Homework 3
Vibhuti Mahajan
uni : vm2486

Problem 1:

a) The code for Gaussian Process is in vm2486_hw3.py

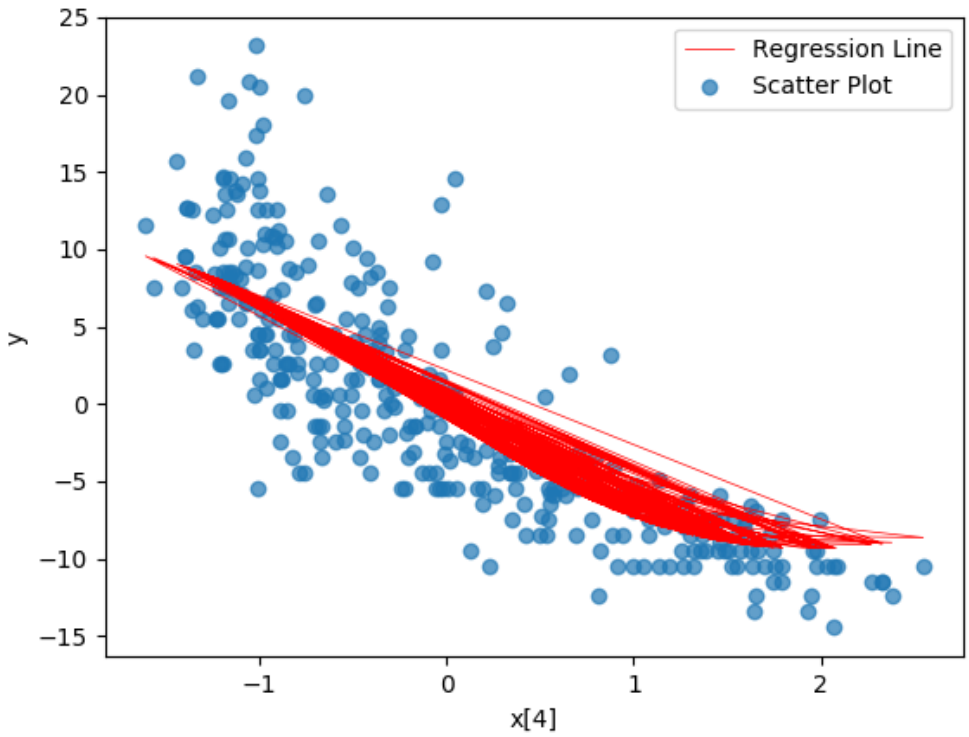
b) Table for rmse values for the given values of b and σ^2:

b	σ_2	$RMSE$
5	0.1	1.9663
5	0.2	1.9331
5	0.3	1.9234
5	0.4	1.9222
5	0.5	1.9248
5	0.6	1.9292
5	0.7	1.9346
5	0.8	1.9406
5	0.9	1.9468
5	1	1.9532
7	0.1	1.9202
7	0.2	1.9049
7	0.3	1.9081
7	0.4	1.9159
7	0.5	1.9248
7	0.6	1.9337
7	0.7	1.9423
7	0.8	1.9504
7	0.9	1.9581
7	1	1.9654
9	0.1	1.8976
9	0.2	1.9025
9	0.3	1.9176
9	0.4	1.9325
9	0.5	1.9457
9	0.6	1.9572
9	0.7	1.9674
9	0.8	1.9765
9	0.9	1.9847
9	1	1.9923
11	0.1	1.8905
11	0.2	1.915
11	0.3	1.9388
11	0.4	1.9579
11	0.5	1.9732
11	0.6	1.9858
11	0.7	1.9964
11	0.8	2.0056
11	0.9	2.0138
11	1	2.0213
13	0.1	1.8958
13	0.2	1.9356

13	0.3	1.9646
13	0.4	1.9855
13	0.5	2.0013
13	0.6	2.0139
13	0.7	2.0243
13	0.8	2.0333
13	0.9	2.0413
13	1	2.0486
15	0.1	1.9096
15	0.2	1.9595
15	0.3	1.9908
15	0.4	2.0119
15	0.5	2.0274
15	0.6	2.0395
15	0.7	2.0495
15	0.8	2.0581
15	0.9	2.0658
15	1	2.073

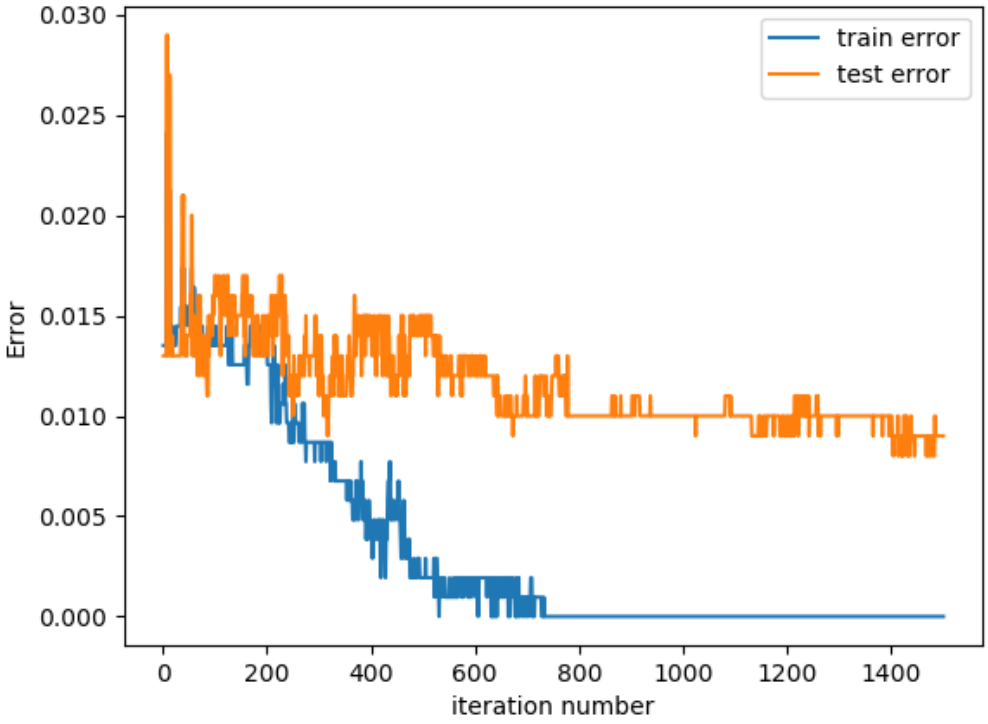
c) Best value of RMSE is 1.8905 achieved for b = 11.0, and $\sigma^2 = 0.1$. As compared to homework 1, the best value of RMSE for the Gaussian processes is lower. For Gaussian process I had stored a kernel matrix that uses $O(n^2)space$, where n is the size of the input data. Also the calculation of this kernel matrix requires $O(n^2 d)$ time where d is the size of the feature space. If n is large (in TB/ PB, which is highly possible with real-life datasets), then storing this kernel matrix will require a huge amount of auxiliary space. Also, we would not prefer to calculate the matrix values again and again due to high time complexity.

d)

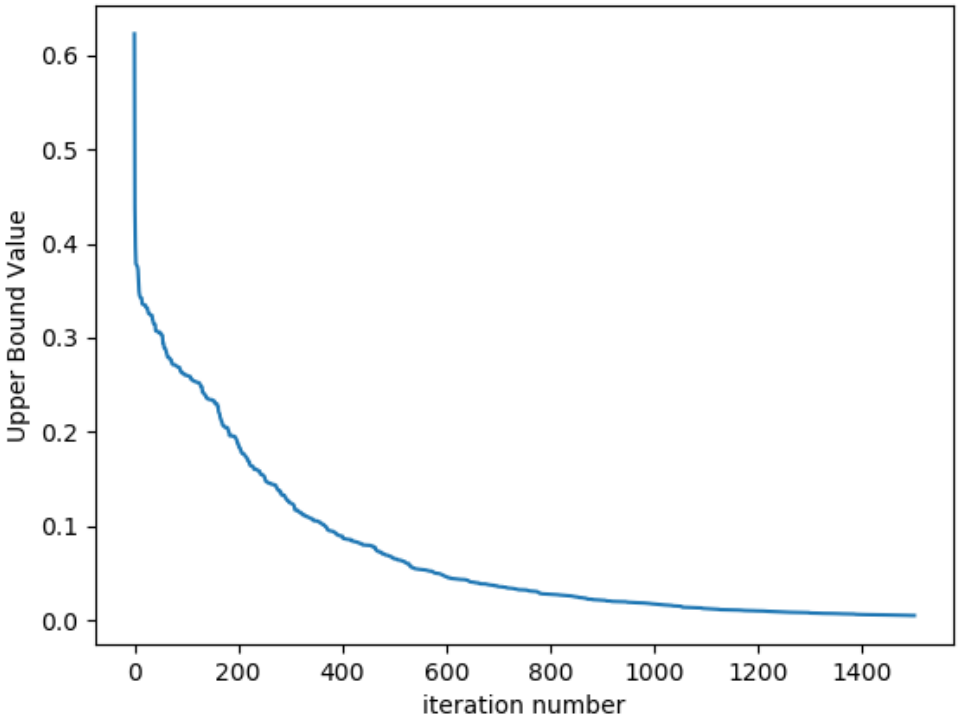


Problem2:

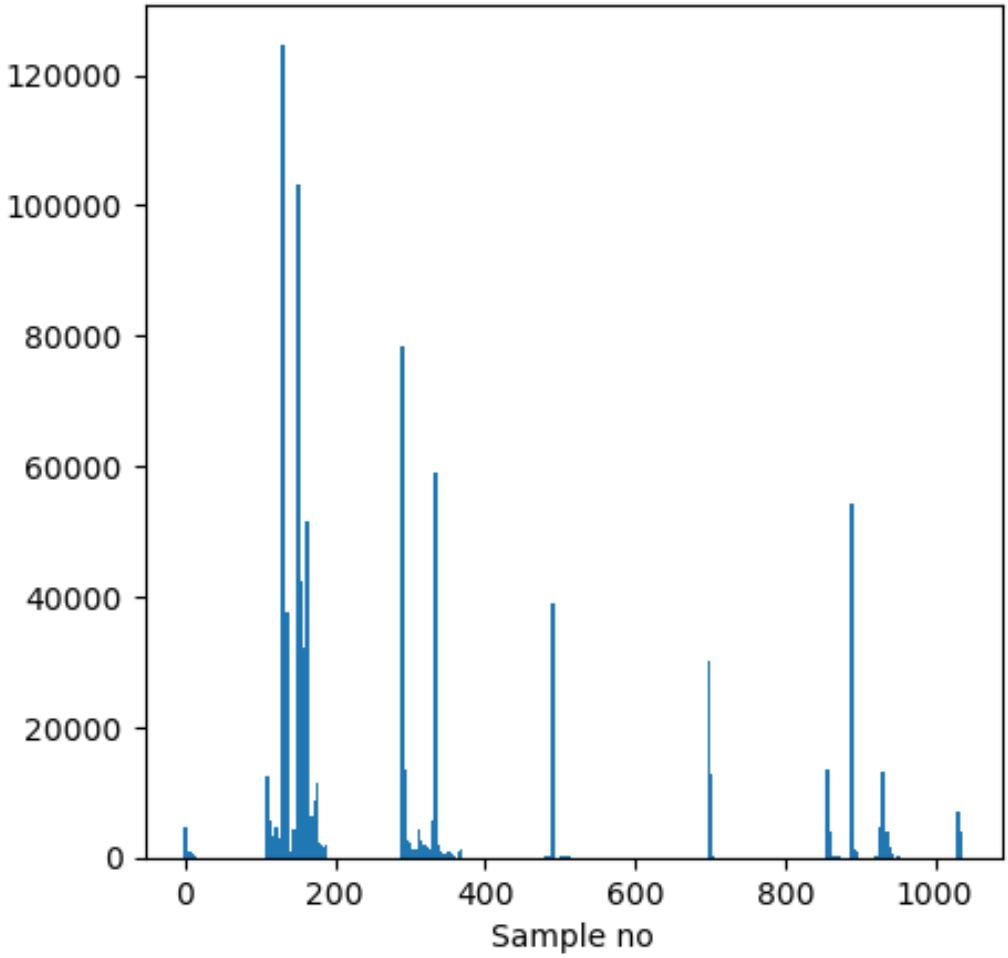
a) Training Error for Gradient Boosting v/s Test Error



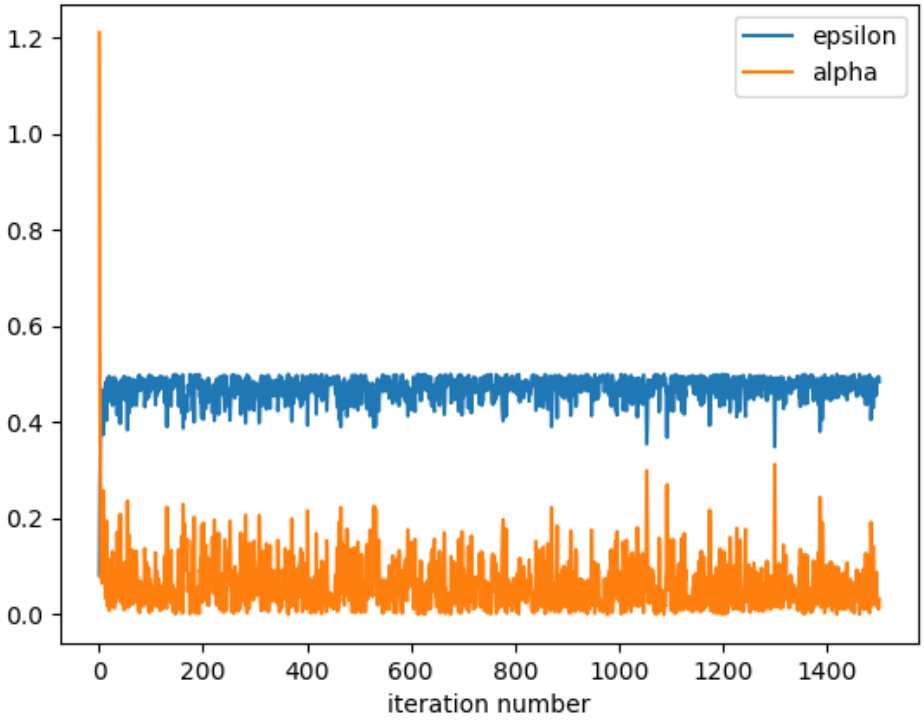
b) Upper bound as a funtion of t

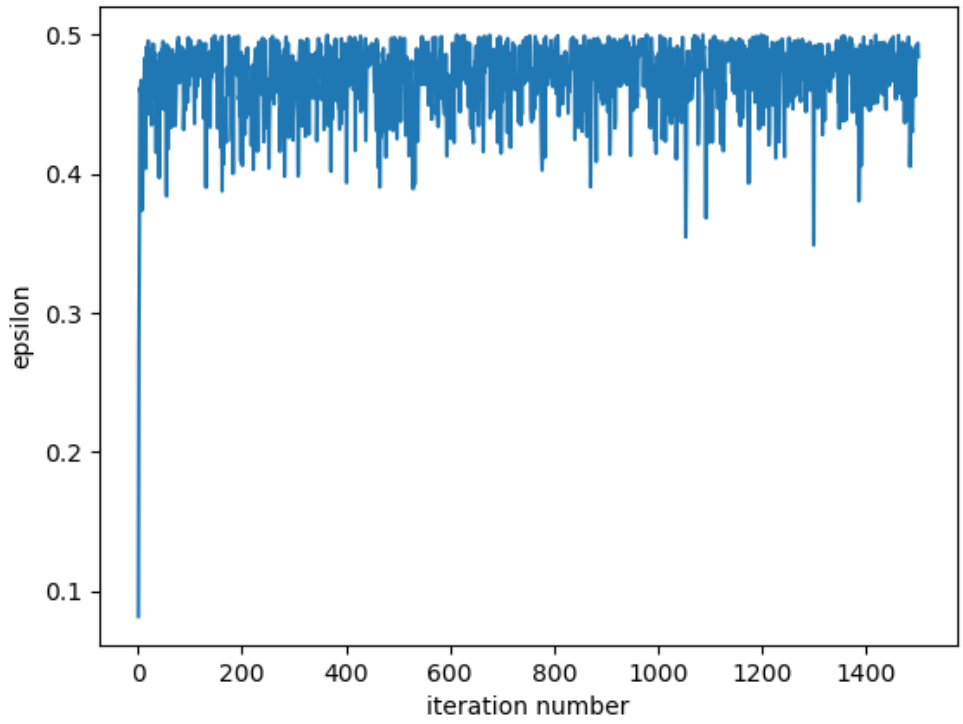


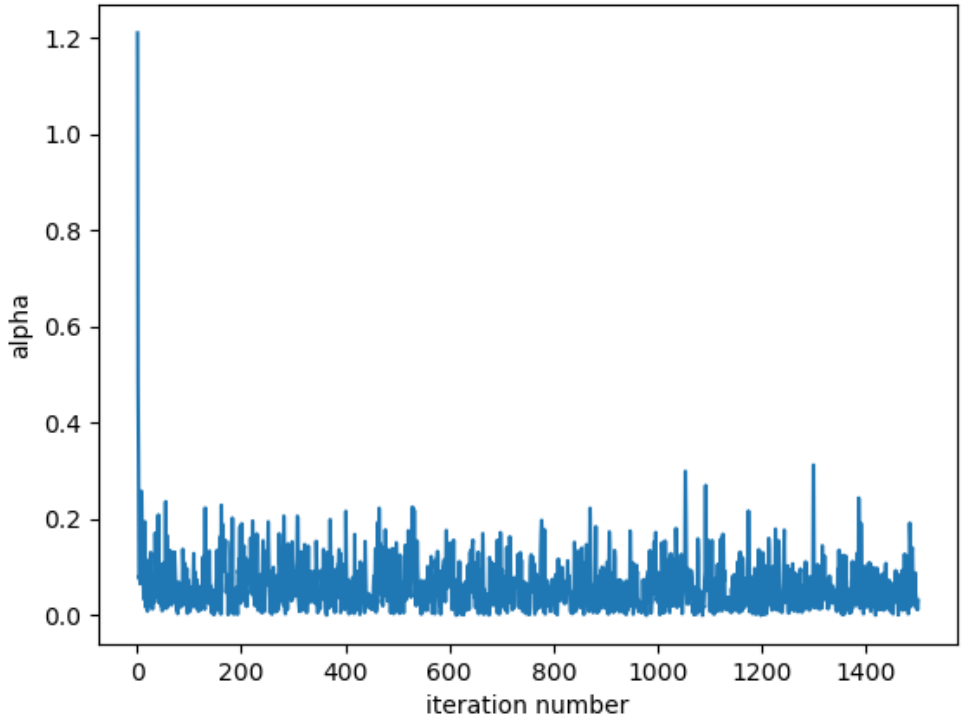
c) Histogram of the number of times a sample is selected by the boosting mechanism:



d) ϵ_t *and* α_t (as a function of t)




0.5
0.4
0.3
0.2
0.1
epsilon
0
200
400
600
800
1000
1200
1400
iteration number


1.2
1.0
0.8
0.6
0.4
0.2
0.0
alpha
0
200
400
600
800
1000
1200
1400
iteration number