Machine Learning Homework 4

Latera Tesfaye Olana

04 March, 2023

Question 1

Question 1a: The following table provides summary of the given data.

Table 1: Summary table of the data

Disease	observations
No Disease	171
Heart Disease	200

The given data has 171 No disease outcome observations and 200 Heart disease observations. The data has two predictors (p=12). Overall, the data has 371 observations. This data has no missing values.

The data is divided into two, training and test with each containing 53.91 and 46.09, for the data respectively.

Question 1b:

Figure 1 shows the overgrown tree.

Question 1c:

The train mis-classification error is 12 %, where as the test mis-classification error is 22.81 %. The model seems to over fit (This might mean that the model is capturing noise and specific patterns in the training data that may not generalize well to new, unseen data).

Question 1d:

Pruning the tree using CV. Figure 2 shows the mis-classification error against sub-tree size.

Index of tree with minimum error is 3 and number of terminal in this tree is 8. Looking at figure 2, tree of size 8 seems to have less mis-classification error. Figure 3 shows a classification tree plot for the selected tree size.

Figure 4, shows a poor attempt of mine to Photoshop both classification tree (pruned and unpruned). The image at the background (blue) is the full tree where as the one highlighted well (red) is the pruned tree.

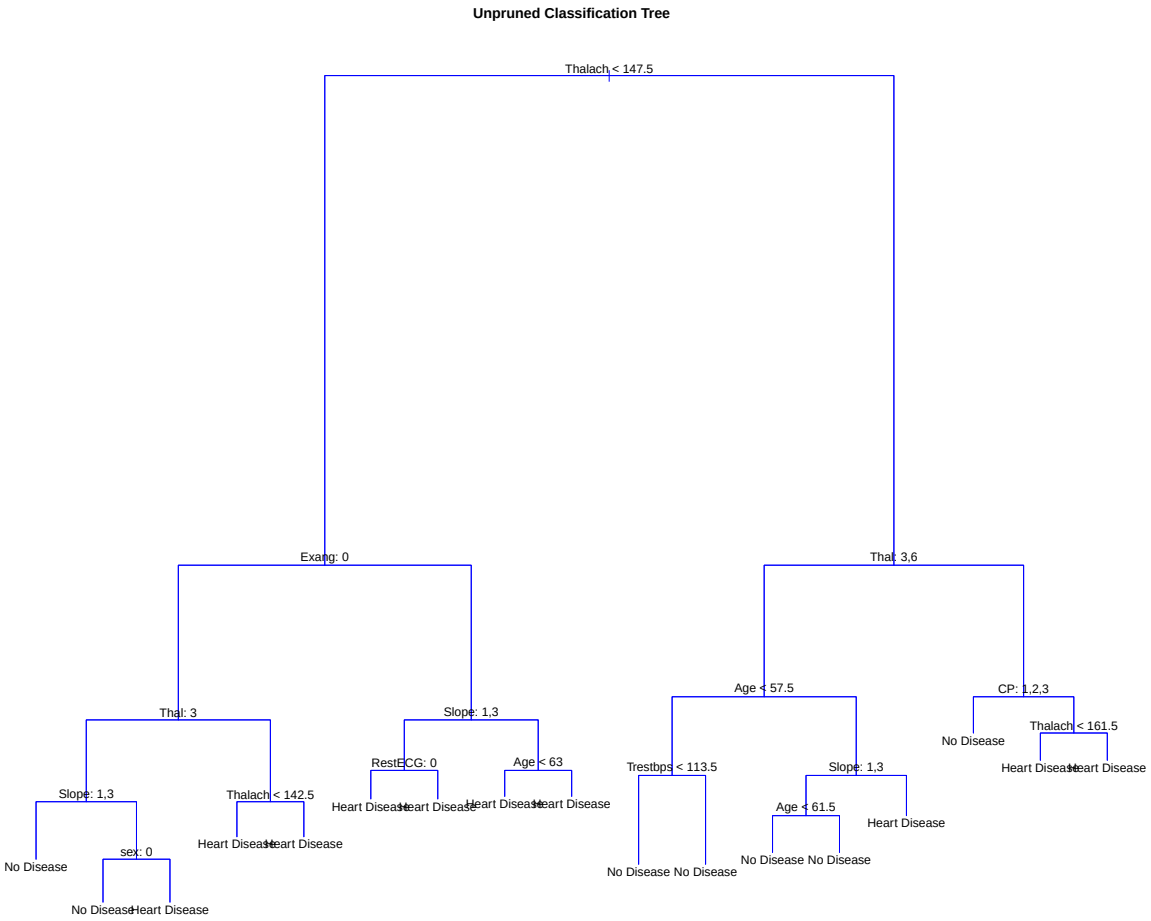

Figure 1: Full grown tree

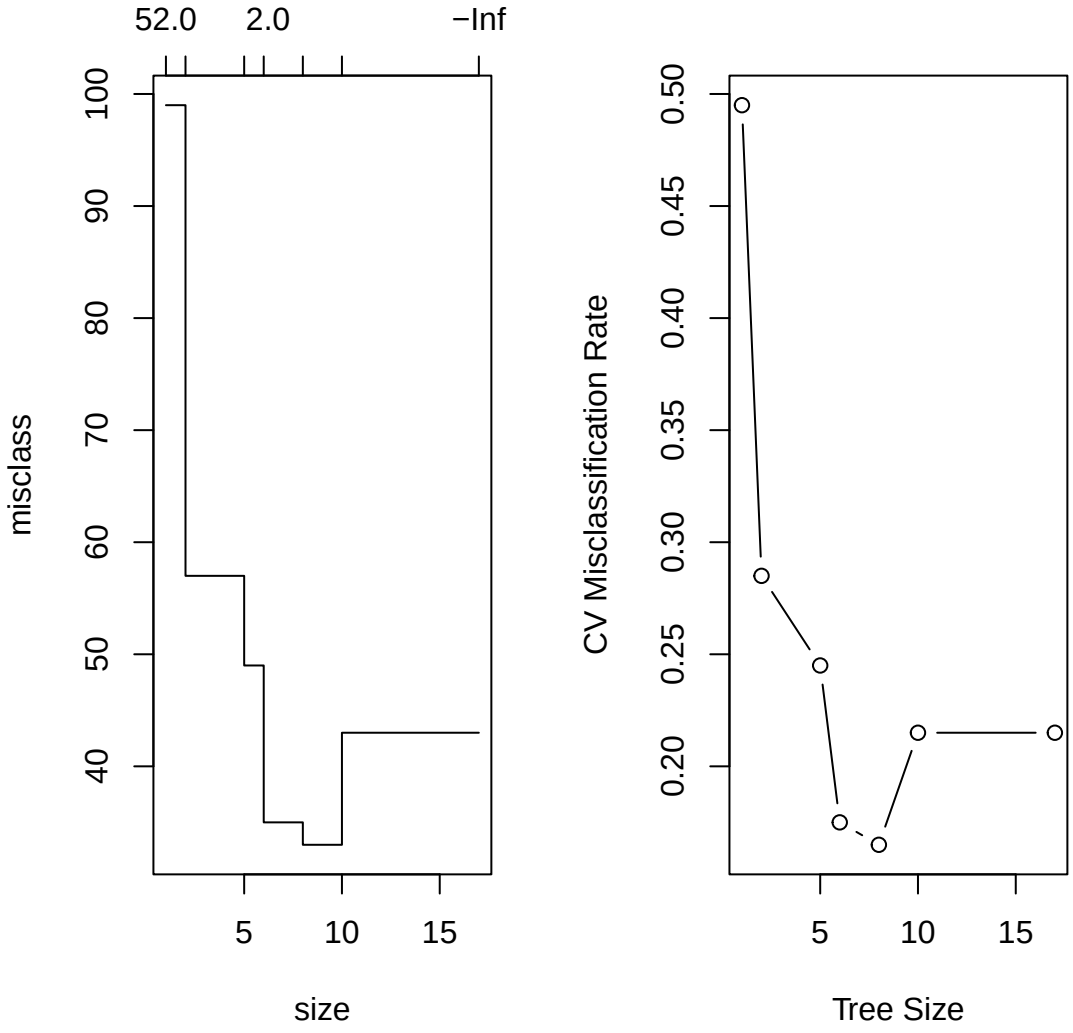

Figure 2: CV misclassification tree and tree sizes

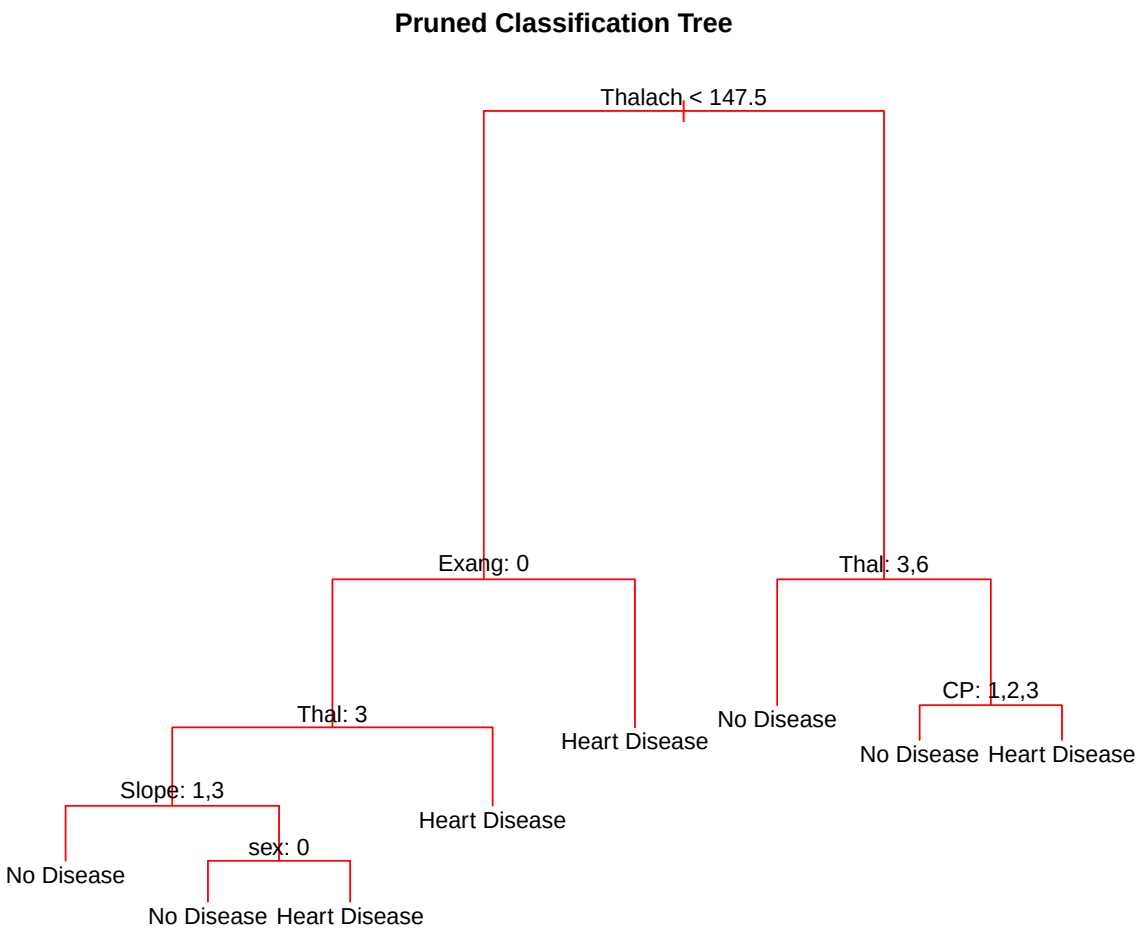

Figure 3: Pruned tree

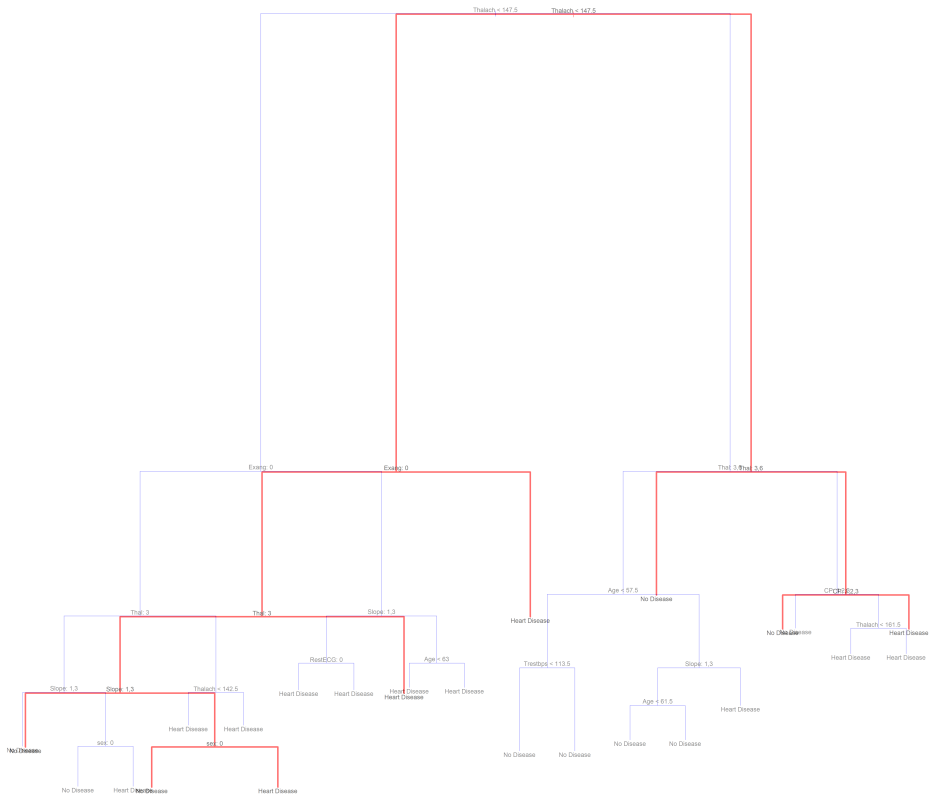

Figure 4: Overlay of full grown tree and pruned tree - red is pruned

Question 1e:

The train mis-classification error is 12.5 %, where as the test mis-classification error is 21.64 %. The model seems to over fit (This might mean that the model is capturing noise and specific patterns in the training data that may not generalize well to new, unseen data). Compared to unpruned tree the pruned did slightly better in terms of increasing the test accuracy as well as the challenges in over fitting. The error report of both methods show that pruning the tree leads to a reduction in the difference between the training and test accuracy, and thus improves the generalization performance of the model. The pruning process has helped to reduce the variance of the unpruned tree by simplifying its structure and removing some of the noise and specific patterns in the training data that do not generalize well to new data.

The lower training accuracy of the pruned tree suggests that it has a higher bias than the overgrown tree, since it is less able to fit the training data. However, this bias is offset by a lower variance, which allows the pruned tree to generalize better to new, unseen data.

Question 1f:

I used $m_{try} = 12$ for bagging.

For this bagging model the train mis-classification error is 0 %, where as the test mis-classification error is 22.81 %. Accordingly, this model has low test accuracy, but high training accuracy. This is a sign of overfitting (high variance).

Question 1g:

I used $m_{try} = 4$ for random forest.

For this random forest model the train mis-classification error is 0 %, where as the test mis-classification error is 21.05 %. This model has also low test accuracy, but high training accuracy. This is a sign of overfitting (high variance), but slightly better than the bagging model.

Question 1h:

Some reasons for why different values might be generated seed is not set:

1. Bagging trees introduces a random component into the tree building process by building many trees on bootstrapped copies of the training data. These bootstrapped copies are sampled with replacement to create multiple bootstrapped datasets. The samples that are selected can differ each time the model is run, resulting in different models being built. (for both models)
2. The Random Forest algorithm also selects a random sample of features to use for each tree. This might result in change.

Question 1i:

For this boosting model the train mis-classification error is 0 %, where as the test mis-classification error is 19.3 %. Well improved performance as compared to the previous models we have seen. Boosting is known to have a high risk of overfitting, especially when the number of trees is large. In our case, we used 500 trees, which might be enough to fit the noise in the data and result in overfitting. Therefore, it is possible that our model is simply memorizing the training data, resulting in perfect accuracy on the training set but failing to generalize to new data.

Question 2a:

X predictors generated.

Question 2b:

Response Y generated.

Question 2c:

The genereted data fitted.

Question 2d:

lambda = 1e(-3) seems to fit the data well with less overfitting. lambda = 1e(-7) appears to overfit the data.

Question 2e:

After CV, the selected value of lambda = 2.8789099×10^{-5}. New model was fitted with this value.

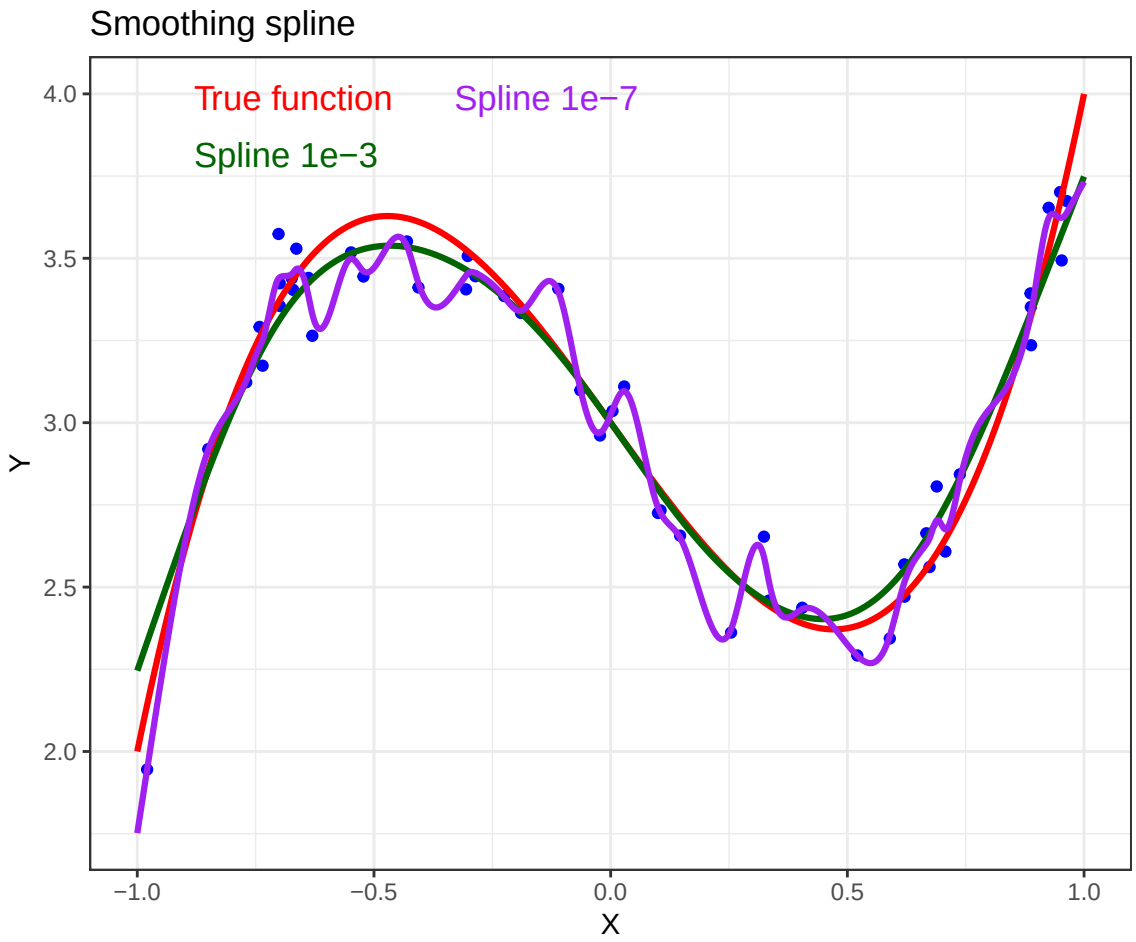

Figure 5: Spline fit for different lambda values

Question 2f:

Plot for new selected value of lambda is shown below.

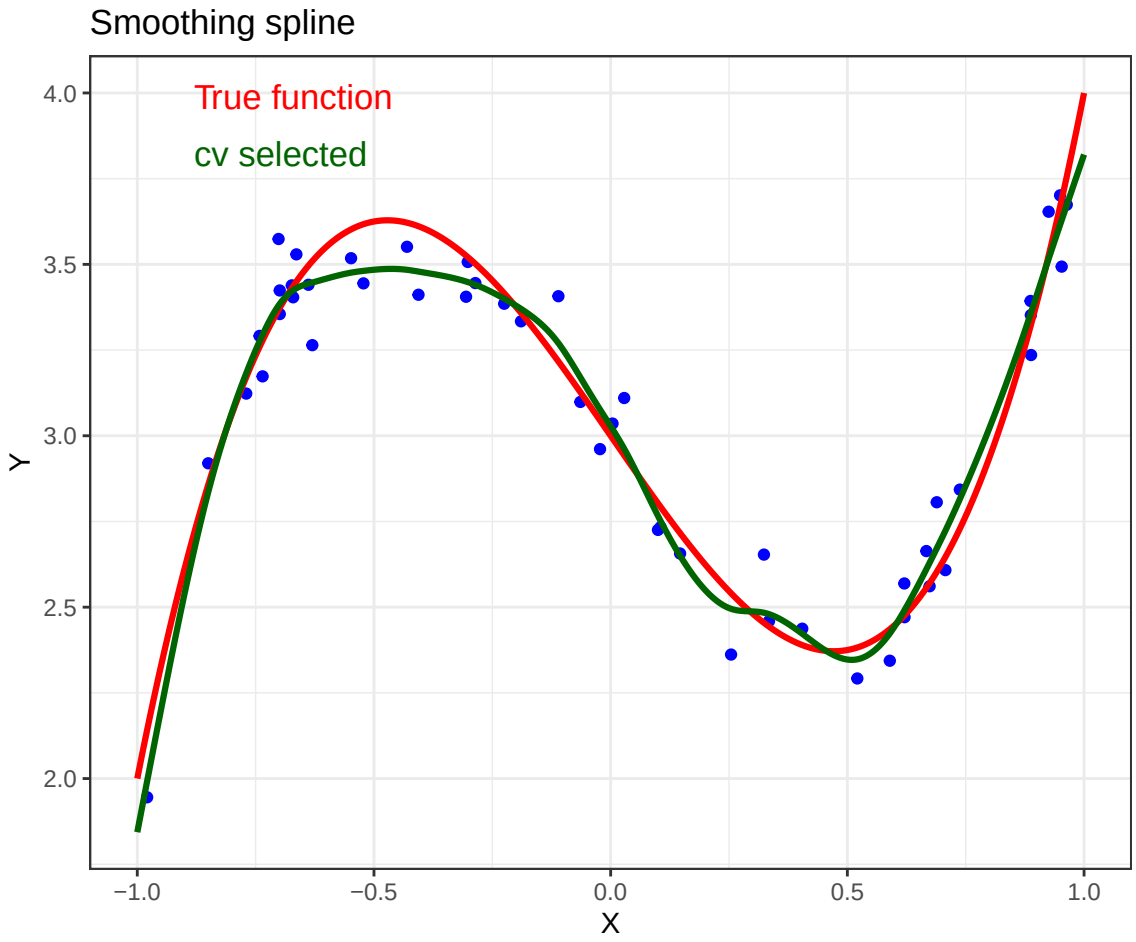

Figure 6: Lamda selected and fitted with CV

Code Appendix

```
knitr::opts_chunk$set(fig.pos = 'H')
### Setting up the packages
library(knitr)
knitr::opts_chunk$set(echo = FALSE)
# check if packages are installed; if not, install them
packages <- c("tidyverse", "readr", "ggExtra", "plotly",
              "ggplot2","ggstatsplot","ggside","rigr","nlme","lmtest",
              "sandwich","gridExtra","broom","janitor","ellipse","caret",
              "pROC","MASS","class","purrr","tidyr","ggcorrplot","glmnet","gbm",
              "tree","randomForest")
not_installed <- setdiff(packages, rownames(installed.packages()))
if (length(not_installed)) install.packages(not_installed)

# load packages
library(sandwich)
library(grid)
library(randomForest)
library(tree)
library(ggcorrplot)
library(gbm)
library(glmnet)
library(janitor)
library(readr)
library(lmtest)
library(class)
library(pROC)
library(nlme)
library(ellipse)
library(broom)
library(ggstatsplot)
library(ggside)
library(caret)
library(rigr)
library(ggExtra)
library(gridExtra)
library(purrr)
library(plotly)
library(ggplot2)
library(MASS)
library(tidyverse)
library(tidyr)

### ------------------------------------------------------------
#Loading working directory of the raw data

#Please load your data/directory by changing it with your work directory
#Throughout this code module you will see a tone of places, where
#data is read and written, so please make sure to change them to your
#working directory folder format

working_directory_data <- setwd("C:/Users/latera/Desktop/ML_ass")
```

```r
study_data <- load("data/heart.RData")

study_data <- full


#Describe the data
#as.tibble(study_data)

#Check if the outcome is factor
is.factor(study_data$Disease)

summary <- study_data %>%
  group_by(Disease) %>%
  dplyr::summarise(observations = n())

knitr::kable(summary, caption = "Summary table of the data")
set.seed(2)
study_data_idx = sample(nrow(study_data), 200)
study_data_trn = study_data[study_data_idx, ]
study_data_tst = study_data[-study_data_idx, ]

overgrown_tree <- tree(Disease ~ ., data = study_data_trn)
plot(overgrown_tree, col="blue")
text(overgrown_tree, pretty = 0)

title(main = "Unpruned Classification Tree")

#Calculating accuracy
calc_acc = function(actual, predicted) {
  mean(actual == predicted)
}
train_pred <- predict(overgrown_tree, newdata = study_data_trn, type = "class")
test_pred <- predict(overgrown_tree, newdata = study_data_tst, type = "class")
set.seed(2)
seat_tree_cv = cv.tree(overgrown_tree, FUN = prune.misclass)
par(mfrow = c(1, 2))
# default plot
plot(seat_tree_cv)
# better plot
plot(seat_tree_cv$size, seat_tree_cv$dev / nrow(study_data_trn), type = "b",
     xlab = "Tree Size", ylab = "CV Misclassification Rate")
min_idx = which.min(seat_tree_cv$dev)
ter <- seat_tree_cv$size[min_idx]
seat_tree_prune = prune.misclass(overgrown_tree, best = 8)

plot(seat_tree_prune,col="red")
text(seat_tree_prune, pretty = 0)

title(main = "Pruned Classification Tree")
knitr::include_graphics("imagek.png")
train_pred_prun <- predict(seat_tree_prune, newdata = study_data_trn, type = "class")
test_pred_prun <- predict(seat_tree_prune, newdata = study_data_tst, type = "class")
set.seed(2)
```

```r
bagged_trees <- randomForest(Disease ~ ., data = study_data_trn, mtry = 12, importance=TRUE)
train_pred_bag <- predict(bagged_trees, study_data_trn)
test_pred_bag <- predict(bagged_trees, study_data_tst)

set.seed(2)
bagged_trees_rf <- randomForest(Disease ~ ., data = study_data_trn, mtry = 4, importance=TRUE)
train_pred_rf <- predict(bagged_trees_rf, study_data_trn)
test_pred_rf <- predict(bagged_trees_rf, study_data_tst)

stu_train_int <- study_data_trn %>% mutate(Disease = as.integer(Disease) - 1L)
stu_test_int <- study_data_tst %>% mutate(Disease = as.integer(Disease) - 1L)


set.seed(2)
boost_model <- gbm(Disease ~ ., data = stu_train_int, distribution = "bernoulli",
                   n.trees = 500, interaction.depth = 2, shrinkage = 0.1)

# Compute the predicted labels for the training and test data
train_preds_boost <- ifelse(predict(boost_model, study_data_trn, type = "response") > 0.5, 1, 0)
test_preds_boost <- ifelse(predict(boost_model, study_data_tst, type =
                                     "response") > 0.5, 1, 0)

set.seed(2)

# Generate predictor
X <- runif(50, -1, 1)

# Generate noise
noise <- rnorm(50, mean = 0, sd = 0.1)
Y <- 3 - 2*X + 3*X^3 + noise
# Fit smoothing spline models
first_fit <- smooth.spline(X, Y, lambda = 1e-3)
second_fit <- smooth.spline(X, Y, lambda = 1e-7)



x_grid <- seq(from = -1, to = 1, length.out = 1000)

# estimate the f_true
f_true <- 3 - 2 * x_grid + 3 * x_grid^3

# fitted splines on the grid
y_pred1 <- predict(first_fit, x_grid)$y
y_pred2 <- predict(second_fit, x_grid)$y



df <- data.frame(X = X, Y = Y)

# Create plot
k<-ggplot(data = df, aes(x = X, y = Y)) +
  geom_point(color = "blue") +
  xlab("X") + ylab("Y") + ggtitle("Smoothing spline") +
```

```
  geom_line(data = data.frame(x_grid = x_grid, f_true = f_true), aes(x = x_grid, y = f_true), color = "
  geom_line(data = data.frame(x_grid = x_grid, y_pred1 = y_pred1), aes(x = x_grid, y = y_pred1), color =
  geom_line(data = data.frame(x_grid = x_grid, y_pred2 = y_pred2), aes(x = x_grid, y = y_pred2), color =
  scale_color_manual(values = c("red", "darkgreen", "purple", "blue"), guide = "legend",
                     labels = c("True function", "Spline 1e-3", "Spline 1e-7", "Observed data")) +
  theme_bw()

  grob <- grobTree(textGrob("True function", x=0.1,  y=0.95, hjust=0,
  gp=gpar(col="red", fontsize=13)))

  grob2 <- grobTree(textGrob("Spline 1e-3", x=0.1,  y=0.88, hjust=0,
  gp=gpar(col="darkgreen", fontsize=13)))

  grob3 <- grobTree(textGrob("Spline 1e-7", x=0.35,  y=0.95, hjust=0,
  gp=gpar(col="purple", fontsize=13)))

# Plot
k + annotation_custom(grob) + annotation_custom(grob2) + annotation_custom(grob3)
cv_func <- smooth.spline(x=X, y=Y, cv=TRUE)
third_fit <- smooth.spline(X, Y, lambda = cv_func$lambda)
x_grid <- seq(from = -1, to = 1, length.out = 1000)

# estimate the f_true
f_true <- 3 - 2 * x_grid + 3 * x_grid^3

# fitted splines on the grid
y_pred_new <- predict(third_fit, x_grid)$y


k<-ggplot(data = df, aes(x = X, y = Y)) +
  geom_point(color = "blue") +
  xlab("X") + ylab("Y") + ggtitle("Smoothing spline") +
  geom_line(data = data.frame(x_grid = x_grid, f_true = f_true), aes(x = x_grid, y = f_true), color = "
  geom_line(data = data.frame(x_grid = x_grid, y_pred_new = y_pred_new), aes(x = x_grid, y = y_pred_new
  scale_color_manual(values = c("red", "darkgreen", "blue"), guide = "legend",
                     labels = c("True function", "cv selected", "Observed data")) +
  theme_bw()

  grob <- grobTree(textGrob("True function", x=0.1,  y=0.95, hjust=0,
  gp=gpar(col="red", fontsize=13)))

  grob2 <- grobTree(textGrob("cv selected", x=0.1,  y=0.88, hjust=0,
  gp=gpar(col="darkgreen", fontsize=13)))


# Plot
k + annotation_custom(grob) + annotation_custom(grob2)
```