ISYE 6740 Homework 6

1. **Conceptual questions**

   1.1. The main difference between boosting and bagging is that boosting runs weak learner on weighted example (training) set, and then combines their outputs linearly based on the weak hypotheses. It require knowledge on the performance of weak learner.

   On the other hand, bagging trains multiple weak learners on different random samples of the training set, averages weak learners predictions and reduces variance.

   Random Forest is the type of bagging because it builds lots of decision trees on random samples, and adds some randomness in feature selection to make the trees less similar so that to improve the model performance.

   1.2. One of the commonly used methods to prevent overfitting in CART is tree pruning, as large tree may overfit data and small trees underfits. Grow a large tree and stop when a minimum node size has reached, then perform pruning by minimizing the cost function can help prevent the overfitting. Another method is to use cross-validation to tune parameters and then to select the best performing models. The third method is to perform feature selection to include the most relevant features. The above three methods are usually used to prevent overfitting in CART.

   1.3. In the regression tree, we can control the data-fit complexity by tree pruning. The method involves growing a large tree and stop when a minium nod size has reached, then pruning the regression tree by minimizing the cost function, so that to stop it from becoming too complex and overfit the training set, as stated in question 2.

   As referred to in the website 'GeeksforGeeks - How to Tune a Decision Tree in Hyperparameter Tuning' https://www.geeksforgeeks.org/how-to-tune-a-decision-tree-in-hyperparameter-tuning/ , one of the hyperparameters is maximum depth. It controls the maximum depth to the regression tree. It allows the tree to capture much more complex patterns in the training dataset, and reducing the errors in the training dataset. To avoid overfitting the model, it is important to tune it so it's not getting too complex while still provides good performance.

   1.4. OOB errors in random forest estimates how well the model performs on predicted dataset. When training takes place, each of the regression trees is trained on the bootstrapped samples. Then the remaining data points, which are OOB (out of bag), are used to evaluate the each of the trees. OOB errors are calculated while comparing the predictions to the actual labels. It gives a good idea of the model's test error without the need to create a separate testset. Due to the nature illustrated above, OOB errors helps in determining the number of trees. Also due to the reason explained above, it is more of an error test rather than training error even it uses the training dataset, as OOB errors uses new data points rather than the same data used in the training process.

1.5. To explain what the bias-variance tradeoff means in the linear regression, there are two types of errors in the linear regression. The first type is bias error that when the assumptions about the data are too simple that cause the important patterns missing, which is called underfitting in such scenario. The second type of error is variance error, which comes from the model being overly sensitive to the training data only, and it is difficult to fit the remaining or new dataset, which is the overfitting scenario. For example, a simple model such as a straight line might have high bias and low variance. On the other hand, a more complex model with polynomials as illustrated in the lecture may have less bias but may come with increased variance. Because we need to avoid the model being underfitting or overfitting, to find a balance between bias error and variance error is important to find the best performing linear regression model.

2. **AdaBoost.**

2.1. To run through 3 iterations of AdaBoost with decision stump on the dataset, we can start with construct $D_t : t = 1, 2, 3$ and initiate $D_1(i)$ for i when $i = 1, 2, 3, 4, 5, 6, 7, 8$:

$$D_1(i) = \frac{1}{m} = \frac{1}{8} = 0.125$$

Round 1 iteration - given D_1 decide weights for classifier ϵ_1

$$\epsilon_1 = \sum_{i=1}^{m} D_1(i)\mathbb{I}\{y^i \neq h_1(x_i)\} = 0.125 * (1 + 1) = 0.25$$

Compute α_1:

$$\alpha_1 = \frac{1}{2} \ln\left(\frac{1-\epsilon_1}{\epsilon_1}\right) = \frac{1}{2} \ln\left(\frac{1-0.3}{0.3}\right) = 0.549(rounded)$$

Compute normalizing constant Z_1:

$$Z_1 = \sum_{i=1}^{m} D_1(i)e^{-\alpha_1 y^i h_1(x^i)} = 0.125 * (6e^{-0.5493} + 2e^{0.5493}) = 0.866$$

Compute the weights for D_2:

$$D_2 i) = \left(\frac{D_1(i)}{Z_1}\right) e^{-\alpha_t y^i h_t(x^i)}$$

If $y^i = h_t(x^i)$ when $i = 5, 6$:

$$D_2(i) = \left(\frac{D_1(i)}{Z_1}\right) e^{-\alpha_1} = \frac{0.125}{0.866} \cdot e^{-0.5493} = 0.0833$$

Otherwise, $i \neq 5, 6$:

$$D_2(i) = \left(\frac{D_1(i)}{Z_1}\right) e^{-\alpha_1} = \frac{0.125}{0.866} \cdot e^{0.5493} = 0.250$$

Round 2 iteration

Choose the decision stump and compute ϵ_2:

$$\epsilon_2 = \sum_{i=1}^{m} D_2(i)\mathbb{I}\{y^i \neq h_2(x_i)\} = 2 * 0.0833 = 0.167$$

Compute α_1

$$\alpha_2 = \frac{1}{2} \ln\left(\frac{1-\epsilon_2}{\epsilon_2}\right) = \frac{1}{2} \ln\left(\frac{1-0.167}{0.167}\right) = 0.805$$

Compute normalizing constant Z_2:

$$Z_2 = \sum_{i=1}^{m} D_2(i)e^{-\alpha_2 y^i h_2(x^i)} = 2 * 0.0833 * e^{0.805} + (4 * 0.0833 + 2 * 0.25) * e^{-0.805} = 0.745$$

Compute the weights for D_3:

$$D_3(i) = \left(\frac{D_2(i)}{Z_2}\right) e^{-\alpha_2 y^i h_2(x^i)}$$

$$D_3(1) = D_3(2) = \frac{0.0833}{0.745} \cdot e^{0.805} = 0.25$$

$$D_3(5) = D_3(6) = \frac{0.25}{0.745} \cdot e^{-0.805} = 0.15$$

$$D_3(3) = D_3(4) = D_3(7) = D_3(8) = \frac{0.0833}{0.745} \cdot e^{-0.805} = 0.05$$

Round 3 iteration
Choose the decision stump and compute ϵ_3:

$$\epsilon_3 = \sum_{i=1}^{m} D_3(i)\mathbb{I}\{y^i \neq h_3(x_i)\} = 2 * 0.05 = 0.1$$

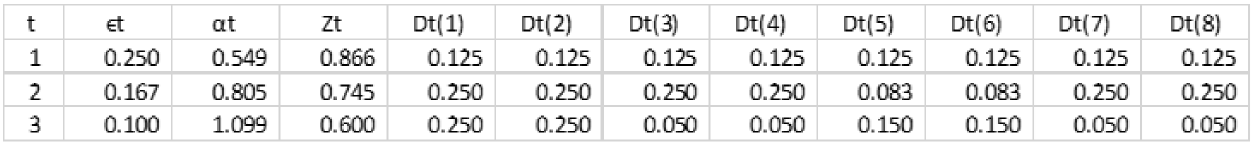
Compute α_3:

$$\alpha_3 = \frac{1}{2}\ln\left(\frac{1-\epsilon_3}{\epsilon_3}\right) = \frac{1}{2}\ln\left(\frac{1-0.1}{0.1}\right) = 1.099$$

Compute normalizing constant Z_3:

$$Z_3 = \sum_{i=1}^{m} D_3(i)e^{-\alpha_3 y^i h_3(x^i)} = 2{*}0.05{*}e^{1.099}+2{*}0.25{*}e^{-1.099}+2{*}0.05{*}e^{-1.099}+2{*}0.15{*}e^{-1.099} = 0.6$$

As shown in Figure 1 below is the results of AdaBoost at each timestep:



t	εt	αt	Zt	Dt(1)	Dt(2)	Dt(3)	Dt(4)	Dt(5)	Dt(6)	Dt(7)	Dt(8)
1	0.250	0.549	0.866	0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125
2	0.167	0.805	0.745	0.250	0.250	0.250	0.250	0.083	0.083	0.250	0.250
3	0.100	1.099	0.600	0.250	0.250	0.050	0.050	0.150	0.150	0.050	0.050

Figure 1: Values of AdaBoost parameters at each timestep

Compute final classifier:

$$H(x) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) = \operatorname{sign}\left(0.549 \cdot h_1(x) + 0.805 \cdot h_2(x) + 1.099 \cdot h_3(x)\right)$$

$$H(1) = \operatorname{sign}\left(0.549 \cdot (+1) + 0.805 \cdot (+1) + 1.099 \cdot (+1)\right) = \operatorname{sign}\left(0.549 + 0.805 + 1.099\right) = \operatorname{sign}\left(2.453\right) = 1$$
$$H(2) = \operatorname{sign}\left(0.549 \cdot (+1) + 0.805 \cdot (+1) + 1.099 \cdot (-1)\right) = \operatorname{sign}\left(0.549 + 0.805 - 1.099\right) = \operatorname{sign}\left(0.255\right) = 1$$
$$H(3) = \operatorname{sign}\left(0.549 \cdot (-1) + 0.805 \cdot (+1) + 1.099 \cdot (-1)\right) = \operatorname{sign}\left(-0.549 + 0.805 - 1.099\right) = \operatorname{sign}\left(-0.843\right) = -1$$
$$H(4) = \operatorname{sign}\left(0.549 \cdot (-1) + 0.805 \cdot (-1) + 1.099 \cdot (-1)\right) = \operatorname{sign}\left(-0.549 - 0.805 - 1.099\right) = \operatorname{sign}\left(-2.453\right) = -1$$
$$H(5) = \operatorname{sign}\left(0.549 \cdot (+1) + 0.805 \cdot (-1) + 1.099 \cdot (+1)\right) = \operatorname{sign}\left(0.549 - 0.805 + 1.099\right) = \operatorname{sign}\left(0.843\right) = 1$$
$$H(6) = \operatorname{sign}\left(0.549 \cdot (+1) + 0.805 \cdot (-1) + 1.099 \cdot (+1)\right) = \operatorname{sign}\left(0.549 - 0.805 + 1.099\right) = \operatorname{sign}\left(0.843\right) = 1$$
$$H(7) = \operatorname{sign}\left(0.549 \cdot (-1) + 0.805 \cdot (+1) + 1.099 \cdot (-1)\right) = \operatorname{sign}\left(-0.549 + 0.805 - 1.099\right) = \operatorname{sign}\left(-0.843\right) = -1$$
$$H(8) = \operatorname{sign}\left(0.549 \cdot (-1) + 0.805 \cdot (-1) + 1.099 \cdot (-1)\right) = \operatorname{sign}\left(-0.549 - 0.805 - 1.099\right) = \operatorname{sign}\left(-2.453\right) = -1$$

Each decision stumps are as shown in Fugure 2 - 5 below:

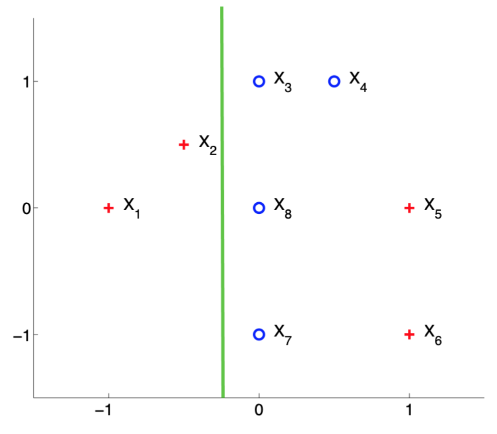


Figure 2: Round 1

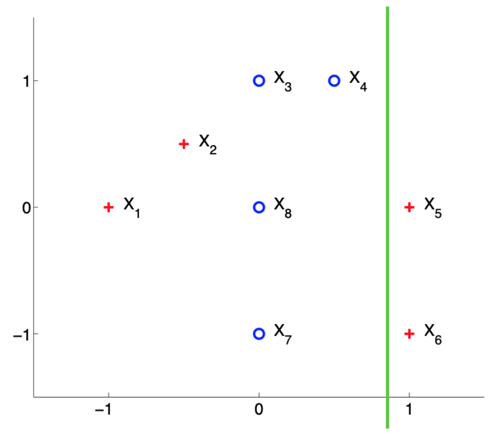


Figure 3: Round 2

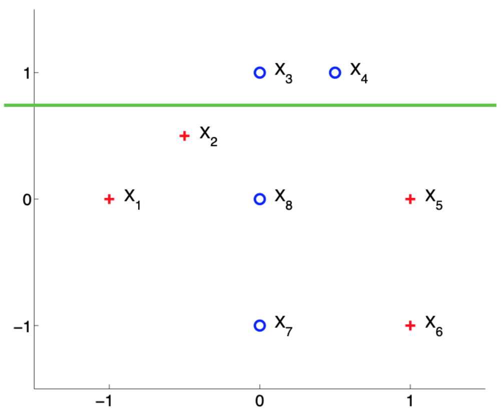


Figure 4: Round 3

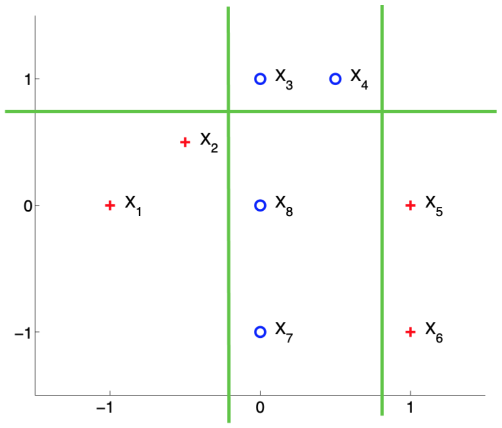


Figure 5: Final classifier

2.2. It is observed that all data points have been classified correctly, as shown in the Figures above, therefore the training error of this AdaBoost is 0. We can draw the conclusion that AdaBoost outperforms a single decision stump because it combines a number of weak classifiers to a strong classifier, and it greatly improved the model accuracy.

3. **Random forest and one-class SVM for email spam classifier**

3.1. Below as shown in Figure 6 is the plot for CART odel. During the plot process, I limited the max depth, and split only when there are enough samples, and each node contains at least 5 samples to help reduce the overfitting and simplify the model.

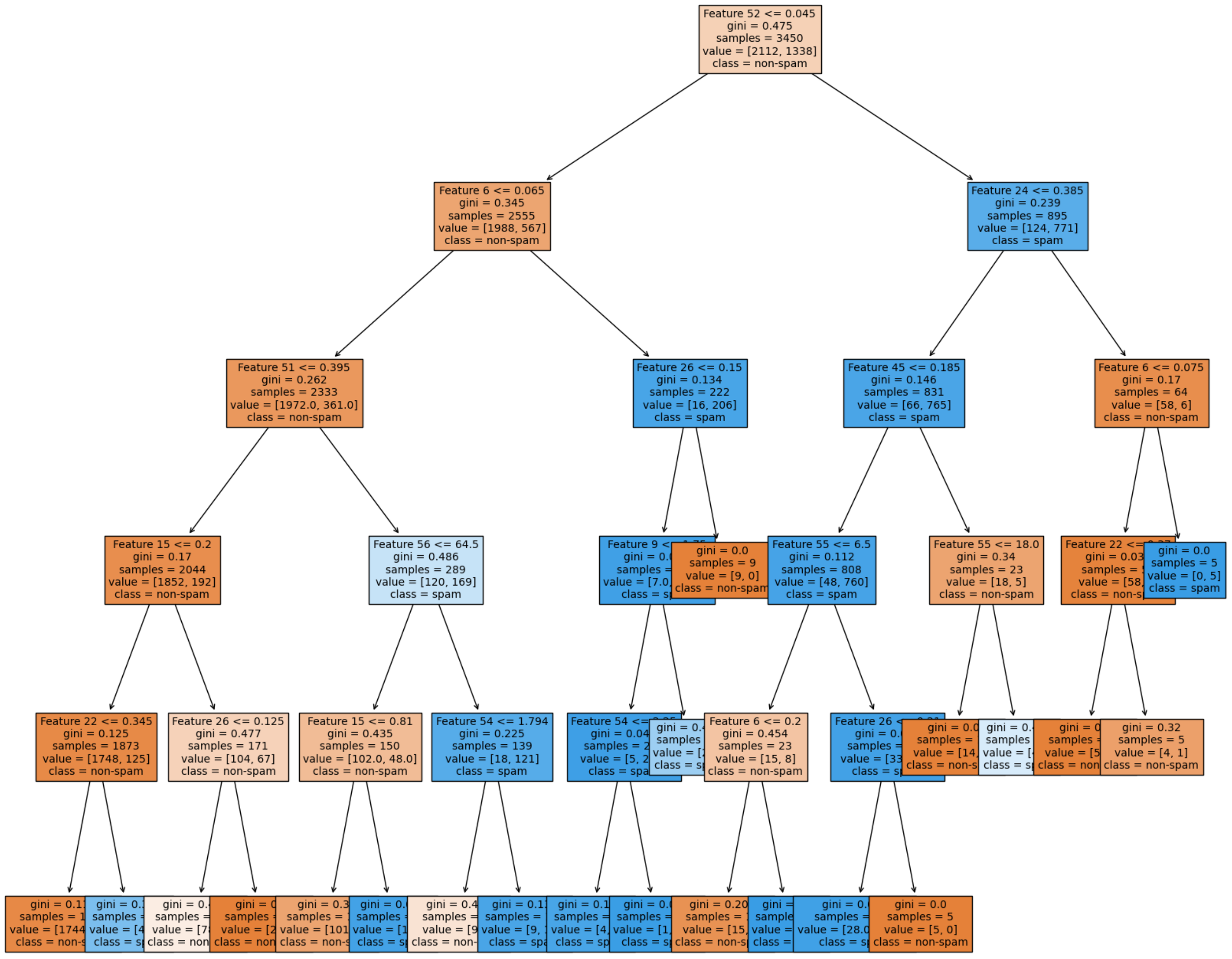


Figure 6: CART

3.2. Building a random forest model using Python, the random forest test error is 0.04170286707211121, while CART test error is 0.08514335360556038. Plot of the test error for the CART model and random forest is as shown in Figure 7. From where we observed that CART is a straight line, while random forest error is zig zag.

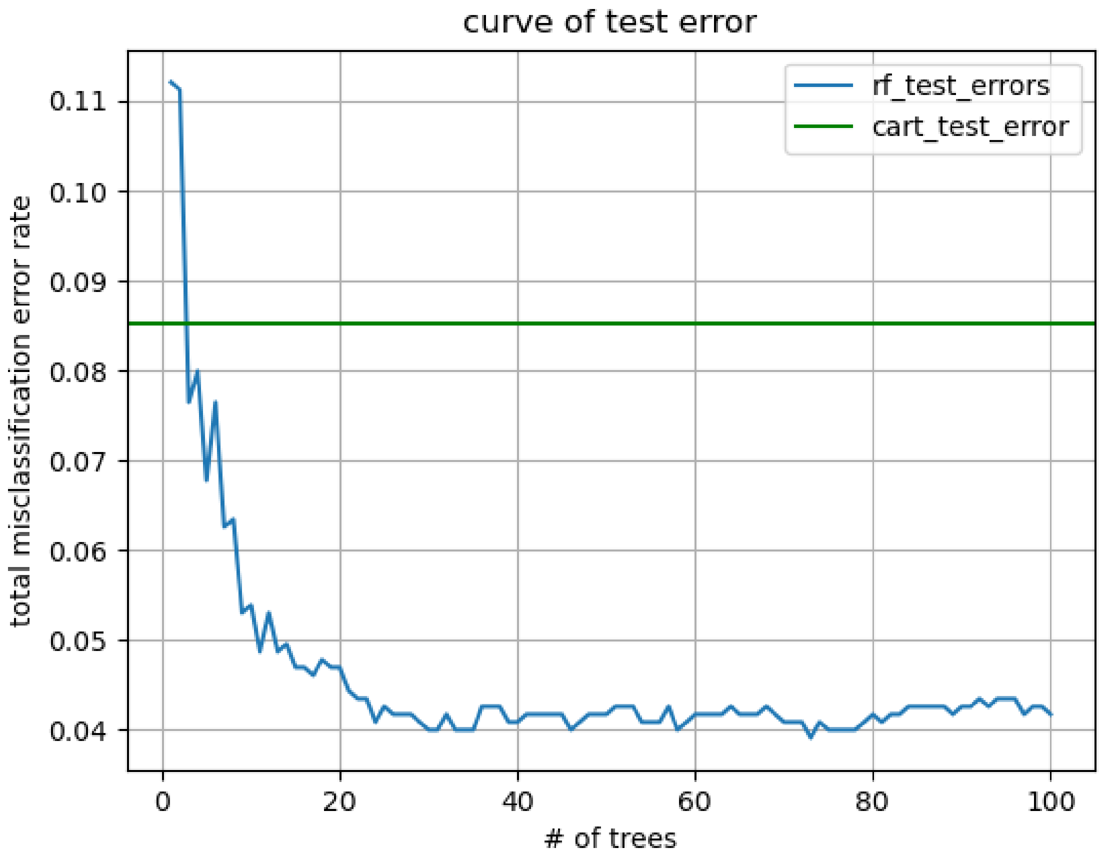


Figure 7: Curve of Test Error

3.3. Plot of both the OOB error as well as the test error against a suitably chosen range of values for ν, as shown in Figure 8 below

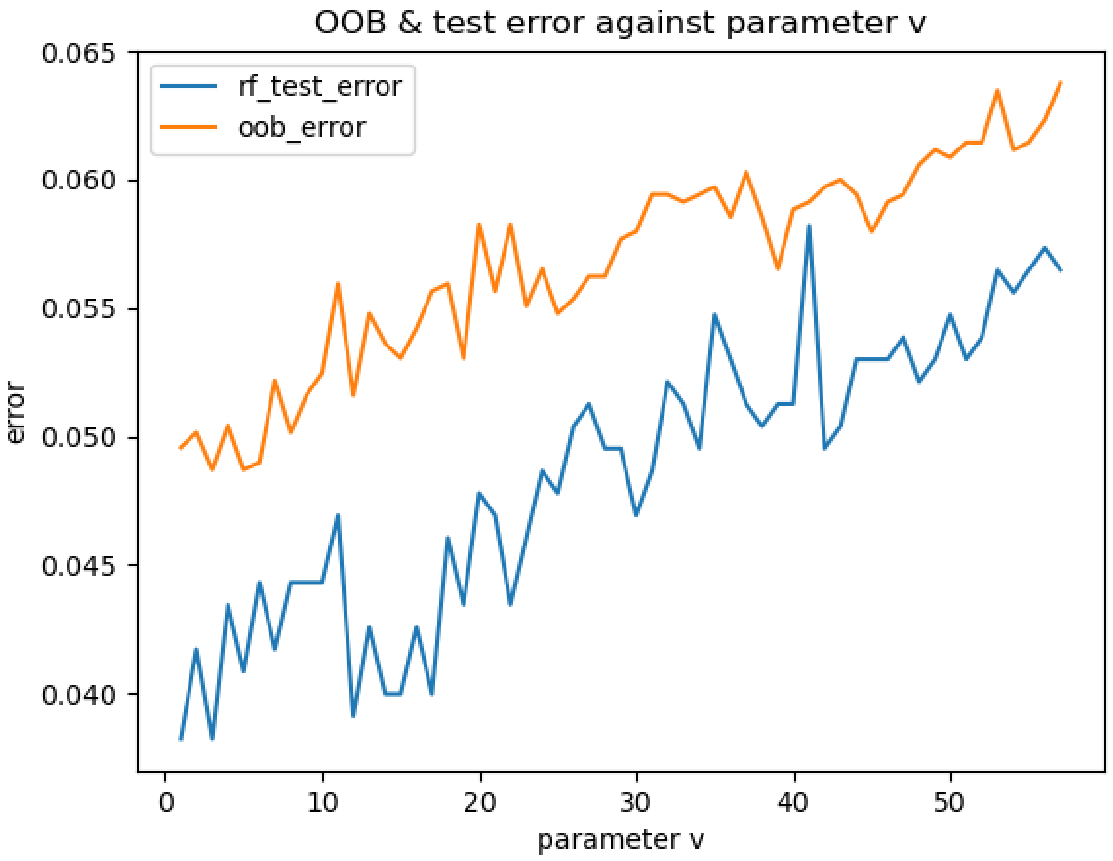


Figure 8: OOB and test error against parameter v

3.4. First, I extracted the non-spam emails from the training data, specifically for the features `'word_freq_george'` and `'word_freq_650'`. Then, I scaled the data and built the one-class SVM model using Python. During this process, I converted the SVM predictions to 1 for non-spam and -1 for spam. Afterward, I converted these predictions into 0 for non-spam and 1 for spam. Finally, I computed the classification error rate as the average of incorrect predictions. The result I obtained is approximately 0.323.

4. **Locally weighted linear regression and bias-variance tradeoff.**

4.1. To find the solution to the local linear regression problem, we can start with defining matrices X, W, and Y to be what they need to be.

Design matrix X is defined as:

$$X = \begin{pmatrix} 1 & (x - x_1)^T \\ 1 & (x - x_2)^T \\ \vdots & \vdots \\ 1 & (x - x_n)^T \end{pmatrix}$$

Weight W is defined as:

$$W = \begin{pmatrix} K_h(x - x_1) & 0 & \cdots & 0 \\ 0 & K_h(x - x_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & K_h(x - x_n) \end{pmatrix},$$

Response Y is:

$$Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

The objective function in matrix form can be rewritten as the following when $\beta := (\beta_0, \beta_1)$:

$$\sum_{i=1}^{n} (y_i - \beta_0 - (x - x_i)^T \beta_1)^2 K_h(x - x_i) = (Y - X\beta)^T W (Y - X\beta)$$

Taking the derivative and setting it to zero to find the minimizer:

$$\frac{\partial}{\partial \beta} (Y - X\beta)^T W (Y - X\beta) = -2X^T W (Y - X\beta) = 0 \implies X^T W Y = X^T W X \beta$$

So that the solution is:

$$\widehat{\beta} = (X^T W X)^{-1} X^T W Y$$

4.2. First we write y hat, which is B hat zero x. From where we get predicted loss of the model for t he data point x. With this setting, we do 5-fold cross validation. Randomly split the training dataset into 5 subsets. For each choice bandwidth H, take the 4 subsets as training data to estimate the parameter, Use left out set as test data and record the test error. Iterate all the subset train test combinations and report the average test data as the error rate for the chosen bandwidth H. Then we draw the cross validation curve to find the optimal bandwidth from there, as shown in Figure 9 below, the optimal bandwidth is 0.2.

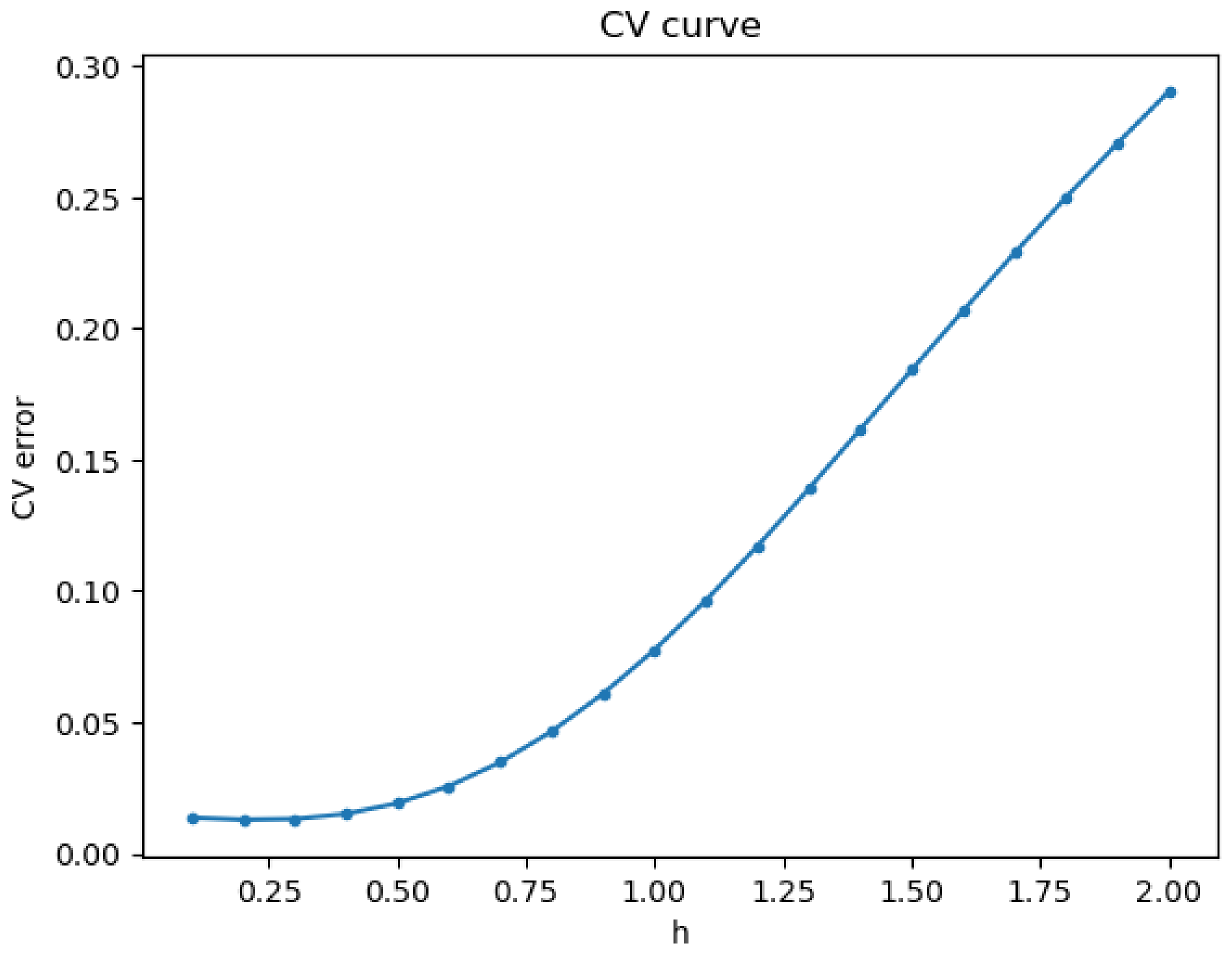


Figure 9: CV curve

4.3. Now we use the tuned hypsometer H to make prediction for x equal to the value. Then we have the test data for x equal to this. From there we get the value of y hat. We draw the curve for x equal to the value, and we will see what the prediction is and what the training data is. We draw another curve to see what the predicted value is, as shown in Figure 10 below, predicted y value for x= -1.5 is approximately 1.8.

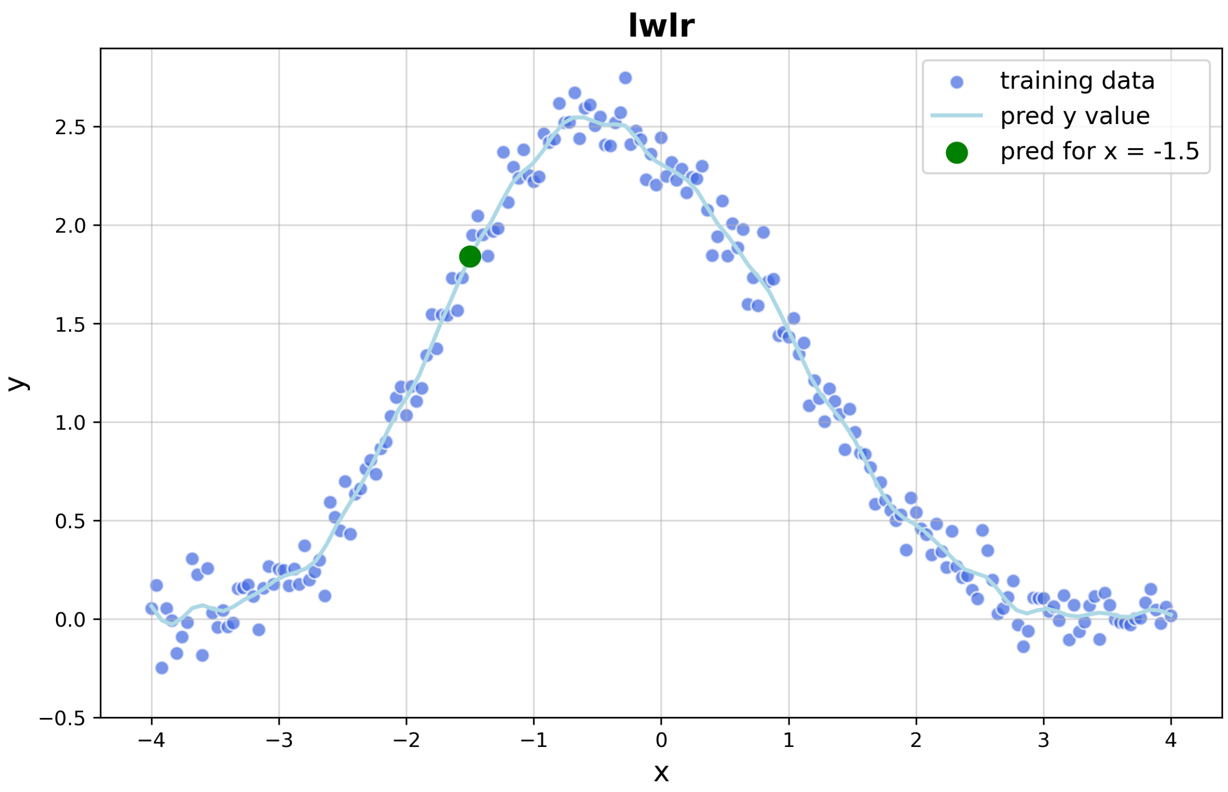


Figure 10: Training data, prediction curve