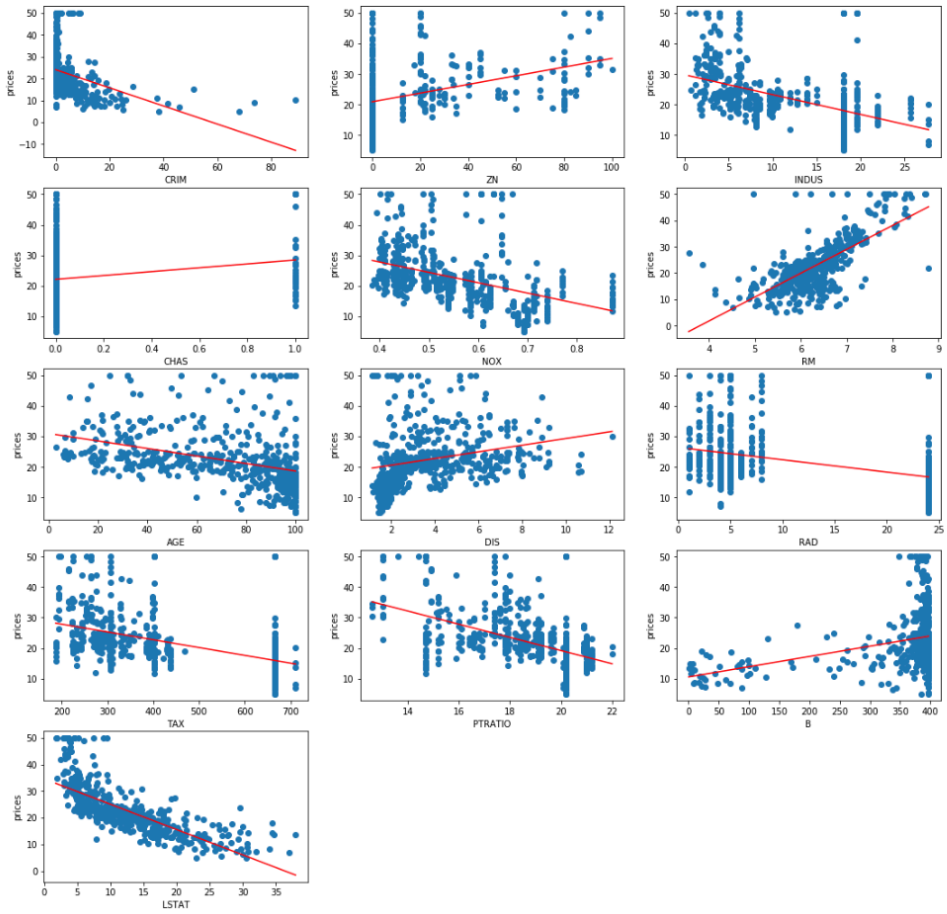
Tittle: Multi-Layer Perception (MLP) FeedForward Neural Network for House Price Prediction

Goal: To Build House Price Predictive Model using MLP.

1.Procedure: In the section, all the steps and procedures that were carried during this project is presented. As per the task, first thing to do was to select one of the deep learning architecture. Thus, I have selected multi-layer perception Feedforward Neural Network since this network is suitable to the dataset (tabular form) like Boston house dataset and even for our problem i.e. regression. Apart from this, it is easy to learn and implement for this problem.

1.1 Data Exploration and preprocessing: The pandas was used to load and explore the dataset. The dataset , Boston house dataset was used for the project. In the dataset, it was found that there were 506 rows and 14 columns including the target variable (MEDV). The NAN values were checked in the dataset and found nothing. The statistic measurement was applied to each variables, so it gives the overall view of each variable; how they are scattered from mean value, etc. For example, in the variable (CRIM), the minimum value is 0.00632 whereas maximum value is 89.977 and mean value is 3.6135. This show that there are outliers in the data. Its better to fix or remove outliers from the dataset. Nevertheless, at first I just train and the test the model without removing outliers and later I did remove outlier and train and test the model. And, for better understanding about how the data is spreaded and their relationship with target variable, the scatter plot was drawn between each variables with target variable(MEDV). The following the screen shoot:



From above figure, it can say that the features like RM, LSTAT, CRIM and PIRATIO have greater impact(i.e in negative or positive way) in prediction of house price.

The data was separated between the target variable and features. The normalisation was applied in the features since the features ranges not same; some feature have max value is 1 where as some has 396. It is better to normalise the features so that model will learn more efficiently. After then, the data was splitted for training and testing dataset with size of 20% for test data. Below screen shot illustrates about the train and test data.

```
'''==== Multi-Layer Perception architecture is used for prediction of house price ===='''

# shuffle and split data into train (~80%) and test (~20%)
X_train, X_test, y_train, y_test = train_test_split(normalized_feature, target.values,
                                                    test_size=0.2, random_state=42)
print('training data shape: ',X_train.shape)
print('testing data shape: ',X_test.shape)
```

```
training data shape:  (404, 13)
testing data shape:  (102, 13)
```

1.2 Build MLP network: The keras was used to build the network. The network has 5 hidden layers of 150 nodes/neurons. The first layer is the input layer of 13 units since we have 13 features those need to use for training and out put layer with single unit. The Below is the summary of the model.

```
#inspect the model
model.summary()
```

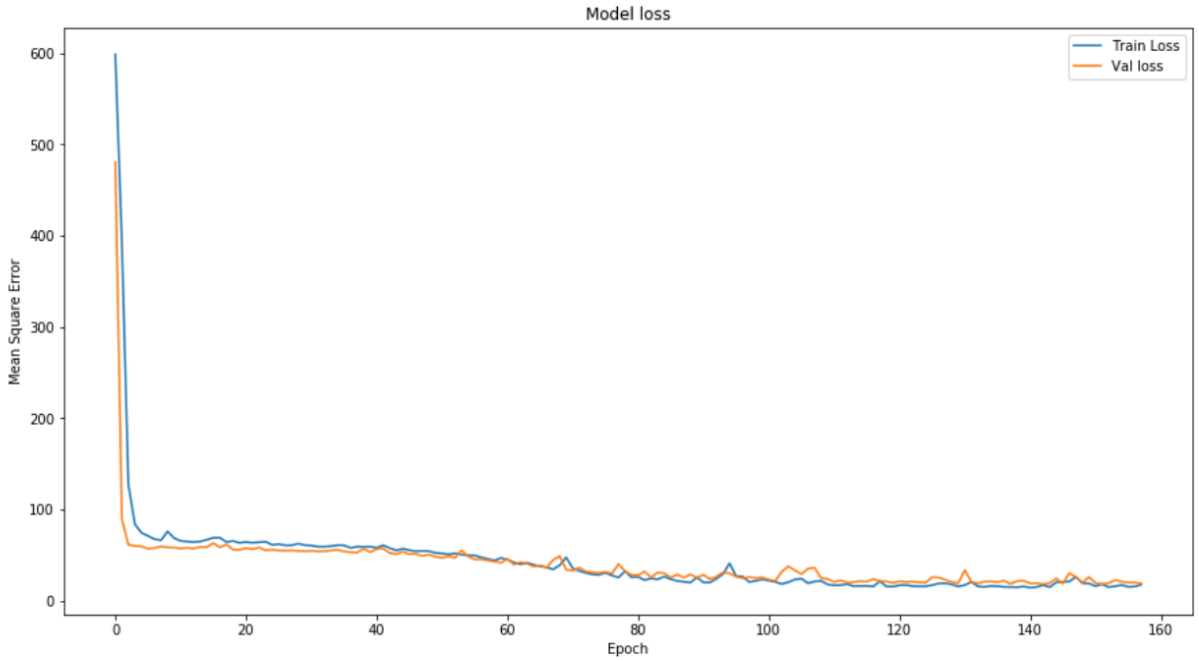
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_316 (Dense)            (None, 150)               2100
_________________________________________________________________
dense_317 (Dense)            (None, 150)               22650
_________________________________________________________________
dense_318 (Dense)            (None, 150)               22650
_________________________________________________________________
dense_319 (Dense)            (None, 150)               22650
_________________________________________________________________
dense_320 (Dense)            (None, 150)               22650
_________________________________________________________________
dense_321 (Dense)            (None, 1)                 151
=================================================================
Total params: 92,851
Trainable params: 92,851
Non-trainable params: 0
_________________________________________________________________
```

The model was complied with 'adam' optimiser and loss value with mean square root. And, Mean absolute error was used as metric.

```
#compile model
model.compile(loss='mse', optimizer='adam', metrics=['mae'])
```

The model was trained using the training data set with 300 epochs and the 'early stop' regularisation method was applied. So, that if the model gets no improvement during the training the 'early stop' stops model to go further repetition(epoch) during training. The validation was also used to validate the model. The minimum loss value found during the training was 16.240 and validation loss was around 19.

To look the training and validation loss during the training phase, I have drawn the model loss graph which is shown below:



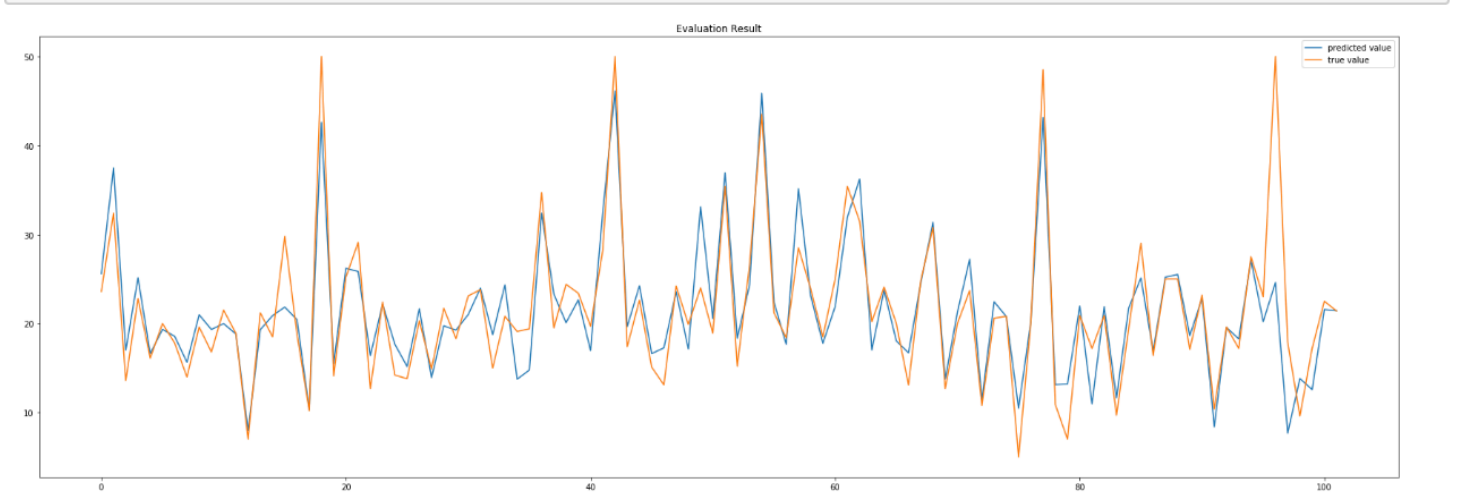
From the figure, it can say that the model drastically drop on both validation and training loss value as the epoch increases. After the certain point of epoch, the validation and training loss values remain almost the same. At the end of graph, it is seeing that validation loss value is more than training loss value. This is true because the validation done in the new set of data. This is reason why validation test is necessary in order to look the model ability/skill to learn the new data.

The model was then tested using the test dataset and it was found that Mean absolute error is 2.633737377 i.e around $2633 less or more in the prediction value. The prediction was performed with the test data set and got the prediction values as below. The table of predicted value and true value is shown in table:

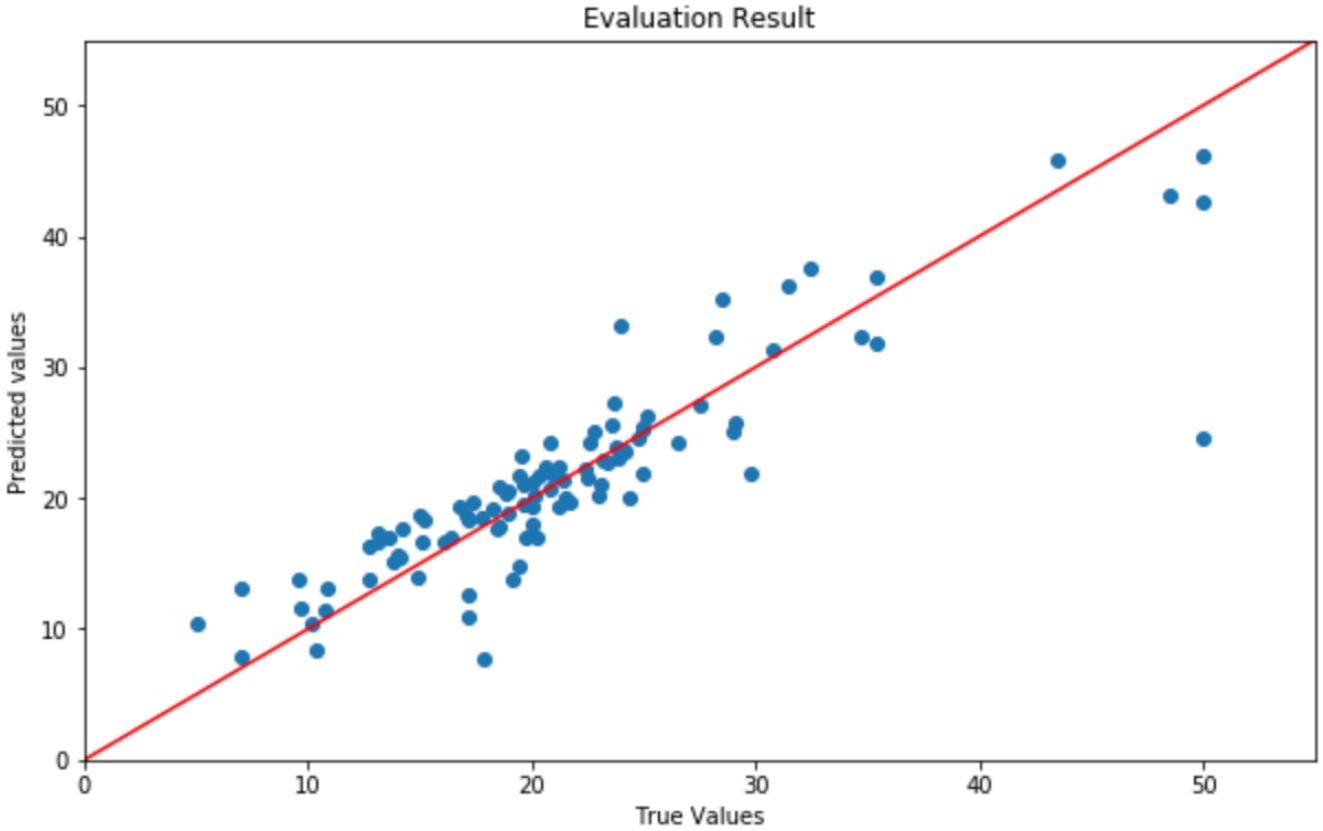
```
# show the true value and predicted value in dataframe
true_predicted = pd.DataFrame(list(zip(y_test, test_predictions)),
                              columns=['True Value','Predicted Value'])
true_predicted.head(10)
```

	True Value	Predicted Value
0	23.6	25.590422
1	32.4	37.486252
2	13.6	17.000551
3	22.8	25.128817
4	16.1	16.607439
5	20.0	19.340845
6	17.8	18.559471
7	14.0	15.650881
8	19.6	20.977253
9	16.8	19.326307

The finally part was to visualise the prediction value from model; how accurate does the model predict the test data. To evaluate the learning capability of the model, firstly I have drawn line graph between predicted and true values, but it doesn't give much meaning, so I have drawn another graph which you can find later. Below is the screen shot of line graph:



Another figure, the scatter plot was drawn between predicted values and true values using the diagonal line(45 degree) in the figure. So, if the predicted value is almost or exactly the same as true value then the data points fall in the 45 degree diagonal line. As for example, if true value(x-axis) is 30 and predicted value (y-axis) is 30 , then coordinate point will be (30, 30) and it must lie is the 45 degree diagonal line. Below is the screen shot:



From the figure, it can see that majority of the data points lie close to red line. Thus, most of test data is predicted very well. The data points far away from the red line means the difference between predicted value and true value is greater amount. The data points that are more close to the red line indicates less error in prediction. In other words, the model predict very well or close to the true values.

In the above figure one data point seems far away from red line, this might be due to outliers in the dataset. Thus, I tried to remove the outliers from the features variables and train and test the model, the following the result got after that. The model learning skill remains almost the same, no improvement after removing outlier also since this may be the less number of data. Below is the prediction of test data after removing outliers.