MACHINE LEARNING ASSIGNMENT 3

Rawan Ghatwary 40-1649

Chapter 1 Introduction

1.1 Regression Analysis

Linear regression (LR), a parametric model and a supervised learning algorithm, is one of the least complicated ML algorithms. It is used to study the linear relationship between a dependent variable and one or more independent variable(s) (features). LR is generally classified into two types: simple linear regression (SLR) and multiple linear regression (MLR). SLR refers to LR with only one independent variable, while MLR refers to LR with multiple independent variables. The LR model representation is shown in figure 1 where a training dataset D of n input-output pairs $D = \{(x_i, y_i), i = 1, \dots, n\}$ is given as an input. The symbol x_i denotes the independent or the regressor variable(s), while y_i denotes the dependent or the response variable of the i^{th} sample. The dataset is then fed to the learning algorithm that aims to output a function, denoted by *h*. The function *h* stands for hypothesis that takes an input *X* and outputs the estimated values $\hat{Y}$ of the dependent variables *Y*.

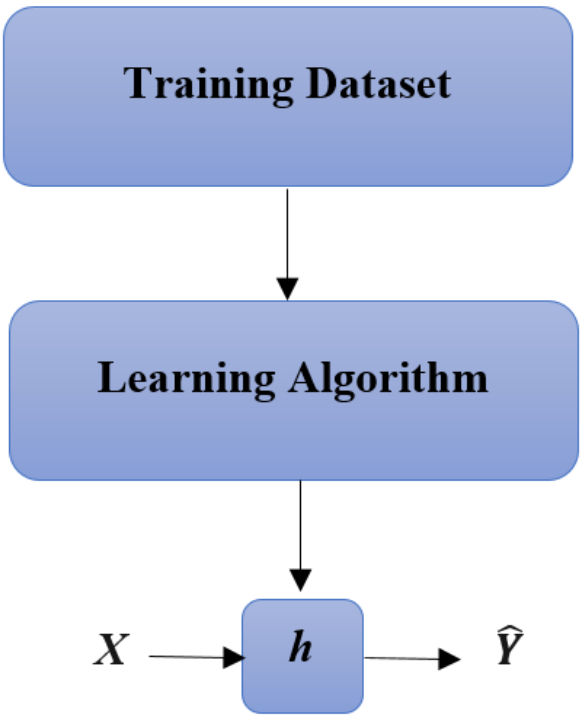


Figure 1 Linear Regression Model Representation

The hypothesis function for SLR model is represented as:

$$h(x_i) = \beta_0 + \beta_1 x_i \tag{1.1}$$

where $h(x_i)$ is the predicted value of the dependent variable y_i, x_i is the independent variable, β_0 is the y-intercept or the bias coefficient that gives an extra degree of freedom to the model, and β_1 is the slope coefficient for the independent variable x_i. In many cases, the involvement of a single independent variable doesn't suffice to explain the dependent variable. Thus, MLR, the extension of SLR, is used to study the linear relationship between a dependent variable and multiple independent variables. The hypothesis function for MLR model is shown as:

$$h(x_i) = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_m x_{im} \tag{1.2}$$

For a dataset that consists of *n* samples and *m* features, the subscripts $x_{i1}, x_{i2}, \dots, x_{im}$ present the independent variables of the i^{th} sample, and $h(x_i)$ is the predicted value of y_i. The unknown function parameters β_j for $j = 0,1, \dots, m$ are called regression coefficients. To represent the hypothesis function in a more efficient way, an additional zero feature $x_0 = 1$ is defined; thus, Eq. (1.2) can be rewritten as:

$$h(x_i) = \beta_0 x_{i0} + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_m x_{im} \tag{1.3}$$

It is more efficient to use matrices when dealing with MLR model. Thus, the hypothesis function of MLR model given by Eq. (1.3) can be rewritten in matrix notation as follows:

$$h(x) = X\beta \tag{1.4}$$

where

$$\begin{bmatrix} h(x_1) \\ \vdots \\ h(x_n) \end{bmatrix} = \begin{bmatrix} 1 & x_{1,1} & \cdots & x_{1,m} \\ \vdots & \vdots & \cdots & \vdots \\ 1 & x_{n,1} & \dots & x_{n,m} \end{bmatrix} * \begin{bmatrix} \beta_0 \\ \vdots \\ \beta_m \end{bmatrix} \tag{1.5}$$

The independent variables are represented in $n \times (m + 1)$ matrix denoted by *X,* where each row corresponds to a sample, and each column corresponds to a feature with an additional column of ones. In addition, the predicted variables are represented in $n \times 1$ vector, denoted by $h(x)$, and β is a $(m + 1) \times 1$ vector of coefficients to be estimated.

In LR, the model targets to achieve the best coefficients by minimizing the cost function, shown in Eq. (1.6). In the learning process, the used cost function is the mean square error (MSE), which measures the average squared difference between a sample's actual and predicted values. The function is then multiplied by constant $\frac{1}{2}$ as a convenience for the computation of the gradient descent.

$$J(\beta_0, \beta_1, \dots, \beta_m) = J(\beta) = \frac{1}{2n} \sum_{i=1}^{n} (h(x_i) - y_i)^2 \tag{1.6}$$

where the cost function $J(\beta)$ is a function of the coefficients' vector, n is the number of samples, x_i is the input value, $h(x_i)$ is the predicted value, and y_i is the actual value of the i^{th} sample. In fact, the coefficients $\beta_0, \beta_1, \beta_2, \dots, \beta_m$ can be found by using different approaches. In the upcoming subsections, two approaches will be explained in MLR case: normal equation approach and gradient descent approach.

1.2 Normal Equation Approach

The normal equation is a non-iterative approach that targets to find the global minimum or the vector of coefficients β by minimizing the cost function. This approach is very effective and reduces the time complexity, especially when using a dataset with small number of features. Therefore, when the number of features increases, the performance of normal equation approach gradually decreases. The reason behind the performance's reduction is the costly matrix computations since it is compulsory to keep all the data in memory for calculations.

As mentioned in the previous subsection, the objective of LR is to find the best coefficients by minimizing the cost function shown in Eq. (1.6). By using Eq. (1.7), the cost function can be represented in matrix notation as follows:

$$J(\beta_0, \beta_1, \dots, \beta_m) = \frac{1}{2n}\sum_{i=1}^{n}(h(x_i) - y_i)^2$$

$$J(\beta) = (X\beta - Y)^T.(X\beta - Y)$$

$$= (\beta^T X^T - Y^T)(X\beta - Y)$$

$$= \beta^T X^T X\beta - \beta^T X^T Y - Y^T X\beta + Y^T Y \quad (1.7)$$

where β is a vector of coefficients needed to be obtained, X is a matrix of the training data that consists of n rows (samples) and m columns (features), and Y is a vector of all target values. Note that $\frac{1}{2n}$ is removed as it won't make any difference in the derivation process; however, it is used in gradient descent approach for mathematical convenience. In addition, to obtain the square of a vector or matrix, a vector/matrix is multiplied by its transpose. Since $\beta^T X^T Y$ is 1×1 matrix or a scalar, and its transpose $(\beta^T X^T Y)^T = Y^T X\beta$ is the same scalar, Eq. (1.7) is rewritten as:

$$J(\beta) = \beta^T X^T X\beta - 2\beta^T X^T Y + Y^T Y \quad (1.8)$$

For the purpose of minimizing the cost function, Eq. (1.8) is differentiated with respect to β, and then the derivative is equated to zero to obtain the optimal coefficients β as shown:

$$\frac{d(J(\beta))}{d\beta} = 2X^T X\beta - 2X^T Y = 0 \quad (1.9)$$

$$2X^T X\beta = 2X^T Y$$

The above equation, which is called normal equation, can be multiplied by the inverse of $X^T X$ to obtain the optimal coefficients β as the following:

$$\beta = (\mathrm{X^T X})^{-1}\mathrm{X^T\ Y} \tag{1.10}$$

Once, the vector of the coefficients β is obtained, the vector of the dependent variables *Y* can be predicted as follows:

$$\hat{Y} = X\beta \tag{1.11}$$

1.3 Gradient Descent Approach

The performance of normal equation approach isn't always guaranteed since it is only applied to certain cases; thus, gradient descent method, is used instead. Gradient descent is a first-order iterative optimization approach that aims to find the local or the global minima of a function. The initial step in gradient descent is to set the coefficients to certain values (e.g. all zeros), and repeatedly adjust them in the direction that reduces the cost function $J(\beta)$ until a minimum value is reached. As the model iterates, it gradually converges towards the minimum where more improvements to the coefficients cause little or no changes in cost function. This procedure is applied by using the following update rule, which is known as gradient descent:

$$\begin{aligned}\beta_j &= \beta_j - \eta \frac{d}{d\beta_j} J(\beta_0, \beta_1, \dots, \beta_m) \\ &= \beta_j - \eta \frac{d}{d\beta_j} \frac{1}{2n} \sum_{i=1}^{n} (h(x_i) - y_i)^2, \quad j = 0,1, \cdots, m\end{aligned} \tag{1.12}$$

The above update rule is repeated for $j = 0,1, \cdots, m$ until convergence, and the coefficients must be updated simultaneously. The symbol η denotes the learning rate, a value between 0 and 1 that controls the sizes of steps taken, and $\frac{d}{d\beta_j} J(\beta_0, \beta_1, \dots, \beta_m)$ is the rate of change of $J(\beta_0, \beta_1, \dots, \beta_m)$ with respect to β_j. The partial derivative of $J(\beta)$ with respect to β_j is the summation of the product of two terms as follows

$$\frac{dJ(\beta)}{d\beta_j} = \frac{1}{2n}\sum_{i=1}^{n}\frac{d}{d\beta_j}(h(x_i) - y_i)^2$$
$$= 2 * \frac{1}{2n}\sum_{i=1}^{n}(h(x_i) - y_i) * \frac{d}{d\beta_j}(\beta_j x_{ij} - y_i)$$
$$= \frac{1}{n}\sum_{i=1}^{n}(h(x_i) - y_i) * x_{ij} \quad (1.13)$$

After solving the derivation, Eq. (1.12) in each iteration can be expanded by substituting Eq. (1.13) into Eq. (1.14) as:

Repeat until convergence{

$$\beta_0 = \beta_0 - \eta\frac{1}{n}\sum_{i=1}^{n}(h(x_i) - y_i) * x_{i0}$$

$$\beta_1 = \beta_1 - \eta\frac{1}{n}\sum_{i=1}^{n}(h(x_i) - y_i) * x_{i1}$$

$$\beta_2 = \beta_2 - \eta\frac{1}{n}\sum_{i=1}^{n}(h(x_i) - y_i) * x_{i2}$$

$$\vdots$$
$$\vdots$$

$$\beta_m = \beta_m - \eta\frac{1}{n}\sum_{i=1}^{n}(h(x_i) - y_i) * x_{im} \quad (1.14)$$

}

It is more efficient to express the equations in matrix notation while implementing the gradient descent algorithm. Therefore, Eq. (1.14) can be rewritten as:

$$\beta = \beta - \eta\nabla J(\beta) \quad (1.15)$$

where $\nabla J(\beta)$ is a $(m+1) \times 1$ vector and can be expressed as:

$$\nabla J(\beta) = \begin{bmatrix} \frac{dJ(\beta)}{d\beta_0} \\ \vdots \\ \frac{dJ(\beta)}{d\beta_m} \end{bmatrix} \quad (1.16)$$

For representing Eq. (1.13) in matrix notation, the summation of the product of two terms can be expressed as the product of two vectors. As a result, by substituting Eq. (1.4) in Eq. (1.13), Eq. (1.13) can be rewritten in matrix notation as:

$$\frac{dJ(\beta)}{d\beta_j} = \frac{1}{n}\bar{x}_j^T(X\beta - Y)$$

$$\nabla J(\beta) = \frac{1}{n}X^T(X\beta - Y) \tag{1.17}$$

where $\bar{x}_j^T$ represents the *n* elements of the j^{th} column in matrix X. Finally, by substituting Eq. (1.17) into Eq. (1.15), the matrix notation of the gradient descent rule shown in Eq. (1.18) is:

$$\beta = \beta - \frac{\eta}{n}X^T(X\beta - Y) \tag{1.18}$$

1.4 Regularization

Regularization is another form of regression that prefer the simplest hypothesis to fit the data. So, in regularization, a regularization term is added to the cost function in Eq. (1.6) as follows:

$$J(\beta_0, \beta_1, \dots, \beta_m) = J(\beta) = \frac{1}{2n}\sum_{i=1}^{n}(h(x_i) - y_i)^2 + \frac{\lambda}{2m}\sum_{j=1}^{m}(\beta_j)^2 \tag{1.19}$$

In the above equation, the regularization term is the summation of all coefficients, multiplied by $\frac{\lambda}{2m}$. The λ is called lambda and m is the number of coefficients. In order to find the best coefficients, the previous discussed gradient descent is also used by using the following equations:

Repeat until convergence{

$$\beta_0 = \beta_0 - \eta\frac{1}{n}\sum_{i=1}^{n}(h(x_i) - y_i)$$

$$\beta_j = \beta_j - \eta\frac{1}{n}\left(\left(\sum_{i=1}^{n}(h(x_i) - y_i) * x_{ij}\right) + \lambda\beta_j\right) \tag{1.20}$$

}

Chapter 2 Methodology

2.1 Data Analysis

First of all, I read the data which contains 21 features and I dropped the id and date features as they aren't important. Then, I plot each feature with the output (Price), shown in the graphs below:

Figure 2

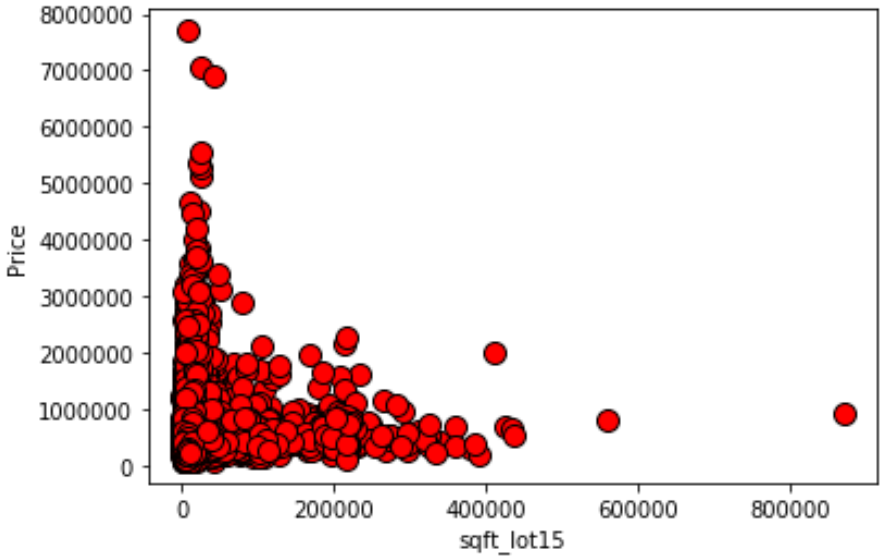


Figure 3

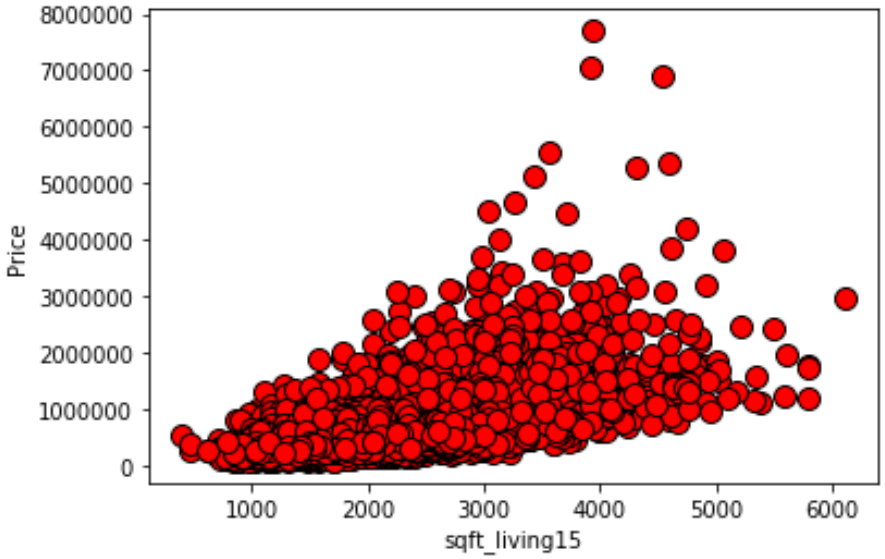


Figure 4

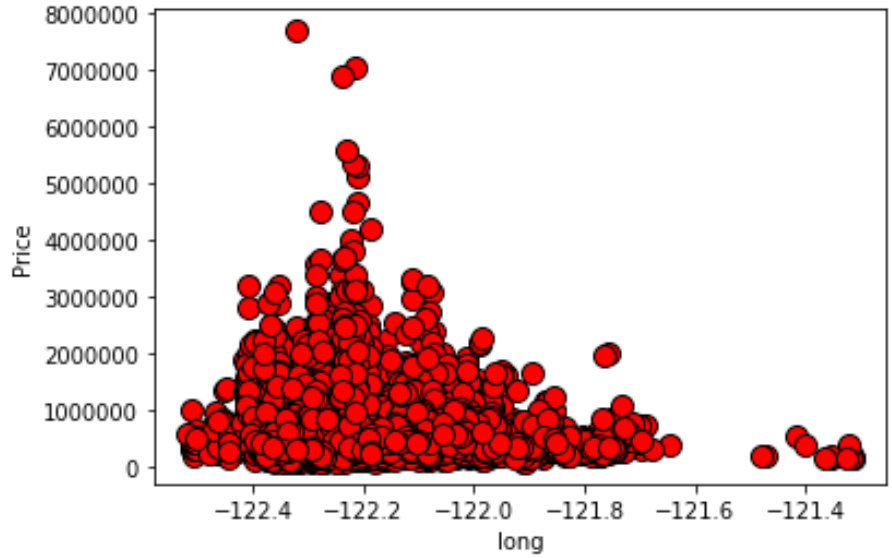


Figure 5

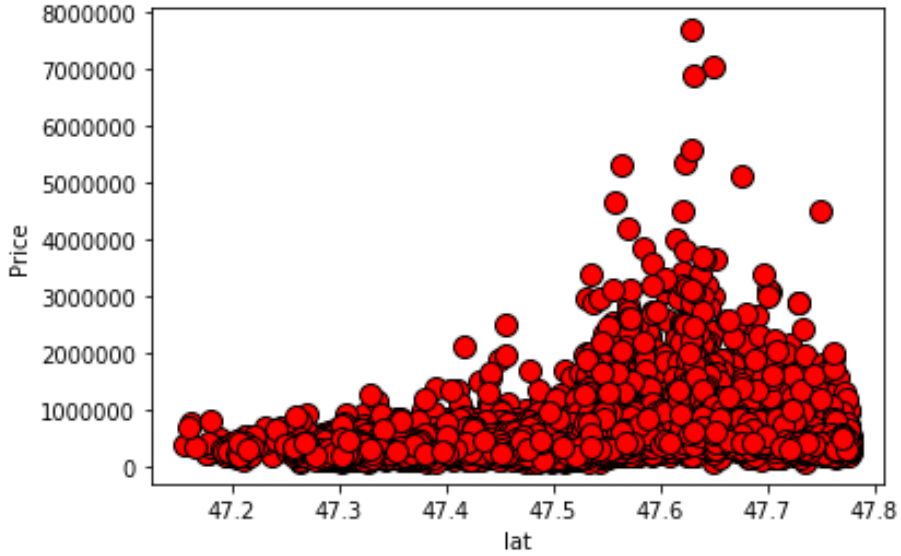


Figure 6

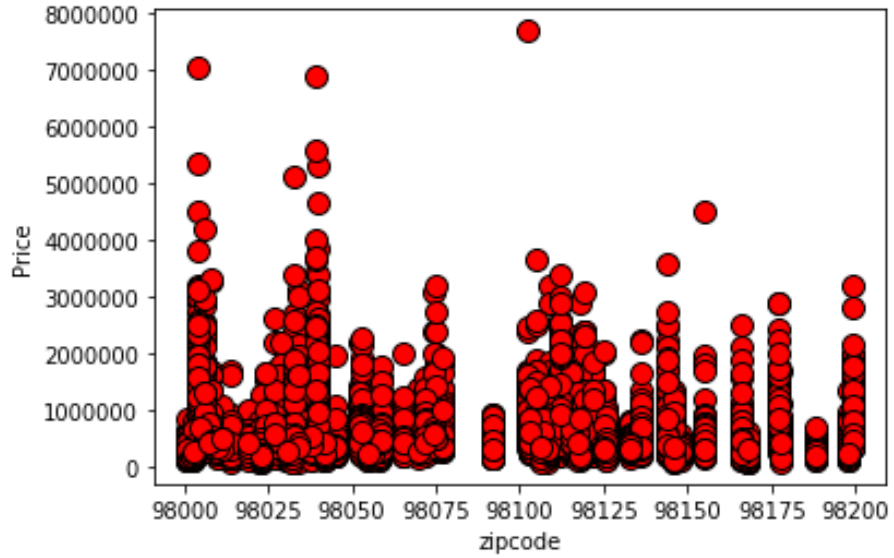


Figure 7

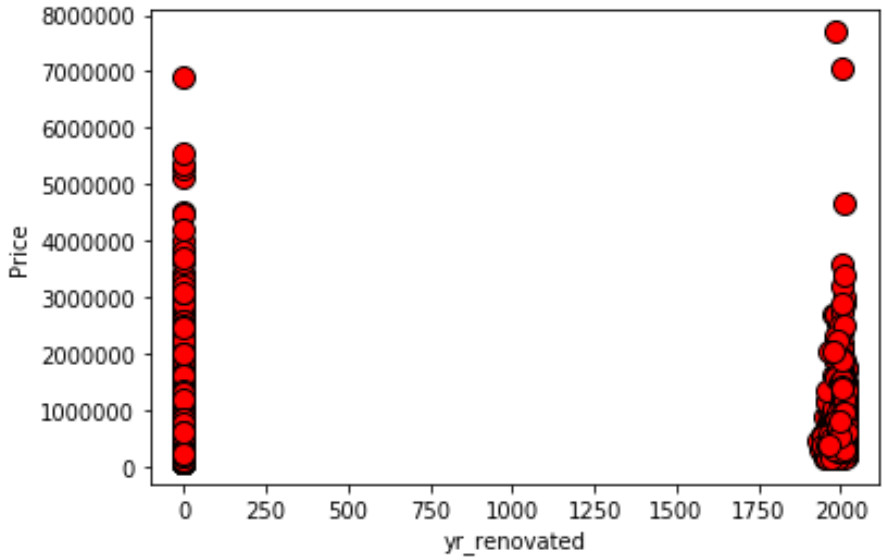


Figure 8

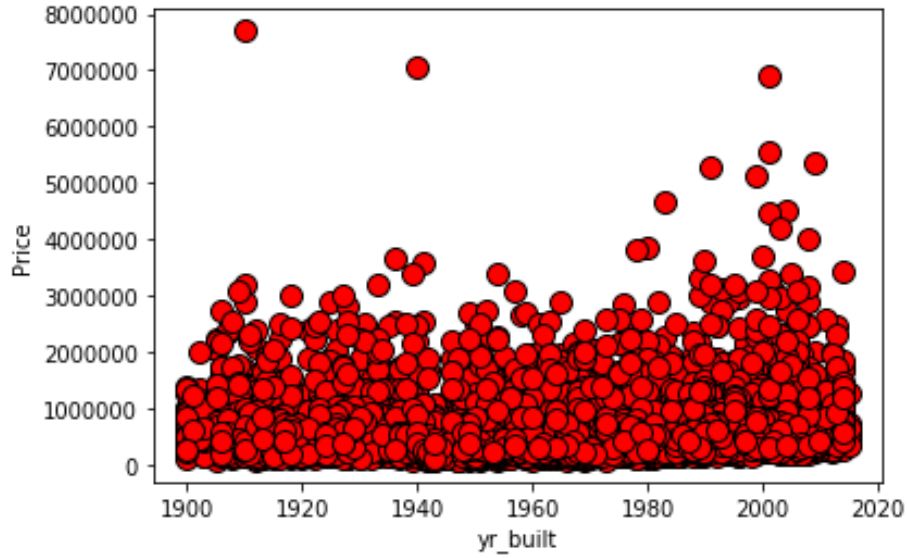


Figure 9

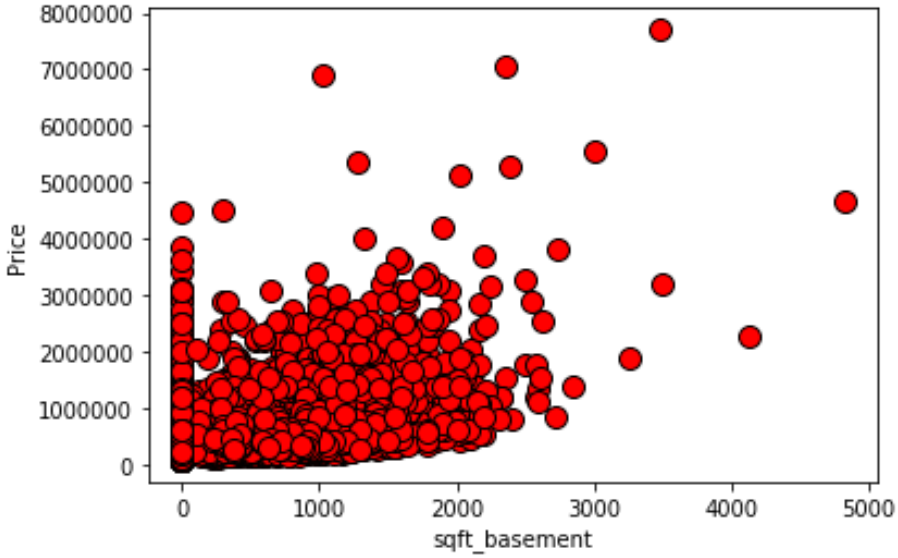


Figure 10

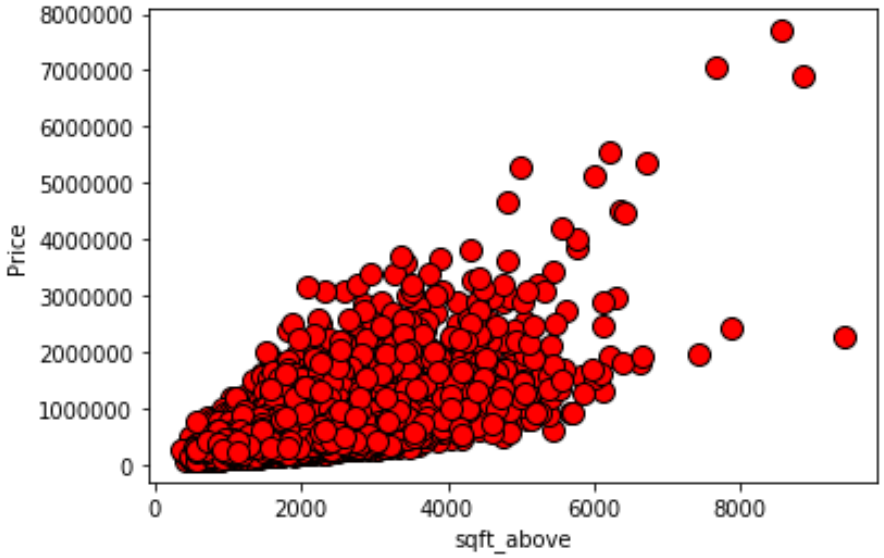


Figure 11

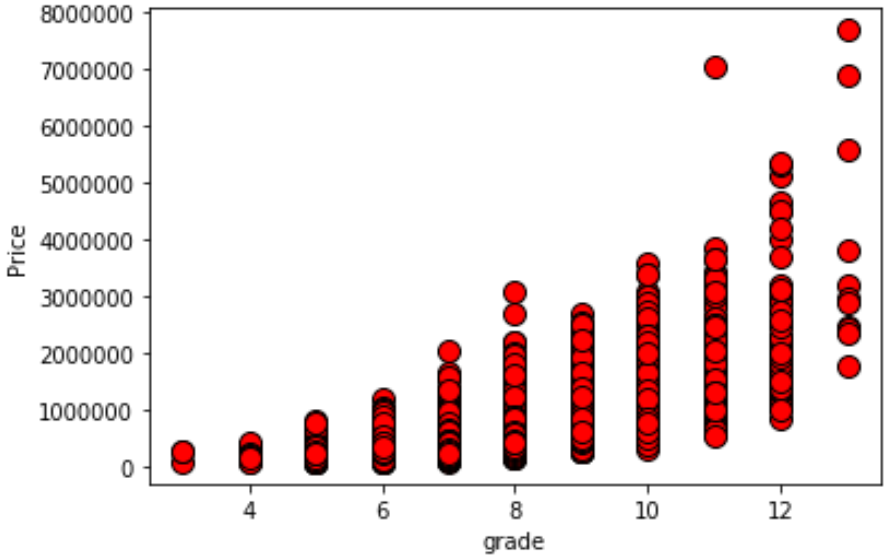


Figure 12

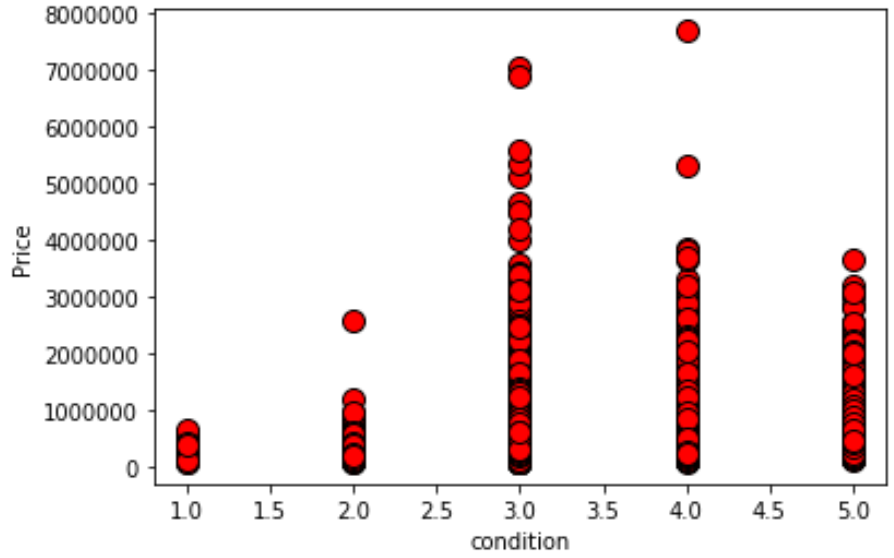


Figure 13

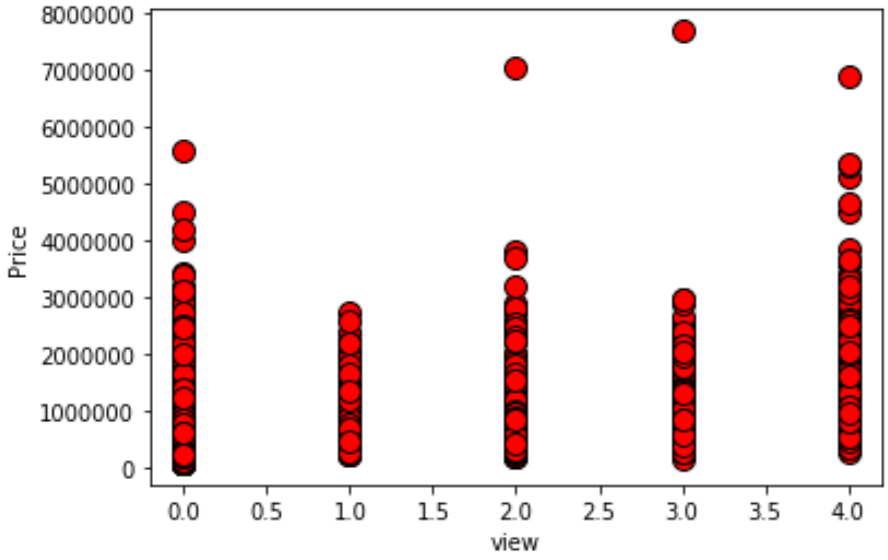


Figure 14

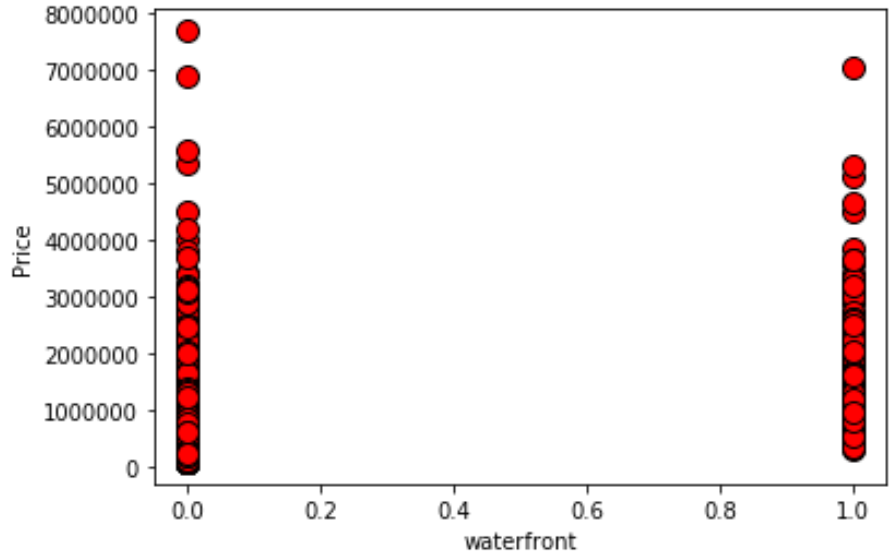


Figure 15

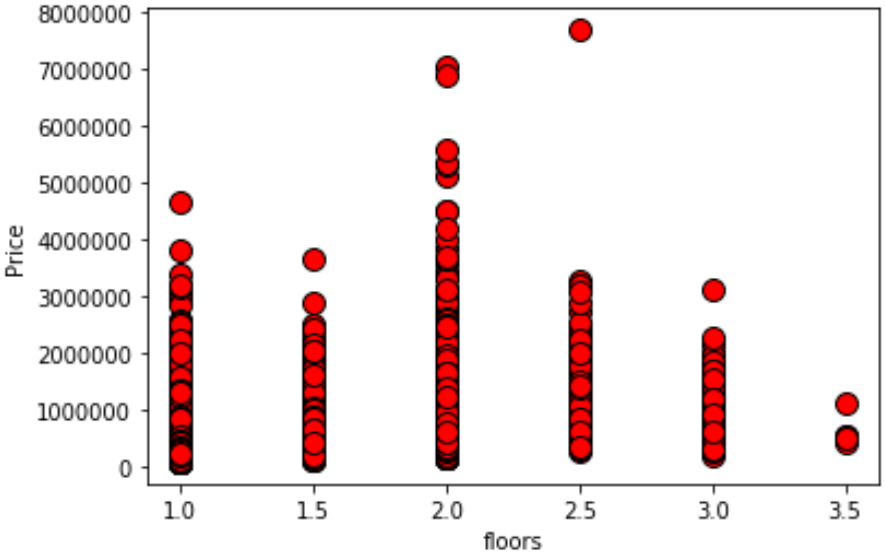


Figure 16

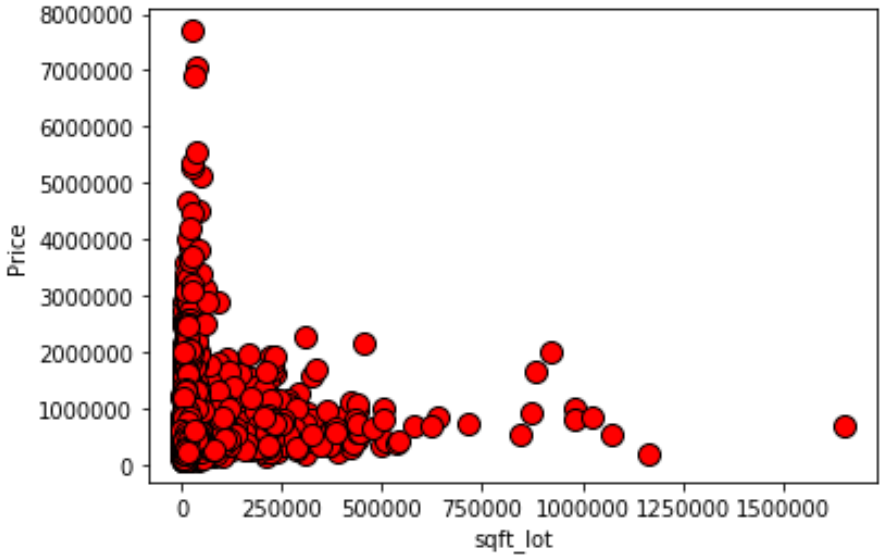


Figure 17

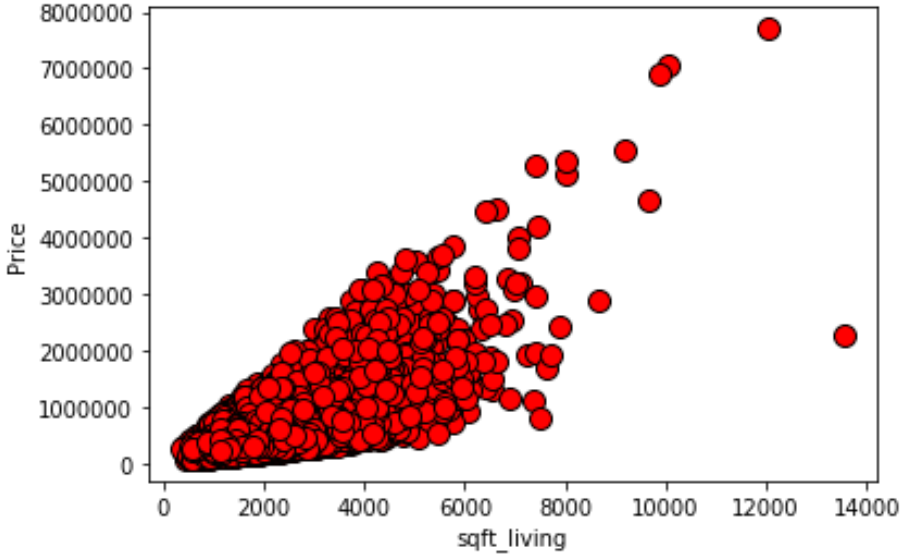


Figure 18

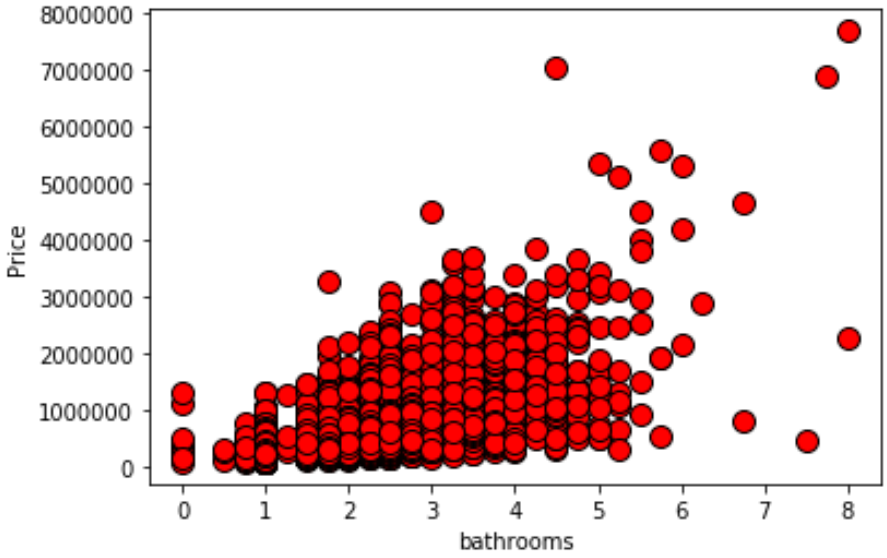
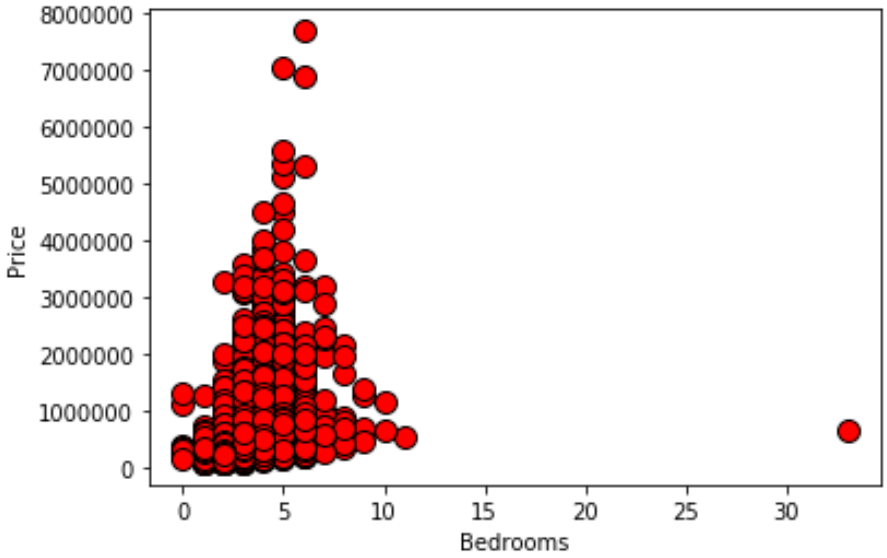


Figure 19



By visualizing the graphs, we can see that there are multiple discrete features, floors (fig 15), waterfront (fig 14), view (fig 13), condition (fig 12), grade (fig 11), and yr_renovated (fig 7), and zipcode (fig 6). Also, we can know from figures (17, 11, 10, 3) that sqft_living, grade, sqft_above, sqft_living15 features are highly correlated with the price output.

2.2 Feature Selection Strategy

The feature selection strategy used here is called Filter method which filters data and chooses only the relevant features. The filtering is done by using correlation matrix that uses Pearson correlation. The Pearson correlation values are between [-1, 1], a value closer to 0 implies weaker correlation, a value closer to 1 implies stronger positive correlation, and a value closer to -1 implies stringer negative correlation. So first the Pearson correlation heatmap is plotted as shown in fig 20, and then we see the correlation between the independent variables and the Price output variable. Finally, we only select the features which has a correlation of above 0.4 with the output variable. The selected features are shown in table 1 with their correlation values.

Figure 20

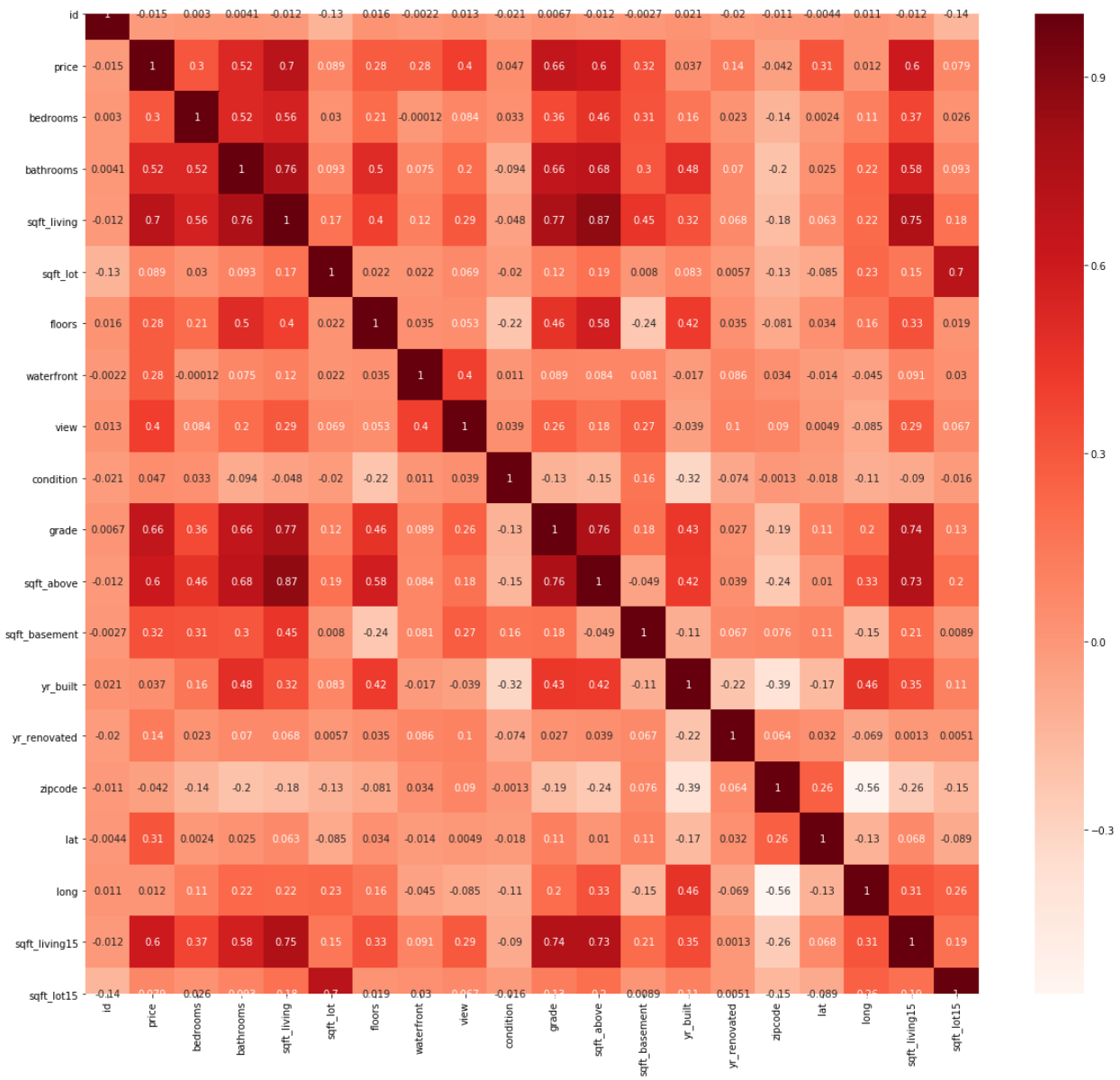


Table 1 Selected Features with correlation above 0.4

	Price
Price	1.000
Bathrooms	0.524
Sqft_living	0.701
view	0.403
grade	0.663
Sqft_above	0.604
Sqft_living15	0.599

2.3 Dividing and Normalizing the Data

After dropping the irrelevant features, we divide dataset into 60% training set, 20% testing set, and 20% validation set. Then, we normalize the dataset by transforming the features to be within a specific range so that no feature is dominated by the other. Here, the Z-score normalization is the used. It transforms the data by calculating the Z-score of each value and then replacing the value with the calculated Z-score. The Z-score can be calculated as follows:

$$\dot{x}(i) = \frac{x(i) - \text{mean}(x)}{\text{stand\_dev(x)}} \tag{2.1}$$

where $x(i)$ is the old feature value, $\text{mean}(x)$ is the average of a feature column, stand_dev(x) is the standard deviation of a feature column, and $\dot{x}(i)$ is the new calculated Z-score.

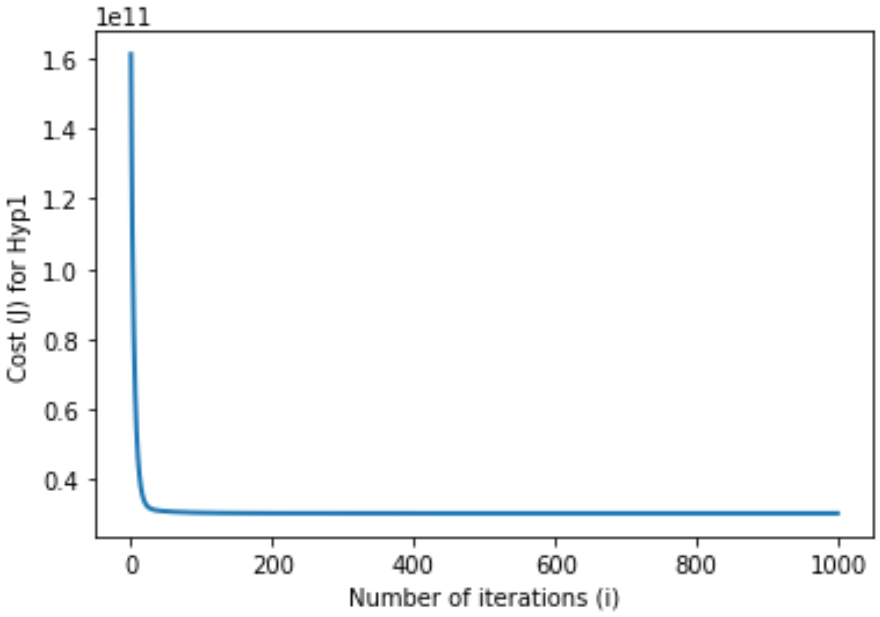
2.4 Four Hypotheses used

2.4.1 First hypothesis

The first hypothesis used is the normal in which an intercept term is added to our features and no any operations are applied on the selected features. Then, gradient decent approach is applied to get the best coefficients (thetas). In the gradient descent, the chosen learning rate (alpha) is 0.1 and the chosen number of iterations is 1000. These two values are the best for getting the best coefficients that minimize the cost function. Figure 21 shows the cost of hypothesis 1 versus the number of iterations, and it shows how quickly convergence is reached as the cost decreases fast. Hypothesis 1 is shown below as follows:

$$h(x_1) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 + \beta_6 x_6 \quad (2.2)$$

Figure 21 Cost of Hyp1 vs no. of iterations

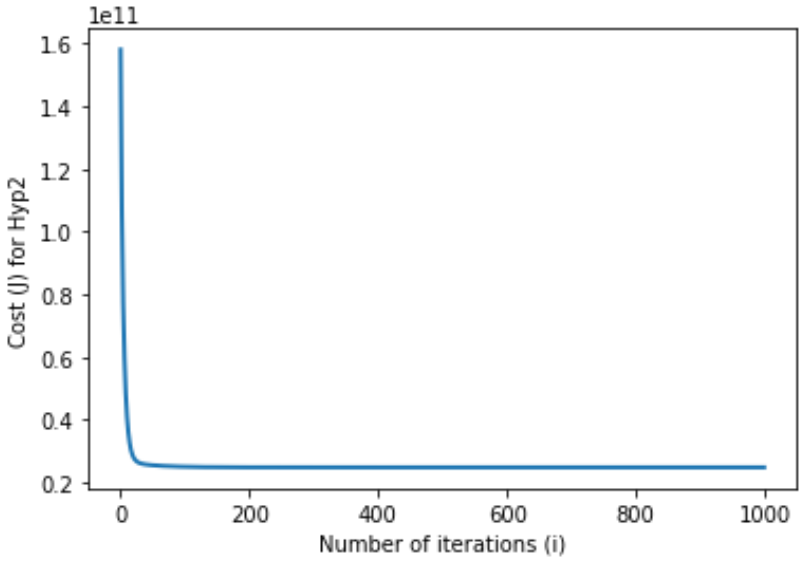


2.4.2 Second hypothesis

The second hypothesis used is squaring all selected features. Then, data is normalized by z-score normalization, then an intercept term is added to our features, then gradient decent approach is applied to get the best coefficients (thetas). In the gradient descent, the chosen learning rate (alpha) is also 0.1 and the chosen number of iterations is 1000. These two values are the best for getting the best coefficients that minimize the cost function. Figure 22 shows the cost of hypothesis 2 versus the number of iterations, and it shows how quickly convergence is reached as the cost decreases fast. Hypothesis 2 is shown below as follows:

$$h(x_2) = \beta_0 + \beta_1 {x_1}^2 + \beta_2 {x_2}^2 + \beta_3 {x_3}^2 + \beta_4 {x_4}^2 + \beta_5 {x_5}^2 + \beta_6 {x_6}^2 \quad (2.3)$$

Figure 22 Cost of Hyp2 vs no. of iterations

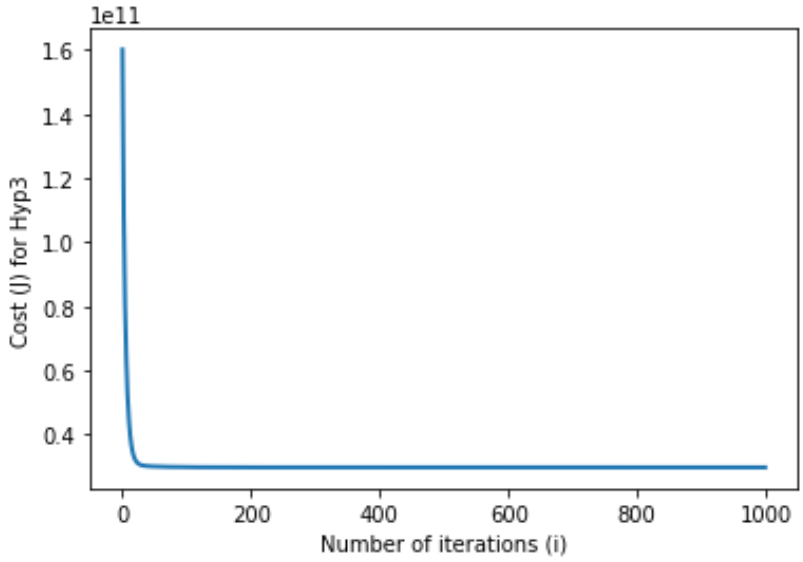


2.4.3 Third hypothesis

The third hypothesis used is squaring only the features that have correlation value less than 0.6. Then, data is normalized by z-score normalization, then an intercept term is added to our features, then gradient decent approach is applied to get the best coefficients (thetas). In the gradient descent, the chosen learning rate (alpha) is also 0.1 and the chosen number of iterations is 1000. These two values are the best for getting the best coefficients that minimize the cost function. Figure 23 shows the cost of hypothesis 3 versus the number of iterations, and it shows how quickly convergence is reached as the cost decreases fast. Hypothesis 3 is shown below as follows:

$$h(x_3) = \beta_0 + \beta_1 {x_1}^2 + \beta_2 x_2 + \beta_3 {x_3}^2 + \beta_4 x_4 + \beta_5 x_5 + \beta_6 {x_6}^2 \quad (2.4)$$

Figure 23 Cost of Hyp3 vs no. of iterations

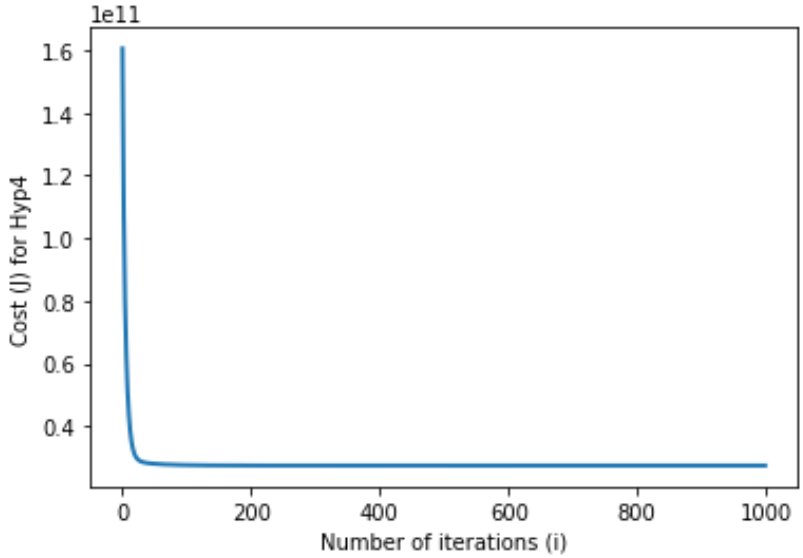


2.4.4 Fourth hypothesis

The fourth hypothesis used is applying on view feature power of 2, applying on grade feature power of 4, and applying on sqft_living15 feature power of 4. This combination is selected by trial and error as it decreases the error. Then, data is normalized by z-score normalization, then an intercept term is added to our features, then gradient decent approach is applied to get the best coefficients (thetas). In the gradient descent, the chosen learning rate (alpha) is also 0.1 and the chosen number of iterations is 1000. These two values are the best for getting the best coefficients that minimize the cost function. Figure 24 shows the cost of hypothesis 4 versus the number of iterations, and it shows how quickly convergence is reached as the cost decreases fast. Hypothesis 4 is shown below as follows:

$$h(x_4) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 {x_3}^2 + \beta_4 {x_4}^4 + \beta_5 x_5 + \beta_6 {x_6}^4 \quad (2.5)$$

Figure 24 Cost of Hyp4 vs no. of iterations



Chapter 3 Results

In the results, computing the error is done by using cost function Eq. (1.6), which measures the average squared difference between a sample's actual and predicted values.

3.1 Four Hypotheses Results by using gradient descent

	Validation data cost	Train data cost	Test data cost
Hyp 1	26708682832.96772	30370607036.861923	24851917025.30008
Hyp 2	28736246061.583313	24893955802.923107	22963394667.807583
Hyp 3	26417422926.21171	29723050176.508636	23936790571.34409
Hyp 4	**25035960512.987022**	27470522599.85691	**22966793464.372013**

As shown in the table above, **Hypothesis four** has the lowest error in the validation test. So, hypothesis four is chosen.

3.2 Comparison between gradient descent and normal equation approach

To compare between two approaches, I uses the best hypothesis which is the fourth one.

	Validation data cost	Train data cost	Test data cost
Gradient descent_hyp4	25035960512.987022	27470522599.85691	22966793464.372013
Normal equation_hyp4	25035958225.85694	27470522599.672333	22966798530.603966

Both approaches have the same errors in validation, train, and test data. These results ensures that I chose the best parameters for gradient descent approach that gives me the best coefficients.

3.3 Comparison between regularization and gradient descent

For comparing between regularization and gradient descent, I chose hypothesis one. In regularization, the chosen alpha is 0.1, the chosen lambda is 0.5, and the chosen number of iterations is 1000.

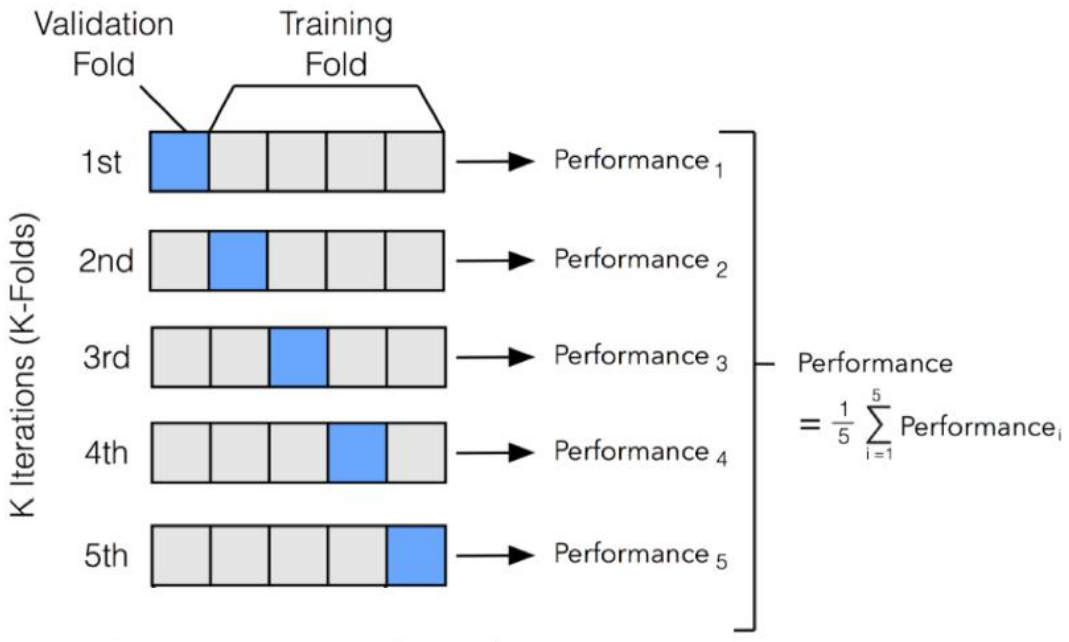
	Validation data cost	Train data cost	Test data cost
Gradient descent_hyp1	26708682832.96772	30370607036.861923	24851917025.30008
regularization_hyp1	26708677953.223972	30370607662.563126	24851495634.03272

From the above table, I can realize that regularization didn't have any effect on the cost. Therefore, I didn't apply on it the best hypothesis which is four.

3.4 Kfolds

In order to avoid overfitting in machine learning models, kfold process is applied. The process of K-fold is as follows: splitting the training set further into K number of subsets, called folds, then training the model on K-1 of the folds, and finally evaluating the model on the K^{th} fold. This process is shown in figure 25, in which a model with $K = 5$ is fitted. In the first iteration, the model is trained by using the last four folds and then evaluated on the first one. In the second iteration, the model is evaluated on second fold, and trained on the remained folds. The process is repeated three more times, each time evaluating on a different fold, and the average performance of all iterations is then calculated.

Figure 25 5-fold



In this assignment, I chose K=10 and I applied 10-fold process with the fourth hypothesis, the best hypothesis. The final average of the 10 fold, using fourth hypothesis, is **26121993089.51937.** The final average of the 10 fold, using first hypothesis, is **28558696865.363354. So, hypothesis four is the best one.**