Artificial Intelligence Lab_5

Turan Abbasli, Rashad Mammadov

March 24, 2024

Abstract

This report presents the analysis and evaluation of two machine learning models, Multi-Layer Perceptron (MLP) and Decision Tree (DT), for predicting the risk of heart disease in patients. The dataset used for this analysis contains 14 features, including both categorical and numerical attributes. The models were trained using data preprocessing techniques such as One-Hot encoding and standardization. Evaluation metrics such as precision, accuracy, sensitivity, and specificity were used to assess the performance of the models. The results show that the MLP model outperformed the DT model in terms of accuracy, sensitivity, and specificity, making it a preferable choice for predicting the risk of heart disease.

Contents

1 Introduction

This study compares the performance of two models, Multi-Layer Perceptron (MLP) and Decision Tree (DT), in predicting heart disease risk. The dataset used contains 14 features, including demographic and medical attributes. Preprocessing steps such as OneHot encoding and standardization were applied to the data before training the models. The MLP model, with its three-layer structure and ReLU activation function, outperformed the DT model in accuracy, sensitivity, and specificity, making it a more suitable choice for predicting heart disease risk.

2 Methodology

2.1 Data Preprocessing

The heart disease dataset is imported using pandas. It consists of 14 features, where 13 of them are X features and the attribute we want to predict is denoted as *target*. Since the *target* attribute has boolean values, the prediction task is binary classification. The attributes *sex, chest_pain_type, resting_blood_pressure, fasting_blood_sugar, rest_ecg, exercise_induced_angina, st_slope,* and *thalassemia* have numerical values representing categories, so these features are considered categorical, while the rest are considered numerical attributes.

Since the values of these categorical attributes have no comparative relation between each other, *OneHot* encoding is used.

Before feeding data to the *Multi-Layer Perceptron (MLP)*, it is necessary to normalize them, as the units of the features in the data are different. To preserve the variance of the dataset, the *standardization* method is used, as shown in Equation 1, where X is the original value, μ is the mean, and σ is the standard deviation.

$$Standardized\_value = \frac{X - \mu}{\sigma} \tag{1}$$

Standard scaling uses the standard deviation and mean of the whole dataset to scale values, so before scaling, the data must be separated into train and test sets. Otherwise, the values in the train dataset will be affected by those in the test dataset.

2.2 Multi-Layer Perceptron (MLP)

The structure of the MLP consists of three layers: the input layer, hidden layer, and output layer. After OneHot encoding, the number of features in the X data became 23. Therefore, the same number of neurons is used in the input layer, 5 neurons in the hidden layer, and 1 neuron in the output layer, as the task is binary classification as shown in Figure 1.

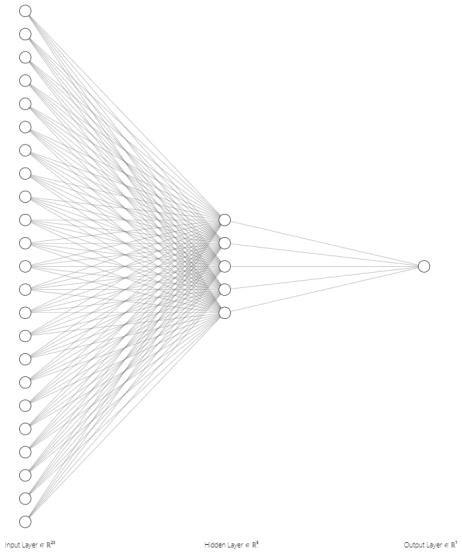

Figure 1: MLP structure

The hidden layer uses the $ReLU$ activation function, and the output layer uses the *Sigmoid* activation function.

The $ReLU$ (Rectified Linear Unit) activation function is defined as:

$$f(x) = \max(0, x) \tag{2}$$

The *Sigmoid* activation function is defined as:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3}$$

In Equation 2, x represents the input to the $ReLU$ function, and in Equation 3, x represents the input to the *Sigmoid* function.

Matrix Dimensions

The dimensions of the matrices used to represent the model are as follows:

- Inputs: $23 \times m$ (where m is the number of examples in a batch)
- Parameters:
  - $W1$: 5×23 (weights for hidden layer)
  - $b1$: 5×1 (biases for hidden layer)
  - $W2$: 1×5 (weights for output layer)
  - $b2$: 1×1 (biases for output layer)
- Outputs: $1 \times m$ (predictions for each example in a batch)

Mini-Batch Training

To integrate the concept of mini-batch training, the `train_epoch` method can be modified to iterate over mini-batches of the training data. Here's the updated method:

```
def train_epoch(self, X_train, y_train, learning_rate, batch_size):
    permutation = np.random.permutation(X_train.shape[1])
    X_train_shuffled = X_train[:, permutation]
    y_train_shuffled = y_train[permutation]

    for i in range(0, X_train.shape[1], batch_size):
        X_batch = X_train_shuffled[:, i:i+batch_size]
        y_batch = y_train_shuffled[i:i+batch_size]

        Z1, Z2, A1, A2 = self.feed_forward(X_batch)
        self.back_propagation(Z1, A1, A2, X_batch, y_batch, learning_rate)
```

In this method, the training data is shuffled and then divided into mini-batches of size `batch_size`. The `feed_forward` and `back_propagation` methods are then called for each mini-batch to update the weights and biases of the model based on the mini-batch.

Forward and Backward propagation

The model's loss is determined by the categorical cross entropy loss. During each epoch, the data is fed into the model, predictions are made, and the loss is calculated based on those predictions. Using gradient descent, the weight and bias matrices of the Neural Network are updated.

For training, a mini-batch size of 4 is used, and the learning rate is set to 0.01. The model continues the learning process until the average loss of 10 consecutive training epochs becomes higher than the previous one.

2.3 Decision Tree

This code implements a decision tree classifier using the CART (Classification and Regression Trees) algorithm. This implementation uses entropy and information gain to determine the optimal splits while growing the decision tree, ultimately leading to a classifier capable of making predictions based on input feature vectors. Random feature selection and a few hyperparameters like minimum samples per split and maximum depth are also incorporated to control the tree's growth and prevent overfitting.

Building the Tree

The decision tree is built recursively by selecting the best feature and threshold to split the data at each node. This is done until a stopping criterion is met, such as reaching the maximum depth of the tree, having only one class in the node, or having fewer samples than the minimum required for splitting.

Entropy

Entropy is a measure of impurity or randomness in a set of labels. In this implementation, entropy is calculated using the formula:

$$Entropy = -\sum_{i=1}^{n} p_i \log_2(p_i) \tag{4}$$

where p_i is the proportion of instances in class i and n is the number of classes.

Information Gain

Information gain is used to decide the best feature and threshold to split the data at each node. It measures the reduction in entropy achieved by splitting the data based on a particular feature and threshold. Information gain is calculated using the formula:

$$IG = Entropy(parent) - \left(\frac{N_{left}}{N} \times Entropy(left) + \frac{N_{right}}{N} \times Entropy(right) \right) \tag{5}$$

where $Entropy(parent)$ is the entropy of the parent node before the split, N_{left} and N_{right} are the number of instances in the left and right child nodes after the split, N is the total number of instances in the parent node, $Entropy(left)$ and $Entropy(right)$ are the entropies of the left and right child nodes, respectively.

3 Results

To evaluate the each model, several metrics such as *precision*, *accuracy*, *sensitivity*, and *specificity* are used.

1. Precision:

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

2. Accuracy:

$$Accuracy = \frac{TruePositives + TrueNegatives}{TotalExamples}$$

3. Sensitivity (Recall):

$$Sensitivity = \frac{TruePositives}{TruePositives + FalseNegatives}$$

4. Specificity:

$$Specificity = \frac{TrueNegatives}{TrueNegatives + FalsePositives}$$

3.1 Evaluation of MLP

The confusion matrix of the MLP model reveals the following values, 2:

- The precision of the MLP model is 0.853.
- The accuracy of the MLP model is 0.85.
- The sensitivity of the MLP model is 0.879.
- The specificity of the MLP model is 1.

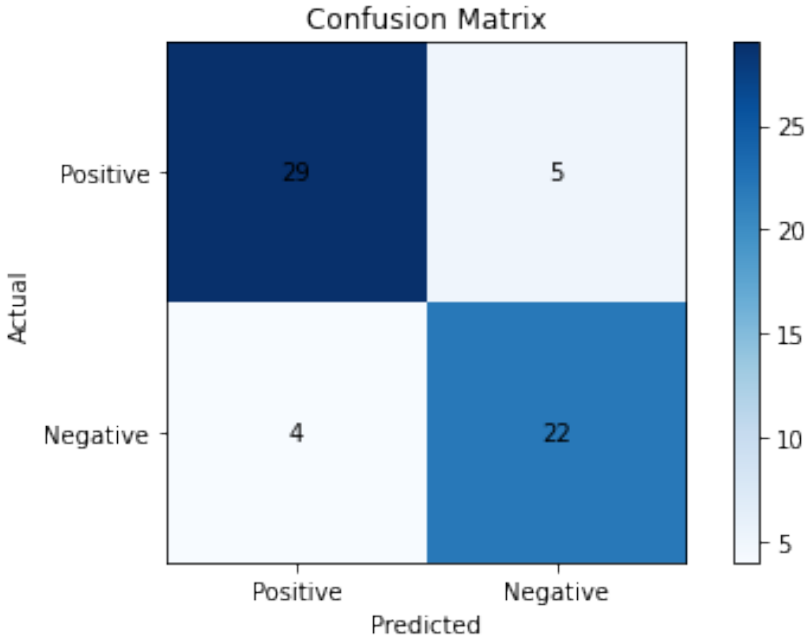

Figure 2: Confusion matrix of MLP

3.2 Evaluation of Decition Tree (DT)

The confusion matrix of the decision tree model reveals the following values, 3:

- The precision of the DT model is 0.889.
- The accuracy of the DT model is 0.82.
- The sensitivity of the DT model is 0.75.
- The specificity of the DT model is 1.

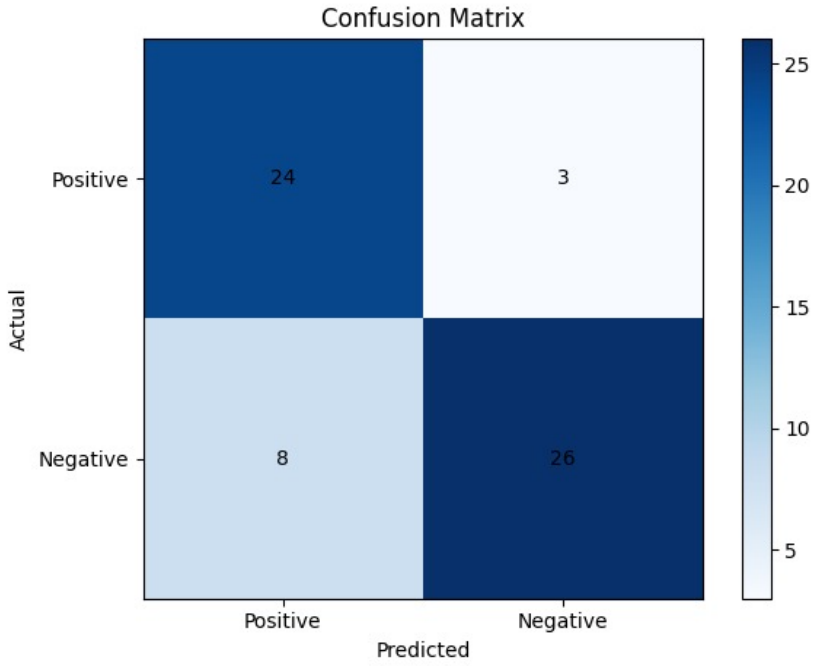

Figure 3: Confusion matrix of Decision Tree

4 Conclusion

In the case of predicting the risk of heart disease in patients, it is generally more important for the model to be sensitive rather than specific.

Sensitivity (also known as recall) measures the proportion of actual positive cases that are correctly identified by the model. In the context of predicting heart disease risk, high sensitivity means that the model is able to correctly identify a high percentage of patients who actually have the disease, which is crucial for early detection and intervention.

On the other hand, *specificity* measures the proportion of actual negative cases that are correctly identified by the model. While specificity is also important, especially for avoiding false alarms and unnecessary treatments, in the context of heart disease prediction, it may be more acceptable to have a slightly lower specificity if it means achieving higher sensitivity and catching more cases of the disease early.

Based on evaluation metrices, the MLP model has slightly lower precision compared to the DT model, but it has higher accuracy, sensitivity, and specificity. The MLP model's higher sensitivity indicates that it is better at correctly identifying patients with heart disease, which is crucial for early detection and treatment.

Considering those cases, MLP model is much more preferable to predict the risk of heart disease in patients due to its overall better performance across the evaluation metrics.