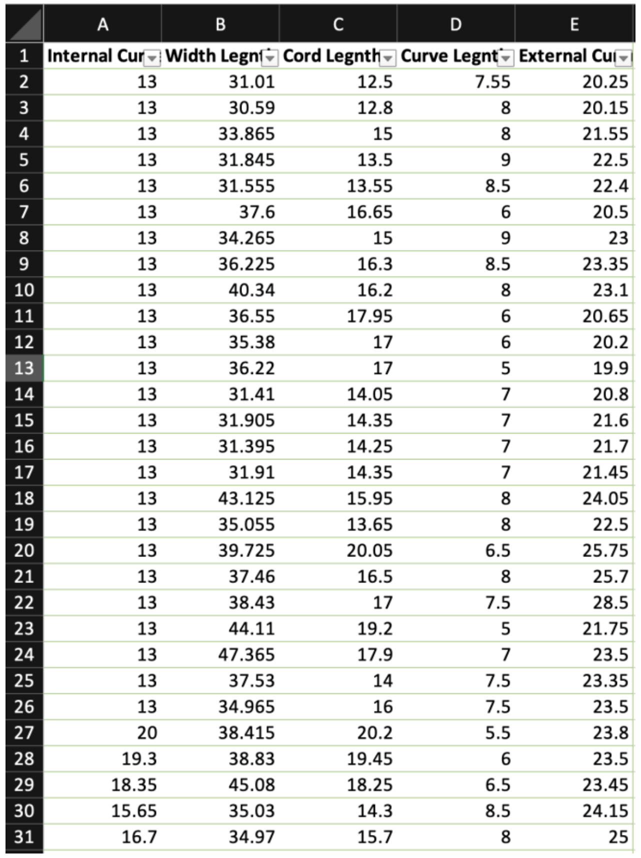
Optimizing Artificial Neural Network Parameters for Predicting Horseshoe External Curve Lengths

Alina Ahmed

Abstract

- An artificial neural network was built and optimised to find the external curve length of a horse shoe from the 4 predictor variables; Internal curve length width length, cord length and internal curve length
- The model was trained on a dataset of 219 samples, 4 data points were identified to be outliers and were modified so as not to bias the system with invalid data
- 4 main parameters were experimented with and evaluated using mean squared error and cross validation to build an optimised model which provides the most accurate predictions of the external curve length of a horseshoe. Graphs and plots were also generated to visually assess the impact of different parameters on the models performance.
- The optimised model had the following parameters as a result of multiple iterative processes:
  - Data pre-processing – data normalised with min-max scaling, outliers modified
  - Topology – 1 hidden layer with 9 hidden units
  - Activation function of hidden layer - radbas
  - Training function - traincgp
  - Data splicing ratio – 80:10:10 for training, validation and testing
- The final optimised model had a mean square error of: 0.0541 and a mean squared loss value from cross validation of: 0.0051



	A	B	C	D	E
1	Internal Cur	Width Legnt	Cord Legnth	Curve Legnt	External Cu
2	13	31.01	12.5	7.55	20.25
3	13	30.59	12.8	8	20.15
4	13	33.865	15	8	21.55
5	13	31.845	13.5	9	22.5
6	13	31.555	13.55	8.5	22.4
7	13	37.6	16.65	6	20.5
8	13	34.265	15	9	23
9	13	36.225	16.3	8.5	23.35
10	13	40.34	16.2	8	23.1
11	13	36.55	17.95	6	20.65
12	13	35.38	17	6	20.2
13	13	36.22	17	5	19.9
14	13	31.41	14.05	7	20.8
15	13	31.905	14.35	7	21.6
16	13	31.395	14.25	7	21.7
17	13	31.91	14.35	7	21.45
18	13	43.125	15.95	8	24.05
19	13	35.055	13.65	8	22.5
20	13	39.725	20.05	6.5	25.75
21	13	37.46	16.5	8	25.7
22	13	38.43	17	7.5	28.5
23	13	44.11	19.2	5	21.75
24	13	47.365	17.9	7	23.5
25	13	37.53	14	7.5	23.35
26	13	34.965	16	7.5	23.5
27	20	38.415	20.2	5.5	23.8
28	19.3	38.83	19.45	6	23.5
29	18.35	45.08	18.25	6.5	23.45
30	15.65	35.03	14.3	8.5	24.15
31	16.7	34.97	15.7	8	25

Snippet of the dataset provided

Introduction

- Horseshoes are metal plates fitted to horses for protection or extra traction, depending on the type of work the horse will be doing. Currently the correct size is found through manual fitting and adjusting and trial and error which can be time consuming and inefficient as wel as a waste of resources.
- The task was to create an ANN to predict the external curve of a horseshoe based on 4 predictor variables. (Internal curve length (cm), Width Length (mm), Cord Length (cm), Curve Length (cm)
- This problem was a multiple linear regression problem since there is more than one predictor variable. The importance of ensuring an accurate prediction of the external curve length is so we can optimize the design and fitting of horseshoes to ensure comfort and performance of horses.
- A mathematical equation could be derived and used in a program instead but a model is a more sophisticated solution as it can capture complex patterns and relationships in data and represent any noise or variability in the data which an equation could not so therefore it would be a better solution.
- The focus was on systematically exploring the impact of different parameters on the models predictive performance. The goal was to create a neural network model with the best combination of parameters to most accurately predict the external curve length. Through systematic experimentation to find the optimal:
  - Activation function
  - Training function
  - Network topology
  - Data Splicing ratio

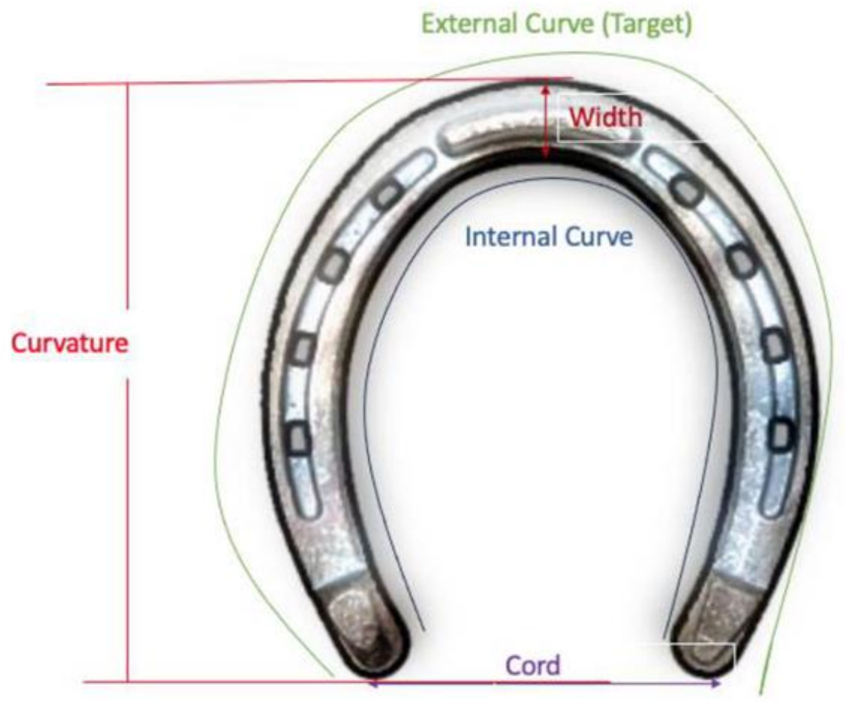


Diagram of the measurements of a horseshoe

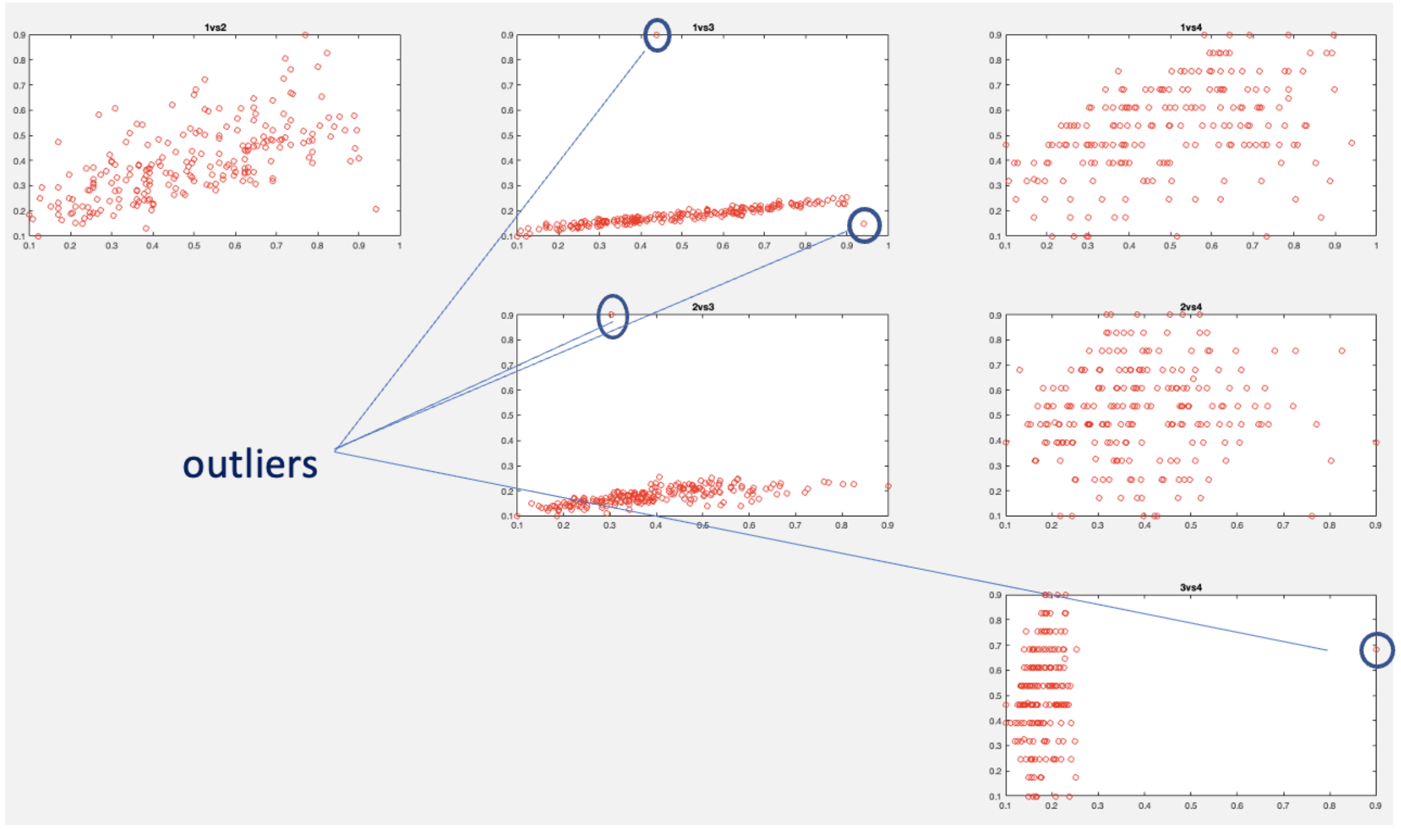
Data Pre-Processing

Dataset Normalization

- 219 samples
- Normalized by min/max scaling
- At the end of the method explored other methods of normalization

Outliers

- Generated graphs to visually spot outliers
- Changed the outliers to the mean value – there were not many, so did not remove due to small dataset



Plot generated in MATLAB of each column vs each column of all the rows of data. I’ve circled what I've identified to be outliers

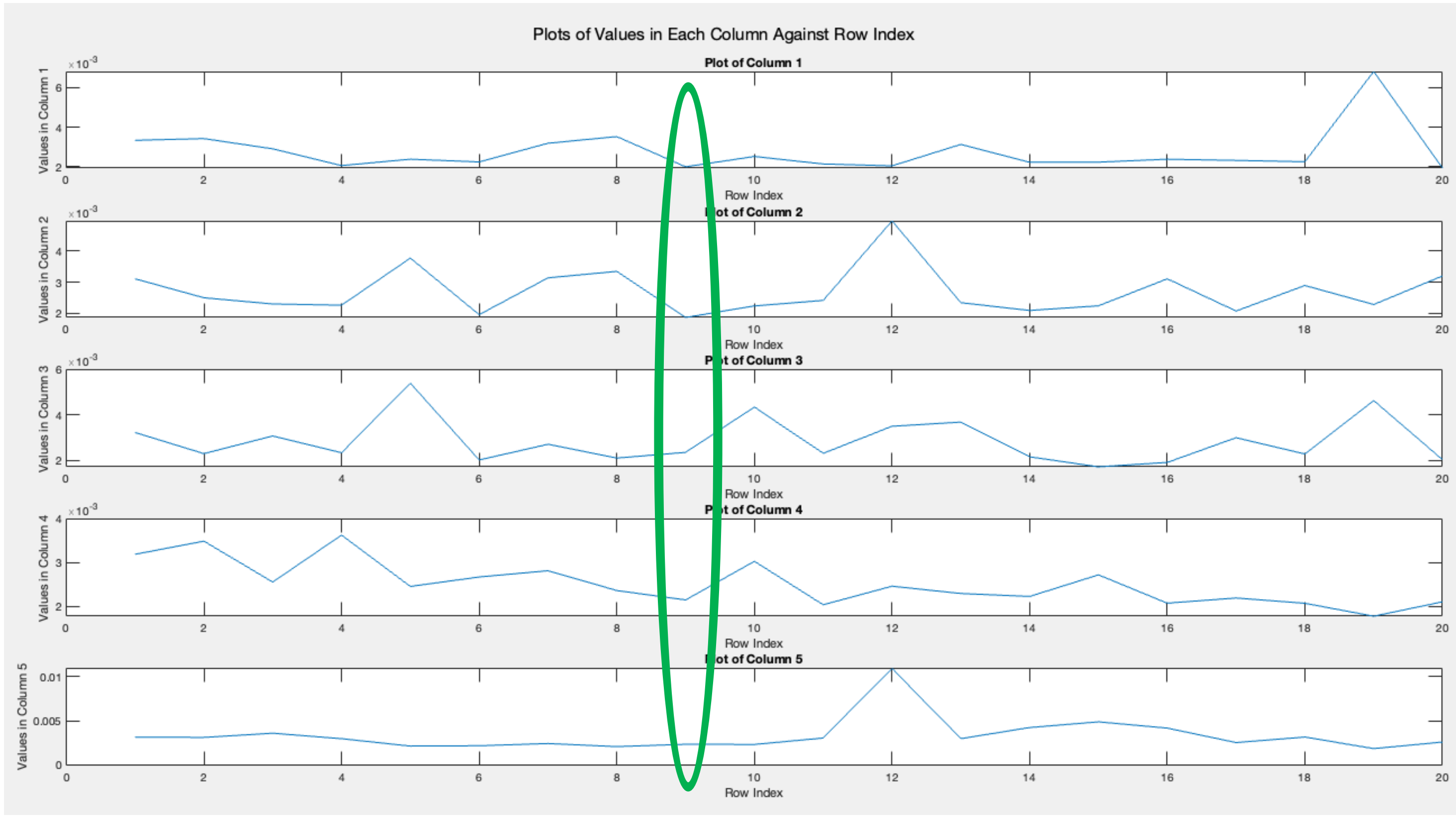
Topological Exploration

Conclusion

- Input layer –> 4 neurons
- Hidden Layer 1 –> 9 neurons
- Output layer –> 1 neuron

Methodology

- Input layers and Output layers were not interfered with
- Number of hidden layers was decided to be 1, experimentation of the number of hidden units
- Through multiple iterative processes, the number of neurons for the hidden layer was selected to be one which produced the smallest Mean Squared Error
    - Tested 1-20 neurons for the hidden layer
        - Used parallel processing to reduce training time
    - Repeated each test for a neuron 5 times
    - Identified which neuron contained the lowest MSE from a matrix of MSE values



Plot generated in MATLAB of each row index (number of hidden units) vs MSE values.

Suitable values for the best topology could be: 6(average mse=0.00222), 9(average mse=0.00214), 11(average mse=0.00236), 14(average mse=0.00258), 18(average mse=0.00256)

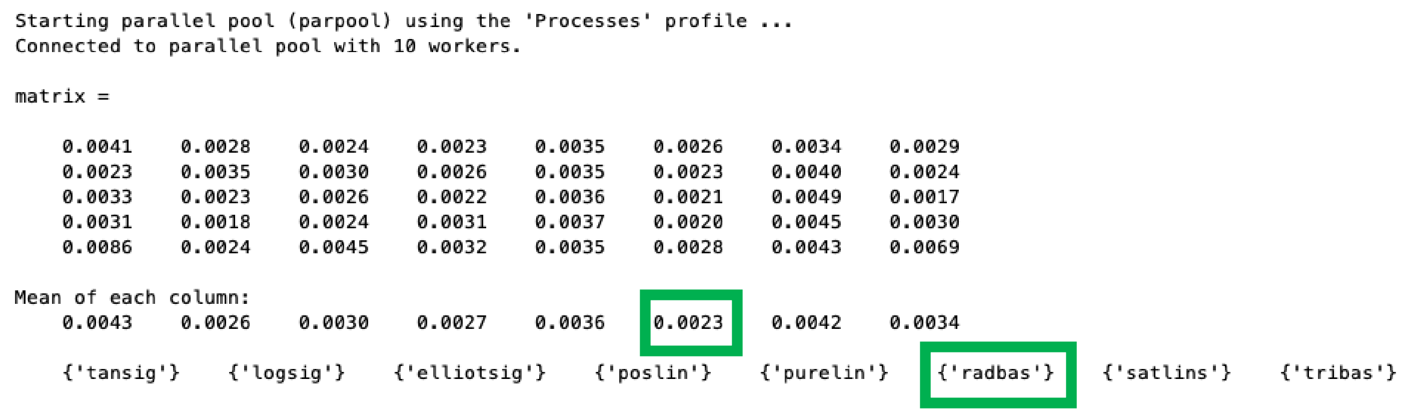
Activation (Transfer) Function Exploration

Outcome Decision

- Hidden Layer 1 activation function: radbas
- Output Layer activation function: purelin

Methodology

- Only experimented with the activation functions which were suitable for regression tasks: tansig, logsig, elliotsig, poslin, purelin, radbas, satlins, tribas
    - Functions such as softmax which are more suited to classification tasks were not tried



```
Starting parallel pool (parpool) using the 'Processes' profile ...
Connected to parallel pool with 10 workers.

matrix =

    0.0041    0.0028    0.0024    0.0023    0.0035    0.0026    0.0034    0.0029
    0.0023    0.0035    0.0030    0.0026    0.0035    0.0023    0.0040    0.0024
    0.0033    0.0023    0.0026    0.0022    0.0036    0.0021    0.0049    0.0017
    0.0031    0.0018    0.0024    0.0031    0.0037    0.0020    0.0045    0.0030
    0.0086    0.0024    0.0045    0.0032    0.0035    0.0028    0.0043    0.0069

Mean of each column:
    0.0043    0.0026    0.0030    0.0027    0.0036    0.0023    0.0042    0.0034

    {'tansig'}    {'logsig'}    {'elliotsig'}    {'poslin'}    {'purelin'}    {'radbas'}    {'satlins'}    {'tribas'}
```

Matrix generated where each column represents a different activation function tried and the MSE error result. The average of each column is then calculated to show the lowest MSE error value is for radbas (0.0023)

- Through multiple iterative processes, the activation function was selected to be one which produced the smallest Mean Squared Error
    - Tested 8 different activation functions
    - Repeated each test for each different function 5 times and then found the average
    - Identified which function produced the lowest average MSE value

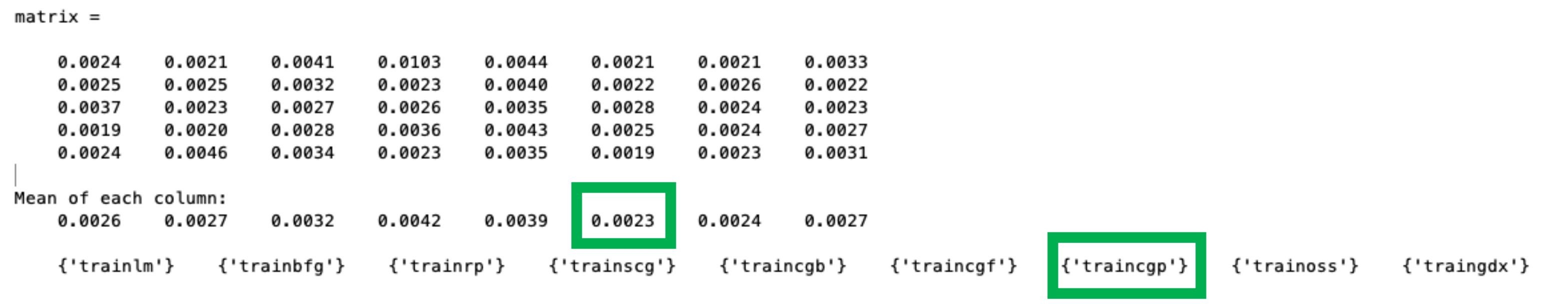
Training Function Exploration

Outcome Decision

- Training function: traincgp

Methodology

- Experimented with all 9 matlab training functions: trainlm, trainbfg, trainrp, trainscg, traincgb, traincgf, traincgp, trainoss, traingdx
- Through multiple iterative processes, the training function was selected to be one which produced the smallest Mean Squared Error
    - Tested 9 different training functions
    - Repeated each test for each different function 5 times and then found the average
    - Identified which function produced the lowest average MSE value



```
matrix =

    0.0024    0.0021    0.0041    0.0103    0.0044    0.0021    0.0021    0.0033
    0.0025    0.0025    0.0032    0.0023    0.0040    0.0022    0.0026    0.0022
    0.0037    0.0023    0.0027    0.0026    0.0035    0.0028    0.0024    0.0023
    0.0019    0.0020    0.0028    0.0036    0.0043    0.0025    0.0024    0.0027
    0.0024    0.0046    0.0034    0.0023    0.0035    0.0019    0.0023    0.0031

Mean of each column:
    0.0026    0.0027    0.0032    0.0042    0.0039    0.0023    0.0024    0.0027

    {'trainlm'}    {'trainbfg'}    {'trainrp'}    {'trainscg'}    {'traincgb'}    {'traincgf'}    {'traincgp'}    {'trainoss'}    {'traingdx'}
```

Matrix generated where each column represents a different training function tried and the MSE error result. The average of each column is then calculated to show the lowest MSE error value is for traincgp (0.0023)

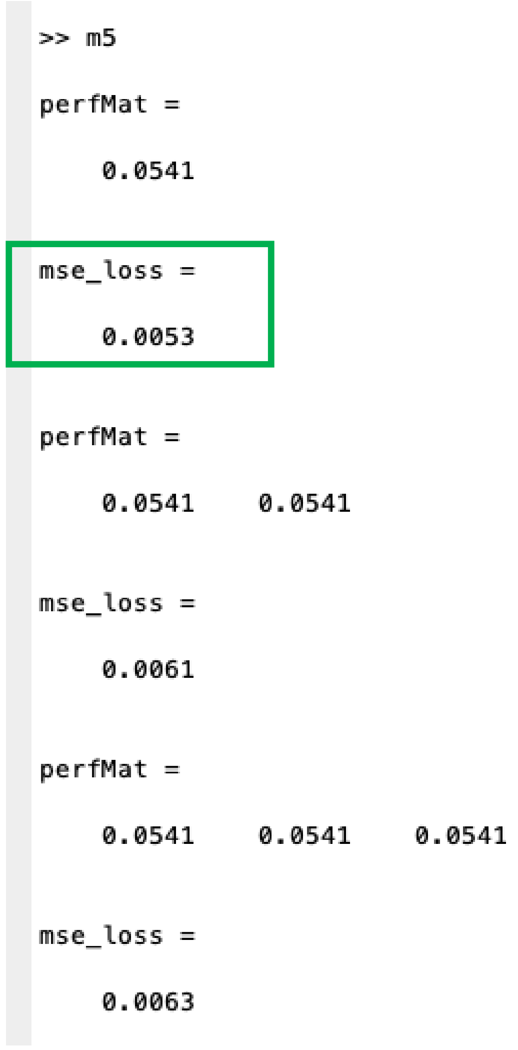
Data Splicing Ratio Exploration

Conclusion

- Data splitting ratio: 80 for training, 10 for validation and 10 for testing
- Data was split into the following 3 categories:
  - Training set: data for the network to learn patterns and relationships and update its weights and biases
  - Validation set: data to identify and minimize overfitting so the network does not learn the training data and can perform well with unseen data
  - Test set: data that is unseen and assesses the final performance of the trained model on its ability to generalize new data

Methodology

- Experimented with 3 different common ratios of splitting of data:
  - 80:10:10
  - 70:15:15
  - 60:20:20
- Evaluated the performance of the model with the different ratios using MSE
- MSE indicated the same error value for each ratio so cross validation was also used as an evaluation tool
- The lowest value indicated by cross validation was selected as the optimal ratio for data splitting



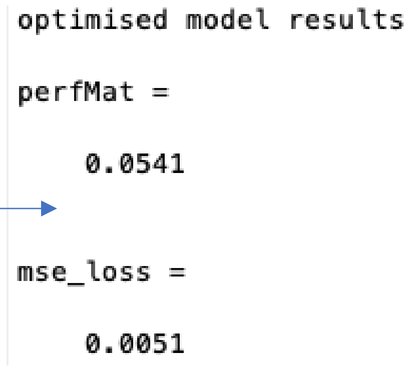
```
>> m5

perfMat =

    0.0541


mse_loss =

    0.0053


perfMat =

    0.0541    0.0541


mse_loss =

    0.0061


perfMat =

    0.0541    0.0541    0.0541


mse_loss =

    0.0063
```

Results and Conclusions

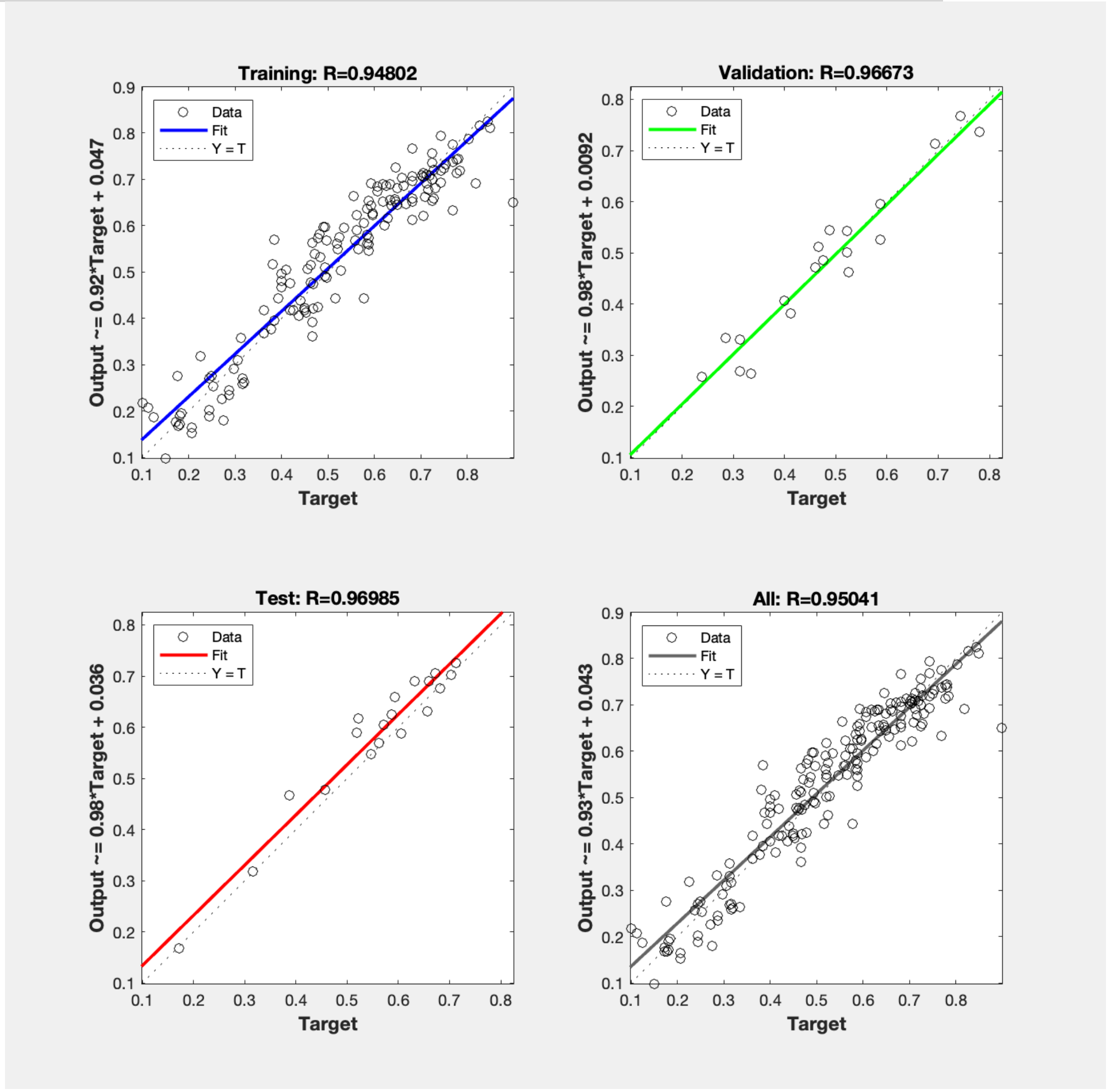
- The model's key limitation is the size of the dataset. The dataset contains only 219 samples, a larger sample size would be more representative and better at allowing the model to train.
- A recommendation for future work would be to explore the results of deriving a mathematical equation to calculate the external curve length compared to the predictions of a model and observe which is more accurate.
    - A mathematical solution may be more efficient, however, a model is able to represent noise and variability in data

Result of mean squared error

Result of cross validation



```
optimised model results

perfMat =

    0.0541

mse_loss =

    0.0051
```


Training: R=0.94802
Output ~= 0.92*Target + 0.047
Validation: R=0.96673
Output ~= 0.98*Target + 0.0092
Test: R=0.96985
Output ~= 0.98*Target + 0.036
All: R=0.95041
Output ~= 0.93*Target + 0.043
Target
Data
Fit
Y = T