```r
library(dplyr)
library(MASS)
library(glmnet)
library(randomForest)
library(e1071)
library(ggplot2)
library(reshape)
library(gridExtra)

options(expressions = 5e5)

a = read.csv("pdata.csv")
dim(a)
colnames(a)[dim(a)[2]] = c("Class")
colMeans(a)
t.data = a

# standardize the dataset
t.data[,-dim(t.data)[2]] = scale(t.data[,-dim(t.data)[2]])
colMeans(t.data)
apply(t.data, 2, sd)
mean(t.data$Class)
attach(t.data)

# data structure and rates
n1 = dim(t.data %>% filter(Class == 1))[1]
n0 = dim(t.data %>% filter(Class == 0))[1]
n = n1 + n0
p = dim(t.data)[2]-1

# Modelling factors
iterations = 100
Dlearn_rate = 0.5
sampling.rate = 1
weight = c("0" = 1/n0, "1" = 1/n1)

# train and test error rate matrix
train_error = matrix(0, nrow = iterations, ncol = 7)
colnames(train_error) = c("RF", "R-SVM", "Log", "Log-LASSO","Log-Ridge", "OOB", "OOBsd")

cv_error = matrix(0, nrow = iterations, ncol = 3)
colnames(cv_error) = c("R-SVM", "Log-LASSO","Log-Ridge")

```

```r
test_error = matrix(0, nrow = iterations, ncol = 5)
colnames(test_error) = c("RF", "R-SVM", "Log", "Log-LASSO","Log-Ridge")

lasso.coef = matrix(0, ncol = iterations, nrow = p+1)
ridge.coef = matrix(0, ncol = iterations, nrow = p+1)

# convert to data frame
train_error = data.frame(train_error)
test_error = data.frame(test_error)

# time of cv and fit
time.cv = matrix(0, nrow = iterations, ncol = 3)
colnames(time.cv) = c("SVM", "LASSO", "Ridge")

time.fit = matrix(0, nrow = iterations, ncol = 4)
colnames(time.fit) = c("SVM", "LASSO", "Ridge", "RF")

# rf importance
rf.importace = matrix(0, nrow = iterations, ncol = 400)

# sampling from t.data
sampling = sample(n,n*sampling.rate)
sampling.data = data.frame(t.data[sampling,])
sampling.n = dim(sampling.data)[1]

# preparation for lasso and ridge
X = model.matrix(Class ~., sampling.data)[,-1]
y = sampling.data$Class


# 100 iteration for error rates, time, and coefficients
for(m in 1:iterations){


  # create a training data vector for dividing the data set.
  train = sample(sampling.n, sampling.n*Dlearn_rate)

  dat   = data.frame(sampling.data[train,])
  datt  = data.frame(sampling.data[-train,])

  # svm
  # record svm cv time
  ptm = proc.time()
  tune.svm  =   tune(svm, as.factor(Class)~., data=dat,
                     ranges = list(cost = 10^seq(-2,2,length.out = 5),
                                   gamma = 10^seq(-2,2,length.out = 5)), scale = F)
```

```r
ptm = proc.time() - ptm
time.cv[m,1]  = ptm["elapsed"]

# record cv error
cv_error[m,1] = tune.svm$best.performance
# tune.svm$performances
# tune.svm$best.parameters[1]
# record svm fit time
ptm = proc.time()
# svm.fit = svm(as.factor(Class)~., data = dat,
#              cost = tune.svm$best.parameters[1], gamma = tune.svm$best.parameters[2])
svm.fit = tune.svm$best.model
ptm = proc.time() - ptm
time.fit[m,1]  = ptm["elapsed"]

svm.pred = predict(svm.fit, dat, type = "class")
train_error[m,2] = mean(dat[,dim(sampling.data)[2]] != svm.pred)
# table(dat[,dim(sampling.data)[2]],svm.pred)
svm.pred = predict(svm.fit, datt, type = "class")
test_error[m,2] = mean(datt[,dim(sampling.data)[2]] != svm.pred)
# table(datt[,dim(sampling.data)[2]], svm.pred)

# logistic regression
log.mod = glm(Class ~., data = dat, family = "binomial",
              weights = ifelse(dat$Class == 0, 1/n0, 1/n1))
log.pred = predict(log.mod, newdata = sampling.data[train,], type = "response")
log.pred = ifelse(log.pred > 0.5, 1, 0)
train_error[m,3] = mean( sampling.data[train,dim(sampling.data)[2]]!= log.pred)
log.pred = predict(log.mod, newdata = sampling.data[-train,], type = "response")
log.pred = ifelse(log.pred > 0.5, 1, 0)
test_error[m,3] = mean( sampling.data[-train,dim(sampling.data)[2]]!= log.pred)

# lasso cross validation and tune lambda
# record lasso cv time
ptm = proc.time()
cv.lasso = cv.glmnet(X[train,], y[train], alpha = 1, family = "binomial",
                     intercept = T, type.measure="class",
                     weights = ifelse(y[train] == 0, 1/n0, 1/n1))
ptm = proc.time() - ptm
time.cv[m,2]  = ptm["elapsed"]

cv_error[m,2] = min(cv.lasso$cvm)
bestlam = cv.lasso$lambda.min

# record lasso fit time
ptm = proc.time()
```

```r
  lasso.mod = glmnet(X[train,], y[train], alpha = 1, family = "binomial",
                     intercept = T, lambda = bestlam,
                     standardize = F)
  ptm = proc.time() - ptm
  time.fit[m,2]  = ptm["elapsed"]

  lasso.coef[,m] = coef(lasso.mod)[,1]
  lasso.pred = predict(lasso.mod, s = bestlam, newx = X[train,], type ="class")
  train_error[m,4] = mean(y[train] != lasso.pred)
  lasso.pred = predict(lasso.mod, s = bestlam, newx = X[-train,], type="class",)
  test_error[m,4] = mean(y[-train] != lasso.pred)

  # ridge cross validation and tune lambda
  # record ridge cv time
  ptm = proc.time()
  cv.ridge = cv.glmnet(X[train,], y[train], alpha = 0, family = "binomial",
                       intercept = T, type.measure="class",
                       weights = ifelse(y[train] == 0, 1/n0, 1/n1))
  ptm = proc.time() - ptm
  time.cv[m,3]  = ptm["elapsed"]

  cv_error[m,3] = min(cv.ridge$cvm)
  bestlam = cv.ridge$lambda.min

  # record ridge fit time
  ptm = proc.time()
  ridge.mod = glmnet(X[train,], y[train], alpha = 0, family = "binomial",
                     intercept = T, lambda = bestlam,
                     standardize = F)
  ptm = proc.time() - ptm
  time.fit[m,3]  = ptm["elapsed"]

  ridge.coef[,m] = as.matrix(coef(ridge.mod))
  ridge.pred = predict(ridge.mod, s = bestlam, newx = X[train,], type = "class")
  train_error[m,5] = mean(y[train] != ridge.pred)
  ridge.pred = predict(ridge.mod, s = bestlam, newx = X[-train,], type = "class")
  test_error[m,5] = mean(y[-train] != ridge.pred)

  #random forest with 500 bootstrapped trees
  ptm = proc.time()
  rf = randomForest(x = sampling.data[train,-dim(sampling.data)[2]],
                    y = as.factor(sampling.data[train,dim(sampling.data)[2]]), data =
sampling.data[train,],
                    mtry = sqrt(p), classwt = weight)
  ptm = proc.time() - ptm
  time.fit[m,4]  = ptm["elapsed"]

  rf.pred = predict(rf, sampling.data[train,-dim(sampling.data)[2]], type = "class")
  train_error[m,1] = mean(sampling.data[train,dim(sampling.data)[2]] != rf.pred)
  train_error[m,6] = mean(rf$err.rate[,1])
  train_error[m,7] = sd(rf$err.rate[,1])
  rf.pred = predict(rf, sampling.data[-train,], type = "class")
  test_error[m,1] = mean(sampling.data[-train,dim(sampling.data)[2]] != rf.pred)


}

############################################
############################################

# store error rate and coef
write.csv(ridge.coef, file = "D50_rdige_coef.csv")
write.csv(lasso.coef, file = "D50_lasso_coef.csv")

write.csv(cv_error, file = "D50_cv_error.csv")
write.csv(test_error, file = "D50_test_error.csv")
write.csv(train_error, file = "D50_train_error.csv")

write.csv(time.cv, file = "D50_time_cv.csv")
write.csv(time.fit, file = "D50_time_fit.csv")

# read csv files of project results
D5.r.coef = read.csv("D50_rdige_coef.csv")
D5.l.coef = read.csv("D50_lasso_coef.csv")

D5.cv.error = read.csv("D50_cv_error.csv")
D5.test.error = read.csv("D50_test_error.csv")
D5.train.error = read.csv("D50_train_error.csv")
colnames(D5.train.error)[7:8] = c("RF.OOB", "RF.OOBsd")

D5.time.cv = read.csv("D50_time_cv.csv")
D5.time.fit = read.csv("D50_time_fit.csv")

D9.r.coef = read.csv("D90_rdige_coef.csv")
D9.l.coef = read.csv("D90_lasso_coef.csv")

D9.cv.error = read.csv("D90_cv_error.csv")
D9.test.error = read.csv("D90_test_error.csv")
D9.train.error = read.csv("D90_train_error.csv")
colnames(D9.train.error)[7:8] = c("RF.OOB", "RF.OOBsd")

D9.time.cv = read.csv("D90_time_cv.csv")
```

```
D9.time.fit = read.csv("D90_time_fit.csv")

# boxplot of error rates for each nlearn
f1_1 = ggplot(melt(D5.train.error[,2:7]), aes(x = variable, y = value, color = variable)) +
  geom_boxplot() + ylim(0,0.25) + theme(legend.position="none") + scale_color_brewer(palette="Dark2") +
  labs(x = element_blank(), y = "Train Error Rate", title = expression(n[learn]~"="~n/2~~Train~Error~Rate))
f1_2 = ggplot(melt(D5.test.error[,2:6]), aes(x = variable, y = value, color = variable)) +
  geom_boxplot() + ylim(0,0.25) + theme(legend.position="none") + scale_color_brewer(palette="Dark2") +
  labs(x = element_blank(), y = "Test Error Rate", title = expression(n[learn]~"="~n/2~~Test~Error~Rate))
f1_3 = ggplot(melt(D5.cv.error[,2:4]), aes(x = variable, y = value, color = variable)) +
  geom_boxplot() + ylim(0,0.25) + theme(legend.position="none") + scale_color_brewer(palette="Dark2") +
  labs(x = element_blank(), y = "CV Error Rate", title = expression(n[learn]~"="~n/2~~CV~Error~Rate))

f1_4 = ggplot(melt(D9.train.error[,2:7]), aes(x = variable, y = value, color = variable)) +
  geom_boxplot() + ylim(0,0.25) + theme(legend.position="none") + scale_color_brewer(palette="Dark2") +
  labs(x = element_blank(), y = "Train Error Rate", title = expression(n[learn]~"="~0.9~n~~Train~Error~Rate))
f1_5 = ggplot(melt(D9.test.error[,2:6]), aes(x = variable, y = value, color = variable)) +
  geom_boxplot() + ylim(0,0.25) + theme(legend.position="none") + scale_color_brewer(palette="Dark2") +
  labs(x = element_blank(), y = "Test Error Rate", title = expression(n[learn]~"="~0.9~n~~Test~Error~Rate))
f1_6 = ggplot(melt(D9.cv.error[,2:4]), aes(x = variable, y = value, color = variable)) +
  geom_boxplot() + ylim(0,0.25) + theme(legend.position="none") + scale_color_brewer(palette="Dark2") +
  labs(x = element_blank(), y = "CV Error Rate", title = expression(n[learn]~"="~0.9~n~~CV~Error~Rate))
print(f1_3)
f1.1 = grid.arrange(f1_1, f1_2, f1_3, nrow = 1, widths = c(1.5,1.5,1))
f1.2 = grid.arrange(f1_4, f1_5, f1_6, nrow = 1, widths = c(1.5,1.5,1))

# cv ridge error
cv.ridge$lambda
length(cv.ridge$lambda)
which.min(cv.ridge$cvm)

cv.ridge.coef = matrix(0, nrow = length(cv.ridge$lambda), ncol = 1)
for (i in 1:length(cv.ridge$lambda)) {
  cv.ridge.coef[i,] = sqrt(sum(coef(cv.ridge, s = cv.ridge$lambda[i])[-1,]^2))
```

```r
}
cv.ridge.rate.coef = round(cv.ridge.coef/cv.ridge.coef[length(cv.ridge$lambda),1],3)
cv.ridge.error.and.rate.coef = data.frame(cbind(cv.ridge.rate.coef,cv.ridge$cvm))

# cv lasso error
length(cv.lasso$lambda)
which.min(cv.lasso$cvm)
cv.lasso.coef = matrix(0, nrow = length(cv.lasso$lambda), ncol = 1)
for (j in 1:length(cv.lasso$lambda)) {
  cv.lasso.coef[j,] = sqrt(sum(coef(cv.lasso, s = cv.lasso$lambda[j])[-1,]^2))
}
cv.lasso.rate.coef = round(cv.lasso.coef/cv.lasso.coef[length(cv.lasso$lambda),1],3)
cv.lasso.error.and.rate.coef = data.frame(cbind(cv.lasso.rate.coef,cv.lasso$cvm))

D9.cv.lasso.error.and.rate.coef = read.csv("D90_cv.lasso.error.and.rate.coef.csv")
D9.cv.ridge.error.and.rate.coef = read.csv("D90_cv.ridge.error.and.rate.coef.csv")

# svm cv error rate
cv.svm.error = data.frame(tune.svm$performances)
D9.cv.svm.error = read.csv("D9_cv.svm.error.csv")

# plot cv error rate for lasso, ridge, and svm
f2_1 = ggplot() +
  geom_line(data = cv.lasso.error.and.rate.coef, aes(x = X1, y = X2, color = "LASSO")) +
  geom_point(data = cv.lasso.error.and.rate.coef, aes(x = X1[which.min(X2)],
                                                      y = min(X2), color = "LASSO")) +
  geom_line(data = cv.ridge.error.and.rate.coef, aes(x = X1, y = X2, color = "Ridge")) +
  geom_point(data = cv.ridge.error.and.rate.coef, aes(x = X1[which.min(X2)],
                                                      y = min(X2), color = "Ridge")) +
  labs(x = "L2 Norm Beta Hat Ratio", y = "CV Error Rate",
       title = expression(n[learn]~"="~n/2~~CV~LASSO~and~Ridge~Error~Rate)) +
  scale_color_manual(name = element_blank(), values = c("LASSO" = "red", "Ridge" = "blue")) +
  ylim(0,.6)
print(f2_1)
f2_2 = ggplot(data = cv.svm.error, aes(as.factor(cost), as.factor(gamma), fill = error)) +
  geom_tile()+
  labs(x = "cost", y = "gamma",
       title = expression(n[learn]~"="~n/2~~CV~SVM~Error~Rate),fill = "CV Error Rate")
f2_3 = ggplot() +
  geom_line(data = D9.cv.lasso.error.and.rate.coef, aes(x = X1, y = X2, color = "LASSO")) +
  geom_point(data = cv.lasso.error.and.rate.coef, aes(x = X1[which.min(X2)],
                                                      y = min(X2), color = "LASSO")) +
  geom_line(data = D9.cv.ridge.error.and.rate.coef, aes(x = X1, y = X2, color = "Ridge")) +
  geom_point(data = D9.cv.ridge.error.and.rate.coef, aes(x = X1[which.min(X2)],
                                                      y = min(X2), color = "Ridge")) +
  labs(x = "L2 Norm Beta Hat Ratio", y = "CV Error Rate",
       title = expression(n[learn]~"="~0.9~n~~CV~LASSO~and~Ridge~Error~Rate)) +
  scale_color_manual(name = element_blank(), values = c("LASSO" = "red", "Ridge" = "blue")) +
  ylim(0,.6)
f2_4 = ggplot(data = D9.cv.svm.error, aes(as.factor(cost), as.factor(gamma), fill = error)) +
  geom_tile()+
  labs(x = "cost", y = "gamma",
       title = expression(n[learn]~"="~0.9~n~~CV~SVM~Error~Rate),fill = "CV Error Rate")
f2 = grid.arrange(f2_1, f2_2, f2_3, f2_4, nrow = 2 , widths = c(1,1))

# time analysis
colMeans(D5.time.cv)
apply(D5.time.cv, 2, sd)
colMeans(D9.time.cv)
apply(D9.time.cv, 2, sd)

colMeans(D5.time.fit)
apply(D5.time.fit, 2, sd)
colMeans(D9.time.fit)
apply(D9.time.fit, 2, sd)

colMeans(D5.test.error)
apply(D5.test.error, 2, sd)
colMeans(D9.test.error)
apply(D9.test.error, 2, sd)

# variable importance
D5.l.coef = D5.l.coef[-1,-1]
D5.l.variable.importance = data.frame(t(abs(rowMeans(D5.l.coef))))
sort(D5.l.variable.importance, decreasing = T)[1:10]

D5.r.coef = D5.r.coef[-1,-1]
D5.r.variable.importance = data.frame(t(abs(rowMeans(D5.r.coef))))
sort(D5.r.variable.importance, decreasing = T)[1:10]

D5.rf.variable.importance = data.frame(t(read.csv("D5_rf.variable.importance.csv")[-1]))
sort(D5.rf.variable.importance, decreasing = T)[1:10]

D9.l.coef = D9.l.coef[-1,-1]
D9.l.variable.importance = data.frame(t(abs(rowMeans(D9.l.coef))))
sort(D9.l.variable.importance, decreasing = T)[1:10]

D9.r.coef = D9.r.coef[-1,-1]
D9.r.variable.importance = data.frame(t(abs(rowMeans(D9.r.coef))))
sort(D9.r.variable.importance, decreasing = T)[1:10]

D9.rf.variable.importance = data.frame(t(read.csv("D9_rf.variable.importance.csv")[-1]))
```

```r
sort(D9.rf.variable.importance, decreasing = T)[1:10]

# names of important variables
important.variables = rbind(names(sort(D5.l.variable.importance, decreasing = T)[1:10]),
names(sort(D5.r.variable.importance, decreasing = T)[1:10]),
names(sort(D5.rf.variable.importance, decreasing = T)[1:10]),
names(sort(D9.l.variable.importance, decreasing = T)[1:10]),
names(sort(D9.r.variable.importance, decreasing = T)[1:10]),
names(sort(D9.rf.variable.importance, decreasing = T)[1:10]))
write.csv(important.variables, file = "important.variables.csv")


f3_1 = ggplot(melt(D5.l.variable.importance), aes(x = variable, y = value, color = variable)) +
  geom_bar(stat="identity") + ylim(0,1) + theme(legend.position="none")+
  labs(x = element_blank(), y = "Absolute Value of Coefficients",
       title = expression(n[learn]~"="~n/2~~LASSO~Varible~Importance)) +
  theme(axis.text.x=element_blank(), axis.ticks.x=element_blank())
f3_2 = ggplot(melt(D5.r.variable.importance), aes(x = variable, y = value, color = variable)) +
  geom_bar(stat="identity") + ylim(0,1) + theme(legend.position="none")+
  labs(x = element_blank(), y = "Absolute Value of Coefficients",
       title = expression(n[learn]~"="~n/2~~Ridge~Varible~Importance)) +
  theme(axis.text.x=element_blank(), axis.ticks.x=element_blank())
f3_3 = ggplot(melt(D5.rf.variable.importance), aes(x = variable, y = value, color = variable)) +
  geom_bar(stat="identity") + ylim(0,20) + theme(legend.position="none")+
  labs(x = element_blank(), y = "Variable Importance",
       title = expression(n[learn]~"="~n/2~~RF~Varible~Importance)) +
  theme(axis.text.x=element_blank(), axis.ticks.x=element_blank())
f3_4 = ggplot(melt(D9.l.variable.importance), aes(x = variable, y = value, color = variable)) +
  geom_bar(stat="identity") + ylim(0,1) + theme(legend.position="none")+
  labs(x = element_blank(), y = "Absolute Value of Coefficients",
       title = expression(n[learn]~"="~0.9~n~~LASSO~Varible~Importance)) +
  theme(axis.text.x=element_blank(), axis.ticks.x=element_blank())
f3_5 = ggplot(melt(D9.r.variable.importance), aes(x = variable, y = value, color = variable)) +
  geom_bar(stat="identity") + ylim(0,1) + theme(legend.position="none")+
  labs(x = element_blank(), y = "Absolute Value of Coefficients",
       title = expression(n[learn]~"="~0.9~n~~Ridge~Varible~Importance)) +
  theme(axis.text.x=element_blank(), axis.ticks.x=element_blank())
f3_6 = ggplot(melt(D9.rf.variable.importance), aes(x = variable, y = value, color = variable)) +
  geom_bar(stat="identity") + ylim(0,20) + theme(legend.position="none")+
  labs(x = element_blank(), y = "Variable Importance",
       title = expression(n[learn]~"="~0.9~n~~RF~Varible~Importance)) +
  theme(axis.text.x=element_blank(), axis.ticks.x=element_blank())
f3 = grid.arrange(f3_1, f3_2, f3_3, f3_4, f3_5, f3_6, nrow = 2)

D5.importance = matrix(0,nrow = 400,ncol = 6)
D5.importance[,1:6] = cbind(t(D5.l.variable.importance)[,1], t(D5.r.variable.importance)[,1],
                      t(D5.rf.variable.importance)[,1],t(D9.l.variable.importance)[,1],
                      t(D9.r.variable.importance)[,1],
                      t(D9.rf.variable.importance)[,1])
colMeans(D5.importance)
D9.l.variable.importance = data.frame(t(abs(rowMeans(D9.l.coef))))
sort(D9.l.variable.importance, decreasing = T)[1:10]

D9.r.coef = D9.r.coef[-1,-1]
D9.r.variable.importance = data.frame(t(abs(rowMeans(D9.r.coef))))
sort(D9.r.variable.importance, decreasing = T)[1:10]

D9.rf.variable.importance = data.frame(t(read.csv("D9_rf.variable.importance.csv")[-1]))
sort(D9.rf.variable.importance, decreasing = T)[1:10]


x = rep(1:20,20)
y = c(rep(1,20),rep(2,20),rep(3,20),rep(4,20),rep(5,20),
      rep(6,20),rep(7,20),rep(8,20),rep(9,20),rep(10,20),
      rep(11,20),rep(12,20),rep(13,20),rep(14,20),rep(15,20),
      rep(16,20),rep(17,20),rep(18,20),rep(19,20),rep(20,20))

w = data.frame(cbind(x,y,D5.importance[,1:6]))
f5_1 = ggplot(w, aes(x = x, y = y, fill = w[,3])) +
  geom_tile() + labs(x = "X pixel", y = "Y pixel",
                  title = "n/2, Estimated Position of Nucleus by LASSO",fill = "V.Imp")
f5_2 = ggplot(w, aes(x = x, y = y, fill = w[,4])) +
  geom_tile() + labs(x = "X pixel", y = "Y pixel",
                     title = "n/2, Estimated Position of Nucleus by Ridge",fill = "V.Imp")
f5_3 = ggplot(w, aes(x = x, y = y, fill = w[,5])) +
  geom_tile() + labs(x = "X pixel", y = "Y pixel",
                     title = "n/2, Estimated Position of Nucleus by RF",fill = "V.Imp")
f5_4 = ggplot(w, aes(x = x, y = y, fill = w[,6])) +
  geom_tile() + labs(x = "X pixel", y = "Y pixel",
                     title = "0.9n, Estimated Position of Nucleus by LASSO",fill = "V.Imp")
f5_5 = ggplot(w, aes(x = x, y = y, fill = w[,7])) +
  geom_tile() + labs(x = "X pixel", y = "Y pixel",
                     title = "0.9n, Estimated Position of Nucleus by Ridge",fill = "V.Imp")
f5_6 = ggplot(w, aes(x = x, y = y, fill = w[,8])) +
  geom_tile() + labs(x = "X pixel", y = "Y pixel",
                     title = "0.9n, Estimated Position of Nucleus by RF",fill = "V.Imp")
```

```r
f5 = grid.arrange(f5_1, f5_2, f5_3, f5_4, f5_5, f5_6, nrow = 2)


# two test error rate difference t.test
test.error = cbind(D5.test.error, D9.test.error[,-1])
test.error.difference = matrix(0, nrow = 100, ncol = 5)
test.error.difference = test.error[,2:6] - test.error[,7:11]
t.test(test.error.difference$RF, mu = 0)
t.test(test.error.difference$R.SVM, mu = 0)
t.test(test.error.difference$Log, mu = 0)
t.test(test.error.difference$Log.LASSO, mu = 0)
t.test(test.error.difference$Log.Ridge, mu = 0)

```