Liver disease prediction using ensemble learning methods

STATS/CSE 790 Assignment 3

2024-02-14

Pao Zhu Vivian Hsu (400547994)

Introduction

In this paper, we perform boosting, bagging, and random forest to predict whether a patient has liver disease or not. The data in this study comes from the UCI Machine Learning Repository (Ramana & Venkateswarlu, 2012). It contains 583 rows of patient data and 11 variables measuring different aspects of the patient. The response variable is categorical variable that indicates whether the patient is diagnosed with liver disease or not. The remaining variables include age, gender, total Bilirubin, direct Bilirubin, total proteins, albumin, A/G ratio, SGPT, SGOT and Alkphos.

Methods

We first began the study by applying transformations to the data. Precisely, this includes handling missing data values and changing categorical variables to one-hot encoding. We then investigated the pattern of the data in a pairs plot. Figure 1 shows a subset of this plot.

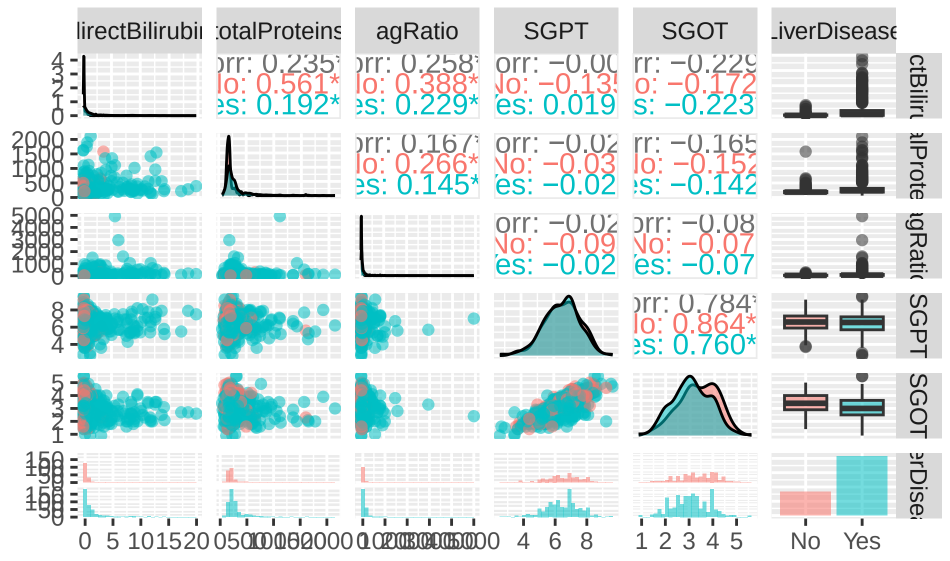

Figure 1: Pairs plot of variables

Next, we split the data into two equal parts to form training and testing sets. Once these were established, we then applied three different ensemble methods to the data: boosting, bagging, and random forest.

The first method was boosting, which involves fitting small trees and using the output of the current tree as the input of the next. This process can be represented as $\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x)$ where B is the number of trees, λ is the shrinkage parameter that controls the learning rate, and d is the interaction depth used to fit each tree $\hat{f}^b$ (King-Yu, 2024a). For our boosting, we used a Bernoulli distribution because the response is binary, 3000 trees, and an interaction depth of 1.

The second method was bagging, which involves the generation of multiple models using a learning method and combining them to form a final model. The learning method we applied was tree based. This process can be represented as $\hat{f}_{bag}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^b(x)$ where B is the number of trees and $\hat{f}^b$ is a tree (King-Yu, 2024b). For our bagging, we used all 10 predictor variables at each split.

The last method was random forest, which involves using a subset of the predictor variables to build multiple trees (King-Yu, 2024c). For our random forest, we used 5-fold cross validation to tune the number of trees and variables randomly sampled at each split. Precisely, we checked 1 to 100 trees and obtained 69 as the optimal value. We also checked 1 to 10 variables and obtained 2 as the optimal value.

Results

Table 1 below summarizes the results of each ensemble method. The adjusted Rand index (ARI) is quite poor for each of the methods. Of the three methods, bagging and random forest performed the best with an ARI value of 0.131 while boosting had an ARI value of 0.086. We see a similar pattern for the misclassification error rate where it is lowest for the bagging and random forest and slightly higher for boosting.

Table 1: Ensemble learning methods performance comparison

Method	Adjusted Rand Index	Misclassification Error Rate
Boosting	0.086	0.318
Bagging	0.131	0.277
Random Forest	0.123	0.281

Each of the methods also provided us with insight on the most important factors associated with liver disease. For the boosting method, A/G ratio and total proteins were the top two most influential factors as shown in Figure 2.

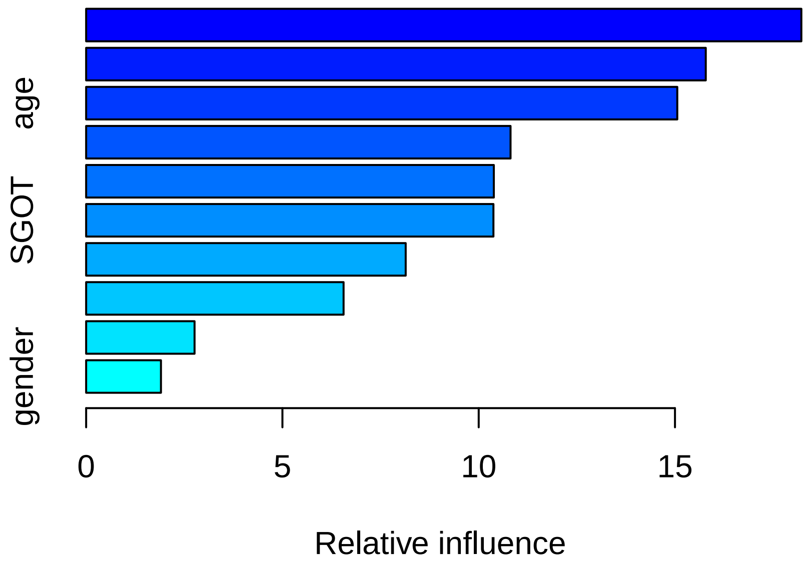

Figure 2: Pairs plot of variables

The bagging and random forest methods also produced similar results as illustrated in Figure 3 and Figure 4.

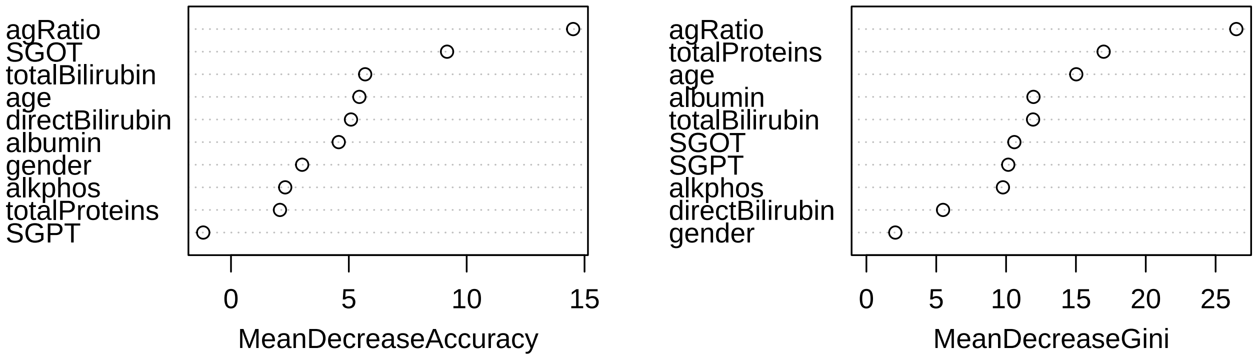

Figure 3: Important factors associated with liver disease for bagging

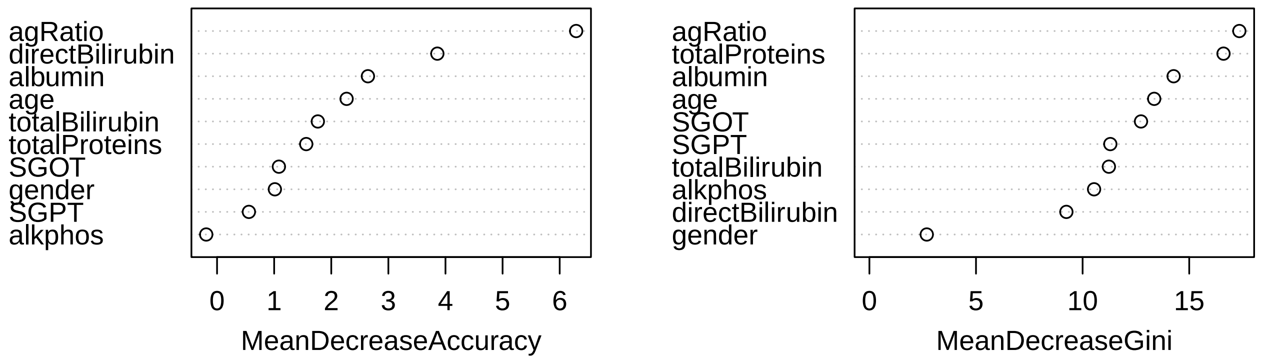

Figure 4: Important factors associated with liver disease for random forest

Conclusion

Overall, our study shows that the bagging and random forest methods provide the strongest predictions for liver disease compared to boosting. However, the accuracy of the models are still quite low and could be improved. A few ways to do this include feature engineering, performing more extensive data cleansing prior to modelling, and incorporating subject-matter expertise when interpreting results.

References

King-Yu, S. (2024a). *Statistical learning lecture 6: Ensemble learning part i boosting.*

King-Yu, S. (2024b). *Statistical learning lecture 7: Ensemble learning part II bagging.*

King-Yu, S. (2024c). *Statistical learning lecture 8: Ensemble learning part III random forest.*

Ramana, B., & Venkateswarlu, N. (2012). *ILPD (indian liver patient dataset).* UC Irvine Machine Learning Repository. https://doi.org/10.24432/C5D02C

Appendix

```
# ----- PACKAGE & DATA SETUP ----- #
library(tidyverse)
library(GGally)
library(gbm)
library(tree)
library(randomForest)
library(e1071)
library(kableExtra)

# Load data
liver_raw <- read_csv("Indian Liver Patient Dataset (ILPD).csv", col_names = FALSE)
colnames(liver_raw) <- c("age", "gender", "totalBilirubin", "directBilirubin",
                         "totalProteins", "albumin","agRatio", "SGPT", "SGOT",
                         "alkphos", "diagnosis")

# ----- DATA TRANSFORMATION ----- #
# Check for missing data
sum(sapply(liver_raw, function(col){ifelse(is.na(col), 1, 0)}))
sum(sapply(liver_raw, function(col){ifelse(sum(col == ""), 1, 0)}))

# Data transformation
liver <- liver_raw %>%
  mutate(diagnosis = ifelse(diagnosis == 1, 1, 0), # One hot encoding
         gender = ifelse(gender == "Male", 0, 1), # One hot encoding
         alkphos = ifelse(is.na(alkphos), mean(alkphos, na.rm=TRUE), alkphos) # Impute missi
         )

# Check for missing data
sum(sapply(liver, function(col){ifelse(is.na(col), 1, 0)}))
sum(sapply(liver, function(col){ifelse(sum(col == ""), 1, 0)}))
```

```r
# Full pairs plot
liverV <- within(liver, LiverDisease <- ifelse(diagnosis==1,"Yes", "No"))
ggpairs(data=liverV[,-c(11)], aes(colour=LiverDisease, alpha=0.4))

# Smaller pairs plot
liverV <- within(liver, LiverDisease <- ifelse(diagnosis==1,"Yes", "No"))
ggpairs(data=liverV[,-c(11,1,2,3,6,10)], aes(colour=LiverDisease, alpha=0.4))

# Split data into train and test
set.seed(1)
train.ind <- sample(1:nrow(liver), nrow(liver) / 2)
liver.train <- liver[train.ind,]
liver.test <- liver[-train.ind,]
liver.test.labs <- liver[-train.ind, "diagnosis"]

# ----- BOOSTING ------ #
set.seed(1)
boost.liver <- gbm(diagnosis ~ ., data = liver.train, distribution="bernoulli",
               n.trees = 3000, interaction.depth = 1)
boost.summary <- summary(boost.liver)
boost.summary
plot(boost.liver, i = "agRatio")
plot(boost.liver, i = "totalProteins")
# Function to harden probabilities
harden <- function(probs){
  n <- length(liver.test$diagnosis)
  pred.labs <- rep(0,n)
  for(i in 1:n){
    pred.labs[i] <- ifelse(probs[i] < 0.5, 0, 1)
  }
  return(pred.labs)
}

# Predictions
yhat.boost <- predict(boost.liver, newdata = liver.test,
                      n.trees = 3000, distribution = "bernoulli",
                      type = "response")
boost.pred.labs <- harden(yhat.boost)
tab1 <- table(liver.test.labs$diagnosis, boost.pred.labs)
tab1
# ----- BAGGING ------ #
# Decision tree
tree.liver <- tree(diagnosis ~ . - diagnosis, liver)
summary(tree.liver)
plot(tree.liver)
text(tree.liver, pretty = 0)
# Bagging
bagging.liver <- randomForest(as.factor(diagnosis) ~ ., data=liver, subset=train.ind,
                              mtry=10, importance=TRUE, type="class")
importance(bagging.liver)
varImpPlot(bagging.liver, main="")
# Predictions
liver.pred <- predict(bagging.liver, liver.test, type="class")
bagging.pred.labs <- harden(as.numeric(liver.pred)-1)
tab2 <- table(liver.test.labs$diagnosis, bagging.pred.labs)
tab2

# ----- RANDOM FOREST ------ #
# Tune mtry and ntree parameters
set.seed(1)
forest.tune <- tune.randomForest(as.factor(diagnosis) ~ .,
```

```r
                                 data = liver.train,
                                 mtry = 1:10,
                                 ntree=1:100,
                                 tunecontrol = tune.control(sampling = "cross",cross=5),
                                 type="class")
summary(forest.tune)
plot(forest.tune)
# Apply the random forest based on the optimal mtry and ntree values
forest.liver <- randomForest(as.factor(diagnosis) ~ ., data = liver,
                             subset = train.ind, mtry=2, ntree=69,
                             importance=TRUE, type="class")
forest.liver
varImpPlot(forest.liver, main="")
# Predictions
forest.pred <- predict(forest.liver, liver.test, type="class")
tab3 <- table(liver.test.labs$diagnosis, forest.pred)
tab3

# ----- RESULT SUMMARY ------ #
model <- c("Boosting", "Bagging", "Random Forest")
crand <- c(classAgreement(tab1)$crand,
           classAgreement(tab2)$crand,
           classAgreement(tab3)$crand)
diag <- c(1-classAgreement(tab1)$diag,
          1-classAgreement(tab2)$diag,
          1-classAgreement(tab3)$diag)
summary <- data.frame("Method" = model,
                      "Adjusted Rand Index" = round(crand,3),
                      "Misclassification Error Rate" = round(diag,3),
                      check.names = FALSE)
kable(summary)
```