Problem 1

a) Please refer to R file for the code

b) RMSE values in a table

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
5	1.966276	1.933135	1.923420	1.922198	1.924769	1.929213	1.934634	1.940583	1.946820	1.953213
7	1.920163	1.904877	1.908080	1.915902	1.924804	1.933701	1.942254	1.950380	1.958093	1.965438
9	1.897649	1.902519	1.917648	1.932514	1.945699	1.957235	1.967403	1.976492	1.984741	1.992341
11	1.890507	1.914981	1.938849	1.957936	1.973216	1.985764	1.996375	2.005603	2.013835	2.021345
13	1.895849	1.935586	1.964597	1.985502	2.001314	2.013878	2.024310	2.033307	2.041317	2.048642
15	1.909603	1.959549	1.990804	2.011915	2.027370	2.039465	2.049463	2.058105	2.065845	2.072976

c) In HW 1, the least value of the RMSE (HW 1 – Problem 2 Part 1c)), was **2.633644**. This corresponded to the value of lambda =0 i.e. the solution of the linear regression problem without regularization.

In HW 1 (Problem 2, Part 2, (d)), when we used a polynomial model, the lowest value of RMSE was approximately **2.2**

By using Gaussian Process, we find the min RMSE to be 1.890507 (the 11thvalue, in the 4th row and 1st column).

	row	col
11	4	1

Clearly, this value is **BETTER** than the lowest RMSE value in HW 1.

This value is achieved when **b=11**, and **sigma squared=0.1**

Drawbacks of this approach relative to HW 1

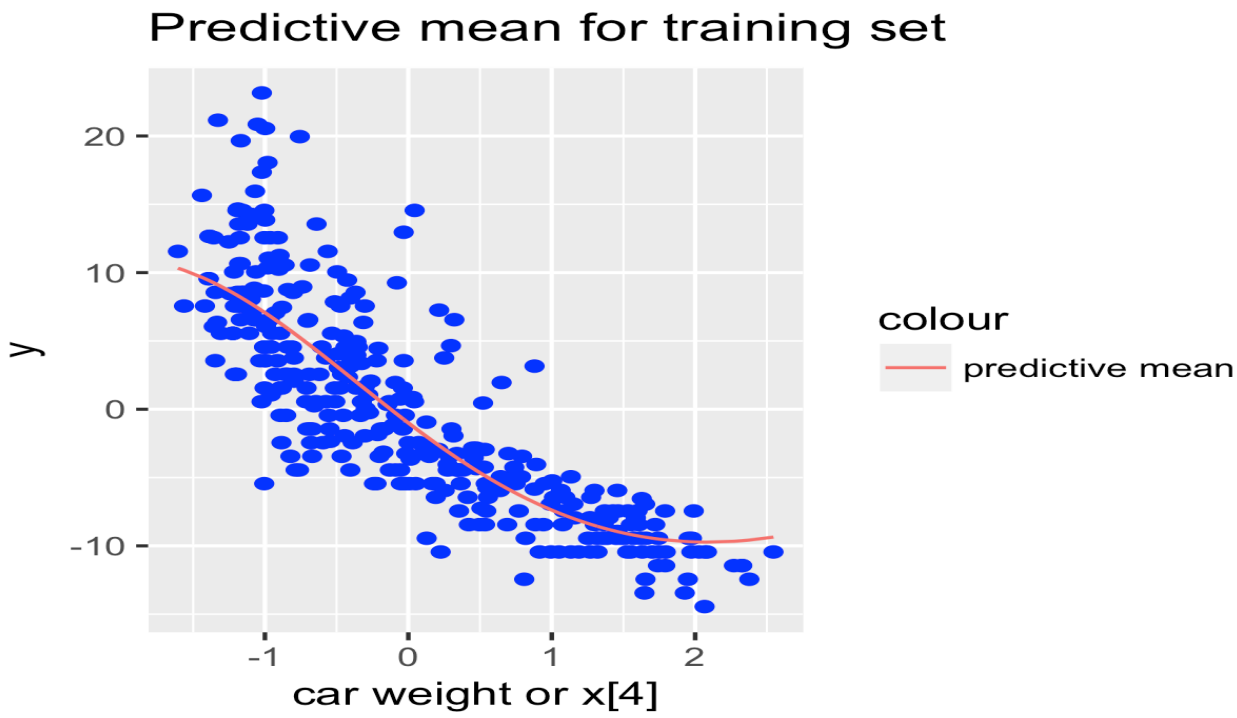
1. **Computationally**, this approach is **much more expensive**. To get each element of the mean vector of predictive distribution of the ys, each observation of the test set needs to be kernelized with the training data. This is in addition to the kernel matrix we need to create using the training data. The same process is required to compute the covariance matrix of the predictive distribution of the ys (although we haven't done that here).

Therefore, the order of magnitude of complexity is much higher here compared to solution in HW 1.

2. **Parameters need to be optimized numerically** – there is no elegant closed form solution as in the case of OLS. Lasso doesn't have a closed form solution but OLS and ridge do.

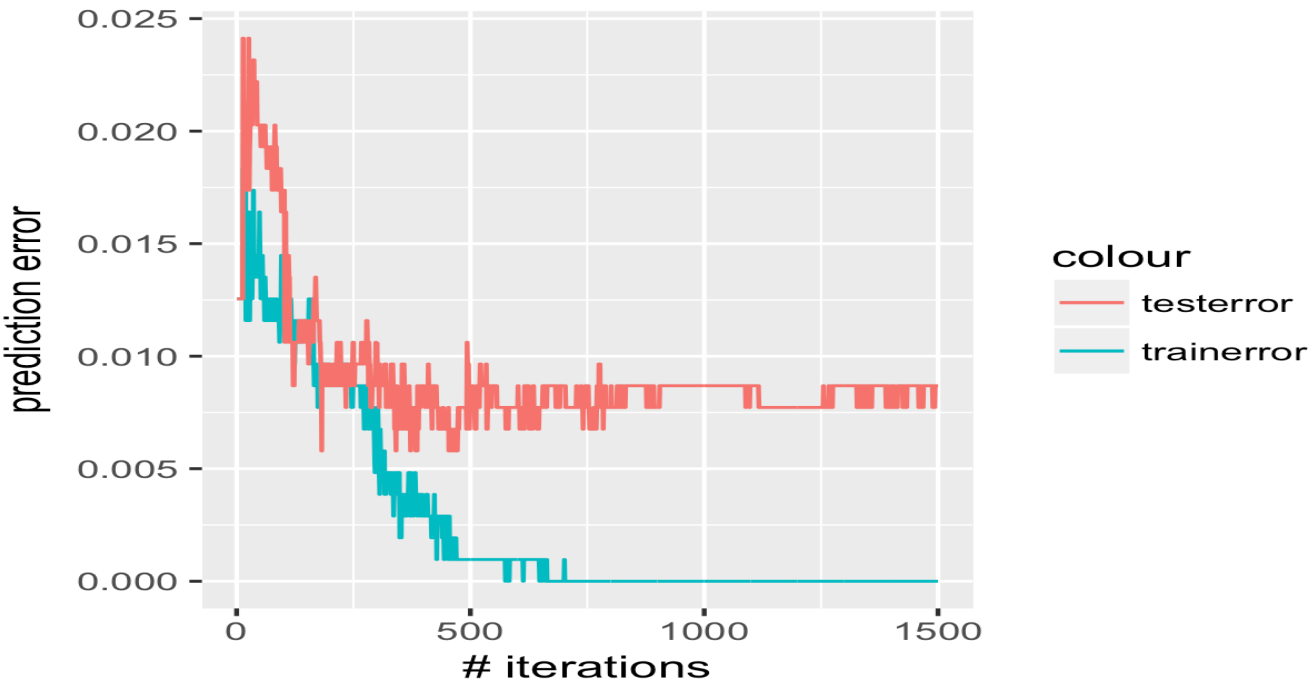
Ridge and Lasso need **one** parameter to be optimized (lambda). However in our current approach, we need to optimize **two** parameters (And we have fixed a third parameter, a=1). In our current problem we tried 60 different pairs. Therefore, it is much more expensive to determine the optimal values of the parameters through cross validation.

d)



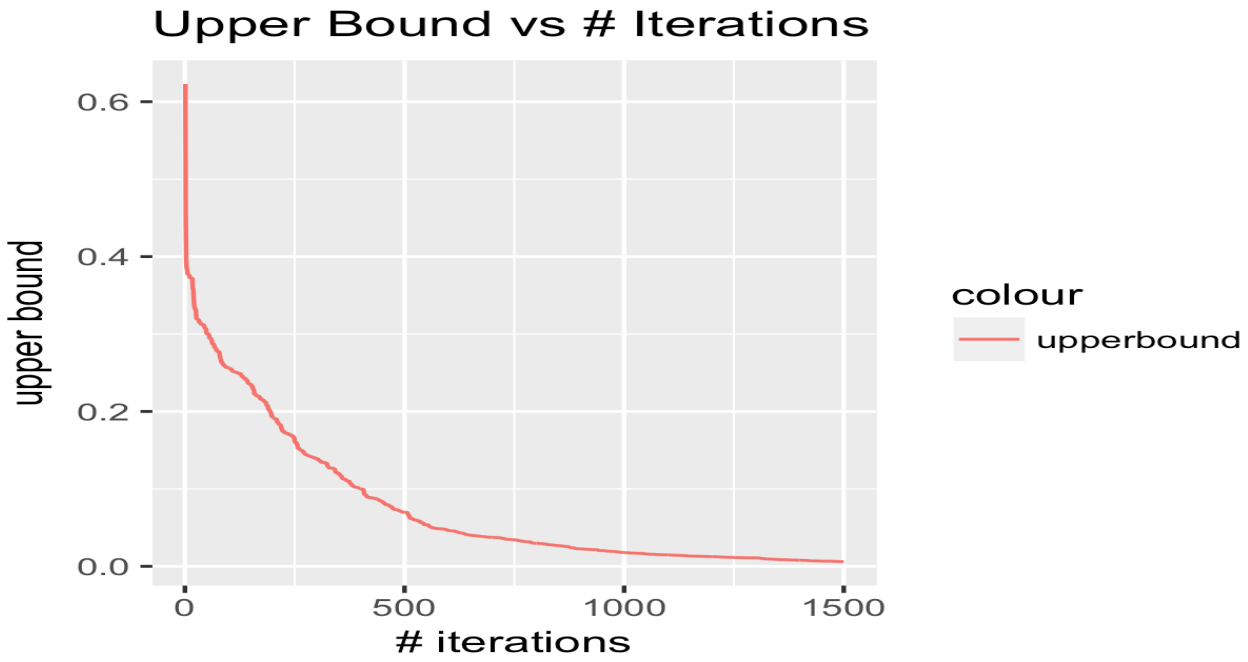
Problem 2 (Boosting)

a) We can see from the following plot that the training error converges to 0. The test error doesn't converge to 0. However, it decreases, *almost* (but not exactly) monotonically. This shows us that boosting does not overfit.

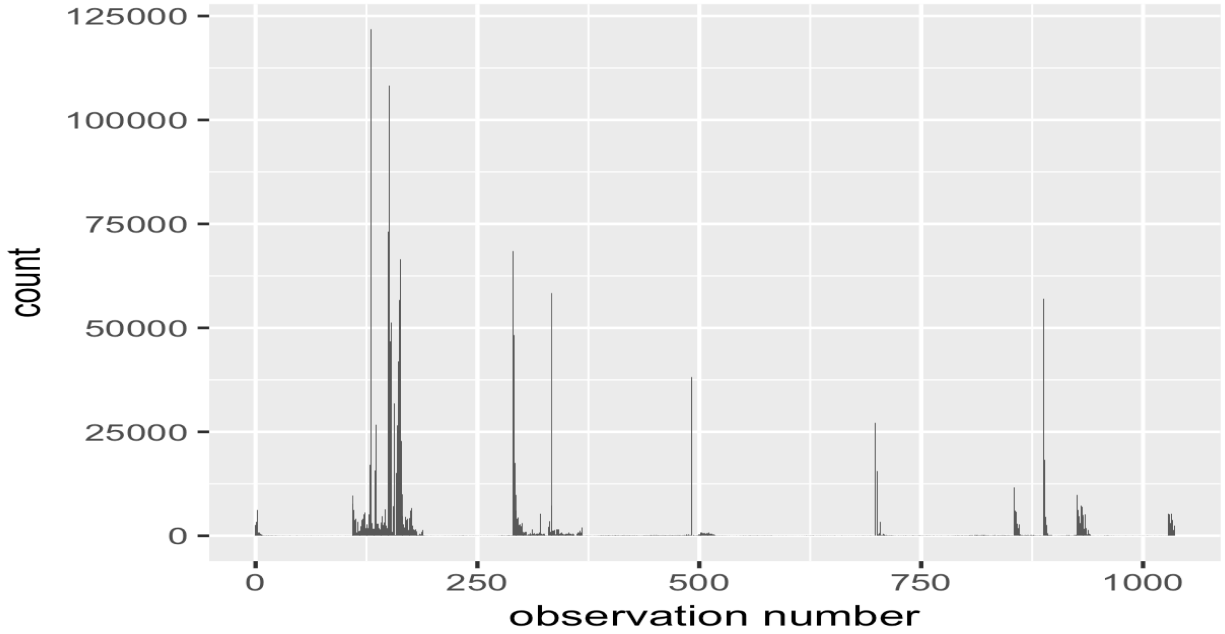


b) The upper bound as a function of t

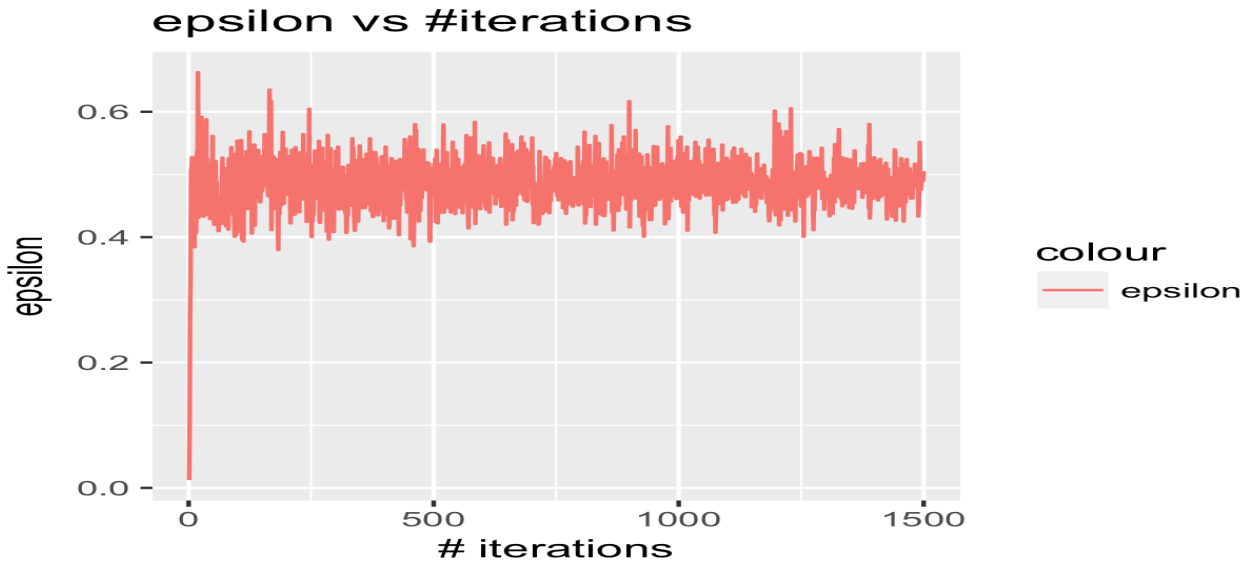
We find that the upper bound decreases monotonically as a function of t

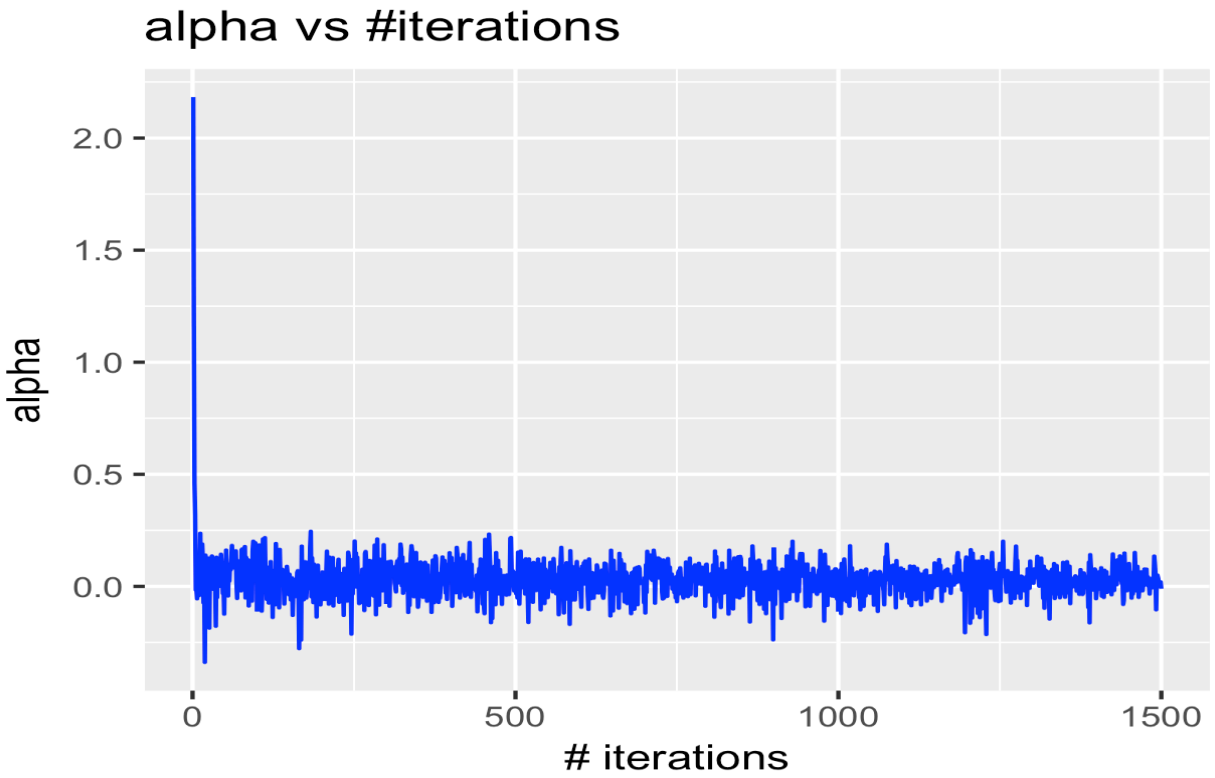


c) We notice that some points are selected much more often than others. These are the points that were being classified by our classifying model. And hence, they were picked most often because a higher weight was attached to them before the next iteration of boosting.



d)


epsilon vs #iterations
epsilon
0.6
0.4
0.2
0.0
0
500
1000
1500
iterations
colour
epsilon


alpha vs #iterations
alpha
2.0
1.5
1.0
0.5
0.0
0
500
1000
1500
iterations