Machine Learning Project 1

Project Report

House Price Prediction using Linear Regression

By
Qazi Umer Jamil
NUST Registration No: 317920

NUST School of Mechanical and Manufacturing Engineering
Sector H-12, Islamabad, Pakistan

Dated: 21-10-2019

Contents

1 Introduction **1**
1.1 Environment 1
1.2 Files 1
1.3 Data Set 2
1.3.1 Data Headers 2

2 Data Processing **3**
2.1 Data Cleaning 3
2.2 Feature Selection and Engineering 3
2.3 Logarithmic Transformation 4
2.4 Testing and Training Dataset 5
2.5 Feature Normalization 5

3 Multivariate Linear Regression Model **6**
3.1 Gradient Descent Algorithm 6
3.2 Prediction 8
3.3 Model Evaluation Metrics 8

4 Summary **9**

Bibliography **10**

List of Figures

2.1 Histogram of prices variable without transformation 5
2.2 Histogram of prices variable with logarithmic transformation 5

3.1 Cost Function Values vs. Number of epochs 7

List of Tables

1.1 Data Headers 2

2.1 Selected Features 4

3.1 Regression coefficients 7
3.2 Model Evaluation Metrics 8

1. Introduction

The task is to implement and train a linear regression model on the given data for predicting housing prices.

1.1 Environment

MATLAB R2018a is used for model implementation and training.

1.2 Files

The attached zip file contains the following files:

- `main.m`
  Main file of the program
- `dataCleaning_featureEngg.m`
  Implementation of Data Cleaning and Feature Engineering process
- `featureNorm.m`
  Feature Normalization
- `GD.m`
  Implementation of Gradient Descent Algorithm
- `calCost.m`
  Implementation of Cost Function
- `Prediction.m`
  Implementation of Prediction Function
- `modelEval.m`
  Implementation of Model Evaluation Metrics (RMSE, R-Squared)
- `sumFunc.m`
  To calculate sum of columns of input matrices.
- `mulFunc.m`
  To multiply two given matrices.

- `meanFunc.m`

  Calculates the mean of of individual columns of input matrix. Returns a row matrix.

- `createMatrix.m`

  Create a Matrix of input rows, columns, element value.

- `data3.csv`

  Given Dataset

1.3 Data Set

The data is provided in a *.csv* format. To make the data processing task easier, the header of the file (which is the 1st row) and first two columns containing Id and date features are removed manually (Further explained in Sectio 2.2). The new data is saved in the file *data.csv* and is provided with the source code.

1.3.1 Data Headers

The data headers [3] in the given *.csv* file are described in Table 1.1

Table 1.1: Data Headers

Sr. No	**Variable**	**Description**
1	Id	Unique ID of each home sold
2	Date	Date of the home sale
3	Price	Price of each home sold
4	Bedrooms	Number of Bedrooms
5	Bathrooms	Number of Bathrooms
6	Sqft_living	Square footage of apartment interior living space
7	Sqft_lot	Square footage of land space
8	Floors	Number of floors
9	Waterfront	A dummy variable whether the apartment was overlooking the waterfront or not
10	View	An index from 0-4 of how good the view is
11	Condition	An index from 1-5 on condition of the apartment
12	Grade	An index regarding quality level of construction and design
13	Sqft_above	The square footage of the interior housing space that is above ground level
14	Sqft_basement	The square footage of the interior house space that is below ground level
15	Yr_built	The year the house was initially built
16	Yr_renovated	The year of the house's last renovation
17	Zip Code	What zip code area are the house is in
18	Lat	Latitude
19	Long	Longitude
20	Sqft_living15	The square footage of interior housing living space for the nearest 15 neighbours
21	Sqft_lot15	The square footage of the land lots of the nearest 15 neighbours

2. Data Processing

2.1 Data Cleaning

By just scrolling through the data, I found out that there are some training examples in which the number of bedrooms are 0 or 33, which is certainly not possible in reality. Therefore, it is pertinent to remove such training examples. I manually removed every training example which has bedrooms either 0 or greater than 11.

2.2 Feature Selection and Engineering

The Id feature is not relevant in training the linear regression model. Similarly, the date on which a house sold cannot simply tell us about the price as there can be a case when the seller only found a buyer after a long time and this did not had any effect on the price of the house whatsoever.

Also, the feature 'Sqft_above' is linearly dependent on 'Sqft_living' and 'Sqft_basement' features. The relationship is given in equation 2.1

$$Sqft\_above = Sqft\_living - Sqft\_basement \tag{2.1}$$

Having correlated features in the data can lead to the poor performance of the linear regression model. Hence, the feature vector 'Sqft_above' is removed from the data set.

The lat, long coordinates represent a point in 3-dimensional space. Thus, these two features can not be treated independently in training the regression model. One way to address this problem is by using the lat long coordinates of the house to calculate the distance from the lat, long coordinates of a reference location. One such reference location can be the economic hub/center of the city. As this was not given, I took the reference location of the house, which has the maximum price. If the price of a house is higher, chances are the house located in its vicinity will also tend to have higher prices. A house that is farther from the location of the house of the highest price will tend to have a lesser price.

The lat, long coordinates of a house which has a maximum price in the given data can be easily obtained and is implemented in *dataCleaning_featureEngg.m* file. Once the reference lat, long coordinates are known, we can use the Haversine formula to calculate the distance between the two given set of lat long coordinates. The Haversine formula is given in equation 2.2

$$dist = 2r \arcsin(\sqrt{\sin^2(\frac{lat2 - lat1}{2}) + \cos(lat1)\cos(lat2)\sin^2(\frac{long2 - long1}{2})}) \tag{2.2}$$

Here r is the radius of sphere i.e Earth. (lat1, long1) and (lat2, long2) are the latitudes and longitudes of two locations. A new feature vector is created (called dist), which is the distance of a house from the house of the highest price. This new feature vector is appended to the given data, and the feature vectors containing lat, long coordinates were removed as there is no need to include them in the data.

The 'Zipcode' feature represent categorical data, and this type of data is more useful in models such as Decision Trees and K-Means Clustering. Since we already have encoded the geographic information of the house using its lat, long coordinates, I decided to remove this feature vector too. Furthermore, Yr_built and Yr_renovated feature vectors also represent categorical data, and hence these were removed from the data set. Table 2.1 shows the selected features for the model.

Table 2.1: Selected Features

Sr. No	**Feature**	**Description**
1	Bedrooms	Number of Bedrooms
2	Bathrooms	Number of Bathrooms
3	Sqft_living	Square footage of apartment interior living space
4	Sqft_lot	Square footage of land space
5	Floors	Number of floors
6	Waterfront	A dummy variable whether the apartment was overlooking the waterfront or not
7	View	An index from 0-4 of how good the view is
8	Condition	An index from 1-5 on condition of the apartment
9	Grade	An index regarding quality level of construction and design
10	Sqft_basement	The square footage of the interior house space that is below ground level
11	Sqft_living15	The square footage of interior housing living space for the nearest 15 neighbours
12	Sqft_lot15	The square footage of the land lots of the nearest 15 neighbours
13	dist	Distance of house from the house of highest price

2.3 Logarithmic Transformation

The distribution of *prices* variable in the given data is highly skewed positively as shown in Figure 2.1. To transform the *prices* variable into a more symmetrically distribution (Normal Distribution), I used logarithmic transformation [1]. The resulting distribution is shown in Figure 2.2.

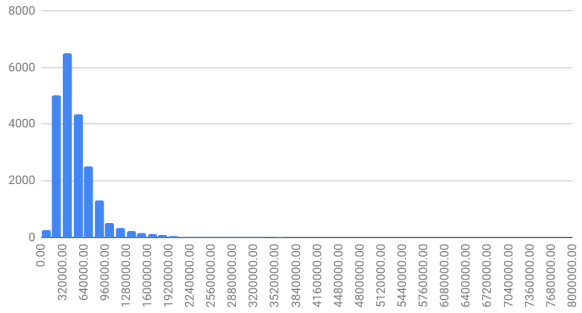


Figure 2.1: Histogram of prices variable without transformation

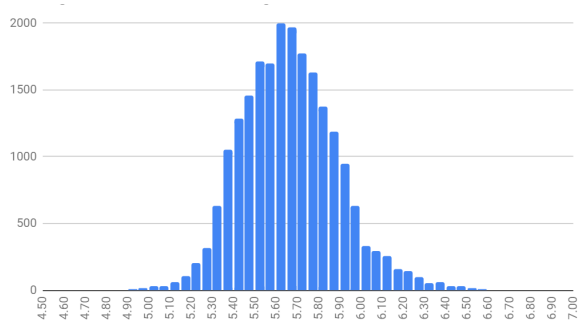


Figure 2.2: Histogram of prices variable with logarithmic transformation

2.4 Testing and Training Dataset

Now, the data is split in a random manner into the training and testing set. The training set contains 80% of the given data, while the testing set contains the remaining 20% of the provided data. After data splitting, corresponding feature and label vectors are created.

2.5 Feature Normalization

As the given data set contains features that vary highly in their ranges, it is necessary to normalize these. If features are not normalized and have different ranges, the result is in slow convergence of the Gradient Descent algorithm. To address this, all the features are normalized using Standardization. Feature normalization is done for all the training examples (both testing and training). Standardization basically replaces the values by their corresponding Z scores given by equation 2.3

$$x^{'} = \frac{x - \bar{x}}{\sigma} \tag{2.3}$$

The new features created have their mean $\mu = 0$ and standard deviation $\sigma = 1$.

3. Multivariate Linear Regression Model

Multivariate Linear regression represents a linear mathematical model for determining the value of one dependent variable from the values of given independent variables. The linear regression model for multivariate is given in equation 3.1

$$h_\theta(x) = \theta_0\, x_0 + \theta_1\, x_1 + \cdots + \theta_n\, x_n \tag{3.1}$$

Where $x_0 = 1$. In our case, the number of features (n) are 13. $h_\theta(x)$ is the dependent variable, and $x_1, \ldots, x_n$ are the independent variables. $\theta_0, \ldots, \theta_n$ are regression coefficients/parameters. We need to find the values of $\theta_0, \ldots, \theta_n$ that best fit our training set. This can be done using the Gradient Descent algorithm.

3.1 Gradient Descent Algorithm

Gradient Descent is an iterative optimization algorithm. The objective of linear regression is to minimize the cost function given by:

$$\min_{\theta_0,\cdots,\theta_n} \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2 \tag{3.2}$$

In gradient descent, we simultaneously update $\theta_0, \ldots, \theta_n$ for a specific number of iterations for every $j = 0, \ldots, n$

$$\theta_j \; = \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2 x_j^{(i)} \tag{3.3}$$

Definition of some related terms is as follows:

- Learning rate (α): It determines how fast and slow the gradient descent algorithm converges. I tested multiple learning rates (0.1, 0.01, 0.001) and settled on using 0.01.
- m = number of examples in the training set.
- Epochs: In one epoch, an entire dataset is passed through the model only once. I used 3000 epochs for training the model. After 3000 epochs, the cost function value was 0.031229

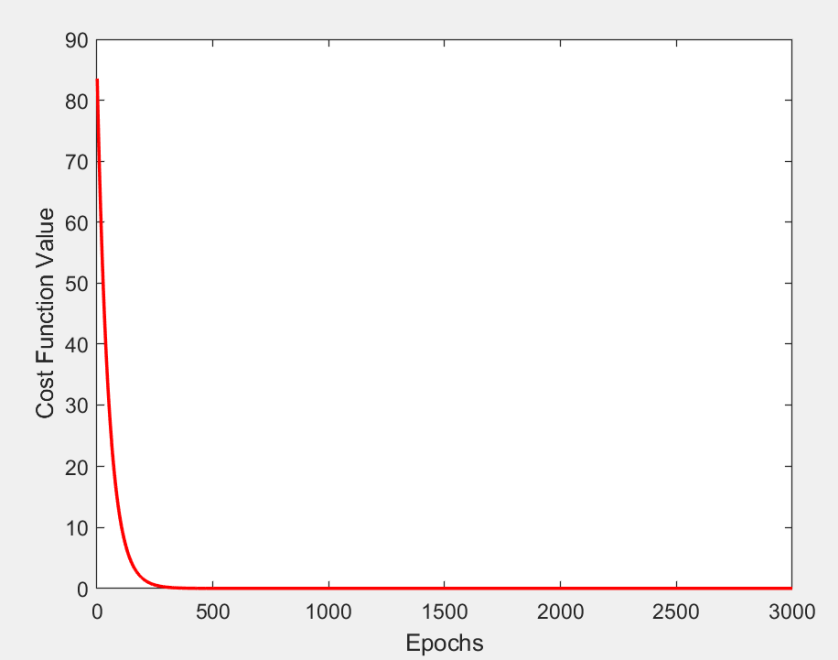
The gradient descent algorithm is implemented in *GD.m*, and the computation of cost function is being implemented in *calCost.m* The parameters obtained using gradient descent are shown in Table 3.1.

Table 3.1: Regression coefficients

θ_n	**Value**
θ_0	13.047217
θ_1	-0.010757
θ_2	0.033670
θ_3	0.174719
θ_4	0.032227
θ_5	-0.003825
θ_6	0.035045
θ_7	0.039553
θ_8	0.049751
θ_9	0.141792
θ_{10}	-0.029253
θ_{11}	0.091982
θ_{12}	0.017286
θ_{13}	-0.242666

Figure 3.1 shows the graph of Cost Function Values vs. Number of epochs. The graph also confirms that our cost function is decreasing with the number of epochs, which means that our algorithm is converging.

Figure 3.1: Cost Function Values vs. Number of epochs



3.2 Prediction

As required, the *prediction*() function is implemented in *prediction.m* file. The function takes two arguments as input (θ_n and features) and returns a vector containing predicted prices.

3.3 Model Evaluation Metrics

For a regression model, the following metrics can be used to determine the accuracy of the model [2]:

- **RMSE (Root Mean Square Error):** RMSE measures the average error performed by the model in predicting the outcome for an observation. The lower the RMSE, the better the model.

$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(x_i - \hat{x_i})} \tag{3.4}$$

- R^2 **value:** R^2 corresponds to the squared correlation between the actual outcome values and the predicted values by the model. The higher the R^2, the better the model. R^2 value range is [0, 1].

$$R^2 = 1 - \frac{\sum_{i=1}^{m}(x_i - \hat{x_i})}{\sum_{i=1}^{m}(x_i - \bar{x})} \tag{3.5}$$

  Here, x_i is the actual(observed), $\hat{x_i}$ is predicted output and $\bar{x}$ is the mean of actual(observed) output of the model.

Both the above evaluation metrics are implemented in file *modelEval.m*. The RMSE and R^2 values for training and testing set are shown in Table 3.2

Table 3.2: Model Evaluation Metrics

Dataset	**RMSE**	**R-Squared**
Training	0.249918	0.774634
Testing	0.249404	0.776507

4. Summary

So, I was able to achieve the R^2 value of around 0.77 and RMSE value of 0.24 for both the training set and the testing set.

Bibliography

[1] https://dev.to/rokaandy/logarithmic-transformation-in-linear-regression-models-why-when-3a7c, last accessed on 18 oct, 2019.

[2] https://stats.stackexchange.com/questions/142873/how-to-determine-the-accuracy-of-regression-which-measure-should-be-used, last accessed on 18 oct, 2019.

[3] https://www.slideshare.net/pawanshivhare1/predicting-king-county-house-prices, last accessed on 15 oct, 2019.