Problem 1

a) Write code to implement the Gaussian process and to make predictions on test data.

```
class GaussianProcess(object):

    def kernel(self, x1, x2, b):
        return np.exp(-(np.linalg.norm(x1 - x2, 2)**2) / b)

    def fit(self, X, y, b):
        self.X = X.copy()
        self.y = y.copy()
        self.b = b
        self.Kn = np.zeros(shape = (len(X), len(X)))
        for i in range(len(X)):
            for j in range(len(X)):
                self.Kn[i][j] = self.kernel(X[i], X[j], b)

    def predict(self, X, sigma):
        K = np.zeros(shape = (len(X), len(self.X)))
        for i in range(len(X)):
            for j in range(len(self.X)):
                K[i][j] = self.kernel(X[i], self.X[j], self.b)
        self.mu = np.dot(np.dot(K, np.linalg.inv(np.add(sigma * np.eye(len(self.X)), self.Kn))), self.y)

    def score(self, y):
        diff = self.mu - y
        MSE = 1/len(y) * np.dot(np.transpose(diff), diff)
        return np.sqrt(MSE)
```

b) For b ∈ {5,7,9,11,13,15} and σ^2 ∈ {.1,.2,.3,.4,.5,.6,.7,.8,.9,1}—so 60 total pairs (b, σ^2)—calculate the RMSE on the 42 test points as you did in the first homework. Use the mean of the Gaussian process at the test point as your prediction. Show your results in a table.

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
5	1.97E+00	1.93E+00	1.92E+00	1.92E+00	1.92E+00	1.93E+00	1.93E+00	1.94E+00	1.95E+00	1.95E+00
7	1.92E+00	1.90E+00	1.91E+00	1.92E+00	1.92E+00	1.93E+00	1.94E+00	1.95E+00	1.96E+00	1.97E+00
9	1.90E+00	1.90E+00	1.92E+00	1.93E+00	1.95E+00	1.96E+00	1.97E+00	1.98E+00	1.98E+00	1.99E+00
11	1.89E+00	1.91E+00	1.94E+00	1.96E+00	1.97E+00	1.99E+00	2.00E+00	2.01E+00	2.01E+00	2.02E+00
13	1.90E+00	1.94E+00	1.96E+00	1.99E+00	2.00E+00	2.01E+00	2.02E+00	2.03E+00	2.04E+00	2.05E+00
15	1.91E+00	1.96E+00	1.99E+00	2.01E+00	2.03E+00	2.04E+00	2.05E+00	2.06E+00	2.07E+00	2.07E+00

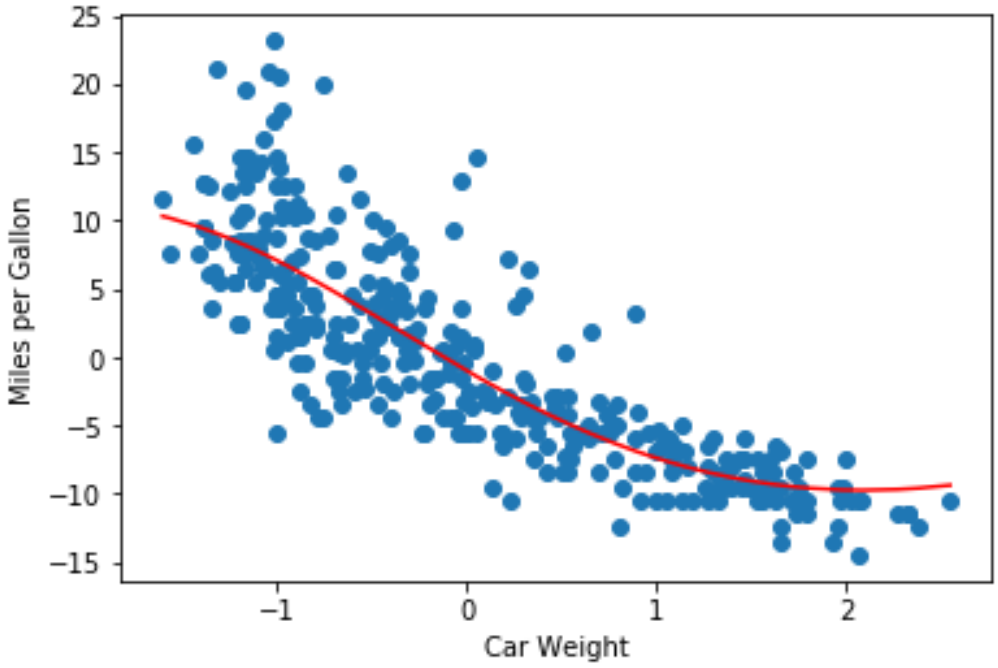
c) Which value was the best and how does this compare with the first homework? What might be a drawback of the approach in this homework (as given) compared with homework 1?

The best RMSE value of 1.89 was achieved when $b = 11, \sigma^2 = 0.1$. This is lower compared to the best RMSE value achieved using ridge regression in the first assignment which was about 2.2.

Some drawbacks of this approach are:

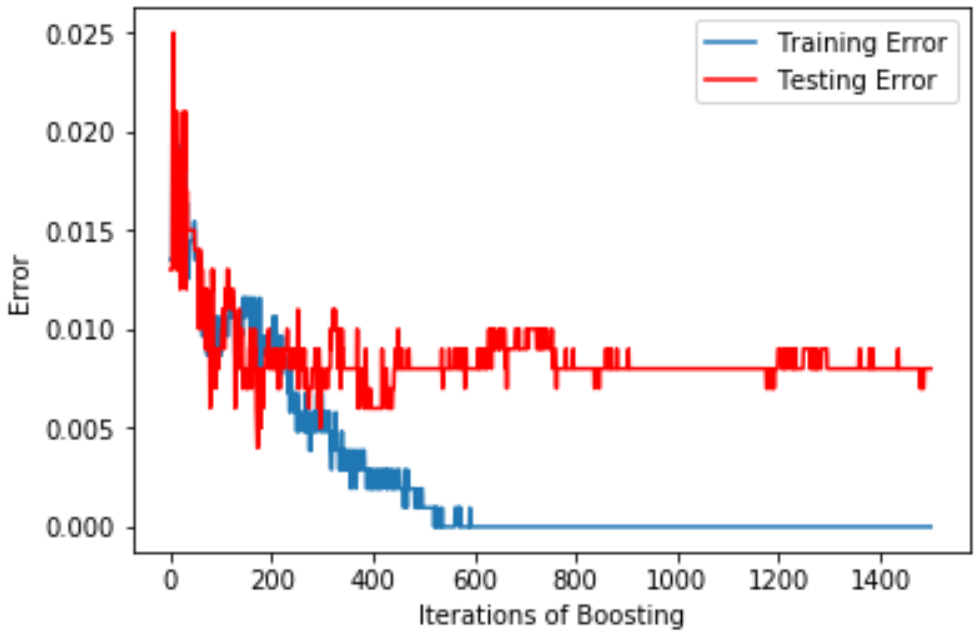
- Computational cost: the closed form solution for a gaussian process requires a matrix inversion of an $n \times n$ matrix which has a complexity of $O(n^3)$ and so will quickly become computationally infeasible as n approaches extremely large values.
- Overfitting: the Gaussian process maps the data to infinite dimensional Hilbert space where there is likely to be a separating hyperplane. This requires careful selection of the regularization parameter to avoid overfitting.

d)

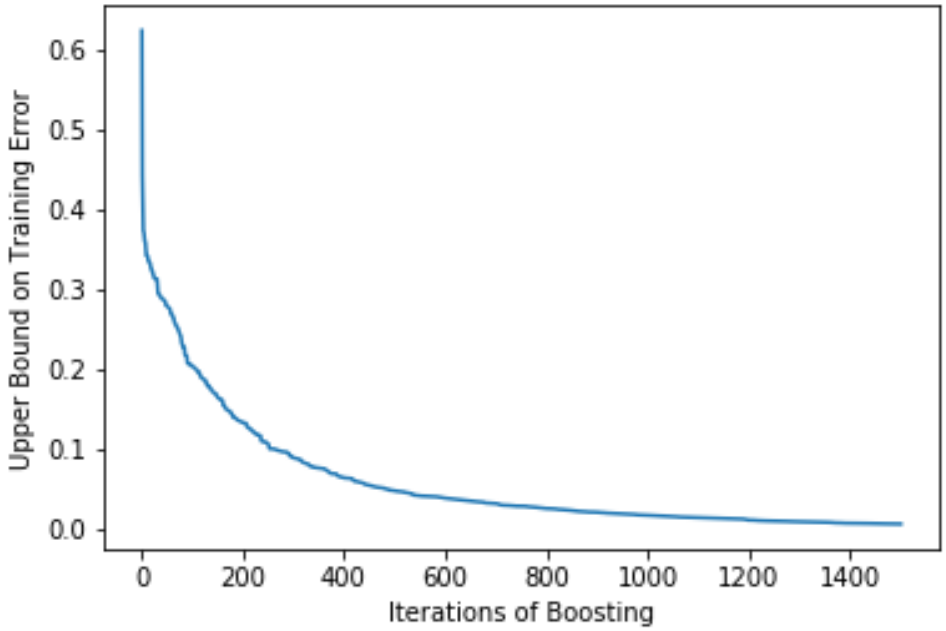


Problem 2

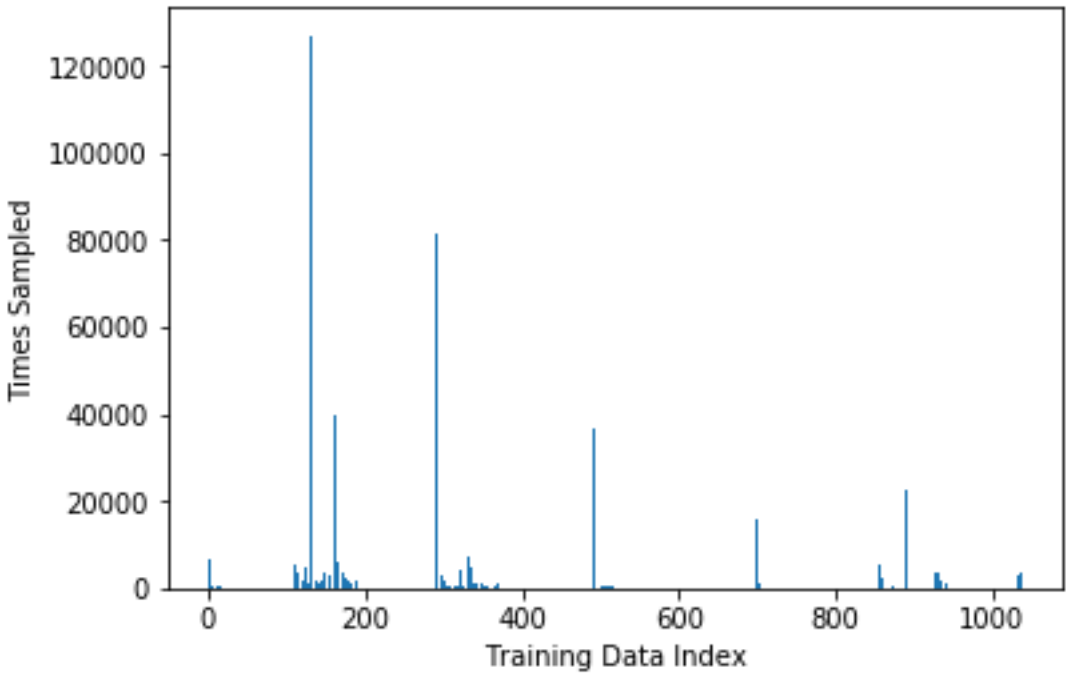
a) The plot below shows the training error (blue) and testing error (red) of the $f_{boost}^{(t)}(\cdot)$ for $t = 1, \ldots, T$.



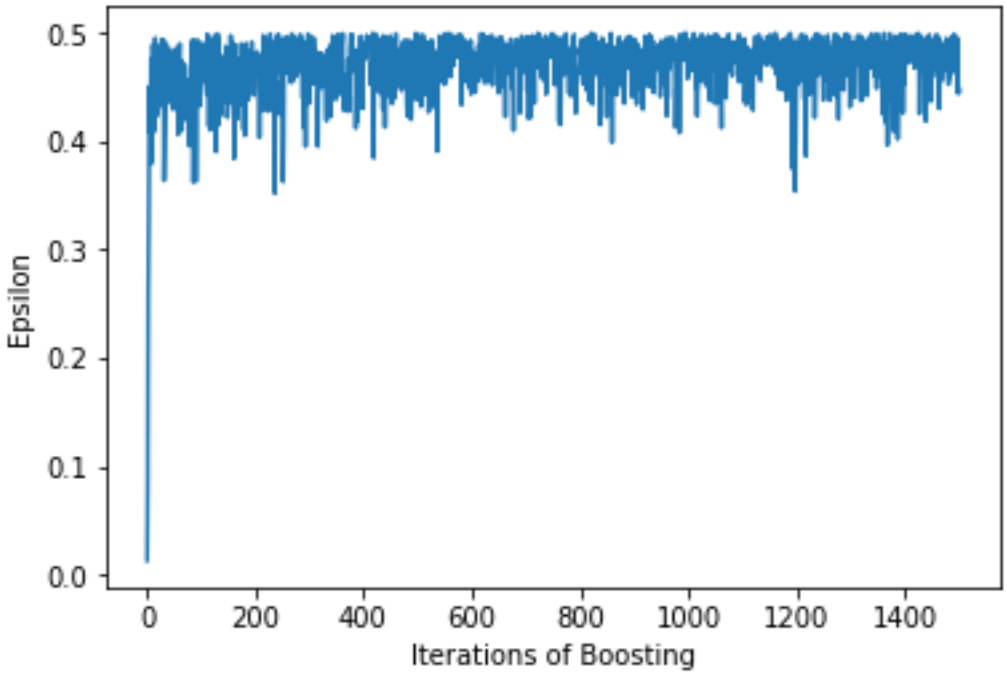
b) The plot below shows the upper bound of the training error against the number of boosting iterations.



c) The histogram below shows the frequency of selection of each data point in the training set during the boosting process.



d) The plot below shows the value of epsilon against iterations of boosting.



The plot below shows the value of alpha against iterations of boosting.