Hellenic Mediterranean University
Department of Informatics Engineering

Artificial Neural Networks based on Haberman’s Survival dataset

Final project to course Artificial Neural Networks (Spring 2020)

Imrich Kaščák
er1602@edu.hmu.gr

Heraklion 2020

Introduction

Classification is a task that is often found in everyday life. A classification process involves assigning objects into predefined groups or classes based on a number of observed attributes related to those objects. Although there are some more traditional tools for classification, such as certain statistical procedures, neural networks have shown to be an effective solution for this type of problems. There is a number of advantages to using neural network – they are data driven, self-adaptive, can approximate any function – linear as well as non-linear (which is quite important in this case because groups often cannot be divided by linear functions). Neural networks classify objects rather simply – they taka data as input, derive rules based on those data and make decisions.

In this project is shown how we can use neural networks with problems of classification. There is constructed a multilayer feed forward neural network with several architecture which are trained to predict whether a patient survived after breast cancer surgery when it is given other attributes as input. There is determined which architecture gives the best solution to the problem and which does not. At the end, the results are compared and displayed. There is also created a Kohonen network to provide unsupervised learning to that task and results are presented by *Python*'s library *matplotlib*.

Dataset visualization

In this project were used data from the Haberman’s survival data set which contains cases from a study that was conducted between 1958 and 1970 at the University of Chicago’s Billings Hospital of the survival of patients who had undergone surgery for breast cancer. This dataset contains 306 cases with these attributes:

1. age of patient at time of operation,
2. patient’s year of operation (year — 1900),
3. number of positive auxiliary nodes detected,
4. survival status (class attribute):

   1 = the patient survived 5 years or longer; 2 = the patient died within 5 years.

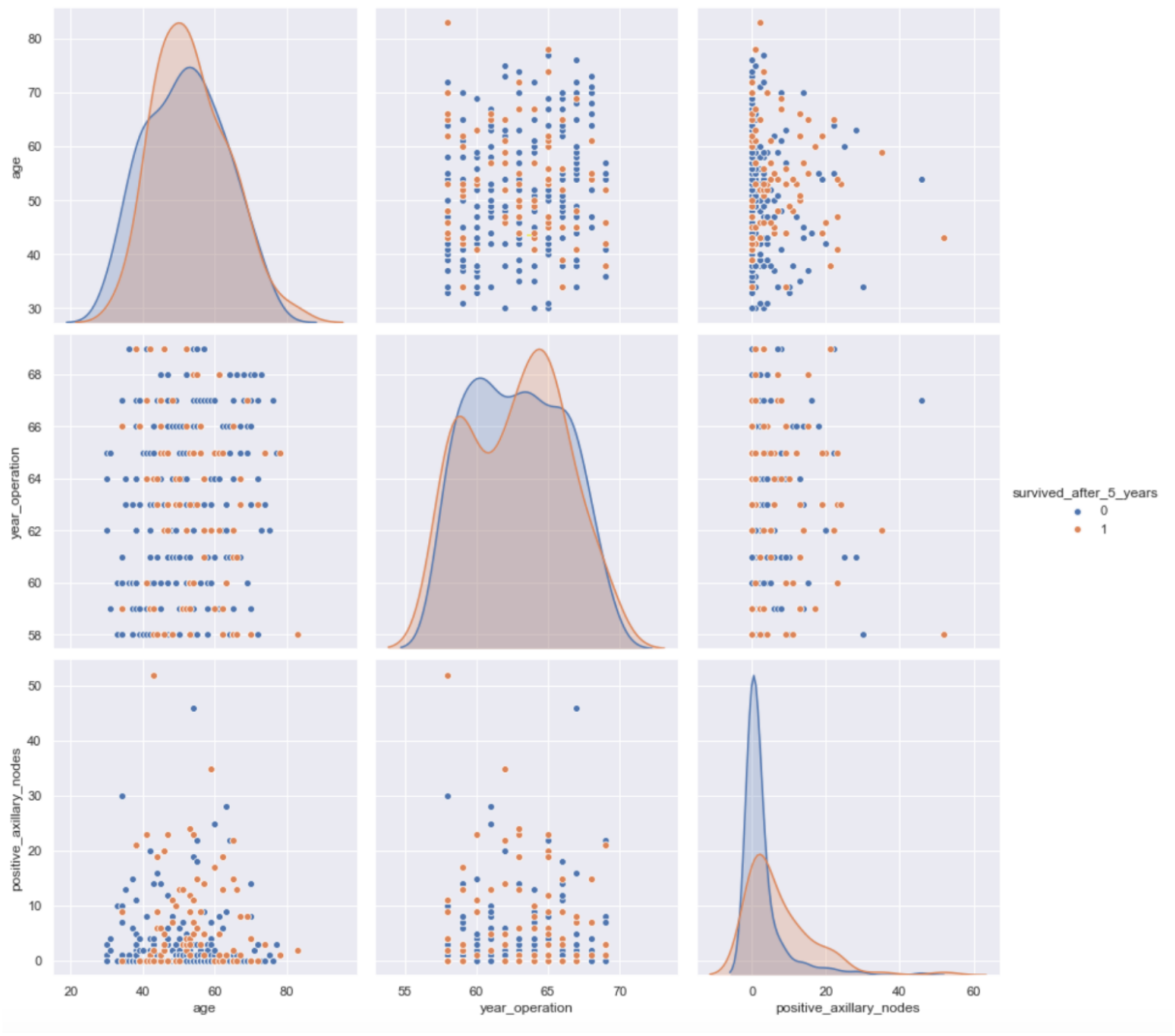

Figure 1: Dataset visualization

Visualization of these samples is on the Figure 1, using a scatter plot using seaborn packages methods. On this figure we can see how exact attributes are related to each other. The legend is the attribute survival status: 0 means that patient survived 5 years or longer, 1 means that patient died within 5 years (see further data pre-processing).

Dataset pre-processing

All information from this dataset is needed so there wasn’t necessity to remove some values. There were also no outliers in this dataset, no missing values, therefore replacing specific values with some mean values is not needed. The only changes to this dataset were made by mapping class attribute values from 1 to 0 (the patient survived 5 years or longer) and from 2 to 1 (the patient died within 5 years) to have these values within range [0,1].

Before creating and training neural network data are preprocessed by a few techniques. One of them is scaling data to the range [0,1] either by using:

1. *formula* ***Xn = ((X - Xmin) / (Xmax - Xmin)) * (-1) + 1*** *or*
2. ***MinMaxScaler()*** function from the *scikit-learn* library.

After trying both techniques there is used just the second one for the further processing of neural network due to better results in training (by checking accuracy of simulation after network training). Before the creation and training of neural network, data is separated with function by a separation rate of 70% - 30% for training and test set.

Network Architecture

After training and test set creation a multilayer feed forward network is created. The network architecture consists of 2 neurons in hidden layer and based on output classes (0 or 1) 1 neuron in output layer. There was also changed the transfer function for the output layer to Log-Sigmoid. Afterwards the network was trained with the following parameter's setup:

- learning rate: **0.3**,
- maximum number of epochs to train: **3000**,
- performance goal: **1e-5**.

There is used the matplotlib library for displaying results. On the Figure 2 we can see the progress of training by increasing epochs. As we can see, in epoch number 1500 and 3000 the error is almost the same. That means that the network knows the majority of the given dataset in epoch number 1500 already but as we are still not getting performance goal, the computing continues until maximum number of epochs to train is reached.

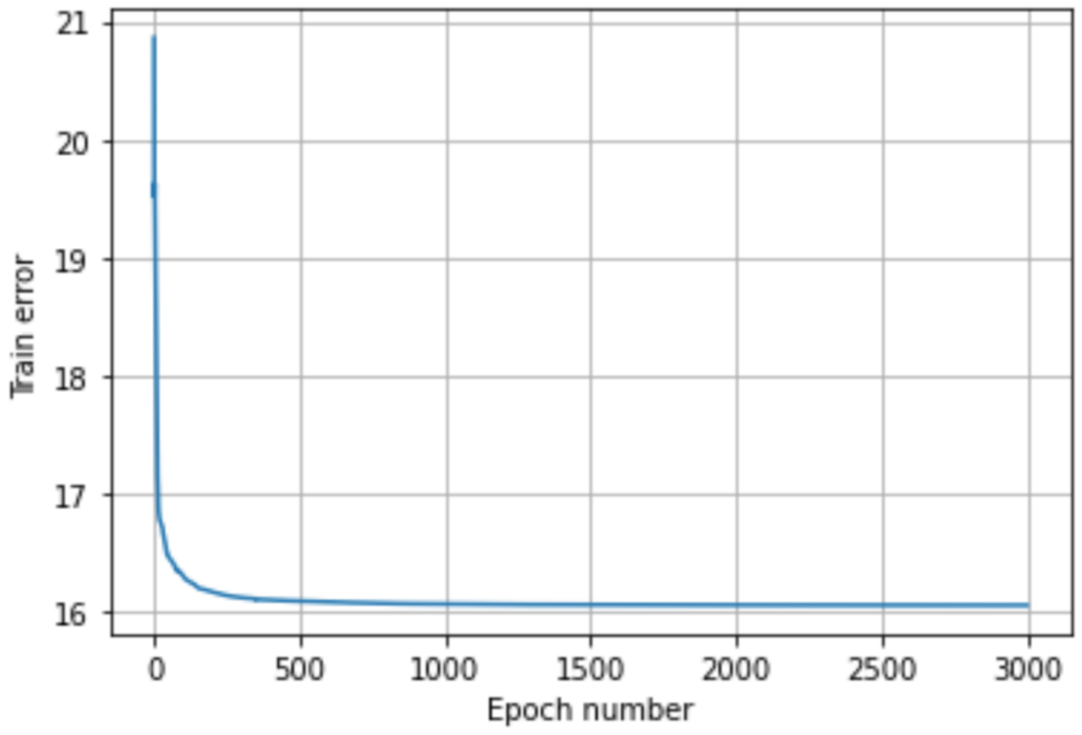

Figure 2: Multilayer feed forward network training progress

Now we can present to the trained neural network the test set to display the categorized patterns by simulating network with the given test set to check and compare results in relation to the actual class. First of all, an output of the simulation is compared to a target of the test set which is, of course, **False** - there is more than **16% error rate** in training. The next step is to find out the accuracy based on comparison of these two types of output:

Accuracy: 0.6956521739130435 (69.57 %).

For better understanding, a comparison of results on the Figure 3 is displayed. As we can see, the samples from the test set is assigned to the class they belong to, based on the previous neural network training. There are few mistakes in assignment, but results are not so bad.

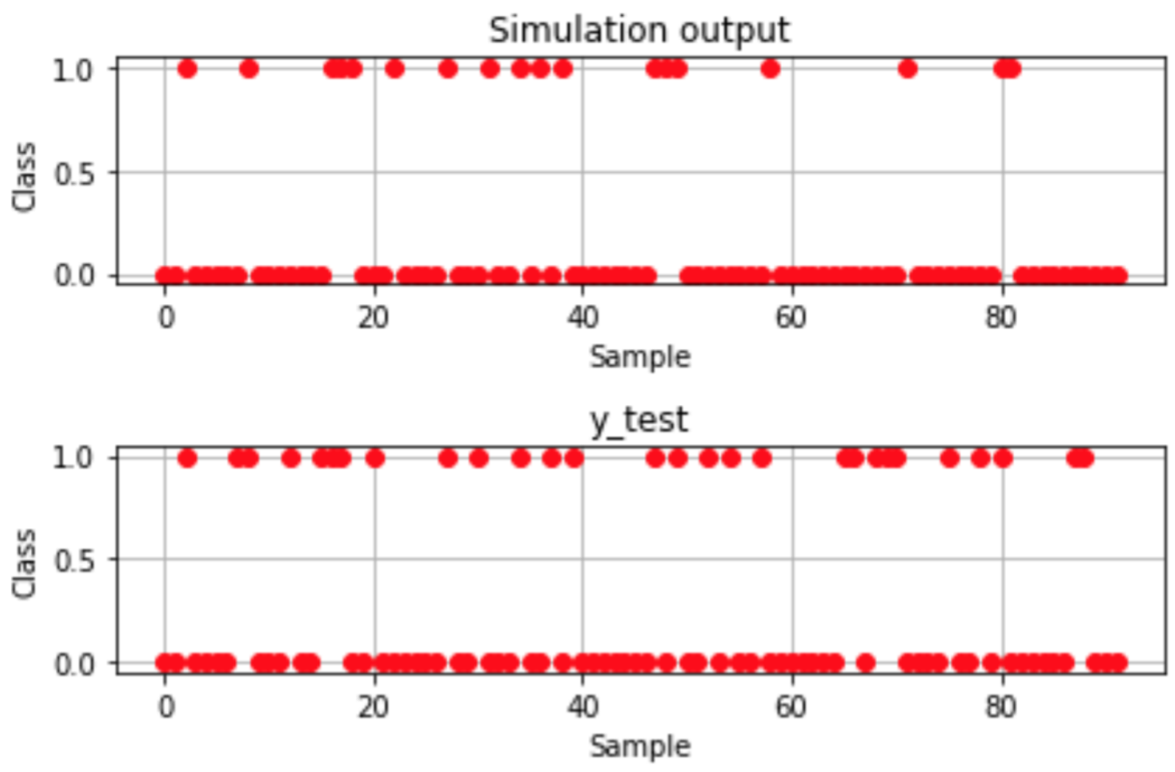

Figure 3: Comparison of simulation to the actual class

To have even better view on the results, there are chosen 10 random patterns from the test set: sample 21, 29, 35, 37, 46, 54, 66, 76, 81 and 87. As we can see on the Figure 4 there are comparison of these samples in the simulation of trained network and the real result. In this case we have 5 mistakes: in the sample number 37, 54, 66, 81 and 87. Now we can say that results are not satisfying therefore we need to continue in a network optimization.

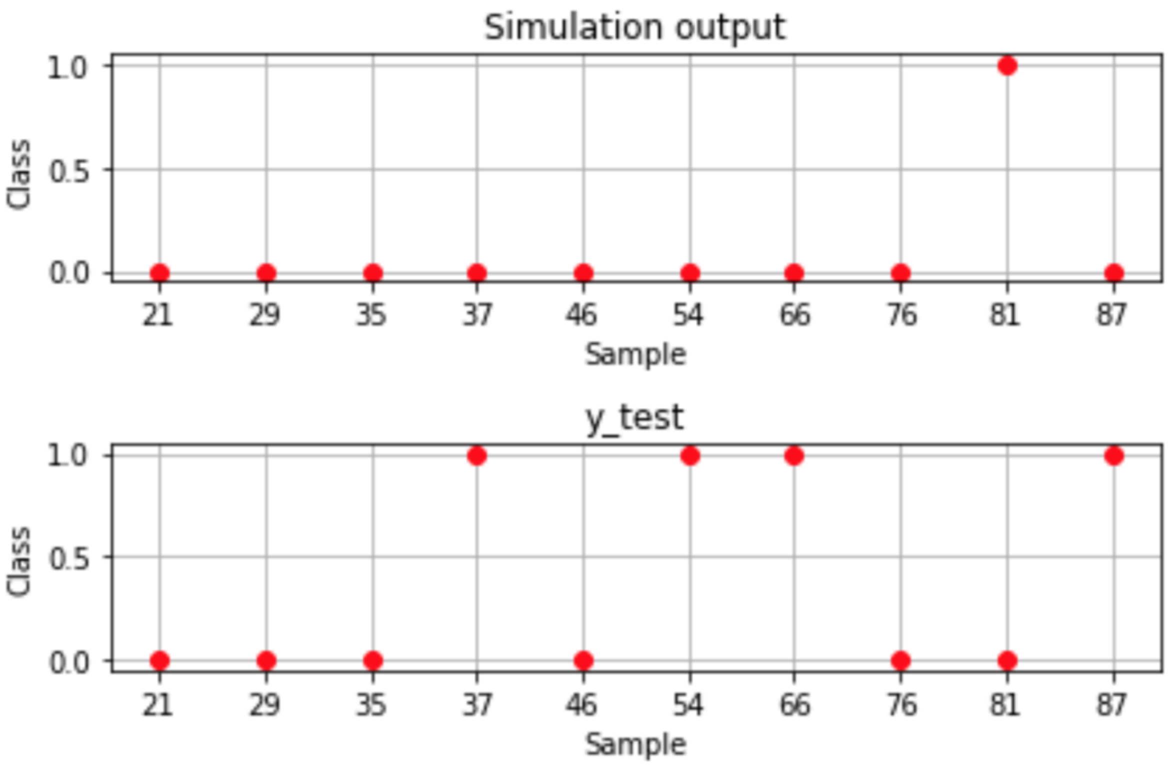

Figure 4: Comparison of 10 random patterns

Network optimization

There were made some optimizations in network architecture and also in method of training the neural network by changing some attributes, for example adding or subtracting neurons in hidden layer, changing the number of epochs, learning rate etc. Every change is recorded in following tables same as the result accuracy with their mean values after a few tries. First, we start with changing of neurons in hidden layer.

Hidden Layer Neurons

Epochs	Goal	Learning Rate		Hidden Layer Neurons	
3000	1e-5	0.3		**3**	
Try	1.	2.	3.	4.	Mean value
Accuracy (%)	72.83	79.35	67.39	60.87	**70.11**

Epochs	Goal	Learning Rate		Hidden Layer Neurons	
3000	1e-5	0.3		**4**	
Try	1.	2.	3.	4.	Mean value
Accuracy (%)	69.57	73.91	79.35	70.65	**73.37**

Epochs	Goal	Learning Rate		Hidden Layer Neurons	
3000	1e-5	0.3		**6**	
Try	1.	2.	3.	4.	Mean value
Accuracy (%)	67.39	66.30	76.09	63.04	**68.21**

As we can see, the test with an increase in neurons hidden layer did not show significantly better results. The best result was achieved with the number **4**, so we will continue with it in the next optimization.

Epochs number

There were made several changes in the number of epochs in this test, from 2 epochs to 3500. The individual results are recorded in the tables. As in the previous test, several experiments were performed, from which the mean value is calculated.

Epochs	Goal	Learning Rate		Hidden Layer Neurons	
2	1e-5	0.3		4	
Try	1.	2.	3.	4.	Mean value
Accuracy (%)	41.30	53.26	69.57	22.83	**46.74**

Epochs	Goal	Learning Rate		Hidden Layer Neurons	
10	1e-5	0.3		4	
Try	1.	2.	3.	4.	Mean value
Accuracy (%)	73.91	80.43	75.00	66.30	**73.91**

Epochs	Goal	Learning Rate		Hidden Layer Neurons	
50	1e-5	0.3		4	
Try	1.	2.	3.	4.	Mean value
Accuracy (%)	69.57	78.26	80.43	78.26	**76.63**

Epochs	Goal	Learning Rate		Hidden Layer Neurons	
100	1e-5	0.3		4	
Try	1.	2.	3.	4.	Mean value
Accuracy (%)	77.17	73.91	69.57	68.48	**72.28**

Epochs	Goal	Learning Rate		Hidden Layer Neurons	
300	1e-5	0.3		4	
Try	1.	2.	3.	4.	Mean value
Accuracy (%)	73.91	65.22	68.48	76.09	**70.92**

Epochs	Goal	Learning Rate		Hidden Layer Neurons	
500	1e-5	0.3		4	
Try	1.	2.	3.	4.	Mean value
Accuracy (%)	71.74	68.48	64.13	76.09	**70.11**

Epochs	Goal	Learning Rate		Hidden Layer Neurons	
1000	1e-5	0.3		4	
Try	1.	2.	3.	4.	Mean value
Accuracy (%)	73.91	65.22	63.04	71.74	**68.48**

Epochs	Goal	Learning Rate		Hidden Layer Neurons	
3500	1e-5	0.3		4	
Try	1.	2.	3.	4.	Mean value
Accuracy (%)	66.30	76.09	72.83	72.83	**72.01**

From these tests can be seen that the best results are achieved with a setting of 50 epochs. From the point of view of neural network performance, it is unnecessary to use such a large number of epochs as before (3000), because the results are not much better, and the computational complexity is much higher. We will continue with further optimization with **50** epochs.

Learning Rate

The last of the parameter testing was the learning rate. We will try to use the learning rate from 0.05, to 0.8 to see which gave us the best results. As in the previous test, several experiments were performed, from which the mean value is calculated.

Epochs	Goal	Learning Rate		Hidden Layer Neurons	
50	1e-5	**0.05**		4	
Try	1.	2.	3.	4.	Mean value
Accuracy (%)	76.09	75.00	72.83	68.48	**73.10**

Epochs	Goal	Learning Rate		Hidden Layer Neurons	
50	1e-5	**0.1**		4	
Try	1.	2.	3.	4.	Mean value
Accuracy (%)	78.26	79.35	71.74	65.22	**73.64**

Epochs	Goal	Learning Rate		Hidden Layer Neurons	
50	1e-5	**0.2**		4	
Try	1.	2.	3.	4.	Mean value
Accuracy (%)	81.52	73.91	75.00	67.39	**74.46**

Epochs	Goal	Learning Rate		Hidden Layer Neurons	
50	1e-5	**0.4**		4	
Try	1.	2.	3.	4.	Mean value
Accuracy (%)	69.57	83.70	71.74	80.43	**76.36**

Epochs	Goal	Learning Rate		Hidden Layer Neurons	
50	1e-5	**0.5**		4	
Try	1.	2.	3.	4.	Mean value
Accuracy (%)	70.65	71.74	73.91	66.3	**70.65**

Epochs	Goal	Learning Rate		Hidden Layer Neurons	
50	1e-5	**0.8**		4	
Try	1.	2.	3.	4.	Mean value
Accuracy (%)	75.00	72.83	71.74	78.26	**74.46**

As we can see from the tests, the best was the learning rate **0.4**. On the basis of all performed tests it can be stated that we managed to optimize the network from original parameters to those that show significantly better results in the classification of individual patterns from this dataset. Now we will repeat the procedure for this network architecture with different split rates described below.

Split rate changing

At this point, we already have a multilayer feed forward neural network optimized with the following parameters, listed in the table below. Next, we will try to change the ratio of training and test data to evaluate the results.

Parameters of optimized neural network

Epochs	Goal	Learning Rate	Hidden Layer Neurons
50	1e-5	0.4	4

Split rate Training set - Test set: 50% - 50%

Try	1.	2.	3.	4.	Mean value
Accuracy (%)	71.24	75.16	77.12	73.20	**74.18**

Split rate Training set - Test set: 60% - 40%

Try	1.	2.	3.	4.	Mean value
Accuracy (%)	71.54	73.98	76.42	75.61	**74.39**

Split rate Training set - Test set: 70% - 30%

Try	1.	2.	3.	4.	Mean value
Accuracy (%)	69.57	83.70	71.74	80.43	**76.36**

Split rate Training set - Test set: 80% - 20%

Try	1.	2.	3.	4.	Mean value
Accuracy (%)	77.42	75.81	80.65	79.03	**78.23**

Split rate Training set - Test set: 90% - 10%

Try	1.	2.	3.	4.	Mean value
Accuracy (%)	77.42	83.87	93.55	74.19	**82.26**

Summary

As we can see from the performed tests, the more training data is selected, the better the overall result is. The reason why this is like that is because the neural network is trained with much more cases and therefore has more knowledge and subsequent accuracy. The curve of mean values of accuracies from the individual split rates are displayed on the Figure 5.

Split rate	**50:50**	**60:40**	**70:30**	**80:20**	**90:10**
Accuracy (%)	74.18	74.39	76.36	78.23	82.26

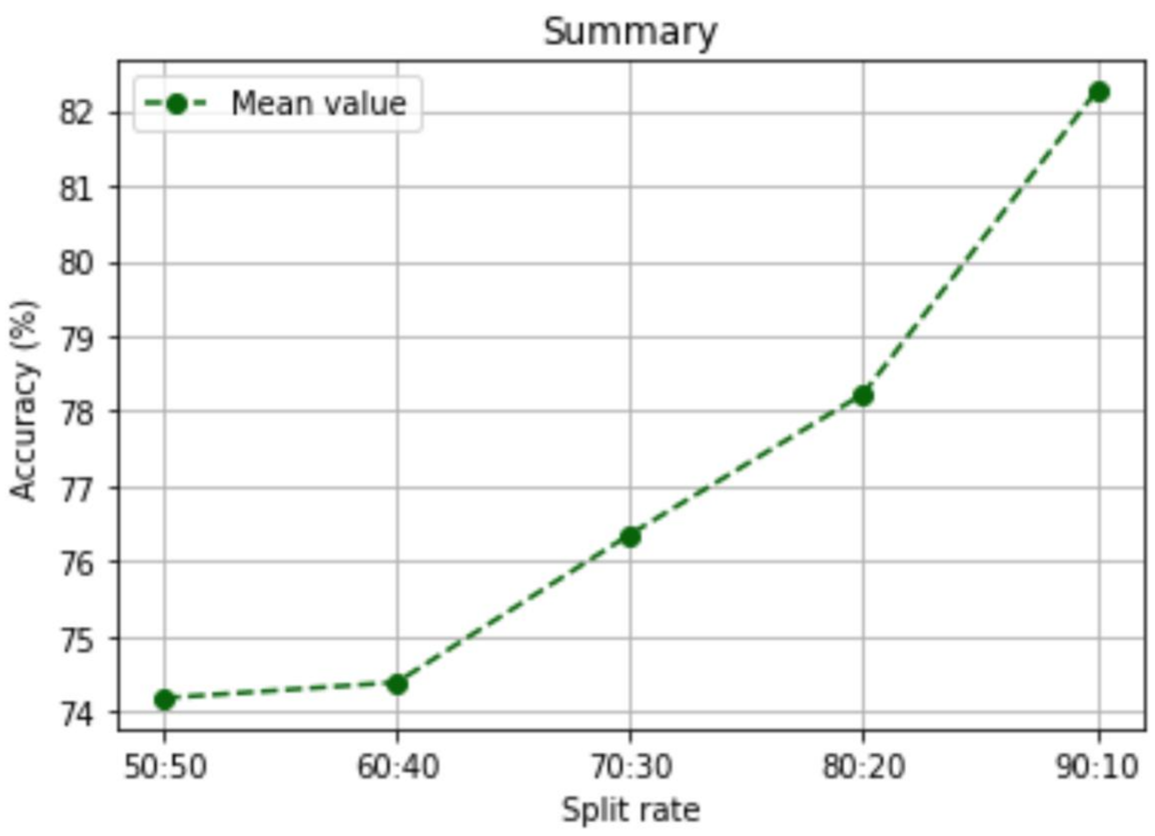

Figure 5: Curve of split rate accuracy

Kohonen Network

The work done so far has been part of supervised learning, i.e. that neural network training took place on the basis that we knew the belonging of individual patterns to classes. In this case, however, we will work with unsupervised learning, so we assume that we do not know the affiliation of individual patterns to classes, but we will calculate the Euclidean distance of individual attributes these patterns using the Kohonen network.

First, before we train this network, we cluster data – calculate distortion for a range of number of clusters. There is used the elbow method. The elbow method is a useful graphical tool to estimate the optimal number of clusters *k* for a given task. Intuitively, we can say that if *k* increases, the within-cluster SSE (distortion) will decrease due to the samples will be closer to the centroids they are assigned to. Using this method we have found that in this dataset there will be 2 clusters of individual patterns, what we can see on Figure 6.

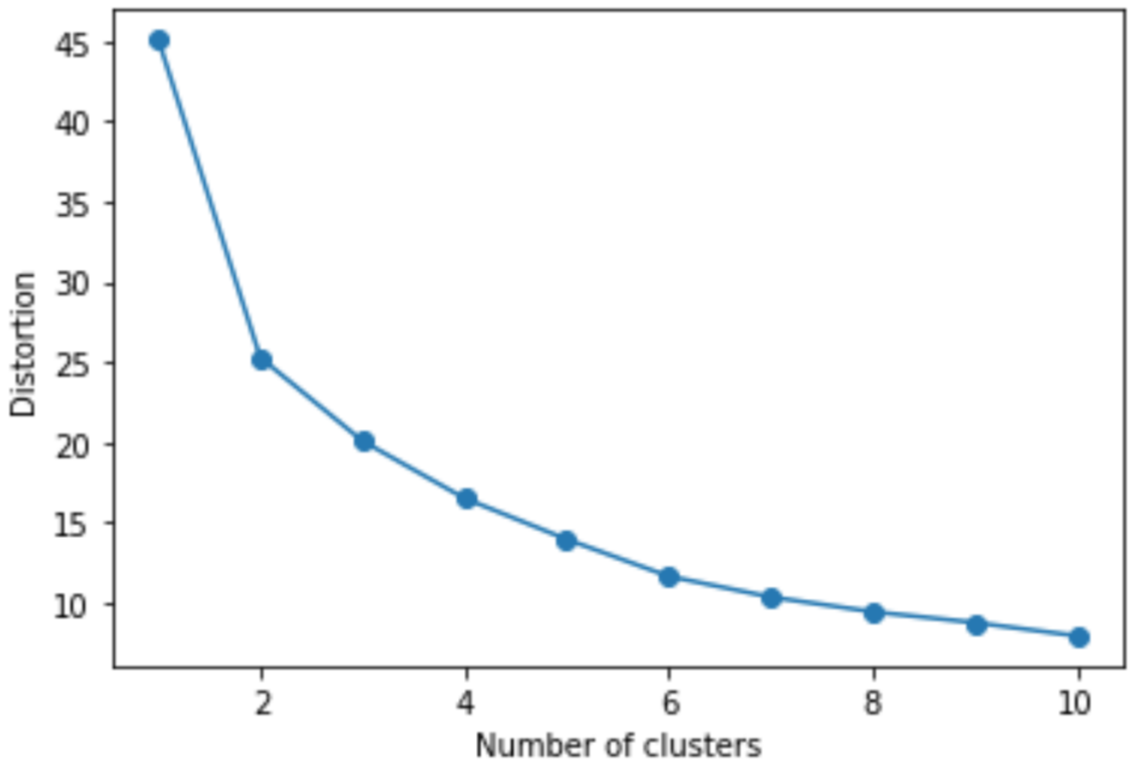

Figure 6: Clustering using K-means algorithm

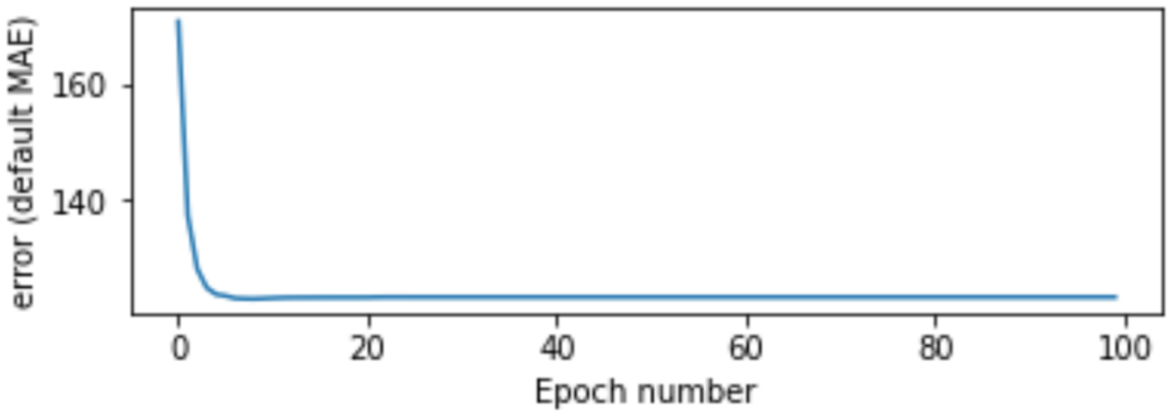

Figure 7: Kohonen network training progress

Now we can make a competitive layer (Kohonen neural network) with 2 output neurons. Kohonen network consist of just one layer. This type of network uses the competitive transfer function and calculates the Euclidean distance for comparing the characteristic values of each pattern with the weights. After creation of this network we train it with rule: Winner Take All algorithm (WTA). The training progress is displayed on Figure 7.

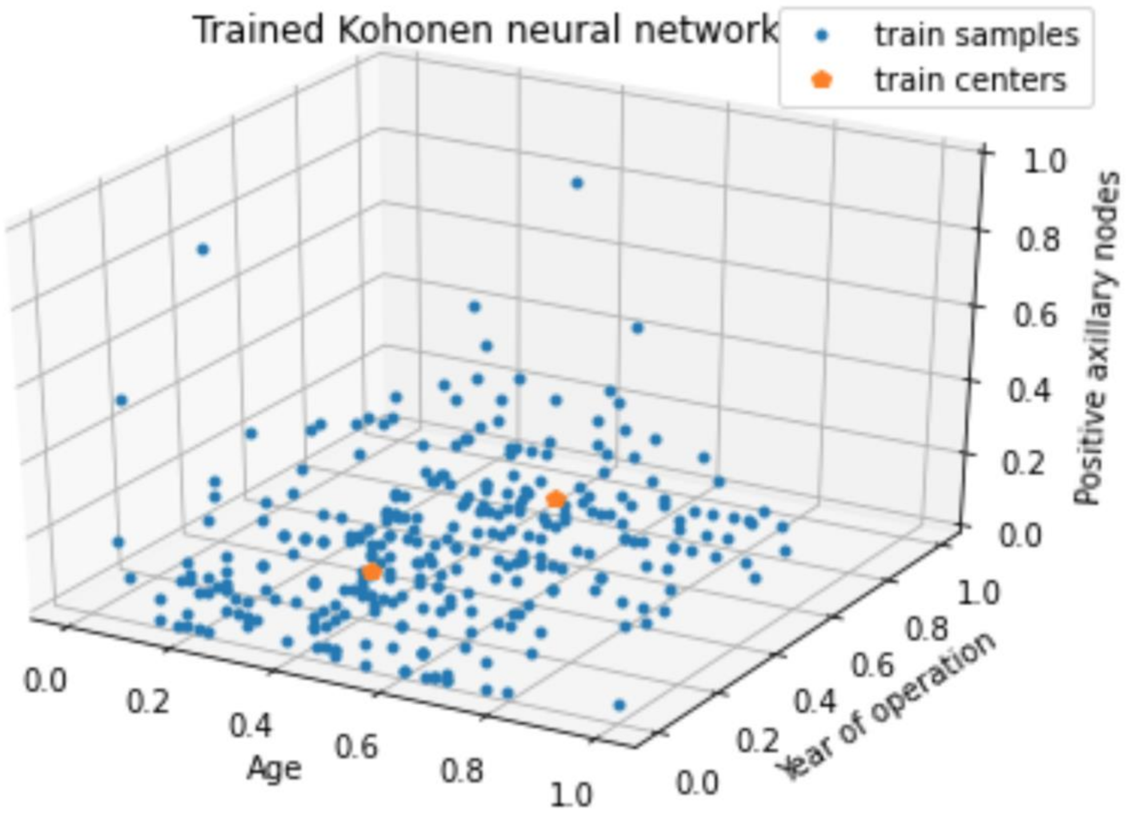

Figure 8: Trained Kohonen network

Based on data retrieval, clustering, creation and subsequent training of the Kohonen neural network, we can present the obtained results in a 3D on the Figure 8. From it, it is really possible to know 2 assigned training centers, which resulted from the method of computation used by this type of network.

Conclusion

Different solutions which were tested in this experiment have shown that the choice of the number of hidden neurons is crucial to the effectiveness of a neural network. Also, the experiment showed that the success of a neural network is very sensitive to parameters chosen in the training process. At the end, the results have shown that the total mean square error does not reflect directly the success of a network.

Based on the selected dataset, it is necessary to consider what type of neural network will be used. For the correct use of the given dataset, it is then necessary to modify the data by various methods and then to train the given neural network with suitable parameters. Based on these results, it can be stated that the sensitivity of some parameters is very high, which even a small change can have a big impact on the result.