# A Project on optimization of profit and cost of land through Linear Regression.

Arunkumar Ramachandran
Student ID: 903928488

## 1. Abstract

This project focuses on implementing Linear regression to predict the profit or loss values that can occur in a restaurant business and focus on the cost of starting a new restaurant in a city. Assuming a scenario that you are the CEO of a restaurant franchise and are considering different cities for opening a new outlet. An estimate for profits and populations from the cities has been collected and has been saved in file ex1data1.txt.

The second part of the project focuses on how to implement this idea into the real world by indulging further into some of the necessary factors needed to start a restaurant, such as the expense of the restaurant and the size of the land in case we decide to start a restaurant. The size, expense, and the number of dining rooms data have been saved in ex2_data2.txt.

To achieve the goal of this project, we write a basic MATLAB code to implement the gradient descent method to minimize the cost function. We have to determine the profit and loss depending on the population in different cities using linear regression and then determine the cost of starting a restaurant by taking expenses, dining rooms, and the size of the plot into consideration. All values achieved from the code have been placed in the results at the end.

## 2. Introduction

Linear Regression is a supervised machine learning algorithm where the anticipated output is continuous and has a constant slope. It is used to predict values within a specific range (e.g., sales, price) rather than trying to classify them, such as cat, dog, etc. [1].

Linear regression can be used on either a univariate regression model or a multivariable regression model.

Univariate Linear regression, also known as simple linear regression, uses the slope-intercept form, which represents the equation of a line. The following intercept formula can be given as follows:

$$Y=mx+c \qquad (1)$$

A more complex, multivariable linear equation has the same form but more than one variable that the output function depends on. Equation 2 below is an example of a multivariate service for linear regression.

$$F(\theta_0, \theta_1, \theta_3) = \theta_0 x + \theta_1 y + \theta_3 \quad (2)$$

Where $\theta$ represents the coefficients or weights, a model will try to learn. And $\theta_0, \theta_1 and\ \theta_3$ are the input variables.

## 2.1. Cost Functions

Machine learning, specifically supervised learning, can be described as the desire to use available data to learn a function that best maps inputs to outputs. Optimization is one of the most significant parts of machine learning. Almost every machine learning algorithm has an optimization algorithm implemented in it. The function used for optimization is generally known as the cost function.

We can measure the accuracy of our hypothesis function by using a cost function ($\boldsymbol{J_\theta}$). This takes an average squared difference of all the results of the hypothesis with inputs from x's and the actual output y's. The formula for the cost function can be given as follow:

$$J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(y_i - y_1)^2 \quad (3)$$
$$= \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x_i) - y_i)^2$$

To break it up, the cost is $\frac{1}{2}\bar{x}$ where $\bar{x}$ is the mean of the squares of $h_\theta(x_i) - y_i$ , or in other words, it is the difference between the predicted value and the actual value.

## 2.2. Hypothesis/Prediction function:

Technically, this is a problem called function approximation, where we are approximating an unknown target function (that we assume exists) that can make the best map inputs to outputs on all possible observations from the problem domain. An example of a model that approximates the target function and performs mappings of inputs to outputs is called a hypothesis in machine learning.

The choice of algorithm (e.g., neural network) and the configuration of the algorithm (e.g., network topology and hyperparameters) define the space of possible hypotheses that the

model may represent. Equations 1 & 2 illustrated in the previous section are examples of hypothesis functions.

## 2.3. Gradient Descent Method

Gradient descent is a simple optimization procedure that you can use with many machine learning algorithms. Gradient descent algorithm is used to find the values of parameters (coefficients) of a function (f) that minimizes a cost function (cost). It is best used when the parameters cannot be calculated analytically (e.g., using linear algebra) and must be searched for by an optimization algorithm [3].

We have our hypothesis function, and we have a way of measuring how well it fits into the data. Now we need to estimate the parameters in the hypothesis function. That is where gradient descent comes in, as shown in equation (2).

Imagine that we graph our hypothesis function based on its fields $\theta_0$ and $\theta_1$ (we are graphing the cost function as a function of the parameter estimates). We are not graphing x and y itself, but the parameter range of our hypothesis function and the cost resulting from selecting a set of parameters.

We put $\theta_0$ on the x-axis and $\theta_1$ on the y-axis, with the cost function on the vertical z-axis. The points on our graph will be the result of the cost function using our hypothesis with those specific theta parameters. The graph below depicts such a setup.

Weight update:

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta} J(\theta_0, \theta_1) \tag{4}$$

Where j=0, 1,…,n represents the feature index number, and n is the number of weights.

At each iteration j, one should simultaneously update the parameters $\theta_1, \theta_2 \ldots . \theta_n$. Updating a specific parameter before calculating another one on the $j^{th}$ iteration would yield to the wrong implementation.

$$temp\ 0 := \theta_0 - \alpha \frac{\partial}{\partial \theta} J(\theta_0, \theta_1) \tag{5}$$

$$temp\ 1 := \theta_1 - \alpha \frac{\partial}{\partial \theta} J(\theta_0, \theta_1) \tag{6}$$

$$\theta_0 := temp0 \tag{7}$$

$$\theta_1 := temp1 \tag{8}$$

When specifically applied to the case of linear regression, a new form of the gradient descent equation can be derived. We can substitute our actual cost function and our actual hypothesis function.

Repeat the below algorithm until convergence:

$$\theta_0 := \theta_0 - \frac{\alpha}{m}\sum_{i=1}^{m}(h_\theta(x_i) - y_i) \tag{9}$$

$$\theta_1 := \theta_1 - \frac{\alpha}{m}\sum_{i=1}^{m}((h_\theta(x_i) - y_i)x_i) \tag{10}$$

where

- m is the size of the training set
- $\theta_0$ is a constant that will be changing simultaneously with $\theta_1$
- $x_i, y_i$ are values of the given training set (data).

## 2.4. Feature Normalization

Most of the time, your dataset will contain features highly varying in magnitudes, units, and range. But since most of the machine learning algorithms use Euclidean distance between two data points in their computations, this is a problem. The feature normalization is used here to subtract the mean value of each feature from the data set. After subtracting the mean, we divide the feature values by their respective "standard deviations."

The formula can be given as follows:

$$\mathrm{X}=\frac{x_i-\mu_i}{n} \tag{11}$$

Where;

- $x_i$= size of the given set of parameters
- $\mu_i$= Average value of the given parameters
- $n$= Range of the given parameter

The range of all features should be normalized so that each feature contributes approximately proportionately to the final distance. Another reason why feature scaling is applied is that gradient descent converges much faster with feature scaling than without it.

## 2.5. Normal Equations:

The linear regression closed form can be given as follows:

$$\theta = (X^T X)^{-1} X^T y \tag{12}$$

Using this formula does not require any feature scaling, and you will get an exact solution in one calculation: there is no “loop until convergence" like in gradient descent.

## 2.6. Data Set

The data sets used in this project have been saved in the text file ex1data1 for the Part1 of this project and ex1_data2 for the second part.

## 2.7. Part1 dataset

The data for part1 contains information collected about the profit or loss a restaurant business is making in a city to populations in different cities. The data set is of size 97x2 in which the first column represents the population of a city, and the second column represents profit or loss. The following table is an illustration of data in ex1data1. Each row in the data represents the population and profit relating to a single city.

| **Population** | **Profit/Loss** |
|---|---|
| 61101 | 175920 |
| 55277 | 91302 |
| 85186 | 136620 |
| 70032 | 118540 |
| 58598 | 68233 |
| .<br>.<br>.<br>83829 | .<br>.<br>.<br>118860 |

Figure 1 illustrates the population versus profit values. The negative values under the profit column indicate loss occurred with respect to the population.

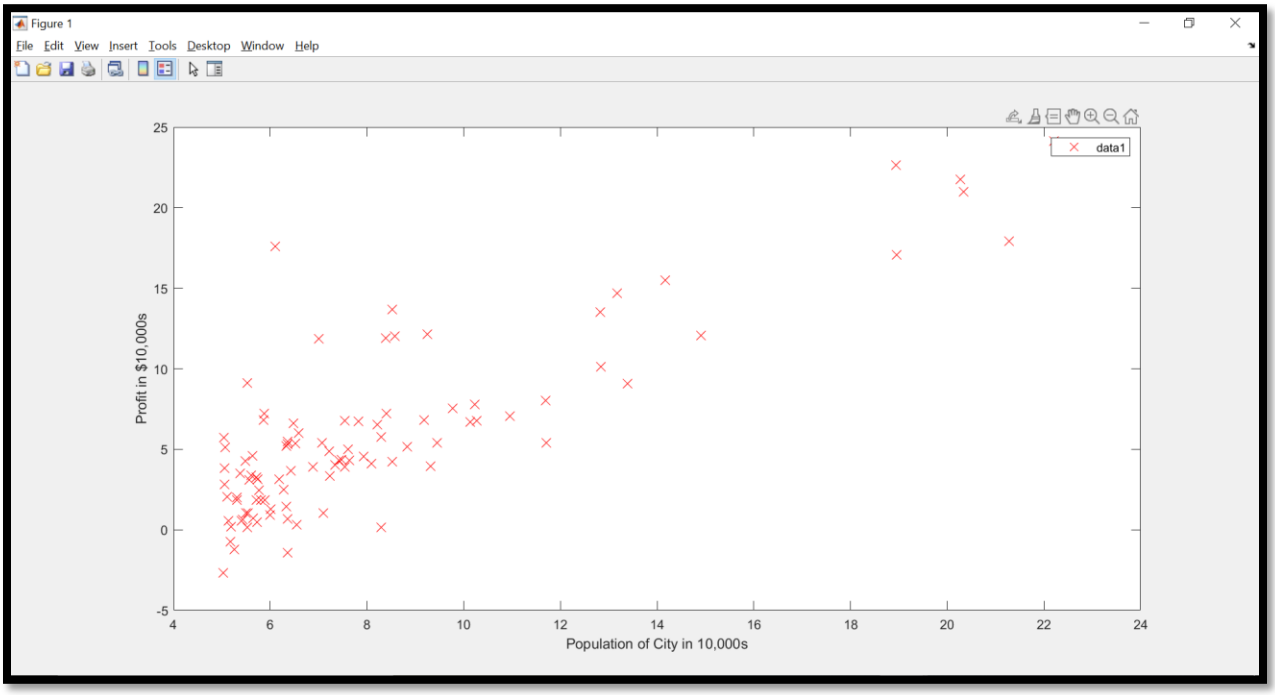


*Figure 1: Profit vs. Population Graph*

## 2.8. Part 2 data set

The data for part2 contains information collected about the area of the restaurant in square feet, the number of dining rooms, and the total cost of land. The data set is of size 47x3 in which the first column represents the usable area of the restaurant in square feet, and the second column represents the number of dining rooms available, and the final column represents the total cost of the land. The dataset consists of data from 47 different locations, and every row in it represents a new location.

The dataset for part 2 is illustrated as follows:

| **Size or area in sq feet** | **Number of dining rooms** | **The total cost of the land** |
|---|---|---|
| 2104 | 3 | 399900 |
| 1600 | 3 | 329900 |
| 2400 | 3 | 369000 |
| 1416 | 3 | 232000 |
| 3000 | 4 | 539900 |
| 1985 | 4 | 299900 |

| | | |
|---|---|---|
| 1534 | 3 | 314900 |
| .<br>.<br>.<br>1203 | .<br>.<br>.<br>3 | .<br>.<br>.<br>239500 |

# 3. PART 1: Linear Regression to determine profit or loss with respect to population

To begin with the training for part 1, we first need to define our hypothesis function to calculate the cost. For this part, we use the only population as our input variable; the hypothesis function used is one that of univariate linear regression. The hypothesis function used is as shown below:

$$h_\theta = \theta^T x \qquad (13)$$

Where.

- $h_\theta$= Hypothesis parameter
- $\theta^T$= Transpose of theta

Once the dataset is loaded, we can compute the error between the predicted values calculated using the hypothesis above (Equation 13), and the known outputs from the database. Optimization of parameters using gradient descent is then performed, and weights are calculated until the cost function settles down.

## 3.1. Code outline

The part 1 of this project has been performed in Complete_Code.m. Initially, we start the program by calling in ex1data1.txt file and load the data set. We do this to plot the data set, which contains values for both population and profit. Once the graph is plotted, we initialize the values for the gradient descent settings by giving the number of iteration values as 1500 so that the program will run for 1500 iterations, and we also provide a starting value for alpha as 0.01. We then run the gradient function by calling theta. This is done so that it will give us the optimal theta values once the gradient descent is computed. At the end of the program, we predict the profit values obtained for a population of 35000 and 70000 along with the optimal theta values.

# 4. PART 2: Linear Regression using multivariable to determine the cost with respect to size, dining room and expense to start a restaurant

Linear regression in one variable has been implemented in part 1 to determine the estimated profit or loss in that particular city. As for the second part, other factors come into the picture. Some of them might include the following:

1. The size of the Land.
2. The number of dining rooms.
3. The expense of the land.

To implement these values, multivariable regression can be used, and variables can be assigned to these parameters mentioned above. Some of the key factors to be calculated are as follows:

## 4.1. Feature Normalization:

The equation for feature normalization is mentioned in (*Equation11*). The standard deviation is a way of measuring how much variation there is in the range of values of the particular feature (most data points lie within +/- 2 standard deviations of the mean). This is an alternative to taking the range of values (max-min). For example, inside the code featureNormalize.m, the quantity X(:,1)contains all the values of x1 (size of the land) in the training set, so std(X(:,1)) computes the standard deviation of the size of the land. At the time, featureNormalize.m is called, the extra column of 1's corresponding to $x_0 = 1$ has not yet been added to X.

To begin with the training for part 2, we first need to define our hypothesis function to calculate the cost. The hypothesis function and its parameters can be seen in (*Equation 12*).

Once the dataset is loaded, we can compute the error between the predicted values calculated using the hypothesis above in the cost equation *(formula mentioned in Equation 3)*, and the known outputs from the database. Optimization of parameters using gradient descent is then performed, and weights are calculated until the cost function settles down.

## 4.2. Code outline

Part 2 of this project has been performed in Complete_Code_Multi_Variable.m. Initially, we start the program by calling in ex1_data2.txt file and load the data set. We then pick 10 example values from the data set and normalize the feature. Then we have to run the gradient descent method, and to do so, we take the alpha value as 0.01, and the number of iterations as 400 and the program will then run for 400 iterations. We then run the gradient function by calling theta. This is done so that it will give us the optimal theta values once the gradient descent is computed. At the end of the program, we have to enter values for the

size of the land by picking values from the ex1_data2.txt file. By doing this, we are trying to compare the predicted expense value with the actual expense value, whether they are the same, smaller, or larger than each other.

## 5. Result

### PART 1:

The results obtained are as follows:

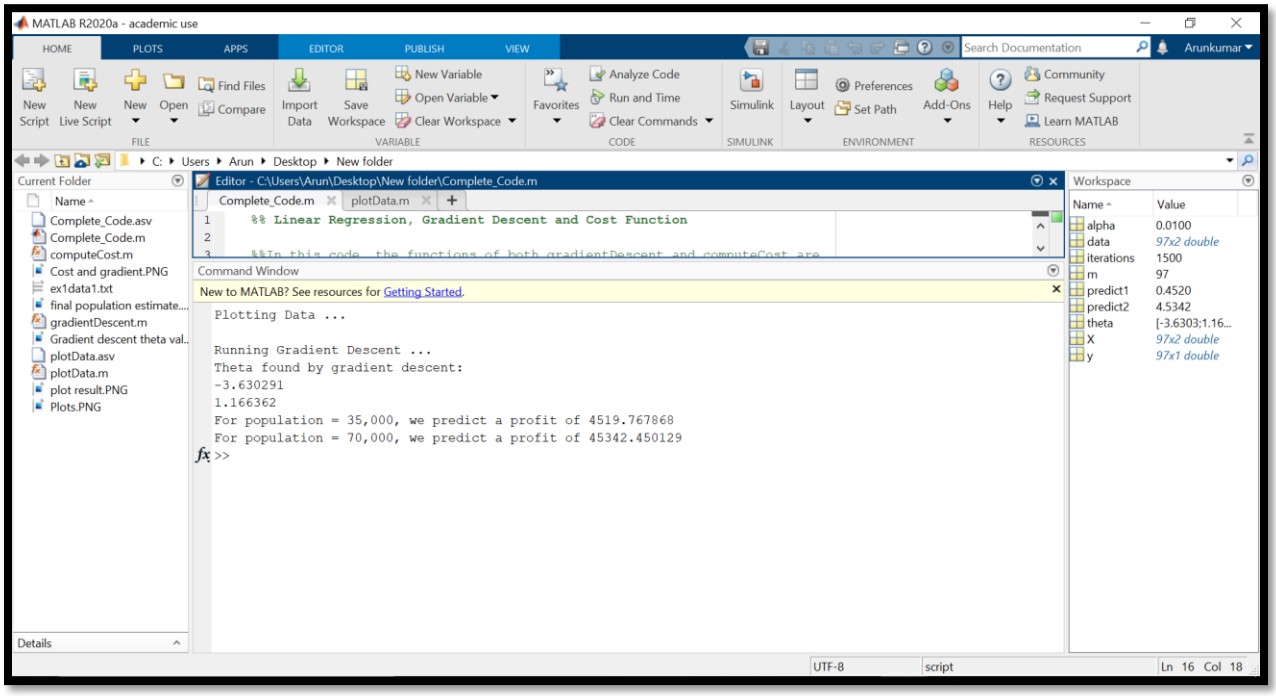


*Figure 2 Optimal Theta Values and Predicted Profit values obtained for populations 35000 and 75000.*

As shown in the figure above, the optimal theta values are -3.63 and 1.166. For a population of 35000, the predicted profit is 4519.76, and for a population of 70000, the predicted profit is 45342.45.

The final plot with linear regression is shown below:

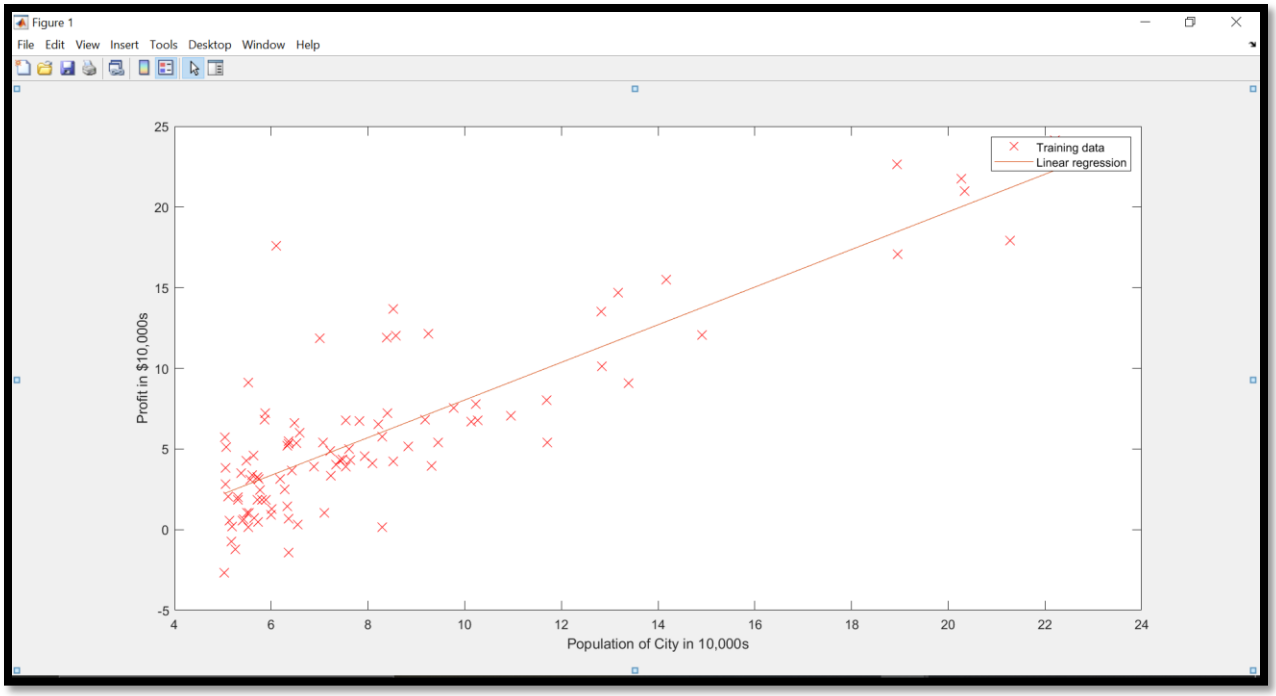


*Figure 3 Profit vs. Population Graph along with linear regression.*

## PART2:

The results obtained are as follows:

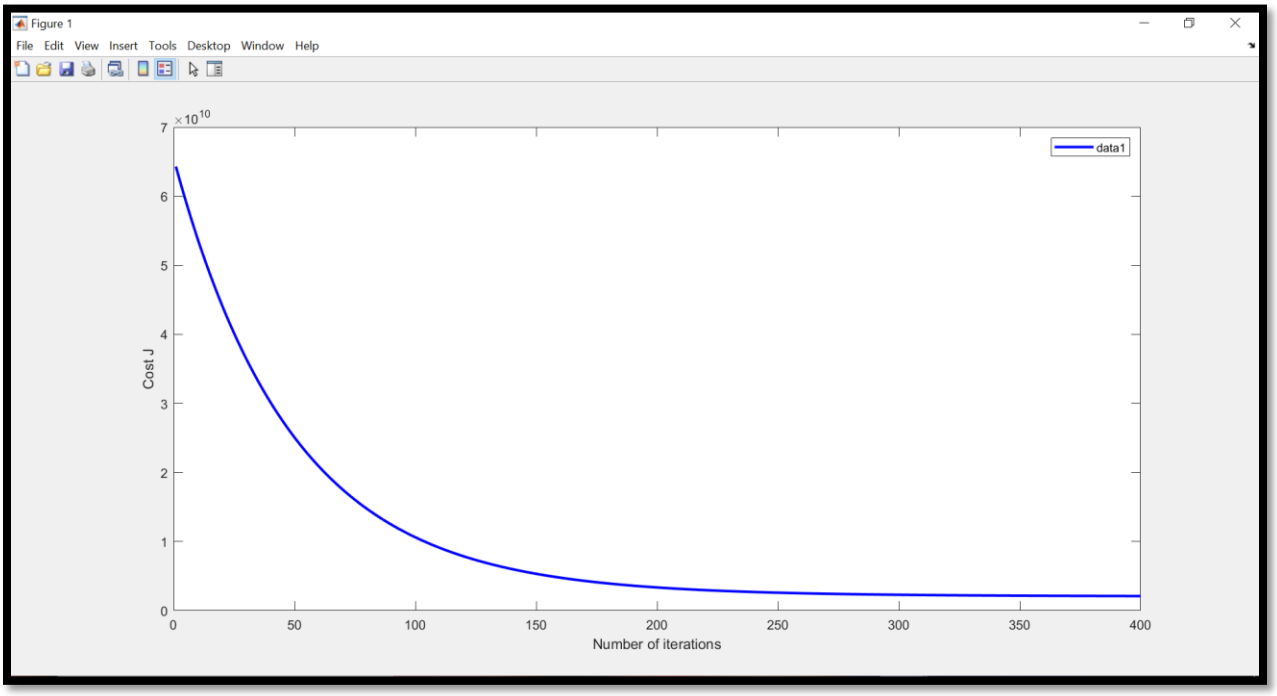


*Figure 4 Cost function graph*

From the above graph, we can see that the cost function decreases, and at a certain point, it becomes constant throughout, though the cost values are still high. Another prominent

thing to note is that the predicted expense and the expense value do not necessarily have to be almost the same. This is because we have used the linear regression and not polynomial regression to plot the graph.

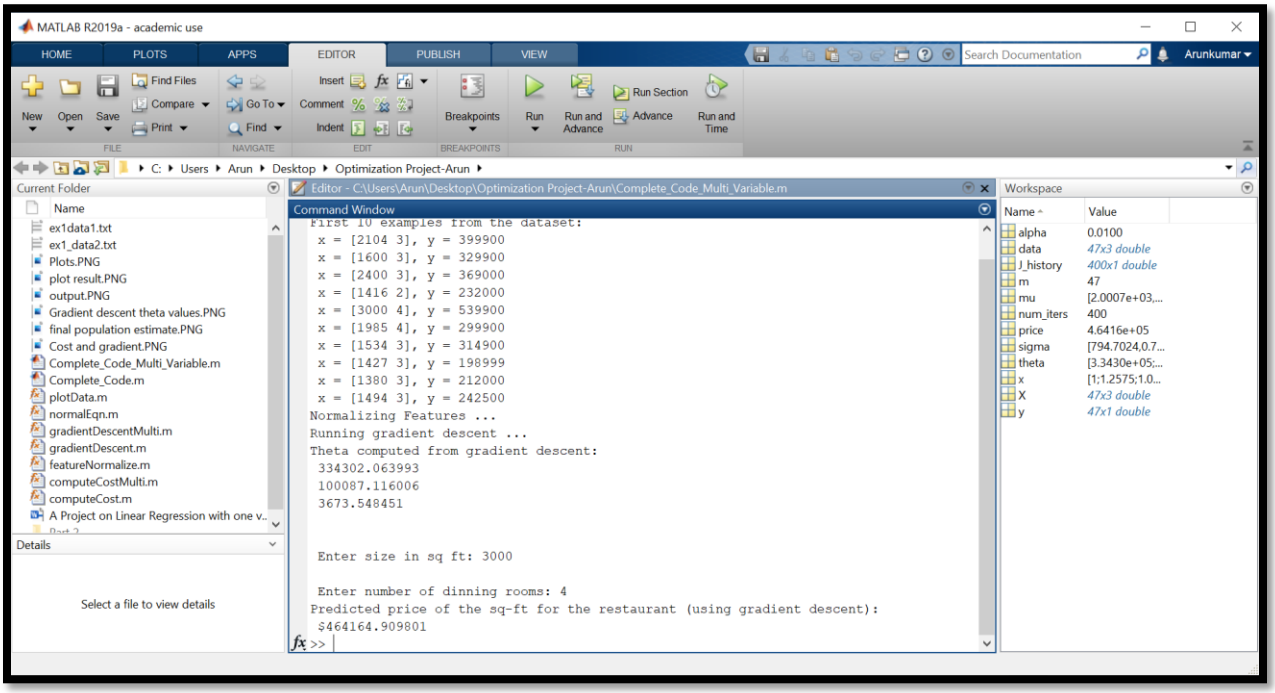


*Figure 5 Optimal Theta Values obtained and Predicted Profit values obtained according to the number of dining rooms and the cost of the land.*

From the above figure, we can see the size of the plot in sq ft (value taken from the ex2_data2.txt) and is tested. We can see that the predicted value is 464164.909801, and the actual value is 539900. The computed theta values have also been shown in the figure. Another trial output is shown in the figure below:

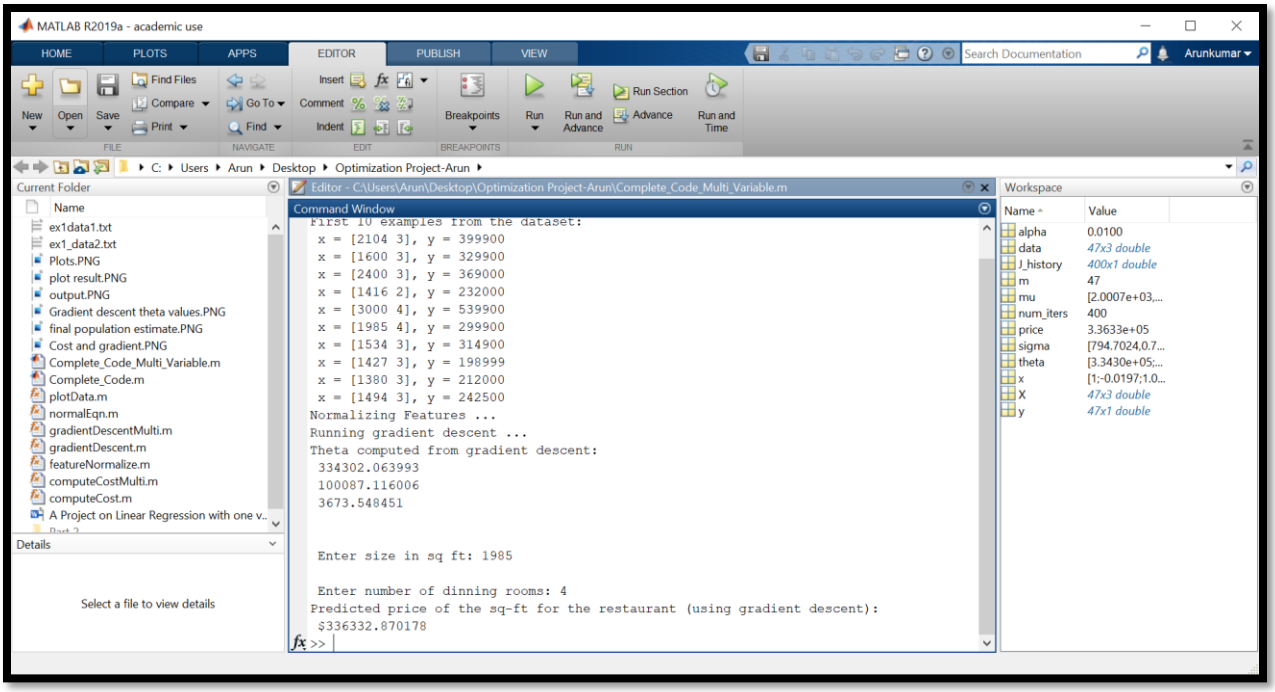


*Figure 6 Optimal Theta Values obtained and Predicted Profit values obtained according to the number of dining rooms and the cost of the land.*

From the above figure, we can see that the predicted value of the cost is 336332.870178, while the actual value is 299900.

## 6. Conclusion

From PART 1, we have deduced the predicted profit or loss with respect to the population in a city. This was done by using the gradient descent method to optimize the cost function to determine for a certain number of populations, how much will be the profit or loss. The output can be seen in *(Figure 2)* and *(Figure 3).*

From PART 2, we try to implement the idea in part 1 further by assuming that, in real life, we decide to buy a land and start a restaurant business. To do this, we take some features such as the expense of plot, number of dining rooms, and size of the plot into consideration. We also implement gradient descent here to minimize the cost function and use feature normalizing method to normalize the range of independent variables. The output can be seen in *(Figure 4), (Figure 5)* and *(Figure 6).*

# Reference


- [1]Xiaoharper. (n.d.). Normalize Data - ML Studio (classic) - Azure. Retrieved June 17, 2020, from https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/normalize-data.
- [2]Linear Regression¶. (n.d.). Retrieved June 17, 2020, from https://ml-cheatsheet.readthedocs.io/en/latest/linear_regression.html
- [3]Brownlee, J. (2019, August 12). Gradient Descent For Machine Learning. Retrieved June 17, 2020, from https://machinelearningmastery.com/gradient-descent-for-machine-learning/