León Andrés Rodríguez Guzmán **A01700687**
Juan Manuel Ledesma Rangel **A01206568**
Laboratory #7
November 14th, 2018
Intelligent Systems Group #1 Tuesday and Friday 5:30pm
Rubén Stranders

19. Lab: Implementing Neural Networks

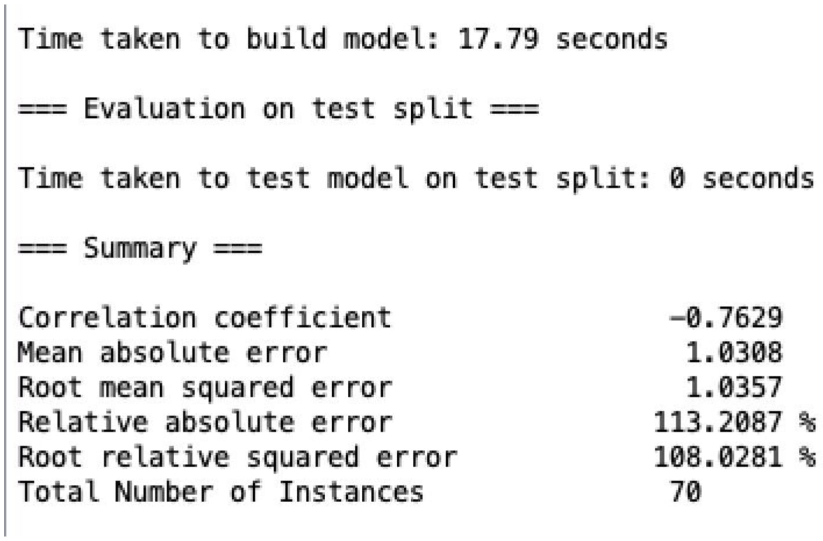
Goal: understand how to create an ANN and the pros and cons of training it **(6 hours)**

Using WEKA load and run ANN (multilayer perceptron) example and analyse the inputs and outputs.

Move around the factors such as learning rate, number of layers and number of nodes. Record the behaviour of the network. We are interested in observing if it converges or not, how long it takes to learn, and how precise it is. *These measurements and comparisons should be included in your report. Find an interesting training set here and use it to train a network.*

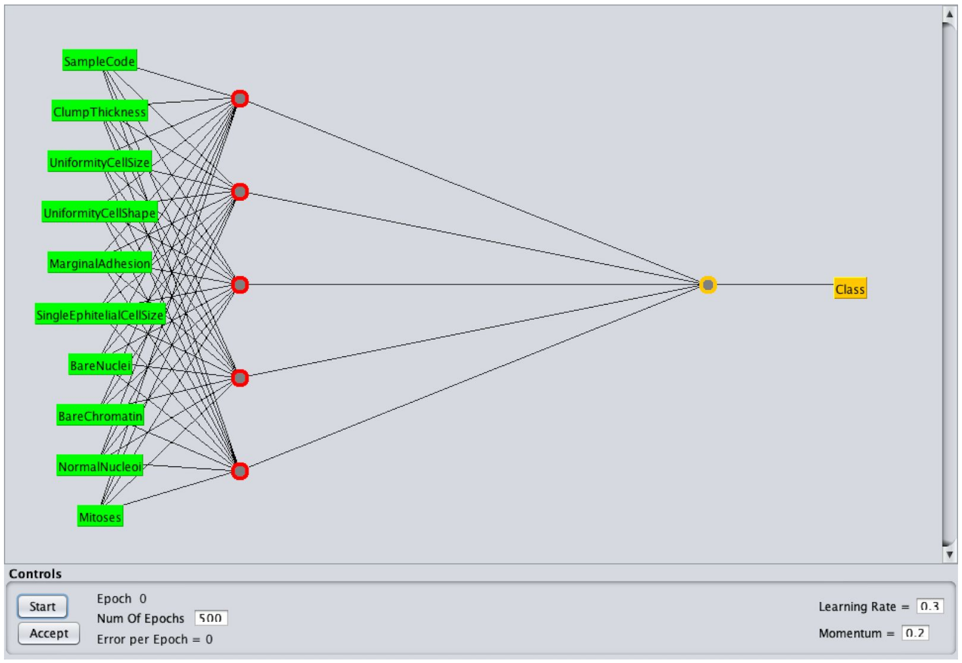
For this part, we decided to use the following data set, referred to breast cancer: https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29

We decided to use from the whole data set, a 90% for the training set and 10% for testing. The results where:



```
Time taken to build model: 17.79 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0 seconds

=== Summary ===

Correlation coefficient                 -0.7629
Mean absolute error                      1.0308
Root mean squared error                  1.0357
Relative absolute error                113.2087 %
Root relative squared error            108.0281 %
Total Number of Instances               70
```

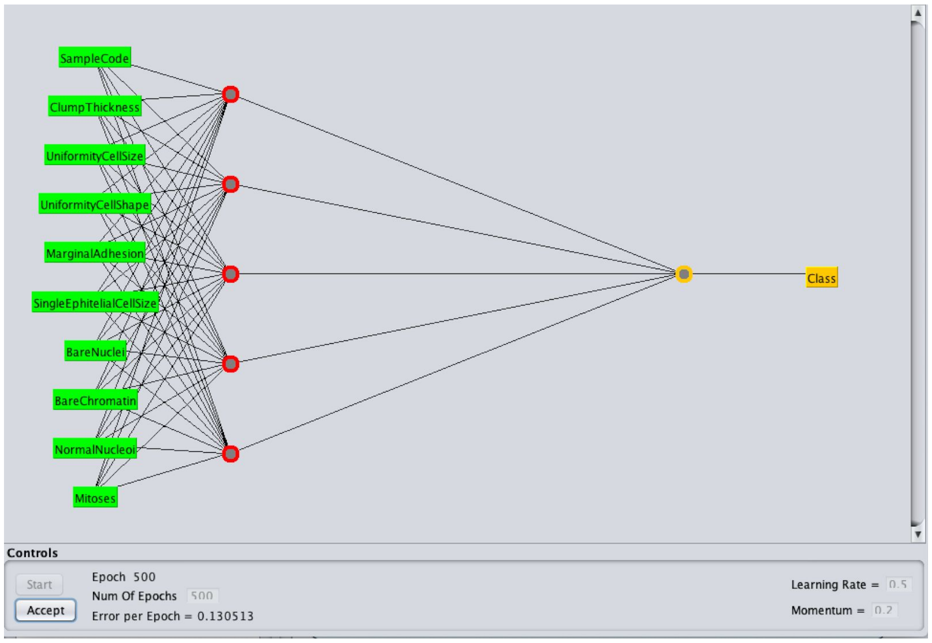
As it can be seen, the error is really low and the time it took wasn't that much. The MNN formed is the following:



Epoch 0
Num Of Epochs 500
Error per Epoch = 0
Learning Rate = 0.3
Momentum = 0.2

We were asked to modify the 3 factors, learning rate, number of layers and number of nodes. We decided to vary this values twice for learning rate and twice for both number of layers and number of nodes; additionally, to try to make overfitting and to challenge our network we'll do a mixture of everything, so we obtained 5 different outcomes that are shown as follows:

Learning rate of 0.5:



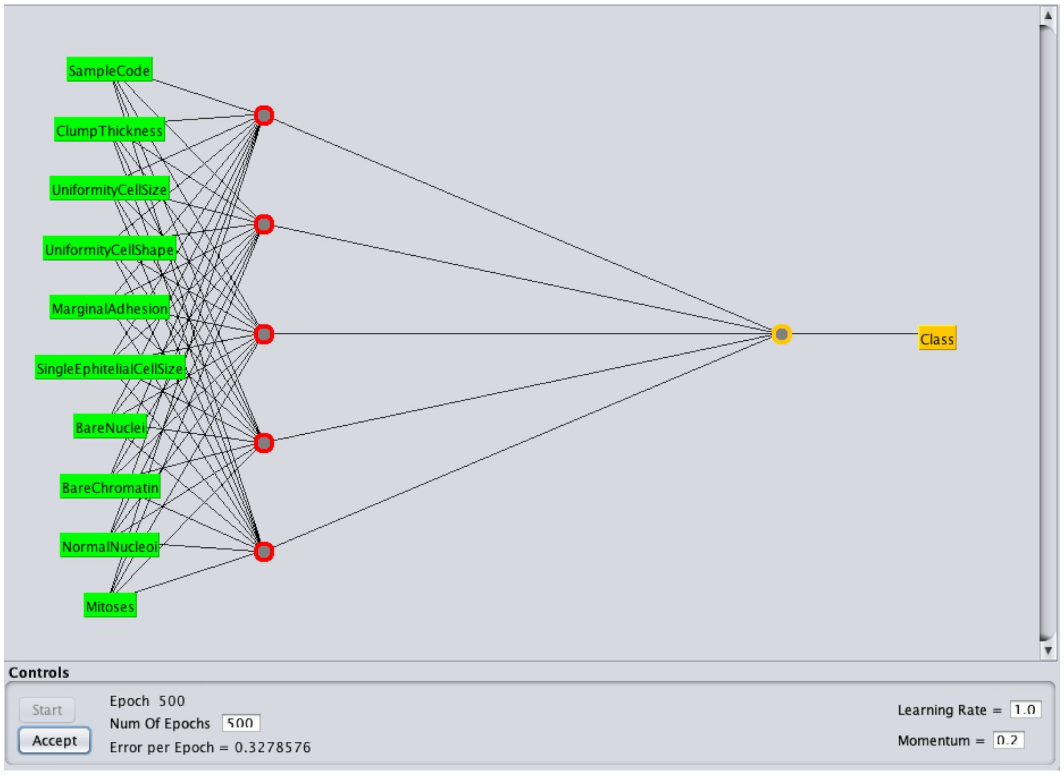
Epoch 500
Num Of Epochs 500
Error per Epoch = 0.130513
Learning Rate = 0.5
Momentum = 0.2

The error was of 0.13 per epoch.

```
Time taken to build model: 5.65 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0 seconds

=== Summary ===

Correlation coefficient                  0.8813
Mean absolute error                      0.1467
Root mean squared error                  0.4729
Relative absolute error                 16.1089 %
Root relative squared error             49.3307 %
Total Number of Instances               70
```

The error was lowered a lot, and the time lowered as well.

Learning rate of 1:



Epoch 500
Num Of Epochs 500
Error per Epoch = 0.3278576
Learning Rate = 1.0
Momentum = 0.2

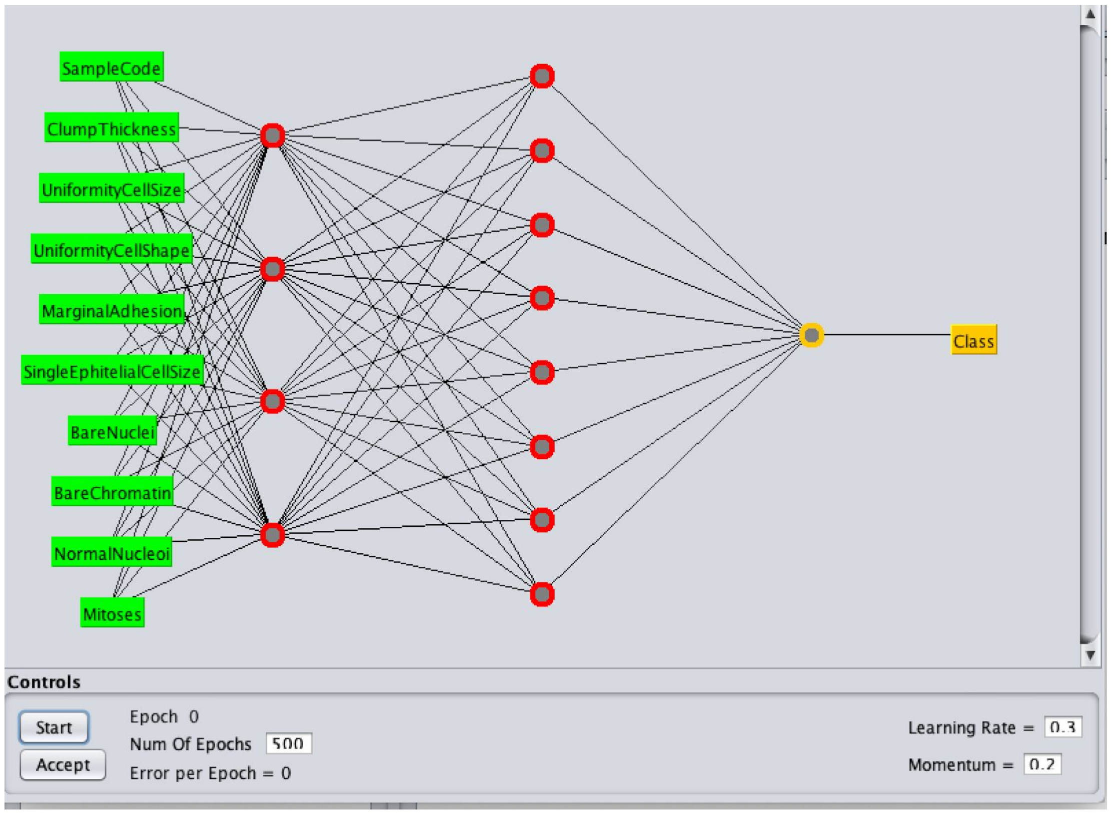
The learning error was doubled.

```
Time taken to build model: 208.47 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0 seconds

=== Summary ===

Correlation coefficient                  0.8795
Mean absolute error                      0.1181
Root mean squared error                  0.4785
Relative absolute error                 12.972  %
Root relative squared error             49.9147 %
Total Number of Instances               70
```

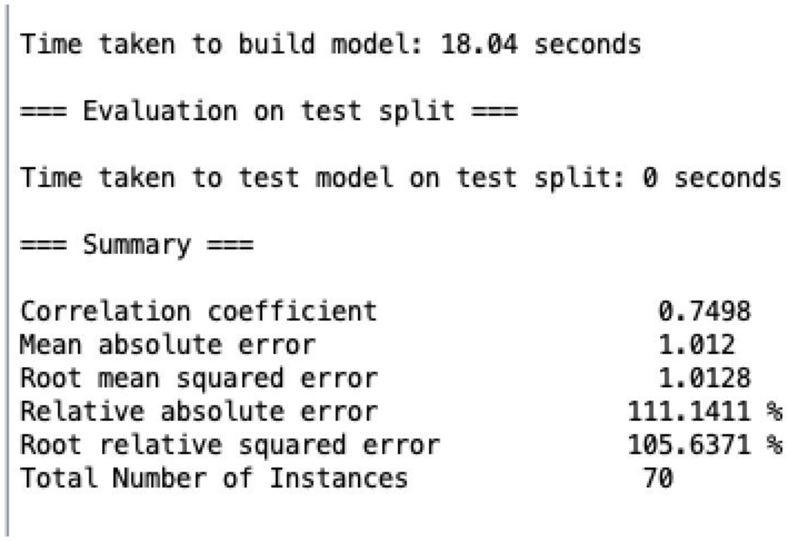
Again, the absolute error was lowered to almost 0.1, but now the time increased.

The last two factors can only be observed as pairs, so we proposed two combinations:

Number of layers 2 and number of nodes 4 and 8.

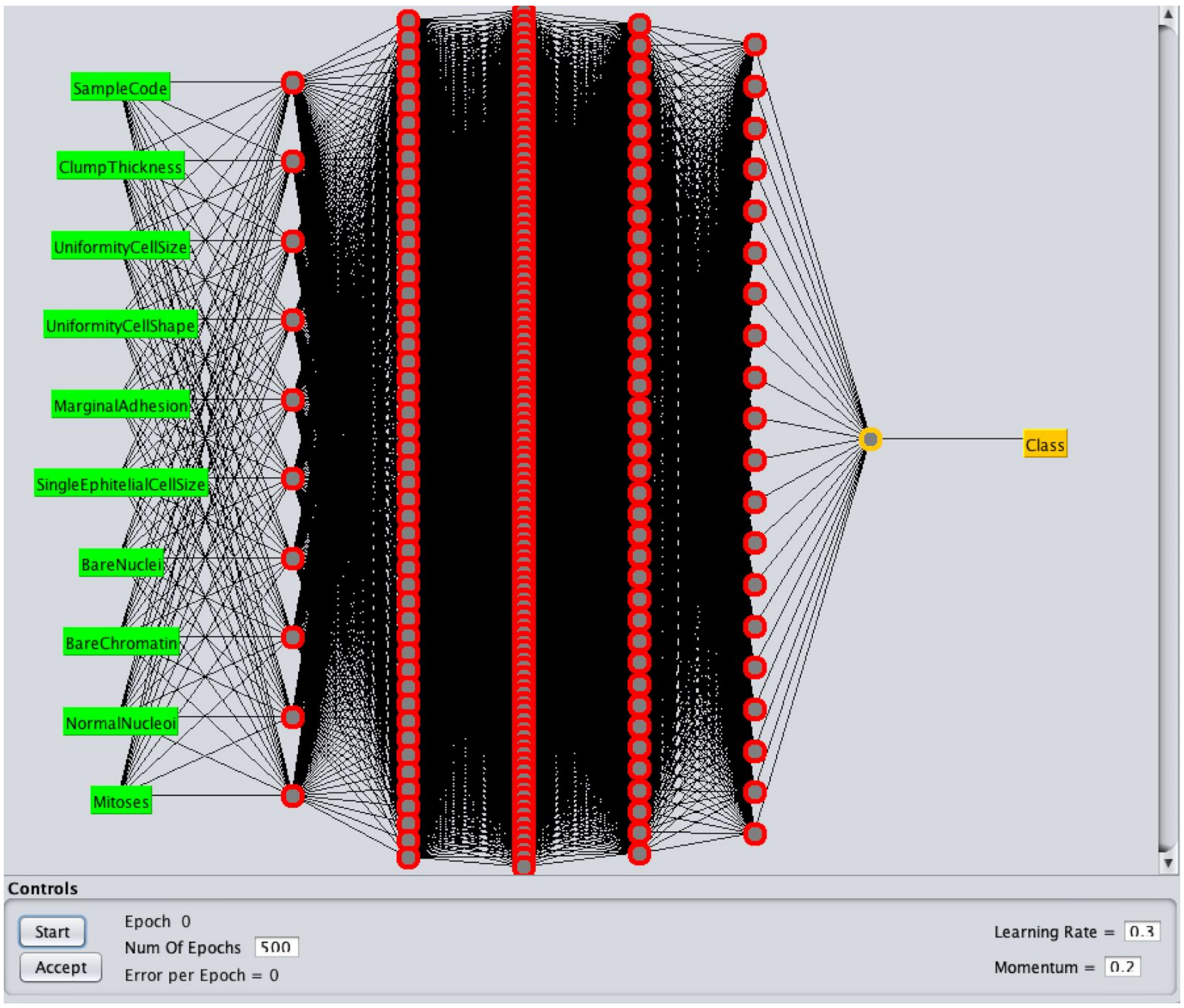


Epoch 0
Num Of Epochs 500
Error per Epoch = 0
Learning Rate = 0.3
Momentum = 0.2

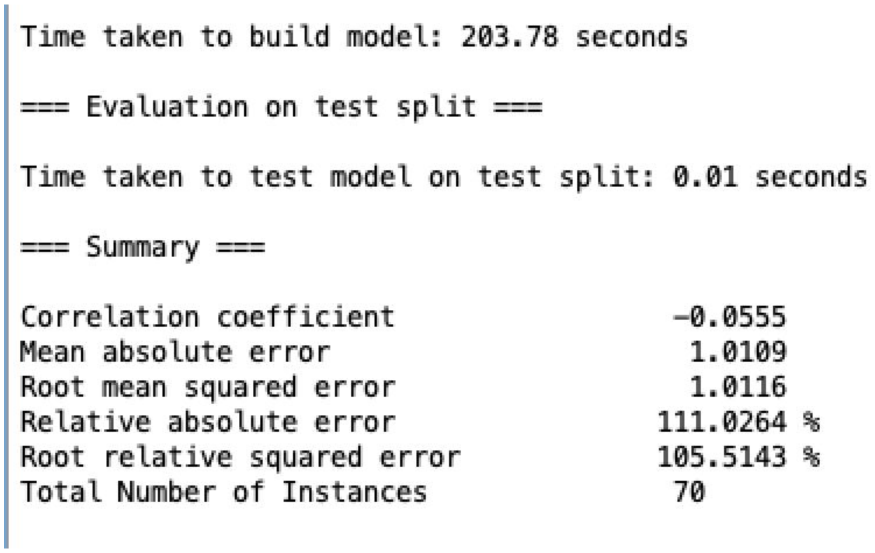


```
Time taken to build model: 18.04 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0 seconds

=== Summary ===

Correlation coefficient                  0.7498
Mean absolute error                      1.012
Root mean squared error                  1.0128
Relative absolute error                111.1411 %
Root relative squared error            105.6371 %
Total Number of Instances               70
```

Number of layers 5 and number of nodes 10, 50, 100, 40 and 20.

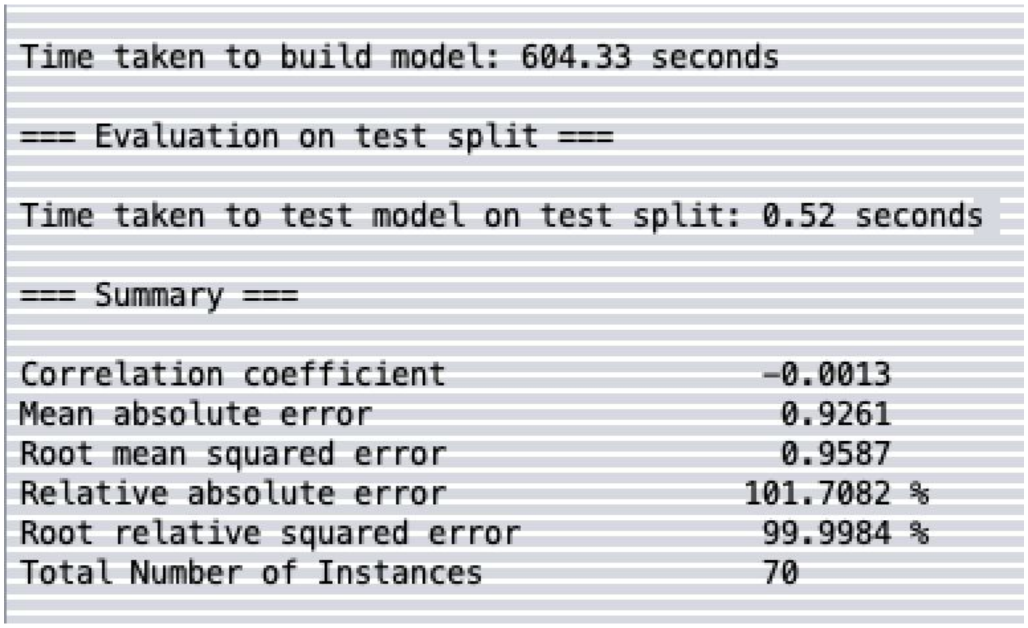


Epoch 0
Num Of Epochs 500
Error per Epoch = 0
Learning Rate = 0.3
Momentum = 0.2



```
Time taken to build model: 203.78 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0.01 seconds

=== Summary ===

Correlation coefficient                 -0.0555
Mean absolute error                      1.0109
Root mean squared error                  1.0116
Relative absolute error                111.0264 %
Root relative squared error            105.5143 %
Total Number of Instances               70
```

Learning rate 0.3, number of layers 8 and number of nodes 10, 50, 100, 40, 20, 1000, 500 and 600, training time 25 iterations. We first tried a training time of 1500 iterations, but it wasn't able to finish after 3 hours.



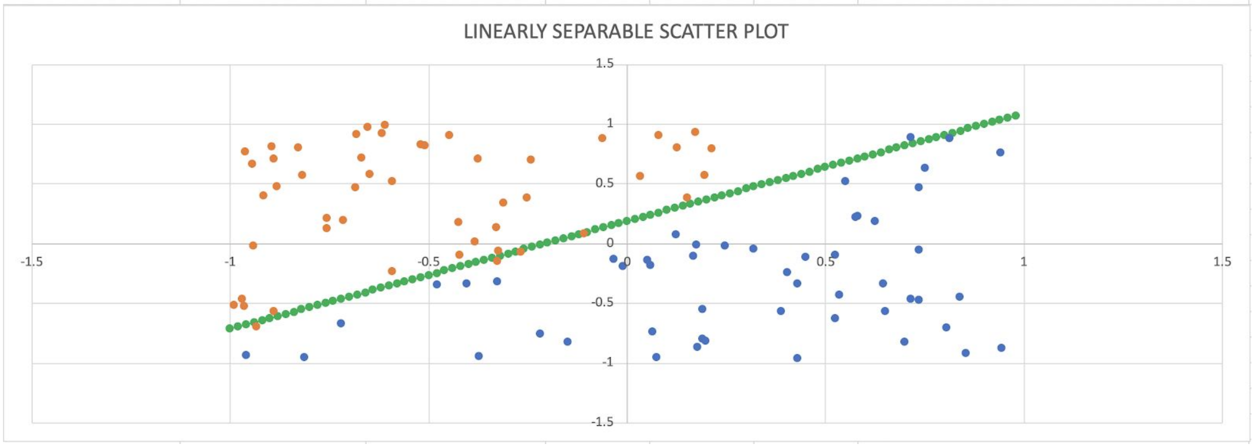
```
Time taken to build model: 604.33 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0.52 seconds

=== Summary ===

Correlation coefficient                 -0.0013
Mean absolute error                      0.9261
Root mean squared error                  0.9587
Relative absolute error                101.7082 %
Root relative squared error             99.9984 %
Total Number of Instances               70
```

The report should have two sections, one for each part of this lab.

Part 1

Create a scatter plot of the training set of the linearly_separable example in the tests section. Use different colors or symbols for the two classes. Draw the line that the perceptron algorithm found. Include the mathematical equation of the line in your report.

Weight 1	Weight 2	Bias		Slope	B	Equation
0.559346288641311	-0.620026333757251	0.115343030851556		0.9021331	0.1860292	y = 0.902133116269063x + 0.186029245165439



The excel used with all the mathematical equations and procedure can be found at: https://goo.gl/12CrBc

When uploaded to drive, the graph was distorted, so that's why an image of the graph is presented.

It's important to mention that even though on the graph there might be some points that aren't completely separated by the line, they're right. This is proven by the column point check that checks the values of $x_1 w_1 + x_2 w_2 > bias$.

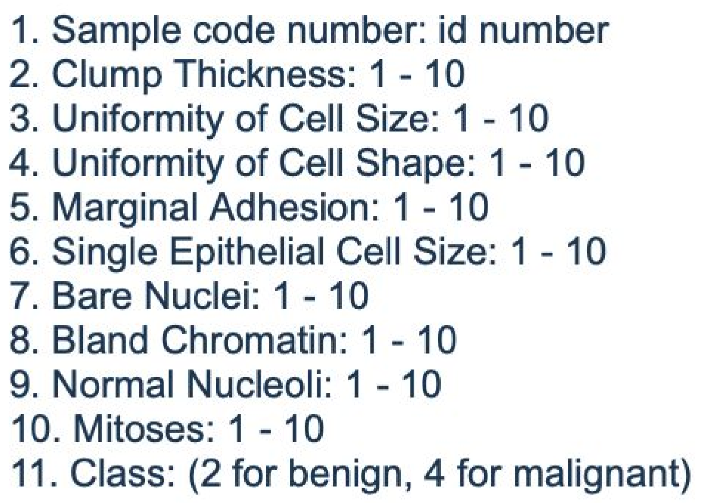
Part 2

Write an analysis of training the ANN with different parameters. Include an image of the network, the parameters used, and the error and time it took to train each network. Write a brief reflection about what you think is happening in the different ANNs, and why the ANN is behaving like it is. How did the choice of parameters influence the behavior of the ANN?

Other interesting aspects you could include in your reflection are:

- *Explanations as to what are ANNs good for.*
- *Where would you use them?*
- *Are they worth the effort implementing or not?*
- *What kinds of problems do they not solve?*

The images asked where already shown at the beginning of the document, so here we'll only provide an image of the parameters and the number of instances of the chosen data set.



1. Sample code number: id number
2. Clump Thickness: 1 - 10
3. Uniformity of Cell Size: 1 - 10
4. Uniformity of Cell Shape: 1 - 10
5. Marginal Adhesion: 1 - 10
6. Single Epithelial Cell Size: 1 - 10
7. Bare Nuclei: 1 - 10
8. Bland Chromatin: 1 - 10
9. Normal Nucleoli: 1 - 10
10. Mitoses: 1 - 10
11. Class: (2 for benign, 4 for malignant)

Those are the parameters, and the number of instances where 699.

For what we observed we can conclude that the more nodes and layers we use and the bigger the learning rate, the more time it will take for the network to be build properly. This was proven by our last example in which the time was of: 23423. Apart from that, we observed that the more nodes and layers we used, the better the network behaved, this till a limit of course, because then the network started to overfit. We believe that what's needed in an ANN is to find the perfect balance regarding number on nodes and layers, so that the network can behave as optimal as possible. ANN are good for many things, if we deal with linearity, then, a single preceptron will do the trick, however if we need to deal with non-linear data, then a multilayer network will work just fine. However, if we have data that changes quickly over time then an ANN or MNN won't work properly, especially if the number of nodes and data is high. Another disadvantage they present is that the usually don't see the big picture, because they focus on local minima. Compared to other machine learning structures like decision trees, ANN present significant advantages for example, their ability to learn through time each time more data is added or the structure of the network is changed. However, they're of course better machine learning structures that don't present the problems that ANN do, for example, support vector machines (SVM) don't present the disadvantages presented for ANN.

References:

Countz, T. (2018). *Calculate the decision boundary of a single perceptron - visualizing linear separability*. Retrieved from https://medium.com/@thomascountz/calculate-the-decision-boundary-of-a-single-perceptron-visualizing-linear-separability-c4d77099ef38