Machine Learning: Assignment 1

Part (a):

Model parameters for $\lambda = 0$:-

θ_0 = -101502.516871
θ_1 = 1.05363315568
θ_2 = 94624.4513358
θ_3 = 57091.2982001
θ_4 = 154412.027973

Price = $\theta_0 + (\theta_1 * \text{sqft}) + (\theta_2 * \text{floors}) + (\theta_3 * \text{bedrooms}) + (\theta_4 * \text{bathrooms})$

Model parameters for $\lambda = 1$:-

θ_0' = -101478.246666
θ_1' = 1.05352778772
θ_2' = 94619.812371
θ_3' = 57088.6430526
θ_4' = 154408.620888

Price = $\theta_0' + (\theta_1' * \text{sqft}) + (\theta_2' * \text{floors}) + (\theta_3' * \text{bedrooms}) + (\theta_4' * \text{bathrooms})$

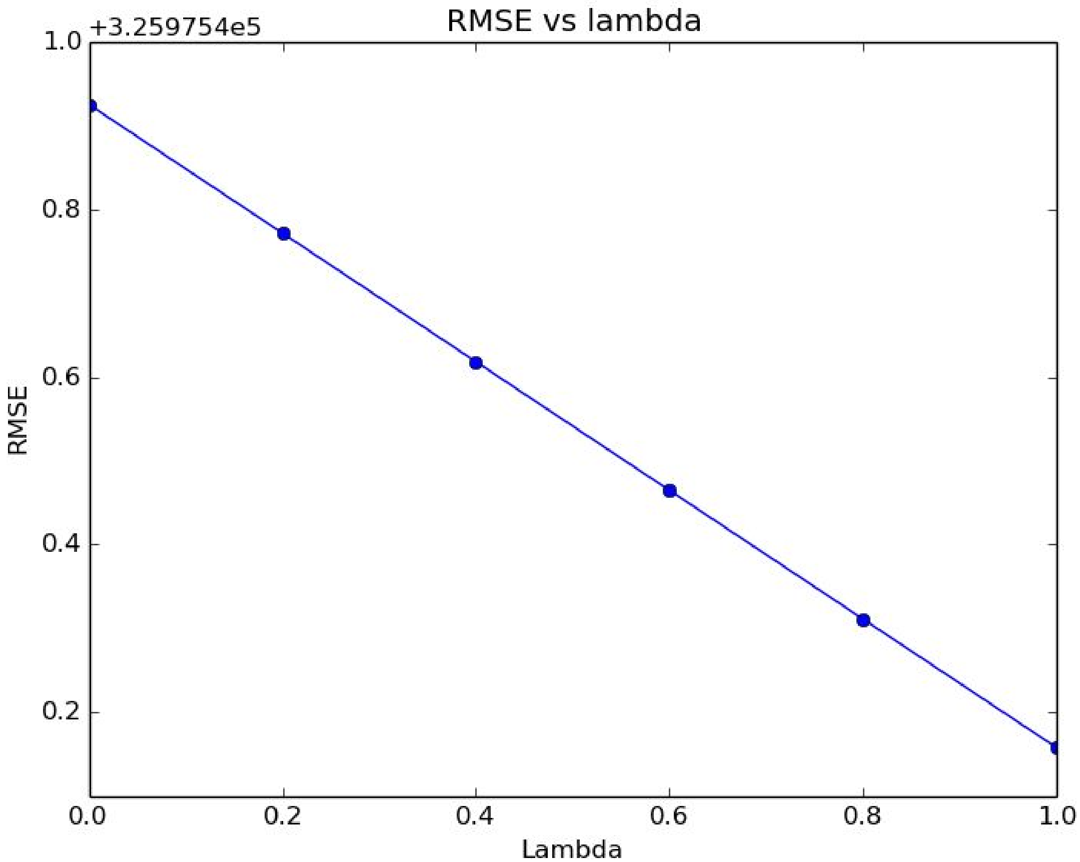
Note :-

1) First 80% of the data set was used to train the model and the remaining 20% was used to compute RMSE (root mean square error).

2) Values of some parameters were taken as follows :-

   Learning rate = 0.05
   Number of iterations of gradient descent = 50
   Lambda = 1

Plots :-



Part (b):

Model parameters for gradient descent optimization :-

θ_0 = -101502.516871
θ_1 = 1.05363315568
θ_2 = 94624.4513358
θ_3 = 57091.2982001
θ_4 = 154412.027973

Price = θ_0 + (θ_1* sqft) + (θ_2* floors) + (θ_3* bedrooms) + (θ_4* bathrooms)

Model parameters for iterative re-weighted least squares optimization :-

θ_0' = -2572007.44041
θ_1' = 8.77352989607
θ_2' = 715748.291503

θ_3' = 389895.978197
θ_4' = 481802.44599

Price = θ_0' + (θ_1' * sqft) + (θ_2' * floors) + (θ_3' * bedrooms) + (θ_4' * bathrooms)

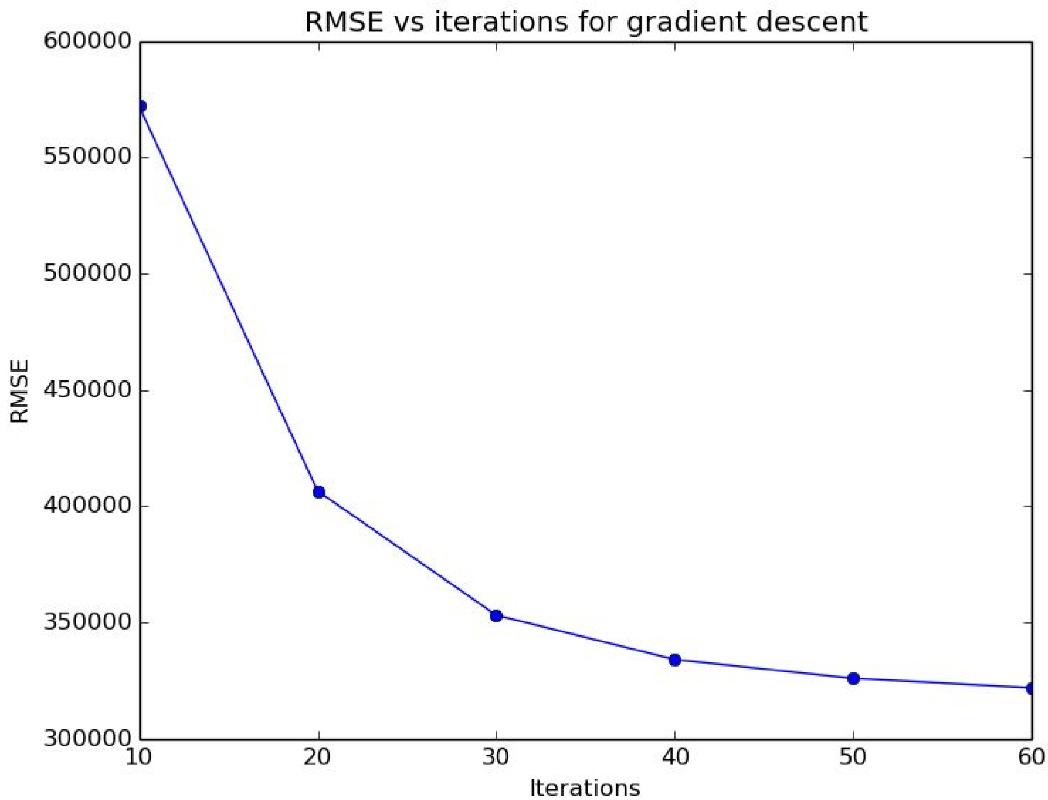
Note :-

1) First 80% of the data set was used to train the model and the remaining 20% was used to compute RMSE (root mean square error).

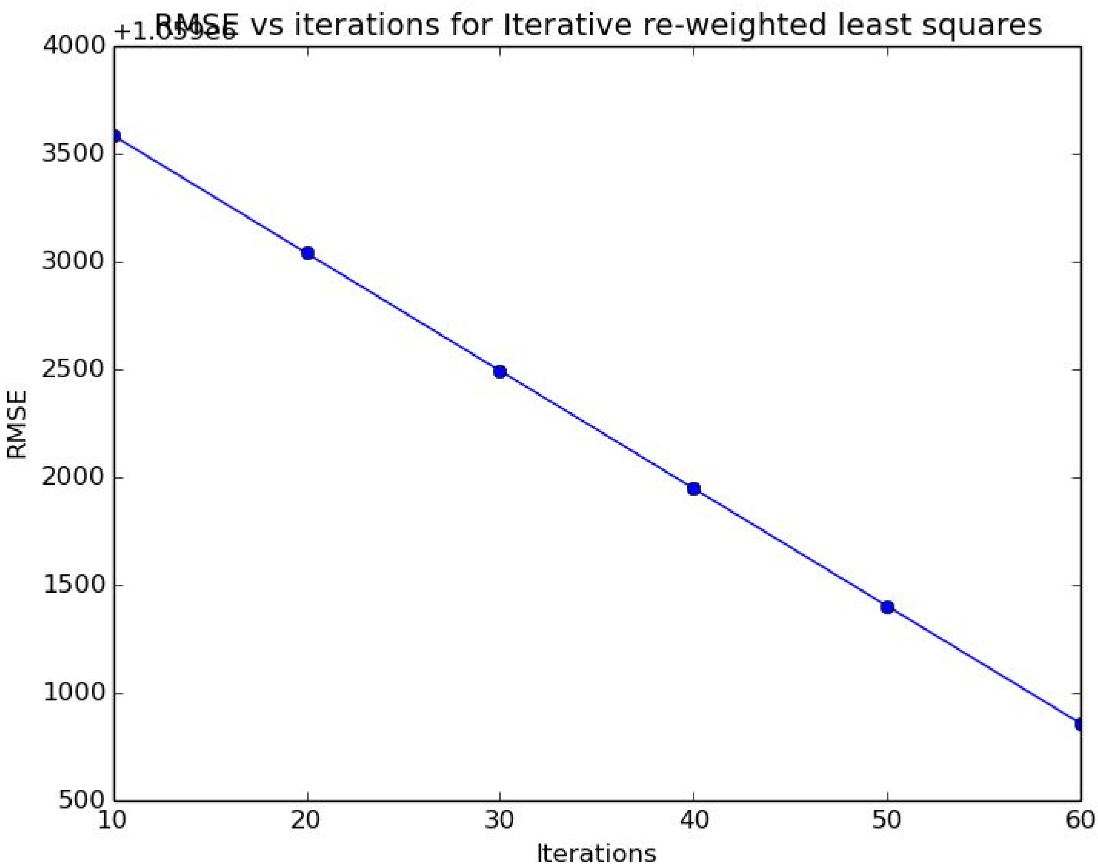
2) Values of some parameters were taken as follows :-

   Learning rate for gradient descent = 0.05
   Number of iterations = 50 (for both methods)
   Lambda = 0 (for both methods)

Plots :-



RMSE vs iterations for gradient descent
600000
550000
500000
450000
400000
350000
300000
RMSE
10
20
30
40
50
60
Iterations



Conclusion :-

I would prefer Iterative Re-weighted Least Squares method because in this method cost function J reaches close to minimum in very less number of iterations whereas in Gradient Descent method, cost function takes higher number of iterations to compute an optimal theta vector.

Part (c):

Model parameters for linear combination :-

θ_0 = -101502.516871

θ_1 = 1.05363315568

θ_2 = 94624.4513358

θ_3 = 57091.2982001

θ_4 = 154412.027973

Price = θ_0 + (θ_1* sqft) + (θ_2* floors) + (θ_3* bedrooms) + (θ_4* bathrooms)

Model parameters for quadratic combination :-

θ_0' = 240674.876584
θ_1' = 1.0958727625e-06
θ_2' = 24570.3276792
θ_3' = 3570.33127621
θ_4' = 45150.3517955

Price = θ_0' + (θ_1' * $sqft^2$) + (θ_2' * $floors^2$) + (θ_3' * $bedrooms^2$) + (θ_4' * $bathrooms^2$)

Model parameters for cubic combination :-

θ_0'' = 368131.485933
θ_1'' = 8.10115960394e-13
θ_2'' = 10562.1282525
θ_3'' = 109.80827526
θ_4'' = 11381.0515774

Price = θ_0'' + (θ_1'' * $sqft^3$) + (θ_2'' * $floors^3$) + (θ_3'' * $bedrooms^3$) + (θ_4'' * $bathrooms^3$)

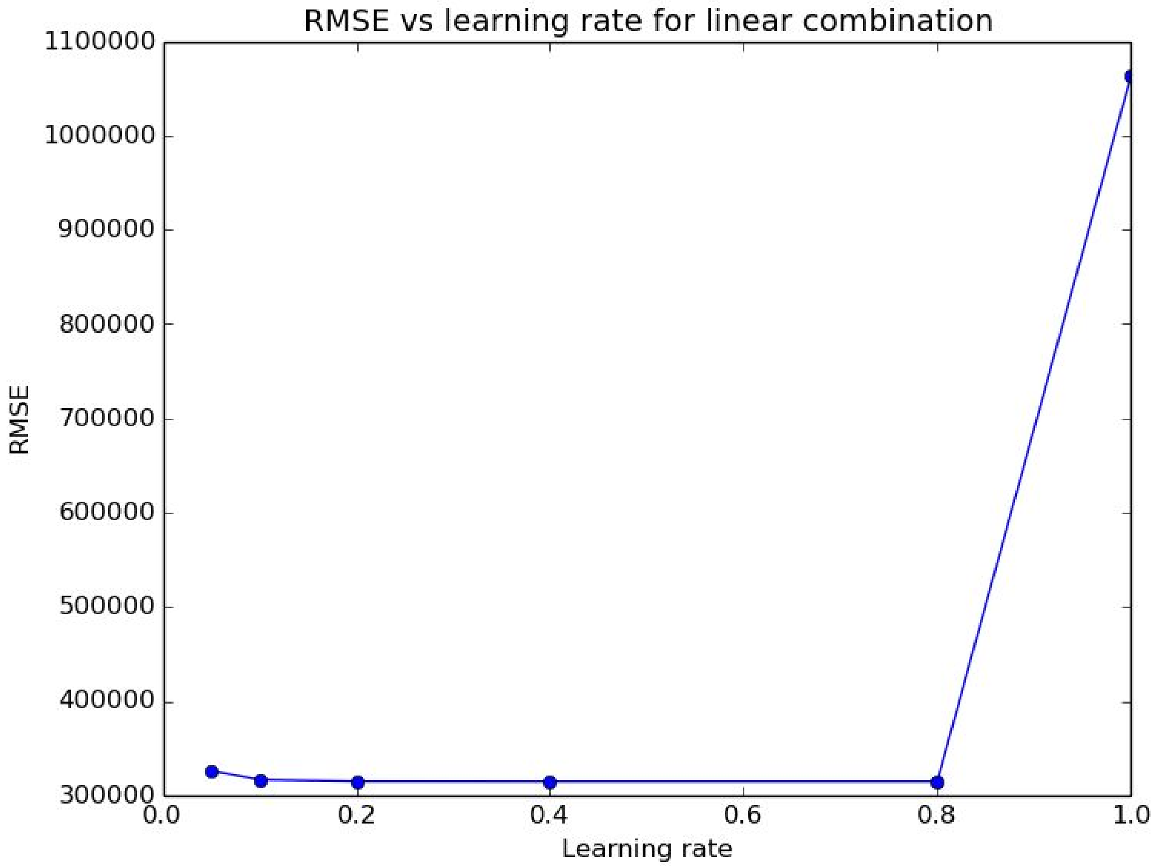
Note :-

1) First 80% of the data set was used to train the model and the remaining 20% was used to compute RMSE (root mean square error).

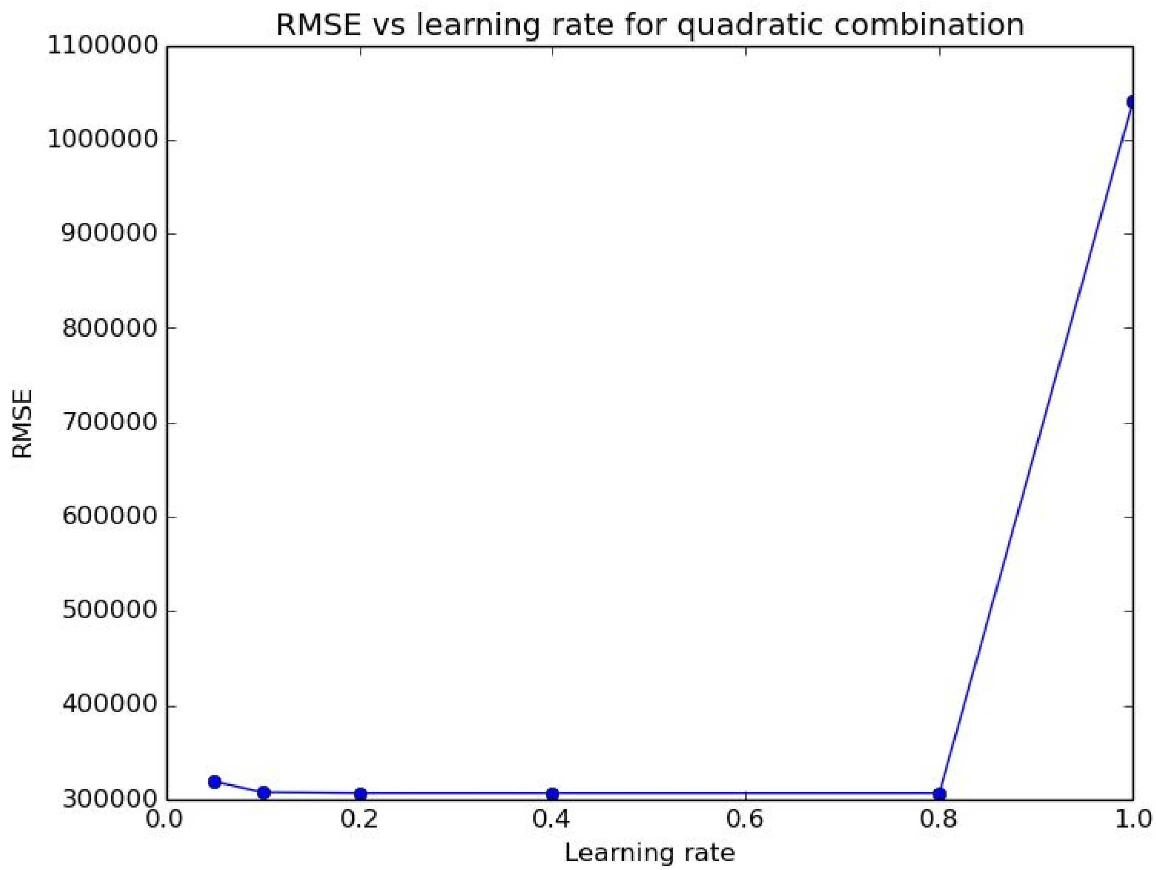
2) Values of some parameters were taken as follows :-
   Learning rate = 0.05
   Number of iterations = 50
   Lambda = 0

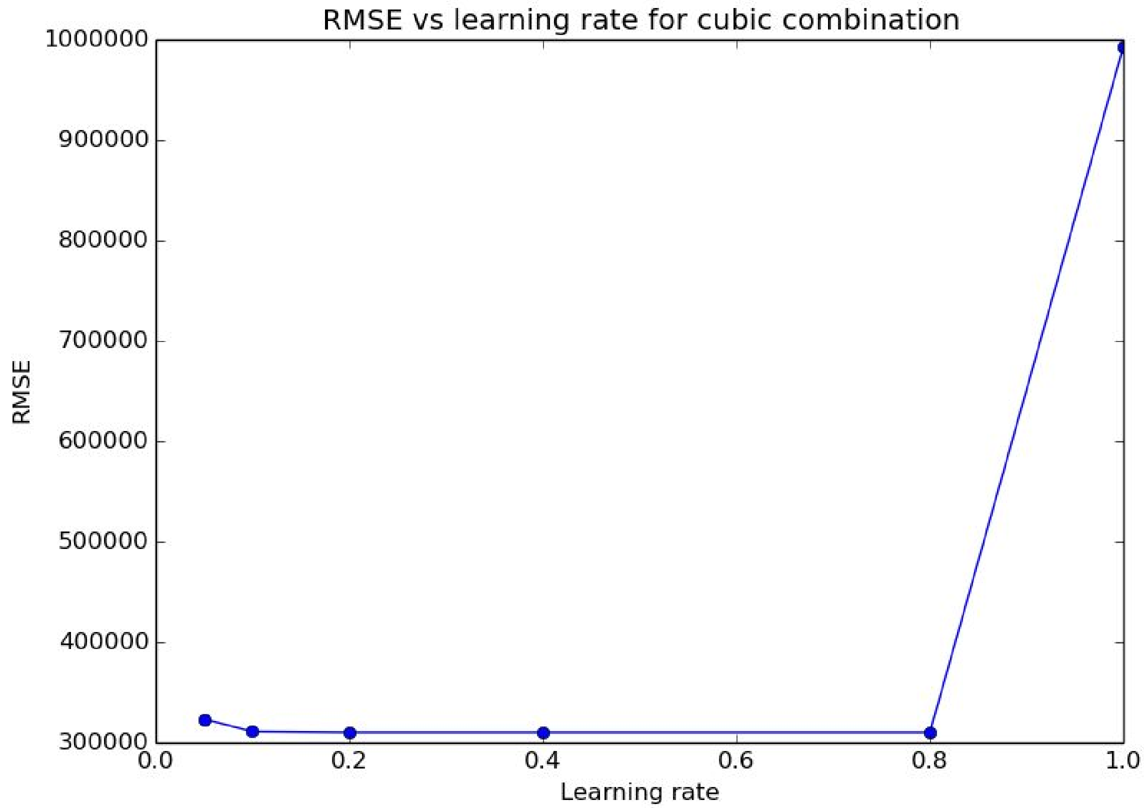
3) For quadratic or cubic combination of features, new features like sqft*$floor^2$ (in cubic) and sqft*floor (in quadratic) were not included to reduce complexity. Only terms with power 2 or 3 on a single feature were included to simplify computation.

Plots :-



RMSE vs learning rate for linear combination
1100000
1000000
900000
800000
700000
600000
500000
400000
300000
RMSE
0.0
0.2
0.4
0.6
0.8
1.0
Learning rate


RMSE vs learning rate for quadratic combination
RMSE
Learning rate
1100000
1000000
900000
800000
700000
600000
500000
400000
300000
0.0
0.2
0.4
0.6
0.8
1.0



<u>Conclusion</u> :-
Based on above curves it is difficult to claim that one of them is better than the other because RMSE vs learning rate curve is almost the same for all of them. So a better decision would be to observe the cost function in all three cases.

I would prefer cubic combination because cost function is minimized most when cubic combinations are used than when quadratic or linear combinations are used.

Part (d):

Model parameters with cost function = Mean Absolute Error :-

θ_0 = 44998.1596681
θ_1 = 0.423008234451
θ_2 = 37645.4140531
θ_3 = 19508.8260764
θ_4 = 147766.950772

Price = θ_0 + (θ_1* sqft) + (θ_2* floors) + (θ_3* bedrooms) + (θ_4* bathrooms)

Model parameters with cost function = Mean Squared Error :-

θ_0' = -39798.9153911
θ_1' = 0.377385243716
θ_2' = 20140.2688916
θ_3' = 17474.6650559
θ_4' = 232400.500135

Price = θ_0' + (θ_1' * sqft) + (θ_2' * floors) + (θ_3' * bedrooms^2) + (θ_4' * bathrooms)

Model parameters with cost function = Mean Cube Error :-

θ_0'' = 5296080.78825
θ_1'' = -60.7477934194
θ_2'' = -775481.435759
θ_3'' = -641280.825687
θ_4'' = -649481.677648

Price = θ_0'' + (θ_1'' * sqft) + (θ_2'' * floors) + (θ_3'' * bedrooms) + (θ_4'' * bathrooms)

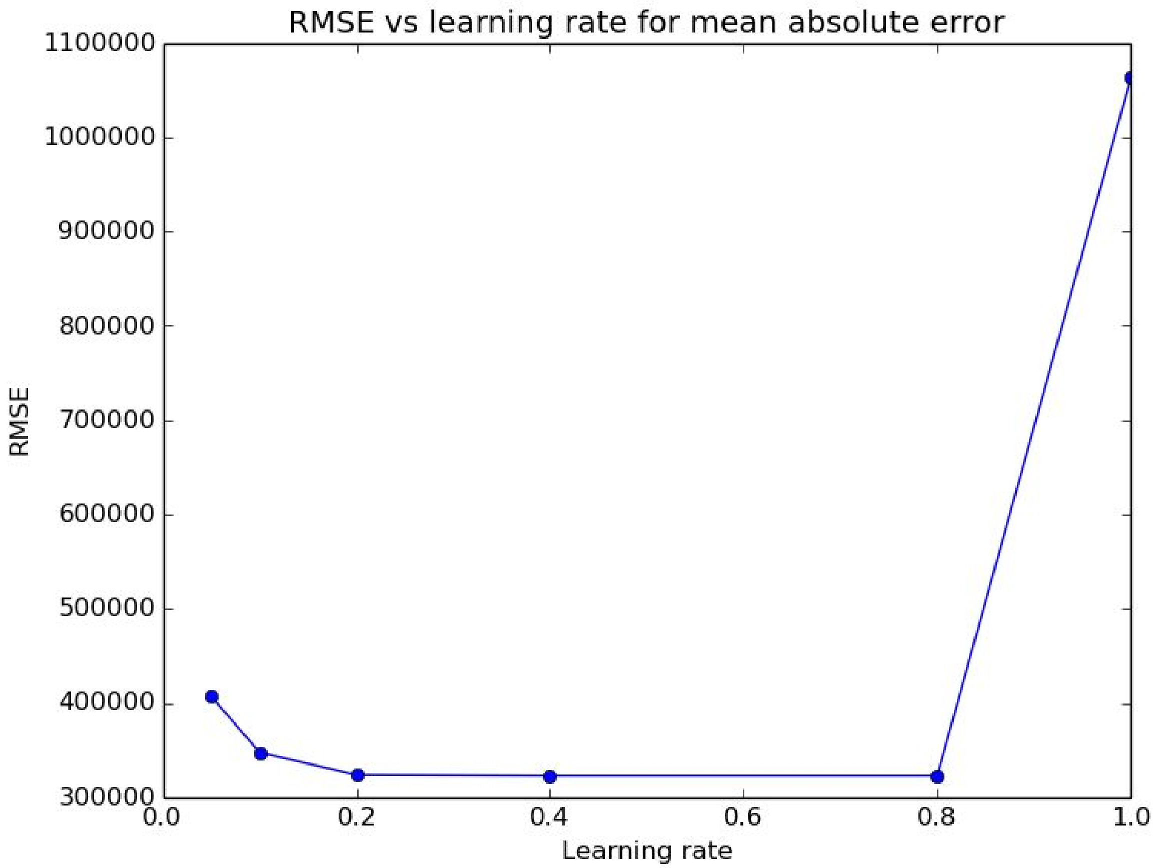
<u>Note</u> :-

1) First 80% of the data set was used to train the model and the remaining 20% was used to compute RMSE (root mean square error).

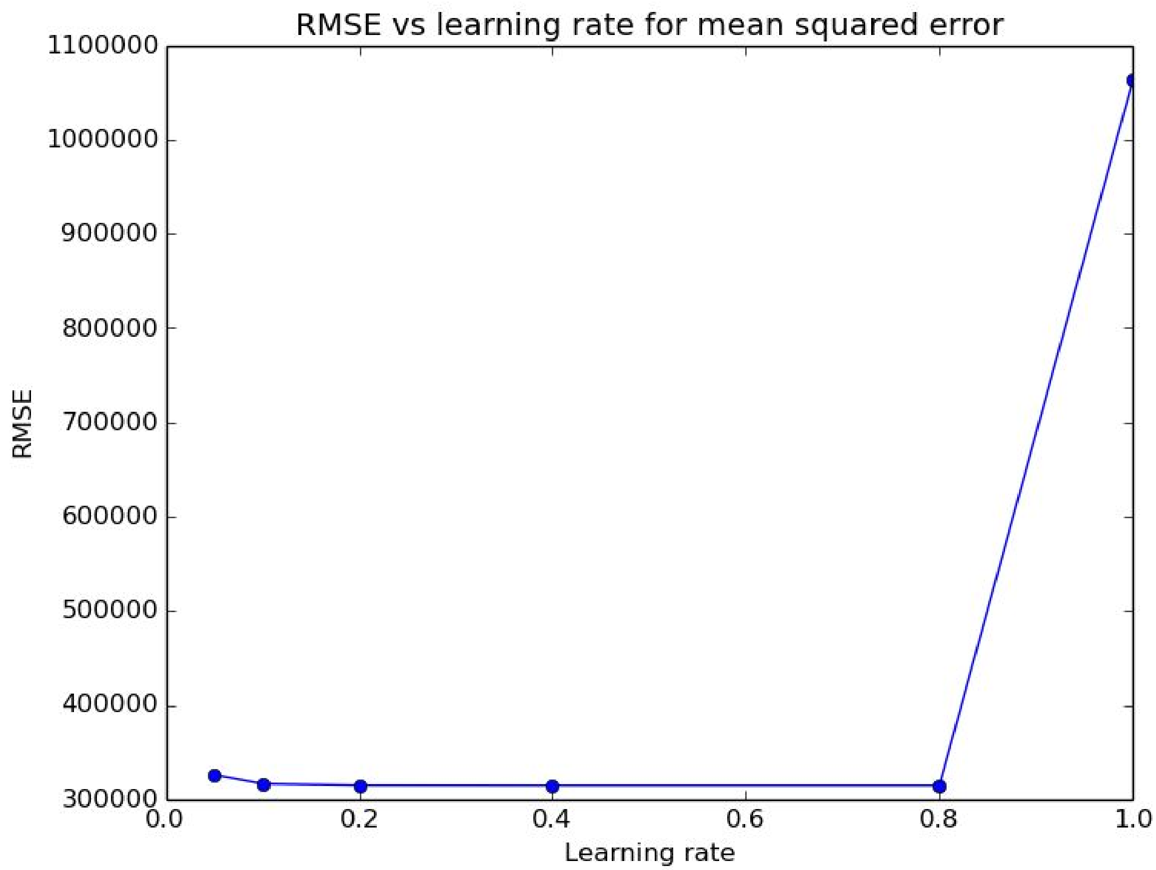
2) Values of some parameters were taken as follows :-

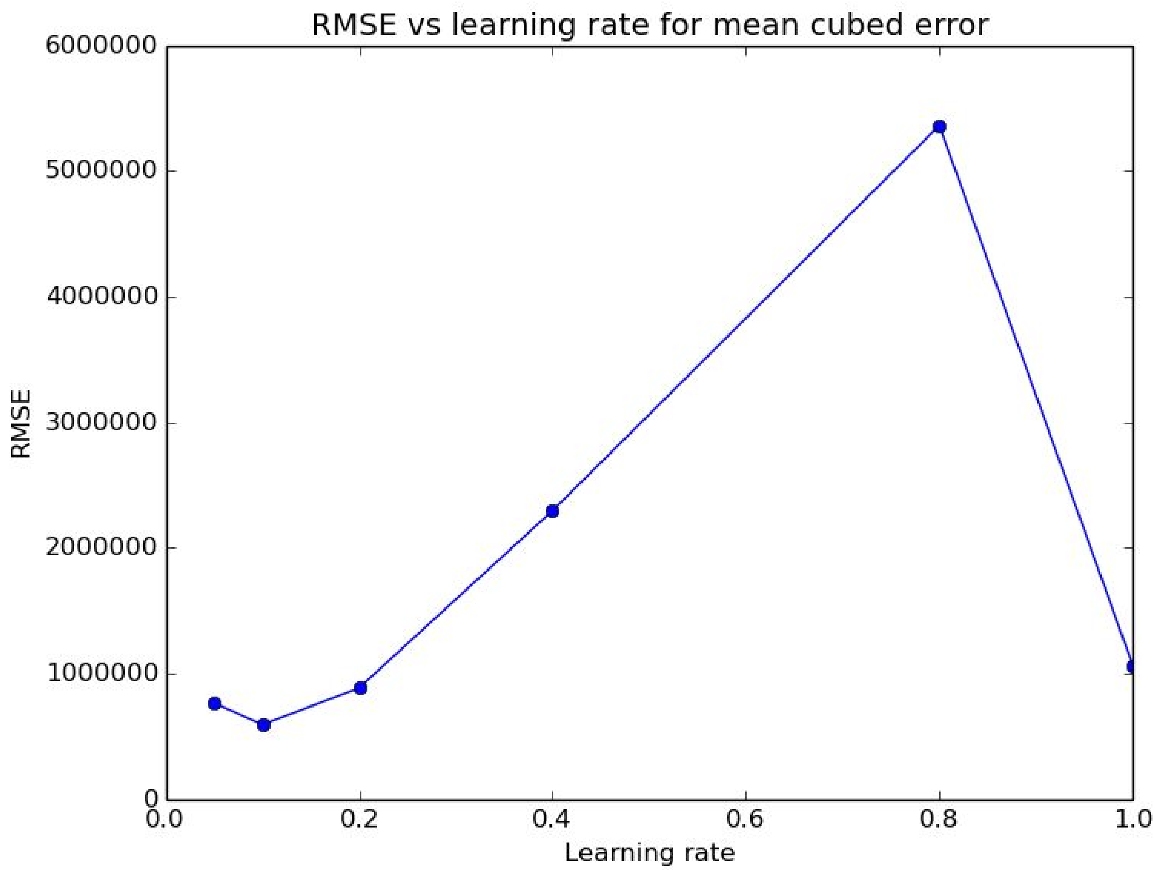
   Learning rate = 0.05
   Number of iterations for MAE and MSE = 50
   Lambda = 0
   Number of iterations for MCE = 2 (higher iterations give Nan as output because of non-convergence)

Plots :-



RMSE vs learning rate for mean absolute error
1100000
1000000
900000
800000
700000
600000
500000
400000
300000
RMSE
0.0
0.2
0.4
0.6
0.8
1.0
Learning rate


RMSE vs learning rate for mean squared error
1100000
1000000
900000
800000
700000
600000
500000
400000
300000
RMSE
0.0
0.2
0.4
0.6
0.8
1.0
Learning rate



Conclusion :-

I would prefer mean square error cost function because this method requires lesser number of iterations, than other two methods, to obtain optimal theta and reduce cost considerably. Mean cube error was NOT preferred because that method may not converge at all as is evident from its plot.