# Project II

Filip Axelsson

2021-12-02

**Task 1**

**A)**

I am using the package "gbm" to preforme binary classsification using gradient boosting trees. The function is called "gbm()" which can be used with different loss function. For most classification problems either bernoulli or adaboost will be appropriate, however the package recommend bernouilli. In this case we are setting the parameter "distrubution = bernoulli" since it is the best one to use for this task. When using "distrubution = bernoulli", we using the gradient: $z_i = y_i - \frac{1}{1+exp(-f(x_i))}$ and the deviance: $-2\frac{1}{\sum w_i}\sum w_i(y_i f(x_i) - log(1 + exp(f(x_i))))$. Note that the vector for the weights is selected by default by the package. In *Table 1* is an overview of some parameters that could be changed in the function, however $shrinkage = 0.1$ by default, and $n.minobsinnode = 10$. The package "caret" is also used to preforme cross-validation, this is aldready a built-in function at gbm, however we need to plot the standard error bars which is not possible. Therefore is the package "caret" and its function "train()" which combinded with "gmb()" can be used so we can plot those bars.

Table 1: List over parameters in the function gbm()

| Parameter | Information |
|---|---|
| distrubution | Specifying the name of the distribution |
| n.trees | Specifying the total numbers of trees to fit |
| interaction.depth | Specifying the maximum depth of each tree |
| n.minobsinnode | Specifying the minimum number of observations in the terminal nodes of the trees |
| shrinkage | The learning rate |
| cv.folds | Number of cross-validation folds to perform |

**B)**

By creating the fucntion "gb_trees" we can now easy construct gradient boosting trees with different number of nodes and total number of trees. We enable the user to set a list of nodes that should be used, and also the amount of trees that should be grown. The function then loops over the given list of nodes and then setting the parameter "interaction.depth" to that specific value in the "gbm()" function, the parameter "n.trees" will be set to the given value. The rest of the parameters that specify the loss function is already set and the rest of the parameters are by defualt given. Note that the amount of nodes is maximal 49, more than that will provide an error message stating that a value between 1 and 5 should be more sufficient. The error message can be seen in the code in the appendix.

**C)**

By building a new function, "cv_gbm", we can preform cross-validation. The new function is needed becuase cross-validation in the function "gbm()" doesn't provide the information that is needed to plot the cross-validation error agains the number of boosting trees $M$. We can see the result of each cross-validation for each $m$ and different $M$ in *Figure 1*,*Figure 2*, *Figure 3* and *Figure 4* below.

Figure 1: Cross−validation with m = 1 and M = 50,100,200,500

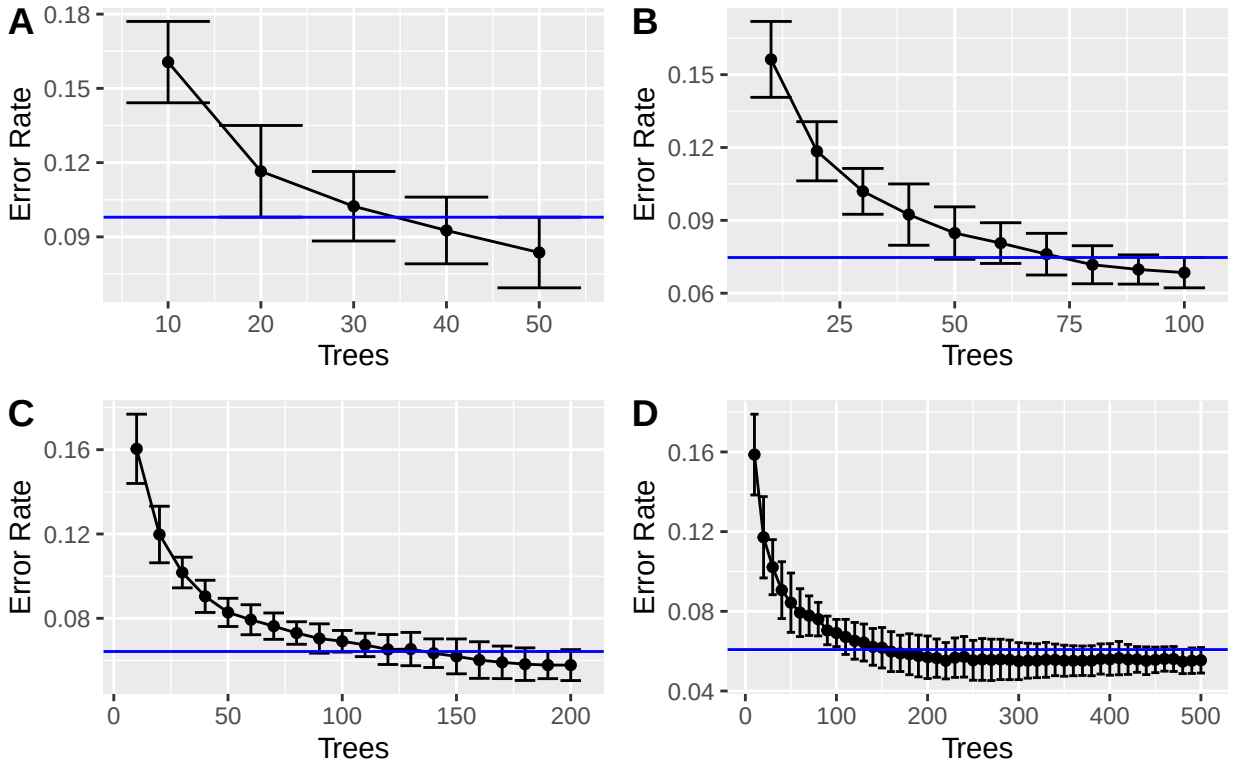


Figure 2: Cross−validation with m = 5 and M = 50,100,200,500

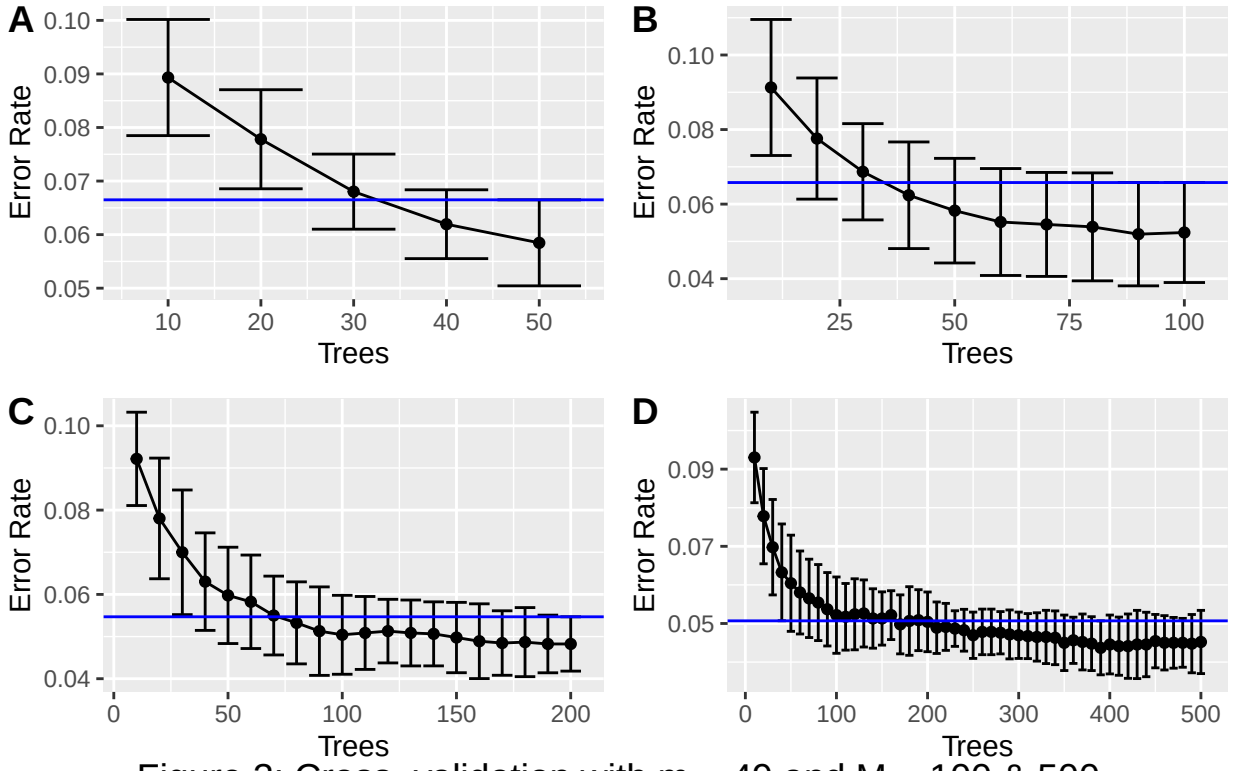



Figure 3: Cross−validation with m = 49 and M = 100 & 500

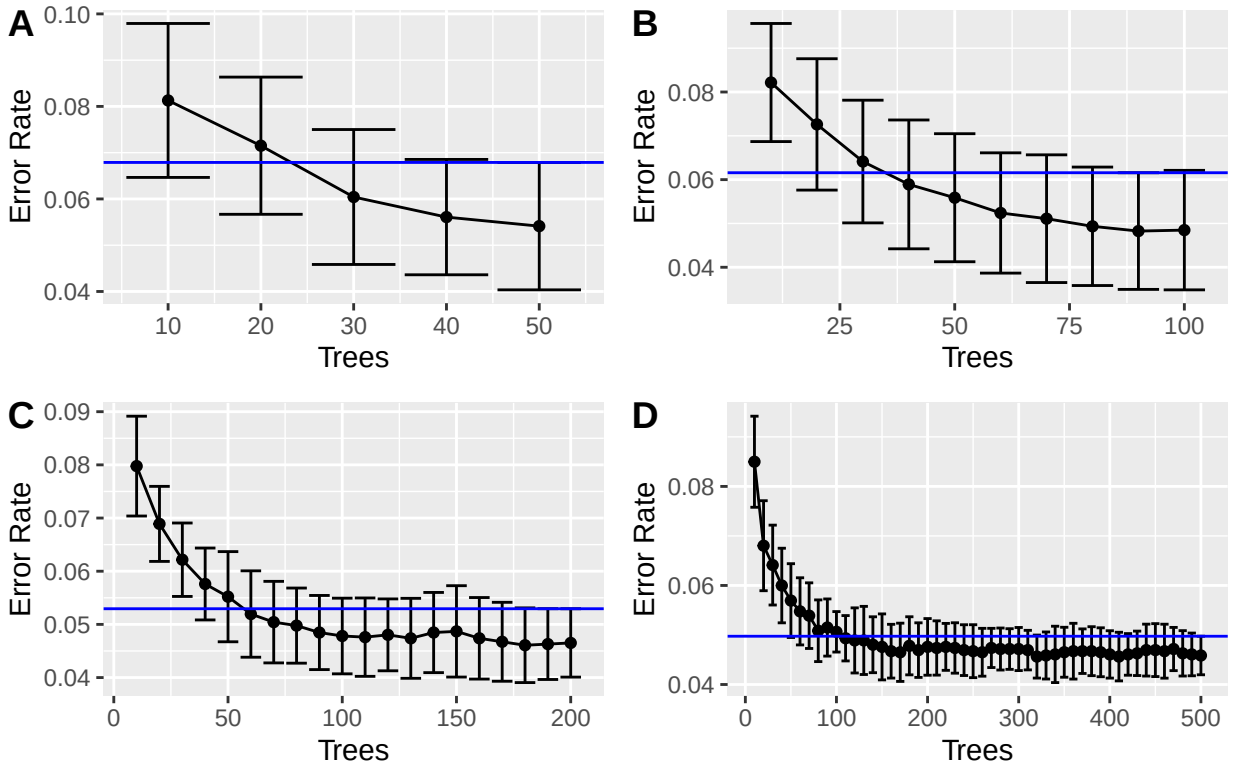


Figure 4: Cross−validation with m = 49 and M = 100 & 500

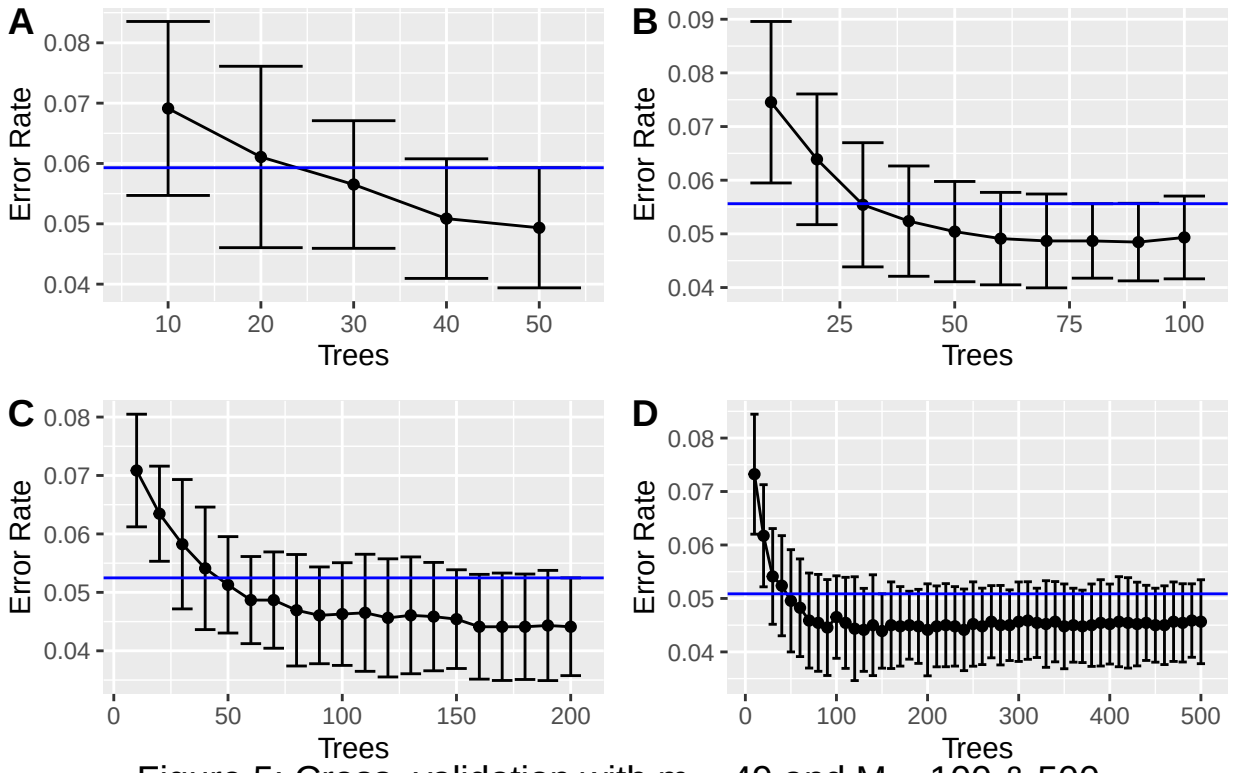
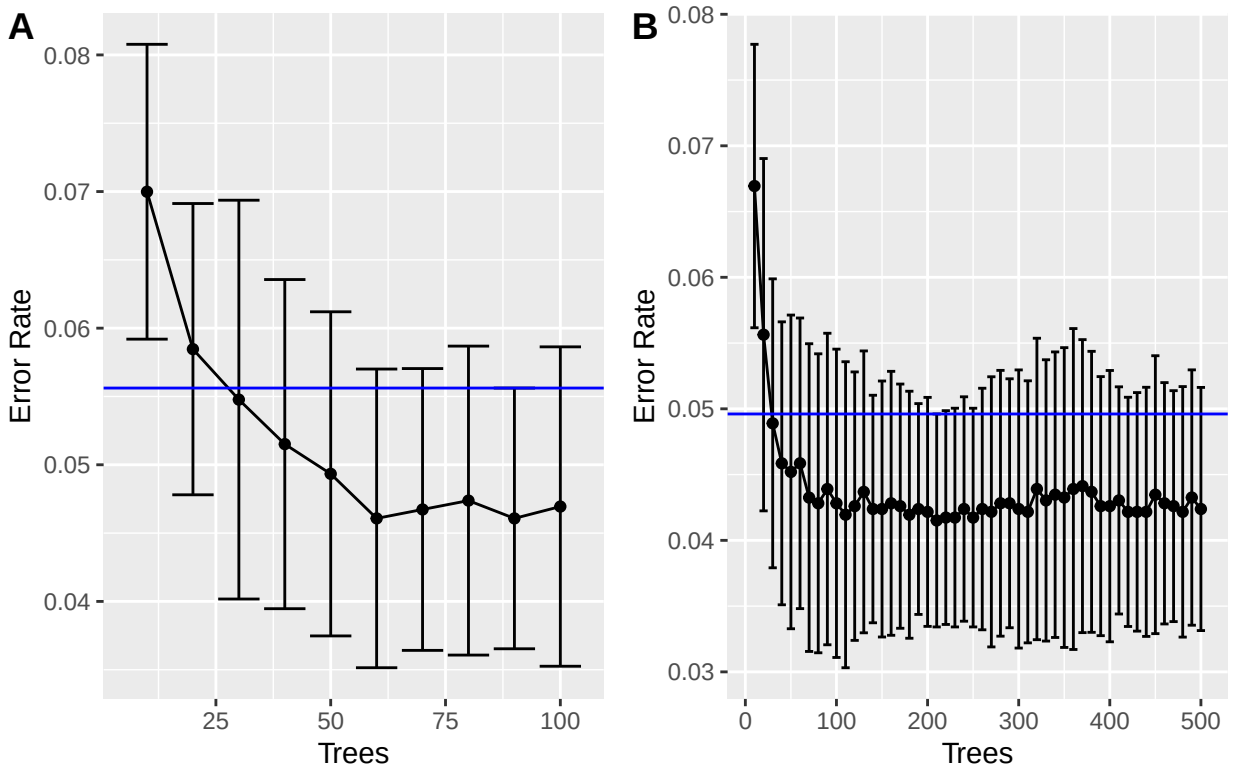



Figure 5: Cross−validation with m = 49 and M = 100 & 500



**D)**

In the plots above we can easy see that the numbers of $m$ has a impact on the cross-validation error, it seems that the lower $m$, nodes, the higher the cross-validation error, but the common theme is that the more trees that should be grown, that is larger $M$ lower the error rate. The blue line in represent the one-standard-error rule which is used to identify the optimal model. This result can then be used to see that when increasing $m$ will result to that the bias get lower however the the variance gets higher. This can also be reasoning that the more paramters you split, that is higher $m$ the higher the chance it will be incorrectly splitted for another dataset. This reasoning ground in that if you split different at the begining the whole tree will be wrong.

**Task 2**

**A)**

The K-class multinomial deviance loss function is defined as:

$$L(y, p(x)) = -\sum_{k=1}^{K} I(y = G_k) \log p_k(x) = -\sum_{k=1}^{K} I(y = G_k) f_k(x) + \log(\sum_{\ell=1}^{K} e^{f_\ell(x)})$$

If we start with derivate the first term $-\sum_{k=1}^{K} I(y = G_k) f_k(x)$:

$$\frac{\partial - \sum_{k=1}^{K} I(y = G_k) f_k(x)}{\partial f_k(x)} = -I(y_i = G_k), \quad \text{note that } f(x_i) = f_{m-1}(x_i)$$

If we now remember how to derivate of $\log(x)$ is equal to $\frac{x'}{x}$. This gives os now that the derivite of the second term is equal to the following:

$$\frac{e^{f_k(x_i)}}{\sum_{\ell=1}^{K} e^{f_\ell(x_i)}} = p_k(x_i), \quad \text{note that } f(x_i) = f_{m-1}(x_i)$$

If we nog combinde the result above and take the negative derivate instead, that is just multiplying with $-1$ we get the following:

$$-g_{mi} = -\frac{\partial L(y_i, f_1(x_i), f_1(x_i) \ldots, f_K(x_i))}{\partial f_k(x_i)} = I(y_i = G_k) - p_k(x_i), \quad \text{note that } f(x_i) = f_{m-1}(x_i)$$

Which is the same result which is displayed in the table 10.2 in the textbook.

**B)**

We are now asked to derive the variance formula $\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$. Remeber that the variance terms of the moments is given by $E[X^2] = var(X) + E[X]^2 \Leftrightarrow var(X) = E[X^2] - E[X]^2$. Note that $X_1, X_2 \ldots, X_B$ is a B identically distibuted random variables with $X_i \sim N(\mu, \sigma^2)$, however they don't have to be independent. We not let $\rho$ denote the positive pairwise correlation, that is:

$$\rho := \frac{1}{\sigma^2} E[(X_i - \mu)(X_j - \mu)] > 0$$

Which implicates that for $i \neq j$ we have $E[X_i X_j] = \rho\sigma^2 + \mu^2$. From this information we can now derive the variance of the average:

$$var(\frac{1}{B}\sum_{i=1}^{B} X_i) = \frac{1}{B^2}(E[(\sum_{i=1}^{B} X_i)^2] - E[\sum_{i=1}^{B} X_i]^2) = \frac{1}{B^2}(B\sigma^2 + B^2\rho\sigma^2 + B^2\mu^2 - B\rho\sigma^2 - B^2\mu^2) = \frac{1}{B^2}(B\sigma^2 + B^2\rho\sigma^2 + B\rho\sigma^2)$$

If we now simply this expression we obtain the following:

$$var(\frac{1}{B}\sum_{i=1}^{B} X_i) = \frac{B\sigma^2}{B^2} + \frac{B^2\rho\sigma^2}{B^2} + \frac{B\rho\sigma^2}{B^2} = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2 \qquad \blacksquare$$

We have now derived the variance formula

### C)

The hyperparameter $m$ in the algortihm for random forest needs to be fixed. This parameter will affect the bias and the bariance of the resulting random forest model. If we start with the extreme case $m = p$ that is we choose between all the variables, this scenario is the same as bagging and we know bagging will genereate a high variance. In general the bias will decrease when $m$ gets larger, and increases when $m$ gets smaller. This is because you will allways split the best variable suitable for the trainingdata, which will then be best tree for the trainingdata and therefore a low bias. However if you split different at the begining the whole tree will be wrong, that is a high chance if you allways choosing the best variable to split for the traningdata, this leads to a higher variance. The variance will increase when the $m$ gets larger and when $m$ gets smaller the variance will decrease.

### Appendix/Code

```
# Reading in the data and the library that will be used
library(tidyverse) # Style of coding
library(knitr) # Makes tables
library(gbm) # Gradient boosting trees
library(caret) # For cross-validation
library(cowplot) # Make pairvise plots

# Reading in the data using tidyverse::read_table2
spam_data <- read_table2("../Statistisk inlärning/spam.data.txt", col_names = F)
```

```
# Creating a table which displays the different parameters in the function gmb()
parameters <- rbind("distrubution","n.trees","interaction.depth",
                    "n.minobsinnode","shrinkage","cv.folds")


explanation <- rbind("Specifying the name of the distribution ",
                     "Specifying the total numbers of trees to fit",
                     "Specifying the maximum depth of each tree",
                     "Specifying the minimum number of observations in the terminal nodes of the trees"
                     "The learning rate", "Number of cross-validation folds to perform")


as_tibble(cbind(parameters,explanation)) %>%  # Makes a tibble and print a table
  kable(
    col.names = c("Parameter","Information"),
    caption = "List over parameters in the function gbm()"
  )
```

```
# So we can redo the simulation/training/fitting
set.seed(1997)
```

```r
# Function that construct gradient boosting trees
# The list_nodes indicates a list of different number of nodes, amount_of trees = #trees
gb_trees <- function(list_nodes, amount_of_trees){
  models <- c() # empty list
  for(i in list_nodes){ # for each choosen number of nodes
    models <- append(models,
                     list(gbm(factor(X58)~., # preforms gbm
                              data = spam_data,
                              distribution = "bernoulli", #Gives us the distr. & loss func.
                              n.trees=amount_of_trees, #Amount of trees that will be grown
                              interaction.depth=i) # Says how big every split will grow
                          )
                     )
  }
  return(models)
}




differnt_nodes_trees <- gb_trees(c(1,5,10,20,49),1000)

# When trying to use 50 nodes the i get the following error message,
# and therefore only uses 49 nodes as the max:
# interaction.depth must be less than 50.
# You should also ask yourself why you want such large interaction terms.
# A value between 1 and 5 should be sufficient for most applications
```

```r
set.seed(1997)
# Since the gbm doest provide all the infromation when applying the
# cv.fold we need to use another function.

cv_gbm <- function(cv_folds,nodes,max_trees){

# Decide how we should train the model, that is with cv.
train.control <- trainControl(method = "cv", number = cv_folds)

parameter_grid <- expand.grid(
  interaction.depth = nodes,
  n.trees = c(seq(from = 10 , to =max_trees, by = 10)), # Gives us M = 10,20,30,... etc
  shrinkage = 0.1, # Using what is defualt in the function gbm()
  n.minobsinnode = 10 # Using what is defualt in the function gbm()
                                        )
# Train the model
cv_model <- train(factor(X58) ~.,
                  data = spam_data,
                  method = "gbm",
                  trControl = train.control, # specifyies that cv should be preformed
                  distribution = "bernoulli", # Biniary classification
                  verbose = FALSE, # Doenst print out every model
                  metric = "Accuracy",# Adding the accuracy rate for each cv
                  tuneGrid = parameter_grid # specify the parameters
                  )
```

```
return(cv_model)
}


# CV for M = 50:500 for m = 1
cv_1_nodes_50_trees <- cv_gbm(10,1,50)
cv_1_nodes_100_trees <- cv_gbm(10,1,100)
cv_1_nodes_200_trees <- cv_gbm(10,1,200)
cv_1_nodes_500_trees <- cv_gbm(10,1,500)

# CV for M = 50:500 for m = 5
cv_5_nodes_50_trees <- cv_gbm(10,5,50)
cv_5_nodes_100_trees <- cv_gbm(10,5,100)
cv_5_nodes_200_trees <- cv_gbm(10,5,200)
cv_5_nodes_500_trees <- cv_gbm(10,5,500)

# CV for M = 50:500 for m = 10
cv_10_nodes_50_trees <- cv_gbm(10,10,50)
cv_10_nodes_100_trees <- cv_gbm(10,10,100)
cv_10_nodes_200_trees <- cv_gbm(10,10,200)
cv_10_nodes_500_trees <- cv_gbm(10,10,500)



# CV for M = 50:500 for m = 20
cv_20_nodes_50_trees <- cv_gbm(10,20,50)
cv_20_nodes_100_trees <- cv_gbm(10,20,100)
cv_20_nodes_200_trees <- cv_gbm(10,20,200)
cv_20_nodes_500_trees <- cv_gbm(10,20,500)


# CV for M=100 & 500 for m = 49
cv_49_nodes_100_trees <- cv_gbm(10,49,100)
cv_49_nodes_500_trees <- cv_gbm(10,49,500)

# Since we are intressted in the error we need to take 1 - accuracy
# m = 1

cv_1_nodes_50_trees$results$Accuracy <- 1-cv_1_nodes_50_trees$results$Accuracy
cv_1_nodes_100_trees$results$Accuracy <- 1-cv_1_nodes_100_trees$results$Accuracy
cv_1_nodes_200_trees$results$Accuracy <- 1-cv_1_nodes_200_trees$results$Accuracy
cv_1_nodes_500_trees$results$Accuracy <- 1-cv_1_nodes_500_trees$results$Accuracy


# m = 5

cv_5_nodes_50_trees$results$Accuracy <- 1-cv_5_nodes_50_trees$results$Accuracy
cv_5_nodes_100_trees$results$Accuracy <- 1-cv_5_nodes_100_trees$results$Accuracy
cv_5_nodes_200_trees$results$Accuracy <- 1-cv_5_nodes_200_trees$results$Accuracy
cv_5_nodes_500_trees$results$Accuracy <- 1-cv_5_nodes_500_trees$results$Accuracy

# m = 10
cv_10_nodes_50_trees$results$Accuracy <- 1-cv_10_nodes_50_trees$results$Accuracy
```

```r
cv_10_nodes_100_trees$results$Accuracy <- 1-cv_10_nodes_100_trees$results$Accuracy
cv_10_nodes_200_trees$results$Accuracy <- 1-cv_10_nodes_200_trees$results$Accuracy
cv_10_nodes_500_trees$results$Accuracy <- 1-cv_10_nodes_500_trees$results$Accuracy

# m = 20

cv_20_nodes_50_trees$results$Accuracy <- 1-cv_20_nodes_50_trees$results$Accuracy
cv_20_nodes_100_trees$results$Accuracy <- 1-cv_20_nodes_100_trees$results$Accuracy
cv_20_nodes_200_trees$results$Accuracy <- 1-cv_20_nodes_200_trees$results$Accuracy
cv_20_nodes_500_trees$results$Accuracy <- 1-cv_20_nodes_500_trees$results$Accuracy


# m = 49

cv_49_nodes_100_trees$results$Accuracy <- 1-cv_49_nodes_100_trees$results$Accuracy
cv_49_nodes_500_trees$results$Accuracy <- 1-cv_49_nodes_500_trees$results$Accuracy



# Function to make a plot for the cv-error
plot_cv_error <- function(cv_file){
  cv_file %>%
    ggplot() +
    geom_errorbar(
    aes(ymin =cv_file$results$Accuracy-cv_file$results$AccuracySD, # calulate the min
        ymax = cv_file$results$Accuracy+cv_file$results$AccuracySD)) + # calulate the max
   scale_y_continuous("Error Rate") +
   scale_x_continuous("Trees") +
   geom_hline(yintercept = min(cv_file$results$Accuracy+cv_file$results$AccuracySD),
              col ="blue")  # one-standard-error line

}


# Plotting the CV error rates with the standard deviation bars.

# m = 1

plot_1_nodes_50_trees <- plot_cv_error(cv_1_nodes_50_trees)
plot_1_nodes_100_trees <- plot_cv_error(cv_1_nodes_100_trees)
plot_1_nodes_200_trees <- plot_cv_error(cv_1_nodes_200_trees)
plot_1_nodes_500_trees <- plot_cv_error(cv_1_nodes_500_trees)



# m = 5

plot_5_nodes_50_trees <- plot_cv_error(cv_5_nodes_50_trees)
plot_5_nodes_100_trees <- plot_cv_error(cv_5_nodes_100_trees)
plot_5_nodes_200_trees <- plot_cv_error(cv_5_nodes_200_trees)
plot_5_nodes_500_trees <- plot_cv_error(cv_5_nodes_500_trees)
```

```r
# m = 10

plot_10_nodes_50_trees <- plot_cv_error(cv_10_nodes_50_trees)
plot_10_nodes_100_trees <- plot_cv_error(cv_10_nodes_100_trees)
plot_10_nodes_200_trees <- plot_cv_error(cv_10_nodes_200_trees)
plot_10_nodes_500_trees <- plot_cv_error(cv_10_nodes_500_trees)


# m = 20

plot_20_nodes_50_trees <- plot_cv_error(cv_20_nodes_50_trees)
plot_20_nodes_100_trees <- plot_cv_error(cv_20_nodes_100_trees)
plot_20_nodes_200_trees <- plot_cv_error(cv_20_nodes_200_trees)
plot_20_nodes_500_trees <- plot_cv_error(cv_20_nodes_500_trees)



# m = 49

plot_49_nodes_100_trees <- plot_cv_error(cv_49_nodes_100_trees)
plot_49_nodes_500_trees <- plot_cv_error(cv_49_nodes_500_trees)



# Ploting  m = 1 for differnt M side by side
 title1 <- ggdraw() + # makes a title for the plot
  draw_label(
    "Figure 1: Cross-validation with m = 1 and M = 50,100,200,500"
  )
 plot_grid(title1,plot_grid(plot_1_nodes_50_trees,plot_1_nodes_100_trees,
                            plot_1_nodes_200_trees,plot_1_nodes_500_trees, labels = "AUTO"),
           ncol=1, rel_heights=c(0.1, 1))


# Ploting  m = 5 for differnt M side by side
 title2 <- ggdraw() + # makes a title for the plot
  draw_label(
    "Figure 2: Cross-validation with m = 5 and M = 50,100,200,500"
  )
plot_grid(title2,plot_grid(plot_5_nodes_50_trees,plot_5_nodes_100_trees,
                           plot_5_nodes_200_trees,plot_5_nodes_500_trees, labels = "AUTO"),
          ncol=1, rel_heights=c(0.1, 1))


# Ploting  m = 10 for differnt M side by side
title3 <- ggdraw() + # makes a title for the plot
  draw_label(
    "Figure 3: Cross-validation with m = 10 and M = 50,100,200,500"
  )
plot_grid(title3,plot_grid(plot_10_nodes_50_trees,plot_10_nodes_100_trees,
                           plot_10_nodes_200_trees,plot_10_nodes_500_trees,labels = "AUTO"),
          ncol=1, rel_heights=c(0.1, 1))
```

```r
# Ploting  m = 20 for differnt M side by side
title4 <- ggdraw() + # makes a title for the plot
  draw_label(
    "Figure 4: Cross-validation with m = 20 and M = 50,100,200,500"
  )
plot_grid(title4,plot_grid(plot_20_nodes_50_trees,plot_20_nodes_100_trees,
                           plot_20_nodes_200_trees,plot_20_nodes_500_trees, labels = "AUTO"),
          ncol=1, rel_heights=c(0.1, 1))

# Ploting  m = 49 for differnt M side by side
title5 <- ggdraw() +  # makes a title for the plot
  draw_label(
    "Figure 5: Cross-validation with m = 49 and M = 100 & 500"
  )

plot_grid(title5, plot_grid(plot_49_nodes_100_trees,plot_49_nodes_500_trees,
                            labels = "AUTO"),
          ncol=1,
          rel_heights=c(0.1, 1))
```