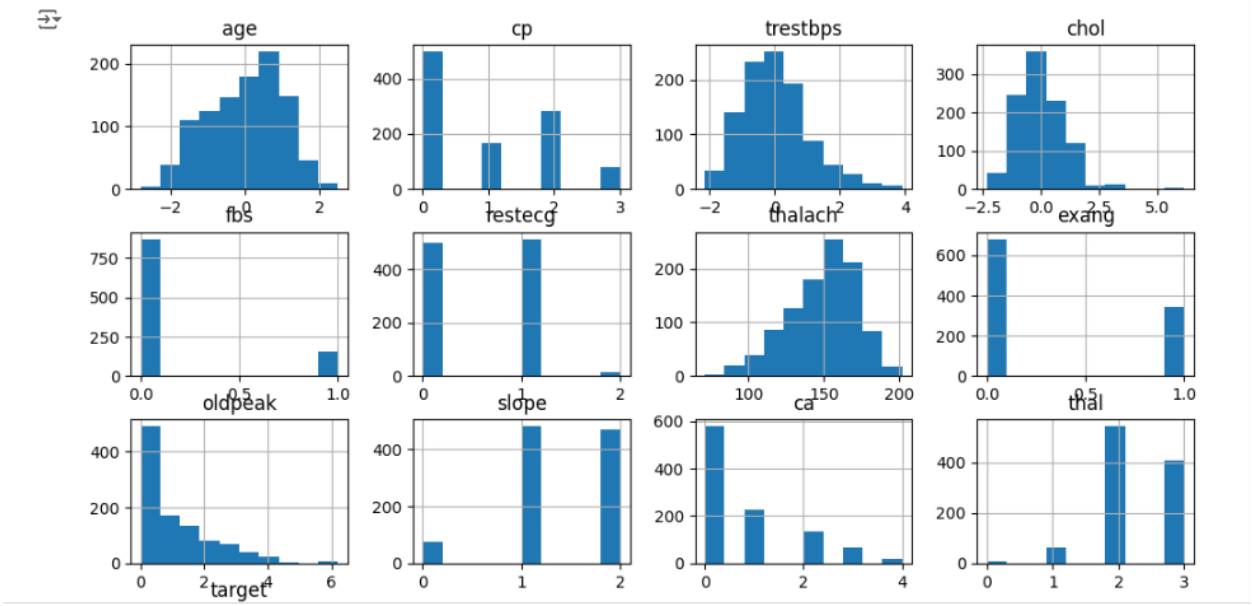
We perform exploratory data analysis to understand the distribution of the data and relationships between variables. This includes summary statistics, histograms, and pair plots.

Perform exploratory data analysis:

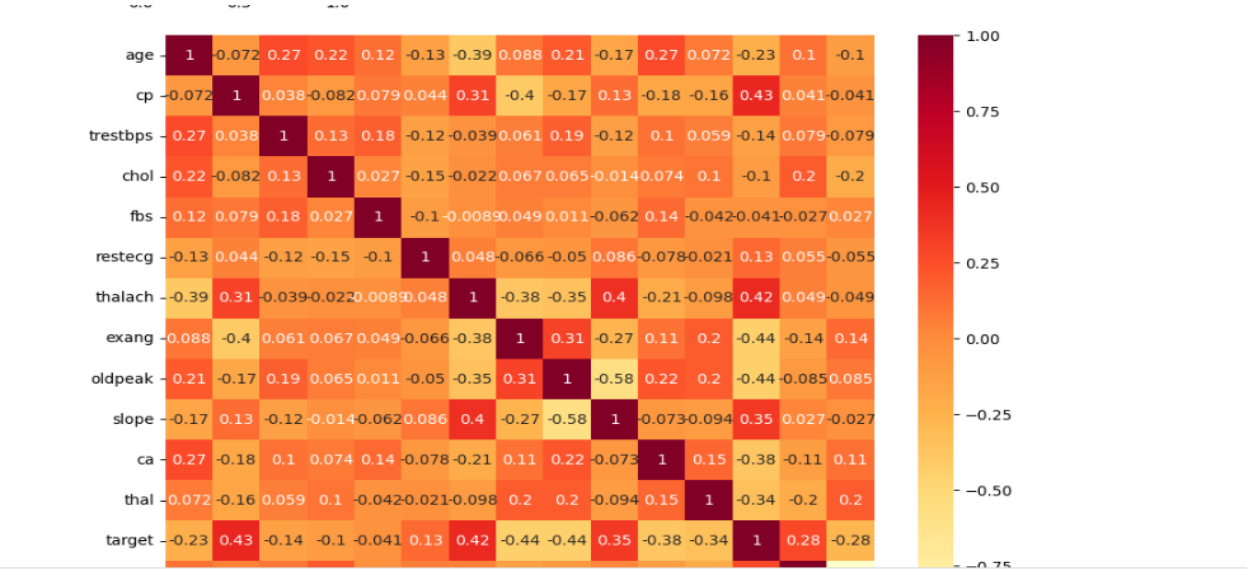
```
[5] import matplotlib.pyplot as plt
    import seaborn as sns

    # Visualize the distribution of the data
    data.hist(figsize=(12, 8))
    plt.show()

    # Visualize the correlations between features and the target variable
    plt.figure(figsize=(10, 8))
    sns.heatmap(data.corr(), annot=True, cmap='YlOrRd')
    plt.show()
```



✓ 0s completed at 2:53 PM

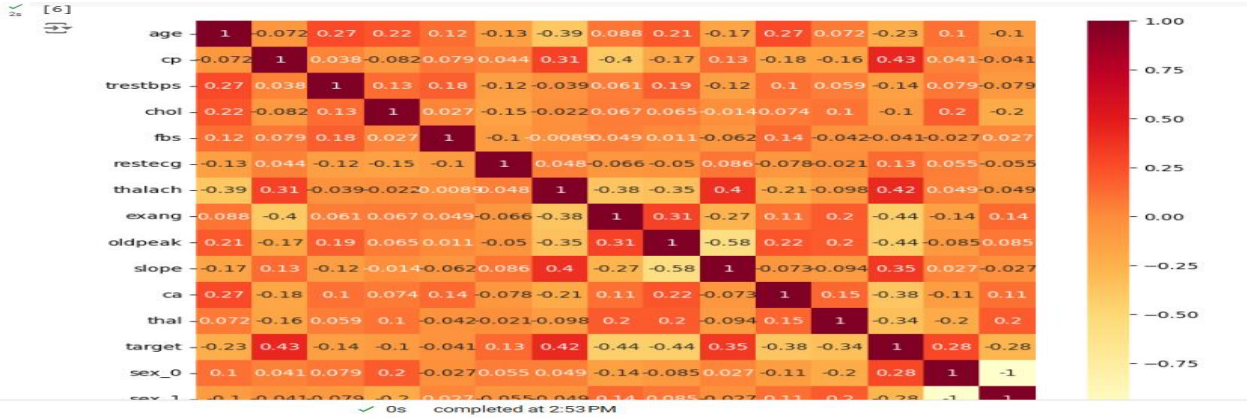


6. **Visualize Correlations**

A correlation matrix is visualized to identify relationships between features and the target variable.

Visualize correlations between features and the target variable:

```
[6]  plt.figure(figsize=(10, 8))
     sns.heatmap(data.corr(), annot=True, cmap='YlOrRd')
     plt.show()
```



7. **Split the Data**

The data is split into training and testing sets using stratified sampling to maintain the distribution of the target variable.

Split the data into training and testing sets

```
[7] from sklearn.model_selection import train_test_split
    X = data.drop('target', axis=1)
    y = data['target']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

8. **Train a Logistic Regression Model**

A logistic regression model with L1 regularization is trained on the training set.

9. **Evaluate the Model**

The model is evaluated on the test set using metrics such as accuracy, precision, recall, F1-score, and ROC-AUC.

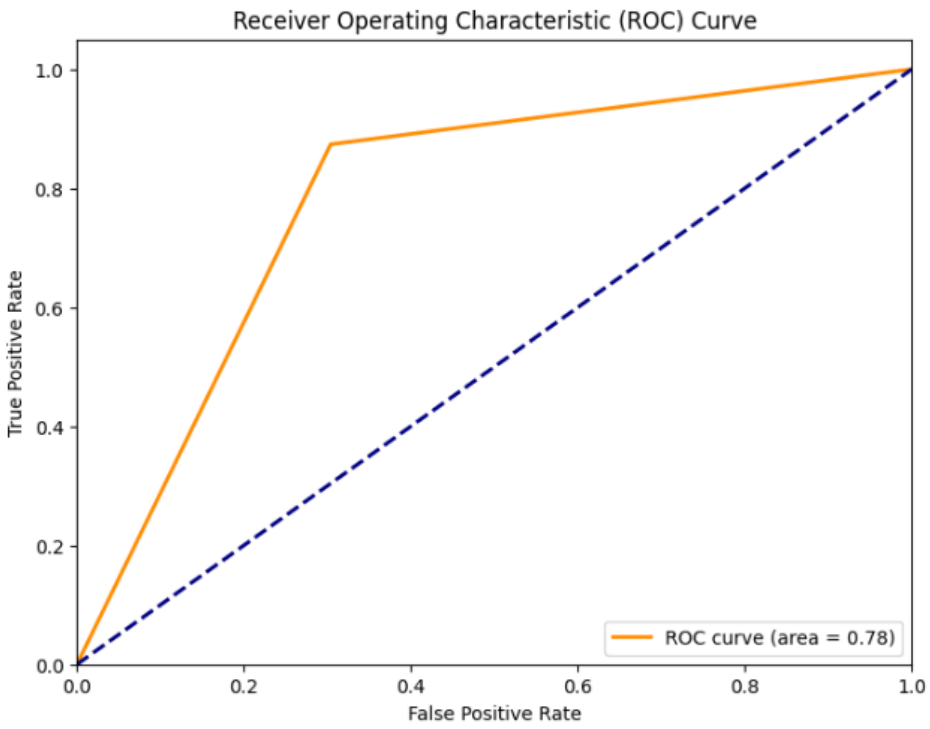
10. **Plot the ROC Curve**

The ROC curve is plotted and the AUC value is calculated to evaluate the model's performance.

Plot the ROC curve and calculate the AUC value:

```
[11] from sklearn.metrics import roc_curve, roc_auc_score
     fpr, tpr, _ = roc_curve(y_test, y_pred)
     plt.figure(figsize=(8, 6))
     plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc_auc_score(y_test, y_pred))
     plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
     plt.xlim([0.0, 1.0])
     plt.ylim([0.0, 1.05])
     plt.xlabel('False Positive Rate')
     plt.ylabel('True Positive Rate')
     plt.title('Receiver Operating Characteristic (ROC) Curve')
     plt.legend(loc="lower right")
     plt.show()
```

[11]



11. **Interpret the Coefficients**

The coefficients of the logistic regression model are interpreted to understand the impact of each feature on the likelihood of having heart disease.

Interpret the coefficients of the logistic regression model:

```
[12] coef = model.coef_[0]
     feature_names = X.columns
     for i, c in enumerate(coef):
         print(f"Feature: {feature_names[i]}, Coefficient: {c:.2f}")
```

```
Feature: age, Coefficient: -0.03
Feature: cp, Coefficient: 0.85
Feature: trestbps, Coefficient: -0.30
Feature: chol, Coefficient: -0.44
Feature: fbs, Coefficient: -0.21
Feature: restecg, Coefficient: 0.23
Feature: thalach, Coefficient: 0.03
Feature: exang, Coefficient: -0.87
Feature: oldpeak, Coefficient: -0.68
Feature: slope, Coefficient: 0.50
Feature: ca, Coefficient: -0.81
Feature: thal, Coefficient: -1.09
Feature: sex_0, Coefficient: 0.50
Feature: sex_1, Coefficient: -1.27
```

12. **Calculate Odds Ratios**

Odds ratios for the features are calculated and explained to interpret the likelihood of having heart disease.

```
[ ] ## Calculate the odds ratios for the features:
```

```
[13] import numpy as np

     odds_ratios = np.exp(coef)
     for i, o in enumerate(odds_ratios):
         print(f"Feature: {feature_names[i]}, Odds Ratio: {o:.2f}")
```

```
Feature: age, Odds Ratio: 0.97
Feature: cp, Odds Ratio: 2.33
Feature: trestbps, Odds Ratio: 0.74
Feature: chol, Odds Ratio: 0.64
Feature: fbs, Odds Ratio: 0.81
Feature: restecg, Odds Ratio: 1.26
Feature: thalach, Odds Ratio: 1.03
Feature: exang, Odds Ratio: 0.42
Feature: oldpeak, Odds Ratio: 0.51
Feature: slope, Odds Ratio: 1.65
Feature: ca, Odds Ratio: 0.44
Feature: thal, Odds Ratio: 0.34
Feature: sex_0, Odds Ratio: 1.64
Feature: sex_1, Odds Ratio: 0.28
```

Conclusion

This documentation provides a detailed step-by-step guide to predicting heart disease using logistic regression. The process involves data preprocessing, model training, evaluation, and interpretation, ensuring a thorough understanding of the dataset and the model's performance.