



FOUNDATIONS OF DATA SCIENCE - A2 REPORT

Course ID: CS F320



Group members:

Parth Krishna Sharma: 2017B3A70907H

Siddhi Burse: 2017B3A70972H

Maurya Grover: 2017B3A71433H

OCTOBER 27, 2020

Table of Contents

A) Overview:.....	2
B) Description of Data:.....	2
C) Regression models:.....	3
I) Linear regression by solving normal equations	4
II) Gradient descent.....	5
III) Stochastic gradient descent.....	7
D) Questions to ponder on	9

A) Overview:

The following 3 algorithm techniques have been implemented to build an appropriate linear regression model to predict the insurance amount based on the given features in the dataset:

1. Linear regression by solving normal equations
2. Gradient descent
3. Stochastic gradient descent

The report contains a comparative analysis of the models built.

Programming language used: Python

Libraries used: NumPy, Pandas, Matplotlib

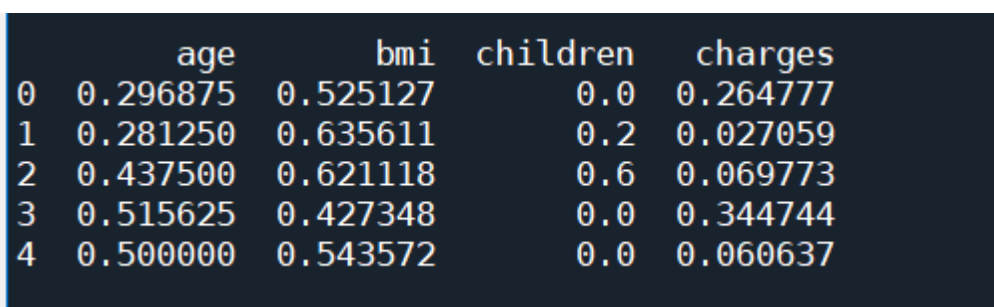
B) Description of Data:

The data used for building the models consists of 1338 datapoints. Each datapoint has 3 features and 1 target. The target variable is the insurance amount and the feature vectors for predicting the target variable are age, bmi and children.

The given features of the data lie in different ranges, for example, the range for age is 18-64, the range of bmi is 15-53, number of children lie between 0 and 5 and insurance amount varies from Rs.1122 to Rs.63770. Hence, we bring the dataset to a common scale without distorting the differences in the values by normalizing it. We normalize the data by dividing by the maximum value of each feature. Therefore, on normalization, each value is brought into the range 0-1.

The data is then shuffled and split into training data (70%) and testing data (30%).

The normalized data is of the form:



	age	bmi	children	charges
0	0.296875	0.525127	0.0	0.264777
1	0.281250	0.635611	0.2	0.027059
2	0.437500	0.621118	0.6	0.069773
3	0.515625	0.427348	0.0	0.344744
4	0.500000	0.543572	0.0	0.060637

Figure 1: Sample normalized data

C) Regression models:

The **linear regression model** to be developed is of the form:

$$y = w_0 + w_1 * x_1 + w_2 * x_2 + w_3 * x_3$$

where, x_1 = age, x_2 = bmi, x_3 = children.

Loss function to be minimized:

$$E = 0.5 * \sum (y_i - t_i)^2$$

where

i ranges from 1 to N

N = size of the training dataset

y_i = predicted value of altitude from model

t_i = given value of altitude in dataset

The dataset is divided into 2 parts: training data (70%) and testing data (30%).

Size of training dataset = 937

Size of testing dataset = 401

X_{train} = 936 rows with 3 features and 1 bias variable (feature vector of training data)

Y_{train} = 936 rows with corresponding target variable (target vector of training data)

X_{test} = 402 rows with 3 features and 1 bias variable (feature vector of testing data)

Y_{test} = 402 rows with corresponding target variable (target vector of testing data)

For each design algorithm, the regression is run by reshuffling and sampling the data 20 times and averaging values obtained from each sample.

The accuracy of the model is determined using the root mean squared error of the training and testing data set. RMSE (root mean square of error) indicates the absolute fit of the model to the data.

The RMSE for each of the 3 models is used to determine the accuracy of the model.

The formula used to calculate RMSE is as below:

$$RMSE = \sqrt{\frac{\sum_{i=1}^N (Predicted_i - Actual_i)^2}{N}}$$

I) Linear regression by solving normal equations

The **error function** to be minimized is:

$$J(X, w) = \frac{1}{2m} (Xw - y)^T (Xw - y)$$

where X is the input vector, Y is output vector and w is the weight vector.

The **final weight vector** is obtained by solving the following equation:

$$\text{weight} = ((X^T * X)^{-1} * X^T) * Y$$

The data is reshuffled and sampled 20 times to get more accurate results.

The **results** of the regression are as given below:

```
The regression line is of the form:
insurance = w0 + w1*age + w2*bmi + w3*children

Weights of the independent variables are:
w0 = -0.10807535219767583
w1 = 0.23744712493663847
w2 = 0.2820565931890515
w3 = 0.04638941111003468

RMSE mean of prediction of training data: 0.1794081854338918
RMSE variance of prediction of training data: 1.0747972397172261e-05

RMSE Mean of prediction of testing data: 0.17528397398786094
RMSE variance of prediction of testing data: 5.786636212307242e-05

Minimum RMSE of regression model using normal equation 0.16573827822659865
```

Figure 2: Regression results by solving normal equations

The minimum RMSE of the model is: 0.165738.

II) Gradient descent

Gradient descent is an optimization algorithm which is used to minimize the cost function value by iteratively moving in the opposite direction of the gradient with a learning rate.

The cost and the gradient functions are as below:

$$\begin{aligned} \text{Cost} \quad J(\theta) &= \frac{1}{2m} \sum_{i=1}^m (h(\theta)^{(i)} - y^{(i)})^2 \\ \text{Gradient} \quad \frac{\partial J(\theta)}{\partial \theta_j} &= \frac{1}{m} \sum_{i=1}^m (h(\theta)^{(i)} - y^{(i)}) \cdot X_j^{(i)} \end{aligned}$$

Figure 3: Cost and gradient function for GD

The weights are computed using the formula:

$$\text{weight} = \text{weight} - (\text{learning rate}) * (\text{gradient})_j$$

The parameters used for building the model:

Number of epochs = 500

Weight are initialized to zero.

We estimate the learning rate by experimenting different values based on the loss vs epoch graph.

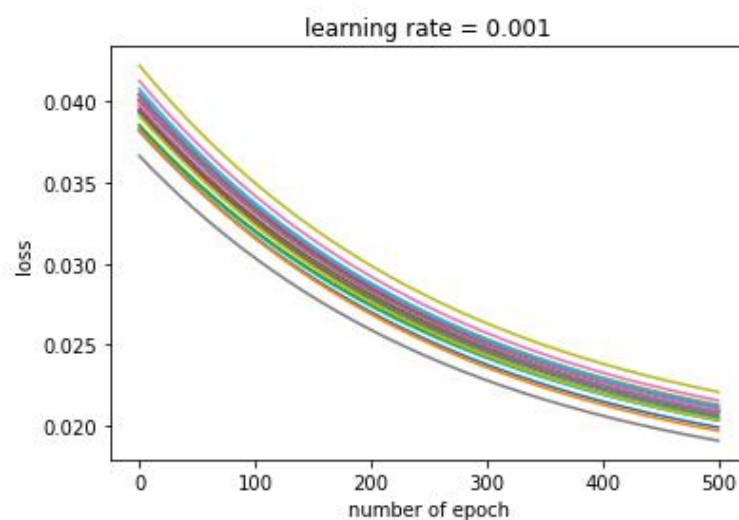


Figure 4: Loss vs Epoch for LR=0.001 (GD)

Learning rate = 0.001 shows a very gradual decline in the gradient and a more optimal choice of learning rate can be made by increasing it.

Hence, we keep experimenting by further increasing the value of learning rate.

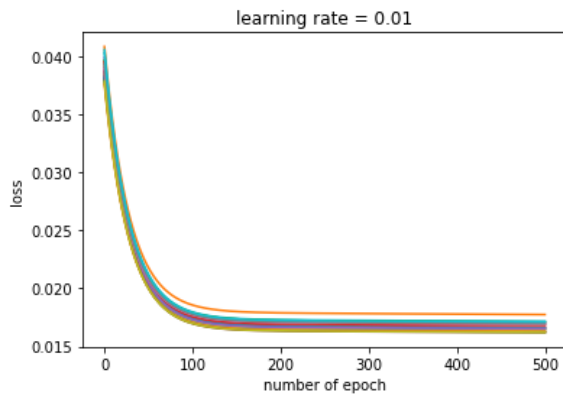


Figure 5: Loss vs Epoch for LR = 0.1 (GD)

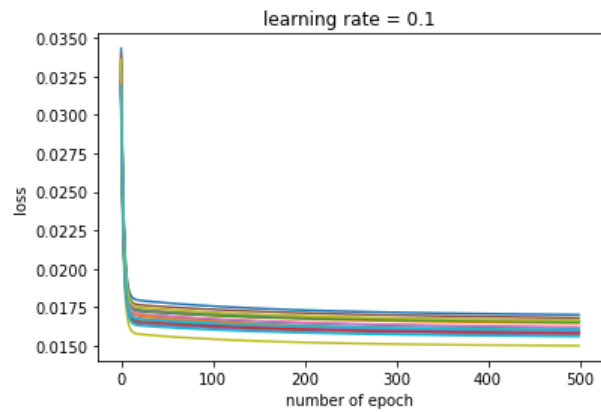


Figure 6: Loss vs Epoch for LR = 0.01 (GD)

The slope of the graph becomes steeper as the learning rate is increased and the loss value converges to zero towards the end of the epochs.

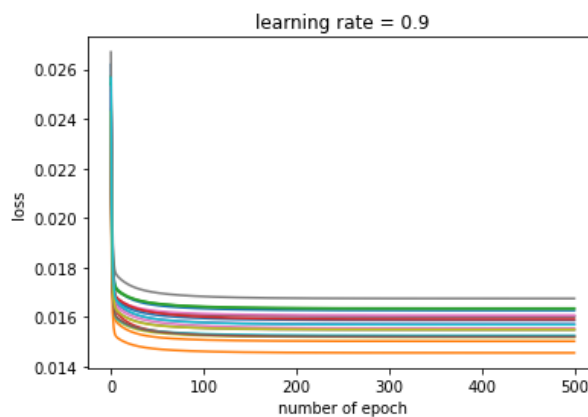


Figure 7: Loss vs Epoch for LR=0.9 (GD)

We finally choose **learning rate = 0.9** as we get minimum value of loss is this case. The graph is very steep in the first few iterations and gradually converges to zero.

The **results** obtained from gradient descent algorithm are as below:

```
The regression line is of the form:
insurance = w0 + w1*age + w2*bmi + w3*children

Weights of the independent variables are:
w0 = -0.11203310168539075
w1 = 0.24599818190691236
w2 = 0.2768488156820524
w3 = 0.04317563354256642

Mean accuracy prediction of training data: 0.17804358485710722
Variance of accuracy prediction of training data: 5.255803812310784e-06

Mean accuracy prediction of testing data: 0.178444966603115
Variance of accuracy prediction of testing data: 2.8248469161456183e-05

Minimum RMSE of regression model using normal equation 0.1671436398000102
```

Figure 8: Regression results of Gradient descent

The minimum RMSE of the model using gradient descent is 0.1671436.

III) Stochastic gradient descent

Stochastic gradient descent further optimizes the gradient descent algorithm by randomly selecting a few samples rather than the whole set to compute the gradient for each iteration.

We sample the data 20 times to get more accurate results.

Number of epochs per sampling of data = 150

Weights are initialized to 0.

Similar to the previous case, we experiment different values of learning rate by plotting the loss vs epoch graph.

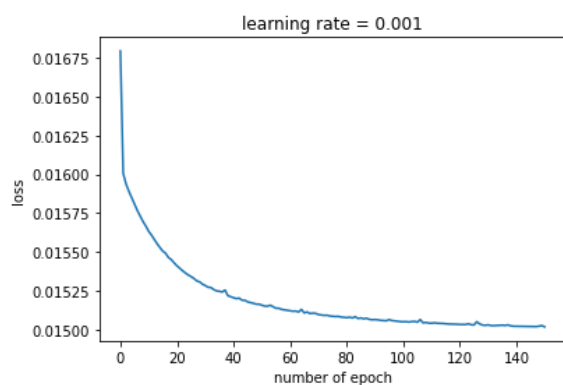


Figure 9: Loss vs Epoch for LR=0.01 (SGD)

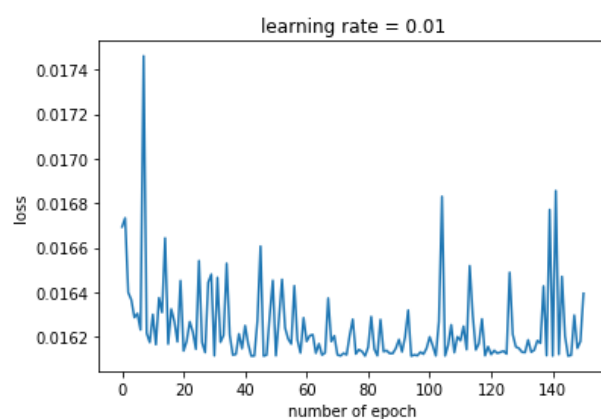


Figure 10: Loss vs Epoch for LR=0.001 (SGD)

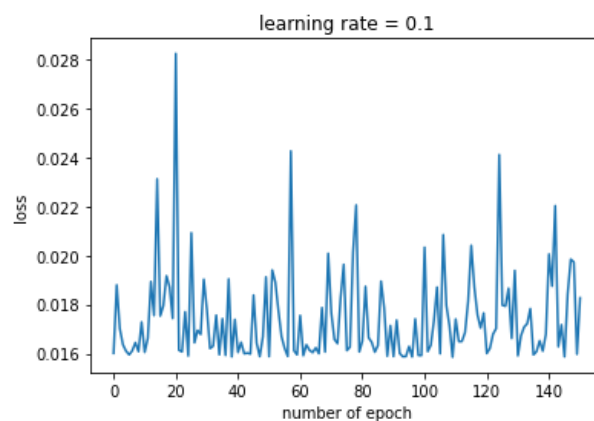


Figure 11: Loss vs Epoch for LR = 0.001 (SGD)

We can observe that the loss vs epoch graph is quite noisy as compared to the gradient descent model. This is because we are choosing random datapoints to calculate the gradient at every iteration rather than using the entire set.

But similar to the previous model, the optimal learning rate would be such that the loss declines rapidly in the first few iterations and converges to zero towards the end.

The first graph having learning rate as 0.001 is close to our required graph. On increasing the learning rate to 0.01 or 0.1 we see that the loss no longer converges to zero towards the

end of the epochs. This is because the value of each step(learning rate) is too large and it crosses the minima of the function.

Based on the observations, we choose **learning rate as 0.005**.

The graph below shows the loss vs epoch graph for 20 samples of the data :

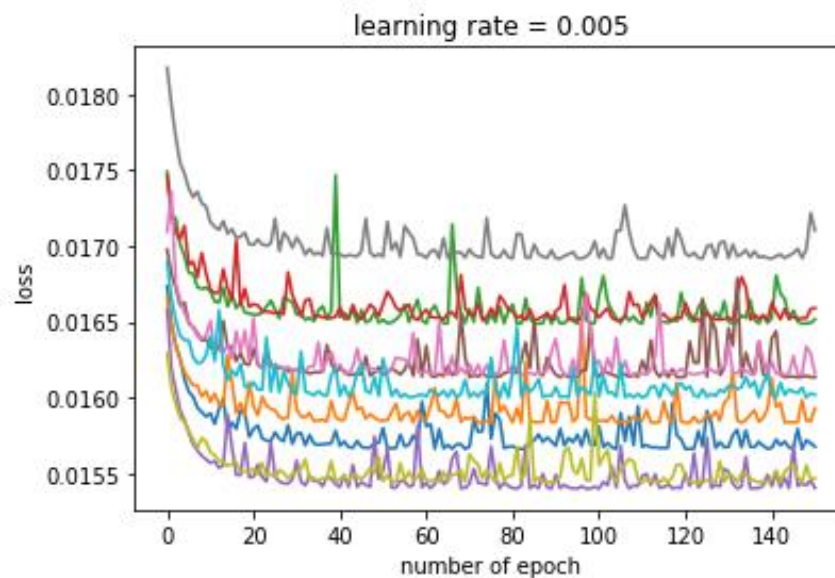


Figure 12: Loss vs Epoch for LR=0.005 (SGD) for 20 samples

The **resulting model** got using the stochastic gradient descent technique is:

```
The regression line is of the form:
insurance = w0 + w1*age + w2*bmi + w3*children

Weights of the independent variables are:
w0 = -0.10368672067594889
w1 = 0.23738174505087284
w2 = 0.270802821985573
w3 = 0.03804654942797673

RMSE Mean of accuracy prediction of training data: 0.1778453117526797
RMSE Variance of accuracy prediction of training data: 9.533321171326615e-06

RMSE Mean of accuracy prediction of testing data: 0.18003483169480292
RMSE Variance of accuracy prediction of testing data: 4.8211129008430176e-05

Minimum RMSE of regression model using normal equation 0.1701335051588025
```

Figure 13: Regression results using Stochastic Gradient Descent

The minimum RMSE of the stochastic gradient model is 0.170133.

D) 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?

A1. The regression models obtained by the 3 methods are as follows:

Linear regression by solving normal equations model:

$$y = -0.1080 + 0.2374*age + 0.2820*bmi + 0.0463*children$$

Model using Gradient descent:

$$y = -0.1120 + 0.2459*age + 0.2768*bmi + 0.0431*children$$

Model using Stochastic gradient descent:

$$y = -0.1036 + 0.2374*age + 0.2708*bmi + 0.03804*children$$

The table below shows the summary statistics of the testing data for each model:

	RMSE mean of accuracy prediction	RMSE variance of accuracy prediction	Minimum RMSE
Regression solving Normal Eqn	0.17528	5.7866e - 5	0.165738
Gradient Descent	0.17844	2.8248e - 5	0.167143
Stochastic Gradient Descent	0.18003	4.82E-05	0.170133

The model which was built solving the normal equations has the lowest RMSE value.

But the error values of the all the 3 methods are relatively close and hence, we can say that all 3 methods yield similar results.

The stochastic gradient descent optimization method would be most efficient while working with real world data. For larger datasets, the stochastic gradient descent algorithm would converge to the minima faster as it performs updates frequently. Also, large datasets contain lot of redundancy which is appropriately approximated in this method.

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

A2. Variables that are measured using different units or scale do not contribute equally to the analysis and hence if used without pre-processing may create a bias in the results.

Hence, in cases where the dataset contains variables whose range is distinctly different from each other, we normalize the data before using it for building the regression model. The dataset is brought to a common scale without distorting the differences in their values. This way the true effect of the predictor variable can be observed.

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

A3. As we increase the number of iterations, the loss keeps on decreasing. The slope of the loss function is very steep in the initial iterations and becomes almost flat when loss value reaches close to 0. The value of loss decreases very slowly towards the end. Hence, after a very large number of epochs, the loss value tends to 0 but there is no significant difference in the values of the loss values as the decrement in loss after each iteration is very very small or negligible.

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

A4. We compare the loss function of gradient descent and stochastic gradient descent for different learning rates, keep the epochs constant.

Learning rate = 0.1

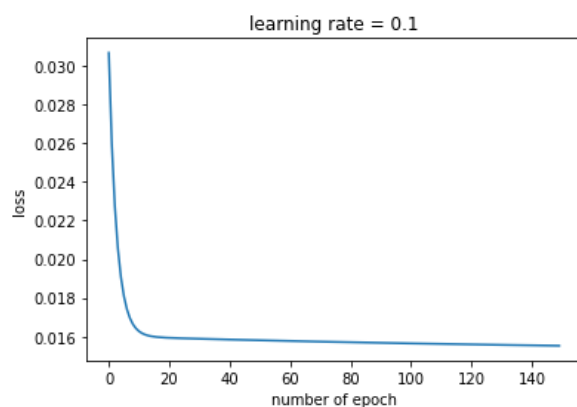


Figure 14: GD with LR=0.1

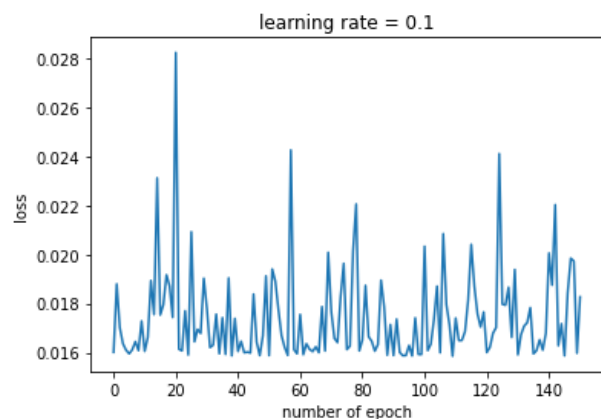


Figure 15: SGD with LR = 0.1

Learning rate = 0.01

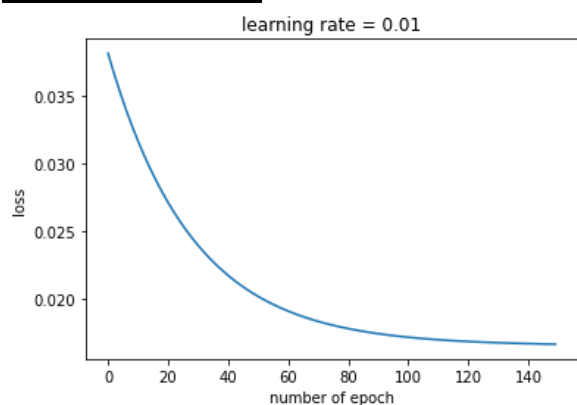


Figure 16: GD with LR=0.01

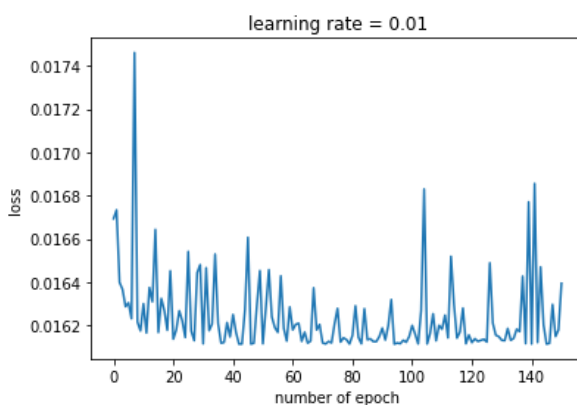


Figure 17: SGD with LR=0.01

Learning rate = 0.001

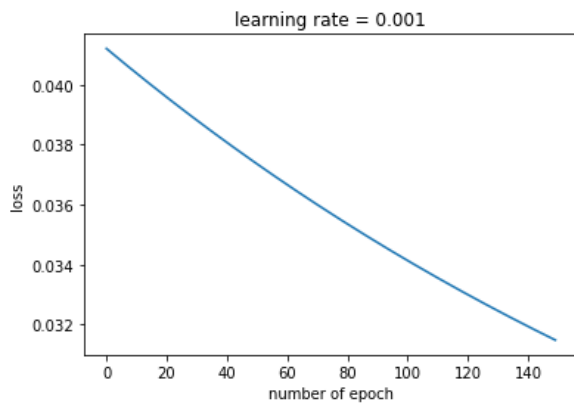


Figure 18: GD with LR=0.001

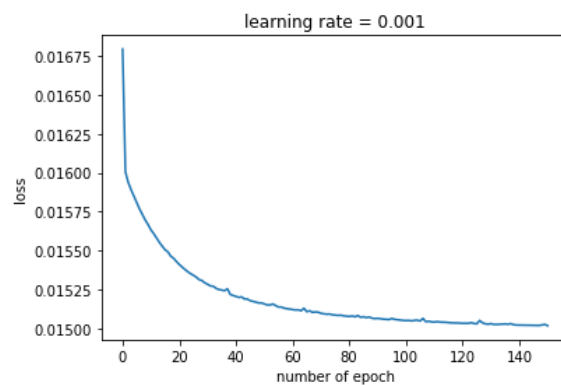


Figure 19: SGD with LR=0.001

The primary difference between the loss vs epoch plots of gradient descent and stochastic gradient descent is that the stochastic gradient descent loss plot is much noisier than gradient descent plot. This is because the stochastic gradient descent algorithm uses random datapoints to calculate the gradient value as opposed to gradient descent which calculates the gradient with respect to all of the datapoints.

We can also observe that the stochastic gradient descent algorithm converges to the minima very fast as compared to the gradient descent as updates are made frequently in this case. We say that the stochastic gradient descent algorithm converges to minima faster as the loss function reaches near 0 values for a low value of learning rate (~ 0.001). While the same data requires a higher learning rate (~ 0.1) to reach the minima of the loss function in case of gradient descent algorithm.

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

A5. If the value of the learning rate is very high, there is a possibility that the loss function crosses the minima and its value increases, which is undesirable (as we want to minimize the error). Hence, it is important to choose an appropriate learning rate that makes the loss function converge to 0.

Q6. Would the minima (minimum error achieved) have changed if the bias term (w_0) were not to be used, i.e. the prediction is given as $Y=WTX$ instead of $Y=WTX+B$?

A6. Yes, the minimum error achieved would have changed if the bias term were omitted from the prediction model. The effect of the bias term would have been added to the error of the model and hence the minimum error achieved would be greater than the one obtained with the bias term.

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?

A7. The weight vector is the coefficient matrix of the model the algorithm is trying to predict. Each weight signifies the influence of the corresponding feature on the target variable. Based on the model of our given dataset, we can observe that bmi has the maximum influence on predicting insurance amount, while number of children has minimum influence on predicting the insurance amount.