Future Value Prediction Using ML & NN

**Boston House Prices**

Abeykoon Mudiyanselage, Nirmal P.

20/21

Contents

[Introduction 3](#_Toc55408459)

[Dataset 3](#_Toc55408460)

[Data prepossessing 4](#_Toc55408461)

[Data Visualization 5](#_Toc55408462)

[Head of the dataset 5](#_Toc55408463)

[Correlation heatmap 6](#_Toc55408464)

[Scatterplot matrix 6](#_Toc55408465)

[Histograms 7](#_Toc55408466)

[Multivariate Regression 8](#_Toc55408467)

[Model brief 8](#_Toc55408468)

[Predictions and accuracy 9](#_Toc55408469)

[Improvements and suggestions 11](#_Toc55408470)

[Changing the training dataset randomness 11](#_Toc55408471)

[Changing the number of training recodes 14](#_Toc55408472)

[Removing features 16](#_Toc55408473)

[Decision Tree - Regression 17](#_Toc55408474)

[Model brief 17](#_Toc55408475)

[Predictions and accuracy 18](#_Toc55408476)

[Improvements and suggestions 19](#_Toc55408477)

[Changing the training dataset randomness 19](#_Toc55408478)

[Changing the number of recodes 23](#_Toc55408479)

[Neural Network - Sequential 26](#_Toc55408480)

[Model brief 26](#_Toc55408481)

[Predictions and accuracy 28](#_Toc55408482)

[Improvements and suggestions 30](#_Toc55408483)

[Changing number of epochs 30](#_Toc55408484)

[Changing number of neurons and layers 31](#_Toc55408485)

[References 32](#_Toc55408486)

# Introduction

In this report, we are going to discuss how we can use machine learning and neural networks to predict house prices. We are going to have a detailed discussion about the dataset that is used and the development of each ML and NN model. Furthermore, we are going to discuss the accuracy of each model and compare them with each other. Finally, we will discuss how we can improve each model.

In the following sections, we will look at how we can predict house prices using Multivariate regression, Decision Tree – Regression, and sequential neural networks.

# Dataset

For this task, we are going to use the Sklearn dataset Boston house prices. To import this dataset, we can use the following line of code. from sklearn.datasets import load\_boston

There are 13 attributes in the dataset. We can view information about attributes with print(boston.DESCR).

Text

Description automatically generated

This dataset has 506 instance data.

## Data prepossessing

There is not much prepossessing required in this dataset. The data is well organized and there are no null values.

Null value percentage (Calculated by using a for loop in python and using .isnull()):

CRIM 0.0%

ZN 0.0%

INDUS 0.0%

CHAS 0.0%

NOX 0.0%

RM 0.0%

AGE 0.0%

DIS 0.0%

RAD 0.0%

TAX 0.0%

PTRATIO 0.0%

B 0.0%

LSTAT 0.0%

For the neural network model, sklearn.prepossessing used to normalise the dataset (scikit learn, 2020).

Attributes array before prepossessing.

[[6.3200e-03 1.8000e+01 2.3100e+00 ... 1.5300e+01 3.9690e+02 4.9800e+00]

[2.7310e-02 0.0000e+00 7.0700e+00 ... 1.7800e+01 3.9690e+02 9.1400e+00]

[2.7290e-02 0.0000e+00 7.0700e+00 ... 1.7800e+01 3.9283e+02 4.0300e+00]

...

[6.0760e-02 0.0000e+00 1.1930e+01 ... 2.1000e+01 3.9690e+02 5.6400e+00]

[1.0959e-01 0.0000e+00 1.1930e+01 ... 2.1000e+01 3.9345e+02 6.4800e+00]

[4.7410e-02 0.0000e+00 1.1930e+01 ... 2.1000e+01 3.9690e+02 7.8800e+00]

Attributes array after using preprocessing.MinMaxScaler() to normalise data.

[[0.00000000e+00 1.80000000e-01 6.78152493e-02 ... 2.87234043e-01

1.00000000e+00 8.96799117e-02]

[2.35922539e-04 0.00000000e+00 2.42302053e-01 ... 5.53191489e-01

1.00000000e+00 2.04470199e-01]

[2.35697744e-04 0.00000000e+00 2.42302053e-01 ... 5.53191489e-01

9.89737254e-01 6.34657837e-02]

...

[6.11892474e-04 0.00000000e+00 4.20454545e-01 ... 8.93617021e-01

1.00000000e+00 1.07891832e-01]

[1.16072990e-03 0.00000000e+00 4.20454545e-01 ... 8.93617021e-01

9.91300620e-01 1.31070640e-01]

[4.61841693e-04 0.00000000e+00 4.20454545e-01 ... 8.93617021e-01

1.00000000e+00 1.69701987e-01]]

we need to add the target values or house prices in this instance manually to our data frame using the following lines of code. This makes easer to visualise data.

data['PRICE'] = boston.target

## Data Visualization

In this section, we will look at the visualization of the data and what information and correlations we can see from it.

### Head of the dataset

With the pre-processing we did earlier the head of the dataset will look like this (without normalisation).

CRIM ZN INDUS CHAS NOX ... TAX PTRATIO B LSTAT PRICE

0 0.00632 18.0 2.31 0.0 0.538 ... 296.0 15.3 396.90 4.98 24.0

1 0.02731 0.0 7.07 0.0 0.469 ... 242.0 17.8 396.90 9.14 21.6

2 0.02729 0.0 7.07 0.0 0.469 ... 242.0 17.8 392.83 4.03 34.7

3 0.03237 0.0 2.18 0.0 0.458 ... 222.0 18.7 394.63 2.94 33.4

4 0.06905 0.0 2.18 0.0 0.458 ... 222.0 18.7 396.90 5.33 36.2

But without adding the house prices in the pre-processing stage the head of the dataset would have looked like this.

CRIM ZN INDUS CHAS NOX ... RAD TAX PTRATIO B LSTAT

0 0.00632 18.0 2.31 0.0 0.538 ... 1.0 296.0 15.3 396.90 4.98

1 0.02731 0.0 7.07 0.0 0.469 ... 2.0 242.0 17.8 396.90 9.14

2 0.02729 0.0 7.07 0.0 0.469 ... 2.0 242.0 17.8 392.83 4.03

3 0.03237 0.0 2.18 0.0 0.458 ... 3.0 222.0 18.7 394.63 2.94

4 0.06905 0.0 2.18 0.0 0.458 ... 3.0 222.0 18.7 396.90 5.33

### Correlation heatmap

A screen shot of a computer

Description automatically generated

Figure 1: Correlation heat map

When looking at this diagram we can see a strong correlation between TAX and RAD attributes. Also, there is a good correlation between TAX and INDUS attributes.

### Scatterplot matrix

Diagram

Description automatically generated

Figure 2: Scatter matrix

This chart shows a scatter matrix between numerical values. Also, this shows a histogram of all the numerical values. We can confirm the correlations between attributes by examining this scatterplot matrix. (Géron, 2020)

### Histograms

Chart, diagram

Description automatically generated

Figure 3:Range of integer data

All the numerical values are displayed in the histogram. According to the chart, the house prices are varied while mid-range priced houses being common. We can see there is a strong match between this and the TAX feature. (Géron, 2020)

# Multivariate Regression

## Model brief

The multivariate regression model is an extinction simple linear regression model. If the data is fit to a line it is called simple linear regression. But when the data is fitted to a plain then it is called multivariate regression. (Figure 4:Multivariate Regression)

Simple regression formula multivariable regression

The multivariate regression model makes predictions by computing a weighted sum of input features plus the intercept also known as bias term. (Géron, 2020)

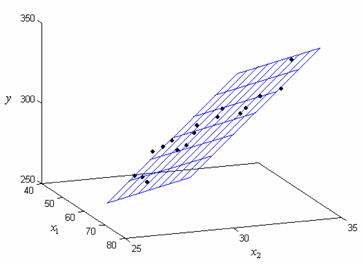


Figure 4:Multivariate Regression with 3 axias

## Predictions and accuracy

In the following section, we can see some of the values predicted by the Multivariate model compared to the actual price (in USD). Furthermore, accuracy, mean squared error(MSE) and root mean square error (RMSE) also listed below.

Accuracy: 71.09991947411514 %

Mean Squared Error: 28.094923004652372

Root Mean Square Error: 5.300464414053958

Actual: 50.0 Prediction: 24.2

Actual: 17.4 Prediction: 16.0

Actual: 23.1 Prediction: 15.6

Actual: 42.8 Prediction: 29.3

Actual: 23.9 Prediction: 27.9

Actual: 22.4 Prediction: 22.6

Actual: 45.4 Prediction: 38.8

Actual: 48.3 Prediction: 36.8

Actual: 20.3 Prediction: 22.3

Actual: 17.8 Prediction: 19.9

Actual: 17.4 Prediction: 17.4

Actual: 8.8 Prediction: 3.5

Actual: 35.4 Prediction: 33.8

Actual: 23.1 Prediction: 24.0

Actual: 13.1 Prediction: 16.1

Actual: 46.7 Prediction: 34.0

Actual: 33.1 Prediction: 34.5

Actual: 20.6 Prediction: 16.7

Actual: 24.8 Prediction: 31.5

Actual: 17.8 Prediction: 22.6

Actual: 23.1 Prediction: 25.5 …

Chart, scatter chart

Description automatically generated

Figure 5:Multivariate Regression results

In a scenario, with 100% accuracy, this chart would give us a straight line(Linear Line) when we compare actual values and predicted values.

In the next section, we will discuss what methods we can implement to increase the accuracy of this base model.

## Improvements and suggestions

### Changing the training dataset randomness

In this model, training dataset randomness is determined by the random state parameter when we use train\_test\_split. In the base model, it was Null. This means the random state was random. In the following section, we will look at what accuracies we can get by pre deciding the randomness of the training dataset. For that, we are using a for loop to test random state between 0 and 10.

Random State= 0

Accuracy: 58.92223849182514 %

Mean Squared Error: 33.448979997676474

Root Mean Square Error: 5.78350931508513

Actual: 22.6 Prediction: 24.9

Actual: 50.0 Prediction: 23.7

Actual: 23.0 Prediction: 29.4

Actual: 8.3 Prediction: 12.1

Actual: 21.2 Prediction: 21.4

Random State= 1

Accuracy: 76.3417443213847 %

Mean Squared Error: 23.380836480270247

Root Mean Square Error: 4.835373458200539

Actual: 28.2 Prediction: 32.7

Actual: 23.9 Prediction: 28.1

Actual: 16.6 Prediction: 18.0

Actual: 22.0 Prediction: 21.5

Actual: 20.8 Prediction: 18.8

Random State= 2

Accuracy: 77.89207451814423 %

Mean Squared Error: 18.495420122448355

Root Mean Square Error: 4.30063020061576

Actual: 20.2 Prediction: 23.0

Actual: 15.3 Prediction: 21.2

Actual: 37.3 Prediction: 33.7

Actual: 32.5 Prediction: 31.6

Actual: 8.8 Prediction: 3.2

Random State= 3

Accuracy: 79.52617563243875 %

Mean Squared Error: 16.943073013833633

Root Mean Square Error: 4.1161964255649455

Actual: 44.8 Prediction: 37.5

Actual: 17.1 Prediction: 18.8

Actual: 17.8 Prediction: 22.4

Actual: 33.1 Prediction: 32.8

Actual: 21.9 Prediction: 24.3

Random State= 4

Accuracy: 72.63451459702503 %

Mean Squared Error: 25.41958712682191

Root Mean Square Error: 5.041784121402057

Actual: 16.5 Prediction: 12.1

Actual: 24.8 Prediction: 27.0

Actual: 17.4 Prediction: 17.6

Actual: 19.3 Prediction: 18.2

Actual: 37.6 Prediction: 36.9

Random State= 5

Accuracy: 73.3449214745311 %

Mean Squared Error: 20.86929218377053

Root Mean Square Error: 4.5682920423031765

Actual: 37.6 Prediction: 37.6

Actual: 27.9 Prediction: 32.1

Actual: 22.6 Prediction: 27.1

Actual: 13.8 Prediction: 5.7

Actual: 35.2 Prediction: 35.1

Random State= 6

Accuracy: 68.39026890069033 %

Mean Squared Error: 27.22333372429668

Root Mean Square Error: 5.21759846330634

Actual: 15.0 Prediction: 25.4

Actual: 23.1 Prediction: 25.1

Actual: 30.1 Prediction: 30.2

Actual: 22.4 Prediction: 23.4

Actual: 20.1 Prediction: 19.9

Random State= 7

Accuracy: 57.85415472763411 %

Mean Squared Error: 34.05648134887459

Root Mean Square Error: 5.835793120808395

Actual: 21.7 Prediction: 23.0

Actual: 18.5 Prediction: 19.2

Actual: 22.2 Prediction: 19.9

Actual: 20.4 Prediction: 19.2

Actual: 8.8 Prediction: 4.7

Random State= 8

Accuracy: 70.7962796713453 %

Mean Squared Error: 21.63823440647877

Root Mean Square Error: 4.651691563988177

Actual: 18.5 Prediction: 19.2

Actual: 12.7 Prediction: 11.0

Actual: 21.9 Prediction: 38.4

Actual: 22.0 Prediction: 27.2

Actual: 50.0 Prediction: 41.0

Random State= 9

Accuracy: 76.60111574904009 %

Mean Squared Error: 23.676620280791305

Root Mean Square Error: 4.865862747837356

Actual: 21.4 Prediction: 20.4

Actual: 8.4 Prediction: 15.1

Actual: 33.1 Prediction: 34.0

Actual: 13.6 Prediction: 14.2

Actual: 18.5 Prediction: 19.5

Figure 6:Accuracy in each random state

When we analyze the results, we can see that random sate 3 produces the best test data split. It produces the best prediction results with an accuracy of 79.64%. This is an 8% increase compared to the base accuracy we saw with the first model.

### Changing the number of training recodes

In this section, we will discuss how the number of training recodes going to affect the accuracy of the model. we will split the dataset and see how the accuracy change related to that.

The original model uses 404 recodes to change the model.

For this test, we will change the value of the “test\_size” parameter in the train\_test\_split to reduce the number of training recodes.

Number of training recodes= 404

Accuracy: 71.09991947411514 %

Mean Squared Error: 28.094923004652372

Root Mean Square Error: 5.300464414053958

Number of training recodes= 303

Accuracy: 73.69068178995884 %

Mean Squared Error: 21.59771610673382

Root Mean Square Error: 4.647334301159518

Number of training recodes= 202

Accuracy: 71.75822011858357 %

Mean Squared Error: 24.169240779805182

Root Mean Square Error: 4.916222206105536

Number of training recodes= 101

Accuracy: 70.63759044921267 %

Mean Squared Error: 25.256209554021062

Root Mean Square Error: 5.025555646296344

Number of training recodes= 50

Accuracy: 61.309805255527074 %

Mean Squared Error: 33.13190034840654

Root Mean Square Error: 5.756031649357614

Number of training recodes= 25

Accuracy: 52.65887423955115 %

Mean Squared Error: 39.60299724334496

Root Mean Square Error: 6.293091231131562

Figure 7:Accuracy compared to the number of recodes

When we study the graph, we can see although there is a loss of accuracy it isn’t notable until the number of recodes reaches the threshold of 100. After that point accuracy of the model drops rapidly.

### Removing features

In this section, we will see how the feature correlation affects the accuracy of the model. For this, we will drop the TAX, RAD, and NOX features from the dataset and see how the accuracy changes from the base accuracy of 71.09991947411514 %.

After dropping these well-correlated features the accuracy drops to 61.67771314870223 %

# Decision Tree - Regression

## Model brief

Decision tree regression builds models in a form of a tree structure. The model breakdown the training dataset into small sub-sections. If graphed the final tree of the decision tree will be a tree with decision nodes and leaf nodes. A decision node will represent the attributes tested. A decision node may have two or more branches. The leaf node will represent the decisions made by the tree as a numerical value. The top predictor is called the root node. (AnthonyJ.Myles, 2004)

Chart

Description automatically generated

Figure 8:Part of the tree that was generated for this task.

Created using sklearn.export\_graphviz

Models make predictions by asking several questions from the dataset until the model can make a prediction. These questions are all in a true or false form. When the model reaches a point when it's confidant enough it will make a prediction.

Commonly decision tree regression is used for classification it can be used for prediction. (Geron, 2019)

## Predictions and accuracy

In the following section, we can see some of the values predicted by the Decision Tree - Regression model compared to the actual price (in USD). The Accuracy, MSE, and RMSE are also listed below.

Accuracy: 75.91407750628234 %

Mean Squared Error: 19.880000000000003

Root Mean Square Error: 4.458699361921591

Actual: 28.2 Prediction: 23.9

Actual: 24.3 Prediction: 27.5

Actual: 12.0 Prediction: 17.2

Actual: 32.4 Prediction: 33.1

Actual: 21.7 Prediction: 21.2

Actual: 17.4 Prediction: 23.8

Actual: 34.9 Prediction: 32.9

Actual: 29.6 Prediction: 33.1

Actual: 19.8 Prediction: 20.1

Actual: 15.6 Prediction: 16.1

Actual: 14.5 Prediction: 18.4

Actual: 17.4 Prediction: 19.4

Actual: 23.9 Prediction: 22.6

Actual: 21.9 Prediction: 35.2

Actual: 17.2 Prediction: 17.5

Actual: 14.0 Prediction: 15.2

Actual: 21.5 Prediction: 18.8

Actual: 42.8 Prediction: 35.2

Actual: 24.0 Prediction: 32.5 …

Chart, scatter chart

Description automatically generated

Figure 9:Decision Tree – Regression

If the accuracy was 100%, we will see a linear line and in this case, we can see the scatter of data points meaning the accuracy of less than 100%.

## Improvements and suggestions

### Changing the training dataset randomness

In this section, we will look at how the random state of the training dataset affects the accuracy of the model.

Random State= 0

Accuracy: 77.53554263293289 %

Mean Squared Error: 20.879411764705885

Root Mean Square Error: 4.569399497166546

Actual: 20.1 Prediction: 20.6

Actual: 18.3 Prediction: 19.8

Actual: 44.8 Prediction: 50.0

Actual: 21.8 Prediction: 20.6

Actual: 26.2 Prediction: 31.5

Random State= 1

Accuracy: 79.7028031845233 %

Mean Squared Error: 22.113921568627454

Root Mean Square Error: 4.702544159136355

Actual: 33.8 Prediction: 30.3

Actual: 20.9 Prediction: 22.0

Actual: 20.6 Prediction: 22.7

Actual: 18.6 Prediction: 18.2

Actual: 8.5 Prediction: 7.0

Random State= 2

Accuracy: 82.70039067278516 %

Mean Squared Error: 13.819607843137256

Root Mean Square Error: 3.717473314381323

Actual: 31.2 Prediction: 23.9

Actual: 25.3 Prediction: 24.7

Actual: 19.9 Prediction: 20.0

Actual: 32.4 Prediction: 35.1

Actual: 36.5 Prediction: 36.0

Random State= 3

Accuracy: 56.35624740634147 %

Mean Squared Error: 30.58039215686274

Root Mean Square Error: 5.5299540827083495

Actual: 22.0 Prediction: 23.2

Actual: 24.4 Prediction: 28.7

Actual: 19.5 Prediction: 21.4

Actual: 20.5 Prediction: 21.7

Actual: 19.5 Prediction: 19.0

Random State= 4

Accuracy: 64.27431303531395 %

Mean Squared Error: 31.275294117647068

Root Mean Square Error: 5.592431860796077

Actual: 20.8 Prediction: 19.6

Actual: 15.6 Prediction: 23.2

Actual: 36.2 Prediction: 33.0

Actual: 6.3 Prediction: 8.5

Actual: 23.7 Prediction: 24.4

Random State= 5

Accuracy: 56.07919130152332 %

Mean Squared Error: 39.22382352941177

Root Mean Square Error: 6.262892584853406

Actual: 13.4 Prediction: 11.7

Actual: 24.7 Prediction: 22.1

Actual: 21.1 Prediction: 20.6

Actual: 14.5 Prediction: 11.7

Actual: 16.6 Prediction: 17.4

Random State= 6

Accuracy: 81.47181312862828 %

Mean Squared Error: 17.465392156862745

Root Mean Square Error: 4.179161657182305

Actual: 24.6 Prediction: 23.0

Actual: 22.8 Prediction: 31.5

Actual: 32.5 Prediction: 36.2

Actual: 13.4 Prediction: 9.5

Actual: 50.0 Prediction: 23.1

Random State= 7

Accuracy: 70.99614759860098 %

Mean Squared Error: 15.233823529411763

Root Mean Square Error: 3.9030531035859304

Actual: 29.4 Prediction: 27.9

Actual: 22.7 Prediction: 23.0

Actual: 23.0 Prediction: 23.1

Actual: 24.4 Prediction: 20.6

Actual: 21.2 Prediction: 20.0

Random State= 8

Accuracy: 75.98802603939282 %

Mean Squared Error: 24.233039215686276

Root Mean Square Error: 4.922706492945347

Actual: 8.3 Prediction: 9.5

Actual: 19.6 Prediction: 19.9

Actual: 42.3 Prediction: 50.0

Actual: 28.5 Prediction: 30.8

Actual: 15.2 Prediction: 17.8

Random State= 9

Accuracy: 84.5866050170488 %

Mean Squared Error: 13.560784313725488

Root Mean Square Error: 3.682497021550118

Actual: 32.9 Prediction: 34.9

Actual: 18.9 Prediction: 21.0

Actual: 23.8 Prediction: 23.1

Actual: 19.9 Prediction: 21.8

Actual: 25.2 Prediction: 24.4

Figure 10:Accuracy in each random state

When we analyze the results, we can see that random sate 9 produces the best accuracy. Compared to the base accuracy of 75.0% this is a 9% improvement.

### Changing the number of recodes

In this section, we will use the method we used in multivariate regression to reduce the number of recodes and how it affects the accuracy.

Number of training recodes= 404

Accuracy: 75.91407750628234 %

Mean Squared Error: 19.880000000000003

Root Mean Square Error: 4.458699361921591

Number of training recodes= 303

Accuracy: 75.62868016495759 %

Mean Squared Error: 22.69596059113301

Root Mean Square Error: 4.764027769769296

Number of training recodes= 202

Accuracy: 69.33334467635808 %

Mean Squared Error: 26.059769736842107

Root Mean Square Error: 5.1048770540378445

Number of training recodes= 101

Accuracy: 56.7140701907938 %

Mean Squared Error: 38.211012345679016

Root Mean Square Error: 6.181505669792677

Number of training recodes= 50

Accuracy: 49.16766057406796 %

Mean Squared Error: 44.716184210526315

Root Mean Square Error: 6.687016091690397

Number of training recodes= 25

Accuracy: 20.430498028661003 %

Mean Squared Error: 68.50133056133056

Root Mean Square Error: 8.276553060382719

Figure 11:Accuracy compared to the number of recodes

Unlike the multivariable model, the accuracy of the tree model sees a massive reduction when the number of recodes decreases. With this, we can conclude that decision tree regression models heavily rely on the number of training records.

# Neural Network - Sequential

In this section, we will discuss how we can develop a Sequential neural network model to predict Boston house prices.

## Model brief

A neural network is an algorithm that was inspired by the structure of our brain's neural network. Just like an ML algorithm, neural network algorithm uses the data we provide to train the model and provide predictions. This learning process can be supervised or unsupervised. For this task, we are going to use a supervised learning model. This means we are going to use labelled data to train our model. (Géron, 2020)

Few elements determine the performance of a neural network. They are:

* Model type
* Layers
* Activation function
* Optimizer

We will discuss how I used these attributes to create the neural network model.

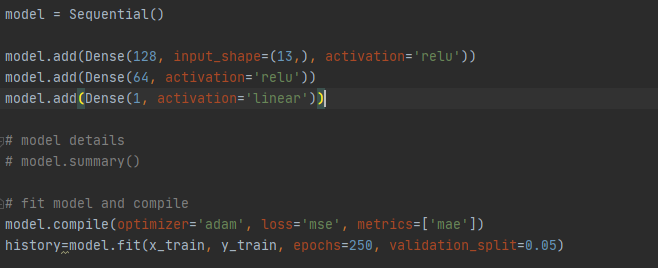


Figure 12:Neural network model

This is a Sequential model. This means this model has layers that stacked beside each other. This model has 3 hidden layers, one output layer, and one output layer. (Figure 13:Model graph)

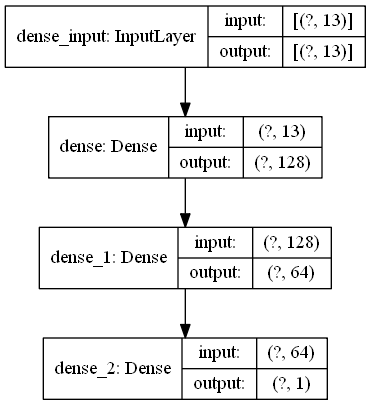


Figure 13:Model graph (inputs and outputs to each layer)

All the layers are dense layers and Each of the hidden layers has 128, 64, and 1 neurons. Dense layers are simple layers that connect input layers and output layers.

We use linear and relu activation functions for this model. Activate function determines the output of any neuron for any given input. These are based on our brain activity and how the neurons behave. “relu” which is short for “rectified linear unit” transforms the input to a maximum of 0 or input itself. This means if a neuron is positive more activated it is. The final activation function is linear this will act like any common liner function. (Agarap, 2019)

Furthermore, the optimizer determines how the model learns. For this model, we use Adam optimizer. (Diederik P. Kingma, 2015)

## Predictions and accuracy

In this section, we can see some of the predictions that the model made. Furthermore, graph 10 shows how the predictions are different from the actual predictions.

Actual: 14.4 Prediction: [13.53715]

Actual: 19.5 Prediction: [18.686243]

Actual: 22.6 Prediction: [22.146353]

Actual: 27.9 Prediction: [25.869482]

Actual: 18.3 Prediction: [18.261211]

Actual: 7.0 Prediction: [5.347152]

Actual: 24.5 Prediction: [26.404129]

Actual: 8.8 Prediction: [11.228392]

Actual: 27.9 Prediction: [31.363394]

Actual: 14.5 Prediction: [16.484726]

Actual: 24.4 Prediction: [21.89481]

Actual: 17.4 Prediction: [18.538921]

Actual: 20.7 Prediction: [23.464483]

Chart, scatter chart

Description automatically generated

Figure 14:NN model predictions

Altho there are many loss functions For this model we are going to use MSE know as mean squared error.

We can calculate MSE for input by getting the difference between the predicted value and the actual value. Then by adding all the squared value of all the inputs and dividing it by the number of inputs (Chollet, 2018).

In this chart, we can see the model loss changing according to the number of epochs.

A picture containing graphical user interface

Description automatically generated

Figure 15: Model Loss

For the matrix we are using MAE mean absolute error. Mean absolute error is calculated by adding all the difference between the predicted and actual value and getting the average.

A picture containing chart

Description automatically generated

Figure 16: Model MAE

This chart shows the MAE changes with the number of epochs.

## Improvements and suggestions

In this section, we will discuss how can we improve the accuracy of the model.

### Changing the number of epochs

In this section, we will discuss how increasing the number of epochs going to influence the accuracy of our neural network.

First, let us look at what the word epoch means. One Epoch is when a dataset is passed through the neural network entirely. To optimize the learning process the dataset must pass through the neural network more than once (Chollet, 2018).

When we consider figure13 and figure14 we can see that both MSE and MAE decreasing after each epoch. But when the epoch count reaches a certain threshold, we cannot see further decrees of MSE or MAE. That number is 100 for MSE and 150 MAE. When considering this data, we can conclude that the optimal number of epochs is between 100-150.

### Changing the number of neurons

In this section, we are going to discuss the effects of number layers and the number of neurons has in the model accuracy.

By looking at the figure 13 we can see that the model has 2 hidden layers, one input layer and one output layer. Also, these layers have neurons ranging from 1 -128. Neurons in the input layer and output layer has to be a set number. In our model 13 neurons for the input layer(to match the input array shape) and one neuron for the output layer. First, we will change the number of neurons in the 2nd and 3rd layer to see how the results change. For each test we are going to use 100 epochs.

If we are to half the neurons on each layer, we get the following results.

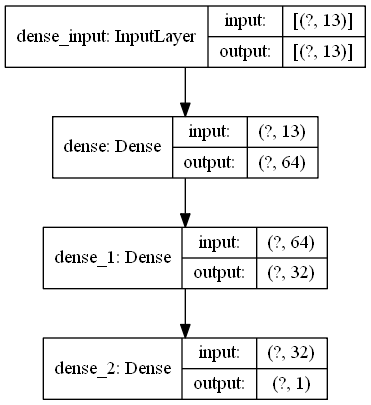


Figure 17: number of neurons halved for each hidden layer.

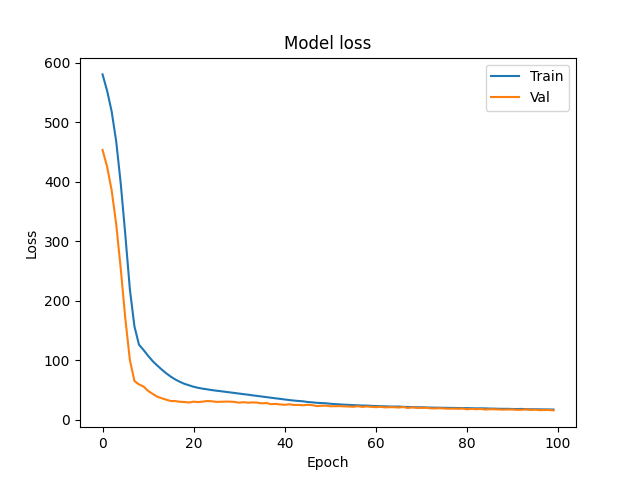


Figure 18:MSE when neurons are halved

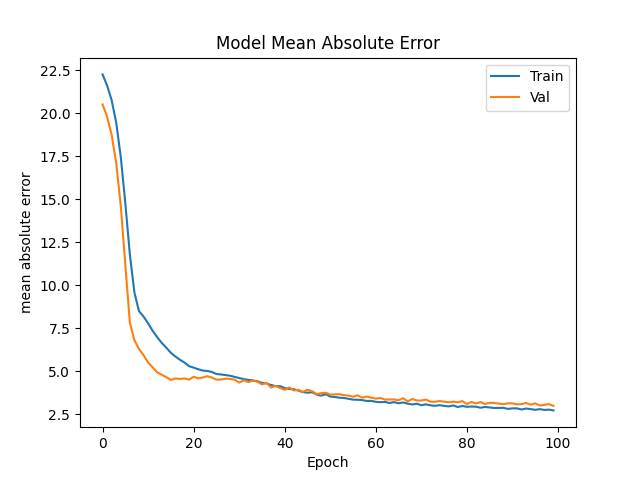


Figure 19:MAE when neurons are halved

Next, we will see the results when the number of neurons is doubled.

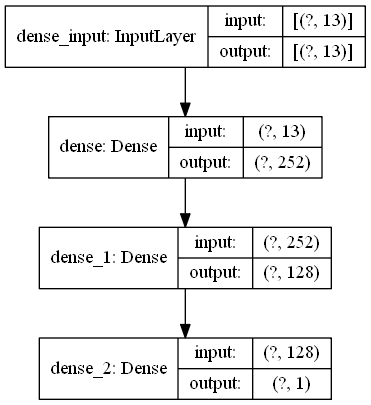


Figure 20:Number of neurons Doubled

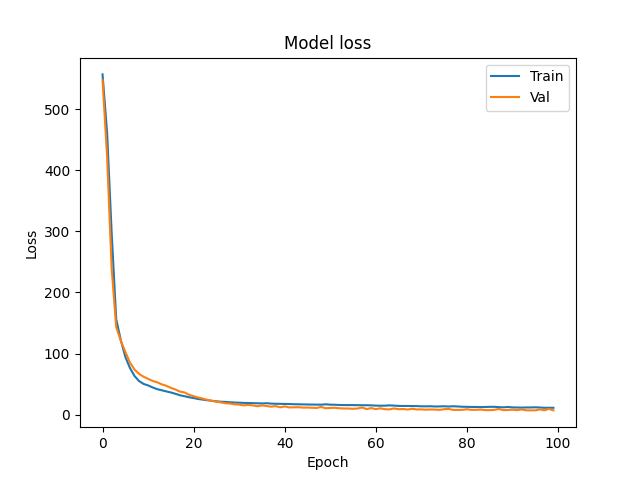


Figure 21:MSE when neurons Doubled

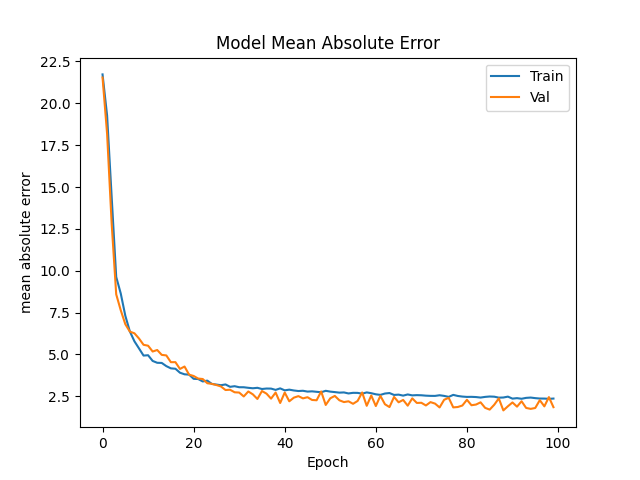


Figure 22:MAE when neurons Doubled

When We study these data, we can see that the loss of the model did not improve significantly even we doubled the number of neurons compared to the base model.

But we can see increase in MSE when we halve the number of neurons in a layer. This means a loss in accuracy.

# References

Agarap, A. F. M., 2019. *Deep Learning using Rectified Linear Units (ReLU),* s.l.: Cornell University.

AnthonyJ.Myles, R., 2004. An introduction to decision tree modeling. *JOURNAL OF CHEMOMETRICS,* 18(275–285), p. 11.

Chollet, F., 2018. *Deep Learning with python.* 1st ed. Shelter Island: Manning Publications.

Diederik P. Kingma, J. L. B., 2015. *ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION,* s.l.: a conference paper at ICLR.

Geron, A., 2019. *Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow.* 2nd ed. Sebastopol: O'Reily Media, Inc..

Géron, A., 2020. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow : Concepts, Tools, and Techniques to Build Intelligent Systems.* Second ed. Sebastopol: O’Reilly Media.

H. Zhao, O. G. I. F. a. J. K., 2017. Loss Functions for Image Restoration With Neural Networks. *IEEE Transactions on Computational Imaging,* 3(10.1109/TCI.2016.2644865), pp. 47-57.

Jurman, D. C. &. G., 2020. *Machine learning can predict survival of patients with heart failure from serum creatinine and ejection fraction alone.* [Online]   
Available at: https://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/s12911-020-1023-5  
[Accessed 22 10 2020].

Russell, S. P. N., 2016. *Artificial Intelligence : a Modern Approach.* Third ed. Harlow, Essex, England: Pearson.

scikit learn, 2020. *Preprocessing data.* [Online]   
Available at: https://scikit-learn.org/stable/modules/preprocessing.html  
[Accessed 1 10 2020].