# Practicum 2 – Linear Regression Technical Report

## Abstract

The objective of practicum 2 is to design and implement a linear regression algorithm using gradient descent to predict the housing prices in Boston. Evaluation metrics chosen are root mean square error (RMSE) and R-squared ($R^2$).

The implementation of the linear regression algorithm was conducted for two experiment setups as follows.

- All 13 input features provided in the dataset, and
- 10 input features after feature selection.

The table below is a summary of the root mean squared error (RMSE) and r-squared ($R^2$) score of the experiment setups.

| Experiment Setup | Library / Function | Data | RMSE | $R^2$ |
|---|---|---|---|---|
| All 13 input features | User-defined gradient descent function | Train | 4.824 | 0.741 |
| | | Test | 3.285 | 0.430 |
| | Scikit learn official library / API | Train | 4.820 | 0.742 |
| | | Test | 3.287 | 0.429 |
| 10 input features after feature selection | User-defined gradient descent function | Train | 5.006 | 0.721 |
| | | Test | 3.229 | 0.449 |
| | Scikit learn official library / API | Train | 5.005 | 0.721 |
| | | Test | 3.222 | 0.451 |

In summary, there is a slight improvement in $R^2$ by around 0.2 for the setup with 10 input features after feature selection and a marginal improvement of the RMSE score by approximately 0.06.

However, the trained model may not be optimal for predicting the house prices accurately due to the comparatively lower $R^2$ scores for the test data set in all instances. This reflects that the proportion of variance in the dependent variable that can be explained by the independent variables is much lesser in the test data. It is possible that there is insufficient data for the model to learn the important and key patterns or signals required for the accurate prediction. Alternative machine learning algorithms can also be explored to improve the predictions.

## Dataset

The dataset is taken from the UCI Machine Learning repository. It consists of 506 records with a total of 13 attributes as follows.

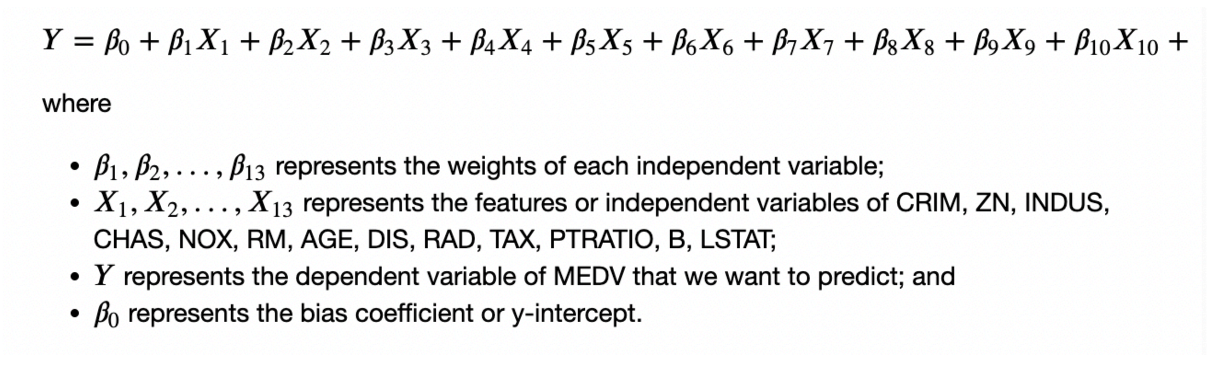
- CRIM: per capita crime rate by town
- ZN: proportion of residential land zoned for lots over 25,000 sq. ft.
- INDUS: proportion of non-retail business acres per town

- CHAS: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
- NOX:  nitric oxides concentration (parts per 10 million)
- RM: average number of rooms per dwelling
- AGE: proportion of owner-occupied units built prior to 1940
- DIS: weighted distances to five Boston employment centres
- RAD: index of accessibility to radial highways
- TAX: full-value property-tax rate per $10,000
- PTRATIO: pupil-teacher ratio by town
- B: 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
- STAT: % lower status of the population
- MEDV (**Target / Dependent variable**): Median value of owner-occupied homes in $1000's

Based on a brief analysis, we noted that there is no missing values in the dataset, which is also affirmed in the data documentation. No data imputation is conducted. Normalisation and rescaling of the features were conducted to ensure that the input features are on a common scale to prevent any unequal contribution to the model which may result in inaccuracies. Outliers were not drop or replaced due to the relatively low number of data points and to prevent any unnecessary bias.

### Linear Regression Equations

There are a total of 13 input features and 1 output. The following regression equation is used to predict the housing prices in Boston. Our objective is to determine the y-intercept, also known as bias, and the coefficients, also known as weights, of each input features such that the RMSE is minimised. The coefficients/weights of the input features represent the amount of change in the output (i.e., housing price) for a unit change in a specific input.



$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \beta_5 X_5 + \beta_6 X_6 + \beta_7 X_7 + \beta_8 X_8 + \beta_9 X_9 + \beta_{10} X_{10} +$$

where

- $\beta_1, \beta_2, \ldots, \beta_{13}$ represents the weights of each independent variable;
- $X_1, X_2, \ldots, X_{13}$ represents the features or independent variables of CRIM, ZN, INDUS, CHAS, NOX, RM, AGE, DIS, RAD, TAX, PTRATIO, B, LSTAT;
- $Y$ represents the dependent variable of MEDV that we want to predict; and
- $\beta_0$ represents the bias coefficient or y-intercept.

Assumptions of linear regression are as follows:

1. Relationship between the independent variables and dependent variable is linear.
2. No multicollinearity. Input features must be independent of each other.
3. Errors should follow a constant variance (i.e., homoscadicity).
4. Both dependent variable and error terms follow a normal distribution.

The final regression equations for the two experiment setups are as follows. Refer to the iPython notebook for the detailed implementation.

- Experiment setup 1: All 13 input features

**Regression Equation (based on user-defined gradient descent function)**

$$MEDV = 23.973 - 8.280(CRIM) + 4.638(ZN) + 0.972(INDUS) + 2.543(CHAS) - 7.977(NOX) + 21.332(RM) + 1.0388(AGE) - 14.682(DIS) + 7.688(RAD) - 7.618(TAX) - 8.302(PTRATIO) + 4.206(B) - 19.605(LSTAT)$$

- Experiment setup 2: 10 input features after feature selection

**Final Regression Equation (based on user-defined model)**

$$MEDV = 21.066 - 7.719(CRIM) + 4.068(ZN) - 3.967(INDUS) + 2.772(CHAS) + 22.294(RM) - 12.314(DIS) + 1.054(RAD) - 6.540(PTRATIO) + 4.404(B) - 20.561(LSTAT)$$

### Evaluation metrics for linear regression - RMSE and R$^2$ Functions

Root Mean Square Error (RMSE) is as a measure that evaluates the predictions of a linear regression model. It shows the difference between the predicted and actual values using Euclidean distance. The formula used to calculate RMSE is a follows.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(y_i - \hat{y})^2}{N}}$$

where

- $y_i$ - actual value of output
- $\hat{y}$ - predicted value of output
- N - number of data points

R squared (R$^2$) is a measure of how far or close the true data points are to the fitted line. It represents the proportion of variation of data points explained by the linear regression model.

$$R^2 = 1 - \frac{RSS}{TSS} = 1 - \frac{\sum_{i=i}^{N}(y_i - \hat{y})^2}{\sum_{i=i}^{N}(y_i - \bar{y})^2}$$

where

- RSS - sum of squares of residuals
- TSS - total sum of squares
- $y_i$ - actual value of output
- $\hat{y}$ - predicted value of output
- $\bar{y}$ - mean value of output
- N - number of data points

Please refer to the iPython notebook for the detailed function implementation.

### Gradient descent algorithm

Gradient descent is an optimisation algorithm to minimise a convex objection function via iterations. A convex function is one where there is only one local minimum which is the global minimum. Linear regression may not be applicable for non-convex function as the local minimum may not be the global minimum.

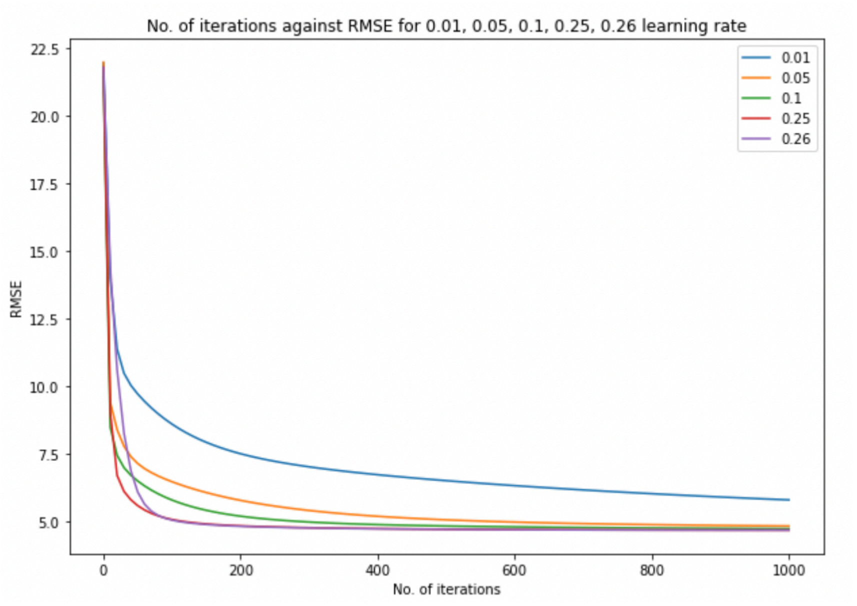
The steps of the gradient descent algorithm is detailed as follows.

1. Initialise a random values for the weight and bias.
2. Calculate the predicted values of y using the initialised weight and bias.
3. Calculate and log the RMSE and R$^2$ of each pair of input features and output values.
4. Update the weight and bias by the scale of learning rate towards the direction where error is minimised by using the differential and partial differential of the cost functions as follows.
   - $bias(\beta_0) = bias - \alpha(\frac{2}{N}\sum(y - \hat{y})^2)$
   - $weights(\beta_0, \beta_1, \ldots, \beta_{13}) = weights - \alpha(\frac{2}{N}\sum(X(y - \hat{y}))^2)$
5. Repeat the process until the minimum RMSE is achieved or not further improvement is possible.

## Optimal learning rate parameter

Learning rate is a hyperparameter that controls how much to scale the bias and weights in the above gradient descent algorithm. The small learning rate might result in a greater number of iterations (i.e., computational resources) to find the bias and weights that minimises the RMSE. A learning rate too large may result in sub-optimal set of bias and weights that is far off the global minimum.

In order to select an optimal learning rate, we can plot the RMSE against a list of learning rates. For learning rate which doesn't result in a reduction in RMSE would automatically be removed from the list. After trial and error, the learning rate above 0.27 is disregarded as it's deemed to be too large and doesn't reduce the RMSE. With reference to the graph below, the learning rate selected is 0.25 as it converges the RMSE score fastest with lower number of iterations required.



## Train and test data

The dataset is split into train and test data set using the ratio of 90:10. The function used to split the data is a user-defined function, splitTT(). The function takes in the normalised pandas dataframe and the percentage of train data and return the split data in a list (i.e., [train, test]). The dataframe is shuffled before splitting. Slicing is used to split the dataframe.

## Implementation of gradient descent algorithm

The analysis was conducted using the user-defined gradient descent algorithm and official Scikit learn linear regression API for both experiment setup below. The codes can be found in the iPython notebook.

**Experiment setup 1: All 13 input features**

| Experiment Setup | Library / Function | Data | RMSE | $R^2$ |
|---|---|---|---|---|
| All 13 input features | User-defined gradient descent function | Train | 4.824 | 0.741 |
| | | Test | 3.285 | 0.430 |
| | Scikit learn official library / API | Train | 4.820 | 0.742 |
| | | Test | 3.287 | 0.429 |

For the RMSE and $R^2$ values above, we note that the differences between the values are very small between 0.001 and 0.004. It is inferred that the predictions between the user-defined function and official API is similar.

Figure 1: Actual vs predicted median house prices in thousands (Train data set)

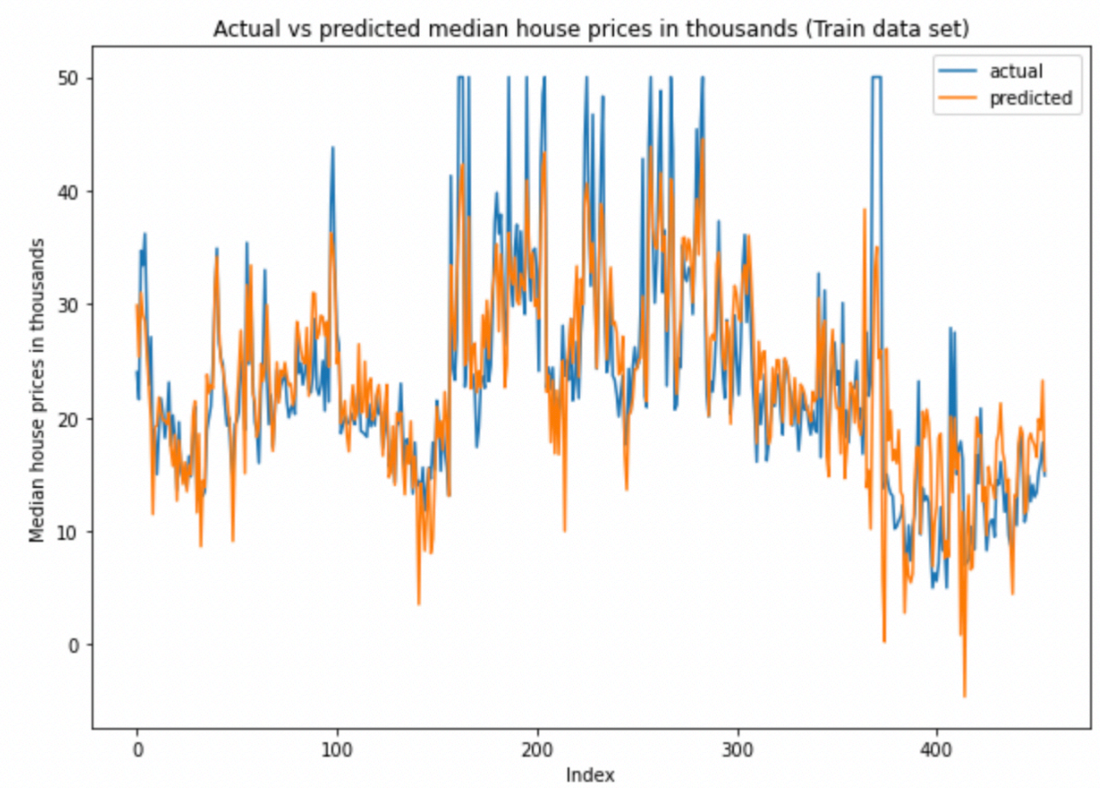


Figure 2: Actual vs predicted median house prices in thousands (Test data set)

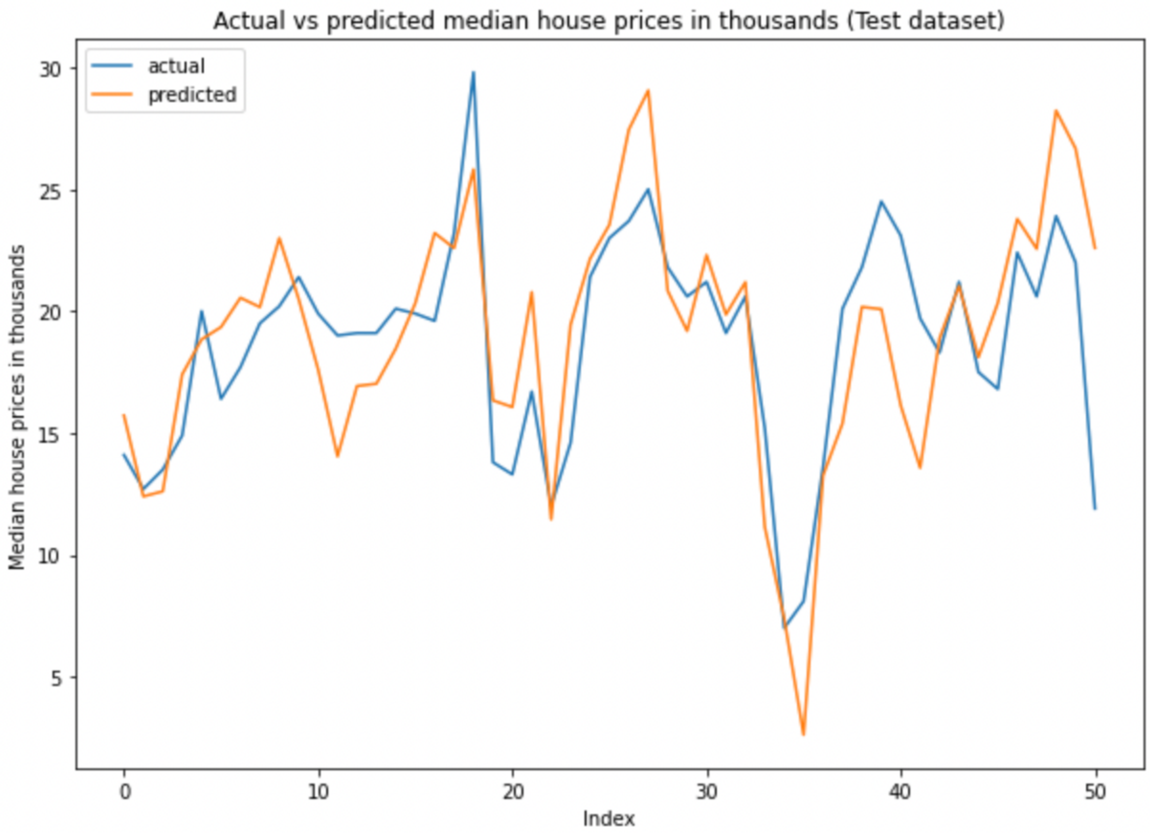
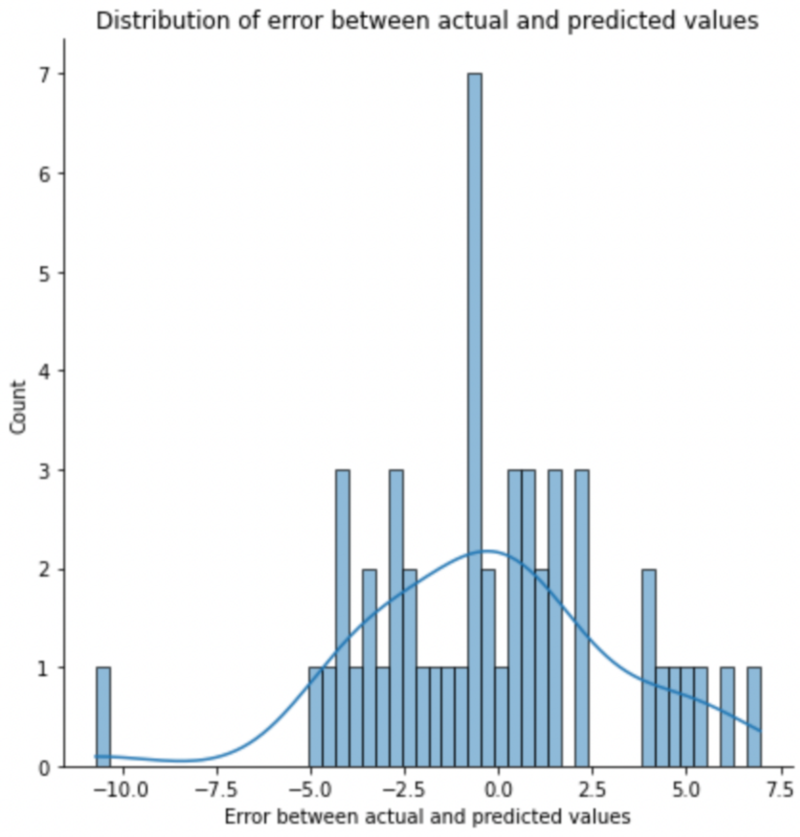


Figure 3: Distribution of error between actual and predicted values for Test data set



In the first experiment setup, the RMSE score for the train set is higher than the test set reflecting that the error between the predicted and actual values are smaller in the test data. However, the $R^2$ squared value of the train set is higher than the test set. This means that the proportion of variance in the dependent variable that can be explained by the independent variables is lesser in test set. We will explore if it's possible to improve the performance through feature selection.

**Experiment setup 2: 10 input features after feature engineering**

| Experiment Setup | Library / Function | Data | RMSE | $R^2$ |
|---|---|---|---|---|
| 10 input features after feature selection | User-defined gradient descent function | Train | 5.006 | 0.721 |
| | | Test | 3.229 | 0.449 |
| | Scikit learn official library / API | Train | 5.005 | 0.721 |
| | | Test | 3.222 | 0.451 |

In this setup, feature selection via backward elimination is done using OLS stats model p-value and analysis of variance inflation factor (VIF) to remove any unnecessary input features may not contribute to the model prediction. The detailed implementation can be found in the iPython notebook. The following features were dropped due to high p-value (>0.05) indicating that the features are insignificant or high VIF (>20) indicating that the feature has high multicollinearity with other input features: NOX, AGE, TAX.

The RMSE and $R^2$ were recalculated with the new model built on the remaining 10 input variables.

Figure 4: Actual vs predicted median house prices in thousands (train data set)

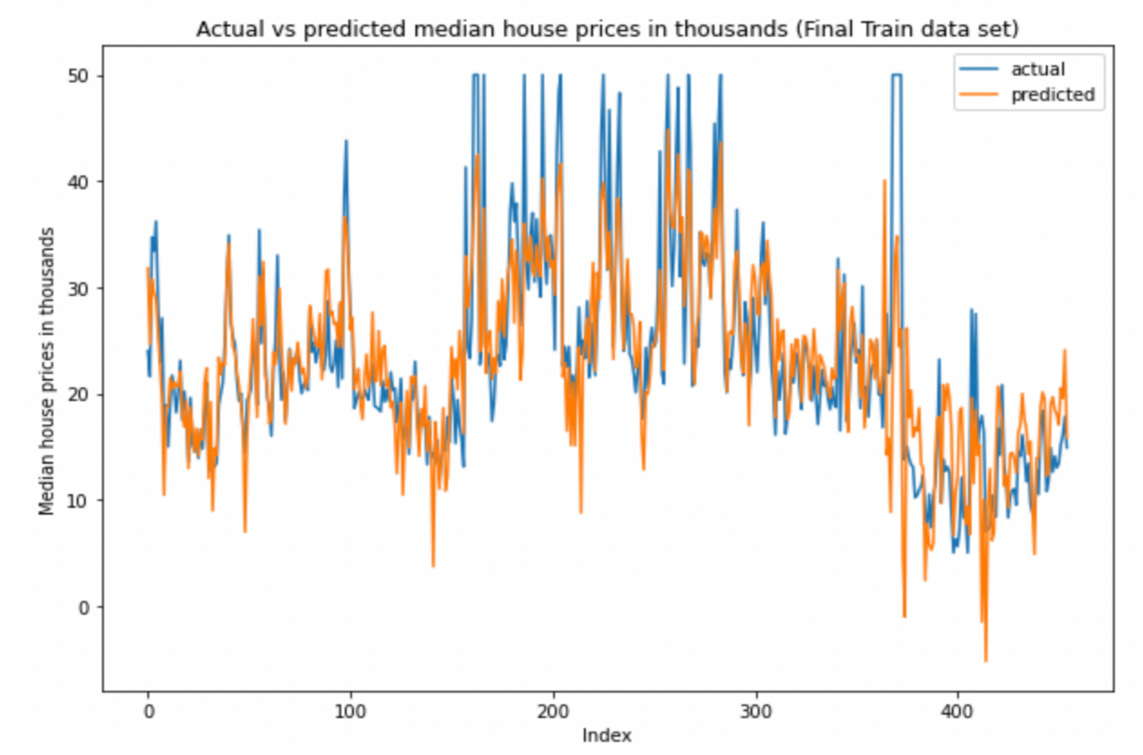


Figure 5: Actual vs predicted median house prices in thousands (test data set)

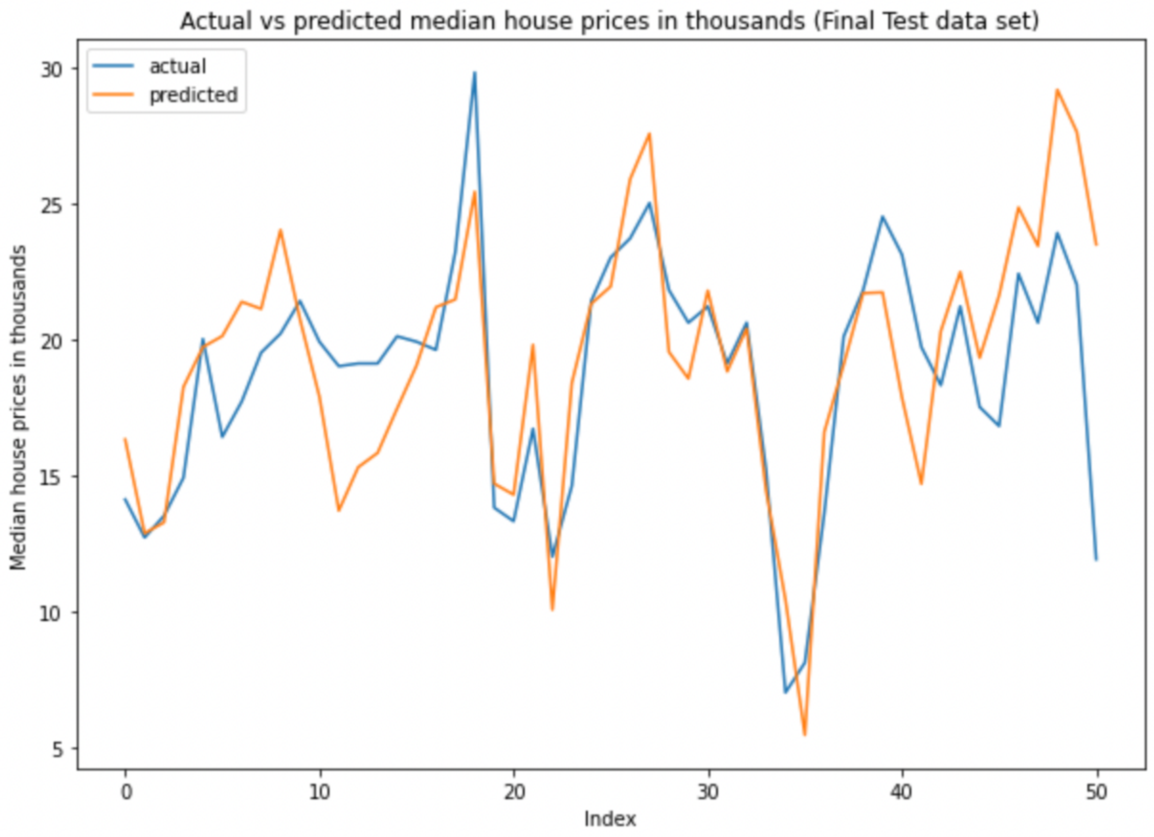
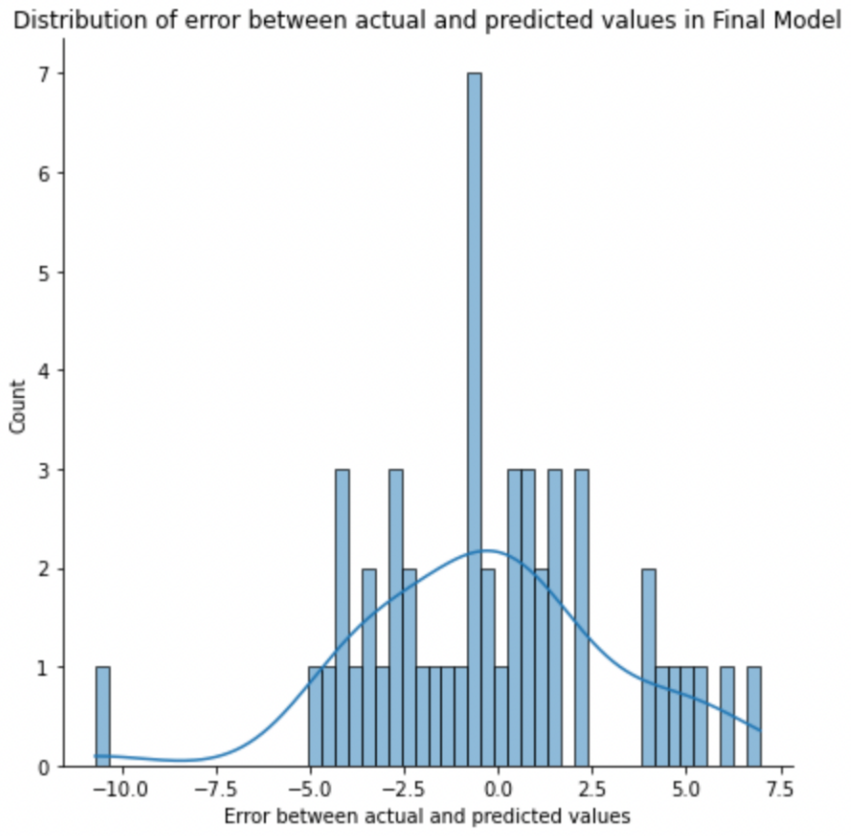


Figure 6: Distribution of error between actual and predicted values (test data set)



Although there is a slight improvement in the RMSE and $R^2$ values by approximately 0.06 and 0.2 as compared to experiment, the improvement is not significant. The $R^2$ value is considered low for the model to accurately predict the housing price. However, a reduction in the number of features used in the model would improve efficiency and reduce computational cost.

It may be worth noting that the RMSE and $R^2$ values of the train data set deteriorate in experiment 2, which may imply that the model fit itself less to the noises in the train data set.

That said, the number of data points and features in the dataset is rather small.  As such, all of the data points are deemed to be considerably important. Pre-processing steps such as dropping any potential outliers were not considered for the modelling due to the low number of data points. It may be possible  to improve the model by collecting more data points and samples so that the model could pick up the important signals and patterns for predicting the house prices during modelling. Alternatively, other machine learning algorithms, such as decision trees, ensemble models, could also be explored which may be more suitable for the data.

## Acknowledgements

1. Scikit Learn: Linear Regression Source Code
2. Statlogy: RMSE vs R-Squared: Which Metric Should You Use
3. Seaborn Official Documentation
4. Towards Data Science: Gradient Descent Algorithm – a deep dive
5. Wikipedia: Gradient descent