**Report**-**Adaline:**

**(i)+ (iii)**

Constants: weights: 0.1, 0.1

bias: 0.1

alpha: 0.1

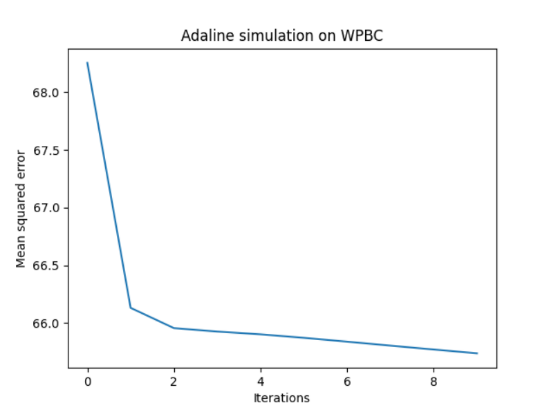
iterations: 10

1. test percent : 25%

train percent : 75%

accuracy: 63.26530612244898%

time: 1.82 sec

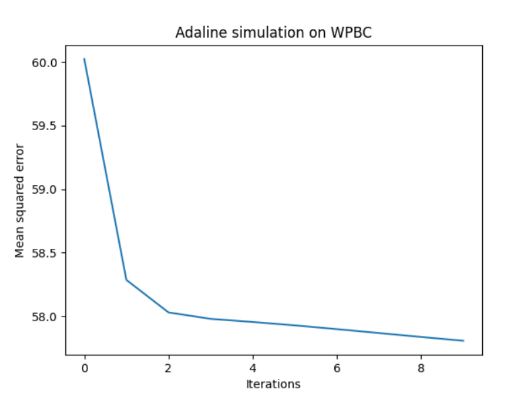


1. test percent : 33%

train percent : 67%

accuracy: 35.38461538461539%

time: 1.73 sec

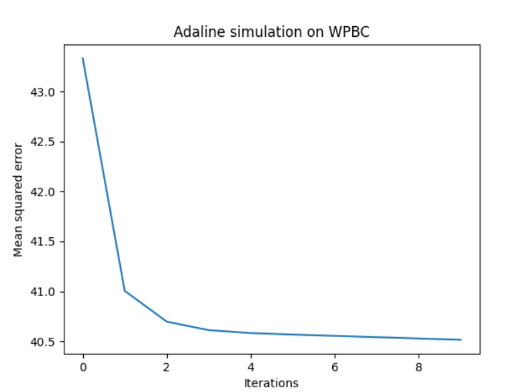


1. test percent : 50%

train percent : 50%

accuracy: 75.51020408163265%

time: 1.81 sec



\*We can see that with low iterations the graphs look the same, that is, from iteration 2 or higher the learning is relatively small.

\*Average of cross validation: 58.053‬%

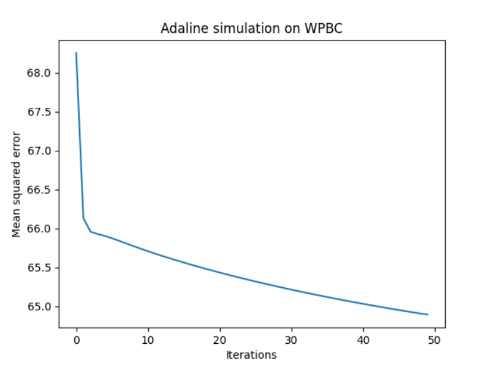
iterations: 50

1. test percent : 25%

train percent : 75%

accuracy: 63.26530612244898%

time: 1.95 sec

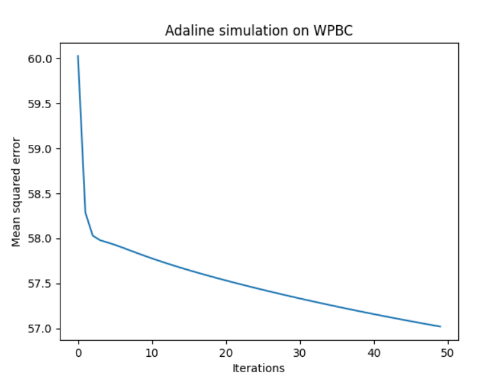


1. test percent : 33%

train percent : 67%

accuracy: 40%

time: 1.56 sec

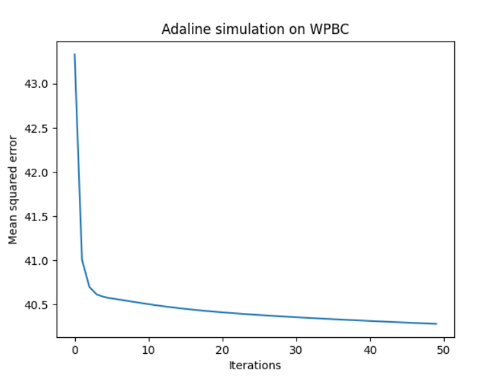


1. test percent : 50%

train percent : 50%

accuracy: 75.51020408163265 %

time: 2.11 sec



\* You can see that with the number of 50 iterations in the graphs, the higher the percentage of testing, the sharper the decrease in the lower iterations. In other words, when the percentages of testing are higher, learning is relatively smaller from iteration 2 or higher.

\*Average of cross validation: 59.591%

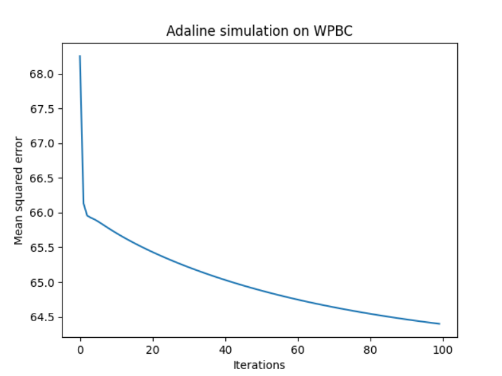
iterations: 100

1. test percent : 25%

train percent : 75%

accuracy: 59.183673469387756%

time: 2.20 sec

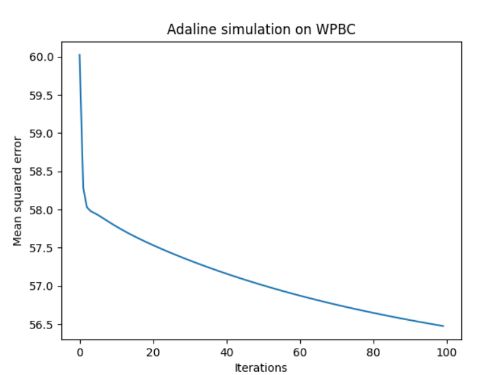


1. test percent : 33%

train percent : 67%

accuracy: 41.53846153846154%

time: 1.98 sec

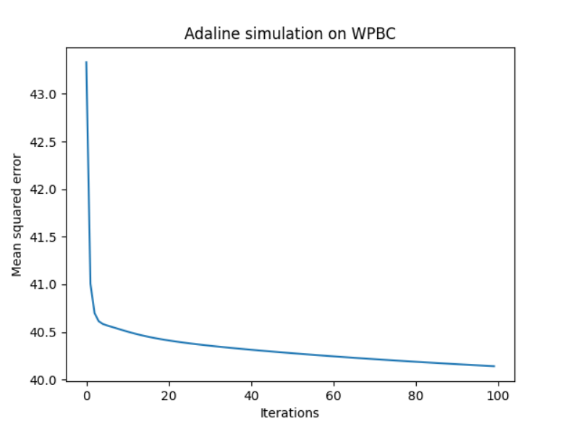


1. test percent : 50%

train percent : 50%

accuracy: 75.51020408163265%

time: 1.97 sec



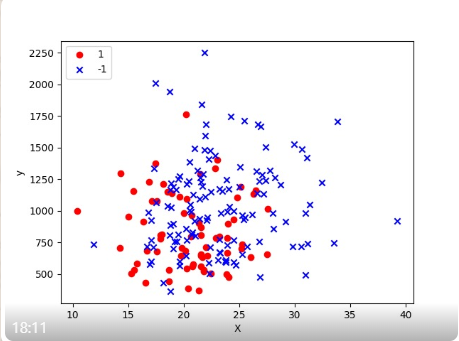
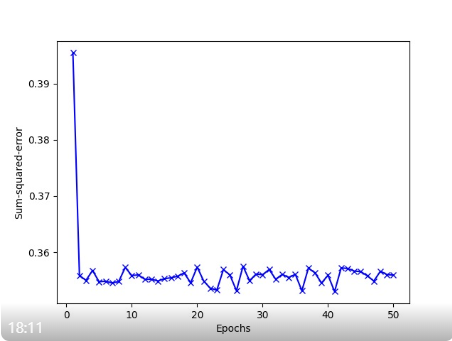
\* As in the previous iterations, the higher the percentage of testing, the smaller the learning.

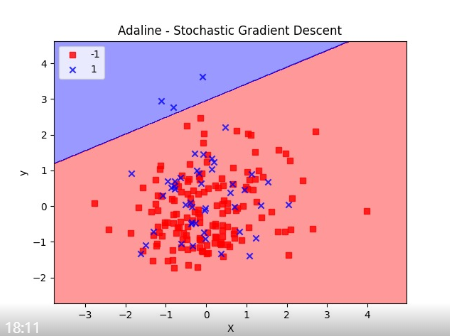
\*Average of cross validation: 58.7436 %

**(ii)** All parameters can be changed simply.

**(iv)** We started the project from collecting information about the Adaline algorithm. Once we understood how the algorithm works , we started writing code in Python. The given examples are only a small but representative selection of all test runs and measurements performed for this project and illustrate the advantages and disadvantages of genetic methods for the training of neuronal networks. Theoretic analysis and the optioned practical results suggest the following conclusions:

From the data we received, it was very difficult to get past the 76% accuracy. Following the use of the linear model:



When we printed the data in the above graphs, we found that we could not pass a linear line that would separate the different data (R and N).

In addition, all the data for training and testing could not be converge. So we only used some of the data.

In a test with 33% and 67% training, as the number of iterations increases, so does the accuracy. Because with many iterations, the algorithm learns from errors of previous iterations.

When we change the training and testing percentages as the number of iterations increases, the accuracy percentages didn’t change and sometimes even decrease. This means that the previous percentage distribution was most efficient in terms of the number of iterations and the percentages of accuracy.