A PROJECT REPORT

ON

Building Linear Regression Using 3 Different Methods

BY

Kumar Pranjal (2018A7PS0163H)

Sneh Lohia (2018A7PS0171H)

Abhishek Mishra (2018A7PS0019H)

Under The Supervision of

Dr. NL Bhanu Murthy

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

HYDERABAD CAMPUS

(OCTOBER, 2020)

Assignment 2: Building Linear Regression Using 3 Different Methods

Introduction: In Statistics, Linear regression is a linear approach to modeling the relationship between a scalar response and one or more explanatory variables. When more than one explanatory variables are involved, the process is called Multiple Linear Regression.

We were given a Dataset called "Insurance.txt." The file consisted of 3 independent variables, namely Age, BMI, and Children. One dependent variable is "Charges." We were asked to build a multivariable regression model using the dataset and predict the charges. Twenty linear regression models are made using three different methods, and the error associated with each algorithm is calculated.

Data Pre-Processing

Mentioned below are the steps involved before building the Model. They are:

- **Data Pre-Processing:** Data Preprocessing is a technique that transforms the given raw data into a clean dataset. This step improves the accuracy of the model. Data normalization has been done for this assignment.
- **Normalization:** This step is to make all the elements lie between 0 and 1, thus bringing all the values of numeric columns in the dataset to a standard scale.

We have used the normalize_dataset function, which normalizes the dataset by using the formula-

$$X_{normalized} = \frac{X - min(X_i)}{max(X_i) - min(X_i)}$$

Functions to calculate Error and Accuracy

We have used the Mean Square Error Method to calculate the Error -

$$MSE = \frac{\sum_{i=1}^{m} (y_i - \hat{y}_i)^2}{2m}$$

To find Accuracy, we have used the R^2 metric -

$$R^2 = 1 - \frac{2 \times MSE}{Var(Y)}$$

Here the value of R^2 can vary from negative infinity to +1 based upon the performance of the model. 0 R^2 means the model is predicting precisely like the mean line. Positive or negative values signify better or worse predictions, respectively.

Steps Used In Building The Model

The data was read from the "Insurance.txt" file. The random shuffling of the data is done each time a regression model is made using the random shuffle function. After random shuffling, the entire dataset is divided into two parts, training set, and testing set. The training set consists of 70% of the dataset, and the testing set consists of the remaining 30% of the dataset. The three different algorithms, the Normal Equations method, the gradient descent method, and Stochastic gradient descent method are run for each of the 20 regression models, and errors are calculated.

Normal Equation Model

The formula used to calculate the Weights using the normal equations method is :

$$W = (X^T X)^{-1}.(X^T Y)$$

Twenty models are created with different train-test splits and mean and variance cost and these models' accuracy is found out.

Mean and Variance Cost of The Training Model -

Mean train cost = 0.01641183708883813

Variance train cost = 2.954623951256763e-07

Mean test cost = 0.016534668300305956

Variance test cost = 1.6278320443356039e-06

Mean and Variance Accuracy of The Training Model -

Mean train accuracy = 0.12219135236608396

Variance train accuracy = 0.00016501768175132512

Mean test accuracy = 0.10995705915849718

Variance test accuracy = 0.0009497830994676786

Visualizing the results

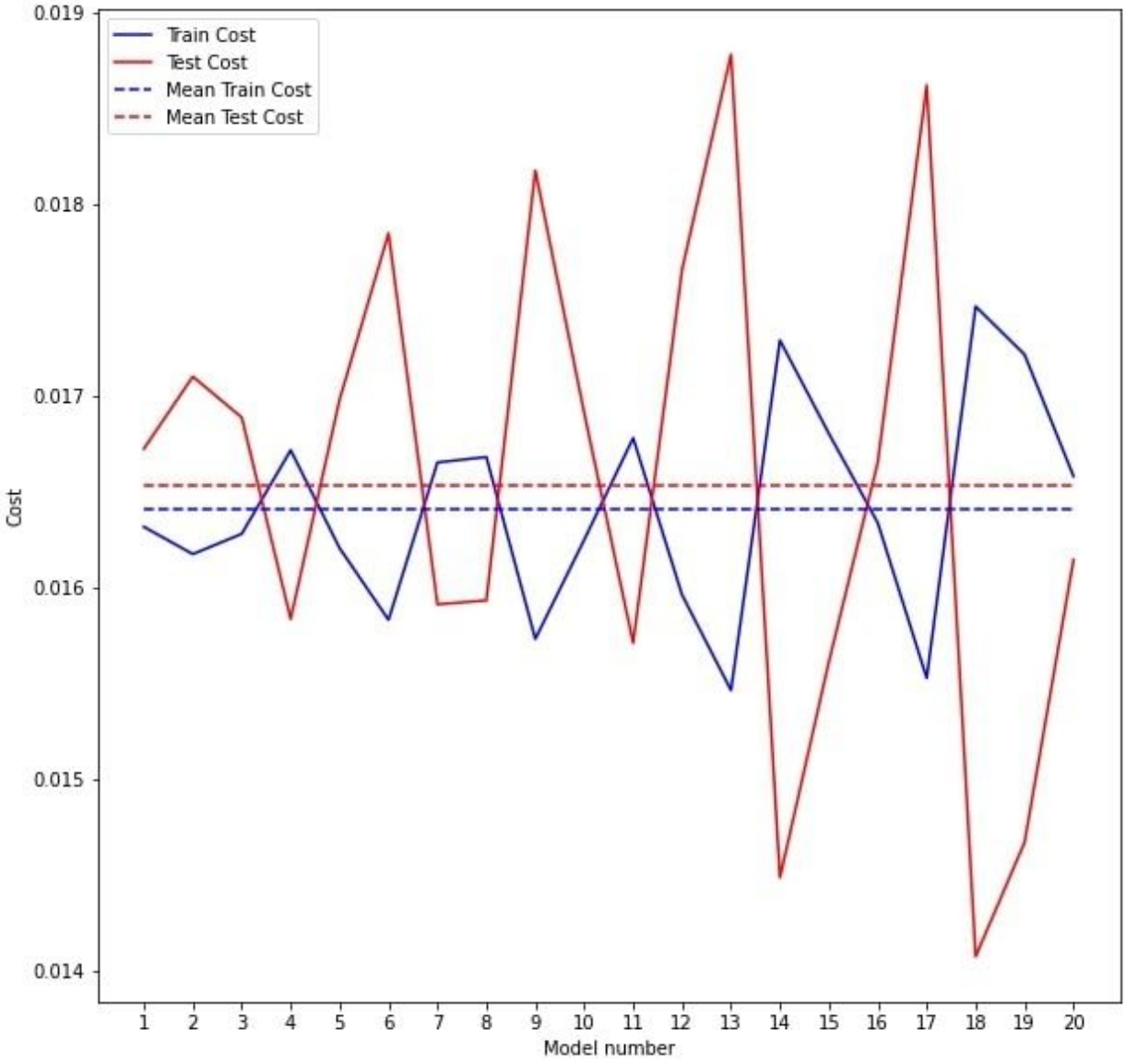

Fig. Cost vs. model number plot

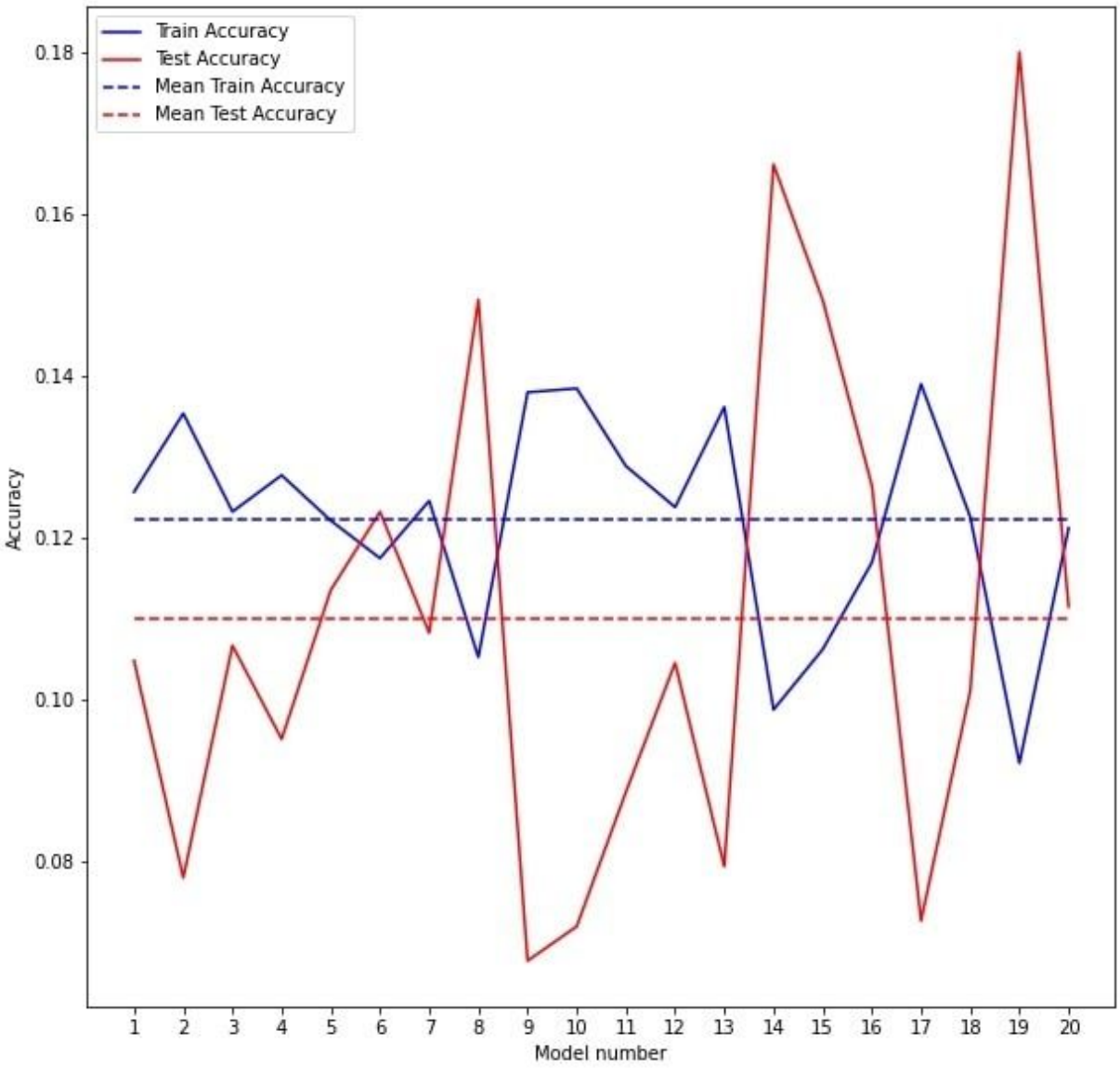

Fig. Accuracy vs. model number plot

Gradient Descent model

Function for finding W using Gradient Descent :

$$dW = (1 / m) * (X^T (XW - y))$$
$$W = W - (eta * dW)$$

The choice of learning rate is crucial for gradient and stochastic gradient descent methods. With a very high learning rate, the function may fail to converge and overshoot the minimum. If the learning rate is very low, the process takes a very long time to converge, which is computationally expensive. The Gradient Descent method was run for three different learning rates. For each learning rate, 20 models with varying train-test splits were trained, and the results were analyzed. The learning rates and the Means and Variance of the model corresponding to them are as follows:

For Learning Rate = 0.001

Mean train cost = 0.016836577590157668
Variance train cost = 3.3400611275115553e-07
Mean test cost = 0.016616375913121158
Variance test cost = 1.9355447109076412e-06

Mean train accuracy = 0.10302810973111243
Variance train accuracy = 7.015727427438704e-05
Mean test accuracy = 0.09522194315652213
Variance test accuracy = 0.00026095822121710436

For Learning Rate = 0.01

Mean train cost = 0.016273019855530348
Variance train cost = 4.2891362338108696e-07
Mean test cost = 0.016877137895295668
Variance test cost = 2.393433058049509e-06

Mean train accuracy = 0.12236687401645611
Variance train accuracy = 9.934865995402785e-05
Mean test accuracy = 0.10451157512906897
Variance test accuracy = 0.0007659438732698235

For Learning Rate = 0.1

Mean train cost = 0.01631007000390367
Variance train cost = 3.4219392811674187e-07
Mean test cost = 0.016780083462784624
Variance test cost = 1.905854572901112e-06

Mean train accuracy = 0.12190824317151625
Variance train accuracy = 0.00011485501258505248
Mean test accuracy = 0.1083302923084678
Variance test accuracy = 0.0007542521961161162278

Visualizing the results

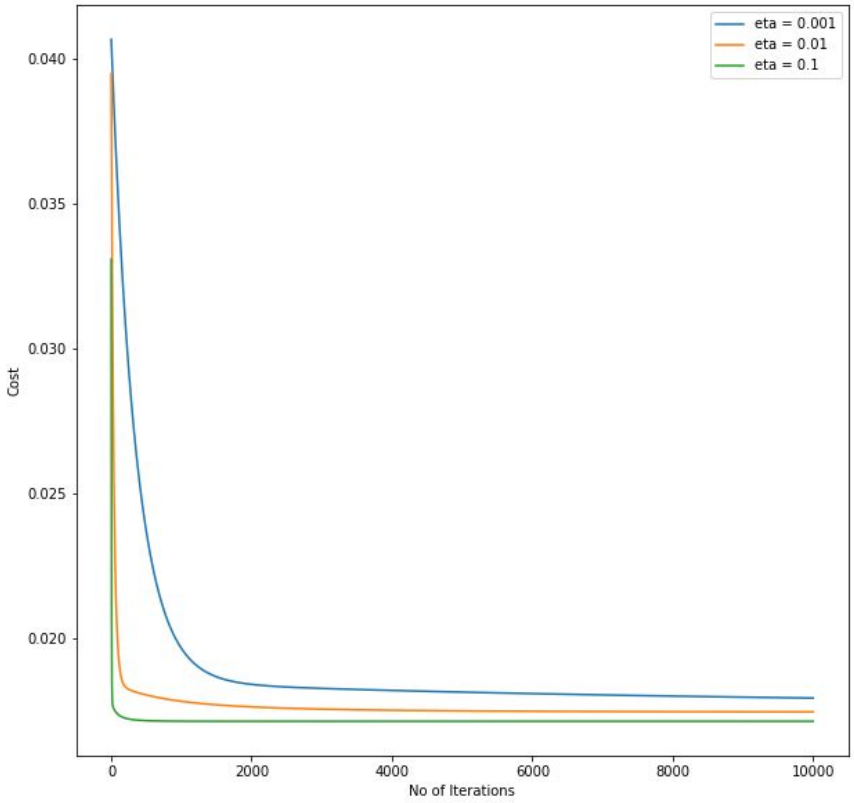

Fig. Cost vs. iterations plot

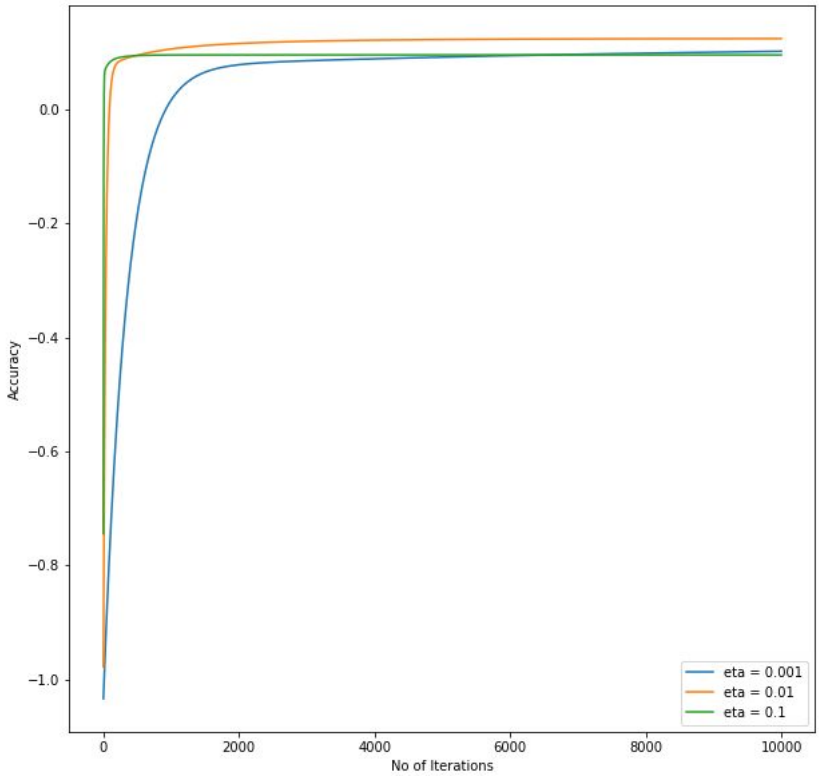

Fig. Accuracy vs. iterations plot

Stochastic Gradient Descent model

SGD picks up a "random" instance of training data at each step and then computes the gradient making it much faster as there are much less data to manipulate at a single time, unlike GD, which picks up the whole batch at once.

Function for finding W using Stochastic Gradient Descent :

$$dW = (1 / m) * (X^T (XW - y))$$
$$W = W - (eta * dW)$$

Stochastic Gradient Descent method was also run for three different learning rates. For each learning rate, 20 models with varying train-test splits were trained, and the results were analyzed. The learning rates and the Means and Variance of the model corresponding to them are as follows:

For Learning Rate = 0.001

Mean train cost = 0.036900632695602106
Variance train cost = 1.6183324262606891e-06
Mean test cost = 0.03677125798583876
Variance test cost = 9.033300099661697e-06

Mean train accuracy = -0.9732743802560485
Variance train accuracy = 0.00041453594166937356
Mean test accuracy = -0.983218370821066
Variance test accuracy = 0.0024243074284995897

For Learning Rate = 0.01

Mean train cost = 0.03196836992290407
Variance train cost = 1.5898673732922881e-06
Mean test cost = 0.03259826825487746
Variance test cost = 1.1058829027842569e-05

Mean train accuracy = -0.7262216136762171
Variance train accuracy = 0.00030892619401280246
Mean test accuracy = -0.7188818192063091
Variance test accuracy = 0.0026910849084654563

For Learning Rate = 0.1

Mean train cost = 0.01831875222340684
Variance train cost = 2.2111382595572255e-07
Mean test cost = 0.01788649538083934
Variance test cost = 1.7039086595185964e-06

Mean train accuracy = 0.026480658447209936
Variance train accuracy = 3.261753572239664e-05
Mean test accuracy = 0.022600187186150127
Variance test accuracy = 0.0003334953061849545

Visualizing the results

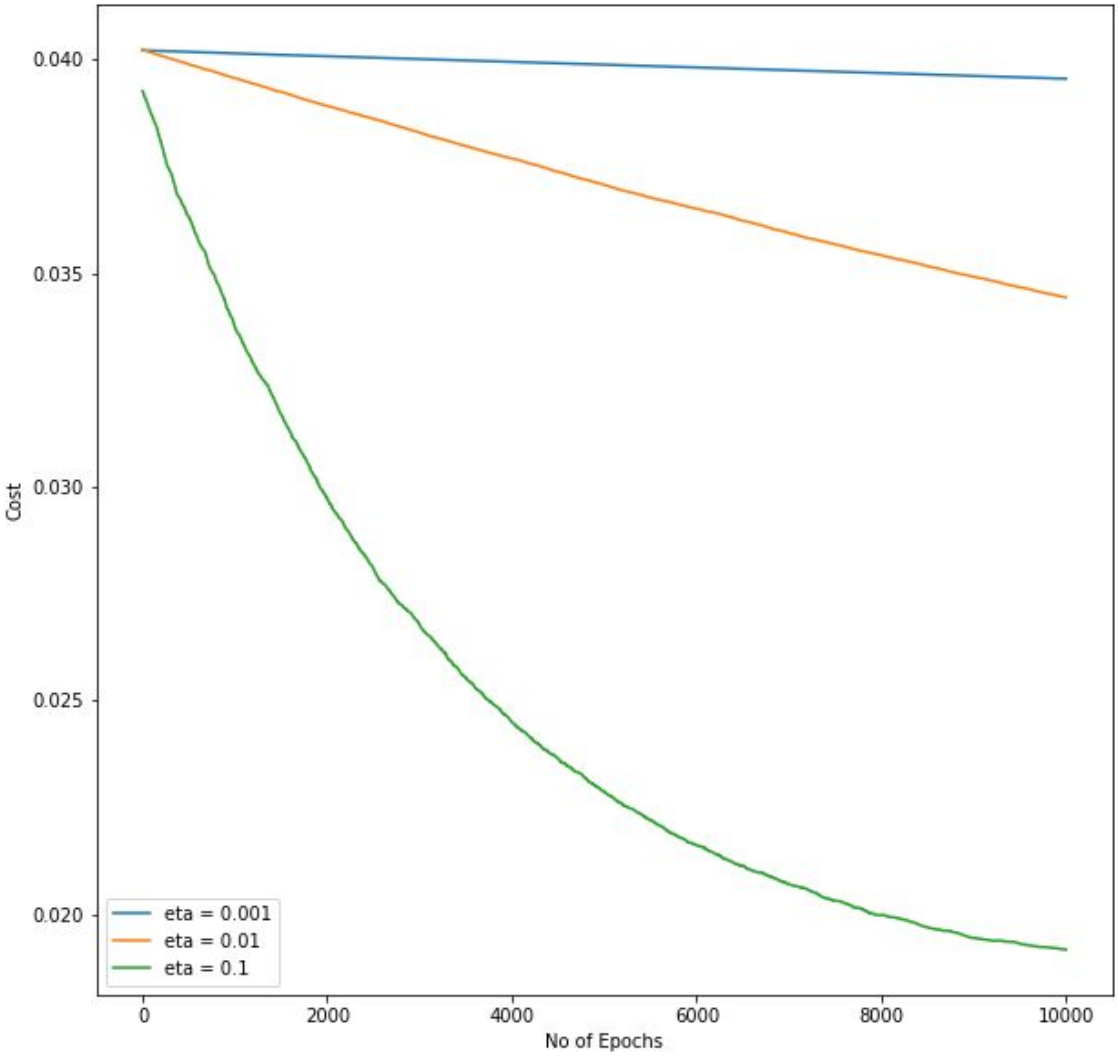

Fig. Cost vs. epochs plot

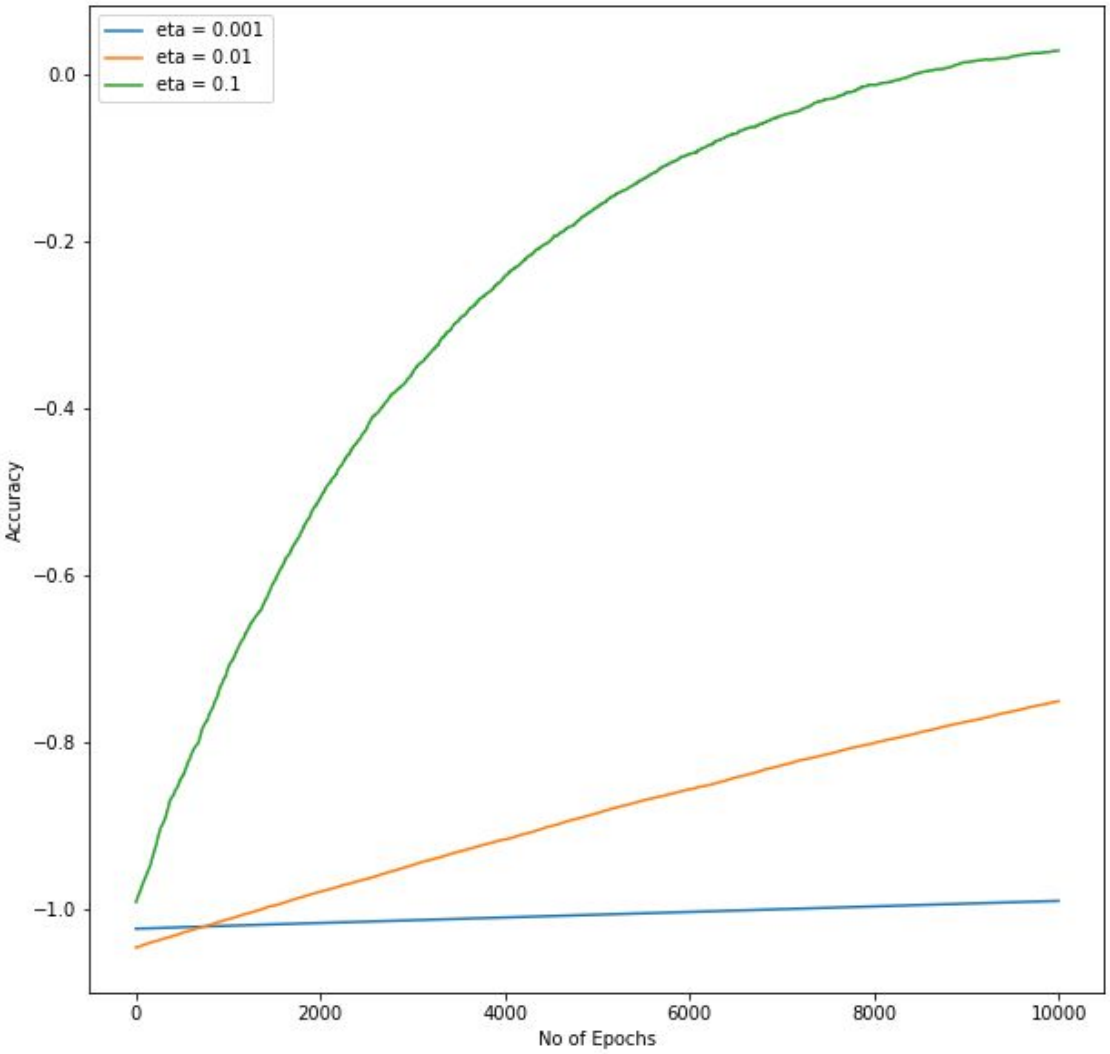

Fig. Accuracy vs. epochs plot

Comparison between GD and SGD

Cost comparison

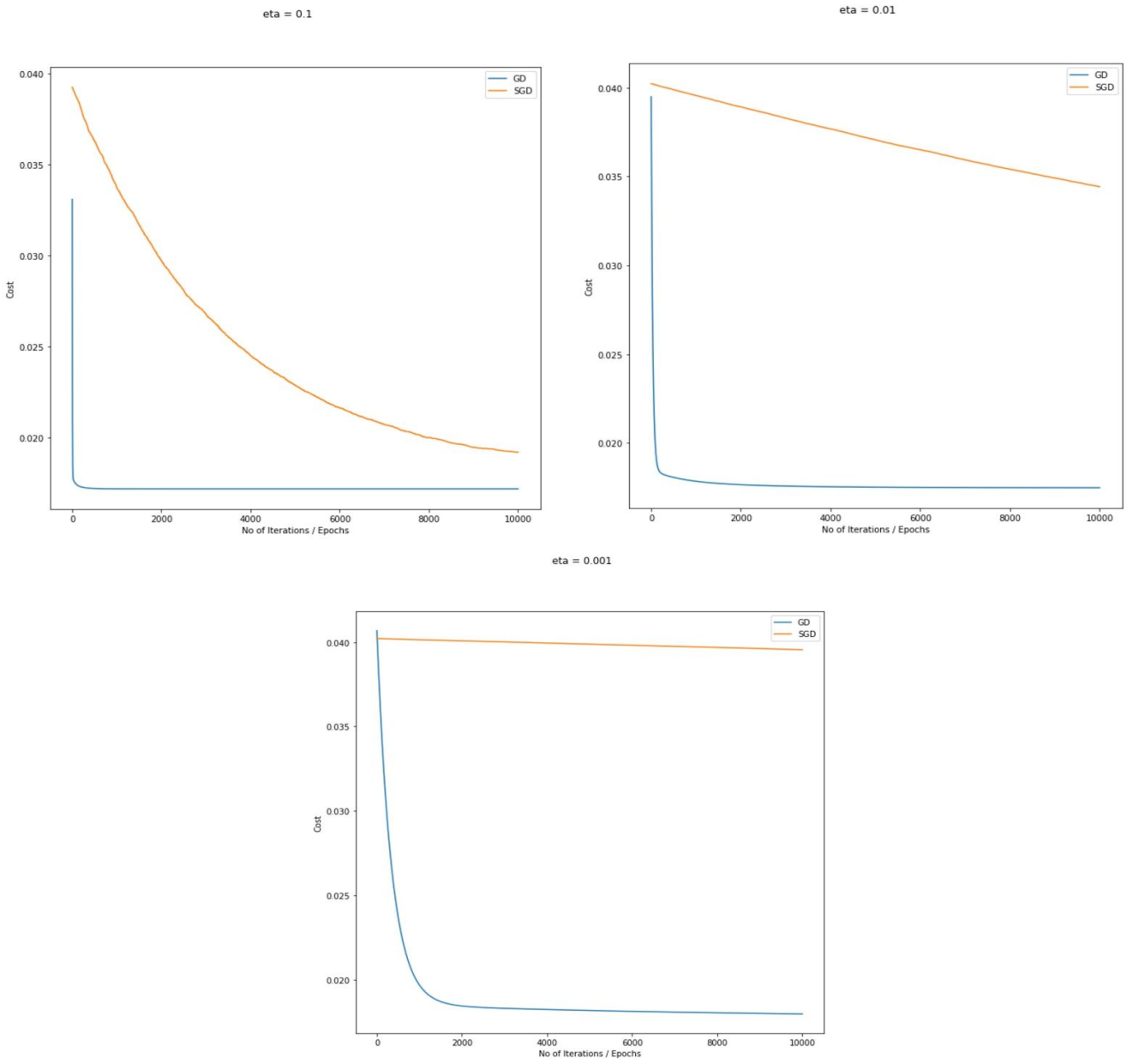

Fig - Cost vs Epochs Plots for Eta = 0.001, 0.01 and 0.1

Accuracy comparison

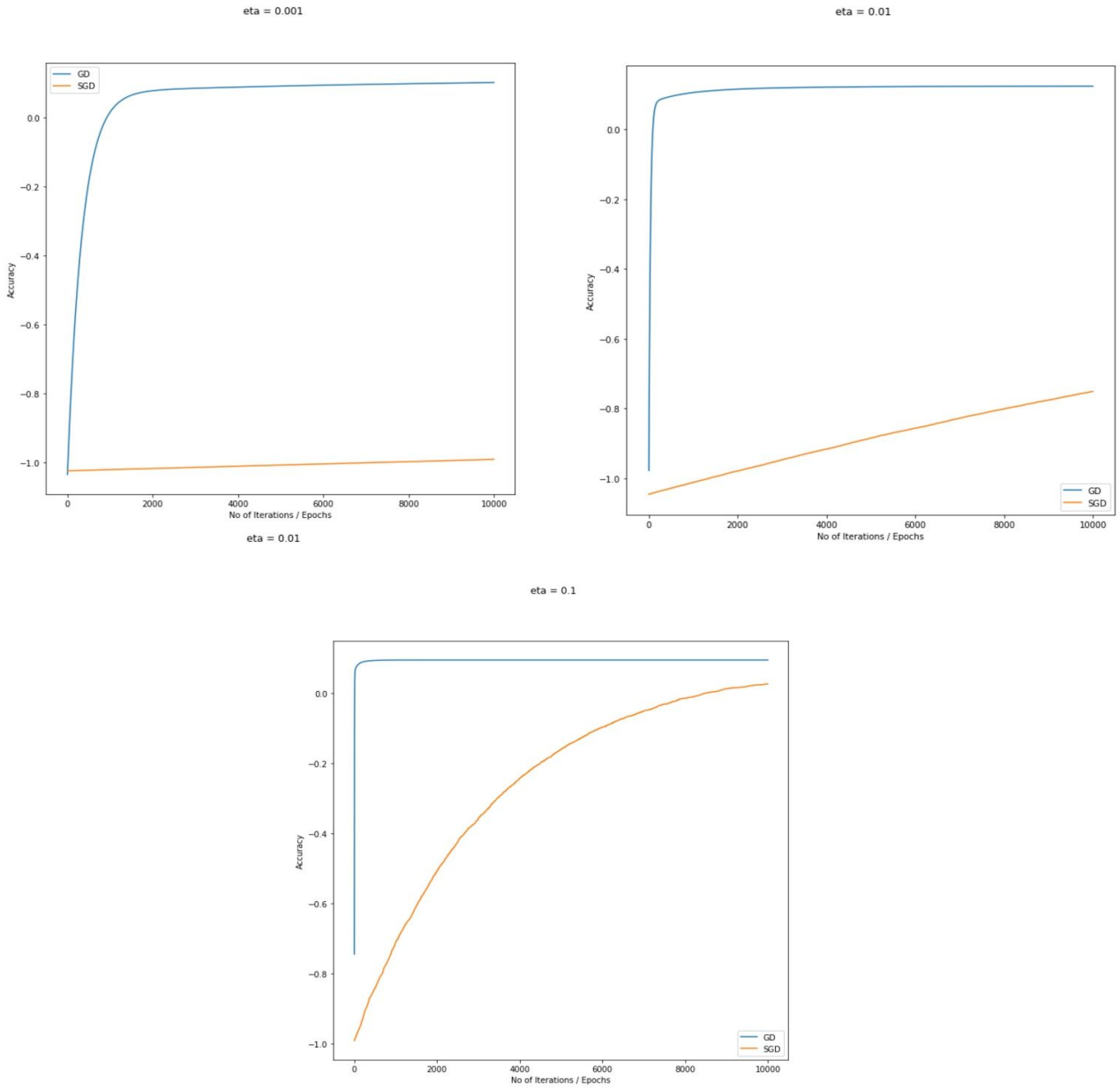

Fig - Accuracy vs Epochs Plots for Eta = 0.001, 0.01 and 0.1

Other Differences

Cost and accuracy graphs for GD are smooth while they aren't for SGD. It is because all the training points are taken at once in GD, which helps to move to global minima. In SGD, only one point is taken at a time, and its minima may not be equal to global minima, and hence cost may increase sometimes.

Questions to Ponder On

Q1. Do all three methods give the same/similar results? If yes, Why? Which method, out of the three would be most efficient while working with real-world data?

No, all three methods do not give the same result. The most accurate result is shown by the method of the Normal equations with the least error as visible from the code. Then the Gradient Descent method calculates the weights very close to the method of the Normal equation, but if the learning rate is significantly less, it becomes computationally costly. The stochastic gradient method calculates the weights with a little more deviation from the actual weights, as seen from the mean and variance calculated. But this method is computationally better compared to the Gradient Descent method.

If the dataset's size is enormous, it is better to use the Stochastic Gradient Descent method. If the dataset's size is tiny and accurate results are expected, the normal equation method can be used. For a medium-sized dataset, Gradient descent works the best.

Q2) How does normalization/standardization help in working with the data?

Data preprocessing is essential because it allows improving the quality of the raw experimental data. The primary aim of preprocessing is to minimize or, eventually, eliminate those small data contributions associated with the experimental error. Normalization scales the data to a comparable scale, and hence it also makes Gradient Descent and Stochastic Gradient Descent faster.

Q3) Does increasing the number of training iterations affect the loss? What happens to the loss after a very large number of epochs (say, ~10^{10})

As the number of Training iterations increases, The Cost value decreases. But for very large values of epochs (~10^{10}), the loss almost saturates, and the line becomes almost parallel to the x-axis.

Q4) What primary difference did you find between the plots of Gradient Descent and Stochastic Gradient Descent?

It was observed that for the same learning rates for both gradient descent and stochastic gradient descent, the gradient descent converge faster compared to the stochastic gradient method. Also, the cost plot for Gradient Descent was smooth, but it wasn't for Stochastic Gradient Descent, as explained earlier.

Q5) What would have happened if a very large value (2 or higher) were to be used for the learning rate in GD/SGD?

Very large values of the learning rate in both GD and SGD can cause the function to overshoot the minimum, and the process may fail to converge. A very high learning rate causes the cost to become infinite, and so, the graph shows infinite value in the y-axis.

Q6) Would the minima (minimum error achieved) have changed if the bias term (w0) were not to be used, i.e., the prediction is given as $Y = W^T X$ instead of $Y = W^T X + B$

The minimum error will change if the bias term was removed from the equation. Without the bias term, the effect of the bias term would add to the total error of the model, and hence the minimum error obtained will be higher as compared to the model where bias was used.

Q7) What does the weight vector after training signify? Which feature, according to you, has the maximum influence on the target value (insurance amount)? Which feature has the minimum influence?

The weights vector tells us the influence of a particular factor (independent variable) on the result (dependent variable). For example, taking the linear relation to be y = 1 + 2x1 + 5x2. A high value of 5 associated with independent variable x2 tells that this term contributes more to the final value y than the weight 2 of x1.

In our model, it was observed that the Age was the most important variable, and then the BMI in determining the changes. The Number of Children caused the minimum influence.