Neural Networks

Project II

By: Rohith Lingala,

Harshith Kasthuri,

Charan Reddy Mannuru,

James Notoma,

Josh Kolesar

Guided by-

Dr. John Miller
CSCI 6360- Data Science II

Oct 31st, 2024

Introduction

This report focuses on the same datasets as project 1. You will notice there are very few changes in data preprocessing and EDA as a result. However, this project focuses on 2L, 3L, and XL neural nets. Across the board performance of these models were better than the models used in project 1. For our pythonic neural nets, we mostly used Keras from Tensorflow due to its more streamlined implementation into datasets, although there was some use of Pytorch. Scala neural nets were optimized according to each dataset. Feel free to reach out to any of the group members listed with any questions or concerns.

Airfoil

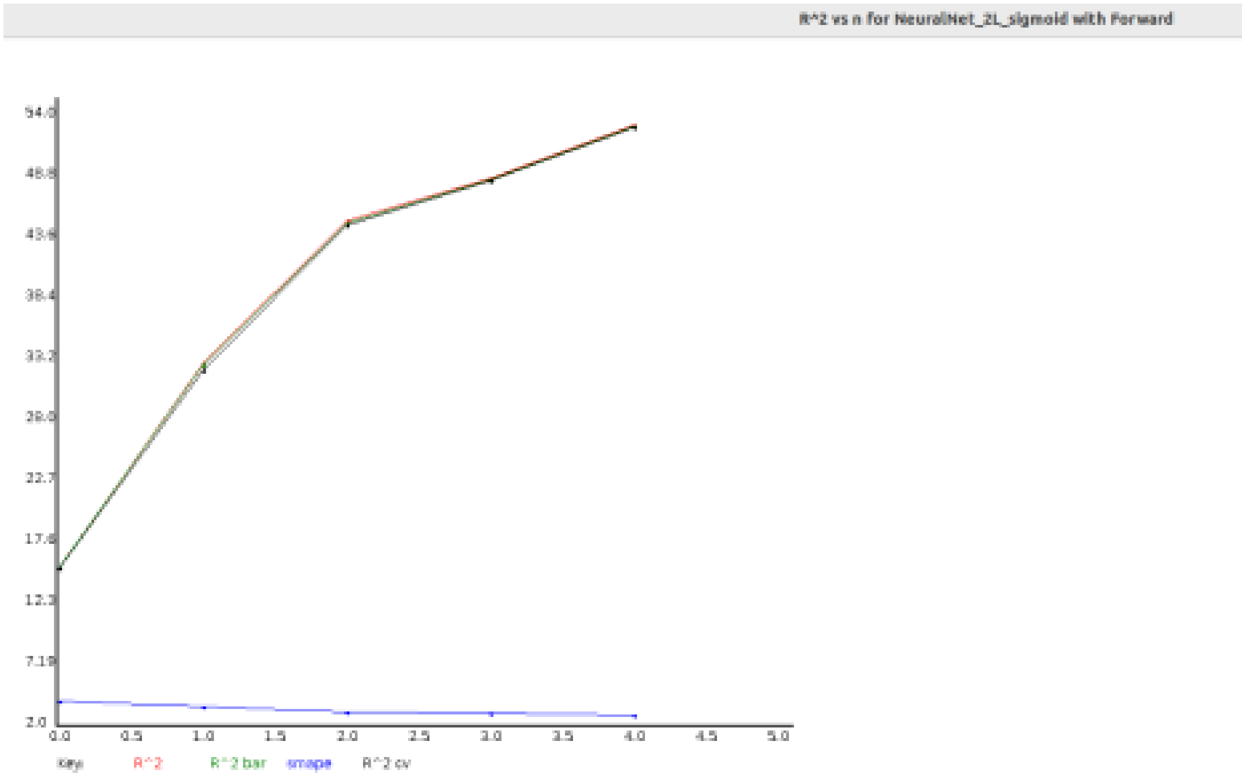
Data preprocessing:

EDA:

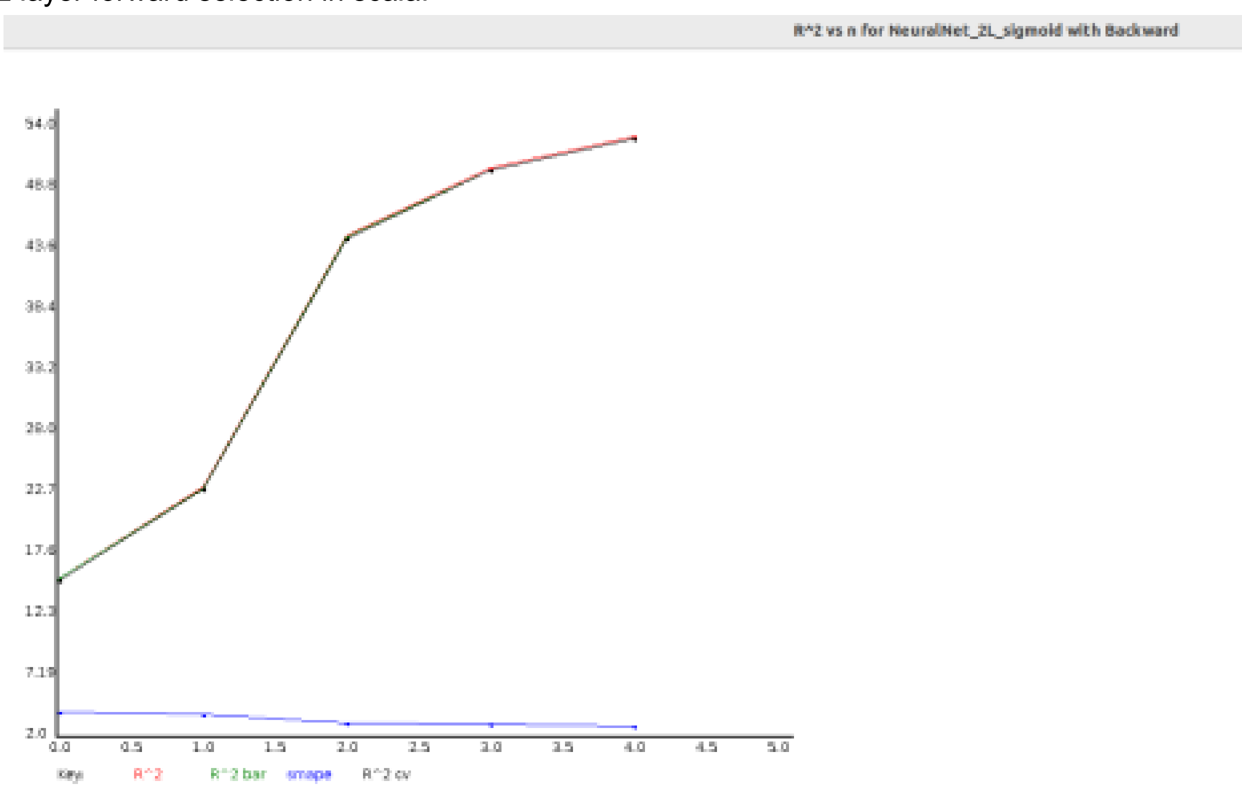
All data preprocessing and EDA performed for this dataset was the same as described in project 1. Please see our previous report for details. The only exception is that I found in building neural nets in Python, performance was dramatically improved when data was normalized beforehand.

Feature Selection:

Backwards and forwards feature selection in scala show steep improvements with the addition of each feature. There seems to be no plateau as we see in the linear regression and variants in the previous project. From my understanding of how neural nets perform this makes sense. As the neural net iterates readjusting weights for each feature and through each node it will passively maximize the effectiveness of each feature's ability to predict our output. Theoretically, only features that have no relation would reduce or slow down performance.



2 layer forward selection in scala.



2 layer backward selection in scala.

Splitting Data Information:

In scala our train and test dataset was split 80-20. K-fold split was also divided into 5 and again 80-20 tnt split. Same thing for python.

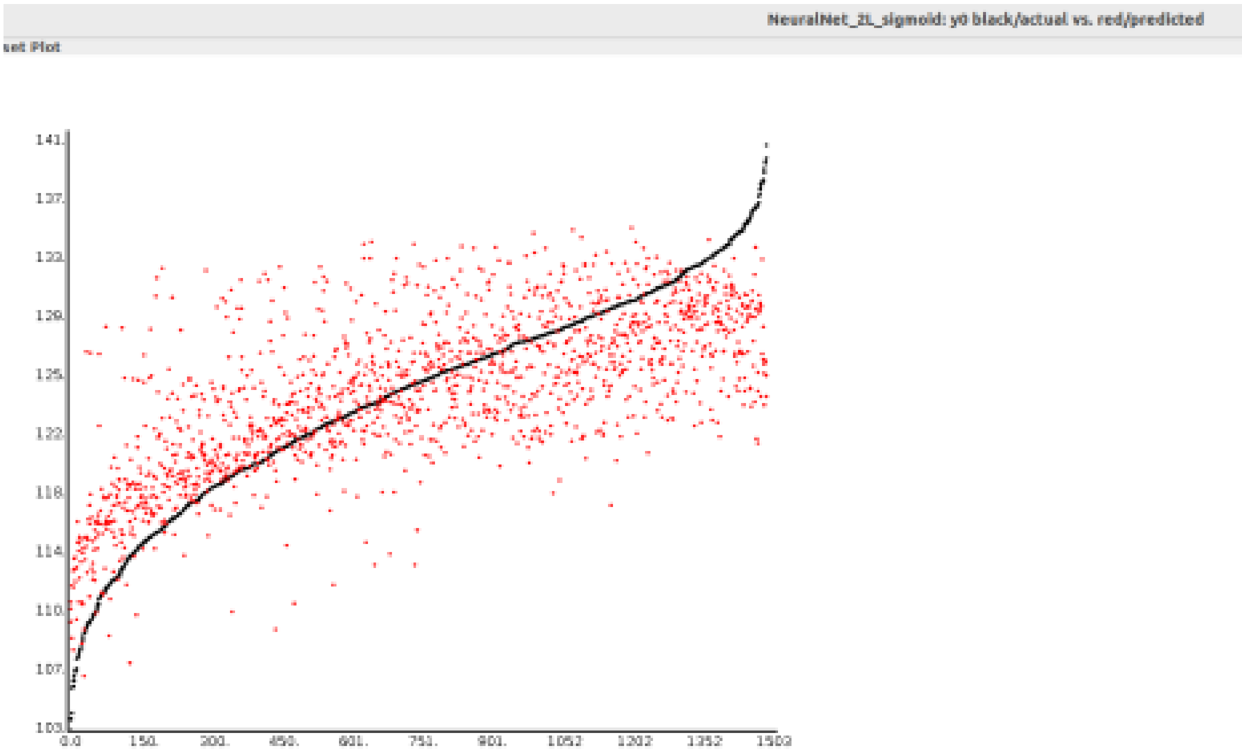
Discussion of Results:

Our two layer neural nets in scala performed best using the sigmoid function. R-squared, r-squared bar, and SMAPE values were 0.533, 0.531, and 2.9 respectively. Tanh performed almost identically with r-squared, r-squared bar, and SMAPE of 0.533, 0.531, and 2.9 respectively. eLU, reLU, and lreLU all performed similarly and slightly worse than sigmoid and tanh. They contained r-squared, r-squared bar, and SMAPE values of 0.516, 0.514, and 3.0 respectively. Because of the better performance of sigmoid which is slightly simpler implementation than tanh we decided to continue 3L and XL analysis with sigmoid.

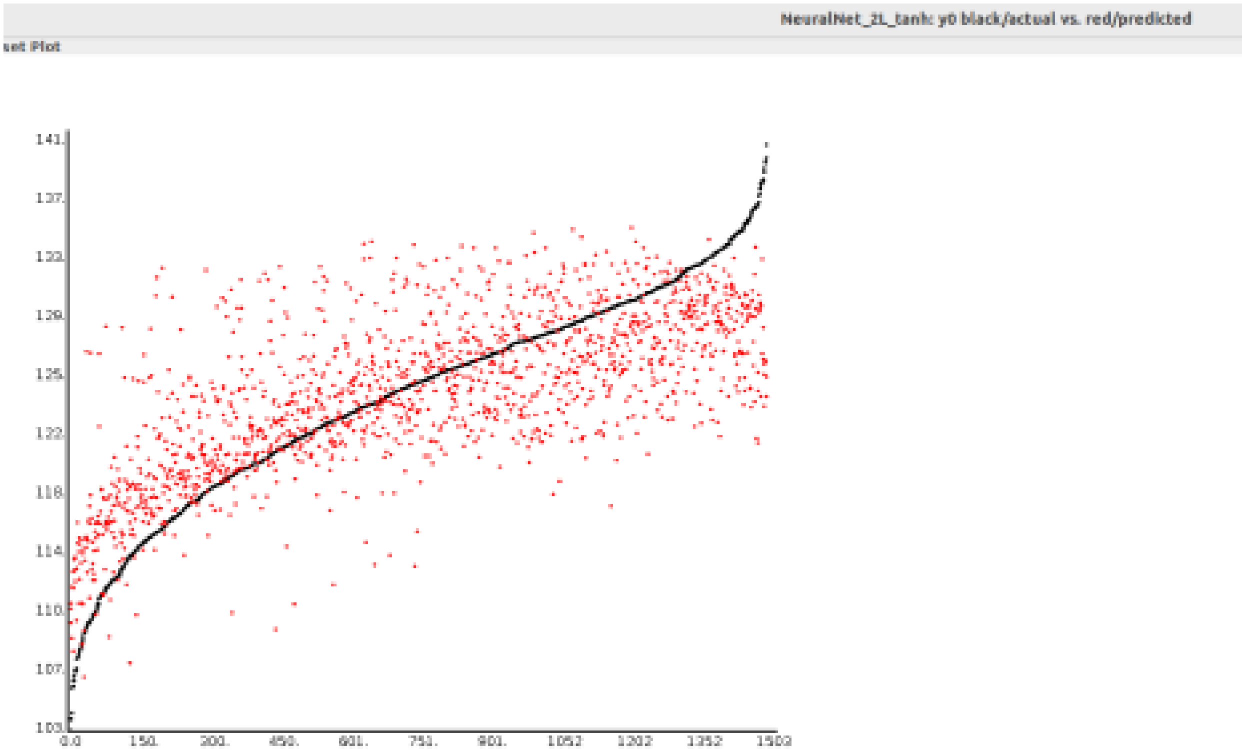
```
[info] modelName mn = NeuralNet_2L_sigmoid
[info] -------------------------------------------------------------------------
[info] hparameter hp = HyperParameter (HashMap(lambda -> (0.0
1,0.01), maxEpochs -> (400,4
00), eta -> (0.1,0.1), nu -> (0.9,0.9), upLimit -> (4,4), beta -> (0.9,0.9), bSize -> (20,20)))
[info] -------------------------------------------------------------------------
[info] features fn
= Array(intercept, Fre
quency, An
gle of attack, Chord length, Free-stream velocity, Suction side displacement thickness)
[info] -------------------------------------------------------------------------
[info] parameter bb = Array(b.w =
[info] MatrixD (-1.45808,
[info] -0.777800,
[info] -0.309472,
[info] -0.310131,
[info] 0.116111,
[info] -0.222863)
[info] b.b = null)
[info] -------------------------------------------------------------------------
[info] fitMap qof =
[info] rSq -> VectorD(0.532686)
[info] rSqBar -> VectorD(0.531125)
[info] sst -> VectorD(71482.4)
[info] sse -> VectorD(33404.7)
[info] mse0 -> VectorD(22.2254)
[info] rmse -> VectorD(4.71438)
[info] mae -> VectorD(3.62733)
[info] dfm -> VectorD(5.00000)
[info] df -> VectorD(1497.00)
[info] fStat -> VectorD(341.282)
[info] aic -> VectorD(-4451.24)
[info] bic -> VectorD(-4419.35)
```

[info] mape -> VectorD(2.91464)
[info] smape -> VectorD(2.90617)
[info] --

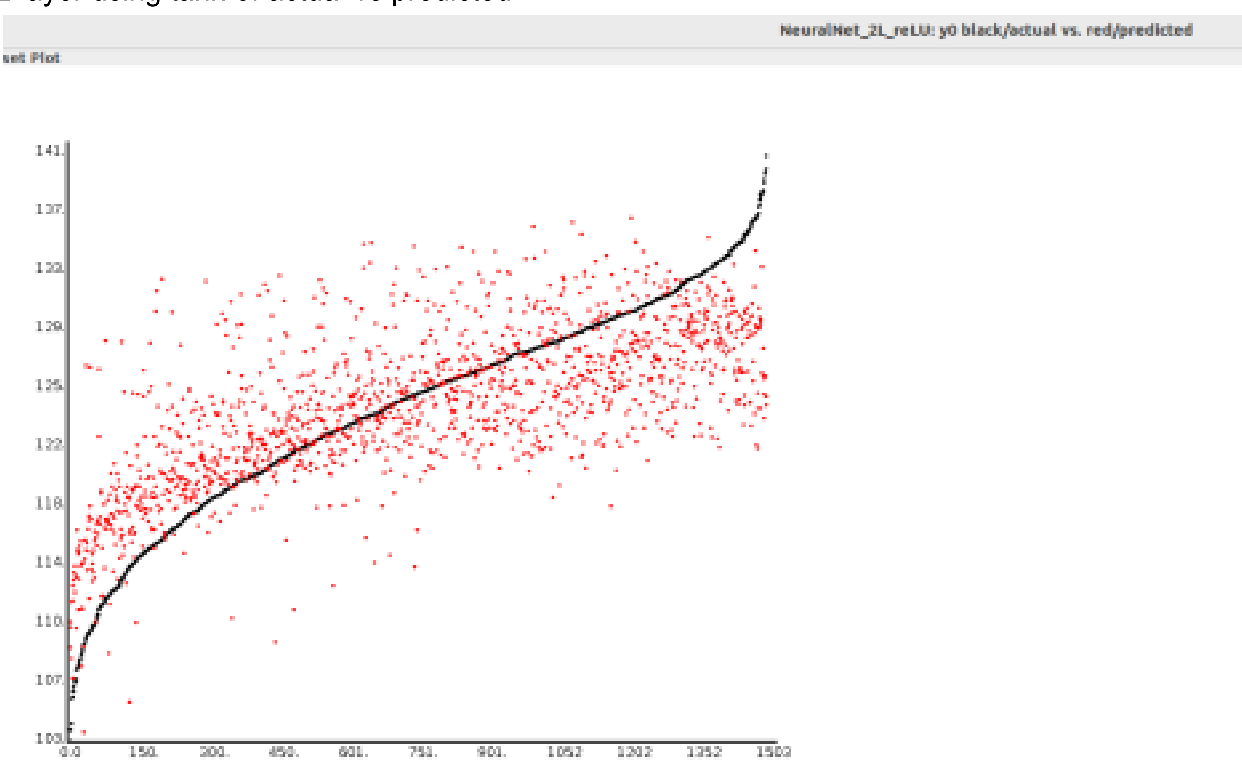
Table of two layer neural net summary statistics with sigmoid.



2 layer using sigmoid of actual vs predicted.



2 layer using tanh of actual vs predicted.



2 layer using relu of actual vs predicted values.

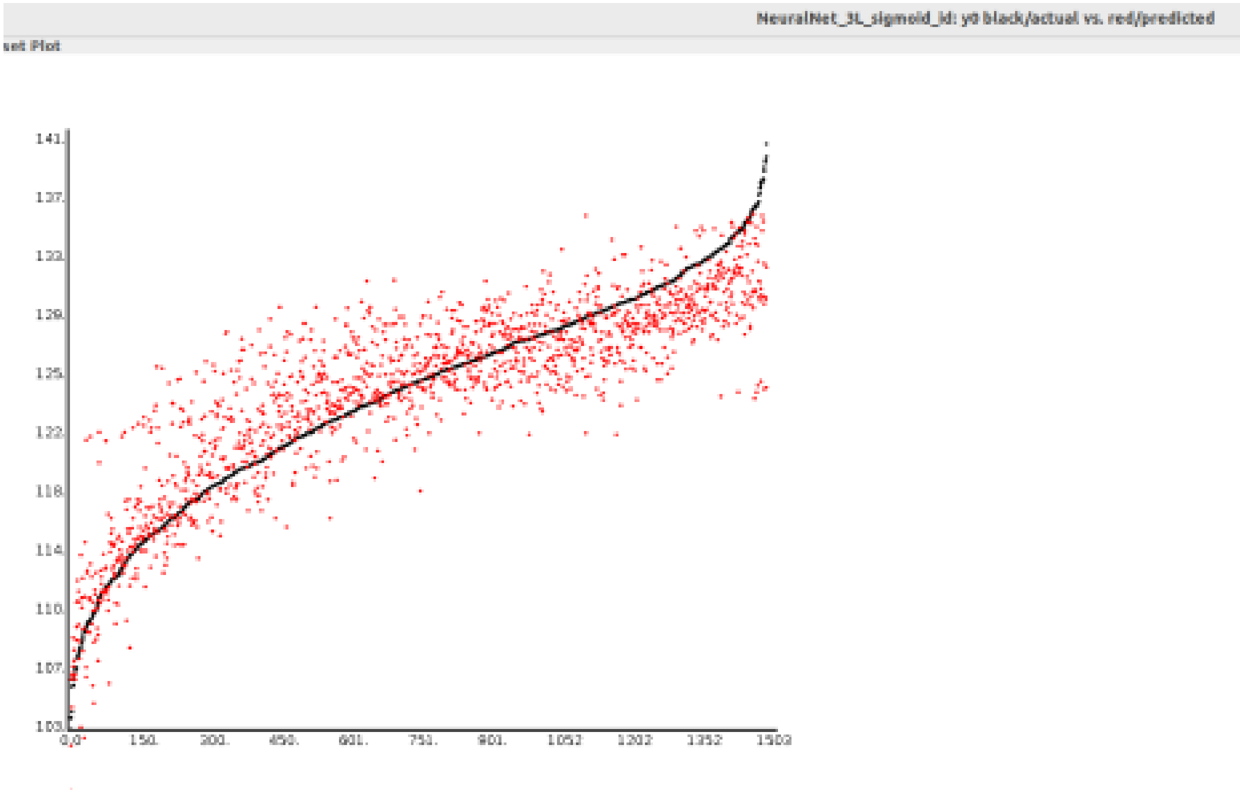
Moving over to our three layer neural net in scala we had a huge improvement in performance. Looking at our actual vs predicted values in graphs we can see predicted values are clustered much tighter around actual values. R-squared, r-squared bar, and SMAPE values improved to 0.676, 0.675, and 2.3. These values were recorded when bSize was 20 and nB was 75. When nB was reduced to 60 r-squared, r-squared bar, and SMAPE values improved to 0.783, 0.783, and 1.9 respectively.

```
[info] --------------------------------------------------------------------------
[info] fitMap qof =
[info] rSq -> VectorD(0.675911)
[info] rSqBar -> VectorD(0.674829)
[info] sst -> VectorD(71482.4)
[info] sse -> VectorD(23166.7)
[info] mse0 -> VectorD(15.4136)
[info] rmse -> VectorD(3.92602)
[info] mae -> VectorD(2.92964)
[info] dfm -> VectorD(5.00000)
[info] df -> VectorD(1498.00)
[info] fStat -> VectorD(624.837)
[info] aic -> VectorD(-4176.21)
[info] bic -> VectorD(-4144.31)
[info] mape -> VectorD(2.35047)
[info] smape -> VectorD(2.33923)
[info] --------------------------------------------------------------------------
```

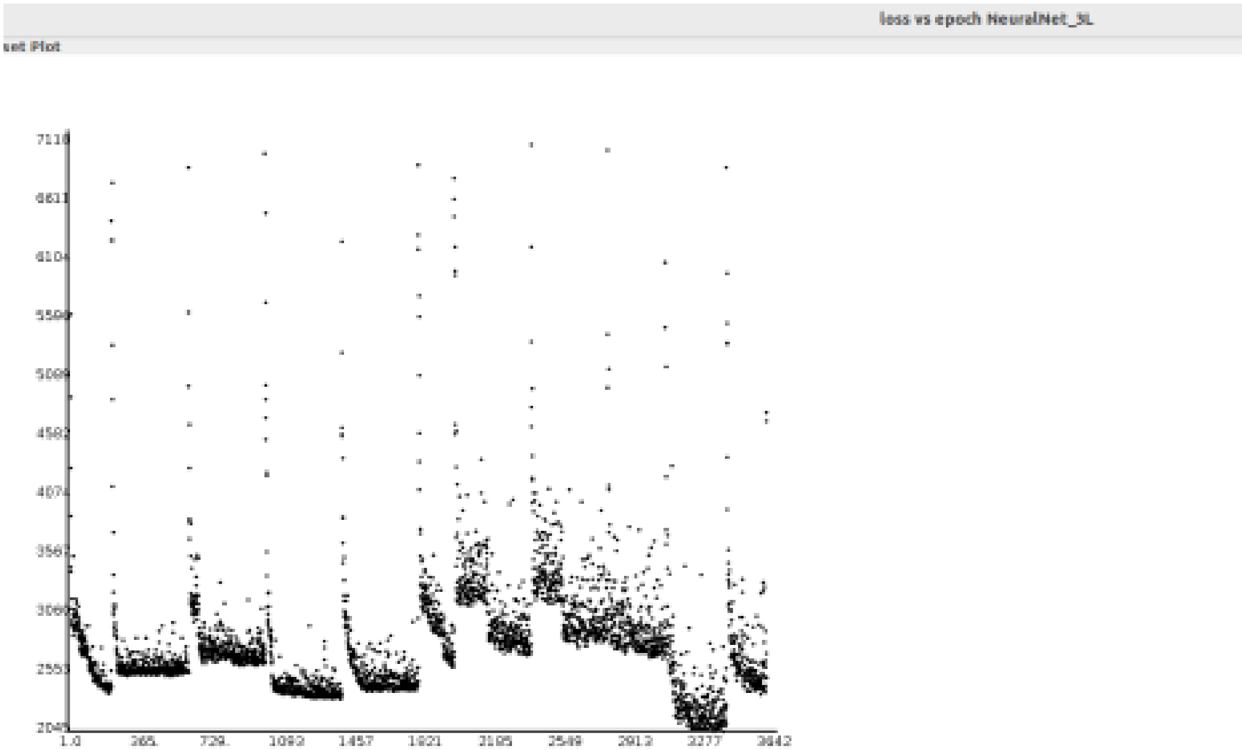
Table of summary statistics of our 3 layer neural net with sigmoid.

```
[info] optimize3: bSize = 20, nB = 60
[info] ending epoch = (11165.834723430013,400)
[info] rSq -> VectorD(0.783368)
[info] rSqBar -> VectorD(0.782645)
[info] sst -> VectorD(13805.5)
[info] sse -> VectorD(2990.71)
[info] mse0 -> VectorD(9.96904)
[info] rmse -> VectorD(3.15738)
[info] mae -> VectorD(2.38720)
[info] dfm -> VectorD(5.00000)
[info] df -> VectorD(1498.00)
[info] fStat -> VectorD(1083.39)
[info] aic -> VectorD(-770.942)
[info] bic -> VectorD(-748.720)
[info] mape -> VectorD(1.91188)
[info] smape -> VectorD(1.90983)
```

Table of summary statistics of our 3 layer neural net with sigmoid.



3 layer using sigmoid id of actual vs predicted values.



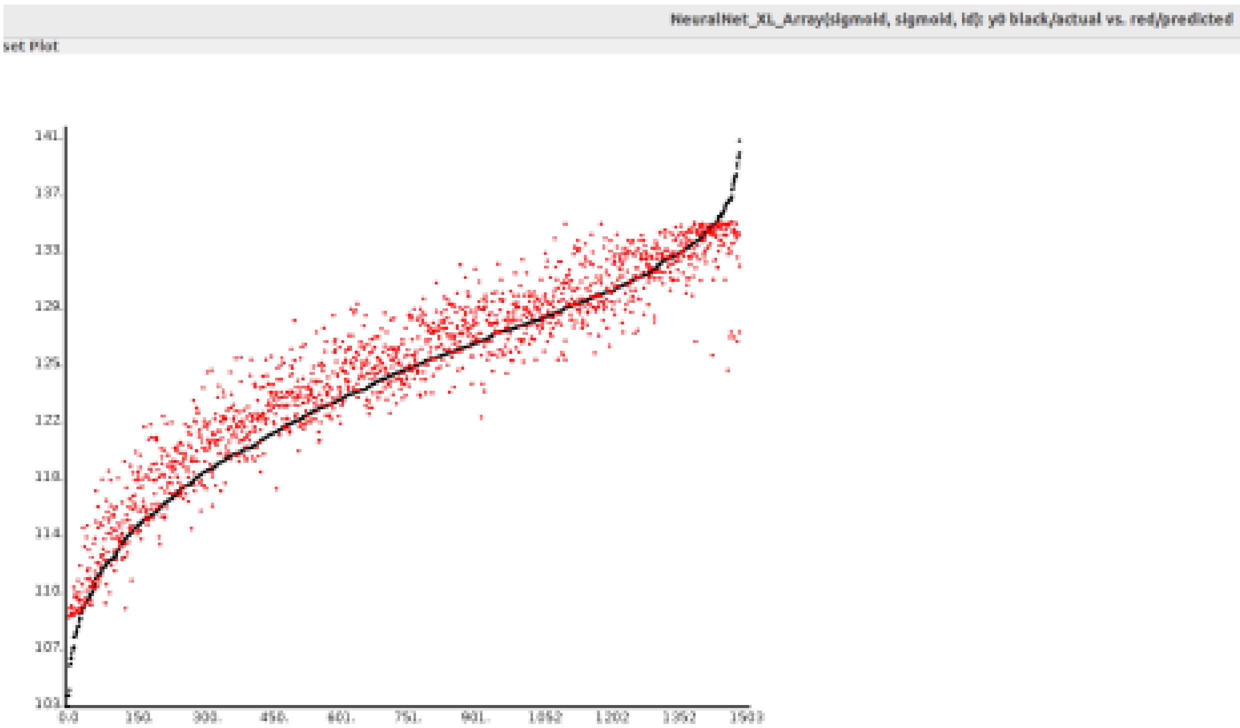
Loss vs epoch in our 3 layer.

Again moving to our XL neural net we see another hike in performance. Looking at our graphs and comparing them to previous ones, we can see that our predicted values are located much more tightly around the actual values. In our cross validated model we had a mean r-squared value of 0.922, mean r-squared bar of 0.922, and mean SMAPE of 1.2. These values are a drastic improvement from our two-layer model and shows how much more accurate models can be developed with the addition of layers in scala. The only thing of concern from this model is our epoch vs loss graph. The huge drop in loss values may indicate some overfitting taking place in this model.

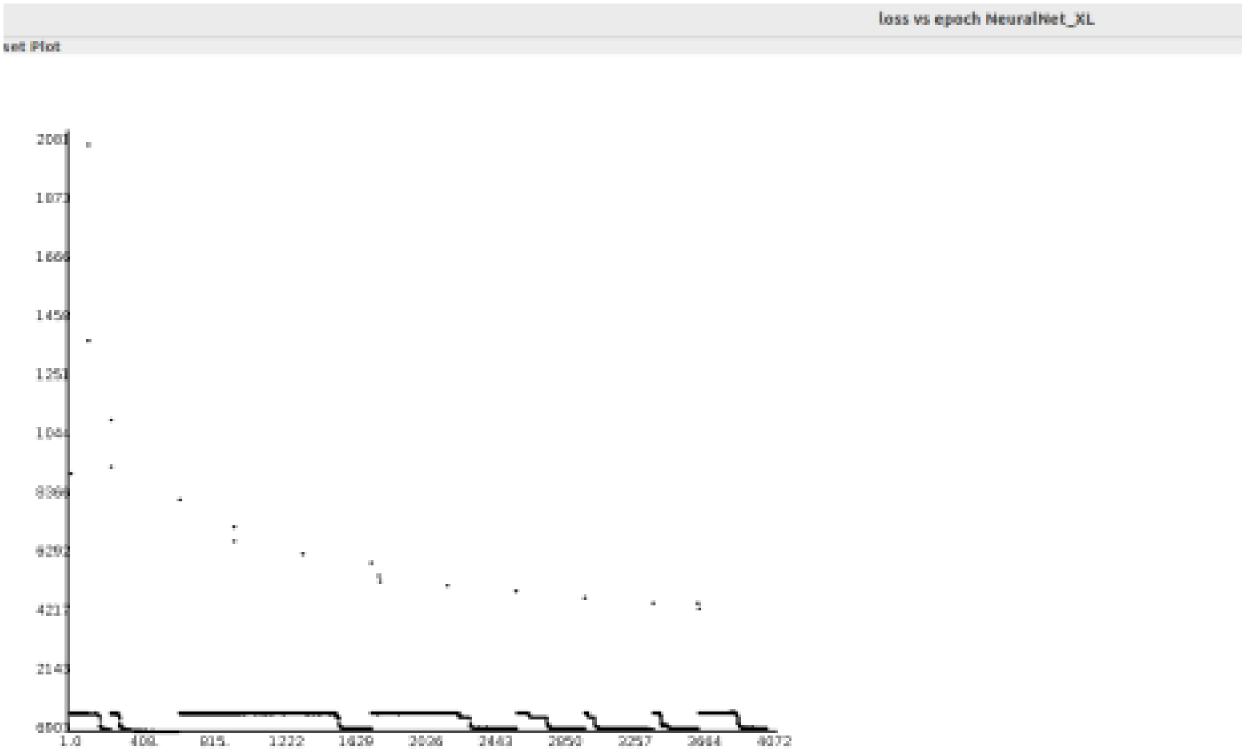
```
[info] | name | num | min | max | mean | stdev | interval |
[info] ------------------------------------------------------------------------------------------
[info] | rSq | 5 | 0.896 | 0.942 | 0.922 | 0.019 | 0.024 |
[info] | rSqBar | 5 | 0.895 | 0.941 | 0.922 | 0.019 | 0.024 |
[info] | sst | 5 | 13493.068 | 14926.373 | 14232.414 | 634.918 | 788.509 |
[info] | sse | 5 | 788.501 | 1552.327 | 1111.443 | 293.369 | 364.337 |
[info] | mse0 | 5 | 2.628 | 5.174 | 3.705 | 0.978 | 1.214 |
[info] | rmse | 5 | 1.621 | 2.275 | 1.912 | 0.250 | 0.311 |
[info] | mae | 5 | 1.245 | 1.607 | 1.444 | 0.144 | 0.178 |
[info] | dfm | 5 | 5.000 | 5.000 | 5.000 | 0.000 | 0.000 |
[info] | df | 5 | 1498.000 | 1498.000 | 1498.000 | 0.000 | 0.000 |
[info] | fStat | 5 | 2567.596 | 4827.244 | 3731.812 | 958.467 | 1190.327 |
[info] | aic | 5 | -859.299 | -851.269 | -854.664 | 3.084 | 3.830 |
```

[info] | bic | 5 | -837.076 | -829.046 | -832.441 | 3.084 | 3.830 |
[info] | mape | 5 | 0.998 | 1.284 | 1.160 | 0.114 | 0.141 |
[info] | smape | 5 | 0.999 | 1.285 | 1.159 | 0.114 | 0.141 |
[info] --

Table of our cross-validated XL model.



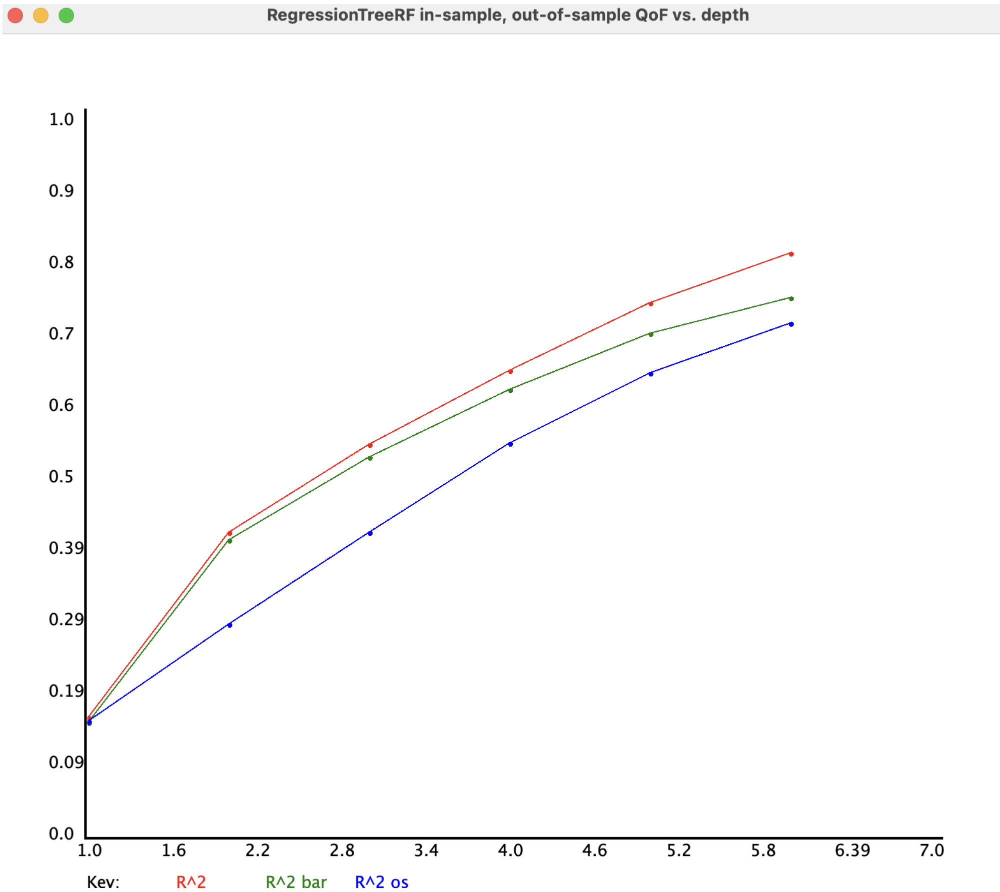
Actual vs predicted values with the use of sigmoid and id in XL neural net.



Loss vs epoch in our XL neural net.

DEBUG @ **RegressionTreeRF**.train: for tree6 ===
()
LinkedHashMap(**rSq -> 0.721805**, rSqBar -> 1.970447, sst -> 13805.490439, sse -> 3840.613948, sde -> 3.579554, mse0 -> 12.802046, rmse -> 3.577995, mae -> 2.833136, dfm -> 386.000000, df -> -86.000000, fStat -> -0.578073, aic -> -48.005993, bic -> 1385.357825, mape -> 2.275591, smape -> 2.273016)

Random Forest Regressor



LinkedHashMap(rSq -> 0.721805, rSqBar -> 1.970447, sst -> 13805.490439, sse -> 3840.613948, sde -> 3.579554, mse0 -> 12.802046, rmse -> 3.577995, mae -> 2.833136, dfm -> 386.000000, df -> -86.000000, fStat -> -0.578073, aic -> -48.005993, bic -> 1385.357825, mape -> 2.275591, smape -> 2.273016)

Python

Note: In this report XL is 3 Hidden layer neural net.

2L layer Neural Net:

The Best performance combination was found with SELU activation function and Stochastic Gradient Descent Optimizer.

Learning Rate: 0.0001.
Batch Size: 128.
Input Layer Size:5.

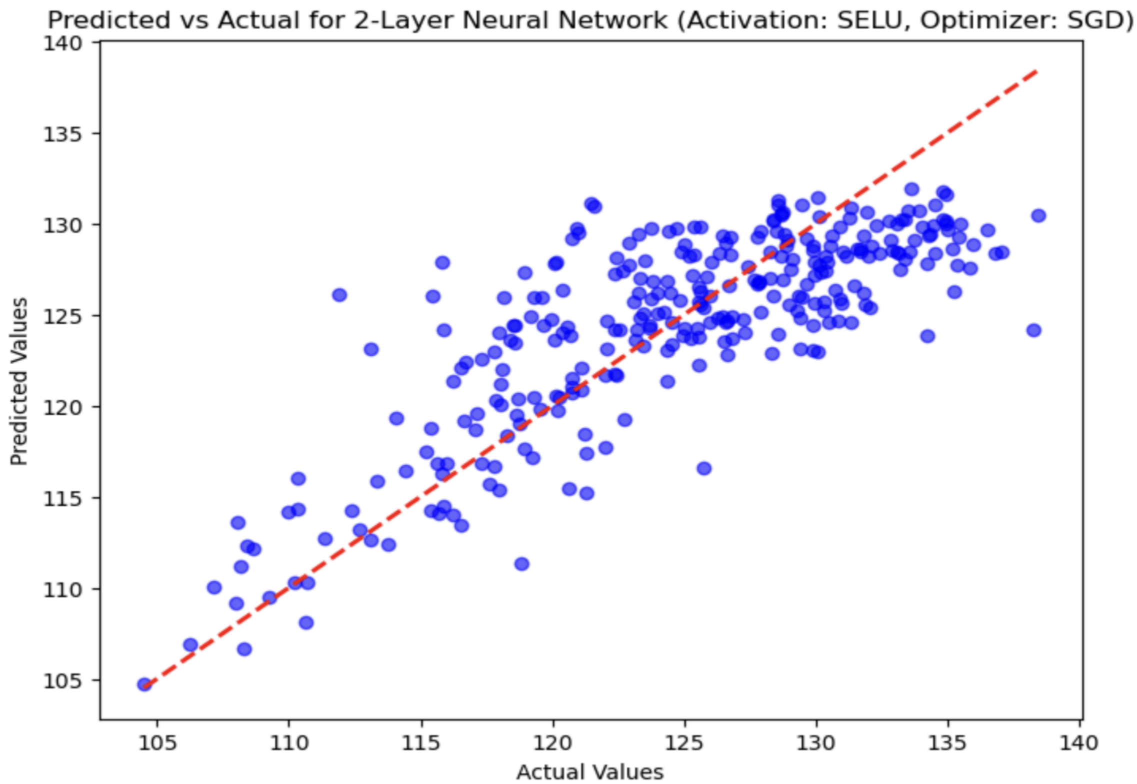
Hidden Layer Size:256.

```
Testing Activation: SELU, Optimizer: SGD
training 2L net
Epoch [100/100], Loss: 18.344952
starting evaluation
5-fold cross-validation evaluation
Epoch [100/100], Loss: 16.253007
Epoch [100/100], Loss: 14.892809
Epoch [100/100], Loss: 14.435249
Epoch [100/100], Loss: 12.853898
Epoch [100/100], Loss: 12.734273
```

Metrics:

The best metric is Cross-validation R2.

```
2-Layer Neural Network:
  Best R²: 0.6960
  Best Metric: Cross-Validation R²
  Activation and Optimizer: Activation: SELU, Optimizer: SGD
  Metrics:
    In-Sample MSE: 18.644153594970703
    In-Sample RMSE: 4.317887783050537
    In-Sample R²: 0.6026
    Validation MSE: 17.83548927307129
    Validation RMSE: 4.223208427429199
    Validation R²: 0.6440
    Cross-Validation MSE: 14.292463302612305
    Cross-Validation RMSE: 3.7420883178710938
    Cross-Validation R²: 0.6960
```



3L Layer Neural Net:
The Best performance combination was found with ReLU activation function and Stochastic Gradient Descent Optimizer.

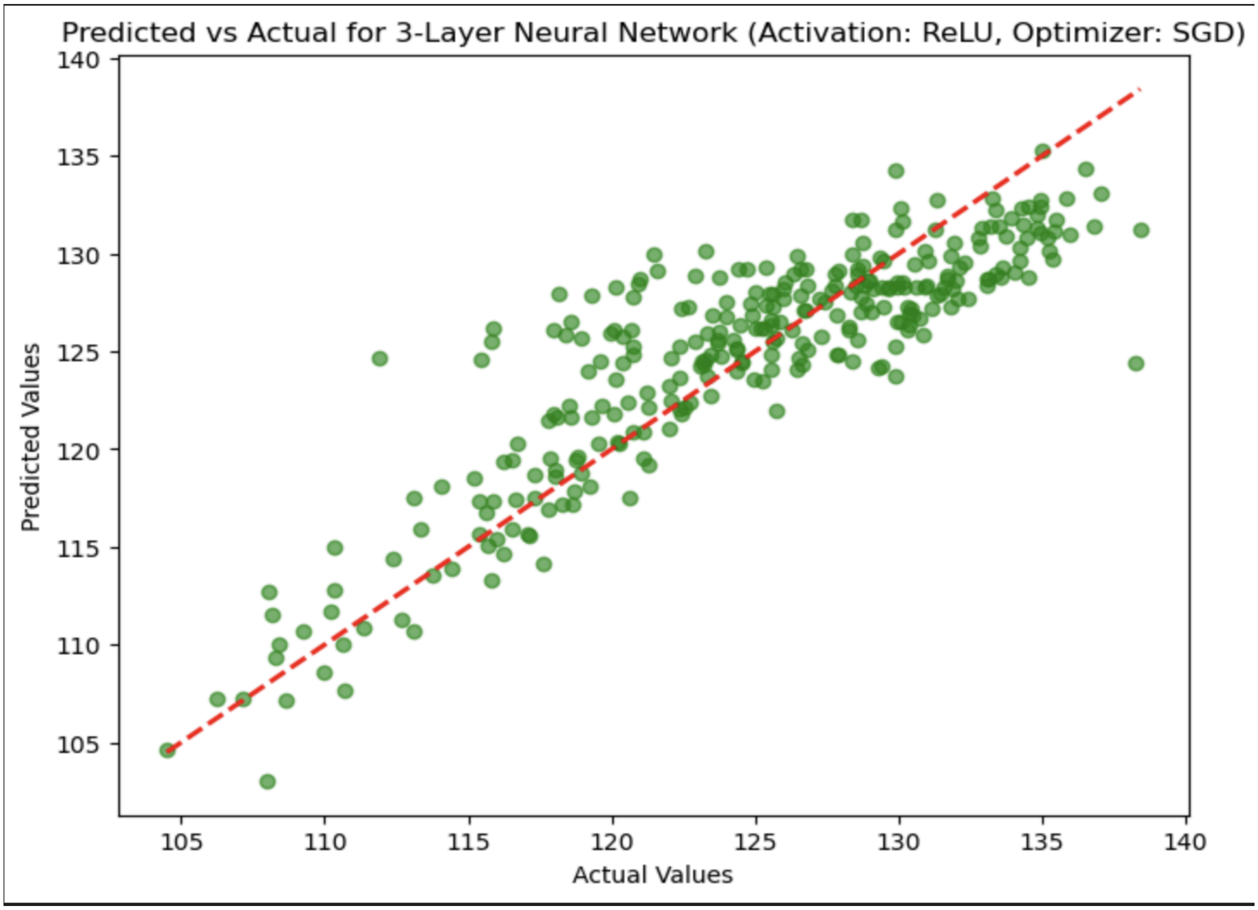
Learning Rate: 0.0001
Batch Size:128
Input Layer Size:5.
Hidden Layer Size:256.

```
training 3l net
Epoch [100/100], Loss: 11.666045

5-fold cross-validation evaluation for 3l
Epoch [100/100], Loss: 8.531790
Epoch [100/100], Loss: 7.000087
Epoch [100/100], Loss: 6.094772
Epoch [100/100], Loss: 5.181322
Epoch [100/100], Loss: 4.520917
```

Metrics:
The Best metric is 3-Layer Neural Network.

```
3-Layer Neural Network:
  Best R²: 0.8630
  Best Metric: Cross-Validation R²
  Activation and Optimizer: Activation: ReLU, Optimizer: SGD
  Metrics:
    In-Sample MSE: 11.668798446655273
```

```
In-Sample RMSE: 3.4159622192382812
In-Sample R²: 0.7513
Validation MSE: 11.992977142333984
Validation RMSE: 3.463087797164917
Validation R²: 0.7606
Cross-Validation MSE: 6.4855217933654785
Cross-Validation RMSE: 2.5006473064422607
Cross-Validation R²: 0.8630
```



XL Neural Net:
The Best performance combination was found with SELU activation function and Stochastic Gradient Descent Optimizer.

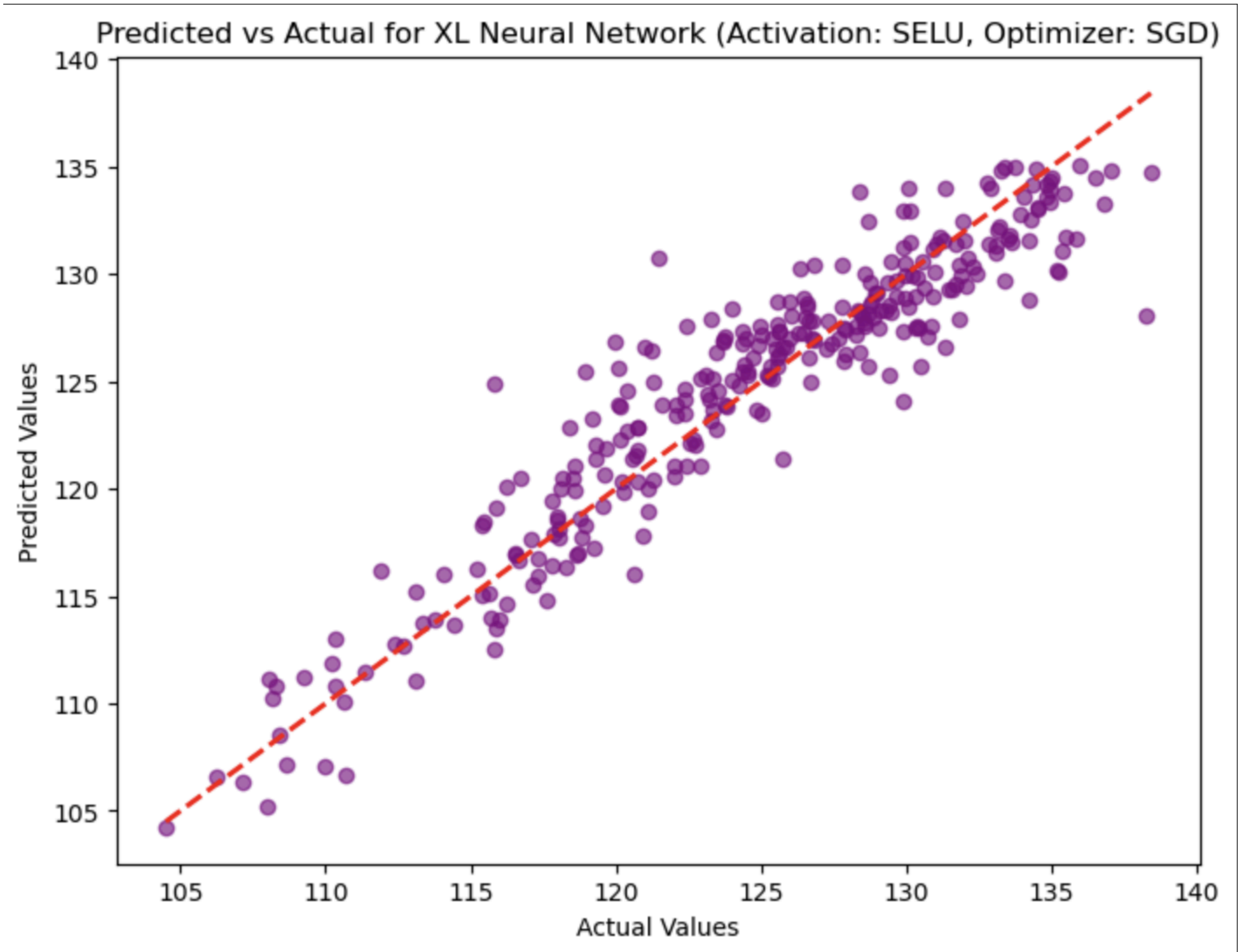
Learning Rate: 0.0001
Batch Size:128
Input Layer Size:5.
Hidden Layer Size:256.

```
training Xl net
Epoch [100/100], Loss: 5.959517
```

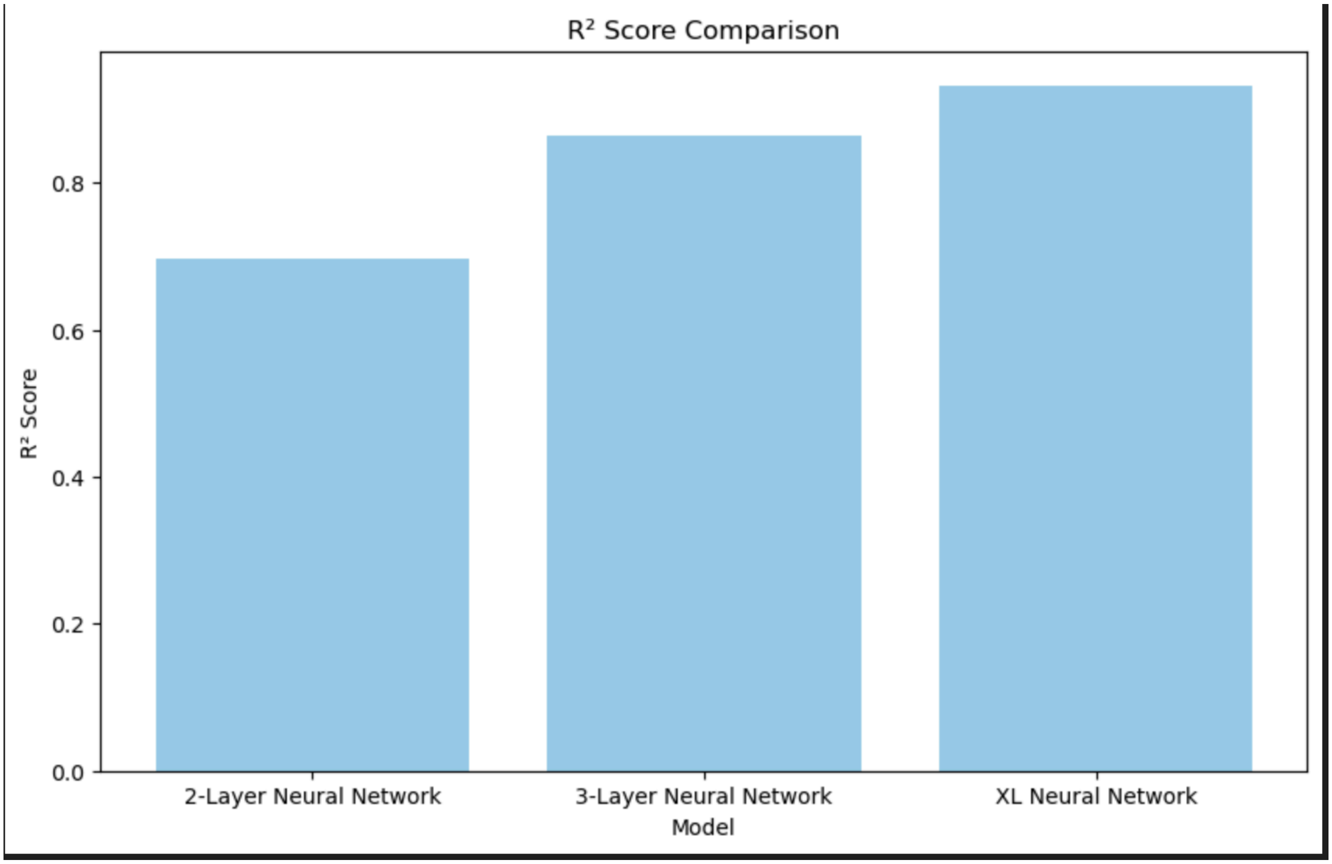
```
5-fold cross-validation evaluation for XL net
Epoch [100/100], Loss: 4.222623
Epoch [100/100], Loss: 3.368778
Epoch [100/100], Loss: 2.898010
Epoch [100/100], Loss: 2.651334
Epoch [100/100], Loss: 2.429800
```

Metrics:
The Best metric is XLayer Neural Network.

```
XL Neural Network:
  Best R²: 0.9323
  Best Metric: Cross-Validation R²
  Activation and Optimizer: Activation: SELU, Optimizer: SGD
  Metrics:
    In-Sample MSE: 6.022091388702393
    In-Sample RMSE: 2.4539949893951416
    In-Sample R²: 0.8716
    Validation MSE: 5.865655422210693
    Validation RMSE: 2.4219114780426025
    Validation R²: 0.8829
    Cross-Validation MSE: 3.1851859092712402
    Cross-Validation RMSE: 1.767377495765686
    Cross-Validation R²: 0.9323
```



The R2 score comparison metrics for all nets.



A **Random Forest Regressor** is initialized with 100 trees (`n_estimators=100`). This means the model will create 100 decision trees, each contributing to the final prediction.

The `random_state=42` ensures reproducibility, meaning each run of this code will yield the same results.

```
Random Forest Test MSE : 3.2960

Random Forest Test RMSE: 1.8155

Random Forest Test R²   : 0.9342
```

GridSearchCV is used to find the optimal hyperparameters, which can improve the model's performance.

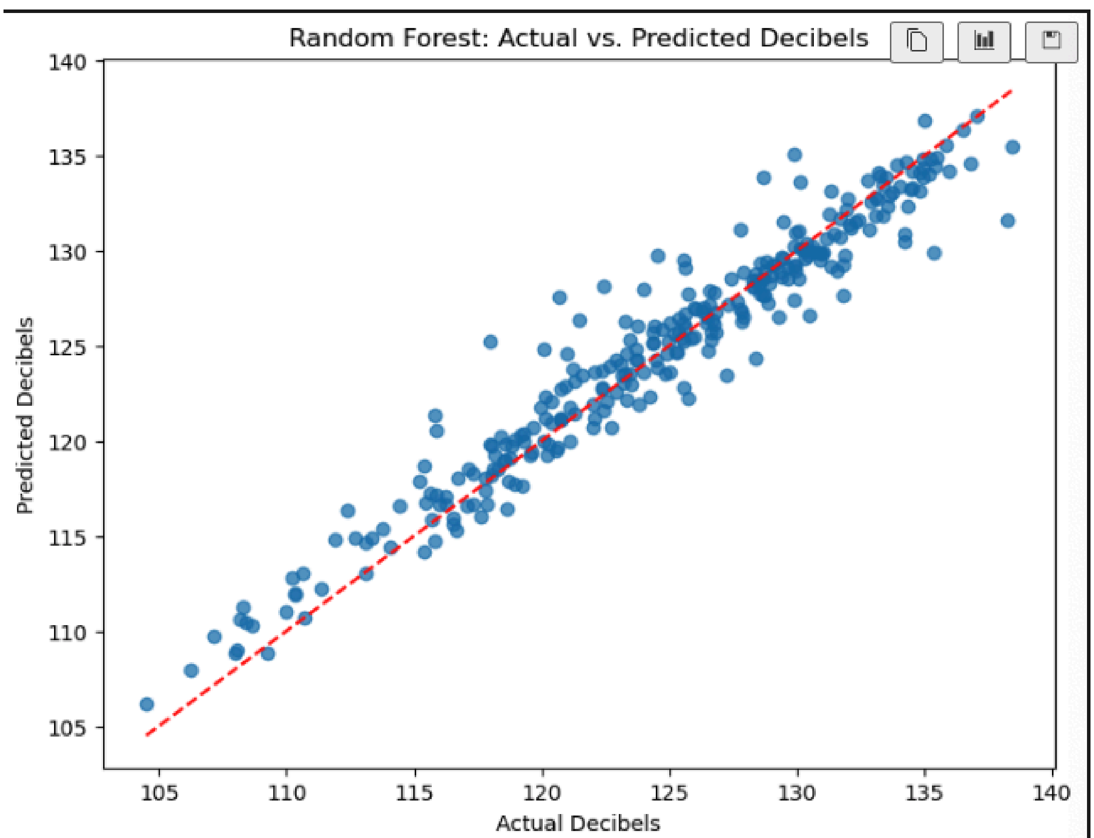
Below is the parameter grid for searching and best combination of parameters.

```
param_grid = {
```

```
    'n_estimators': [100, 200, 300],

    'max_depth': [None, 10, 20, 30],

    'min_samples_split': [2, 5, 10],

    'min_samples_leaf': [1, 2, 4]

}
```

```
Best Parameters: {'max_depth': None, 'min_samples_leaf': 1,
'min_samples_split': 2, 'n_estimators': 200}

Best CV MSE: 0.9225

Best Random Forest Test MSE: 3.2967

Best Random Forest Test R2: 0.9342
```

The actual vs predicted Decibels using Random forest.



Cross Validation R2 using Random Regressor

```
Cross-Validated R² Scores: [0.93686281 0.92761618 0.91976964 0.92730733
0.90115417]

Average Cross-Validated R² Score: 0.9225
```

AutoMPG

Data preprocessing:

EDA:

All data preprocessing and EDA performed for this dataset was the same as described in project 1. Please see our previous report for details.

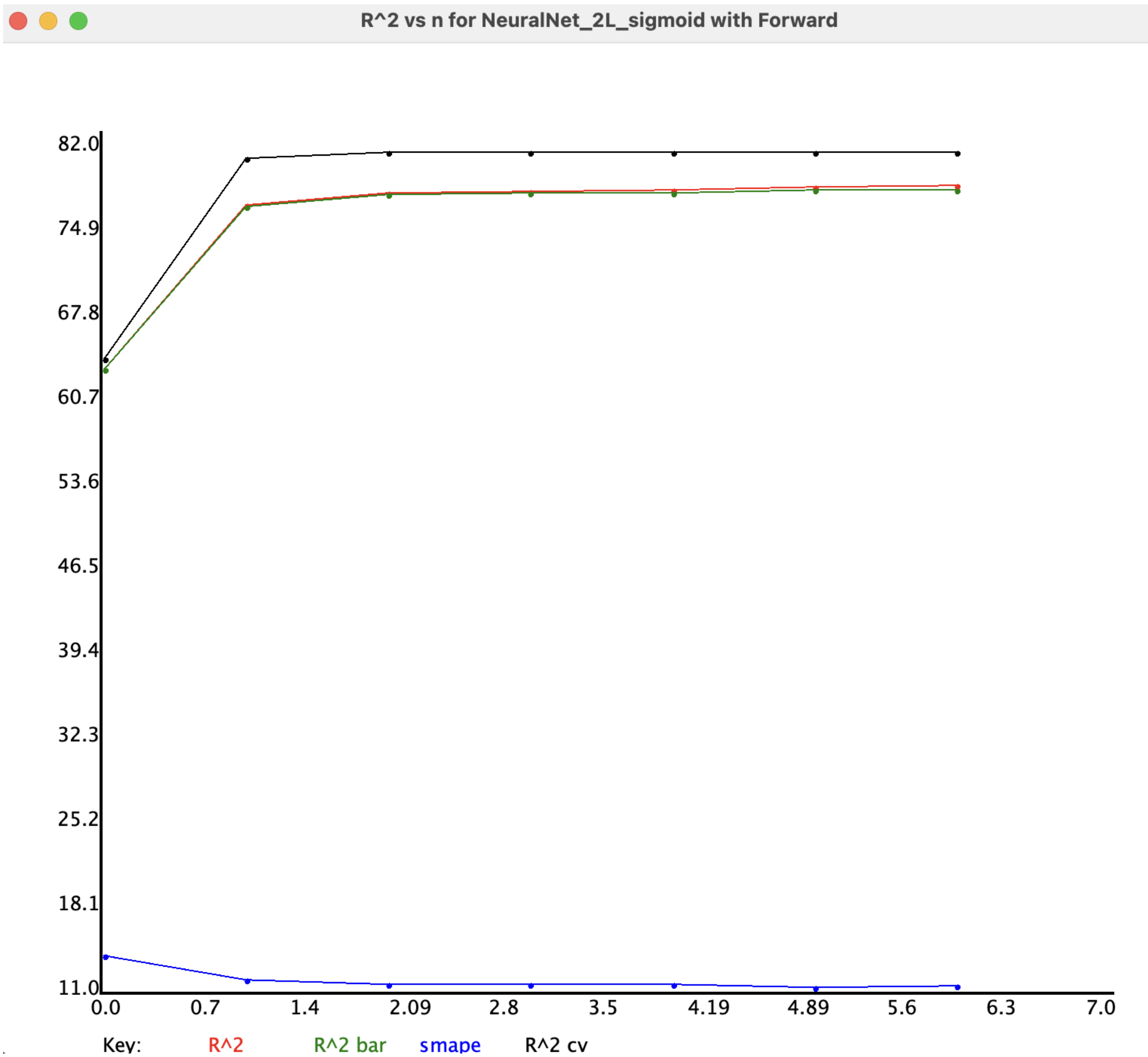
Feature Selection (Forward, Backward, Stepwise)

Neural Net 2L

When using Python, we did not do feature selection for AutoMPG as in the first project, our best performing configurations came from using all available input features. Additionally, when using Scalation, we found that the best model configurations include all input features. This saved us time greatly, as our method for finding the best model configuration when using Python was by using a random grid search.

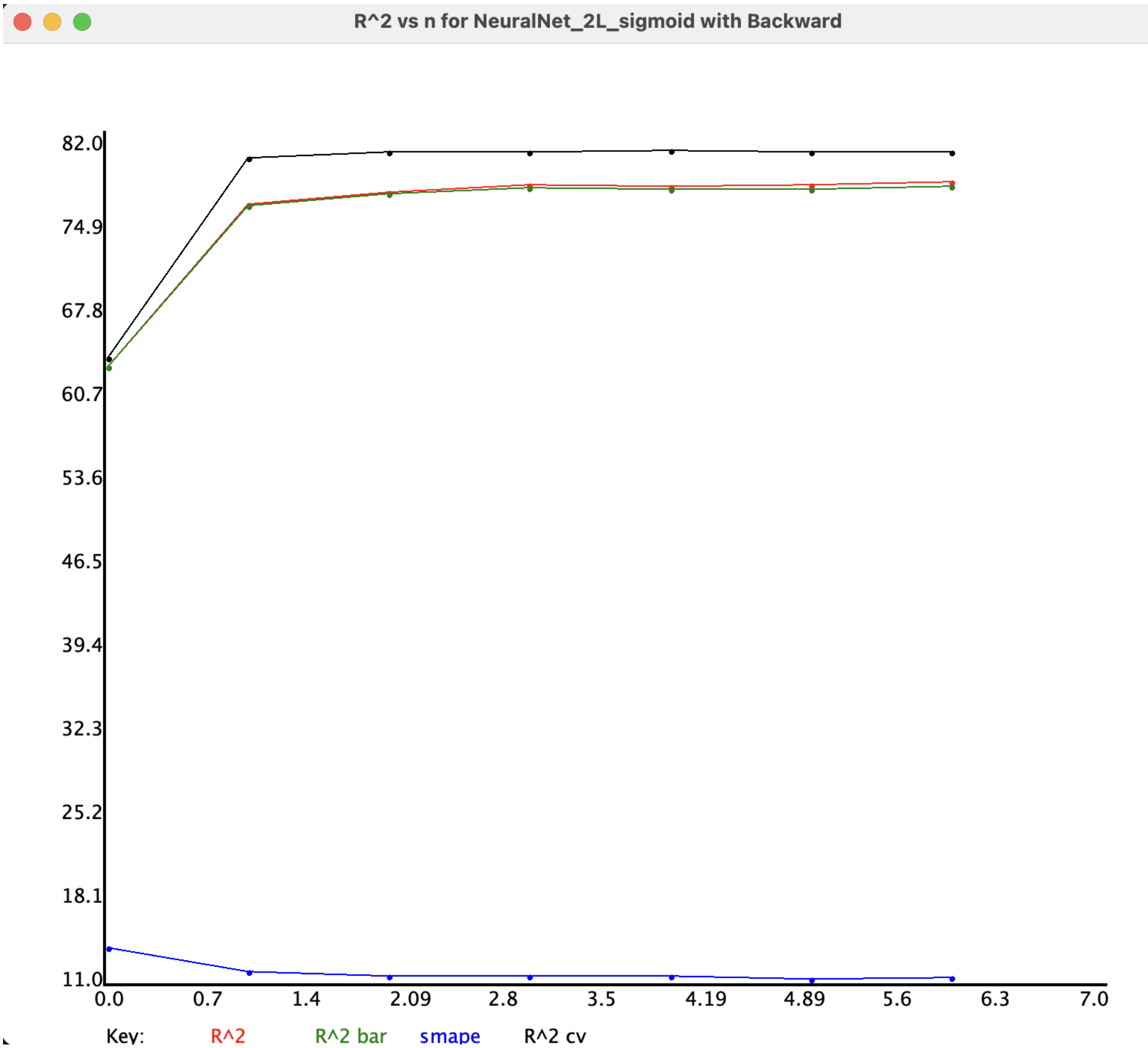
For Neural Net 2L forward selection in scala, we can see from below graph the R squared increased after second feature and doesn't vary much after that. Features were added in the following order.

Best Features: Intercept, weight, model year, horsepower, origin, displacement, cylinders, acceleration



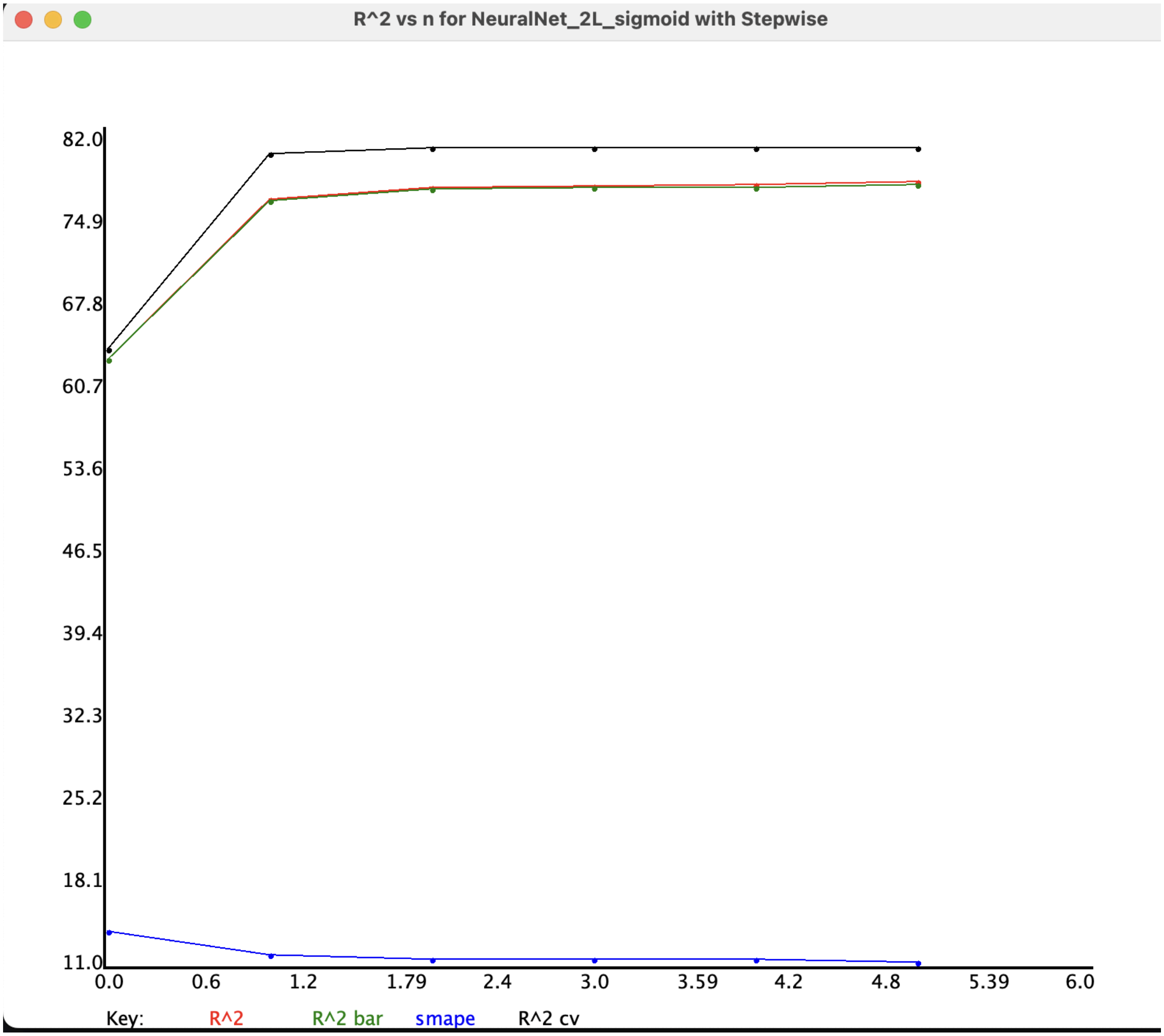
For Neural Net 2L Backward selection in scala also, we can see from below graph the R squared increased after second feature and doesn't vary much after that. Features were removed in the following order.

cylinders, acceleration, displacement, origin, horsepower, model year



For Neural Net 2L Stepwise selection in scala also, we can see from below graph the R squared increased after second feature and doesn't vary much after that. Features were removed in the following order.

Best Features: intercept, weight, model year, horsepower, origin, displacement

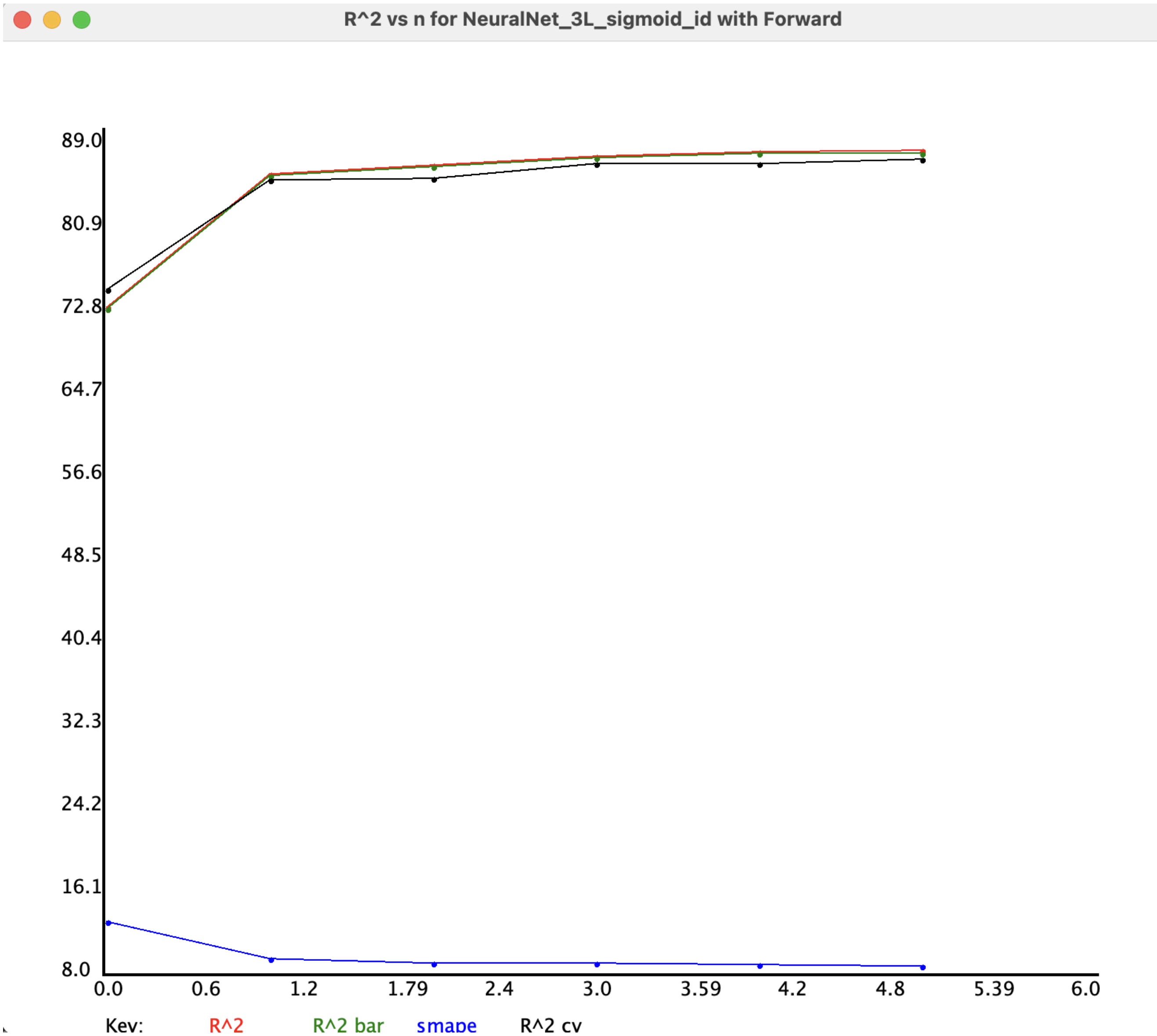


Neural Net 3L

When using Python, we did not do feature selection for AutoMPG as in the first project, our best performing configurations came from using all available input features. Additionally, when using Scalation, we found that the best model configurations include all input features.

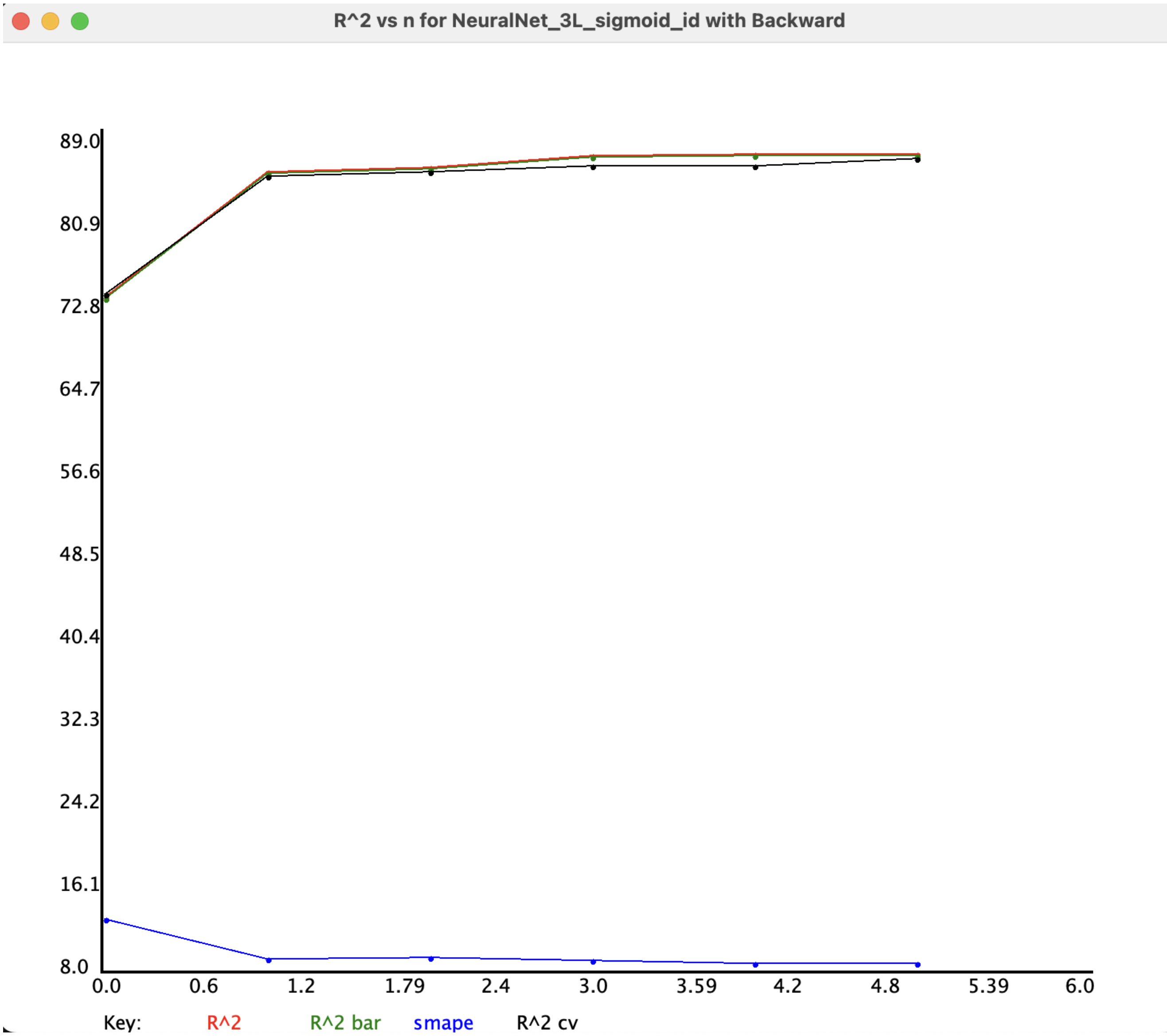
For Neural Net 3L forward selection in scala, we can see from below graph the R squared increased after second feature and slightly increases after that. Features were added in the following order.

Best Features: cylinders, model year, weight, horsepower, origin, acceleration, displacement



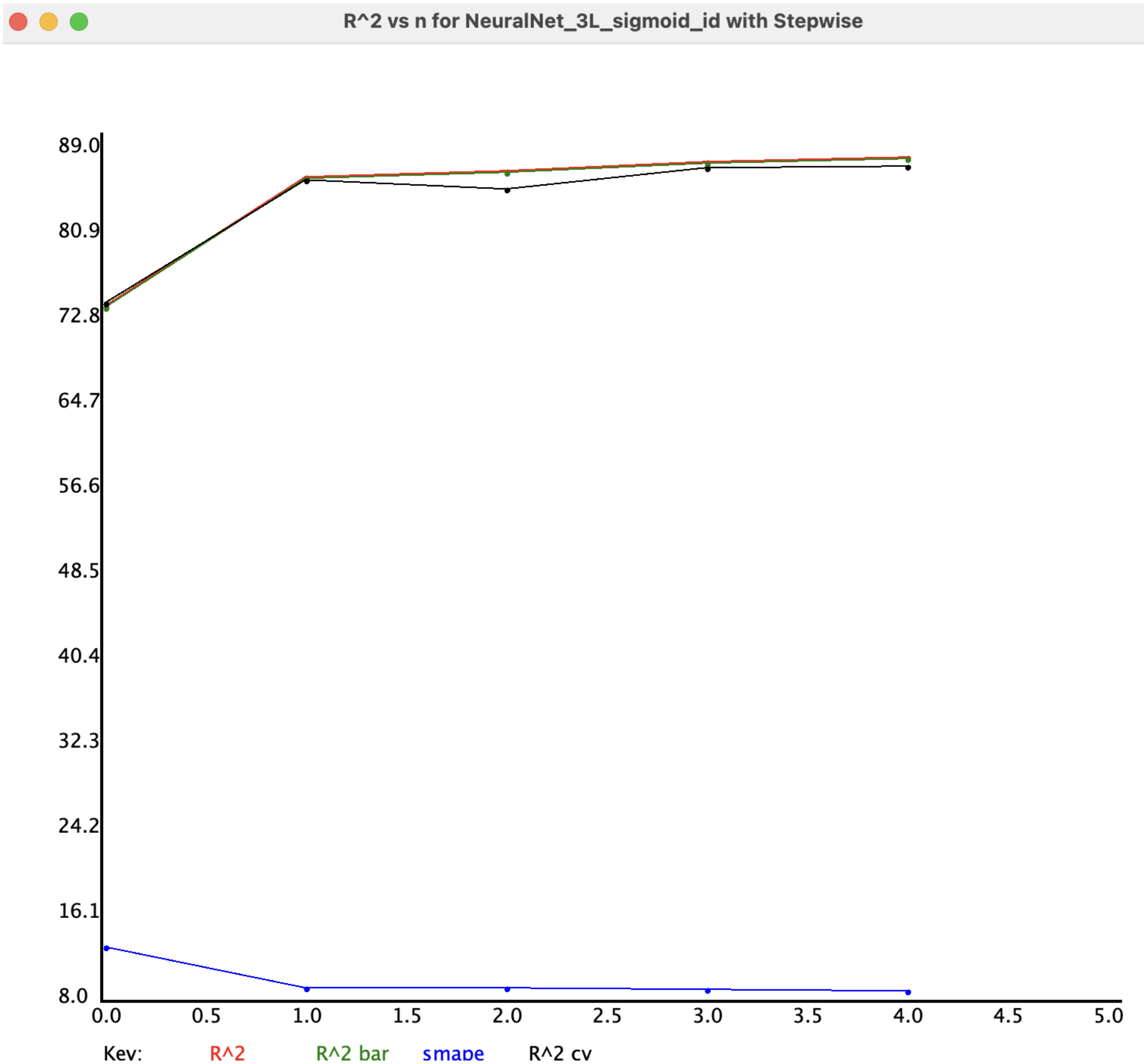
For Neural Net 3L Backward selection in scala also, we can see from below graph the R squared increased after the second feature and doesn't vary much after that. Features were removed in the following order.

displacement, horsepower, acceleration, origin, weight



For Neural Net 3L Stepwise selection in scala also, we can see from below graph the R squared increased after second feature and slightly increases after that. Features were removed in the following order.

Best Features: cylinders, model year, weight, horsepower, origin, acceleration

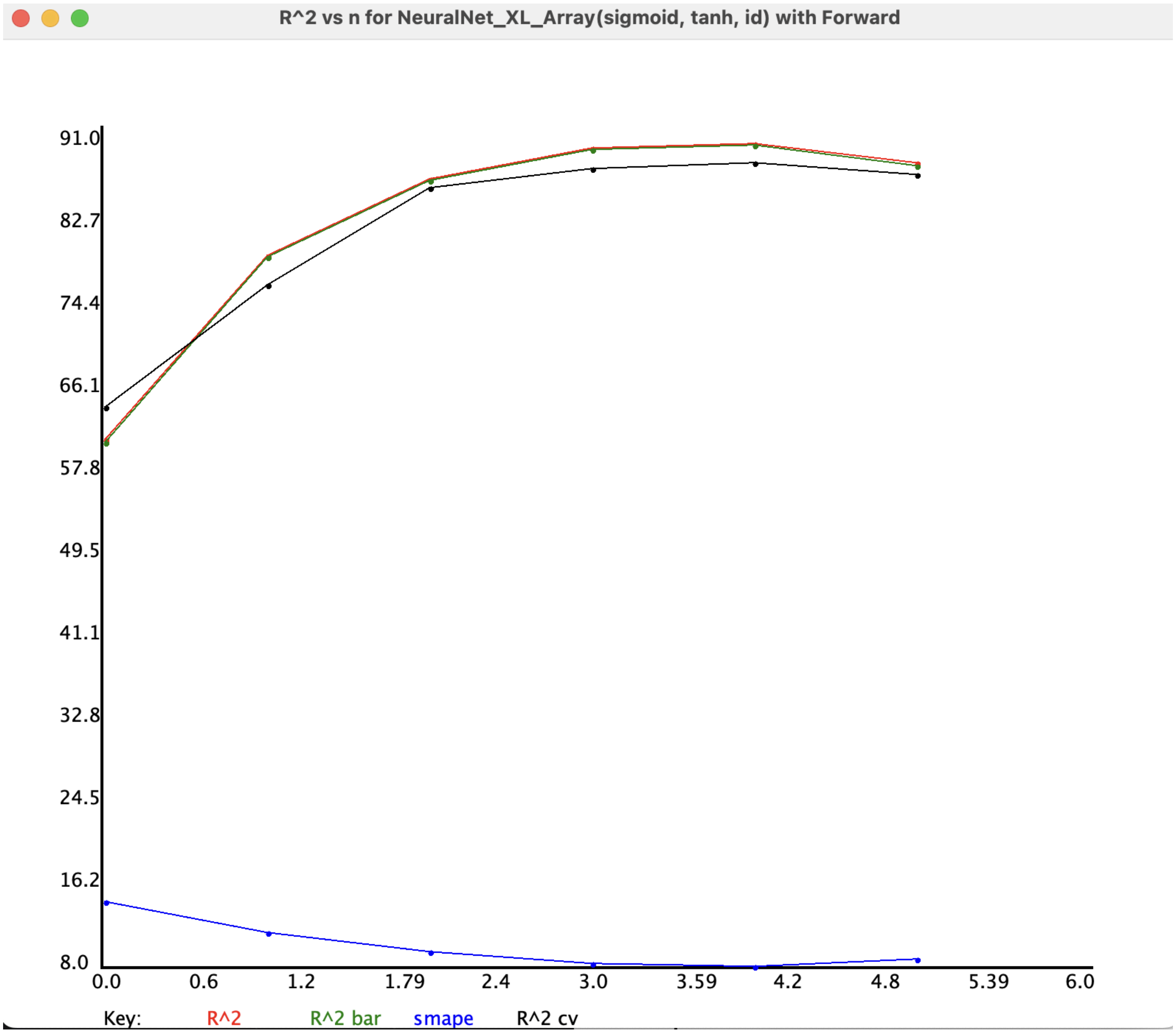


Neural Net XL

When using Python, we did not do feature selection for AutoMPG as in the first project, our best performing configurations came from using all available input features. Additionally, when using Scalation, we found that the best model configurations include all input features.

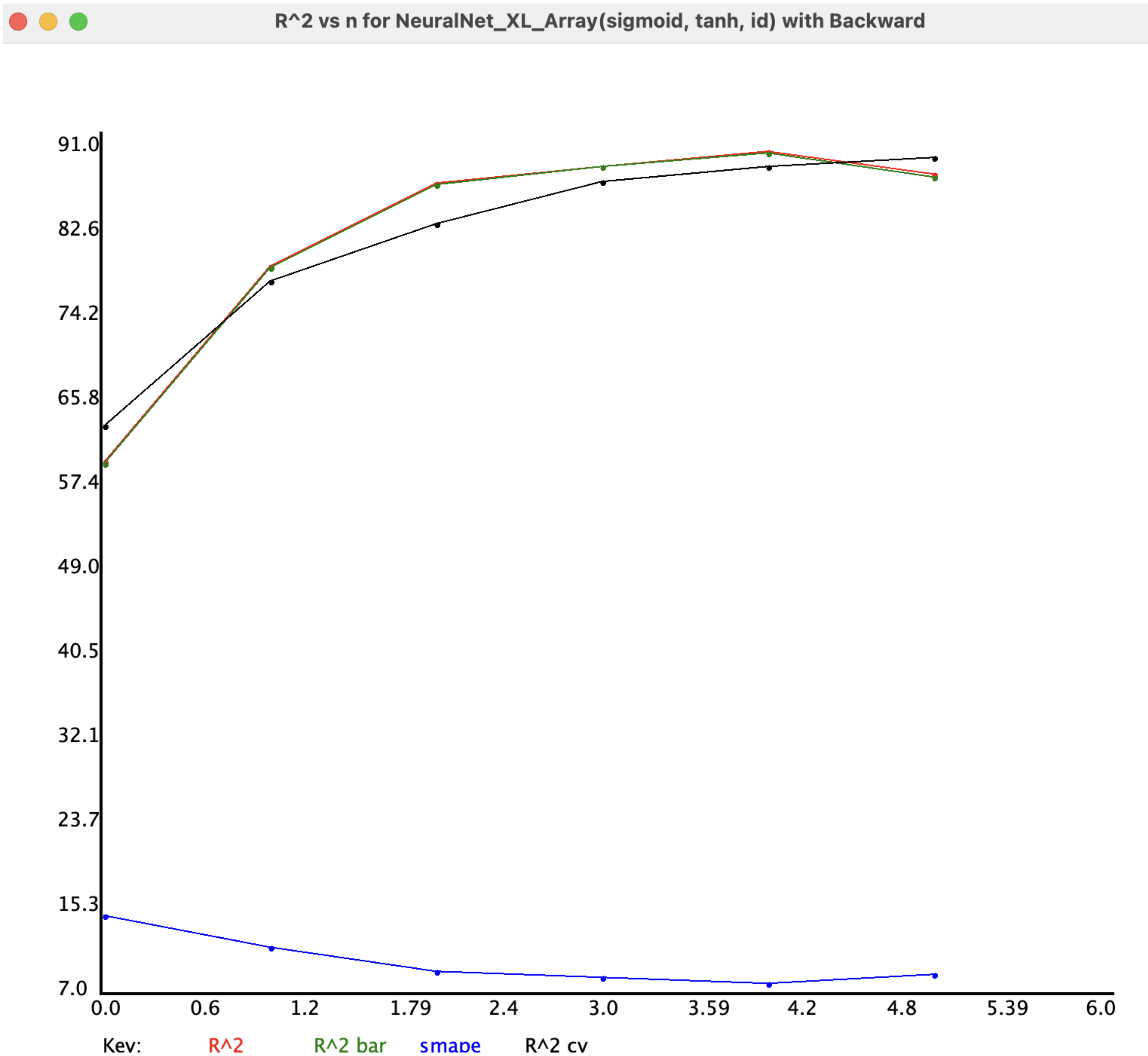
For Neural Net XL forward selection in scala, we can see from below graph the R squared increased after second feature and slightly increases after that. Features were added in the following order.

Best Features: cylinders, model year, weight, origin, acceleration, horsepower, displacement



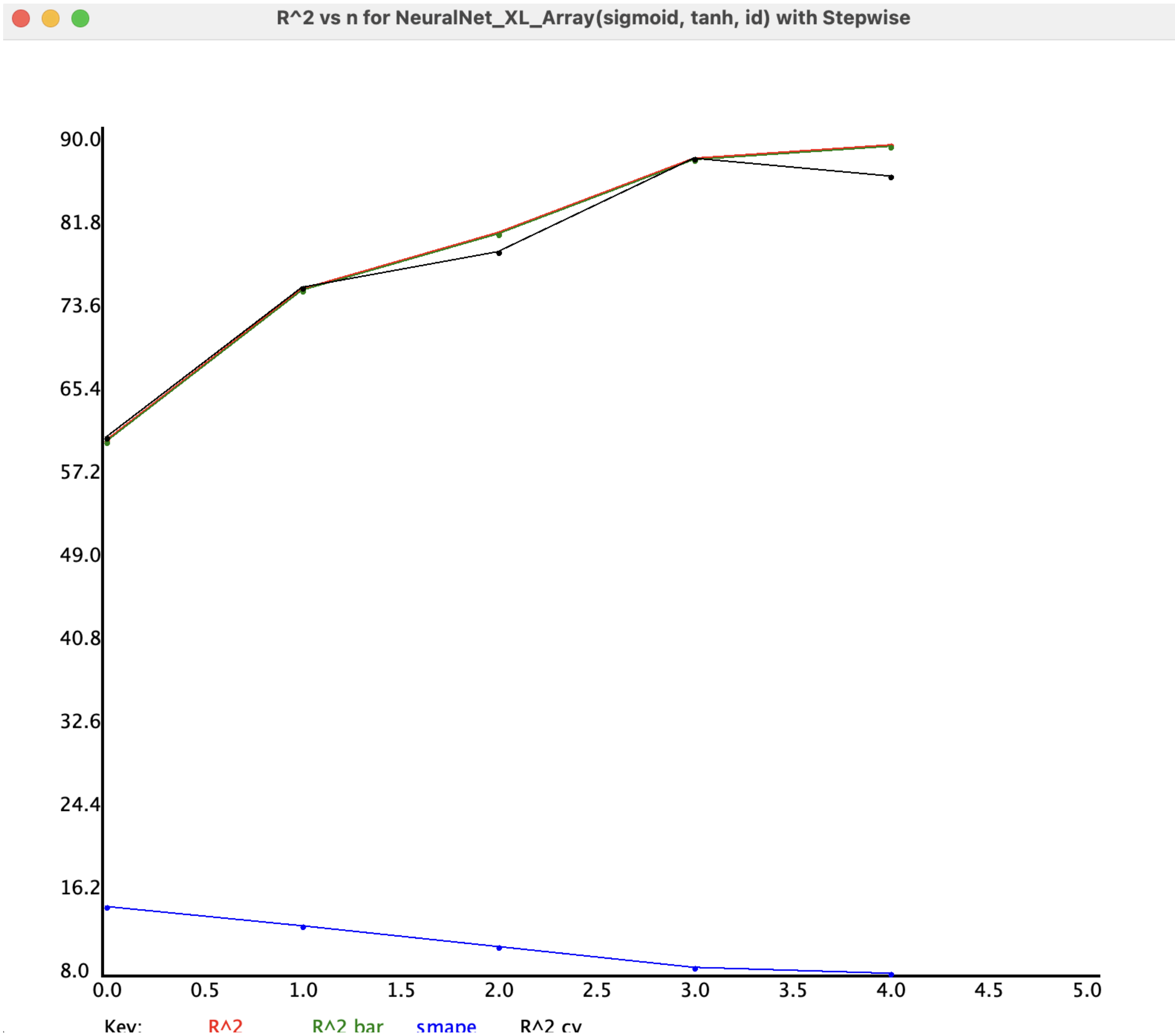
For Neural Net XL Backward selection in scala also, we can see from below graph the R squared increased after the second feature and doesn't vary much after that. Features were removed in the following order.

horsepower, displacement, origin, acceleration, weight



For Neural Net XL Stepwise selection in scala also, we can see from below graph the R squared increased after the second feature and slightly increased after that. Features were removed in the following order.

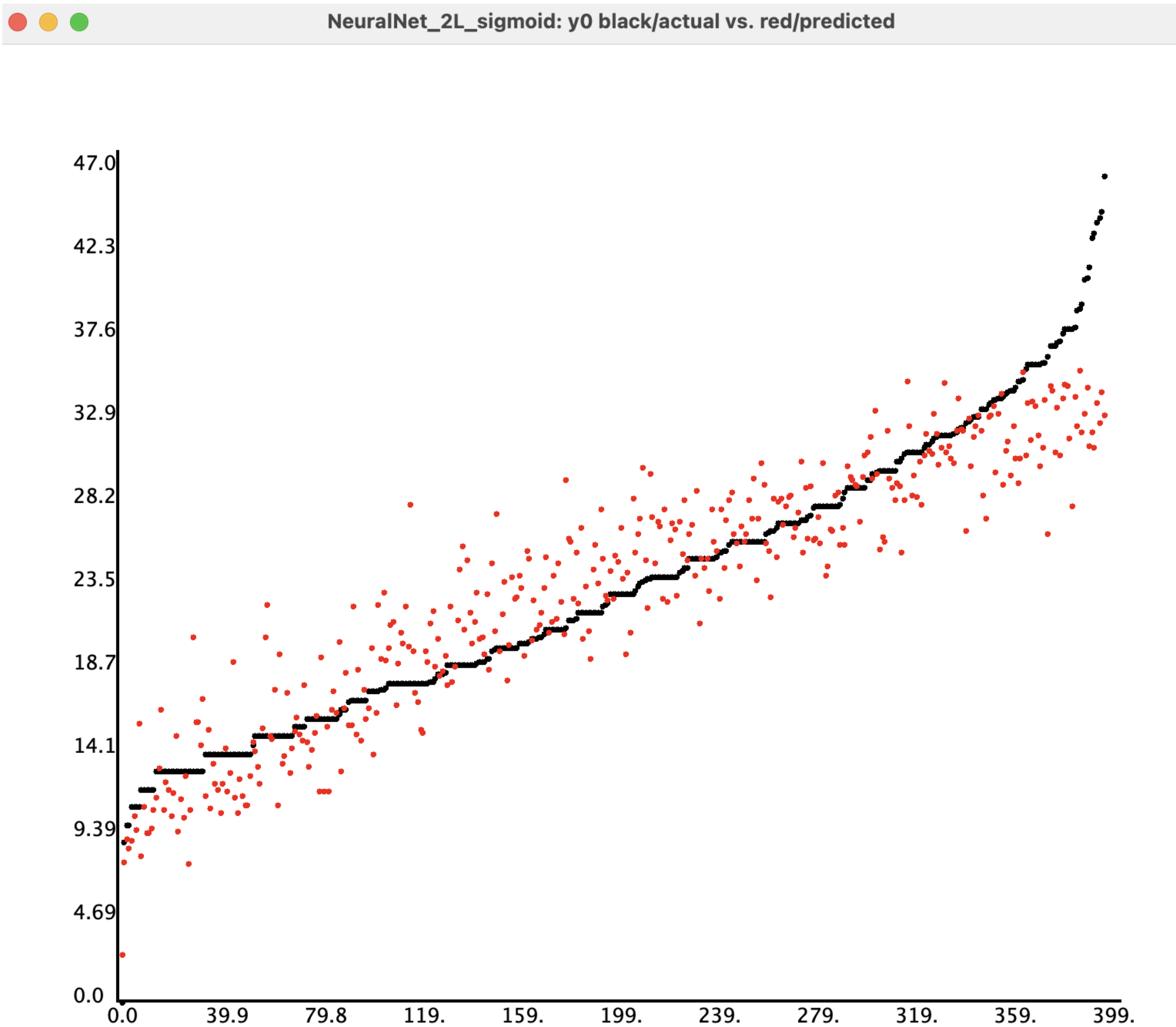
Best Features: cylinders, model year, weight, acceleration, origin, displacement



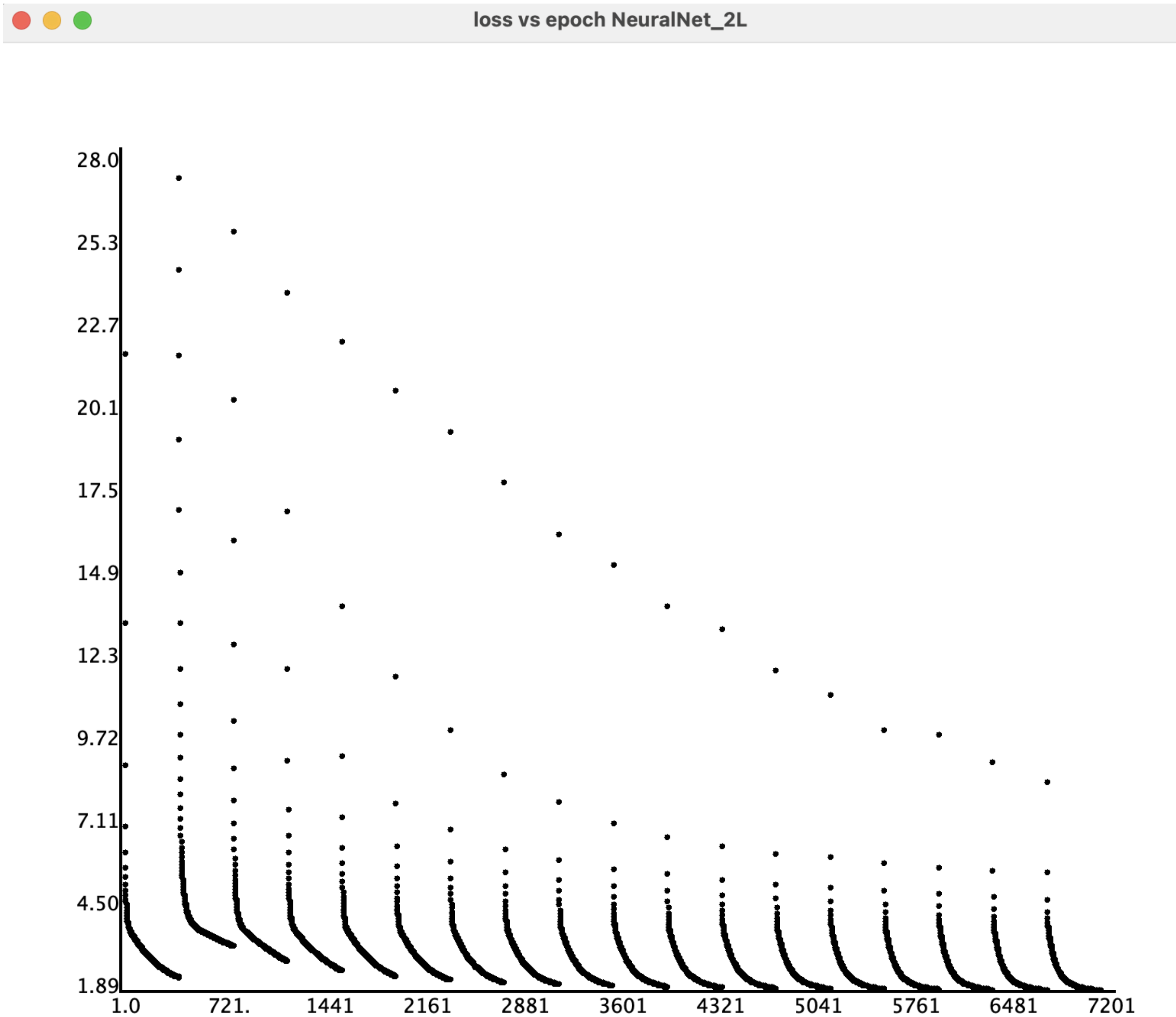
Splitting Data Information:

Discussion of Results:

In scala, the Neural net 2L has provided an R squared of 0.85 with an activation function sigmoid and learning rate of 0.1. Below is the y actual vs y predicted graph from Scala.



Below graph represents the loss vs epochs for the sigmoid activation on Neural net 2L.



REPORT

----

modelName mn = NeuralNet_2L_sigmoid

----

hparameter hp = HyperParameter (HashMap(lambda -> (0.01,0.01), maxEpochs -> (400,400), eta -> (0.1,0.1), nu -> (0.9,0.9), upLimit -> (4,4), beta -> (0.9,0.9), bSize -> (20,20)))

----

features fn = Array(intercept, cylinders, displacement, horsepower, weight, acceleration, model year, origin)

----

parameter bb = Array(b.w =
MatrixD (-0.928601,
-0.00177435,
0.160674,

-0.165400,
-0.820607,
0.0787008,
0.295707,
0.0600795)
b.b = null)

----

fitMap qof =

rSq -> VectorD(0.861906)
rSqBar -> VectorD(0.859428)
sst -> VectorD(24252.6)
sse -> VectorD(3349.12)
mse0 -> VectorD(8.41488)
rmse -> VectorD(2.90084)
mae -> VectorD(2.16566)
dfm -> VectorD(7.00000)
df -> VectorD(390.000)
fStat -> VectorD(347.739)
aic -> VectorD(-972.608)
bic -> VectorD(-940.717)
mape -> VectorD(9.53731)
smape -> VectorD(9.44915)

Below is the Quality of fit table for the Neural Net 2L. This indicates the maximum R squared of 0.88.

| showQofStatTable: Statistical Table for QoF |

| name | num | min | max | mean | stdev | interval |

----

rSq	5	0.799	0.884	0.846	0.041	0.051
rSqBar	5	0.796	0.881	0.843	0.041	0.051		sst	5	4288.643	5682.595	4683.529	587.830	730.030		sse	5	501.613	882.530	714.605	174.409	216.599		mse0	5	6.350	11.171	9.046	2.208	2.742		rmse	5	2.520	3.342	2.989	0.378	0.469
mae	5	1.971	2.500	2.232	0.235	0.292
dfm	5	7.000	7.000	7.000	0.000	0.000
df	5	390.000	390.000	390.000	0.000	0.000		fStat	5	221.920	422.671	327.097	97.341	120.888		aic	5	-193.189	-170.522	-183.196	10.378	12.889		bic	5	-174.233	-151.567	-164.241	10.378	12.889		mape	5	8.923	10.858	9.740	0.873	1.084		smape	5	8.856	10.527	9.701	0.774	0.961

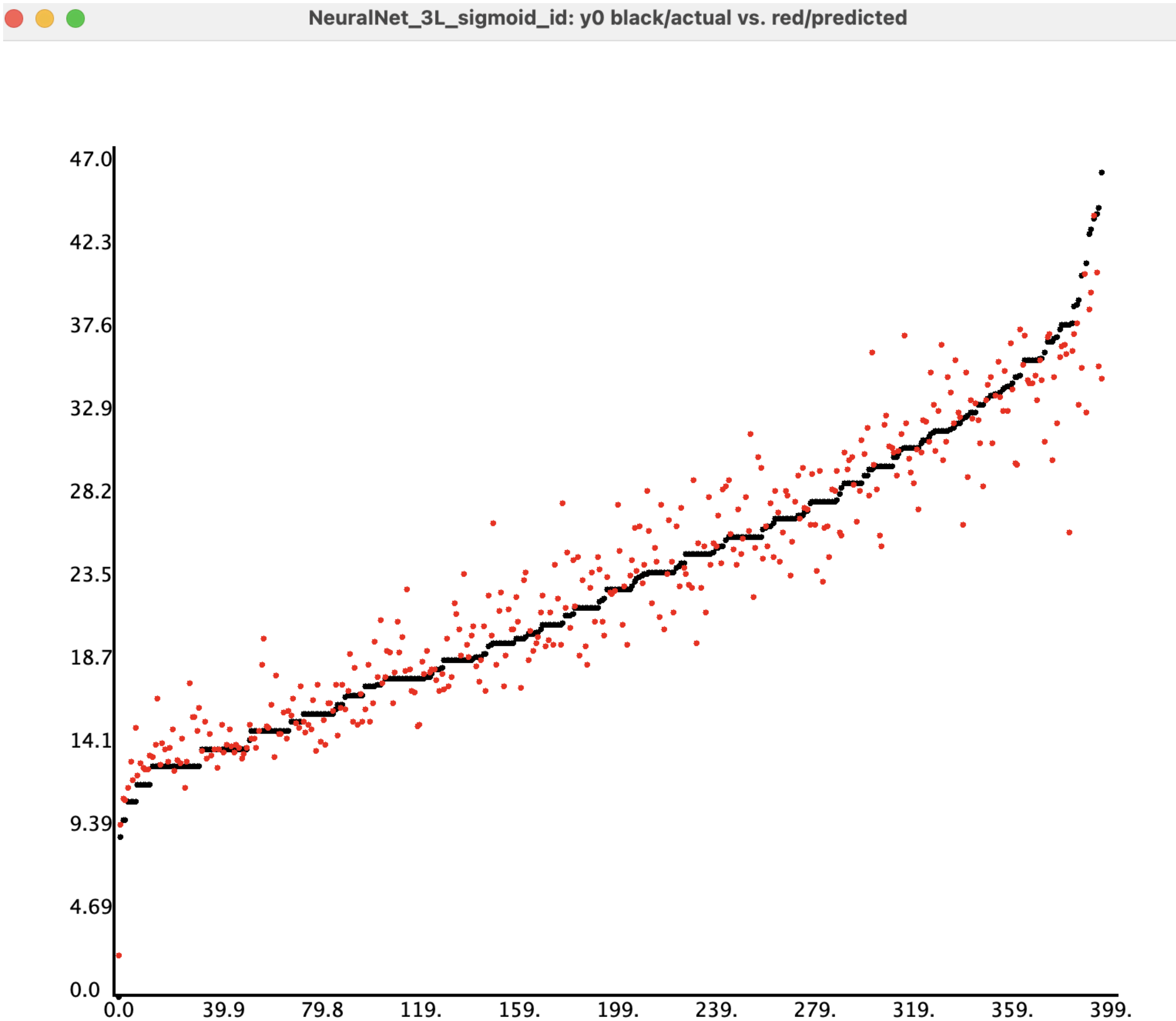
The activation function selection done in scala has given the best results with sigmoid function. Below are the results for this run.

REPORT

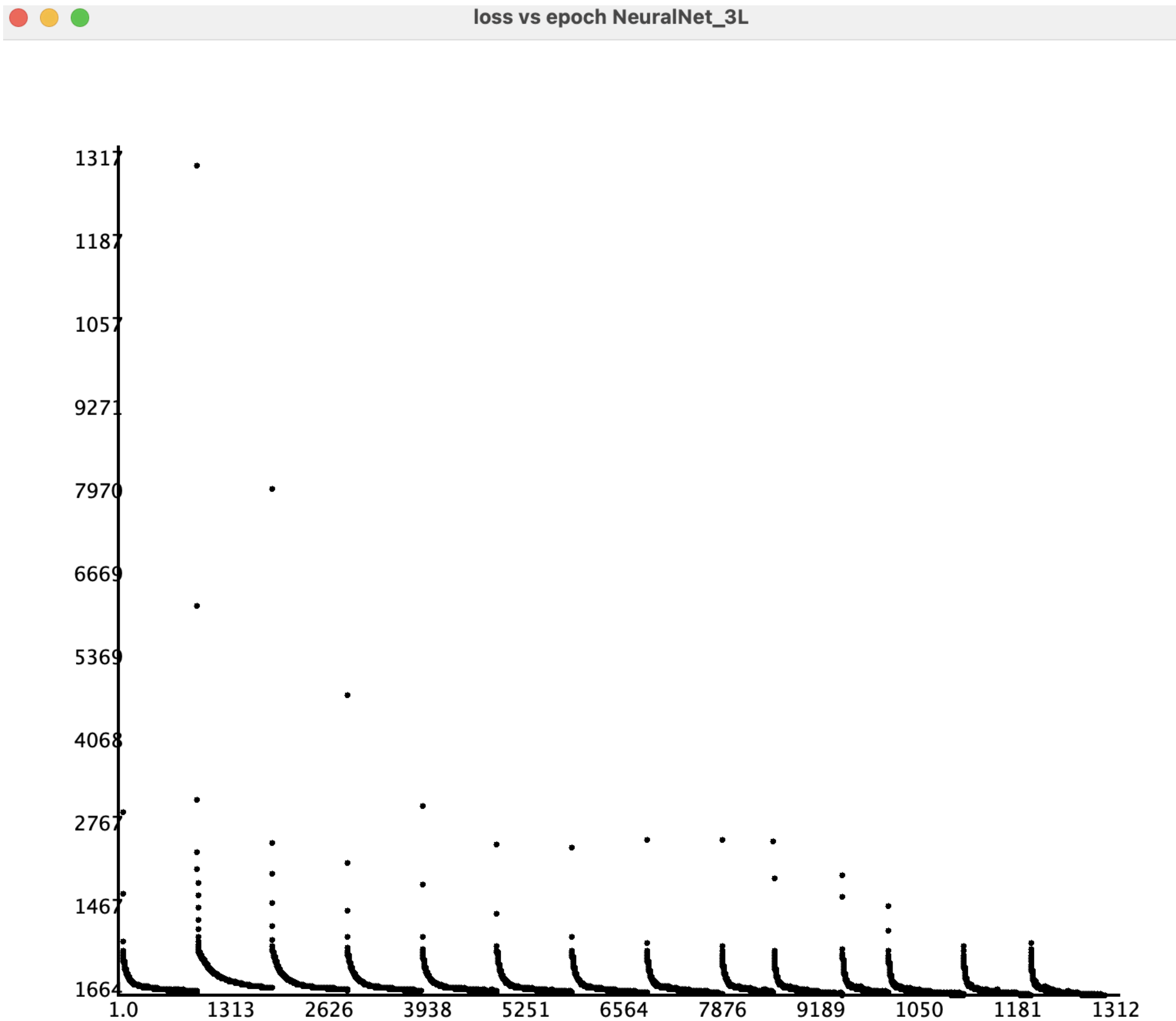
```
    ----------------------------------------------------------------------------
    modelName mn = NeuralNet_2L_sigmoid
    ----------------------------------------------------------------------------
    hparameter hp = HyperParameter (HashMap(lambda -> (0.01,0.01), maxEpochs ->
(400,400), eta -> (0.1,0.1), nu -> (0.9,0.9), upLimit -> (4,4), beta -> (0.9,0.9), bSize -> (20,20)))
----------------------------------------------------------------------------
    features fn = Array(intercept, cylinders, displacement, horsepower, weight, acceleration,
model year, origin)
    ----------------------------------------------------------------------------
    parameter bb = Array(b.w =
MatrixD (-1.05371,
          0.0372628,
          0.104693,
         -0.288520,
         -0.758932,
          0.0230306,
          0.291694,
          0.0586901)
 b.b = null)
    ----------------------------------------------------------------------------
    fitMap qof =
                  rSq -> VectorD(0.863092)
                  rSqBar -> VectorD(0.860635)
                  sst -> VectorD(24252.6)
                  sse -> VectorD(3320.37)
                  mse0 -> VectorD(8.34263)
                  rmse -> VectorD(2.88836)
                  mae -> VectorD(2.15410)
                  dfm -> VectorD(7.00000)
                  df -> VectorD(390.000)
                  fStat -> VectorD(351.233)
                  aic -> VectorD(-970.893)
                  bic -> VectorD(-939.001)
                  mape -> VectorD(9.49008)
                  smape -> VectorD(9.44301)

    ----------------------------------------------------------------------------
```

Neural Net 3L

In scala, the Neural net 3L has provided an R squared of 0.93 with an activation function sigmoid and learning rate of 0.1. Below is the y actual vs y predicted graph from Scala.



Below is the loss vs epochs graph for this run.



REPORT

----

modelName mn = NeuralNet_3L_sigmoid_id

----

hparameter hp = HyperParameter (HashMap(lambda -> (0.01,0.01), maxEpochs -> (1000,400), eta -> (0.01,0.1), nu -> (0.9,0.9), upLimit -> (4,4), beta -> (0.9,0.9), bSize -> (20,20)))

----

features fn = Array(cylinders, displacement, horsepower, weight, acceleration, model year, origin)

----

parameter bb = Array(b.w =
MatrixD (1.86321, 1.04372, -1.06053, 1.01585, -1.12428, -0.771175, 1.24719, -0.504893, 0.776952, 0.409431, -1.34338, 0.0319128, -0.310463, 0.829072, 0.902026,

2.83590, -0.316080, -0.290286, 0.290362, 0.129741, 1.76084, 2.23738, 0.763239, 1.54982, -1.82092, -0.993986, -0.952304, 0.282476, -2.23941, 0.0682018,
-0.681312, 0.382190, 0.118804, -1.42275, -0.247070, -0.274539, -2.04907, -0.220625, -2.57749, -0.931825, -1.98981, -0.221887, -0.441380, -1.30239, -1.51540,
-1.68630, -2.50444, -1.78127, -0.341761, -2.24155, -3.04670, -3.05008, 1.92765, -2.98440, -1.15817, -2.03586, 0.987056, -1.98405, -0.754865, -0.00317017,
2.95935, 0.610676, -0.856629, 0.765101, -0.854784, 1.44259, 0.140657, 2.01778, -1.42050, -1.86556, -0.0582271, -0.120732, -1.20351, -1.49345, 1.52713,
3.98199, -0.938310, 0.0366900, -1.95391, 0.147727, 2.40450, 3.43409, 1.47676, -0.460450, 0.0101855, -2.75515, 4.36123, 1.34455, -0.529500, -2.62039,
2.65680, 0.366207, 0.457576, -1.94836, 0.680804, 0.306730, 3.14585, -3.28973, -1.90143, 2.65097, -1.45560, -1.05770, 0.562446, 1.74775, -2.32025)
b.b = VectorD(-3.86386, -3.52830, -2.43722, 0.958187, -2.67221, -4.68682, -3.27569, 0.444117, -0.492185, -0.185390, -2.56153, -2.22970, -1.55157, -0.0941967, 1.08257), b.w =
MatrixD (5.16147,
3.48298,
2.64462,
2.46878,
3.01807,
4.34762,
5.34847,
3.56074,
3.44188,
2.91350,
3.29997,
4.02907,
2.80294,
2.85156,
3.20587)
b.b = VectorD(0.427617))
----
fitMap qof =
rSq -> VectorD(0.934239)
rSqBar -> VectorD(0.933062)
sst -> VectorD(24252.6)
sse -> VectorD(1594.86)
mse0 -> VectorD(4.00720)
rmse -> VectorD(2.00180)
mae -> VectorD(1.48301)
dfm -> VectorD(7.00000)
df -> VectorD(391.000)
fStat -> VectorD(793.544)
aic -> VectorD(-824.968)

bic -> VectorD(-793.076)
mape -> VectorD(6.46087)
smape -> VectorD(6.41443)

While trying the best activation functions, I have figured the tanh has worked best on the Nueral net 3L and provided a R squared of 0.94. Below are the results for this run.

REPORT

----

modelName mn = NeuralNet_3L_tanh_id

----

hparameter hp = HyperParameter (HashMap(lambda -> (0.01,0.01), maxEpochs -> (400,400), eta -> (0.01,0.1), nu -> (0.9,0.9), upLimit -> (4,4), beta -> (0.9,0.9), bSize -> (20,20)))

----

features fn = Array(cylinders, displacement, horsepower, weight, acceleration, model year, origin)

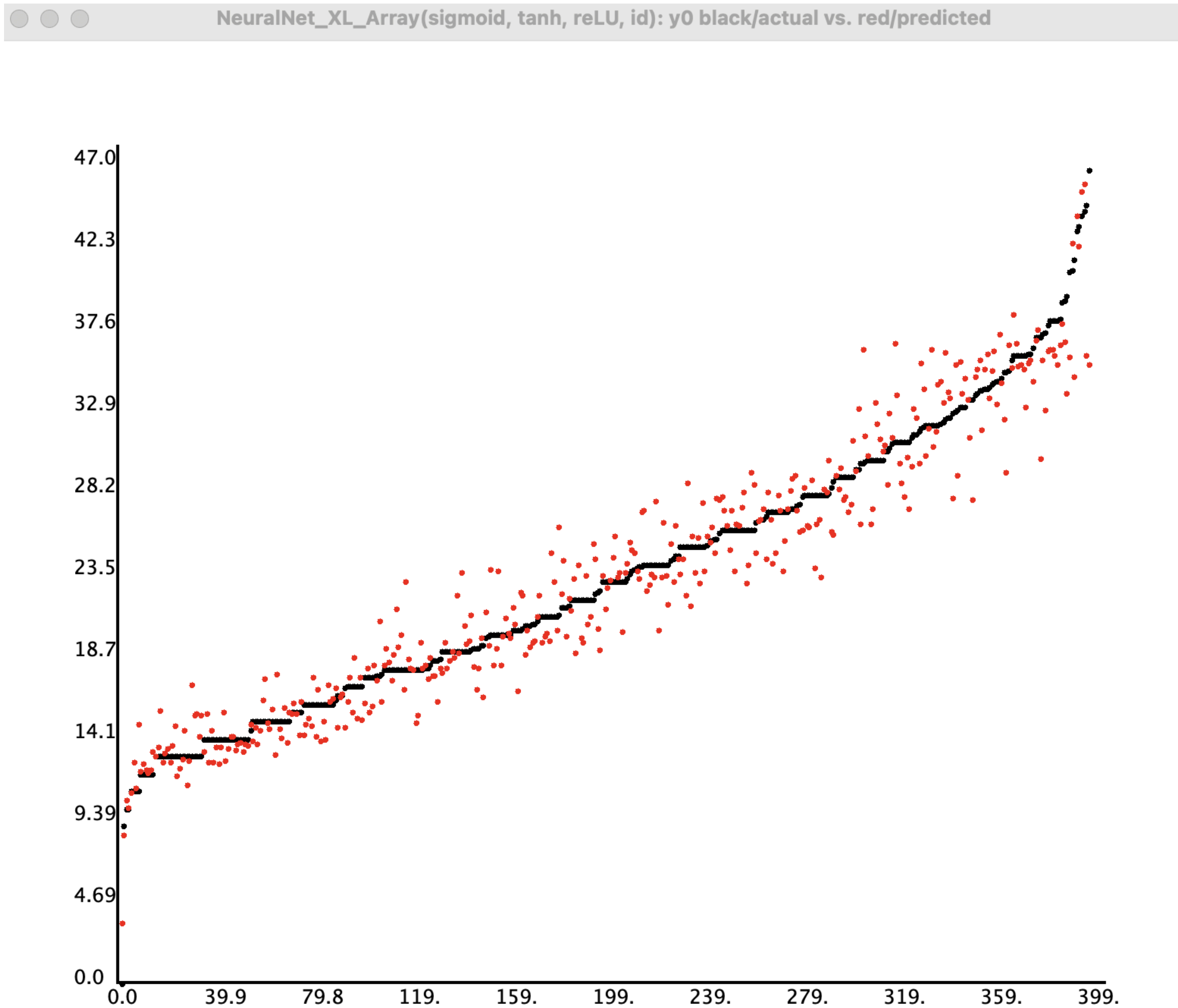
----

parameter bb = Array(b.w =
MatrixD (1.12379, 0.346883, -2.17898, -1.61364, -0.0165722, -0.373094, -0.505182,
-2.49079, -0.614917, 1.69633, 0.431740, -0.364007, 1.28345, -1.05168, -0.0928464,
0.184795, 1.17822, -0.609622, -0.432550, 1.98675, 1.90807, 2.63954, -2.13823,
0.462415, -1.14164, -0.213652, 1.25847, 1.01828, -2.12493, 1.21956,
1.39385, 0.427880, 1.47574, -0.146241, -1.06876, 0.601788, 0.118321, 0.582375,
-2.42572, -1.73644, 1.82587, 0.785387, -0.697205, 1.70771, -1.55954,
1.38219, 0.178048, 1.31951, 1.40874, 1.02822, -0.942454, 0.817721, 2.74497,
1.32609, 1.80583, 1.94591, 0.383129, 0.628785, 1.95933, 2.82338,
2.53010, 1.19676, -0.367348, -0.719374, 1.15945, -2.43225, 0.290985, -2.53699,
0.983692, 0.892733, 1.09385, 2.00865, 0.583003, -0.430222, 0.280728,
-0.367345, 0.141374, -0.0868986, 3.96117, 1.07572, 2.19330, 0.720679,
-2.53318, -2.84328, -3.54975, 0.151685, -0.495712, 0.642296, -2.58630, -0.266640,
2.03063, 1.36688, -0.0952498, -1.55941, 2.09952, 1.60832, -0.496280, -2.00329,
-2.59268, -0.0407647, -1.04554, 1.52022, 1.28049, -2.80735, -0.210231)
b.b = VectorD(-0.886440, -1.68170, -2.06313, -1.52923, -2.05185, -2.95208,
0.321919, 3.31634, 0.747109, 3.44173, 2.46616, -1.92458, -2.27060, 1.98430,
3.05937), b.w =
MatrixD (-2.61953,
-3.64281,
-3.24084,
2.02752,
-2.39243,
-3.04218,
-2.70361,
-4.29510,

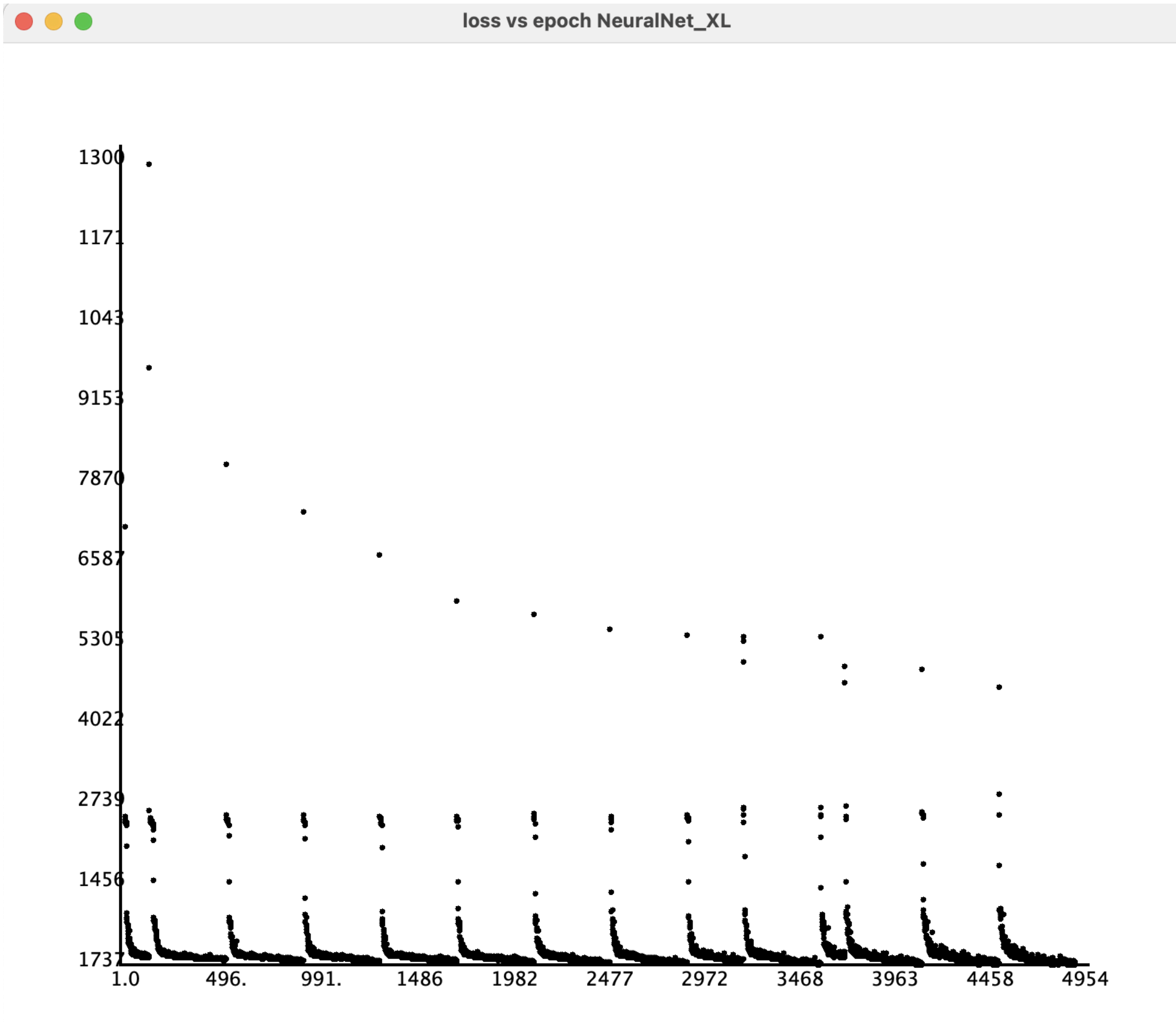
```
        3.72968,
        -2.38708,
        -2.25280,
        -4.18851,
        -3.71806,
        -3.72346,
        -2.24047)
b.b = VectorD(7.00842))
 -----------------------------------------------------------------------------
 fitMap qof =
                 rSq -> VectorD(0.948194)
                 rSqBar -> VectorD(0.947266)
                 sst -> VectorD(24252.6)
                 sse -> VectorD(1256.43)
                 mse0 -> VectorD(3.15687)
                 rmse -> VectorD(1.77676)
                 mae -> VectorD(1.31662)
                 dfm -> VectorD(7.00000)
                 df -> VectorD(391.000)
                 fStat -> VectorD(1022.34)
                 aic -> VectorD(-777.519)
                 bic -> VectorD(-745.627)
                 mape -> VectorD(5.75821)
                 smape -> VectorD(5.66119)

 -----------------------------------------------------------------------------
```

Neural Net XL

In scala, the Neural net XL has provided an R squared of 0.93 with an activation function sigmoid and learning rate of 0.1. Below is the y actual vs y predicted graph from Scala.



Below is the graph for loss vs epochs for this run.



Below results are obtained for Neural net XL with sigmoid, tanh, reLU, id as activation functions and learning rate 0.1.

REPORT

----

modelName mn = NeuralNet_XL_Array(sigmoid, tanh, reLU, id)

----

hparameter hp = HyperParameter (HashMap(lambda -> (0.01,0.01), maxEpochs -> (400,400), eta -> (0.01,0.1), nu -> (0.9,0.9), upLimit -> (4,4), beta -> (0.9,0.9), bSize -> (20,20)))

----

features fn = Array(cylinders, displacement, horsepower, weight, acceleration, model year, origin)

----

fitMap qof =

rSq -> VectorD(0.935822)
rSqBar -> VectorD(0.934673)
sst -> VectorD(24252.6)
sse -> VectorD(1556.49)
mse0 -> VectorD(3.91077)
rmse -> VectorD(1.97757)
mae -> VectorD(1.46390)
dfm -> VectorD(7.00000)
df -> VectorD(391.000)
fStat -> VectorD(814.486)
aic -> VectorD(-820.149)
bic -> VectorD(-788.258)
mape -> VectorD(6.29490)
smape -> VectorD(6.31131)

While searching the activation features, we have found the best results with tanh, tanh, id as the activation functions. This provided a R squared of 0.97 and adjusted R squared of 0.97. This REPORT

----

modelName mn = NeuralNet_XL_Array(tanh, tanh, id)

----

hparameter hp = HyperParameter (HashMap(lambda -> (0.01,0.01), maxEpochs -> (400,400), eta -> (0.01,0.1), nu -> (0.9,0.9), upLimit -> (4,4), beta -> (0.9,0.9), bSize -> (20,20)))

----

features fn = Array(cylinders, displacement, horsepower, weight, acceleration, model year, origin)

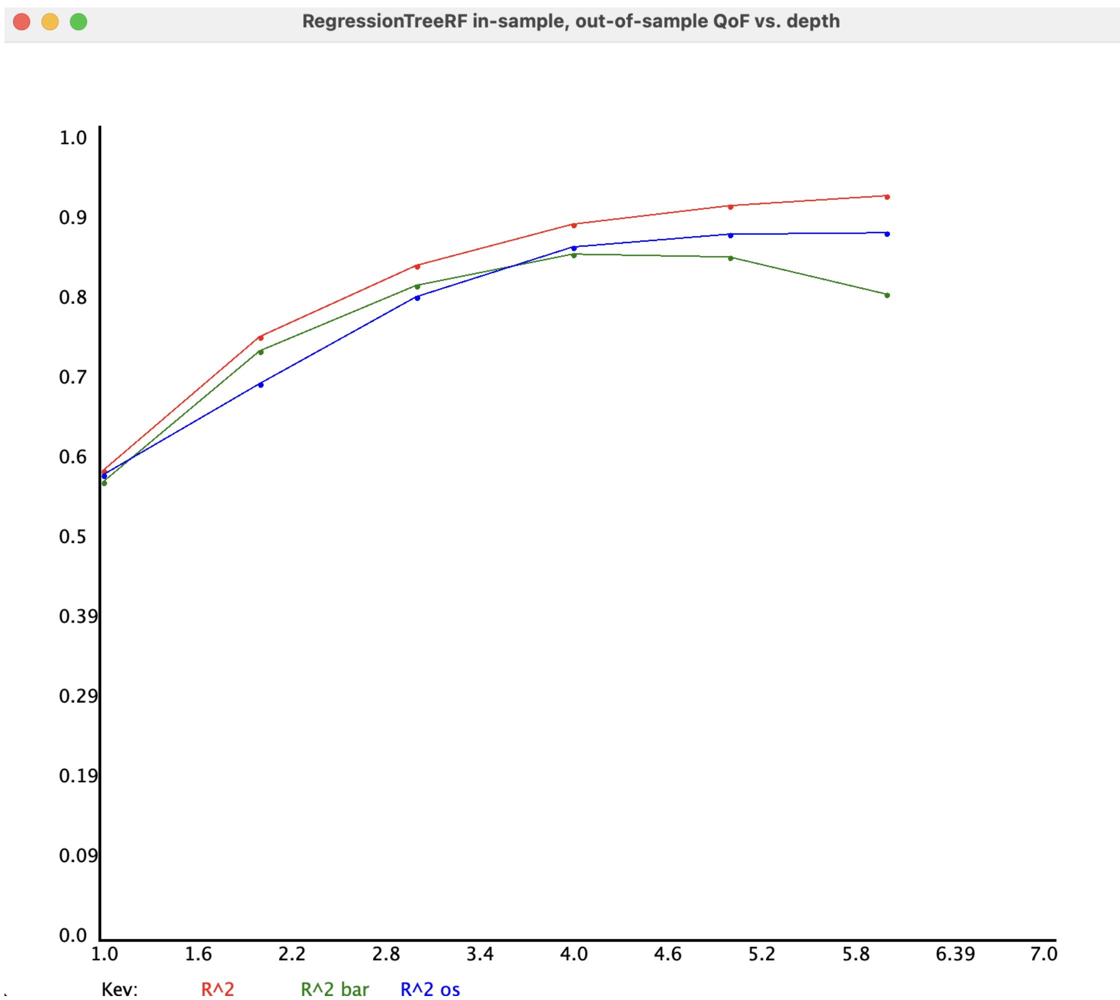
----

fitMap qof =

rSq -> VectorD(0.970755)
rSqBar -> VectorD(0.970231)
sst -> VectorD(24252.6)
sse -> VectorD(709.268)
mse0 -> VectorD(1.78208)
rmse -> VectorD(1.33495)
mae -> VectorD(0.974232)
dfm -> VectorD(7.00000)
df -> VectorD(391.000)
fStat -> VectorD(1854.11)
aic -> VectorD(-663.716)
bic -> VectorD(-631.825)
mape -> VectorD(4.16925)
smape -> VectorD(4.15652)

DEBUG @ **RegressionTreeRF**.train: for tree6 ===
()
LinkedHashMap(**rSq -> 0.887677**, rSqBar -> 1.049025, sst -> 4980.717468, sse -> 559.448162, sde -> 2.677726, mse0 -> 7.081622, rmse -> 2.661132, mae -> 1.962935, dfm -> 260.000000, df -> -181.000000, fStat -> -5.501642, aic -> 325.519226, bic -> 943.945115, mape -> 8.356778, smape -> 8.331557)

Random Forest Regressor



LinkedHashMap(rSq -> 0.887677, rSqBar -> 1.049025, sst -> 4980.717468, sse -> 559.448162, sde -> 2.677726, mse0 -> 7.081622, rmse -> 2.661132, mae -> 1.962935, dfm -> 260.000000, df -> -181.000000, fStat -> -5.501642, aic -> 325.519226, bic -> 943.945115, mape -> 8.356778, smape -> 8.331557)

Python:

2L Neural Net:

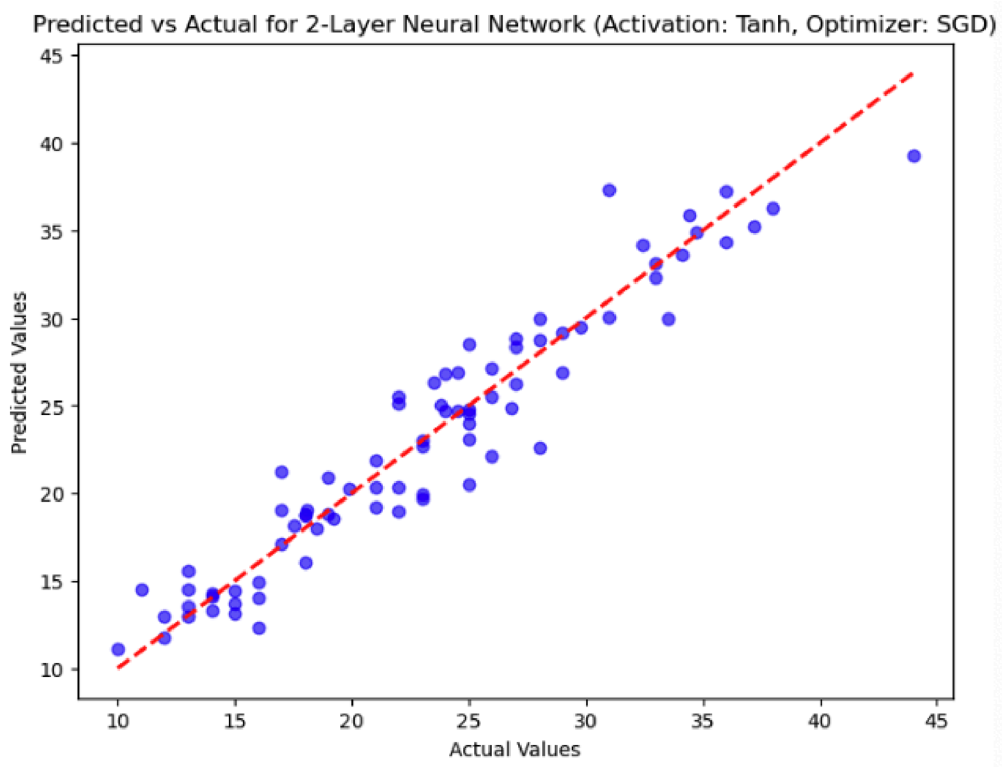
The Best performance combination was found with Tanh activation function and Stochastic Gradient Descent Optimizer.

Learning Rate:0.001
Batch Size:32
Hidden Layer Size:128
Input Size:8

```
training 2L net
Epoch [100/100], Loss: 6.147181
cross validation of 2L net
5-fold cross-validation evaluation
Epoch [100/100], Loss: 5.320034
Epoch [100/100], Loss: 4.774090
Epoch [100/100], Loss: 4.298034
Epoch [100/100], Loss: 3.893799
Epoch [100/100], Loss: 3.546644
```

Metrics:
The best performance was found with Cross-Validation.

```
2-Layer Neural Network:
  Best R²: 0.9277
  Best Metric: Cross-Validation R²
  Activation and Optimizer: Activation: Tanh, Optimizer: SGD
  Metrics:
    In-Sample MSE: 6.1132707595825195
    In-Sample RMSE: 2.4725029468536377
    In-Sample R²: 0.9025
    Validation MSE: 4.470783233642578
    Validation RMSE: 2.114422559738159
    Validation R²: 0.9168
    Cross-Validation MSE: 4.506539821624756
    Cross-Validation RMSE: 2.0948846340179443
    Cross-Validation R²: 0.9277
```



3L Neural Net:

The Best performance combination was found with ReLU activation function and Stochastic Gradient Descent Optimizer.

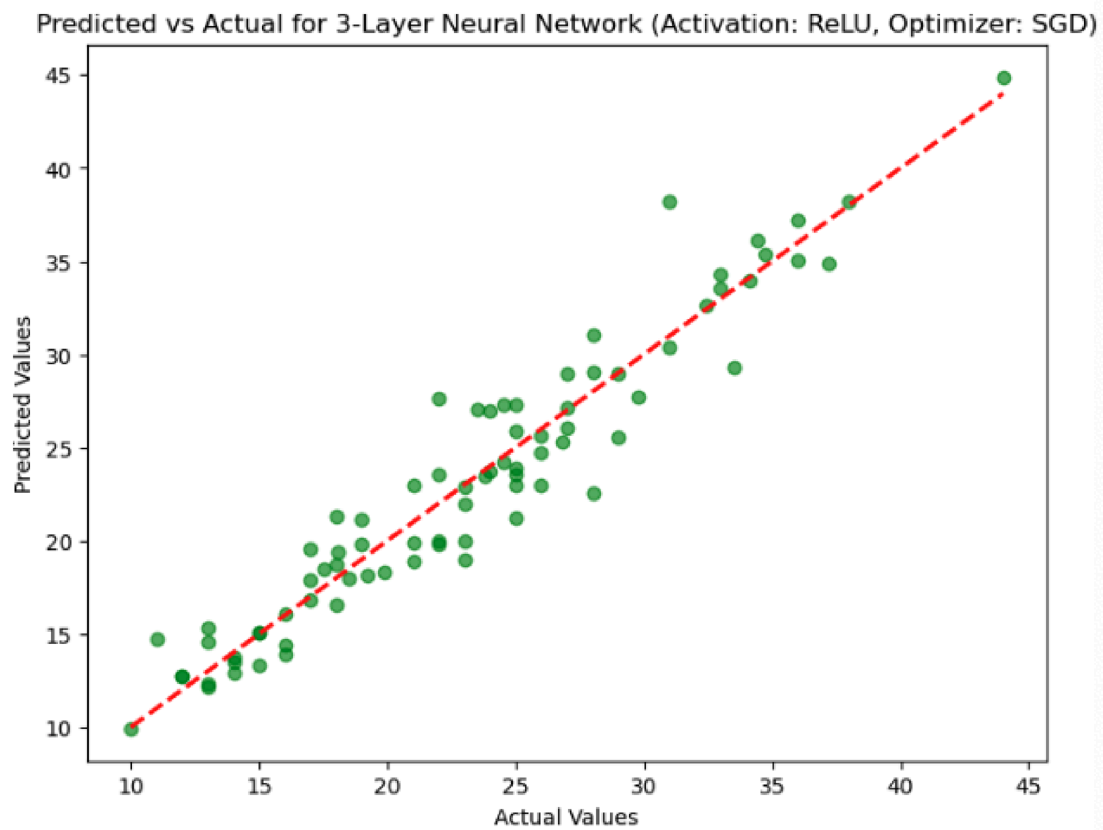
Learning Rate:0.001
Batch Size:32
Hidden Layer Size:128
Input Size:8

```
training 3l net
Epoch [100/100], Loss: 5.104767
cross validation of 3L net
5-fold cross-validation evaluation
Epoch [100/100], Loss: 3.874587
Epoch [100/100], Loss: 3.002533
Epoch [100/100], Loss: 2.314301
Epoch [100/100], Loss: 1.900708
Epoch [100/100], Loss: 1.578532
```

Metrics:
The best performance metric is Cross-validation.

```
3-Layer Neural Network:
  Best R²: 0.9568
  Best Metric: Cross-Validation R²
  Activation and Optimizer: Activation: ReLU, Optimizer: SGD
```

```
Metrics:
  In-Sample MSE: 5.085403919219971
  In-Sample RMSE: 2.2550840377807617
  In-Sample R²: 0.9189
  Validation MSE: 4.543299198150635
  Validation RMSE: 2.1315016746520996
  Validation R²: 0.9155
  Cross-Validation MSE: 2.7511565685272217
  Cross-Validation RMSE: 1.5926520824432373
  Cross-Validation R²: 0.9568
```



XL Neural Net:
The Best performance combination was found with ReLU activation function and Stochastic Gradient Descent Optimizer.

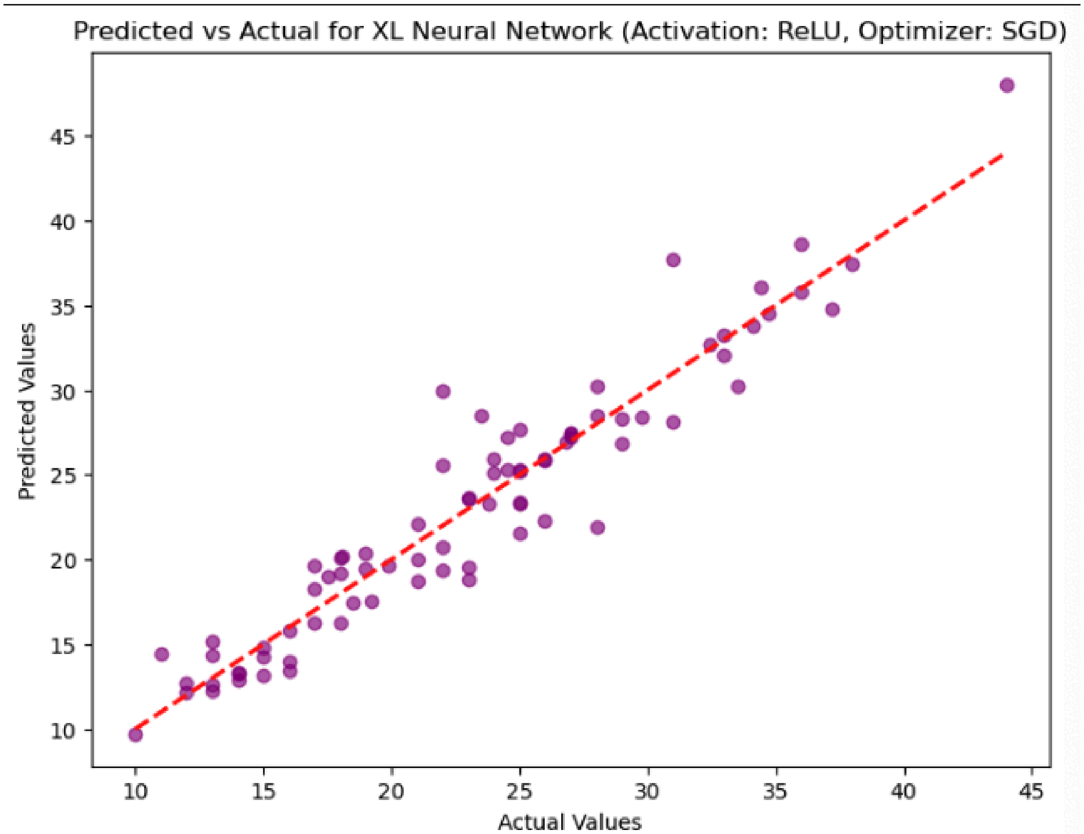
Learning Rate:0.001
Batch Size:32
Hidden Layer Size:128
Input Size:8

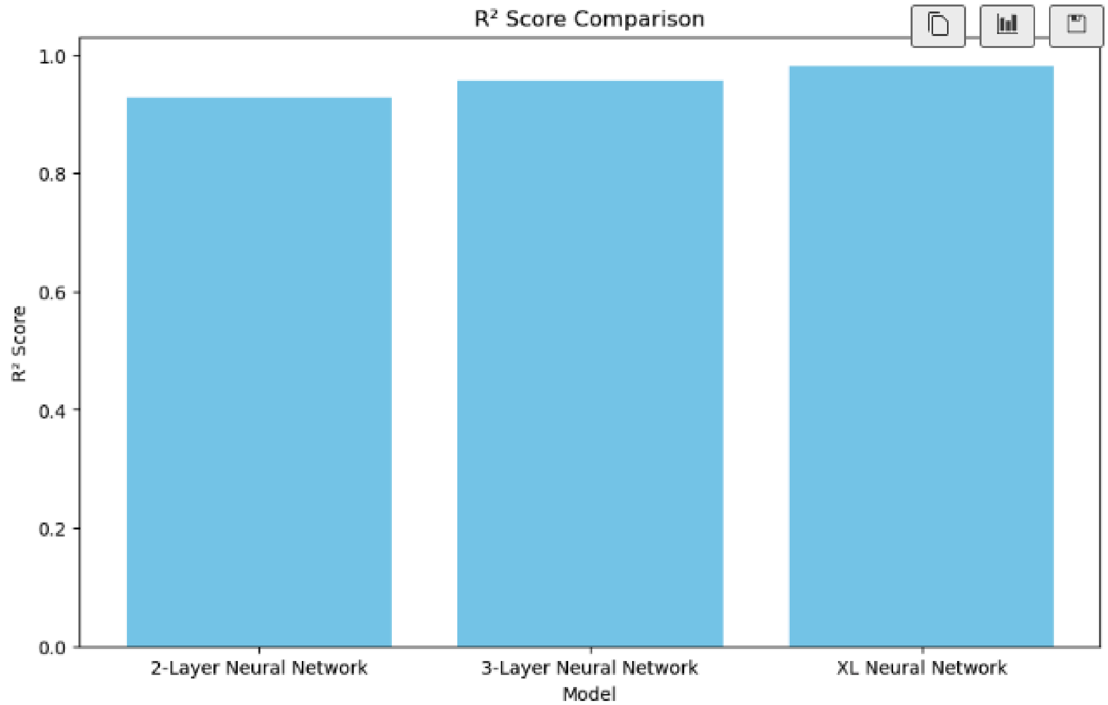
```
training Xl net
Epoch [100/100], Loss: 3.467718
cross validation of XL net
5-fold cross-validation evaluation
```

```
Epoch [100/100], Loss: 2.015366
Epoch [100/100], Loss: 1.341067
Epoch [100/100], Loss: 1.007442
Epoch [100/100], Loss: 0.782131
Epoch [100/100], Loss: 0.623834
```

Metrics:
The best performance was found with Cross-Validation.

```
XL Neural Network:
  Best R²: 0.9818
  Best Metric: Cross-Validation R²
  Activation and Optimizer: Activation: ReLU, Optimizer: SGD
  Metrics:
    In-Sample MSE: 3.450523853302002
    In-Sample RMSE: 1.8575586080551147
    In-Sample R²: 0.9450
    Validation MSE: 5.17495059967041
    Validation RMSE: 2.2748517990112305
    Validation R²: 0.9038
    Cross-Validation MSE: 1.1483126878738403
    Cross-Validation RMSE: 1.0151135921478271
    Cross-Validation R²: 0.9818
```





A **Random Forest Regressor** is initialized with 100 trees (`n_estimators=100`). This means the model will create 100 decision trees, each contributing to the final prediction.

The `random_state=42` ensures reproducibility, meaning each run of this code will yield the same results.

```
Random Forest Test MSE : 4.5168

Random Forest Test RMSE: 2.1253

Random Forest Test R²   : 0.9160
```

GridSearchCV is used to find the optimal hyperparameters, which can improve the model's performance

Below is the parameter grid for searching and best combination of parameters.

param_grid = {

'n_estimators': [100, 200, 300],

'max_depth': [None, 10, 20, 30],

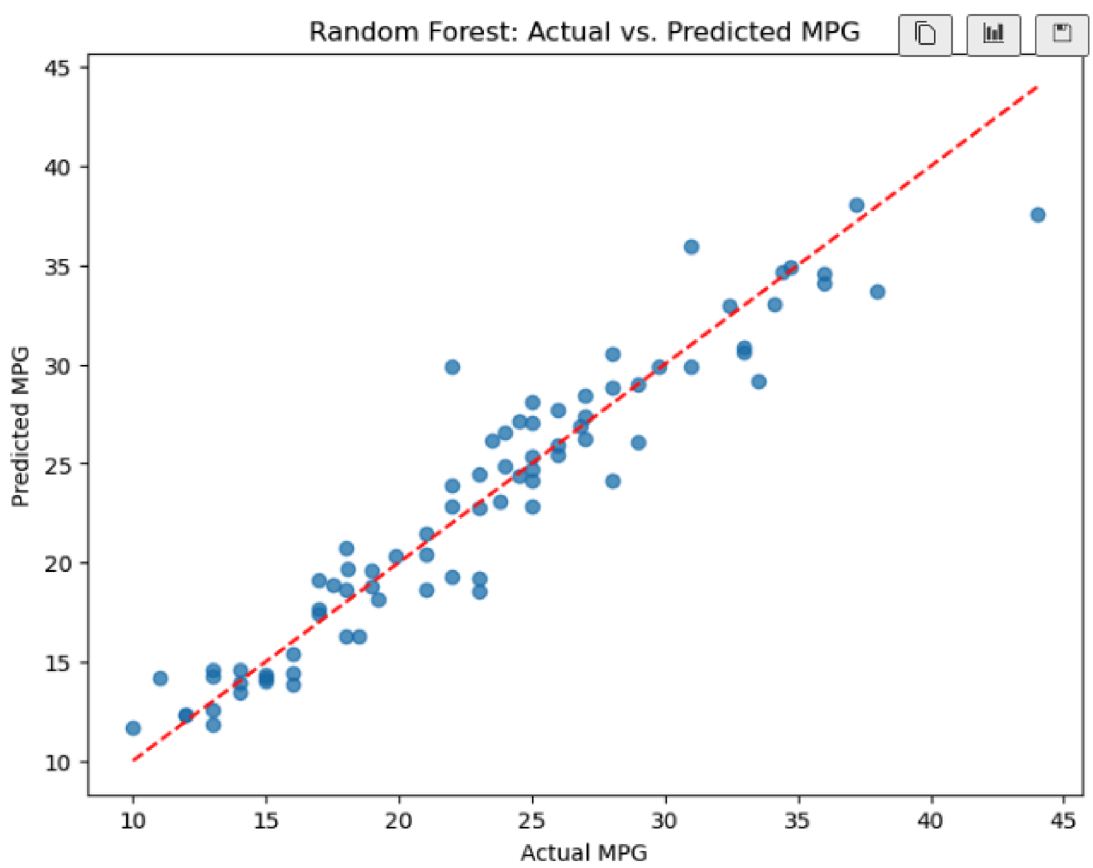
'min_samples_split': [2, 5, 10],

'min_samples_leaf': [1, 2, 4]

}

```
Best Parameters: {'max_depth': None, 'min_samples_leaf': 1,
'min_samples_split': 2, 'n_estimators': 200}

Best CV MSE: 0.8527

Best Random Forest Test MSE: 4.6478
```

Below is the graph for actual vs predicted values using Random Forest Regressor.



The Cross Validation scores are:

```
Cross-Validated R² Scores: [0.89809161 0.86080208 0.84204864 0.77571794
0.88691443]

Average Cross-Validated R² Score: 0.8527
```

Seoul Bike Data

Data Preprocessing Dataset does not have any identifier columns, Null values, hence no imputations are done. However, the dataset contains 3 categorical columns (Seasons, Holiday and Functioning day), these are converted into categorical codes. The Hour column is converted into categorical by cutting into Morning, afternoon, evening and night.

```
Rented Bike Count int64

Hour int64

Temperature(°C) float64

Humidity(%) int64

Wind speed (m/s) float64

Visibility (10m) int64

Dew point temperature(°C) float64

Solar Radiation (MJ/m2) float64

Rainfall(mm) float64

Snowfall (cm) float64

Seasons int32

Holiday int32

Functioning Day int32

hourOfDay int32

dtype: object
```

The outliers in the dataset are checked by Z scores outlier detection method. We have found below outliers for snowfall, rainfall, wind speed, however these scenarios occur rarely but they are a part of realistic data and hence we need not remove these as outliers.

When we have checked the variance and covariance, we observed the covariance Dew Point Temperature with Rented Bike count is close to 0, also Dew point temperature and Temperature are having high collinear values, hence the dew point temperature will be removed during model creation. Also, the date column is dropped as it needs to be converted to timeseries to use in our models.

	Variance
Rented Bike Count	416021.733390
Temperature(°C)	142.678850
Humidity(%)	414.627875
Wind speed (m/s)	1.073918

Visibility (10m) 370027.323001
Dew point temperature(°C) 170.573247
Solar Radiation (MJ/m2) 0.754720
Rainfall(mm) 1.272819

Snowfall (cm) 0.190747

Seasons 1.241906

Holiday 0.046888

Functioning Day 0.032545

hourOfDay 2.833657

Rented Bike Count Temperature(°C) Humidity(%)
Rented Bike Count 416021.73339 4149.257754 -2623.853782
Temperature(°C) 4149.257754 142.67885 38.763038 Humidity(%)
-2623.853782 38.763038 414.627875 Wind speed (m/s) 80.950203
-0.448739 -7.10454 Visibility (10m) 78187.849382 252.817084
-6726.950421 Dew point temperature(°C) 3199.299111 142.400017
142.782065 Solar Radiation (MJ/m2) 146.717508 3.668334 -8.171237
Rainfall(mm) -89.558657 0.677602 5.430677
Snowfall (cm) -39.946114 -1.139387 0.962098
Seasons 258.539368 7.874302 4.294195 Holiday -10.103104
-0.144665 -0.221685

Functioning Day 23.730746 -0.10811 -0.076408 hourOfDay

343.54972 3.120162 -11.464437

Wind speed (m/s) Visibility (10m)
Rented Bike Count 80.950203 78187.849382 Temperature(°C)
-0.448739 252.817084 Humidity(%) -7.10454 -6726.950421
Wind speed (m/s) 1.073918 108.11466 Visibility (10m)
108.11466 370027.323001 Dew point temperature(°C)
-2.388639 -1403.253586 Solar Radiation (MJ/m2) 0.29914
79.130141

Rainfall(mm) -0.023002
-115.040313

Snowfall (cm) -0.001609 -32.330842

Seasons -0.19267
75.906527

Holiday 0.005165

4.185096

Functioning Day 0.000942 -2.853224

hourOfDay 0.607215 86.007307

Dew point temperature(°C) Solar Radiation(MJ/m2)

Rented Bike Count 3199.299111 146.717508 Temperature(°C)

142.400017 3.668334 Humidity(%) 142.782065 -8.171237 Wind speed

(m/s) -2.388639 0.29914 Visibility (10m) -1403.253586 79.130141 Dew

point temperature(°C) 170.573247 1.070865 Solar Radiation (MJ/m2)

1.070865 0.75472

Rainfall(mm)

1.85062 -0.072813

Snowfall (cm) -0.860668 -0.027432 Seasons 8.476852 0.091664

Holiday -0.188799 -0.000955 Functioning Day -0.124492

-0.001201 hourOfDay -0.380226 0.6462

Rainfall(mm) Snowfall (cm) Seasons Holiday

Rented Bike Count -89.558657 -39.946114 258.539368 10.103104
Temperature(°C) 0.677602 -1.139387 7.874302 -0.144665 Humidity(%)
5.430677 0.962098 4.294195 -0.221685 Wind speed (m/s) -0.023002
-0.001609 -0.19267 0.005165 Visibility (10m) -115.040313 -32.330842
75.906527 4.185096

Dew point temperature(°C) 1.85062 -0.860668 8.476852 -0.188799 Solar
Radiation (MJ/m2) -0.072813 -0.027432 0.091664 -0.000955

Rainfall(mm) 1.272819 0.004188 0.042059 -0.003486

Snowfall (cm) 0.004188 0.190747 -0.070796 -0.001191
Seasons 0.042059 -0.070796 1.241906 -0.013903

Holiday -0.003486 -0.001191 -0.013903 0.046888
-0.001079
Functioning Day
0.000418 0.002528 -0.039421

hourOfDay 0.028514 -0.005217 -0.0 -0.0 Functioning Day hourOfDay

Rented Bike Count 23.730746 343.54972

Temperature(°C)

-0.10811 3.120162

Humidity(%) -0.076408 -11.464437

Wind speed (m/s)
0.000942 0.607215

Visibility (10m) -2.853224 86.007307 Dew point temperature(°C) -0.124492 -0.380226 Solar Radiation (MJ/m2) -0.001201 0.6462 Rainfall(mm) 0.000418 0.028514 Snowfall (cm) 0.002528 -0.005217
Seasons -0.039421 -0.0
Holiday -0.001079 -0.0
Functioning Day 0.032545 0.001085 hourOfDay 0.001085 2.833657

----

feature VIF
0 Rented Bike Count 4.385017
1 Temperature(°C) 46.418062
2 Humidity(%) 22.407409
3 Wind speed (m/s) 4.866986
4 Visibility (10m) 9.719935
5 Dew point temperature(°C) 25.000918 6 Solar Radiation (MJ/m2) 3.020596
7 Rainfall(mm) 1.108003
8 Snowfall (cm) 1.123911
9 Seasons 5.117387
10 Holiday 1.063033
11 Functioning Day 32.796315
12 hourOfDay 2.625944

Exploratory Data Analysis Scatterplots and histograms are generated using matplotlib for each column present in the dataset. Most of the columns are continuous apart from a few variables such as Snowfall, Rainfall and categorical variables such as Seasons, Holiday and Functioning day.

When we use the describe() function on the dataset, we can see the mean of Rainfall, snowfall is close to 0, but the Rented bike count covariance with these columns is high, hence we will include these columns in our models.

DEBUG @ **RegressionTreeRF**.train: for tree6 ===
()
LinkedHashMap(**rSq -> 0.962847**, rSqBar -> 0.952032, sst -> 11138737787.460625, sse -> 413837637.347054, sde -> 486.147147, mse0 -> 236208.697116, rmse -> 486.013063, mae -> 343.342250, dfm -> 395.000000, df -> 1357.000000, fStat -> 89.031978, aic -> -12542.468462, bic ->

-10376.937206, mape -> Infinity, smape -> 17.193500)

Python:
2L Neural Net:

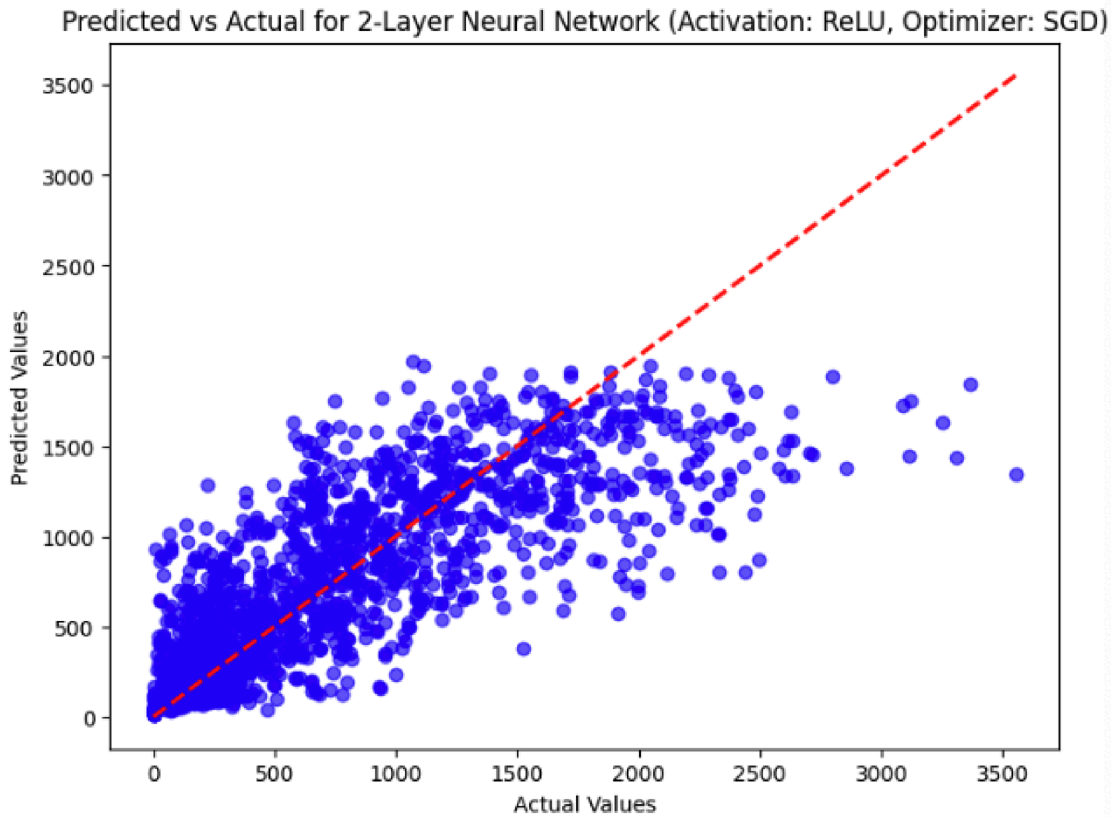
The Best performance combination was found with ReLU activation function and Stochastic Gradient Descent Optimizer.

Learning Rate:1e-5
Batch Size:4096
Hidden Layer Size:512
Input Size:11

```
training 2L net
Epoch [100/100], Loss: 151656.843750
5-fold cross-validation evaluation for 2L net
Epoch [100/100], Loss: 145072.476562
Epoch [100/100], Loss: 141301.726562
Epoch [100/100], Loss: 141973.593750
Epoch [100/100], Loss: 139124.375000
Epoch [100/100], Loss: 137281.640625
```

Metrics:
The best performance was found with Cross Validation.

```
Best Performance for Each Neural Network:

 2-Layer Neural Network:
  Best R²: 0.6602
  Activation and Optimizer: Activation: ReLU, Optimizer: SGD
  Metrics:
    In-Sample MSE: 151686.234375
    In-Sample RMSE: 389.46917724609375
    In-Sample R²: 0.6352
    Validation MSE: 149774.140625
    Validation RMSE: 387.00665283203125
    Validation R²: 0.6405
    Cross-Validation MSE: 141125.546875
    Cross-Validation RMSE: 375.6151428222656
    Cross-Validation R²: 0.6602
```



3L Neural Net:

The Best performance combination was found with SELU activation function and Stochastic Gradient Descent Optimizer.

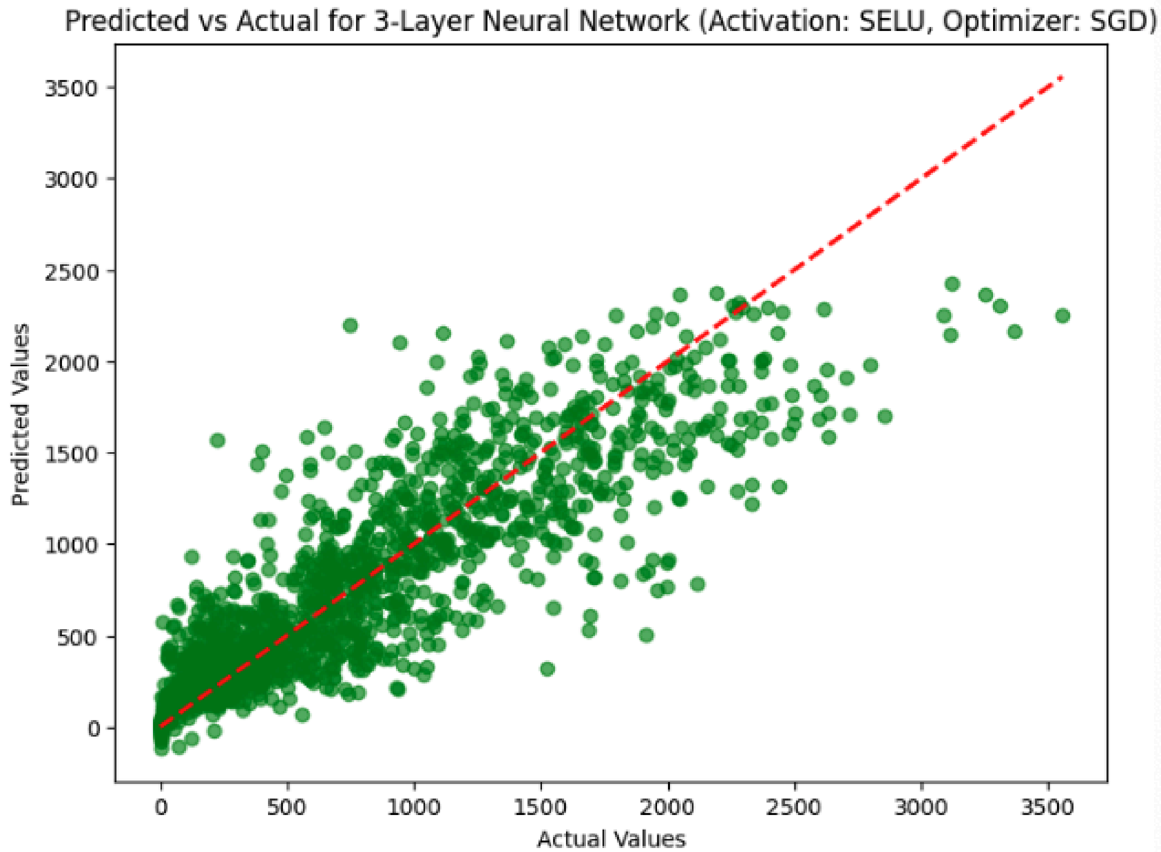
Learning Rate:1e-5
Batch Size:4096
Hidden Layer Size:512
Input Size:11

```
training 3l net
Epoch [100/100], Loss: 90142.054688
5-fold cross-validation evaluation
Epoch [100/100], Loss: 75487.250000
Epoch [100/100], Loss: 68721.410156
Epoch [100/100], Loss: 64444.488281
Epoch [100/100], Loss: 61178.160156
Epoch [100/100], Loss: 59334.775391
```

Metrics:
The best metric was found with Cross validation.

```
3-Layer Neural Network:
  Best R²: 0.8425
  Activation and Optimizer: Activation: SELU, Optimizer: SGD
```

```
Metrics:
  In-Sample MSE: 90175.5234375
  In-Sample RMSE: 300.2923889160156
  In-Sample R²: 0.7831
  Validation MSE: 94231.7421875
  Validation RMSE: 306.9718933105469
  Validation R²: 0.7738
  Cross-Validation MSE: 65384.8125
  Cross-Validation RMSE: 255.5073699951172
  Cross-Validation R²: 0.8425
```



XL Neural Net:
The Best performance combination was found with SELU activation function and Stochastic Gradient Descent Optimizer.

Learning Rate:1e-5
Batch Size:4096
Hidden Layer Size:512
Input Size:11

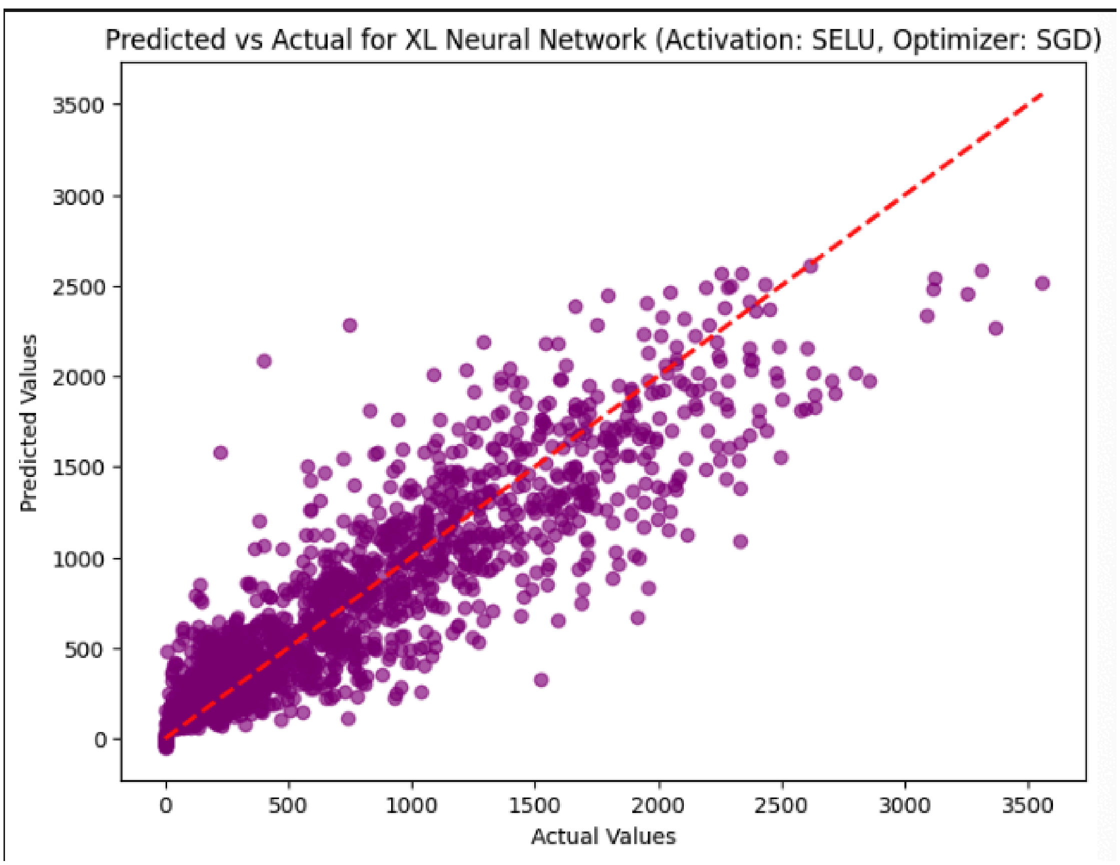
```
training Xl net
Epoch [100/100], Loss: 73971.078125
```

```
5-fold cross-validation evaluation
Epoch [100/100], Loss: 60883.871094
Epoch [100/100], Loss: 55344.636719
Epoch [100/100], Loss: 51475.718750
Epoch [100/100], Loss: 47158.734375
Epoch [100/100], Loss: 43589.261719
```

Metrics:

The best metric was found with Cross validation.

```
XL Neural Network:
  Best R²: 0.8753
  Activation and Optimizer: Activation: SELU, Optimizer: SGD
  Metrics:
    In-Sample MSE: 74270.4140625
    In-Sample RMSE: 272.5260009765625
    In-Sample R²: 0.8214
    Validation MSE: 79367.546875
    Validation RMSE: 281.72247314453125
    Validation R²: 0.8095
    Cross-Validation MSE: 51764.8515625
    Cross-Validation RMSE: 227.0684814453125
    Cross-Validation R²: 0.8753
```



A **Random Forest Regressor** is initialized with 100 trees (`n_estimators=100`). This means the model will create 100 decision trees, each contributing to the final prediction.

The `random_state=42` ensures reproducibility, meaning each run of this code will yield the same results.

```
Random Forest Test MSE : 76185.7484
Random Forest Test RMSE: 276.0177
Random Forest Test R²   : 0.8171
```

GridSearchCV is used to find the optimal hyperparameters, which can improve the model's performance

Below is the parameter grid for searching and best combination of parameters.

param_grid = {

'n_estimators': [100, 200, 300],

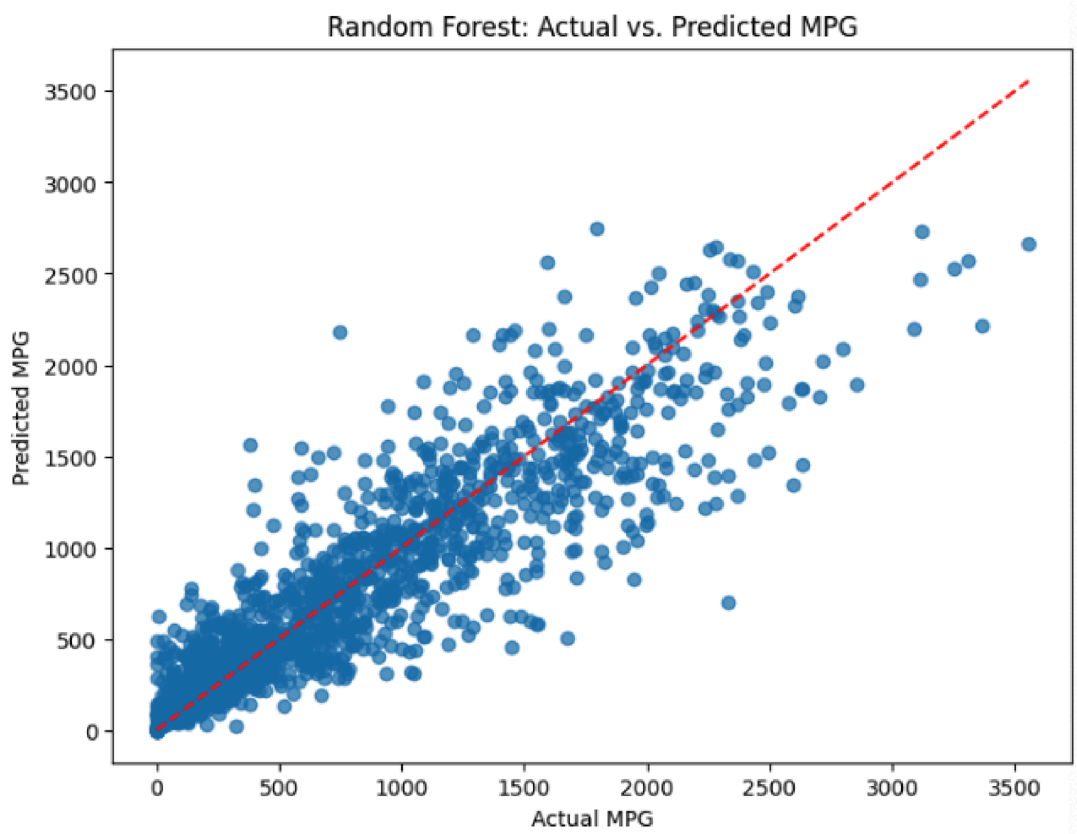
'max_depth': [None, 10, 20, 30],

'min_samples_split': [2, 5, 10],

'min_samples_leaf': [1, 2, 4]

```
Best Parameters: {'max_depth': None, 'min_samples_leaf': 1,
'min_samples_split': 2, 'n_estimators': 100}

Best CV MSE: 0.8283

Best Random Forest Test MSE: 76185.7484
```

Below is the Random Forest Regressor graph:



The Cross validation scores are：

```
Cross-Validated R² Scores: [0.81055458 0.83807724 0.82366025 0.84661243
0.82267248]

Average Cross-Validated R² Score: 0.8283
```

Feature Selection :
Feature Selection has been included in the code for all forward, backward and stepwise.

Bonus 1 : Feature Selection - Implemented for all three datasets.

Note: As discussed with the professor and agreed on implementing Feature selection for AutoMPG would be fine. Whereas we implemented for all three however provided results for only Airfoil and AutoMPG here.

Bonus 2 : Random Forest Regressor
It has been implemented in both Scala and Python and results has been reported here.

Tabular representation of results:

Model / Dataset	AirFoil		AutoMPG		SeoulBike	
	Scala	Python	Scala	Python	Scala	Python
2 Layer Network	0.533	0.6960	0.822	0.9277	0.692	0.6602
3 Layer Network	0.805	0.8630	0.924	0.9568	0.861	0.8425
X Layer Network	0.913	0.9323	0.928	0.9818	0.867	0.8753
RF Regressor	0.820	0.9342	0.933	0.9160	0.962	0.8283