AI Project Report by Javid Mammadli,Elmar Hajizada and Leyla Baghirova

1. INTRODUCTION

1.1. What does this program do?

In this project our aim is to predict the risk of heart disease in patients. We will apply Multilayer Perceptron and Decision tree algorithms on the Cleveland Heart Disease dataset in order to reach our aim.

About Data

This dataset contains 303 instances and each instance is described by 14 attributes. Each of these attributes are phylosiological measurements.This is binary classification problem as our target has two categorical value : 1 - yes (patient is sick) 0 - no (patient is healthy)

The name of the attribute that we are going to predict is (target) which consist of from two categorical values and is considered as a binary classification problem as we said before.

In our dataset we have both continuous and categorical attributes . Our categorical attributes: chest_pain_type,fasting_blood_sugar,rest_ecg , exercise_induced_angina, st_slope , num_major_vessels ,thalassemia

By analysing these attributes we can see that each of these attributes has a discrete amount of categorical values. None of them has continuous values .

Before going into training section we have to normalize our dataset since gradient descent will be effective with normalized(scaled) values . If we would not normalize the data we may have values in different scales .For instance cholesterol level varies a lot in between different age groups . So it will be difficult and time consuming to train on this data . In order to adjust weights our model would take more time to train on this data but if we normalize our data by using normalization techniques we will have numbers in the same scale which will make our model train much faster and gradient descent will be effective in this case.

In order to normalize our data we can use standartization or normalization techniques . But in order to keep distribution of data we will use standartization . Standardized values are useful for tracking data that is otherwise incomparable because of different metrics or circumstances. Since normalization does not keep variance and distribution of the data. Even If we would have outliers we would not be able to keep them by normalization techniques . By standartizing our data we will both keep our distribution of data and also we will gather all our data points around the mean = 0 and our standart deviation will be 1 We will standartize our data before splitting . Then we will divide our data into training and testing sets.

2. Decision Tree Algorithm

How does Decision Tree works ? Decision tree is a type of supervised learning algorithm (having a pre-defined target variable) that is mostly used in classification problems.It works

for both categorical and continuous input and output variables. In this technique, we split the population or sample into two or more homogeneous sets (or sub-populations) based on most significant splitter / differentiator in input variables.

General rules of building a tree
1) Pick the best attribute/feature.
2) The best attribute is one which best splits or separates the data.
3) Ask the relevant question.
4) Follow the answer path.
5) Go to step 1 until you arrive to the answer.

First of all,in order to build efficient tree algorithm for predicting correct output we should analyze our data,how many features,how many categorical(dummy values) and etc exist in our data. Secondly,for finding best features for our tree,we should work with entropies and discriminative powers with the following formulas.

Entropy: - $\sum p(x) * log(p(x))$

Discriminative power =entropy(parent)-[weight average]*entropy(children)

1) In our data,we have attribute like an age,in order split it,we found best middle in order to separate it in two parts.This approach also was used for other attributes as max_heart_rate_achieved,st_depression

2) After that,we found best potential splits in order to decide which part will be left or right node of our tree.

3) While building our tree with nodes,we determine best splits and based on this column and value,we separate our data into two parts(left child and right child of node in a tree).

Recursive part

In recursive part,in each level of tree,we are repeating above explained approaches in order to build our tree.When we there is no question to ask,we set current node as a leaf node.

Main tree attributes
Attribute with the highest information gain was(thalassemia),second most-(num_major_vessels),third-(max_heart_rate_achieved), next-(st_depression) and etc.

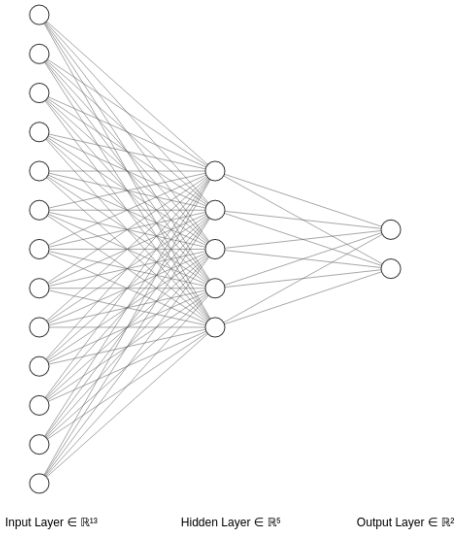
Result With depth 4 we could achieve only 80-82% of accuracy.It can be caused because of having lot of continuous features which requires split data several times.

```
>>> runfile('/home/javid/Desktop/pycharm/ai_project/r
{'thalassemia <= 2.5': [{'num_major_vessels <= 0.5':
accuracy = 80.99%
```

3. Multilayer Perceptron

Lets answer questions about MLP: Our input shape will be the matrix with the shape of (batch_size , number_of_attributes - 1) , because of one hidden layer we will have two weights matrices , one weight will be between input nodes and hidden layers node and another weight will be for interconnection between hidden layer and output layer.

Our network shape will be in this format:

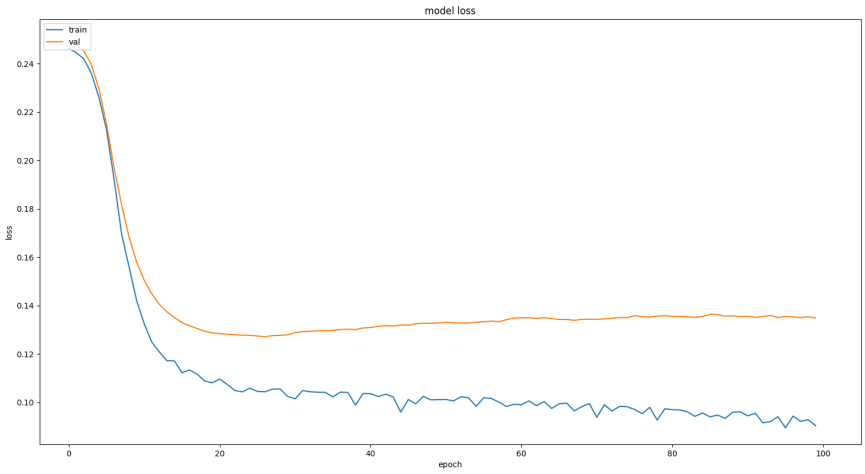


Thus first weight will be a matrix with the shape of (number_of_features - 1 , number_of_neuron_in_hidden) and another weight will be a matrix with the form of (number_of_neuron_in_hidden , number_of_classes) and output layer will be a matrix with the form of (batch_size , number_of_classes)

We will apply mini-batch training technique on our dataset . We are supposed to use batch size as 4 . So our train matrix will be in shape of (batch_size , number_of_neuron_in_hidden).We will load the data instance from our dataset as a number of batch size and we will start to train on that batch , morever we will do forward and backpropogation on that mini-batch . Morever , after finishing that mini-batch training we will take another mini batch from our dataset and we will go in this order to train and update our parametrs once we see all the instances we will finish one epoch of train

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

We can look up to how our test error changes averagely after each 10 epoch.If it is increasing we should stop because we are going to overfit in this case ,so this stopping is called early stopping. For example ,as we see below we had a situation where we had to stop in order to keep our weight and prevent increasing of our test error.So approximately in 20 th epoch we have early stopping.



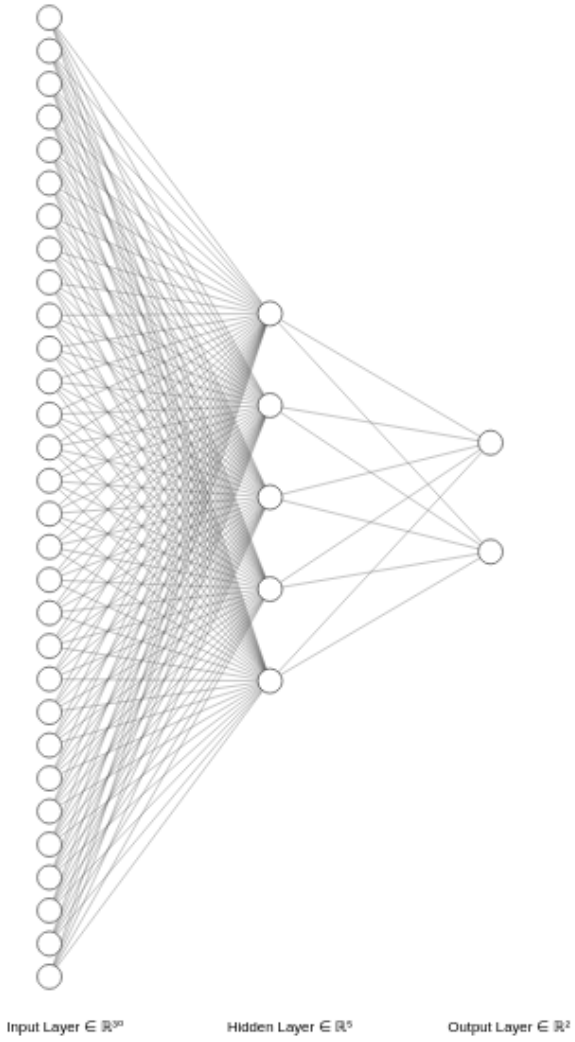
We also evaluated our model based different metrics such as accuracy , recall , specificity , and f1 score. Our model will be sensitive(recall) .Because we know that in this type of example we should avoid from False Negative . False Negative means when patient has disease and we say that you are healthy , so we are predicting that he is not sick but in reality he is sick . There is another situation like False Positive . When we are predicting that he is sick but in reality he is not sick . So if we compare these two situations we can say that the first situation is much more dangerous since if patient is sick and we say that you are healthy , he is in danger . So we should avoid from False Negative .

Our model trains very well since we have standardized our dataset.By this technique our accuracy become very good but we rarely could achived accuracy with 90% and over.

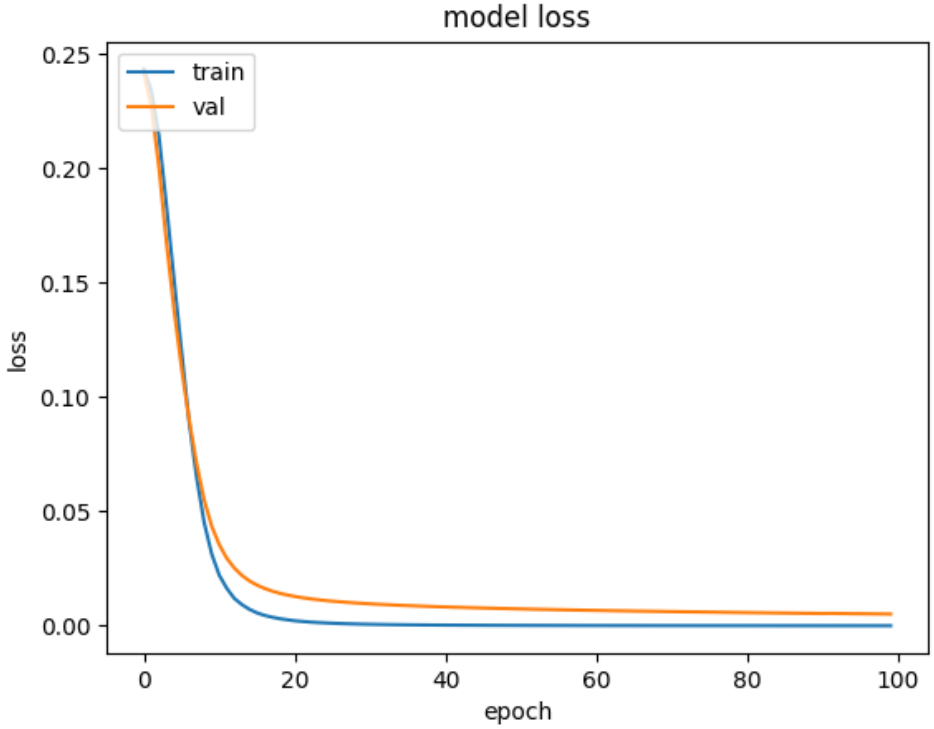
```
elmar@elikjudo:~/Desktop/AI/poject_heart_disease/mlp$ /
Accuracy    : 0.8552631578947368
Recall      : 0.8055555555555556
Precision   : 0.8787878787878788
Specificity :0.9
F1 score    : 0.8405797101449275
```

Then we decided to make our attributes , which are categorical , in an one hot form so that our network can train much better and better . So it is not suprise that we achieved the accuracy of almost 100% and sometimes even 100% itself in this dataset. Since there is no noise and this dataset is not messy , we are not surprised that we achieved to this stage. After making one hot encoding we have reached the input shape of 30 as we had 13 attributes where 7 of them were categorical and after making one hot encoding we have 30 input neurons.

After one hot encoding of categorical attributes:




```
elmar@elikjudo:~/Desktop/AI/poject_heart_disease/mlp$
Accuracy    : 0.9868421052631579
Recall      : 0.9761904761904762
Precision   : 1.0
Specificity :1.0
F1 score    : 0.9879518072289156
```


model loss
train
val
loss
epoch
0.25
0.20
0.15
0.10
0.05
0.00
0
20
40
60
80
100