Heart Disease Prediction - Detailed Documentation

HEART DISEASE PREDICTION - FULL PROJECT DOCUMENTATION

----

1. INTRODUCTION

The Heart Disease Prediction project applies Machine Learning to predict the likelihood of a patient having heart disease based on clinical data.

It uses Pythons Scikit-learn library and logistic regression as the core algorithm, supported by pandas, numpy, and matplotlib for data analysis.

2. DATA COLLECTION AND UNDERSTANDING

The dataset used consists of medical attributes like:

- Age
- Sex
- Chest Pain Type (cp)
- Resting Blood Pressure (trestbps)
- Cholesterol Level (chol)
- Fasting Blood Sugar (fbs)
- Resting Electrocardiographic results (restecg)
- Maximum Heart Rate Achieved (thalach)
- Exercise Induced Angina (exang)
- Oldpeak (ST depression)
- Slope, Ca, Thal, and Target (presence or absence of heart disease)

Data is loaded using pandas:

```
import pandas as pd
df = pd.read_csv('heart.csv')
```

3. DATA PREPROCESSING

Before training the model, preprocessing steps ensure data quality and consistency:

- Handling Missing Values: Dataset is checked for nulls using df.isnull().sum()

- Data Types: Ensured correct numerical formats for all features
- Feature and Target Split:

```
X = df.drop(columns='target', axis=1)
Y = df['target']
```

- Data Standardization:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X)
standardized_data = scaler.transform(X)
```

Standardization is crucial for models like Logistic Regression.

4. DATA SPLITTING

The dataset is divided into training and test sets to evaluate model performance.

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratify=Y, random_state=2)
```

Using stratify=Y ensures proportional representation of target classes.

5. MODEL TRAINING

The Logistic Regression algorithm is used since the problem is binary classification (heart disease present or absent).

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train, Y_train)
```

6. MODEL EVALUATION

Accuracy is calculated to measure model performance:

```
from sklearn.metrics import accuracy_score
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
```

Results are printed to compare train vs test accuracy.

7. MAKING PREDICTIONS

For individual predictions:

```
input_data = (41, 0, 1, 130, 204, 0, 0, 172, 0, 1.4, 2, 0, 2)
input_data_as_numpy_array = np.asarray(input_data)
input_data_reshaped = input_data_as_numpy_array.reshape(1, -1)
prediction = model.predict(input_data_reshaped)
```

If prediction == 0 No heart disease

If prediction == 1 Heart disease detected

8. MODEL DEPLOYMENT CONSIDERATIONS

The model can be integrated into a web app using Flask or Streamlit.

Example Streamlit code snippet:

```
import streamlit as st
st.title('Heart Disease Prediction System')
st.text_input('Age')
st.button('Predict')
```

9. CONCLUSION

This project demonstrates the end-to-end machine learning pipeline:

- Data loading and cleaning
- Feature standardization
- Model training and evaluation
- Predictive inference using Logistic Regression

Accuracy achieved is typically around 8386%, depending on data distribution and preprocessing.

10. FUTURE IMPROVEMENTS

- Try advanced models (Random Forest, SVM, XGBoost)
- Perform hyperparameter tuning
- Include cross-validation
- Add user interface for real-time predictions

AUTHOR

This documentation was prepared by Satyam Gajjar.