Derivation and validation of machine learning approaches to predict acute kidney injury after cardiac surgery

: Supplemental materials

Hyung-Chul Lee, Hyun-Kyu Yoon, Karam Nam, Youn Joung Cho, Tae Kyong Kim, Won Ho Kim, Jae-Hyon Bahk

Department of Anesthesiology and Pain Medicine, Seoul National University Hospital, Seoul National University College of Medicine, Seoul, Republic of Korea

List of supplemental materials

<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Supplemental Figure S1.</strong> Variable importance plot using random forest model.</td>
<td>2</td>
</tr>
<tr>
<td><strong>Supplemental Figure S2.</strong> Gradient boosting tree plot showing the matrix of classification.</td>
<td>3</td>
</tr>
<tr>
<td><strong>Supplemental Figure S3.</strong> Support vector machine classification plot.</td>
<td>4</td>
</tr>
<tr>
<td><strong>Supplemental Table S1.</strong> KDIGO (Kidney Disease Improving Global Outcomes) serum creatinine diagnostic criteria of acute kidney injury.</td>
<td>5</td>
</tr>
<tr>
<td><strong>Supplemental Table S2.</strong> Comparison of area under receiver-operating characteristic curve among the different models for predicting stage 2 or 3 acute kidney injury.</td>
<td>6</td>
</tr>
<tr>
<td><strong>Supplemental Table S3.</strong> Results of multivariable logistic regression analysis for acute kidney injury without stepwise variable selection.</td>
<td>7</td>
</tr>
<tr>
<td><strong>Supplemental Text S1.</strong> R source code to perform machine learning techniques.</td>
<td>9</td>
</tr>
</tbody>
</table>
Supplemental Figure S1. Variable importance plot using random forest model. The abbreviations were the same as the legends of Figure 3.

rf_hyper
**Supplemental Figure S2.** Gradient boosting tree plot showing the matrix of classification. Extreme gradient boosting builds a sequential series of shallow trees.
Supplemental Figure S3. Support vector machine classification plot. This figure shows a simple two-dimensional visual illustration of support vector machine classification to help understand the process of classification by support vector machine. Each triangle and circle means binomial classification of acute kidney injury or not. The open circle or triangle means correct classification and closed circle or triangle means incorrect classification. This figure was drawn by Kernlab package of software R.
**Supplemental Table S1.** KDIGO (Kidney Disease Improving Global Outcomes) serum creatinine diagnostic criteria of acute kidney injury.

<table>
<thead>
<tr>
<th>KDIGO criteria</th>
<th>Serum creatinine criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stage 1</td>
<td>Increase in sCr by 0.3 mg/dl or increase in sCr to 1.5-1.9 times baseline within postoperative 7 days</td>
</tr>
<tr>
<td>Stage 2</td>
<td>Increase in sCr to 2.0-2.9 times baseline within postoperative 7 days</td>
</tr>
<tr>
<td>Stage 3</td>
<td>Increase in sCr to &gt; 4.0 mg/dl or increase in sCr to 3.0 times baseline within postoperative 7 days</td>
</tr>
</tbody>
</table>

sCr = serum creatinine.
**Supplemental Table S2.** Comparison of area under receiver-operating characteristic curve among the different models for predicting stage 2 or 3 acute kidney injury.

<table>
<thead>
<tr>
<th>Model</th>
<th>Software or R Packages</th>
<th>Error rate of test data set</th>
<th>AUC in the test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine learning techniques</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision tree, CART</td>
<td>tree, rpart</td>
<td>14.3%</td>
<td>0.53 (0.50 – 0.56)</td>
</tr>
<tr>
<td>ROSE decision tree</td>
<td>ROSE</td>
<td>9.3%</td>
<td>0.65 (0.61 – 0.69)</td>
</tr>
<tr>
<td>Random forest model</td>
<td>randomForest</td>
<td>15.2%</td>
<td>0.53 (0.50 – 0.56)</td>
</tr>
<tr>
<td>Random forest SMOTE model</td>
<td>DMwR</td>
<td>13.8%</td>
<td>0.55 (0.51 – 0.59)</td>
</tr>
<tr>
<td>Gradient boosting</td>
<td>xgboost</td>
<td>8.5%</td>
<td>0.74 (0.70 – 0.79)*</td>
</tr>
<tr>
<td>Support vector machine, classifier</td>
<td>e1071</td>
<td>18.4%</td>
<td>0.53 (0.50 – 0.57)</td>
</tr>
<tr>
<td>Support vector machine, SMOTE model</td>
<td>UBL</td>
<td>12.4%</td>
<td>0.57 (0.53 – 0.61)</td>
</tr>
<tr>
<td>Support vector machine, least square</td>
<td>Kernlab</td>
<td>13.0%</td>
<td>0.54 (0.51 – 0.57)</td>
</tr>
<tr>
<td>Neural network classifier</td>
<td>nnet</td>
<td>9.9%</td>
<td>0.63 (0.59 – 0.67)</td>
</tr>
<tr>
<td>Neural network classifier</td>
<td>neuralnet</td>
<td>13.5%</td>
<td>0.54 (0.51 – 0.58)</td>
</tr>
<tr>
<td>Deep belief network</td>
<td>h2o</td>
<td>11.5%</td>
<td>0.57 (0.53 – 0.61)</td>
</tr>
</tbody>
</table>

Error rate was defined as sum of the number of cases with false positive and false negative divided by all test set.

* Significantly greater than AUC of all the other techniques.
**Supplemental Table S3.** Development of multivariable logistic regression model to predict acute kidney injury using stepwise variable selection.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Beta-coefficient</th>
<th>Odds Ratio</th>
<th>95% CI</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age per 10 year older</td>
<td>0.13</td>
<td>1.14</td>
<td>1.04 – 1.61</td>
<td>0.004</td>
</tr>
<tr>
<td>Female</td>
<td>-0.24</td>
<td>0.79</td>
<td>0.55 – 1.13</td>
<td>0.201</td>
</tr>
<tr>
<td>Body-mass index, kg/m²</td>
<td>0.01</td>
<td>1.01</td>
<td>0.96 – 1.06</td>
<td>0.758</td>
</tr>
<tr>
<td>History of hypertension</td>
<td>0.32</td>
<td>1.38</td>
<td>1.12 – 1.69</td>
<td>0.002</td>
</tr>
<tr>
<td>History of diabetes mellitus</td>
<td>0.28</td>
<td>1.32</td>
<td>0.91 – 1.92</td>
<td>0.139</td>
</tr>
<tr>
<td>Baseline chronic kidney disease</td>
<td>0.91</td>
<td>2.48</td>
<td>1.62 – 3.78</td>
<td>&lt;0.001</td>
</tr>
<tr>
<td>History of three vessel disease</td>
<td>-0.20</td>
<td>0.97</td>
<td>0.90 – 1.15</td>
<td>0.204</td>
</tr>
<tr>
<td>History of cerebrovascular accident</td>
<td>0.22</td>
<td>1.25</td>
<td>0.78 – 2.00</td>
<td>0.348</td>
</tr>
<tr>
<td>History of previous cardiac disease</td>
<td>0.22</td>
<td>1.25</td>
<td>0.70 – 2.24</td>
<td>0.452</td>
</tr>
<tr>
<td>History of previous coronary stent</td>
<td>-0.50</td>
<td>0.91</td>
<td>0.97 – 1.03</td>
<td>0.159</td>
</tr>
<tr>
<td>History of chronic obstructive pulmonary disease</td>
<td>-0.32</td>
<td>0.73</td>
<td>0.37 – 1.42</td>
<td>0.350</td>
</tr>
<tr>
<td>Preoperative E/e’ &gt; 15</td>
<td>0.45</td>
<td>1.58</td>
<td>1.27 – 1.96</td>
<td>&lt;0.001</td>
</tr>
<tr>
<td>Preoperative LVEF, %</td>
<td>0.00</td>
<td>1.00</td>
<td>0.99 – 1.02</td>
<td>0.783</td>
</tr>
<tr>
<td>Preoperative hematocrit, %</td>
<td>-0.06</td>
<td>0.94</td>
<td>0.92 – 0.96</td>
<td>&lt;0.001</td>
</tr>
<tr>
<td>Preoperative creatinine, mg/dL</td>
<td>0.06</td>
<td>1.06</td>
<td>0.91 – 1.24</td>
<td>0.457</td>
</tr>
<tr>
<td>Preoperative uric acid, mg/dL</td>
<td>0.09</td>
<td>1.10</td>
<td>1.00 – 1.21</td>
<td>0.047</td>
</tr>
<tr>
<td>Preoperative albumin, g/dL</td>
<td>-0.30</td>
<td>0.74</td>
<td>0.53 – 1.04</td>
<td>0.086</td>
</tr>
<tr>
<td>Preoperative blood glucose, mg/dL</td>
<td>0.00</td>
<td>1.00</td>
<td>0.99 – 1.00</td>
<td>0.535</td>
</tr>
<tr>
<td>Preoperative beta-blocker</td>
<td>-0.12</td>
<td>0.89</td>
<td>0.61 – 1.29</td>
<td>0.535</td>
</tr>
<tr>
<td>Preoperative calcium channel blocker</td>
<td>0.21</td>
<td>1.23</td>
<td>0.81 – 1.87</td>
<td>0.334</td>
</tr>
<tr>
<td>Preoperative angiotensin receptor blocker</td>
<td>-0.02</td>
<td>0.98</td>
<td>0.64 – 1.52</td>
<td>0.937</td>
</tr>
<tr>
<td>Preoperative statin</td>
<td>0.18</td>
<td>1.20</td>
<td>0.84 – 1.71</td>
<td>0.320</td>
</tr>
<tr>
<td>Preoperative diuretics</td>
<td>0.26</td>
<td>1.30</td>
<td>0.82 – 2.06</td>
<td>0.272</td>
</tr>
<tr>
<td>Preoperative pulmonary hypertension</td>
<td>-0.17</td>
<td>0.84</td>
<td>0.45 – 1.56</td>
<td>0.580</td>
</tr>
<tr>
<td>Surgery time, per 1 hour</td>
<td>0.07</td>
<td>1.08</td>
<td>1.01 – 1.15</td>
<td>0.036</td>
</tr>
<tr>
<td>Intraoperative crystalloid</td>
<td>0.38</td>
<td>1.11</td>
<td>1.05 – 1.18</td>
<td>0.047</td>
</tr>
<tr>
<td>Intraoperative red blood cell transfusion, unit</td>
<td>0.06</td>
<td>1.06</td>
<td>1.01 – 1.11</td>
<td>0.022</td>
</tr>
</tbody>
</table>
Intraoperative fresh frozen plasma transfusion, unit

<table>
<thead>
<tr>
<th></th>
<th>0.09</th>
<th>1.09</th>
<th>1.03 – 1.15</th>
<th>0.001</th>
</tr>
</thead>
</table>

Intraoperative diuretics use

<table>
<thead>
<tr>
<th></th>
<th>0.63</th>
<th>1.88</th>
<th>1.36 – 2.60</th>
<th>&lt;0.001</th>
</tr>
</thead>
</table>

Intraoperative mean cardiac index, per 0.5 L/min/m\(^2\) decrease

<table>
<thead>
<tr>
<th></th>
<th>0.22</th>
<th>1.05</th>
<th>1.01 – 1.09</th>
<th>0.045</th>
</tr>
</thead>
</table>

Intraoperative mean arterial pressure, per 10 mmHg decrease

<table>
<thead>
<tr>
<th></th>
<th>0.19</th>
<th>1.03</th>
<th>1.00 – 1.04</th>
<th>0.048</th>
</tr>
</thead>
</table>

Intraoperative mean SvO\(_2\), per 5 % decrease

<table>
<thead>
<tr>
<th></th>
<th>0.24</th>
<th>1.06</th>
<th>1.02 – 1.10</th>
<th>0.044</th>
</tr>
</thead>
</table>

Multivariable logistic regression analysis was performed using all the variables in Table 1. Neither univariable screening nor stepwise variable selection process was used. Nagelkerke’s R\(^2\) was 0.38 and Hosmer-Lemeshow goodness-of-fit test showed good calibration (chi-square = 11.2, P=0.193).

CI = confidence interval, E/e’ = ratio of early transmitral flow velocity to early diastolic velocity of the mitral annulus.
**Supplemental Text S1.** R source code to perform machine learning techniques.

```r
#------- Data loading -------#
#---- This applies to all other packages----#
aki <- read.csv("dataset.csv", header=TRUE)
n <- dim(aki)[1]
p <- dim(aki)[2]
n1 <- table(aki$AKI)[1]
n2 <- table(aki$AKI)[2]

#---- Making training and test data sets ----#
#---- This applies to all other packages----#
set.seed(20180202)
head(aki)
ind1 <- sample(which(aki$AKI==0), round(n1/2), replace=F)
ind2 <- sample(which(aki$AKI==1), round(n2/2), replace=F)
NROW(ind1)
NROW(ind2)
tr_ind <- c(ind1,ind2)
NROW(tr_ind)
aki_tr <- aki[tr_ind,]
aki_te <- aki[-tr_ind,]

#--------- Decision Tree Classification ---------#
#---------------------------------------------#
install.packages("tree")
library(tree)
aki_tree = tree(factor(AKI) ~., aki_tr)
summary(aki_tree)
aki_tree

#--------- Model selection for decision tree model ---------#
cv_aki.tree <- cv.tree(aki_tree)
```
plot(cv_aki.tree$size, cv_aki.tree$dev, type='b')

#------------------- Pruning for decision tree model -------------------#
dt_prun <- prune.tree(aki_tree, best=8)
plot(dt_prun)
text(dt_prun, pretty=0)

dt_prun2 = prune.misclass(aki_tree, best=8)
plot(dt_prun2)
text(dt_prun2, all = T)

dt_yht <- predict(dt_prun, newdata=aki_te[, -1], type="class")
y <- aki_te$AKI
mean(y != dt_yht)

install.packages("roc.curve")
library(roc.curve)
roc.curve(aki_te$AKI, dt_yht, plotit = F)

#------------------- Decision Tree Classification -------------------#
#------------------- under balance case based on ROSE -------------------#

install.packages("ROSE")
library(ROSE)
aki_tr_ROSE <- ovun.sample(factor(AKI) ~ ., aki_tr, method="both", seed=1, N=1500)$data

table(aki_tr_ROSE$AKI)

#------------------- Training decision tree model -------------------#
aki_tree_ROSE = tree(factor(AKI) ~ ., aki_tr_ROSE)
summary(aki_tree_ROSE)
aki_tree_ROSE

#------------------- Model selection for decision tree model -------------------#
cv_aki.tree_ROSE <- cv.tree(aki_tree_ROSE)
plot(cv_aki.tree_ROSE$size, cv_aki.tree_ROSE$dev, type="b")

#---------- Pruning for decision tree model ----------#
dt_prun_ROSE <- prune.tree(aki_tree_ROSE,best=25)
plot(dt_prun_ROSE)
text(dt_prun_ROSE,pretty=0)

dt_yht_ROSE <- predict(dt_prun_ROSE, newdata=aki_te[,1],type="class")
y <- aki_te$AKI
mean(y != dt_yht_ROSE)
roc.curve(aki_te$AKI, dt_yht_ROSE, plotit = F)

#---------- Decision Tree Classification ----------#
#---------- rpart package ----------#
install.packages("rpart")
install.packages("rpart.plot")
library(rpart)
library(rpart.plot)
set.seed(21080202)
AKI_rpart = rpart(AKI ~ ., data = aki_tr, method = "class")
plotcp(AKI_rpart)
min_cp = AKI_rpart$cptable[which.min(AKI_rpart$ cptable[,"xerror"]),"CP"]
AKI_rpart_prune = prune (AKI_rpart, cp = min_cp)
prp(AKI_rpart_prune)
prp(AKI_rpart_prune, type = 1)
rpart.plot(AKI_rpart_prune)
prp(AKI_rpart_prune2)
prp(AKI_rpart_prune2, type = 1)
rpart.plot(AKI_rpart_prune2)

# error rate and AUC in test set
dt_rpart_prune <- predict(AKI_rpart_prune, newdata=aki_te[,1],type="class")
y <- aki_te$AKI
mean(y != dt_rpart_prune)
library(ROSE)
roc.curve(aki_te$AKI, dt_rpart_prune, plotit = F)

#----------------- Random Forest Regression -----------------
#-----------------------------------------------------------
install.packages("randomForest",repos="http://cran.nexr.com")
library(randomForest)

#---- Training randomForest model ----#
hyper_tr$AKI <- as.factor(hyper_tr$AKI)
rf_hyper = randomForest(AKI~., data=hyper_tr, mtry=round(sqrt(p-1)), importance=TRUE)
rf_hyper

#------- Importance plot --------#
importance(rf_hyper)
varImpPlot(rf_hyper)

#-------- prediction --------#
rf_yht <- predict(rf_hyper, newdata=hyper_te[-1], type="class")
dim(rf_yht)

#-------- Computing error rate --------#
y <- hyper_te$AKI
dim(y)
mean(y != rf_yht)

#-------- AUC --------#
install.packages("pROC")
library(pROC)
auc <- roc(y, as.numeric(rf_yht))
print(auc)
plot(auc, ylim=c(0,1), print.thres=TRUE, main=paste('AUC:',round(auc$auc[[1]],2)))
abline(h=1,col='blue',lwd=2)
abline(h=0,col='red',lwd=2)

#-------- SMOTE more positive instances ---------#
install.packages("DMwR")
library(DMwR)
hyper_tr$AKI <- as.factor(hyper_tr$AKI)
hyper_tr_S <- SMOTE(AKI ~ ., hyper_tr, perc.over = 1980, perc.under=100)
hyper_tr_S$AKI <- as.numeric(hyper_tr_S$AKI)
hyper_tr_S$AKI <- ifelse(hyper_tr_S$AKI==1,0,1)
table(hyper_tr_S$AKI)

#----- evaluate the SMOTE performance -----#
hyper_tr_S$AKI <- as.factor(hyper_tr_S$AKI)
rf_hyper_S = randomForest(AKI~.,data=hyper_tr_S,mtry=round(sqrt(p-1)),importance=TRUE)
rf_hyper_S

#------- prediction -------#
rf_yht_S <- predict(rf_hyper_S, newdata=hyper_te[,-1],type="class")
y <- hyper_te$AKI
mean(y != rf_yht_S)
auc_S <- roc(y, as.numeric(rf_yht_S))
print(auc_S)

plot(auc_S, ylim=c(0,1), print.thres=TRUE, main=paste('AUC:',round(auc_S$auc[[1]],2)))
abline(h=1,col='blue',lwd=2)
abline(h=0,col='red',lwd=2)
install.packages("xgboost")
library(xgboost)

#----- Making training and test data sets ----#
set.seed(20180202)
n1 <- table(hyper$AKI)[1] # 3012
n2 <- table(hyper$AKI)[2] # 151
ind1 <- sample(which(hyper$AKI==0), round(n1/2), replace=F)
ind2 <- sample(which(hyper$AKI==1), round(n2/2), replace=F)
NROW(ind1)
NROW(ind2)
tr_ind <- c(ind1, ind2)
NROW(tr_ind)
hyper_tr <- hyper[tr_ind,]
hyper_te <- hyper[-tr_ind,]
prop.table(table(hyper_tr$AKI))
prop.table(table(hyper_te$AKI))

#----- Training XGBoost model ----#
# training data
label_tr = as.vector(hyper_tr$AKI)
data_tr = as.matrix(hyper_tr[, -1])
# test data
label_te = as.vector(hyper_te$AKI)
data_te = as.matrix(hyper_te[, -1])

# Train model
bst <- xgboost(data = data_tr, label = label_tr, max_depth = 6, eta = 0.3,
nrounds = 2, objective = "binary:logistic")
bst1 = xgboost(data = data_tr, label = label_tr, max_depth = 6, eta = 0.3,
nrounds = 2, objective = "binary:logistic", eval_metric = "auc")
pred_te <- predict(bst, data_te)
pred_tr <- predict(bst, data_tr)

library(pROC)
roc_tr <- roc(label_tr, pred_tr)
roc_te <- roc(label_te, pred_te)
thres = coords(roc_tr, "best", ret = "threshold")
plot.roc(roc_tr,
    col="red",
    print.auc=TRUE,
    max.auc.polygon=TRUE,
    print.thres=TRUE,
    auc.polygon=TRUE)
plot.roc(roc_te,
    col="red",
    print.auc=TRUE,
    max.auc.polygon=TRUE,
    print.thres=TRUE,
    auc.polygon=TRUE)

print(paste("Threshold: ", thres))
pred_tr <- ifelse (pred_tr > thres, 1, 0)
print(paste("Train Error: ", mean(pred_tr != label_tr)))
pred_te <- ifelse (pred_te > thres, 1, 0)
print(paste("Test Error: ", mean(pred_te != label_te)))
table(label_te, prediction)

#----- View feature importance/influence from the learnt model ------#
importance_matrix <- xgb.importance(model = bst)
print(importance_matrix)
xgb.plot.importance(importance_matrix = importance_matrix)

#----- View the trees from your model ----#
xgb.dump(bst, with_stats = T)

#----- Plot the trees from your model -----#
install.packages("DiagrammeR")
library(DiagrammeR)
xgb.plot.tree(model = bst)

#----- Cross validation --------#
dtrain <- xgb.DMatrix(data_tr,label=label_tr)
cv_res = xgb.cv(data = dtrain, nfold = 5, nround = 2, nthread = 2,
                max_depth = 3, eta = 1, objective = "binary:logistic", eval_metric = "auc")

#-------- Support Vector Machine Regression --------#
#-----------------------------------------------#
install.packages("e1071")
library(e1071)

#------- Tuning the parameters of SVM -------#
svm_tune <- tune(svm, factor(AKI)~., data=aki_tr, ranges = list(cost =c(1,10),
gamma=c(0.1,0.5)))
summary(svm_tune)

svm_tune$best.parameters
svm_after_tune <- svm(factor(AKI)~.,data=aki_tr, scale=F, gamma=0.1, cost=1)
#default: kernel="radial", gamma=1/n, cost=10, epsilon=0.1
summary(svm_after_tune)
svm_after_tune$fitted

y_pred <- predict(svm_after_tune, aki_te[,-1])
mean(aki_te[,1]!= y_pred)

table(aki_te[,1], y_pred)
1- \[ \frac{\text{sum(diag(table(aki_te[,1], y_pred))}}{(\text{dim (aki_te)[1]})} \]

install.packages("pROC")
library(pROC)
auc <- roc(aki_te$AKI, as.numeric(y_pred))
print(auc)

plot(auc, ylim=c(0,1), print.thres=TRUE, main=paste('AUC:',round(auc$auc[[1]],2)))
abline(h=1,col='blue',lwd=2)
abline(h=0,col='red',lwd=2)

#------ Apply SMOTE to aki_tr data ------#
install.packages("UBL")
library(UBL)

aki_tr$AKI <- as.factor(aki_tr$AKI)
aki_tr_s <- SmoteClassif(AKI ~ ., aki_tr, C.perc = "balance")
aki_tr_s <- SmoteClassif(AKI ~ ., aki_tr, C.perc = "balance", dist="HOEM")
aki_tr_s <- SmoteClassif(AKI ~ ., aki_tr, C.perc = "balance", dist="HVDM")
aki_tr_s
aki_tr_s$AKI <- as.numeric(aki_tr_s$AKI)
table(aki_tr_s$AKI)
aki_tr_s$AKI <- ifelse(aki_tr_s$AKI==1,0,1)
table(aki_tr_s$AKI)
prop.table(table(aki_tr_s$AKI))

#------ Tuning the parameters of SVM based on balanced data after SMOTE ------#
svm_tune_s <- tune(svm, factor(AKI)~., data=aki_tr_s, ranges = list(cost =c(1,10),
gamma=c(0.1,0.5)))
summary(svm_tune_s)
svm_tune_s$best.parameters
svm_after_tune_s <- svm(factor(AKI)~.,data=aki_tr_s, scale=F, gamma=0.1, cost=10)
summary(svm_after_tune_s)
svm_after_tune_s$fitted

# Apply to test set
y_pred_s <- predict(svm_after_tune_s, aki_te[,1])
mean(aki_te[,1] != y_pred_s)
table(aki_te[,1], y_pred_s)
1 - sum(diag(table(aki_te[,1], y_pred_s)) / (dim(aki_te)[1]))

install.packages("pROC")
library(pROC)
auc_s <- roc(aki_te$AKI, as.numeric(y_pred_s))
print(auc_s)

plot(auc_s, ylim = c(0, 1), print.thres = TRUE, main = paste('AUC:', round(auc_s$auc[1], 2)))
abline(h = 1, col = 'blue', lwd = 2)
abline(h = 0, col = 'red', lwd = 2)

#-- Least Squares Support Vector Machine Classification (kernlab)--#
#----------------------------------------------------------------------------------#
install.packages("kernlab")
library(kernlab)

#---- Making traing and test data sets ----#
set.seed(20180202)
n1 <- table(aki$AKI)[1]
n2 <- table(aki$AKI)[2]
ind1 <- sample(which(aki$AKI == 0), round(n1/2), replace = F)
ind2 <- sample(which(aki$AKI == 1), round(n2/2), replace = F)
NROW(ind1)
NROW(ind2)
tr_ind <- c(ind1, ind2)
NROW(tr_ind)
aki_tr <- aki[tr_ind,]
aki_te <- aki[-tr_ind,]
prop.table(table(aki_tr$AKI))
prop.table(table(aki_te$AKI))

#------- Training LS-SVM model -------#
lssvm <- lssvm(factor(AKI)~., scaled=F, reduced = F, cross=0,kernel = "rbfdot",
kpar=list(sigma=0.1), tau=0.1, data=aki_tr)
summary(lssvm)
fitted(lssvm); alpha(lssvm); b(lssvm)
ypred_lss <- predict(lssvm, aki_te[,1])
mean(aki_te[,1]!= ypred_lss)

table(aki_te[,1], ypred_lss)
1- sum(diag(table(aki_te[,1], ypred_lss)))/(dim(aki_te)[1])

install.packages("pROC")
library(pROC)
auc_s <- roc(aki_te$AKI, as.numeric(y_pred_lss))
print(auc_s)

#------- Neural Network Regression ---------------#
#-----------------------------------------------#
install.packages("nnet")
library(nnet)

#------- Tuning neural network (nnet): determing the number of hidden nodes for test data ----#
#
err<-c()
for(h_size in seq(6, 8, by = 1))
{ nn <- nnet(factor(AKI)~., data=hyper_tr, size = h_size, decay = 0, trace=F,linout=F)
  py <- predict(nn, hyper_te[,1], type = "class")
  ty <- hyper_te$AKI}
err[h_size] <- mean(ty != py)}
best_size <- which.min(err)

#-------- Predicting neural network (nnet) for test data after tuning --------#
nn_model <- nnet(factor(AKI)~., data=hyper_tr, size = best_size, decay = 0, trace=F, linout=F)
ypred <- predict(nnn_model, hyper_te[-1], type = "class")
mean(hyper_te$AKI != ypred)
table(hyper_te$AKI, ypred)
1 - sum(diag(table(hyper_te$AKI, ypred)))/(dim(hyper_te)[1])

#-------- Neural Network Regression --------------------------#
#-------------------------------------------------------------#
install.packages("neuralnet")
library(neuralnet)

#-- Tuning neural network (neuralnet): determining the number of hidden nodes for test data ---#
err_cv <- c()
for(h_size in seq(6, 8, by = 1))
{neural_cv <- neuralnet(hyper_tr$AKI~
ypr_cv <- compute(neural_cv, hyper_te[-1])
ypred_cv <- ifelse(ypr_cv$net.result > 0.5, 1, 0)
err_cv[h_size] <- mean(hyper_te$AKI != ypred_cv)
}
best_size <- which.min(err_cv)

#------- Prediction based neural network (neuralnet) after tuning -------#
model_form = as.formula(paste("hyper_tr$AKI~ ",paste(names(hyper_tr[-1]),collapse="+")))
neural_f <- neuralnet(hyper_tr$AKI~ Emergency+Surgery_typeCABG+Surgery_typeAorta+Surgery_typeValve+age_per_10yr+gender+BMI+HTN+DM+COPD_asthma+CVA+CKD+Pul_HTN+three_VD+Previous_PCI_stent+Previous_cardiac_surg+Preop_EF+Preop_E_of_e_prime_morethan_15+Preop_Hct+Preop_Cr+Preop_Albumin+preop_uric_acid+Preop_Glucose+Preop_ARB+Preop_ACEi+Preop_CCB+Preop_Beta_blocker+Preop_statin+preop_diuretics+Anesth_technique_TIVA_vs_inhalational+Surgery_time_hour+Anesth_time_hour+mean_art_pressure+mean_SvO2+mean_cardiac_index+Total_Intraop_Fluid_per_1L+Intraop_Colloid_amount_per_500ml+Intraop_pRB_C+Intraop_FFP+Intraop_diuretics, data=hyper_tr[-1], hidden = 6, algorithm = "rprop+", err.fct="ce",linear.output=FALSE)

ypr <- compute(neural_f, hyper_te[-1])
ypred <- ifelse(ypr$net.result > 0.5, 1, 0)

mean(hyper_te$AKI != ypred)
table(hyper_te$AKI, ypred)

#-------- Deep Belief Network based Classification ---------#
#--------------------------------------------------#
install.packages("h2o")
library(h2o)
h2o.init()

#-------- Data loading -----#
hyper <- read.csv('dataset.csv', header=T)
head(hyper,3)

n <- dim(hyper)[1]
p <- dim(hyper)[2]
print(table(hyper$AKI))
print(prop.table(table(hyper$AKI)))

#--- 10-fold cross validation method without pretraining ----#

h1 <- 100; h2 <- 100; h3 <- 100;

dnn_cv <- h2o.deeplearning(x=2:26, y=1, activation="Tanh", #activation="TanhWithDropout",
    #initial_weight_distribution = "Normal",
    mini_batch_size=6, epoch=100,
    hidden=c(h1, h2, h3), training_frame=hyper_tr,
    nfolds=10, balance_classes = FALSE)

acc <- dnn_cv@model$cross_validation_metrics_summary[1,1]

#--------- Training DNN model ---------------#

#----- Pretraining using autoencoder -----#

ae_pretr <- h2o.deeplearning(x = 2:26, training_frame = hyper_tr,
    model_id = "cc_autoencoder", ignore_const_cols = FALSE,
    activation = "Tanh",
    hidden = c(h1, h2, h3), autoencoder = TRUE)

#----- Fine tuning using BP -----#

dnn_fit1 <- h2o.deeplearning(x = 2:26, y = 1, training_frame = hyper_tr,
    ignore_const_cols = FALSE, hidden = c(h1, h2, h3),
    pretrained_autoencoder = "cc_autoencoder")

#--------- Prediction based on DNN model ---------#

h2o.performance(dnn_fit1, hyper_te)

h2o.confusionMatrix(dnn_fit1, hyper_te)

#--------- Training DNN model using BP without pretraining -------#

dnn_fit2 <- h2o.deeplearning(x = 2:26, y = 1, training_frame = hyper_tr,
    ignore_const_cols = FALSE, hidden = c(h1, h2, h3))
h2o.performance(dnn_fit2, hyper_te)
h2o.confusionMatrix(dnn_fit2, hyper_te)