Data Mining and Statistical Learning – Final Project

Hanchao Zhang[1]

I. Question 1 – Jaccard similarity

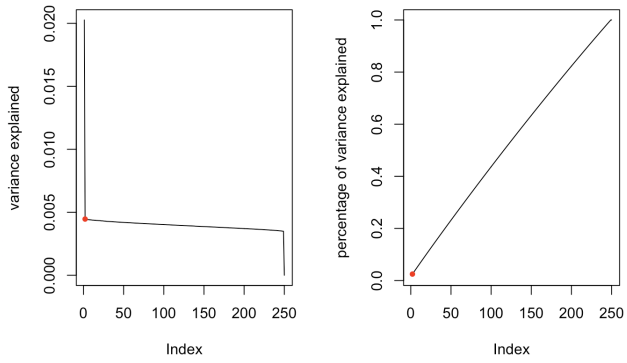
The definition of the Jaccard Similarity matrix is written as follow

$$Jac(A,B) = \frac{|A \bigcap B|}{|A \bigcup B|}$$

However, the way of calculating Jaccard Matrix is little different. THe way of computing the matrix is showed as follow

$$Jac(A,B) = \frac{|A = B = 1|}{|A = 1 \ or \ B = 1|}$$

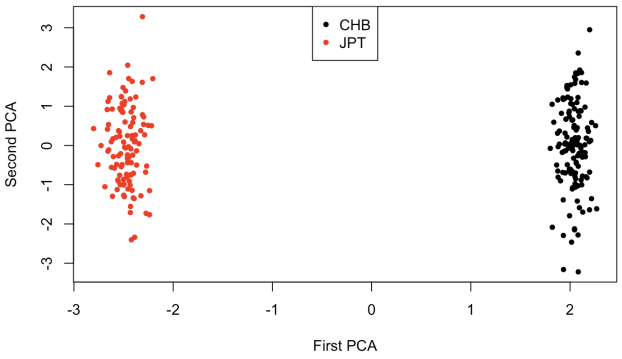
Fig. 1. Variance explained



The two figures show the variance explained and the cumulative variance explained by PCA. About 200 PCA can explain 80% of the variance.

However, as the plot below shows, the first two PCA can separate the two classes, Han Chinese in Beijing and Japanese in Tokyo pretty good.

Fig. 2. Classification of nationality by Jaccard Matrix



II. Question 2 – Classification Model for Cardiac Arrhythmia

Methods can be applied:

1. LDA
2. QDA
3. Best Subset
4. Forward Selection
5. Backward Selection
6. Stepwise Selection
7. Ridge Regression
8. Lasso regression
9. Elastic Net
10. Decision Tree
11. Bagging of the Trees
12. Random Forest
13. Gradient Boosting
14. ADA boosting
15. K-means
16. Hierarchical clustering

Methods not applied:

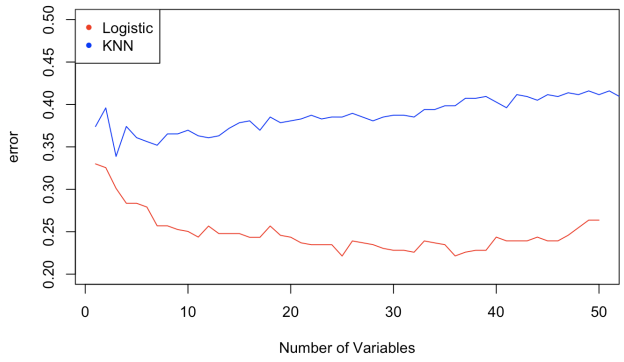
1. LDA is not used since it requires each covariate to be normally distributed.

2. Unsupervised learning is also not used since the label of the data is known.

3. Splines are not used also since most of the predictors are binary variables.

A. General Linear Model and KNN with variables selected by correlation

Fig. 3. GLM and KNN



The x lab of the figure is the number of the variables included in the model (selected by correlation).

As the figure shows, the Logistic regression has lower cross-validation error.

B. General Linear Model with variables selected by AIC

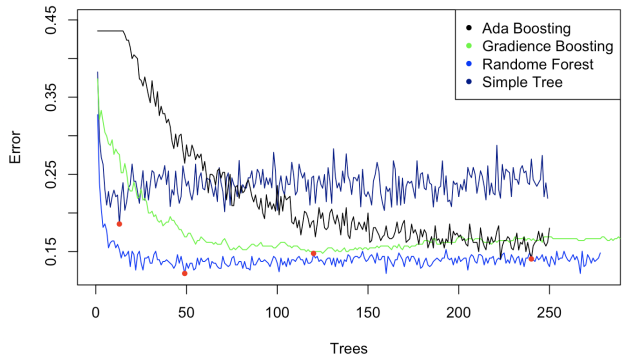
TABLE I
GENERAL LINEAR MODEL WITH VARIABLES SELECTED BY AIC

	Stepwise	Forward	Backward	subset
error	0.20	0.33	0.28	0.25

As the table shows, the stepwise selection has the lowest cv error.

C. Tree Based Methods

Fig. 4. Tree Based Methods

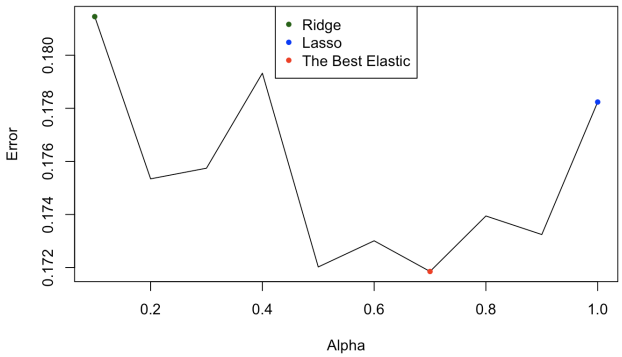


The data was imputed at first, the column with missing value more than 80% was then deleted.

As the figure shows, the Random Forest model has the lowest cv error.

D. Shrinkage Methods

Fig. 5. Shrinkage Methods



The Elastic net returns to the minimum cv error.

E. Model Comparison

Methods	error
Logistic(Number of covariates = 38)	0.14
KNN (Number of covariates = 21)	0.33
Simple Tree(ntree = 23)	0.17
Random Forest (ntree = 122, mtry = 49)	0.12
Vagging of the Tree (ntree = 232, mtry = 279)	0.14
Gradient Boosting(ntree = 123, depth = 1)	0.15
ADA Boosting(ntree = 242, depth = 1)	0.15
Stepwise Selection	0.20
Forward Selection	0.33
Backward Selection	0.28
Best Subset Selection	0.25
Elastic Net ($\alpha = 0.7$, $\lambda = 29$)	0.17
Lasso ($\lambda = 22$)	0.18
Ridge ($\lambda = 26$)	0.18

The random Forest returns the best model with the lowest prediction error.

Compare to the paper. The prediction error is much smaller than the VFI5 method. However, the VFI5 method can categorize the result into 16 classes, it makes sense that VFI5 model has higher prediction error.

REFERENCES

[1] Prokopenko D, Hecker J, Silverman EK, Pagano M, Nthen MM, Dina C, Lange C, Fier HL. Utilizing the Jaccard index to reveal population stratification in sequencing data: a simulation study and an application to the 1000 Genomes Project. Bioinformatics. 2016 May 1;32(9):1366-72. doi: 10.1093/bioinformatics/btv752. Epub 2015 Dec 31. PubMed PMID: 26722118; PubMed Central PMCID: PMC5860507.

[2] H. Altay Guvenir, Burak Acar, Gulsen Demiroz, Ayhan Cekin "A Supervised Machine Learning Algorithm for Arrhythmia Analysis." Proceedings of the Computers in Cardiology Conference, Lund, Sweden, 1997.

[3] https://www.genome.gov/10001688/international-hapmap-project/

APPENDIX

````
---
title: "Untitled"
output: html_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)

library(MASS)
library(tidyverse)
library(nnet)
library(foreach)
library(doMC)
library(class)
library(glmnet)
library(caret)
library(tree)
library(randomForest)
library(pROC)
library(gbm)
library(xgboost)
library(xtable)

library(glmnet)
registerDoMC(10)
```

# Q1
```{r}
dat.hapmap <- read.csv("/Users/hanchao/Library/Mobile␣Documents/com~apple~CloudDocs/
Courses/Cornell␣Semester␣3/Statistical␣Learning/FinalProject/SimulatedHapMap.csv")


```

```{r}
dat.exm <- dat.hapmap[,-1]

for (i in 1:ncol(dat.exm)) {
  if(!is.numeric(dat.exm[,i])){
    dat.exm[,i] <- as.numeric(as.character(dat.exm[,i]))
  }

}

dat.exm[dat.exm > 0] <- 1


dat.exm <- as.data.frame(t(dat.exm))


t1 <- Sys.time()

smi <- list()

for (i in 1:ncol(dat.exm)) {
  a1 = dat.exm[,i] == dat.exm[,1:ncol(dat.exm)] & dat.exm[,i] != 0
  a2 = dat.exm[,i] != 0 |  dat.exm[,1:ncol(dat.exm)] != 0
  smi[[i]] <- colSums(a1)/colSums(a2)
  print(i)
}

dat.jaccard <- do.call(cbind, smi)
dat.jaccard
````
````

```
t2 <- Sys.time()

t2-t1

dim(dat.jaccard)


```
Time difference of 9.851856 mins
Modify Chunk OptionsRun Current ChunkModify Chunk OptionsRun
All Chunks AboveRun Current ChunkModify Chunk OptionsRun All
Chunks AboveRun Current ChunkModify Chunk OptionsRun All Chunks
AboveRun Current Chunk
Console~/
Console
Terminal

~/

```{r}
write.csv(dat.jaccard, "/Users/hanchao/Library/Mobile␣Documents/
com~apple~CloudDocs/Courses/Cornell␣Semester␣3/Statistical␣Learning/FinalProject/jaccard.csv")

```

```{r}
dat.jaccard <- read.csv("/Users/hanchao/Library/Mobile␣Documents/
com~apple~CloudDocs/Courses/Cornell␣Semester␣3/Statistical␣Learning/FinalProject/jaccard.csv")
dat.jaccard <- dat.jaccard[,2:ncol(dat.jaccard)]
```

```{r}
obj.prcomp <- prcomp(dat.jaccard, scale = TRUE)
```

```{r}
par(mfrow = c(1,2))

plot( (obj.prcomp$sdev^2)/sum(obj.prcomp$sdev^2), type = "l", ylab = "variance␣explained")
points(x = 2, y = ((obj.prcomp$sdev^2)/sum(obj.prcomp$sdev^2))[2], col = "red", pch = 20)
plot( cumsum(obj.prcomp$sdev^2 )/ sum(obj.prcomp$sdev^2) , type = "l",
ylab = "percentage␣of␣variance␣explained")
points(x = 2, y = ((cumsum(obj.prcomp$sdev^2 )/ sum(obj.prcomp$sdev^2)))[2], col = "red", pch = 20)

```

```{r}
plot(-obj.prcomp$x[,1],obj.prcomp$x[,2], col = dat.hapmap$X, pch = 20, xlab = "First␣PCA",
ylab = "Second␣PCA")
legend("top", y = as.character(unique(dat.hapmap$X)), col = c("black", "red"), pch = 20)
```


```{r}
dat.ggplot <- data.frame(pca1 = obj.prcomp$x[,1],
pca2 = obj.prcomp$x[,2],
label = dat.hapmap$X)
```

Q 2

```{r}
dat.arr <- read.table("/Users/hanchao/Library/Mobile␣Documents/com~apple~CloudDocs/
Courses/Cornell␣Semester␣3/Statistical␣Learning/FinalProject/arrhythmia.data", sep = ",", header = F)

names(dat.arr)[ncol(dat.arr)] <- "Y"
```

```
dat.arr$Y <- ifelse(dat.arr$Y > 1, yes = 1, no = 0)

```

```{r}
for (i in names(dat.arr)) {
  if(!is.numeric(dat.arr[,i])){
    print(i)
  }
}

for (i in names(dat.arr)) {
  dat.arr[,i][dat.arr[,i] == "?"] <- NA
  dat.arr[,i] <- as.numeric(dat.arr[,i])

  if(max(dat.arr[,i], na.rm = T) == 16 & min(dat.arr[,i], na.rm = T) == 1){
    print(i)
  }
}



for (i in names(dat.arr)) {
  if( mean(is.na(dat.arr[,i])) > 0.5 ){
    print(i)
  }


}

#dat.arr <- dat.arr[,-which(names(dat.arr) == "V14")]

```
LDA

```{r}
cv.lda <- function(data = data, n = "fold", k = "knn-numer", y = "Class"){
  knn.error <- NULL
  dd <- data[sample(nrow(data), nrow(data), replace = F),]
  dat.splited <- split(dd, rep(1:n, each=nrow(dd)/(n)))

  for (i in 1:n) {
    train <- dat.splited[-i]
    train <- do.call(rbind, train)
    valid <- dat.splited[[i]]

    data.frame(
        vairable = 1:279,
        cor = as.numeric(cor(dat.arr$Y, dat.arr[,-280]))
        ) %>%
        arrange(desc(abs(cor))) -> dat.cor
  }

  k.error <- NULL
  for (K in k) {
    fit.lda <- lda(Y~., data = train[, c(dat.cor$vairable[1:K], which(names(train) == y)) ]
)
    pred <- predict(fit.lda, newdata = valid)
    k.error[K] <- mean(pred$class != valid$Y)

  }
  k.error

}
```

```
err.lda <- cv.lda(data = dat.arr, n = 5, k = 1:100, y = "Y")


```


```{r}
cv.lda <- function(data = data, n = "fold", k = "knn-numer", y = "Class"){
  knn.error <- NULL
  dd <- data[sample(nrow(data), nrow(data), replace = F),]
  dat.splited <- split(dd, rep(1:n, each=nrow(dd)/(n)))
  for (K in k) {
    k.error <- NULL
    for (i in 1:n) {
      train <- dat.splited[-i]
      train <- do.call(rbind, train)
      valid <- dat.splited[[i]]

      data.frame(
        vairable = 1:279,
        cor = as.numeric(cor(dat.arr$Y, dat.arr[,-280]))
        ) %>%
        arrange(desc(abs(cor))) -> dat.cor

      dat.train <- train[,c(dat.cor$vairable[1:K],280)]
      dat.valid <- valid[,c(dat.cor$vairable[1:K],280)]
      fit.lda <- lda(Y~., data = dat.train)
      pred <- predict(fit.lda, newdata = dat.valid)
      k.error[i] <- mean(pred$class != dat.valid$Y)

      }
    knn.error[K] <- mean(k.error)
  }

  z <- data.frame(k = k,
                  error = knn.error)
  z
}


err.lda <- cv.lda(data = dat.arr, n = 10, k = 1:50, y = "Y")
err.lda[which.min(err.lda$error),]

```

Logistic

```{r}
cv.logistic <- function(data = data, n = "fold", k = "knn-numer", y = "Class"){
  knn.error <- NULL
  dd <- data[sample(nrow(data), nrow(data), replace = F),]
  dat.splited <- split(dd, rep(1:n, each=nrow(dd)/(n)))

  for (i in 1:n) {
    train <- dat.splited[-i]
    train <- do.call(rbind, train)
    valid <- dat.splited[[i]]

    data.frame(
        vairable = 1:279,
        cor = as.numeric(cor(dat.arr$Y, dat.arr[,-280]))
        ) %>%
        arrange(desc(abs(cor))) -> dat.cor
  }

  k.error <- NULL
```

```
  for (K in k) {
    fit.lda <- glm(Y~., data = train[, c(dat.cor$vairable[1:K],
    which(names(train) == y)) ] , family = binomial() )
    pred <- predict(fit.lda, newdata = valid, type = "response")
    dat.roc <- roc(valid[,y]~pred)
    thres <- coords(dat.roc,"best")[1]
    pred <- ifelse(pred < thres, yes = 0, no = 1)
    k.error[K] <- mean(pred != valid$Y)

  }
  k.error

}

err.logistic <- cv.logistic(data = dat.arr, n = 5, k = 1:100, y = "Y")

```

```{r}

cv.multinomial <- function(data = data, n = "fold", k = "knn-numer", y = "Class"){
  knn.error <- NULL
  dd <- data[sample(nrow(data), nrow(data), replace = F),]
  dat.splited <- split(dd, rep(1:n, each=nrow(dd)/(n)))
  for (K in k) {
    k.error <- NULL
    for (i in 1:n) {
      train <- dat.splited[-i]
      train <- do.call(rbind, train)
      valid <- dat.splited[[i]]

      data.frame(
        vairable = 1:279,
        cor = as.numeric(cor(dat.arr$Y, dat.arr[,-280]))
        ) %>%
        arrange(desc(abs(cor))) -> dat.cor

      dat.train <- train[,c(dat.cor$vairable[1:K],280)]
      dat.valid <- valid[,c(dat.cor$vairable[1:K],280)]
      fit.lda <- glm(Y~., data = dat.train, family = binomial)
      #fit.lda <- stepAIC(fit.lda)
      pred <- predict(fit.lda, newdata = dat.valid, type = "response")
      dat.roc <- roc(dat.valid$Y ~ pred)
      thres <-coords(dat.roc, "best")[1]
      pred <- ifelse(pred < thres, yes = 0, no = 1)

      k.error[i] <- mean(pred != dat.valid$Y)


      }
    knn.error[K] <- mean(k.error)


  }

  z <- data.frame(k = k,
                  error = knn.error)
  z
}


err.multinomial <- cv.multinomial(data = dat.arr, n = 5, k = 1:100, y = "Y")

err.multinomial[which.min(err.multinomial$error),]

```
KNN

```{r}
cv.knn <- function(data = data, n = "fold", k = "knn-numer", y = "Class"){
  knn.error <- NULL
  dd <- data[sample(nrow(data), nrow(data), replace = F),]
  dat.splited <- split(dd, rep(1:n, each=nrow(dd)/(n)))
  for (K in k) {
    k.error <- NULL
    for (i in 1:n) {
      train <- dat.splited[-i]
      train <- do.call(rbind, train)
      valid <- dat.splited[[i]]
      Train <- data.frame(scale(train[,-which(names(train) == y)]),
                  Class = train[,y])
      Valid <- data.frame(scale(valid[,-which(names(valid) == y)]),
                  Class = valid[,y])
      pred.knn <- knn(Train[,-which(colnames(Train) == "Class")],
      Valid[,-which(colnames(Valid) == "Class")], Train[,"Class"], k = K)
      k.error[i] <- mean(pred.knn != Valid[,"Class"])
      }
    knn.error[K] <- mean(k.error)
  }

  z <- data.frame(k = k,
                knn.error)
  z
}

dat.arr.knn <- dat.arr[,colMeans(is.na(dat.arr)) == 0]


for (i in names(dat.arr.knn)) {
  if(mean(duplicated(dat.arr.knn[,i])) > 0.9 ){
    dat.arr.knn[,i] <- NA
  }
}

dat.arr.knn <- dat.arr.knn[,colMeans(is.na(dat.arr.knn)) == 0 ]

dat.arr.knn <- data.frame(dat.arr.knn, Y = dat.arr$Y)

err.knn <- cv.knn(data = dat.arr.knn, n = 5, k = 1:100, y = "Y")

```

```{r}
cv.qda <- function(data = data, n = "fold", k = "knn-numer", y = "Class"){
  knn.error <- NULL
  dd <- data[sample(nrow(data), nrow(data), replace = F),]
  dat.splited <- split(dd, rep(1:n, each=nrow(dd)/(n)))

  for (i in 1:n) {
    train <- dat.splited[-i]
    train <- do.call(rbind, train)
    valid <- dat.splited[[i]]

    data.frame(
        vairable = 1:279,
        cor = as.numeric(cor(dat.arr$Y, dat.arr[,-280]))
        ) %>%
        arrange(desc(abs(cor))) -> dat.cor
  }

  k.error <- NULL
  for (K in k) {
    fit.lda <- qda(Y~., data = train[, c(dat.cor$vairable[1:K], which(names(train) == y)) ]
)
    pred <- predict(fit.lda, newdata = valid)
    k.error[K] <- mean(pred$class != valid$Y)
```

```
  }
  k.error

}


```
Stepwise

```{r}
cv.backward <- function(data = data, n = "fold", k = "knn-numer", y = "Class", direction = "both"){
  knn.error <- NULL
  dd <- data[sample(nrow(data), nrow(data), replace = F),]
  dat.splited <- split(dd, rep(1:n, each=nrow(dd)/(n)))

  for (i in 1:n) {
    train <- dat.splited[-i]
    train <- do.call(rbind, train)
    valid <- dat.splited[[i]]
    fit.lda <- glm(Y~., data = train, family = binomial() )
    fit.lda <- step(fit.lda, glm(Y~., data = train, family = binomial()), direction = direction)
    pred <- predict(fit.lda, newdata = valid, type = "response")
    dat.roc <- roc(valid[,y]~pred)
    thres <- coords(dat.roc,"best")[1]
    pred <- ifelse(pred < thres, yes = 0, no = 1)
    k.error <- mean(pred != valid$Y)
  }

  k.error

}

err.forward <- cv.backward(data = dat.arr.knn, n = 5, k = 1:100, y = "Y", direction = "forward")
err.stepwise <- cv.backward(data = dat.arr.knn, n = 5, k = 1:100, y = "Y", direction = "both")

err.backward <- cv.backward(data = dat.arr.knn, n = 5, k = 1:100, y = "Y")


tbl.aic <- data.frame(error = c(err.stepwise, err.forward,na.omit(err.backward)))
rownames(tbl.aic) <- c("Stepwise", "Forward", "Backward")

t(tbl.aic) %>%
  xtable()
```

```{r}
plot(err.multinomial, type = "l",cex = 0.5, col = "red", ylim = c(0.2,0.5),
xlab = "Number of Variables")
lines(err.knn, col = "blue")

legend(x="topleft", legend = c( "Logistic", "KNN"), col = c( "red", "blue"), pch = 20)


```
Tree

```{r}
cv.tree2 <- function(data = data, n = "fold", k = "knn-numer", y = "Class"){
  knn.error <- NULL
  dd <- data[sample(nrow(data), nrow(data), replace = F),]
  dat.splited <- split(dd, rep(1:n, each=nrow(dd)/(n)))
  for (K in k) {
    k.error <- NULL
    for (i in 1:n) {
      train <- dat.splited[-i]
      train <- do.call(rbind, train)
      valid <- dat.splited[[i]]
```

```
      my.tree <- tree(factor(Y)~., data = train)
      cv.model.pruned <- prune.misclass(my.tree, best=K)
      pred <- predict(cv.model.pruned, newdata = valid)[,2]
      dat.roc <- roc(valid$Y ~ pred)
      thres <- coords(dat.roc, "best")[1]
      pred.knn <- ifelse(pred < thres, yes = 0, no = 1)



      k.error[i] <- mean(pred.knn != valid[,"Y"])
      }
    knn.error[K] <- mean(k.error)
  }

  z <- data.frame(k = k,
                  knn.error)
  z
}

err.tree <- NULL
for (i in 2:500) {
  err.tree[i-1] <- as.numeric(na.omit(cv.tree2(data = dat.arr.tree, n = 5, k = i, y = "Y")$knn.error))

}
err.tree

plot(err.tree, type = "l")
points(which.min(err.tree), min(err.tree), pch = 20, col = "red")

```

```{r}
dat.arr.tree <- dat.arr[,-which(names(dat.arr) == "V14")]

my.tree <- tree(factor(Y)~., data = dat.arr.tree)

mytree.cv <- cv.tree(my.tree,FUN=prune.misclass,K=10)

plot(x = mytree.cv$size, y = mytree.cv$dev, type = "b", xlab = "Size", ylab = "Deviance")
points(x = mytree.cv$size[which.min(mytree.cv$dev)],
y = mytree.cv$dev[which.min(mytree.cv$dev)], col = "red")

best.size <- mytree.cv$size[which(mytree.cv$dev==min(mytree.cv$dev))]

cv.model.pruned <- prune.misclass(my.tree, best=best.size[1])

plot(cv.model.pruned)
text(cv.model.pruned, use.n=TRUE, all=TRUE, cex=.8)
```
RF & bag of trees

```{r}
hd.imputed <- rfImpute(Y~.-Y,data=dat.arr.tree)

err.rf1 <- list()
for (i in seq(1, 278, by = 1)) {
  obj.tree1 <- randomForest(factor(Y)~., data = hd.imputed, mtry=i,ntree=500)
  err.rf1[[i]] <- (obj.tree1$err.rate[,1])

}

a <- do.call(cbind, err.rf1)
err.rf <- apply(a, 2,min)

plot(na.omit(err.rf), type = "l", ylab = "Error", xlab = "Split",
main = "Random␣Forest␣and␣Bagging␣of␣Tree")
```

```
points(which.min(err.rf),min(err.rf), pch = 20, col = "red")

```
GBM

```{r}
library(Matrix)

##need to exclude missing

train.new <- na.omit(dat.arr.tree)
test.new <- na.omit(valid)

covariates_matrix = sparse.model.matrix(Y ~ ., data = train.new)[,-1]
output_vector = train.new[, 'Y'] == 1

#covariates_test_matrix = sparse.model.matrix(AHD ~ ., data = test.new)[,-1]
#output_test_vector = test.new[, 'AHD'] == "Yes"

##Create grid of parameters to try
xgb_grid = expand.grid(
  eta = c(0.1, 0.01, 0.001, 0.0001),
  max_depth = c(1,2)
)

for(i in 1:nrow(xgb_grid)){
assign(paste0("xgb_params.", i), list(
  objective = "binary:logistic",
  eta = xgb_grid[i,1],
  max.depth = xgb_grid[i,2],
  eval_metric = "error"
))}

param.list <- list(xgb_params.1, xgb_params.2, xgb_params.3,
xgb_params.4, xgb_params.5, xgb_params.6,xgb_params.7,xgb_params.8)

set.seed(12345)
best.cv.error <- data.frame()
Eval.cv <- list()
for(i in 1:nrow(xgb_grid)){
xgb_cv = xgb.cv(params = param.list[[i]],
                  data = covariates_matrix,
                  label = output_vector,
                  nrounds = 500,
                  nfold = 5,
                  prediction = TRUE,
                  showsd = TRUE,
                  stratified = TRUE,
                  verbose = TRUE
)

Eval.cv[[i]] <- xgb_cv$evaluation_log[,c(1,4)]
best.cv.error[i,1] <- min(Eval.cv[[i]]$test_error_mean)
}

rownames(best.cv.error) <- c("eta=0.1,␣depth=1", "eta=0.01,␣depth=1", "eta=0.001,␣depth=1", "eta=0.0001,
depth=1", "eta=0.1,␣depth=2", "eta=0.01,␣depth=2", "eta=0.001,␣depth=2", "eta=0.0001,␣depth=2")

which.min(best.cv.error$V1)


```

```{r}
library('e1071')

##find shrinakge parameter
caretGrid.depth.1 <- expand.grid(interaction.depth=1, n.trees = 1000,
```

```
shrinkage=seq(0, 0.01, by = 0.001), n.minobsinnode=10)

set.seed(123)

gbm.caret.depth.1 <- caret::train(factor(Y) ~ ., distribution = "adaboost",
data = train.new, method = "gbm", trControl=trainControl(method="cv", number=5),
verbose=F, tuneGrid=caretGrid.depth.1)

best.tune.1 <- gbm.caret.depth.1$bestTune$shrinkage

names(gbm.caret.depth.1)

gbm.caret.depth.1$results

caretGrid.depth.2 <- expand.grid(interaction.depth=2, n.trees = 1:250,
shrinkage=seq(0, 0.01, by = 0.001), n.minobsinnode=10)

set.seed(123)
gbm.caret.depth.2 <- caret::train(factor(Y) ~ ., distribution = "adaboost",
data = train.new, method = "gbm", trControl=trainControl(method="cv", number=5),
verbose=F, tuneGrid=caretGrid.depth.2)

gbm.caret.depth.2$results[,c(1,2,4,5)]

best.tune.2 <- gbm.caret.depth.2$bestTune$shrinkage

##Fit the models with the parameters selected from cross validation
ada.train <- train.new
ada.train$AHD <- ifelse(ada.train$AHD=="No", 0, 1)
adaboost.depth.1 <- gbm(AHD ~ ., distribution="adaboost", data=ada.train,
shrinkage=best.tune.1, n.trees=2000)
adaboost.depth.2 <- gbm(AHD ~ ., distribution="adaboost", data=ada.train,
shrinkage=best.tune.2, n.trees=2000)

##Predict with 500 trees
pred.out.ada.depth.1 <- ifelse(predict(adaboost.depth.1, test.new, n.trees = 500,
type="response") >0.5, "Yes", "No")
misclass.ada.depth.1 <- mean(pred.out.ada.depth.1 != test.new$AHD)
misclass.ada.depth.1
```

```{r}

err.ada <- NULL

for (i in 1:250) {
  caretGrid.depth.2 <- expand.grid(interaction.depth=2, n.trees = i,
  shrinkage=seq(0, 0.01, by = 0.001), n.minobsinnode=10)
  gbm.caret.depth.2 <- caret::train(factor(Y) ~ ., distribution = "adaboost",
  data = train.new, method = "gbm", trControl=trainControl(method="cv", number=5),
  verbose=F, tuneGrid=caretGrid.depth.2)
  err.ada[i] <- gbm.caret.depth.2$results$Accuracy[which.max(gbm.caret.depth.2$results$Accuracy)]
}

```

```{r}

plot(na.omit(err.rf), type = "l", ylab = "Error", xlab = "Trees", col = "blue",
ylim = c(0.12,0.45), xlim = c(1,240))
points(which.min(err.rf),min(err.rf), pch = 20, col = "red")

lines(err.tree, type = "l", col = "dark␣blue")
points(which.min(err.tree), min(err.tree), pch = 20, col = "red")

lines(Eval.cv[[1]], col = "green")
points(which.min(Eval.cv[[1]]$test_error_mean), min(Eval.cv[[1]]$test_error_mean),
pch = 20, col = "red")
```

```
lines((1-err.ada))
points(x = which.min(1-err.ada), y = min((1-err.ada)), pch = 20, col = "red")
legend(x = "topright",legend = c("Ada␣Boosting", "Gradience␣Boosting","Randome␣Forest",
"Simple␣Tree"), col = c("black", "green", "blue", "dark␣blue"), pch = 20)

```

```{r}
dat.arr.lasso <- na.omit(dat.arr.tree)
x <- model.matrix(Y~.,dat.arr.lasso)[,-1] # First column corresponds to the intercept and is removed
y <- dat.arr.lasso$Y

l <- seq(0,1,by = 0.1)
err.shrinkage <- list()
for (i in 1:10) {
  enet.cv = cv.glmnet(x,y,alpha=l[i],nfolds=5)
  err.shrinkage[[i]] <- (enet.cv$cvm)


}

dat.lasso <- do.call(cbind,err.shrinkage)

dat.lasso[30,] ==  min(err.lasso)
err.lasso <- apply(dat.lasso, 1, min)

which(dat.lasso == min(err.lasso))

plot(err.lasso, type = "l", ylab = "Error", xlab = "Lambda")
points(x = 1, err.lasso[1], col = "dark␣green", pch = 20)
points(x = length(err.lasso), err.lasso[length(err.lasso)], col = "blue", pch = 20)
points(x = which.min(err.lasso), min(err.lasso), col = "red", pch = 20)
legend(x = "top",legend = c("Ridge", "Lasso","The␣Best␣Elastic"),
col = c("dark␣green", "blue", "red"), pch = 20)

```

```{r}
tbl.cp <- data.frame(error = c(min(err.logistic), min(err.rf), min(err.stepwise), min(err.lasso) ))
rownames(tbl.cp) <- c("logistic", "random␣forest", "stepwise", "Elastic")
(tbl.cp) %>%
  xtable()

which.min(err.rf)
```