Heart Disease Prediction Using Machine Learning

Prepared by: Thiru Sudar S L

Role: Machine Learning Intern

Date: [14.12.2024]

Abstract

This project aims to build a machine learning model to predict the likelihood of heart disease in patients based on their medical and demographic data. Using a dataset of patient records, we explored the data, built models, and evaluated their performance to identify the best approach. The tuned Random Forest model achieved an accuracy of 94.96%, demonstrating its efficacy in predicting heart disease. This report outlines the methodologies, results, and key insights from the project.

Introduction

Heart disease is one of the leading causes of death worldwide. Early detection and diagnosis can save millions of lives. This project leverages machine learning techniques to predict the presence of heart disease using patient data. By analyzing key medical and demographic features, the model aims to assist healthcare professionals in making data-driven decisions.

Objective

To predict whether a patient has heart disease based on their medical data using machine learning.

Dataset Description

The dataset contains information about patient demographics, medical history, and test results. It includes the following features:

- Age, Sex
- Chest pain type
- Resting blood pressure, Cholesterol
- Fasting blood sugar
- Resting ECG results, Maximum heart rate achieved
- Exercise-induced angina

- ST slope, Oldpeak (ST depression)
- Target (0 = No heart disease, 1 = Heart disease)

Methodology

1. Data Exploration: Inspected and understood the dataset.
2. Preprocessing: Normalized numerical features and encoded categorical ones.
3. Model Building: Trained a Logistic Regression model and a Random Forest model.
4. Hyperparameter Tuning: Optimized the Random Forest model using GridSearchCV.
5. Evaluation: Measured accuracy, precision, recall, and F1-score to select the best model.
6. Feature Importance: Visualized the key factors contributing to predictions.
7. Prediction: Tested the model with new patient data.

Results

1. Logistic Regression Accuracy: 86.13%.
2. Tuned Random Forest Accuracy: 94.96%.
   - Best Parameters: {'max_depth': 20, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 300}.
   - Precision and Recall were excellent for both classes.
3. Feature Importance: The key predictors included 'ST slope', 'age', and 'cholesterol'.

Discussion

The results demonstrate that the Random Forest model performs exceptionally well in predicting heart disease, with an accuracy of 94.96%. Key features such as ST slope, cholesterol levels, and age were identified as the most influential predictors. Despite the high accuracy, the model's effectiveness is limited to the quality and diversity of the dataset. Further testing on unseen data is recommended.

Conclusion

This project successfully developed a machine learning model to predict heart disease with high accuracy. The insights gained from feature importance analysis highlight critical factors for heart health. Future work could involve deploying the model as an application or API for real-world use.

References

1. Dataset source: [Provide dataset reference]
2. Scikit-learn documentation: https://scikit-learn.org/
3. Relevant research papers or articles (if any).

Appendices

Code Used in the Project

```
# Importing necessary libraries

import pandas as pd
```

```
# Load the dataset
file_path = 'dataset.csv'  # Update this with the correct path to the dataset
data = pd.read_csv(file_path)

# Display the first few rows to understand the structure
print("Dataset Preview:")
print(data.head())

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

# Display basic statistics
print("\nDataset Description:")
print(data.describe())

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Separate features and target variable
X = data.drop(columns=["target"])
y = data["target"]

# Normalize numeric columns
scaler = StandardScaler()
```

```
X_scaled = scaler.fit_transform(X)

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

print("Training set size:", X_train.shape)
print("Testing set size:", X_test.shape)

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Initialize and train the model
model = LogisticRegression()
model.fit(X_train, y_train)

# Predict on test data
y_pred = model.predict(X_test)

# Evaluate the model
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))

#using random forest method
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

```
# Initialize the Random Forest model
rf_model = RandomForestClassifier(random_state=42)

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

# Predict on the test set
rf_y_pred = rf_model.predict(X_test)

# Evaluate the model
print("Accuracy:", accuracy_score(y_test, rf_y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, rf_y_pred))
print("\nClassification Report:\n", classification_report(y_test, rf_y_pred))

from sklearn.model_selection import GridSearchCV

# Define the parameter grid
param_grid = {
   'n_estimators': [100, 200, 300],
   'max_depth': [10, 20, None],
   'min_samples_split': [2, 5, 10],
   'min_samples_leaf': [1, 2, 4]
}

# Initialize the GridSearchCV
```

```
grid_search = GridSearchCV(estimator=RandomForestClassifier(random_state=42),
                           param_grid=param_grid,
                           cv=5,
                           scoring='accuracy',
                           verbose=2,
                           n_jobs=-1)

# Perform the grid search
grid_search.fit(X_train, y_train)

# Best parameters and evaluation
print("Best Parameters:", grid_search.best_params_)
best_rf_model = grid_search.best_estimator_

# Predict with the tuned model
tuned_y_pred = best_rf_model.predict(X_test)

# Evaluate the tuned model
print("\nTuned Model Accuracy:", accuracy_score(y_test, tuned_y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, tuned_y_pred))
print("\nClassification Report:\n", classification_report(y_test, tuned_y_pred))

import matplotlib.pyplot as plt
import numpy as np

# Get feature importance
```

```
importances = best_rf_model.feature_importances_
feature_names = data.columns[:-1]  # Exclude 'target'
indices = np.argsort(importances)[::-1]

# Plot the feature importance
plt.figure(figsize=(10, 6))
plt.title("Feature Importance")
plt.bar(range(X.shape[1]), importances[indices], align="center")
plt.xticks(range(X.shape[1]), feature_names[indices], rotation=45)
plt.xlabel("Feature")
plt.ylabel("Importance Score")
plt.tight_layout()
plt.show()

import numpy as np
from sklearn.preprocessing import StandardScaler

# New patient's data (features)
new_patient_data = np.array([55, 1, 2, 140, 255, 0, 1, 160, 0, 1.2, 2])

# Reshape data to match the number of features in training data
new_patient_data = new_patient_data.reshape(1, -1)

# Scale the data using the same scaler used during training
new_patient_data_scaled = scaler.transform(new_patient_data)
```

```
# Make a prediction using the trained Random Forest model
prediction = rf_model.predict(new_patient_data_scaled)

# Output the prediction
if prediction == 1:
    print("The patient is likely to have heart disease.")
else:
    print("The patient is likely to be heart disease free.")
```