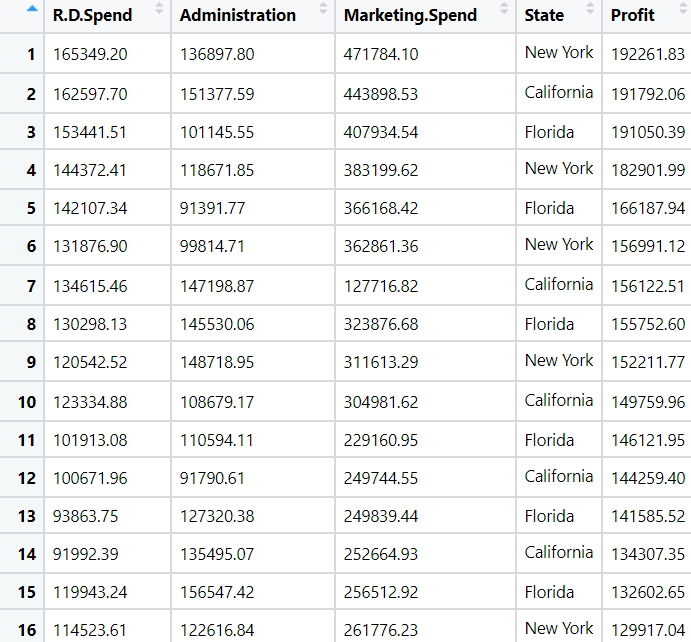
# Topic: Neural Network

1.) Build a Neural Network model for 50\_startups data to predict profit



**Ans:**

**Analyzing the input and output variables:**

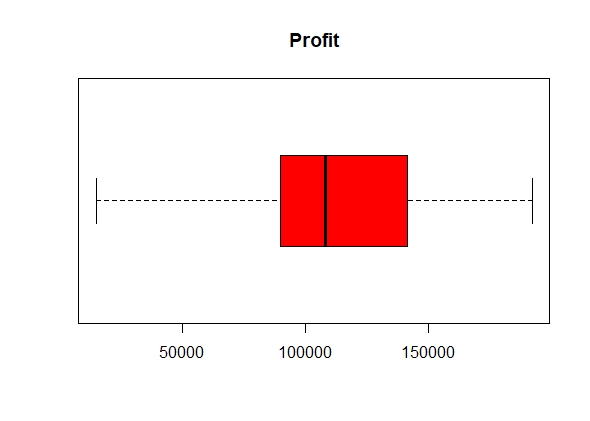
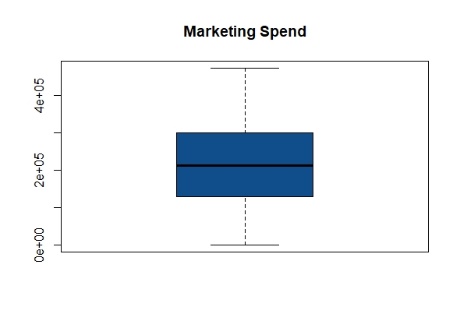
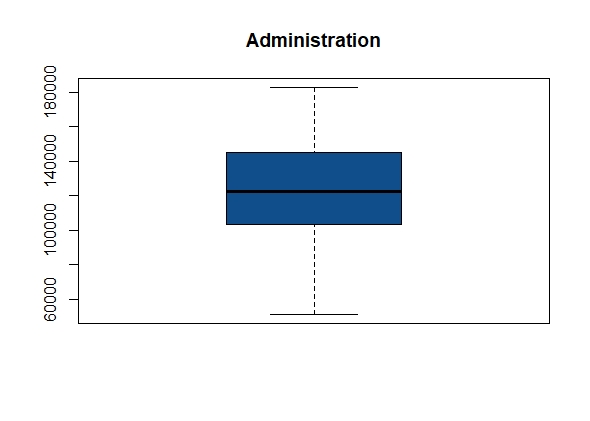
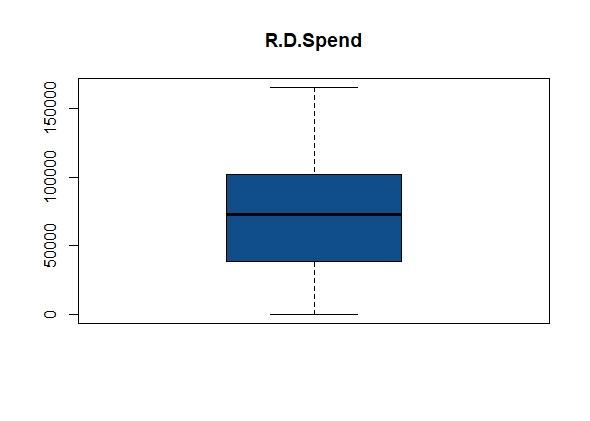
* Input Variables (x) = R.D Spend, Administration, Marketing Spend, State
* Output Variable(y) = Profit

**Data Preprocessing:**

* To make the easy access of variables of input and output, columns are rearranged.
* In R , dummy variable are created automatically when object(x) is created as model. Matrix command has inbuild feature of converting dummy variables whereas, in python we need to write a code for creation of dummy variables.

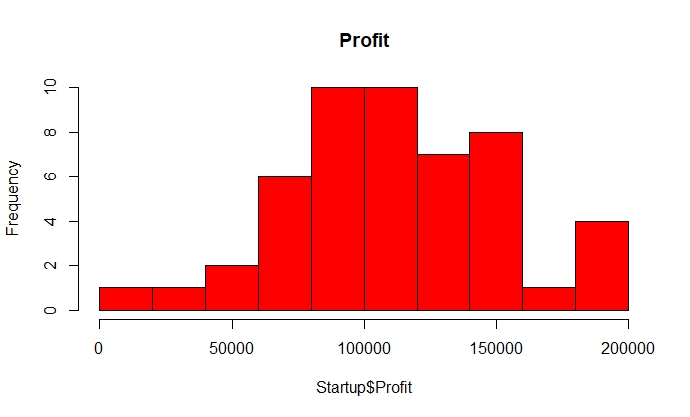
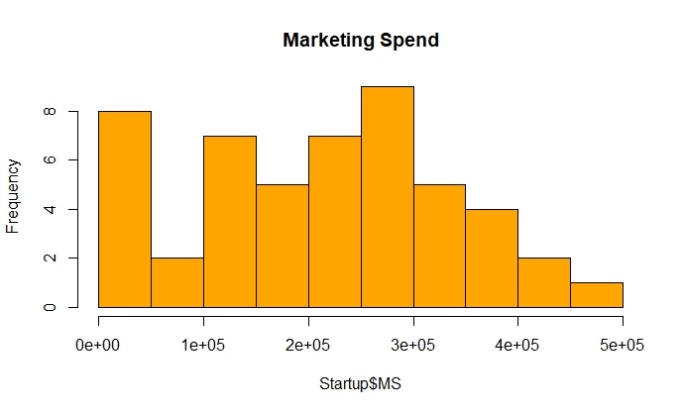
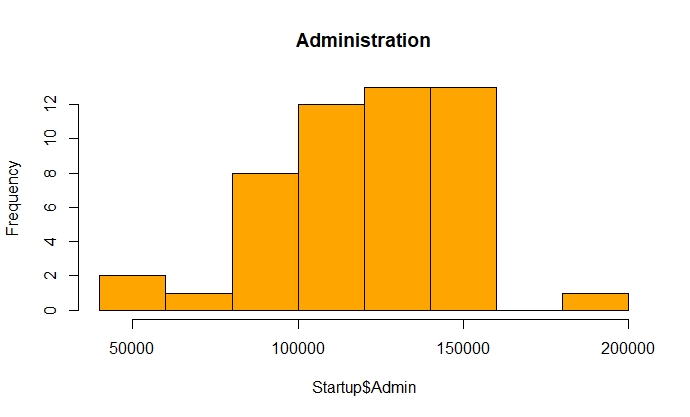
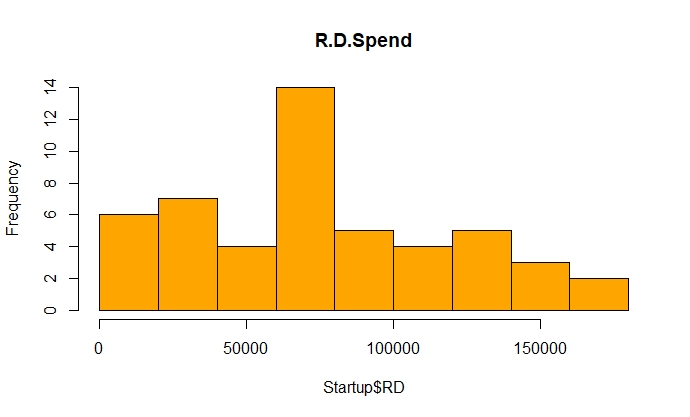
**Exploratory Data Analysis:**

**Boxplot Representation:**

****

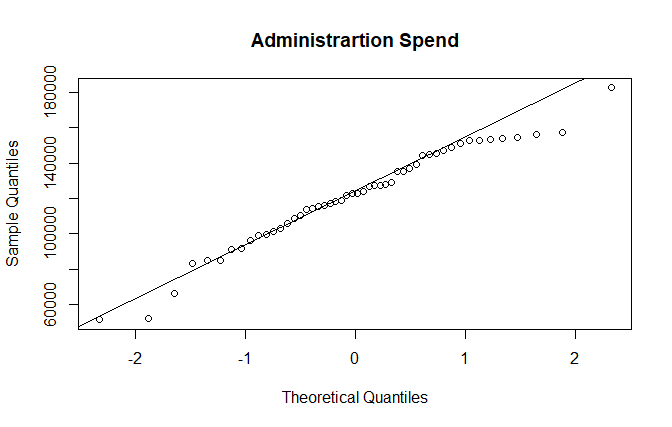
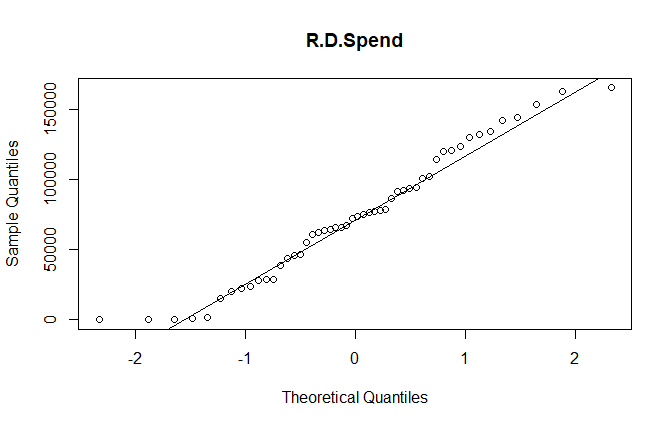
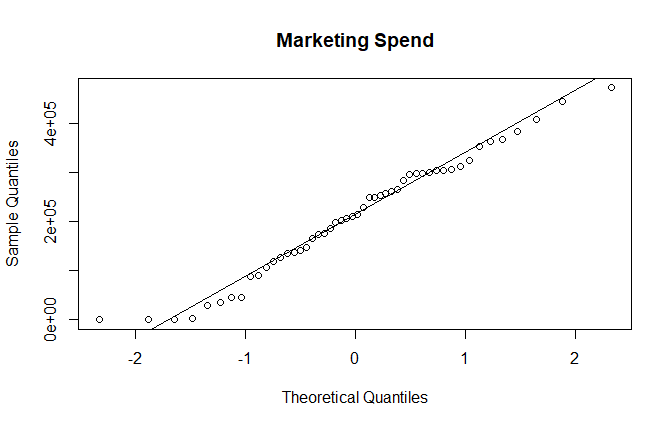
* From the above box plots it shows that there are no outliers exist for individual variables

**Histogram Representation:**

****

* The graphs indicate that the data is normally distributed for the individual variables

**Plotting:**



* From the above given plots, we can that the data is normally distributed and there are some outliers exist

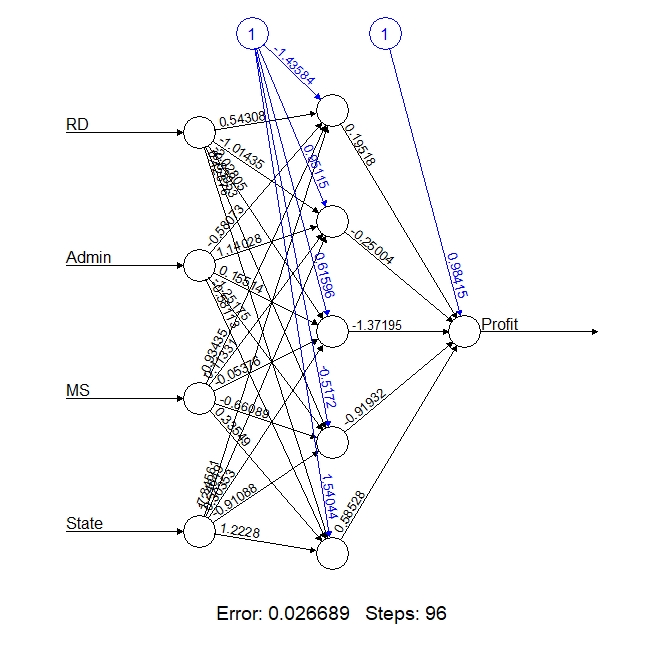
**Normalizing the data:**

* Using the normalization function, the data is normalized and the data will come under same level.

**Model Building:**

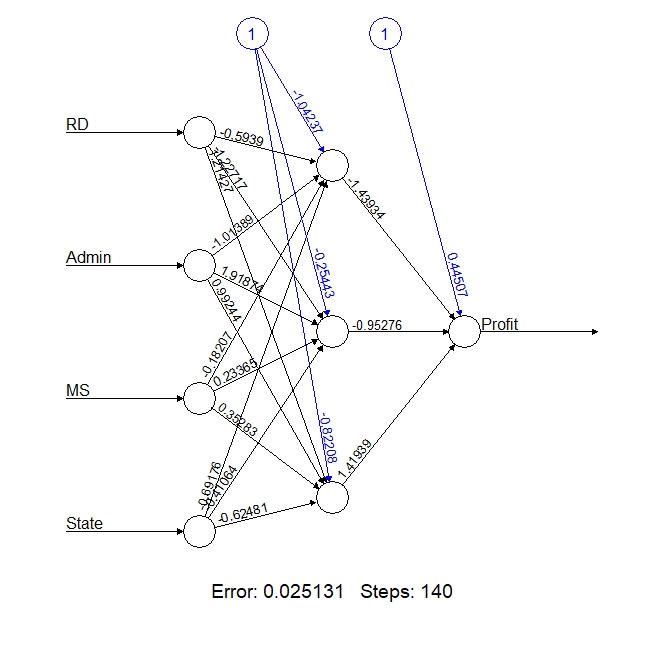
* The data is now divided into training and test to build the Neural network model.

**startup\_Model <- neuralnet(formula = Profit ~ ., data = train)**



**Model 1:**

* After tweaking the with hyperparameter with hidden = 5 the plot is as below with increased accuracy



**PYTHON ANALYSIS:**

* As we see this as a Regression Problem we need to import Kera Regressor and TensorFlow to implement this Neural Network
* Import all the packages related
* Here the cost function is rmse as it is a regression problem
* Now the load data set Startup
* Convert the State Input into Categorical
* Normalize the whole data
* Split the Dataset as input and output variables x and y
* Now by using Minmax scalar transform x and y variables to build the model
* Then Start building the model and then fit with passing input dimension argument for inputs and number of Hidden Layers to Dense

Model: "sequential\_1"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

dense\_3 (Dense) (None, 12) 84

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_4 (Dense) (None, 12) 156

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_5 (Dense) (None, 8) 104

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_6 (Dense) (None, 1) 9

=================================================================

Total params: 353

Trainable params: 353

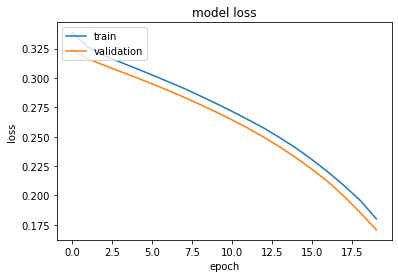
Non-trainable params: 0

After 20 Epochs with batch size 10

mae: 0.3273 - val\_loss: 0.1194 - val\_mse: 0.1194 - val\_mae: 0.2924

Epoch 20/20

**Train and Validation loss plot**



**Conclusion:**

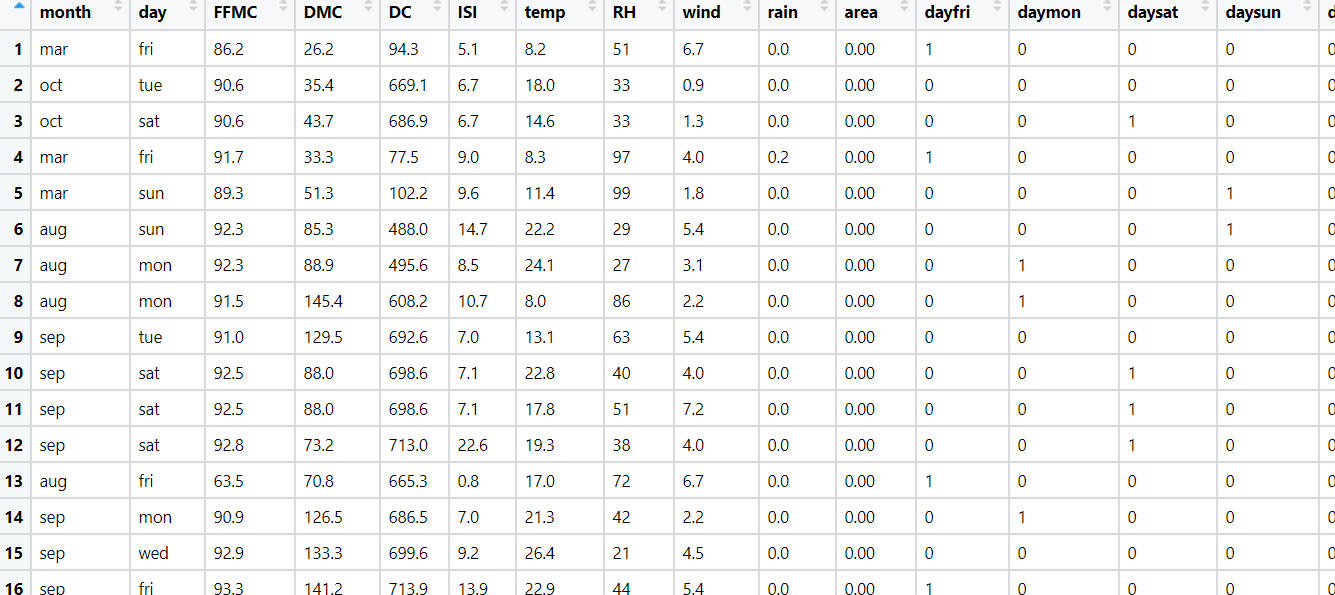
Model built with 1 hidden neuron give the error as 0.028742

Model built with 5 hidden neurons give the error as 0.026689

Model built with 3 hidden neurons give the error as 0.025131

By comparing all the 3 above models the one with 3 hidden neurons gives less error values which is a right fit model

2.) Predict the burned area of forest fires with Neural Networks



**Ans:**

**Analyzing the input and output variables:**

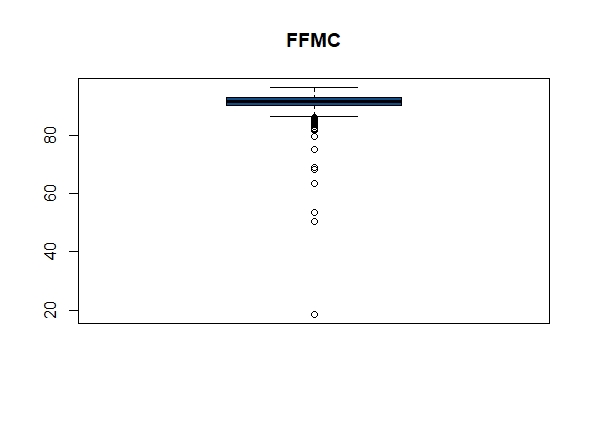
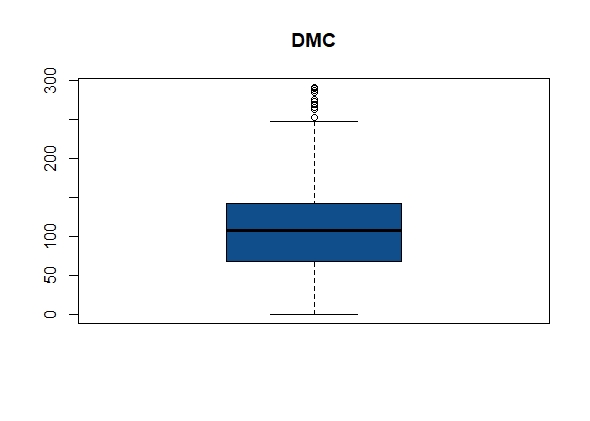
* Input Variables (x) = Other Factors influencing the output variable
* Output Variable(y) = area

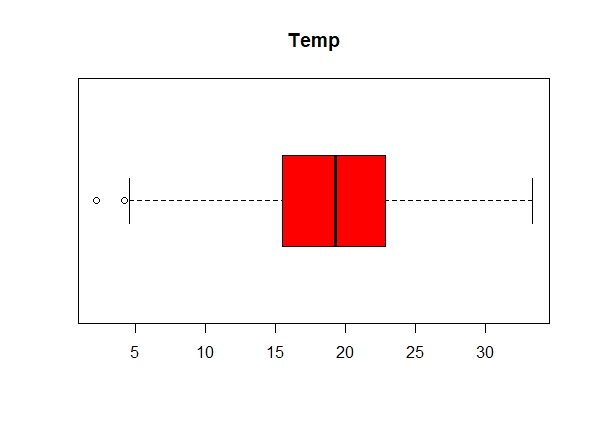
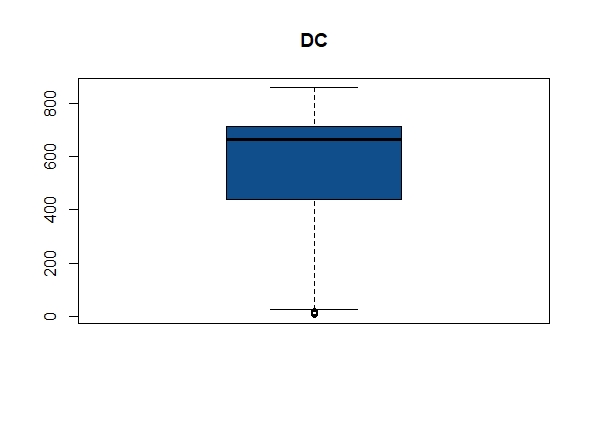
**Data Preprocessing:**

* To make the easy access of variables of input and output, columns are rearranged.
* In R , dummy variable are created automatically when object(x) is created as model. Matrix command has inbuild feature of converting dummy variables or we can factorize the variables whereas, in python we need to write labelencoder () code for creation of dummy variables.
* Removing unnecessary columns and checking for the NA values

**Exploratory Data Analysis:**

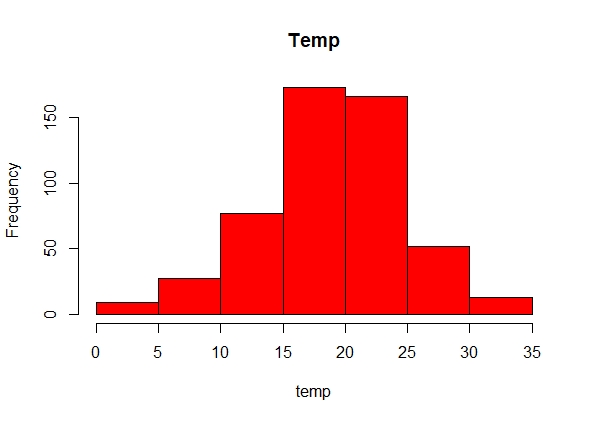
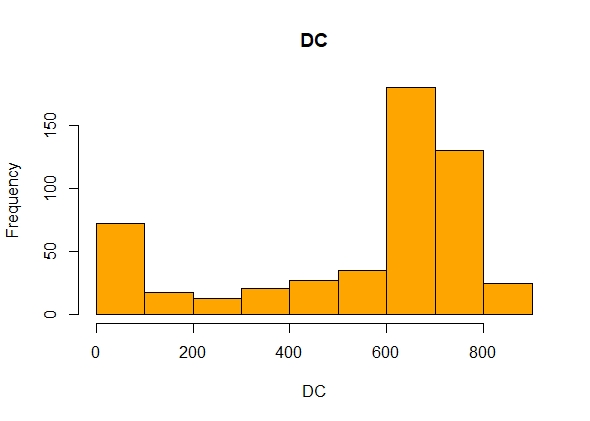
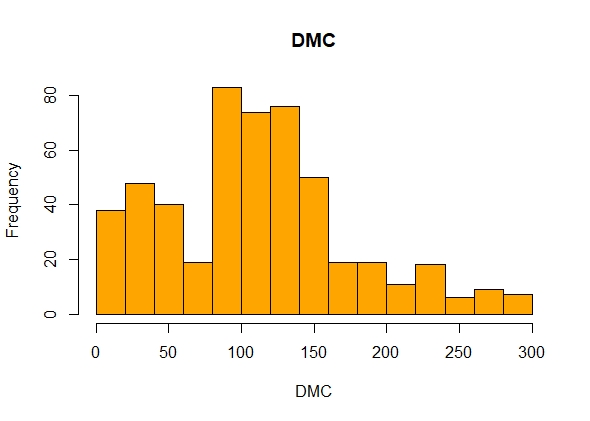
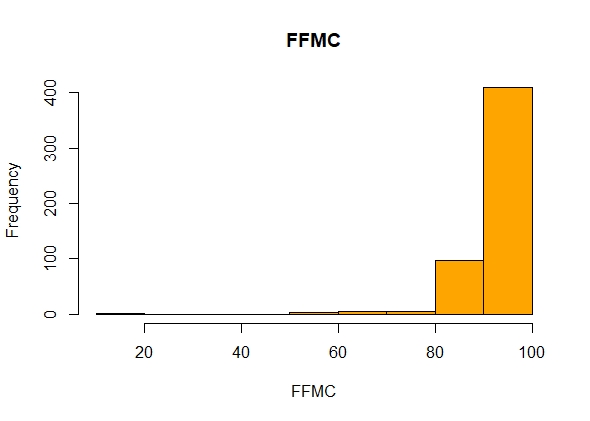
**Boxplot Representation:**

****

****

* The above boxplot represents there exist outlier in the individual variables

**Histogram Representation:**

****

* From the above graphs data is normally distributed in case of temp and DMC but incase of FFMG and DC the data is right skewed

**Normalizing the data:**

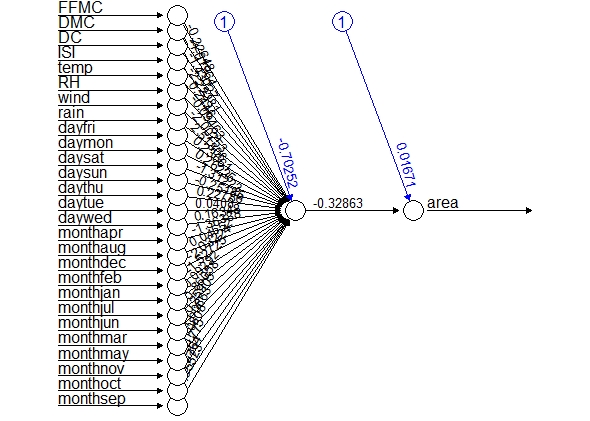
* Using the normalization function, the data is normalized and the data will come under same level.

**Model Building:**

* The data is now divided into training and test to build the Neural network model.

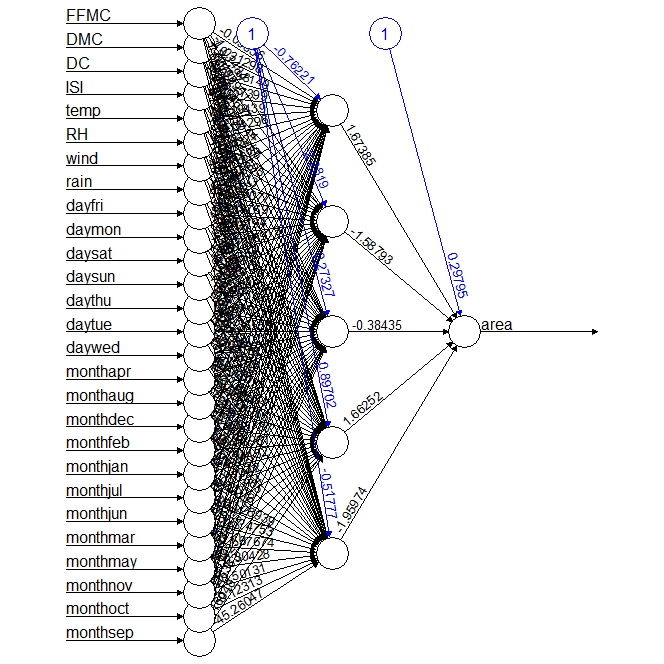
**Model1:**

**FF\_Model <- neuralnet(formula = area ~ ., data = train)**

****

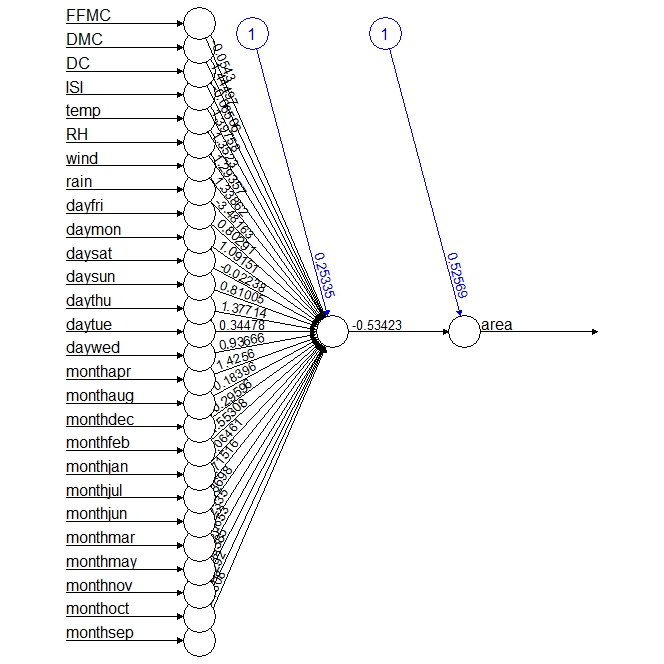
**Model 2:**

* After tweaking the with hyperparameter with hidden = 5 the plot is as below with increased accuracy

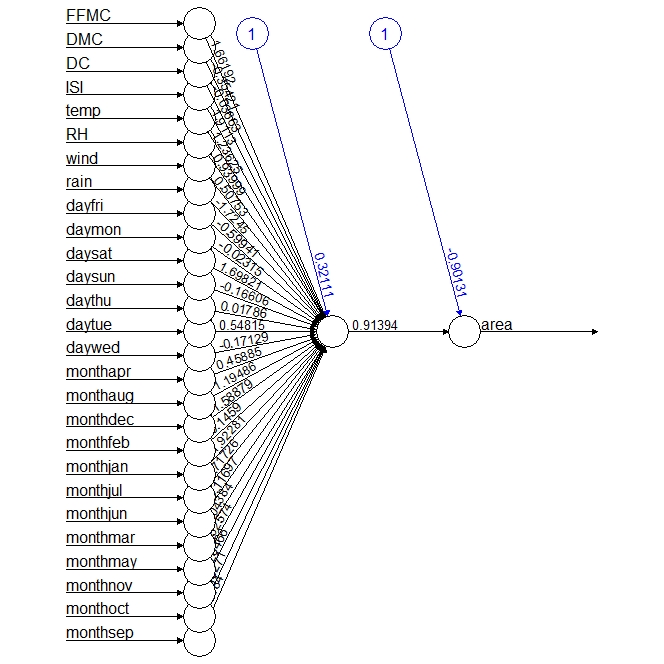
****

**Model 3 and Model 4: Build Using stepmax function**

* After tweaking the with hyperparameter with stepmax the plot is as below with increased correlation



**Forest\_Model4 <- neuralnet(formula = area ~.,data = Forest\_train, stepmax = 1e+05, algorithm = "rprop+", act.fct = "tanh")**



**PYTHON ANALYSIS:**

* As we see this as a Regression Problem, we need to import Keras Regressor and TensorFlow to implement this Neural Network
* Import all the packages related
* Here the cost function is rmse as it is a regression problem
* Now load data set Fireforests
* Normalize the whole data
* Split the Dataset as input and output variables x and y
* Now by using Minmax scalar transform x and y variables to build the model
* Then Start building the model and then fit with passing input dimension argument for inputs and number of Hidden Layers to Dense

Model: "sequential\_1"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

dense\_3 (Dense) (None, 12) 324

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_4 (Dense) (None, 8) 104

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_5 (Dense) (None, 1) 9

=================================================================

Total params: 437

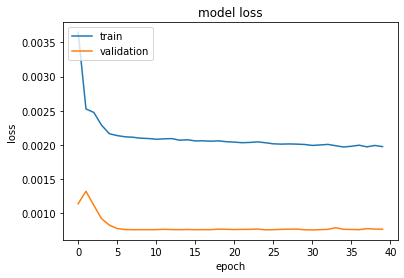
Trainable params: 437

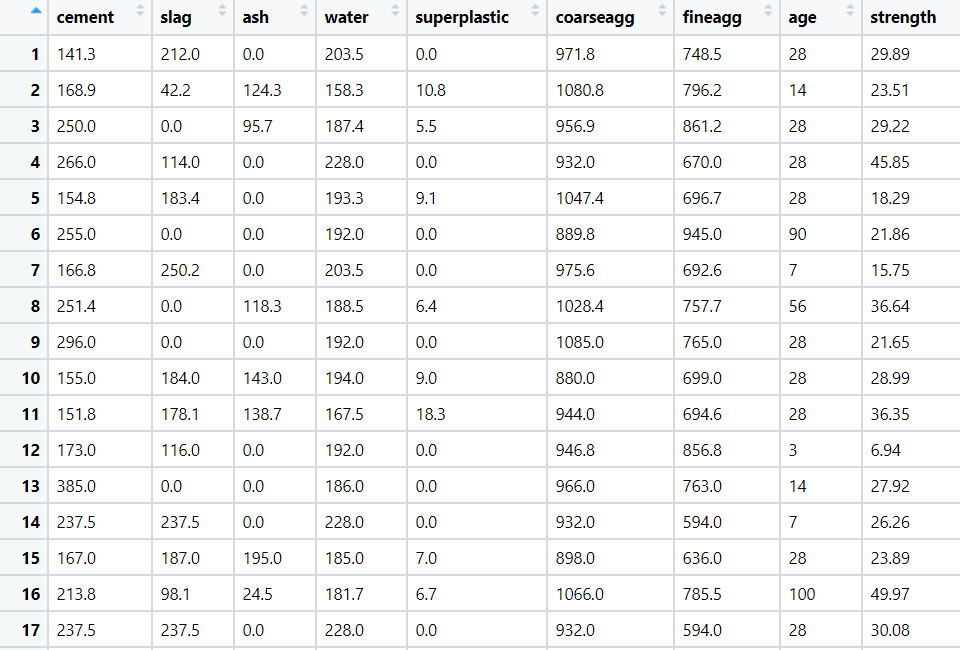
Non-trainable params: 0

After 40 Epochs with batch size 50

Epoch 40/40

7/7 [==============================] - 0s 6ms/step - loss: 0.0047 - mse: 0.0047 - mae: 0.0191 - val\_loss: 8.2901e-04 - val\_mse: 8.2901e-04 - val\_mae: 0.0165



3.) Prepare a model for strength of concrete data using Neural Networks

**Ans:**

**Analyzing the input and output variables:**

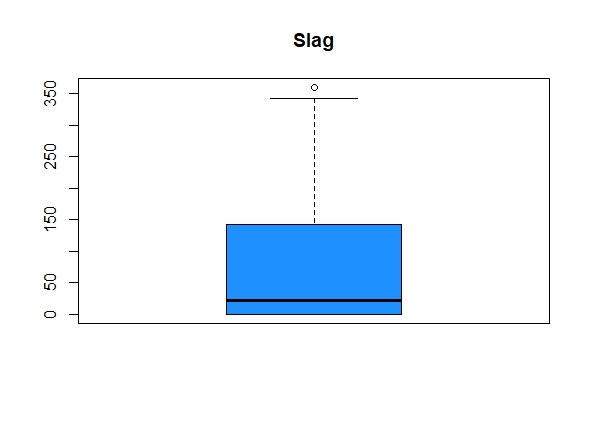
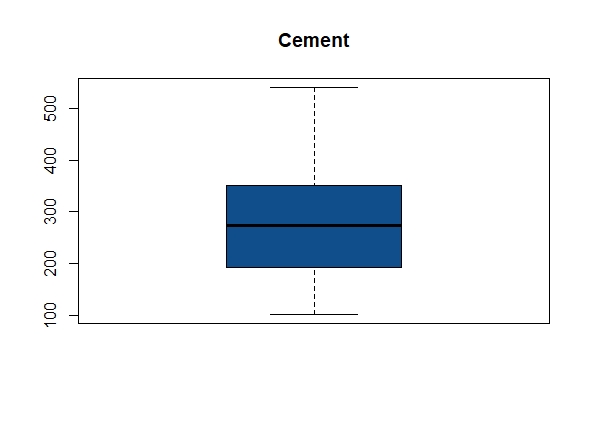
* Input Variables (x) = Other Factors influencing the output variable
* Output Variable(y) = area

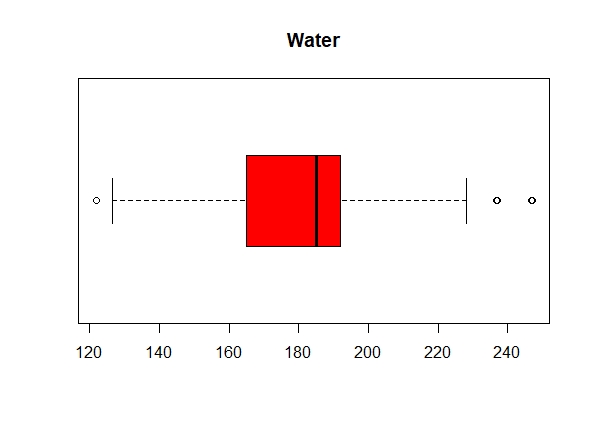
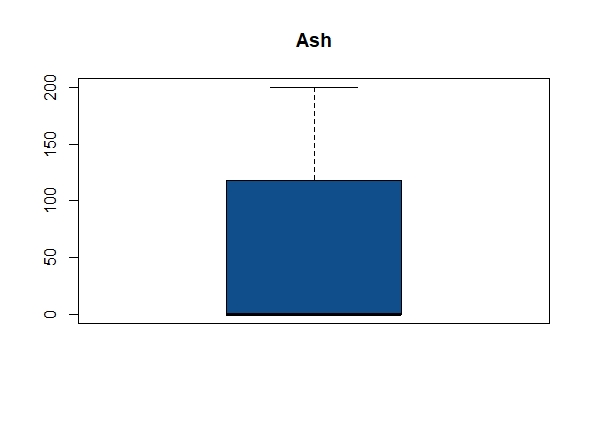
**Data Preprocessing:**

* To make the easy access of variables of input and output, columns are rearranged.
* In R , dummy variable are created automatically when object(x) is created as model. Matrix command has inbuild feature of converting dummy variables or we can factorize the variables whereas, in python we need to write labelencoder () code for creation of dummy variables.
* Removing unnecessary columns and checking for the NA values

**Exploratory Data Analysis:**

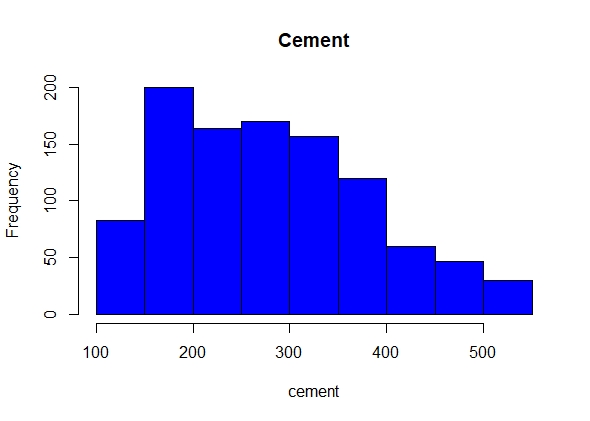
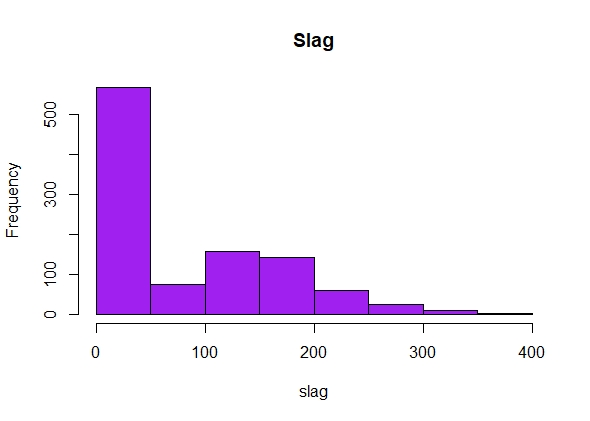
**Boxplot Representation:**

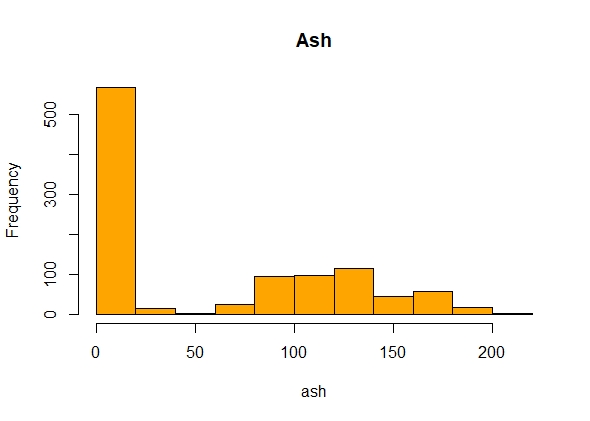
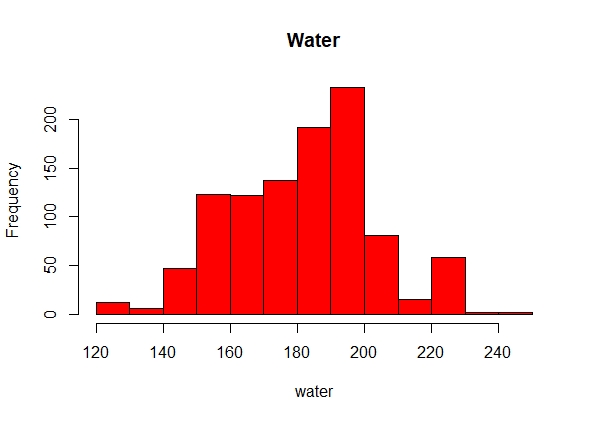
****

****

* From the above box plots it shows that there are outliers exist for Slag & Water variables.

**Histogram Representation:**

****

****

* From the above graphs data is normally distributed in case of Water and Cement but in case of Ash and Slag the data is left skewed

**Normalizing the data:**

* Using the normalization function, the data is normalized and the data will come under same level.

**Model Building:**

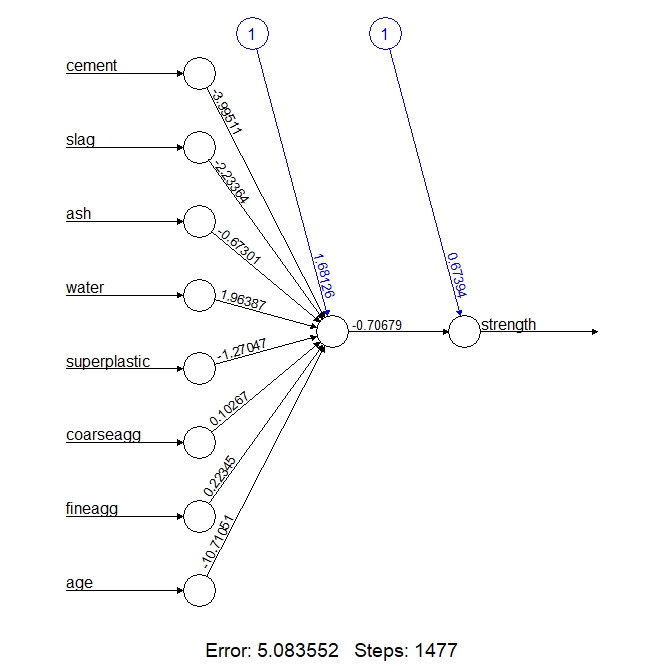
* The data is now divided into training and test to build the Neural network model.
* Predicting the strength of concrete using Neural Network with below Equation

concrete\_model <- neuralnet(formula = strength ~ cement + slag +

ash + water + superplastic +

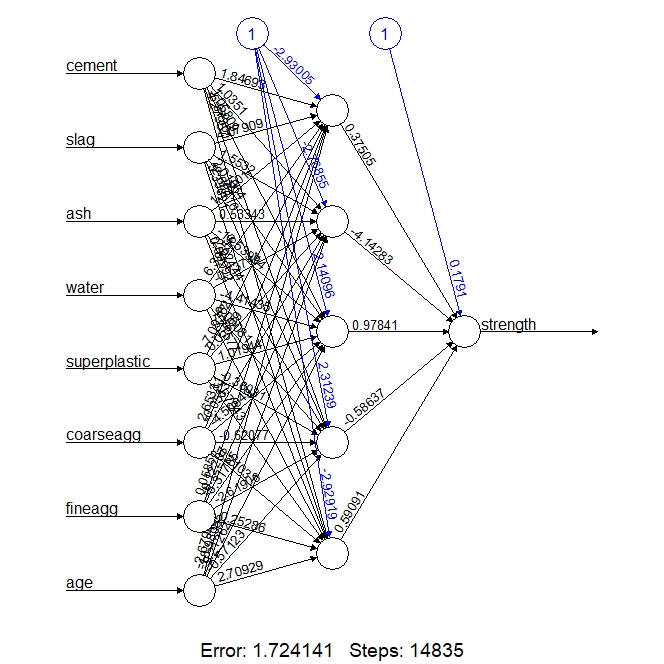
coarseagg + fineagg + age,

data = concrete\_train)



**MODEL CORRELATION – 0.806**

**After tweaking the hyperparameters the correlation is increased to 0.93 and the plot is below.**



**Conclusion:**

Model built with 1 hidden neuron give the error as 5.083552

Model built with 5 hidden neurons give the error as 1.724141

By comparing all the 2 above models the one with 5 hidden neurons gives less error values which is a right fit model

**PYTHON ANALYSIS:**

* As we see this as a Regression Problem, we need to import Keras Regressor and TensorFlow to implement this Neural Network
* Import all the packages related
* Here the cost function is rmse as it is a regression problem
* Now load data set Concrete
* Normalize the whole data
* Split the Dataset as input and output variables x and y
* Now by using Minmax scalar transform x and y variables to build the model
* Then Start building the model and then fit with passing input dimension argument for inputs and number of Hidden Layers to Dense

Model: "sequential\_2"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

dense\_6 (Dense) (None, 12) 108

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_7 (Dense) (None, 8) 104

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_8 (Dense) (None, 1) 9

=================================================================

Total params: 221

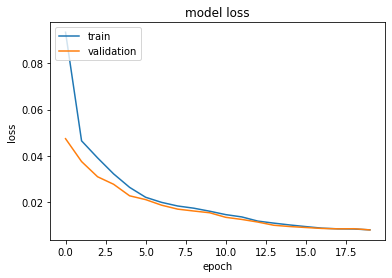
Trainable params: 221

Non-trainable params: 0

After 20 epochs and batch size = 10

Epoch 20/20

62/62 [==============================] - 0s 2ms/step - loss: 0.0084 - mse: 0.0084 - mae: 0.0723 - val\_loss: 0.0073 - val\_mse: 0.0073 - val\_mae: 0.0654



**Hints:**

1. Business Problem
   1. Objective
   2. Constraints (if any)
2. Data Pre-processing

2.1 Data cleaning, Feature Engineering, EDA etc.

1. Model Building
   1. Partition the dataset
   2. Model(s) - Reasons to choose any algorithm
   3. Model(s) Improvement steps
   4. Model Evaluation
   5. Python and R codes
2. Deployment

4.1 Deploy solutions using R shiny and Python Flask.

1. Result Share the benefits/impact of the solution - how or in what way the business (client) gets benefit from the solution provided.

**Note:**

1. For each assignment the solution should be submitted in the format
2. Research and Perform all possible steps for improving the model(s) accuracy Ex: Feature Engineering, Hyper Parameter tuning etc.
3. All the codes (executable programs) are running without errors
4. Documentation of the module should be submitted along with R & Python codes, elaborating on every step mentioned here