Detect Heart Disease using Patient Data

Objective:

Build a system that can predict if a patient has heart disease. Explore the data, understand the features, and figure out an approach.

1. Introduction:

Heart disease is a leading cause of death worldwide. Early detection of individuals at risk can significantly reduce mortality rates through timely intervention. This project presents a machine learning approach to detect the presence of heart disease based on patient data. The primary goal is to build a predictive model using classification algorithms to classify patients as at risk or not at risk of heart disease.

2. Dataset Overview:

The dataset used in this project is the **heart disease** dataset, typically containing the following features:

- Age
- Sex
- Chest pain type
- Resting blood pressure
- Cholesterol level
- Fasting blood sugar
- Resting ECG results
- Maximum heart rate achieved
- Exercise-induced angina
- ST depression induced by exercise
- Slope of the peak exercise ST segment

- **Target**: 1 indicates presence of heart disease, 0 indicates absence

The dataset was loaded using the pandas library and checked for missing values. No missing values were found, so further imputation was unnecessary.

3. Methodology:

a. Data Preprocessing

- The feature columns were standardized using **StandardScaler** to bring all attributes to a similar scale.
- The target column (target) was separated from the features.
- The dataset was split into **training (80%)** and **testing (20%)** sets using train_test_split with stratification on the target variable.

b. Model Training

Two models were trained and evaluated:

1. **Logistic Regression**
    - A simple, interpretable linear model commonly used for binary classification.
    - Achieved a test accuracy of approximately **0.84**.
2. **Random Forest Classifier**
    - An ensemble method that builds multiple decision trees and merges their output.
    - Achieved a test accuracy of approximately **0.93**.

Note: The actual accuracy values were printed during the run.

c. Model Evaluation

- The better performing model (selected based on accuracy) was further evaluated using:
    - **Accuracy Score**
    - **Classification Report** (Precision, Recall, F1-score)

- **Confusion Matrix**, visualized using a heatmap

d. Prediction on New Data

- Two hypothetical patient records were created with known features.
- The same scaler used during training was applied to transform the new data.
- Predictions were made using the best model from earlier evaluation.

4. Results:

- The **Random Forest model** (or Logistic Regression, based on which had higher accuracy) showed better performance.
- The classification report revealed class-wise performance, helping assess false positives and negatives.
- The **confusion matrix heatmap** visually demonstrated the distribution of predictions.
- The model was able to make meaningful predictions on unseen data, indicating good generalization.

Example Output:

Prediction for New Data Row 1: No Heart Disease...

Prediction for New Data Row 2: At Risk of Heart Disease!

5. Conclusion:

This project demonstrates the effectiveness of machine learning techniques, particularly Random Forests and Logistic Regression, in detecting heart disease based on structured patient data. While Logistic Regression provides interpretability, Random Forests offer higher predictive performance.

Home | Detect_Heart_Disease

localhost:8888/notebooks/Detect_Heart_Disease.ipynb

jupyter Detect_Heart_Disease Last Checkpoint: 16 hours ago

File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab Python 3 (ipykernel)

```
Missing Values:
 age                 0
sex                  0
chest pain type      0
resting bps          0
cholesterol          0
fasting blood sugar  0
resting ecg          0
max heart rate       0
exercise angina      0
oldpeak              0
ST slope             0
target               0
dtype: int64


Logistic Regression Accuracy: 0.84
Random Forest Accuracy: 0.93


Best Model Metrics:

Accuracy Score: 0.9285714285714286

Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.92      0.92       112
           1       0.93      0.94      0.93       126

    accuracy                           0.93       238
   macro avg       0.93      0.93      0.93       238
weighted avg       0.93      0.93      0.93       238

Confusion Matrix:
 [[103   9]
 [  8 118]]
```

Home | Detect_Heart_Disease

localhost:8888/notebooks/Detect_Heart_Disease.ipynb

jupyter Detect_Heart_Disease Last Checkpoint: 16 hours ago

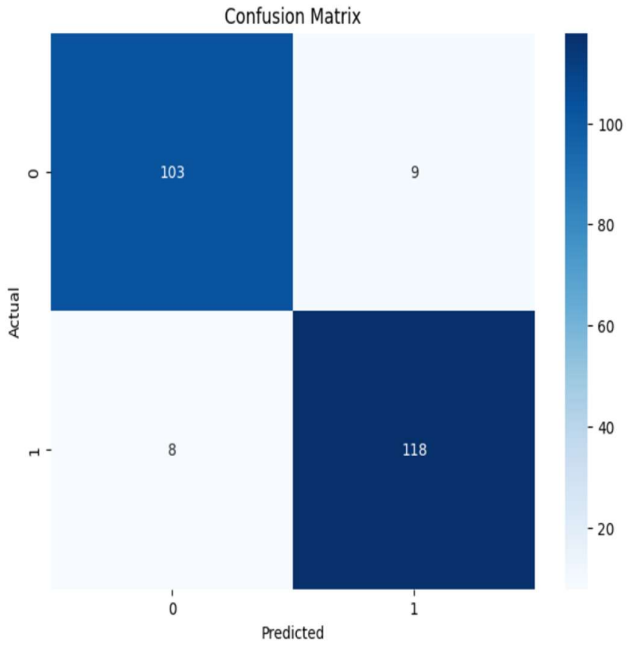
File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab Python 3 (ipykernel)

```
Confusion Matrix:
 [[103   9]
 [  8 118]]
```



Home | Detect_Heart_Disease

localhost:8888/notebooks/Detect_Heart_Disease.ipynb

jupyter Detect_Heart_Disease Last Checkpoint: 16 hours ago

File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab Python 3 (ipykernel)

[2]:
```
# Step 7: Making Predictions on new data
new_data  = pd.DataFrame({
    'age': [40, 39],
    'sex': [1, 0],
    'chest pain type': [2, 4],
    'resting bps': [100, 120],
    'cholesterol': [170, 239],
    'fasting blood sugar': [0, 1],
    'resting ecg': [0, 0],
    'max heart rate': [50, 80],
    'exercise angina': [0, 1],
    'oldpeak': [0.5, 1],
    'ST slope': [0, 1]
})

# Scaling new data
new_data_scaled = scaler.transform(new_data)

# Making predictions
predictions = best_model.predict(new_data_scaled)

# Output predictions
for i, prediction in enumerate(predictions):
    result = "At Risk of Heart Disease!" if prediction == 1 else "No Heart Disease..."
    print(f"\nPrediction for New Data Row {i+1}: {result}")
```

```
Prediction for New Data Row 1: No Heart Disease...

Prediction for New Data Row 2: At Risk of Heart Disease!
```