Heart Disease

Jamila Seyidova, Kamal Aghayev, Amin Azimov
{jamila.seyidova[*], kamal.aghayev[†], amin.azimov[†] }@ufaz.az

December 8, 2019

1 Introduction

Multilayer Perception or **Decision Tree**? The goal of this Practical Project is to compare both models and determine which one is more suitable to use in order to predict the risk of heart disease of patients. We can come to a decision using such metrics as Accuracy, Precision, Recall (Sensitivity), Specificity, F1-Score, Mean Squared Error. It is very important to choose the right model because we want to decrease error, especially in cases when sick patients diagnosed as healthy.

2 Dataset

2.1 Description

1. What is the name of the attribute we want to predict?
   - The name of the attribute that we want to predict is **target** which must show whether certain patient is **sick** or **not**.
2. Is that a binary or a multi-class classification?
   - It is a binary classification, because it shows 2 possible outcomes **0** for **not sick** and **1** for **sick**.
3. Which attributes are categorical?
   - Categorical (non-numeric) attributes in heart disease dataset are:
     - `sex`,
     - `chest_pain_type`,
     - `fasting_ blood_ sugar`,
     - `rest_ ecg`,
     - `exercise_induced_angina`,
     - `st_slope`,
     - `num_major_blood_vessels`,
     - `thalassemia`.
4. How can we encode categorical attributes ?
   - **One-hot encoding.** This is the type of encoding, when for each value of an attribute we create a new attribute which takes either **0** or **1**, i.e. a binary attribute. The reason to use one-hot encoding is that in some cases attributes may not be represented just by number, since some of the values will be greater than others; however, in reality the attribute is a multi class attribute.

[*]Computer Science 1
[†]Computer Science 2

For example, let's consider the case of a `sex` column. In the categorical form males may be represented by 0, while females may be represented by 1. In this case, we let the model know, that women "weight more" than men; however, there is no such dependency between men and women.

2.2 Attribute normalization

1. Which normalization method will be best adapted so we can preserve the variance of the dataset?

   - There are 2 types of normalization that can be used:
     - **Scaling.** This is type of normalization, when data is scaled between two values. For example, we may scale the values of the dataset between 0 and 1 and thus help the model, since there will not be very big gap between data points, when there is a difference of several orders of magnitude between two data points. The formula of scaling is as follows:

$$X' = \frac{X - min(X)}{max(X) - min(X)}$$

     - **Standardization**. This is type of normalization, when data is normalized in such a way that mean value becomes 0 and standard deviation becomes 1. This means that all the data is gathered around 0, while the deviation of the dataset is saved. The formula of standardization is as follows:

$$Z = \frac{X - \mu}{\sigma}$$

       where μ is the mean or expected value of the dataset and σ is the standard deviation of the dataset.

2. Will you normalize the data before or after splitting the dataset in training/testing datasets?

   - The idea of the model is that it is trained on the training dataset, and then, when new data is fed into the model, it must predict whether a patient has a heart disease or not. Normalizing the data before splitting it to training and testing datasets means that we bias the model with the testing dataset, and in the future, when we test the dataset, the tests will not show true values for metrics which is not acceptable. So, it is not correct to normalize before splitting and we must **split and normalize after**.

3. Implement a method/function to normalize the attribute using the adequate normalization method.

```python
def standardize(data, s=None, m=None):
"""
Args:
    data: numpy.array
    s=None: float, standard deviation of the data by default
    m=None: float, mean value of the data by default
Returns:
    (normalized_data, s, m)
    normalized_data: numpy.array
    s, m: numpy.float64
"""
if s is None:
    s = std(data)
if m is None:
    m = mean(data)
return (data - m) / s, s, m
```

3 Multilayer Perceptron (MLP)

3.1 Network architecture

1. What are the dimensions of the matrices you will use to represent your model (inputs, parameters and outputs)? How will you integrate the concept of mini-batch training?

   - Since we applied one-hot encoding on the categorical attributes, number of input nodes of our model increased to 29 +1 bias node. There are 5 nodes +1 bias node in the hidden layer as it was in requirements and there is a node in the output layer representing whether a patient is sick or not. Since we use sigmoid as an activation function of our model, the output node is in the range $[0, 1]$ and by rounding the output we say that a patient is **sick** if the value is 1 and **not sick** if the value is 0.
   - Since we use a fully connected neural network, the dimension of weights between two consecutive layers is $N_i \times N_{i+1}$, where N_i is the number of nodes in layer number i, and N_{i+1} is the number of nodes in the layer number $i + 1$. Thus, the dimensions of the weight matrix between input and hidden layers is 5×29 and the dimensions of the weight matrix between hidden and output layers is 29×1.
   - There are as well weights for bias nodes and the bias weights
   - To integrate the concept of mini-batch training we may use temporary variables for the difference matrices that represent the changes that we need to do to train the model. These temporary variables are applied to the weight matrices and reset to zero matrices after each mini-batch. This way, we can train the exact same model on the data points of the mini-batch and after each mini-batch we update the weights of our model and continue our training.

2. How should you check whether or not you should keep training your model?

   - To determine the point of the end of the training process we may use early stopping. Early stopping is used to identify whether the training process must be stopped. We may stop our training when the loss of the model starts increasing on the testing dataset.
   - Another way of determining the early stopping may be determining the number of epochs after which the loss of the model starts increasing. To do this we may use K-Fold Cross Validation which is a technique to determine the values of different hyperparameters as early stopping, learning rate and etc.

3. Draw your network.

   - See Fig. 1

3.2 Evaluation

The model was evaluated using the following metrics and obtained the following results:

Accuracy	0.875
Precision	0.869
Recall (Sensitivity)	0.901
Specificity	0.845
F1-Score	0.884
Mean Squared Error	0.099

Table 1: Evaluation of MLP

Since our model diagnoses the sickness, the model must be as **sensitive** as possible, while remaining accurate. If the model is not sensitive, we may have problems with people diagnosed as healthy, while they must be treated.

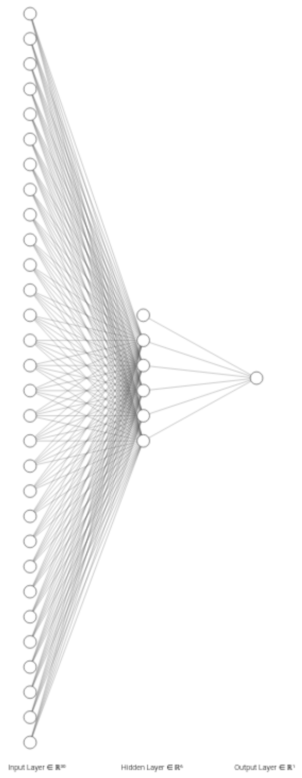

Figure 1: MLP architecture

4 Decision Tree (DT)

4.1 Description

There are different implementations of decision tree algorithm, but all of them have the same idea: decision tree is trained on the data by creating set of rules, when the training process ends, the model may predict the output of data using the rules created during the training process. Before the training process, our model preprocesses the attributes of the data by dividing them into several groups (hyperparameter) by doing the following:

- For the first group, model looks at the value of the $\frac{N}{M}$ $^{\text{th}}$ element of the attribute and this value becomes the upper-bound for the first group.
- For the second group, model looks at the value of the $2\frac{N}{M}$ $^{\text{th}}$ element of the attribute and this value becomes the upper-bound for the second group and the $\frac{N}{M}$ $^{\text{th}}$ element of the attribute becomes the lower-bound of it.

where N is the fraction of the number of data points by the number of groups and M is the number of elements in the dataset. The same process follows for all groups and in the end we have n groups for each attribute. During the training process, the model determines the discriminative power of all the attributes, and the attribute with the biggest discriminative power becomes the one that divides the tree into branches. During this process recursively we obtain a trained tree. During the prediction process, the data is fed into the model, model looks into its rules and determines the branch that is associated to the data point until we arrive to the leaf of a tree.

4.2 Evaluation

The model was evaluated using the following metrics and obtained the following results:

Accuracy	0.703
Precision	0.808
Recall (Sensitivity)	0.632
Specificity	0.8
F1-Score	0.709

Table 2: Evaluation of DT

5 Comparison of MLP and DT

As follows from the evaluation of the models, MLP performs better using all different metrics. Thus it is better to use MLP in this particular case. However, we are lucky to have a simplistic dataset that has a target that may be easily predicted by looking at other parameters. With a more difficult dataset we would probably fail with MLP, due to the number of data points (303 samples). MLP requires a lot of data samples to learn which is one of its main problems. On the other hand, DT does not require a lof of data points and is usually effective with small amount of data. If the models performed in a similar way, we would first look at the recall (sensitivity) metrics. The recall is extremely important in medical models, since the sick person diagnosed healthy may lead to the death of a patient.