library(corrplot)

library(glmnet)

library(caret)

library(CBCgrps)

library(nortest)

library(tidyverse)

library(ggpubr)

library(rms)

library(pROC)

library(viridis)

library(dplyr)

library(tidyr)

library(car)

set.seed(3464)

trainIndex <- createDataPartition(data$diagnosis, p = .7, list = FALSE)

train <- data[trainIndex, ]

test <- data[-trainIndex, ]

write.csv(train, file = "train.csv", row.names = FALSE)

write.csv(test, file = "test.csv", row.names = FALSE)

min\_max\_scale = function(x){

(x-min(x))/(max(x)-min(x))

}

data2 = train%>%

mutate\_if(.predicate = is.numeric,

.funs = min\_max\_scale)%>%

as.data.frame()

set.seed(123) #random number generator

x <- data.matrix(data2[, -1])

y <- data2[, 1]

y<-as.numeric(unlist(y))

lasso <- glmnet(x, y, family = "binomial", nlambda = 1000, alpha = 1)

plot(lasso, xvar = "lambda", label = TRUE)

print(lasso)

plot(lasso, xvar = "lambda", label = FALSE)

lasso.cv = cv.glmnet(x, y,alpha = 1,nfolds =20,family="binomial")

plot(lasso.cv)

lasso.cv$lambda.min #minimum

lasso.cv$lambda.1se #one standard error away

coef(lasso.cv, s = "lambda.1se")

target\_n <- 5

nvar <- lasso.cv$nzero - 1

idx\_tar <- which.min(abs(nvar - target\_n))

lambda\_tar <- lasso.cv$lambda[idx\_tar]

cat(sprintf("λ ≈ %g， %d （%d）\n",

lambda\_tar, nvar[idx\_tar], target\_n))

coef\_mat <- as.matrix(coef(lasso.cv, s = lambda\_tar))

vars <- rownames(coef\_mat)[coef\_mat[,1] != 0]

vars <- setdiff(vars, "(Intercept)")

print(vars)

lassov <- c("diagnosis","sapsii","admissionage","sodium","mbp" )

vif\_fit <- glm(diagnosis ~ sapsii + admissionage + sodium + mbp,

data = train, family = binomial())

vif\_vals <- vif(vif\_fit)

print(round(vif\_vals, 2))

dat.train <- train[,lassov]

dat.test <- test[,lassov]

dat.train = dat.train%>%

mutate\_if(.predicate = is.numeric,

.funs = min\_max\_scale)%>%

as.data.frame()

dat.test = dat.test%>%

mutate\_if(.predicate = is.numeric,

.funs = min\_max\_scale)%>%

as.data.frame()

write.csv(dat.train,file="dat.train.csv")

write.csv(dat.test,file="dat.test.csv")

library(rms)

mydata<-train

attach(mydata)

dd <- datadist(mydata)

options(datadist = 'dd')

fit0 <- lrm(diagnosis ~ sapsii +admissionage +sodium +mbp , data = mydata, x = T, y = T)

fit0

nom0 <- nomogram(fit0,fun = plogis,fun.at = c(.001,.01,.05,.5,.95,.99,.999),

lp = T,funlabel = "diagnosis rate")

plot(nom0)

library(regplot)

regplot(reg = fit0,

plots = c("density", "boxes"),

center = TRUE,

observation = mydata[1,],

title = "",

points = TRUE,

droplines = TRUE,

clickable = FALSE,

showP = TRUE,

rank = "sd",

failtime = c(1, 2, 3),

prfail = FALSE)

regplot(reg = fit0,

plots = c("density", "boxes"),

center = TRUE,

observation = mydata[1,],

title = "Diagnosis Probability",

points = TRUE,

droplines = TRUE,

clickable = FALSE,

showP = TRUE,

rank = "sd",

boxcol = "#808080",

dencol = "#4F94CD",

obscol = "red",

spkcol = "#696969",

cexscales = 0.6,

cexvars = 0.7,

cexcats = 0.7,

leftlabel = FALSE)

gd<-predict(fit0,newdata = train,

se.fit = FALSE,dispersion = NULL,trems= NULL,

na.action = na.pass)

library(pROC)

roc.list<-roc(train$diagnosis,gd)

roc.list

g.list<- ggroc(roc.list,alpha = 1,size = 0.8,legacy.axes = TRUE,color="red")

g.list+theme\_classic()+ggtitle("data")+annotate(geom="segment",x = 0,y = 0,xend =1,yend = 1)

library(pROC)

library(ggplot2)

roc.list <- roc(train$diagnosis, gd)

g.list <- ggroc(roc.list, alpha = 1, size = 1, legacy.axes = TRUE, color = "#377EB8")

g.list <- g.list +

theme\_minimal(base\_size = 14) +

ggtitle("ROC Curve of Diagnostic Test") +

theme(plot.title = element\_text(hjust = 0.5, face = "bold", color = "#377EB8"),

axis.title = element\_text(face = "bold", color = "#4D4D4D"),

axis.text = element\_text(color = "#4D4D4D")) +

geom\_segment(aes(x = 0, y = 0, xend = 1, yend = 1), linetype = "dashed", color = "#E41A1C") +

annotate("text", x = 0.2, y = 0.9, label = sprintf("AUC = %.3f", auc(roc.list)), size = 5, color = "#4D4D4D")

print(g.list)

print(best\_coords)

gd <- predict(fit0, newdata = train, type = "fitted")

roc.list <- roc(train$diagnosis, gd)

best\_coords <- coords(roc.list,

x = "best",

best.method = "youden",

ret = c("threshold", "sensitivity", "specificity", "accuracy", "ppv", "npv"))

print(round(best\_coords, 3))

gd<-predict(fit0,newdata = test,

se.fit = FALSE,dispersion = NULL,trems= NULL,

na.action = na.pass)

library(pROC)

roc.list<-roc(test$diagnosis,gd)

roc.list

g.list<- ggroc(roc.list,alpha = 1,size = 0.8,legacy.axes = TRUE,color="red")

g.list+theme\_classic()+ggtitle("data")+annotate(geom="segment",x = 0,y = 0,xend =1,yend = 1)

library(pROC)

library(ggplot2)

roc.list <- roc(test$diagnosis, gd)

g.list <- ggroc(roc.list, alpha = 1, size = 1, legacy.axes = TRUE, color = "#377EB8")

g.list <- g.list +

theme\_minimal(base\_size = 14) +

ggtitle("ROC Curve of Diagnostic Test") +

theme(plot.title = element\_text(hjust = 0.5, face = "bold", color = "#377EB8"),

axis.title = element\_text(face = "bold", color = "#4D4D4D"),

axis.text = element\_text(color = "#4D4D4D")) +

geom\_segment(aes(x = 0, y = 0, xend = 1, yend = 1), linetype = "dashed", color = "#E41A1C") +

annotate("text", x = 0.2, y = 0.9, label = sprintf("AUC = %.3f", auc(roc.list)), size = 5, color = "#4D4D4D")

print(g.list)

library(pROC)

gd <- predict(fit0, newdata = test, type = "fitted")

roc.list <- roc(test$diagnosis, gd)

best\_coords <- coords(roc.list,

x = "best",

best.method = "youden",

ret = c("threshold", "sensitivity", "specificity", "accuracy", "ppv", "npv"))

print(round(best\_coords, 3))

train <- train[complete.cases(train[, c("diagnosis","sapsii","admissionage","sodium","mbp")]), ]

fit0 <- lrm(diagnosis ~ sapsii + admissionage + sodium + mbp,

data = train, x = TRUE, y = TRUE)

cal <- calibrate(fit0, cmethod="hare", method="boot", B=1000)

plot(cal, xlim=c(0,1), ylim=c(0,1),

xlab="Predicted Survival", ylab="Actual Survival")

y\_train\_num <- as.numeric(as.character(train$diagnosis))

p\_hat\_train <- predict(fit0, type="fitted")

p\_hat\_train <- pmin(pmax(p\_hat\_train, 1e-6), 1-1e-6)

brier\_manual <- mean((p\_hat\_train - y\_train\_num)^2)

df\_cal <- data.frame(logit\_p = qlogis(p\_hat\_train), y = y\_train\_num)

fit\_int <- glm(y ~ offset(logit\_p),

family=binomial(link="logit"), data=df\_cal)

intercept\_manual <- coef(fit\_int)[1]

fit\_slope <- glm(y ~ logit\_p,

family=binomial(link="logit"), data=df\_cal)

slope\_manual <- coef(fit\_slope)[2]

if (!requireNamespace("ResourceSelection", quietly=TRUE)) {

install.packages("ResourceSelection")

}

library(ResourceSelection)

hl\_res <- hoslem.test(y\_train\_num, p\_hat\_train, g=10)

eo\_manual <- sum(p\_hat\_train) / sum(y\_train\_num)

cat("\n--- Manual Calibration Metrics (Training) ---\n")

cat("Intercept :", round(intercept\_manual,3), "\n")

cat("Slope :", round(slope\_manual, 3), "\n")

cat("Brier :", round(brier\_manual, 3), "\n")

cat("HL χ² :", round(hl\_res$statistic,2), "\n")

cat("HL P :", signif(hl\_res$p.value,2), "\n")

cat("E/O Ratio :", round(eo\_manual, 2), "\n")

test <- test[complete.cases(test[, c("diagnosis", "sapsii", "admissionage", "sodium", "mbp")]), ]

fit1 <- lrm(diagnosis ~ sapsii + admissionage + sodium + mbp, data = test, x = TRUE, y = TRUE)

cal <- calibrate(fit1, cmethod = "hare", method = "boot", B = 1000)

par(mar = c(5.1, 4.1, 4.1, 2.1))

plot(cal, xlim = c(0, 1), ylim = c(0, 1),

xlab = "Nomogram Predicted Survival", ylab = "Actual Survival")

test <- test[complete.cases(test[, c("diagnosis","sapsii","admissionage","sodium","mbp")]), ]

y\_test\_num <- as.numeric(as.character(test$diagnosis))

p\_hat\_test <- predict(fit0, newdata = test, type = "fitted")

p\_hat\_test <- pmin(pmax(p\_hat\_test, 1e-6), 1-1e-6)

keep <- complete.cases(p\_hat\_test, y\_test\_num)

p\_hat\_test <- p\_hat\_test[keep]

y\_test\_num <- y\_test\_num[keep]

brier\_test <- mean((p\_hat\_test - y\_test\_num)^2)

df\_cal\_test <- data.frame(

logit\_p = qlogis(p\_hat\_test),

y = y\_test\_num

)

fit\_cal\_int\_test <- glm(

y ~ offset(logit\_p),

family = binomial(link = "logit"),

data = df\_cal\_test

)

intercept\_test <- coef(fit\_cal\_int\_test)[1]

fit\_cal\_slope\_test <- glm(

y ~ logit\_p,

family = binomial(link = "logit"),

data = df\_cal\_test

)

slope\_test <- coef(fit\_cal\_slope\_test)[2]

if (!requireNamespace("ResourceSelection", quietly=TRUE)) install.packages("ResourceSelection")

library(ResourceSelection)

hl\_test <- hoslem.test(y\_test\_num, p\_hat\_test, g = 10)

eo\_test <- sum(p\_hat\_test) / sum(y\_test\_num)

cat("\n--- Manual Calibration Metrics (Validation) ---\n")

cat("Intercept (manual) :", round(intercept\_test, 3), "\n")

cat("Slope (manual) :", round(slope\_test, 3), "\n")

cat("Brier Score :", round(brier\_test, 3), "\n")

cat("HL Chi-square :", round(hl\_test$statistic, 2), "\n")

cat("HL P-value :", signif(hl\_test$p.value, 2), "\n")

cat("E/O Ratio :", round(eo\_test, 2), "\n")

library(ggplot2)

library(rmda)

train <- train[complete.cases(train[, c("diagnosis", "sapsii","admissionage","sodium","mbp")]), ]

dca\_model <- decision\_curve(diagnosis ~ sapsii + admissionage + sodium + mbp,

data = train,

study.design = "cohort",

bootstraps = 100)

plot\_decision\_curve(dca\_model,

curve.names = "Logistic Regression Model",

xlim = c(0, 1),

cost.benefit.axis = TRUE,

confidence.intervals = TRUE) +

theme(

legend.title = element\_blank(),

legend.text = element\_text(size = 8),

legend.key.size = unit(0.6, "lines"),

legend.position = c(1, 1),

legend.justification = c(1, 1)

)

test <- test[complete.cases(test[, c("diagnosis", "sapsii","admissionage","sodium","mbp")]), ]

dca\_model\_test <- decision\_curve(diagnosis ~ sapsii + admissionage + sodium + mbp,

data = test,

study.design = "cohort",

bootstraps = 100)

plot\_decision\_curve(

dca\_model\_test,

curve.names = "Logistic Regression Model",

xlim = c(0, 1),

legend.position = "topright",

cost.benefit.axis = TRUE,

confidence.intervals = TRUE

) +

theme(

legend.title = element\_blank(),

legend.text = element\_text(size = 8),

legend.key.size = unit(0.6, "lines")

)

X\_train <- dat\_train[, -1]

y\_train <- dat\_train[, 1]

dtrain <- xgb.DMatrix(data = as.matrix(X\_train), label = dat\_train$diagnosis)

params <- list(objective = "binary:logistic", eval\_metric = "logloss", eta = 0.1, max\_depth = 3)

nrounds <- 100

xgb\_model\_final <- xgboost(params = params, data = dtrain, nrounds = nrounds)

train\_predictions <- predict(xgb\_model\_final, newdata = dtrain)

train\_predictions1 <- ifelse(train\_predictions > 0.5,1,0)

accuracy <- mean(train\_predictions1 == y\_train)

print(paste, accuracy))

X\_test <- dat\_test[, -1]

y\_test <- as.factor(dat\_test$diagnosis)

dtest <- xgb.DMatrix(data = as.matrix(X\_test))

test\_predictions <- predict(xgb\_model\_final, newdata = dtest)

test\_predictions1 <- ifelse(test\_predictions > 0.5,1,0)

accuracy <- mean(test\_predictions1 == y\_test)

print(paste(accuracy))

library(caret)

train$diagnosis <- factor(train$diagnosis, levels = c(0, 1))

train\_predictions1 <- factor(train\_predictions1, levels = c(0, 1))

train\_predictions1 <- as.factor(train\_predictions1)

conf\_matrix <- confusionMatrix(train\_predictions1, train$diagnosis)

conf\_matrix

test\_predictions1 <- as.factor(test\_predictions1)

conf\_matrix <- confusionMatrix(test\_predictions1, test$diagnosis)

conf\_matrix

library(pROC)

library(ggplot2)

train\_predictions <- predict(xgb\_model\_final, newdata = dtrain)

roc.list1 <- roc(train$diagnosis, train\_predictions)

g.list1 <- ggroc(roc.list1, alpha = 1, size = 1, legacy.axes = TRUE, color = "#377EB8")

g.list1 <- g.list1 +

theme\_minimal(base\_size = 14) +

ggtitle("ROC Curve of Diagnostic Test") +

theme(

plot.title = element\_text(hjust = 0.5, face = "bold", color = "#377EB8"),

axis.title = element\_text(face = "bold", color = "#4D4D4D"),

axis.text = element\_text(color = "#4D4D4D")

) +

geom\_segment(aes(x = 0, y = 0, xend = 1, yend = 1), linetype = "dashed", color = "#E41A1C") +

annotate(

geom = "text",

x = 0.2,

y = 0.9,

label = sprintf("AUC = %.3f", auc(roc.list1)),

size = 5,

color = "#4D4D4D"

)

print(g.list1)

best\_metrics <- coords(

roc.list1,

x = "best",

best.method = "youden",

ret = c("threshold",

"sensitivity",

"specificity",

"accuracy",

"ppv",

"npv"),

transpose = FALSE

)

print(round(best\_metrics, 3))

test\_predictions <- predict(xgb\_model\_final, newdata = dtest)

roc.list2 <- roc(test$diagnosis, test\_predictions, quiet = TRUE)

g.list2 <- ggroc(roc.list2, alpha = 1, size = 1, legacy.axes = TRUE,

color = "#377EB8") +

theme\_minimal(14) +

ggtitle("ROC Curve of Diagnostic Test") +

theme(plot.title = element\_text(hjust = .5, face = "bold", color = "#377EB8"),

axis.title = element\_text(face = "bold", color = "#4D4D4D"),

axis.text = element\_text(color = "#4D4D4D")) +

geom\_segment(aes(0,0,xend=1,yend=1), lty = 2, col = "#E41A1C") +

annotate("text", .2, .9,

label = sprintf("AUC = %.3f", auc(roc.list2)),

size = 5, color = "#4D4D4D")

print(g.list2)

cutoff <- 0.656

test\_class <- ifelse(test\_predictions >= cutoff, 1, 0)

pred\_fact <- factor(test\_class, levels = c(0,1))

ref\_fact <- factor(test$diagnosis, levels = c(0,1))

cm\_test <- confusionMatrix(pred\_fact, ref\_fact, positive = "1")

metrics <- c(cm\_test$byClass[c("Sensitivity","Specificity",

"Pos Pred Value","Neg Pred Value")],

Accuracy = cm\_test$overall["Accuracy"])

print(round(metrics, 3))

calibration\_data <- data.frame(

diagnosis = train$diagnosis,

predictions = train\_predictions

)

dd <- datadist(calibration\_data)

options(datadist = 'dd')

fit\_xgb <- lrm(diagnosis ~ predictions, data = calibration\_data, x = TRUE, y = TRUE)

cal\_xgb <- calibrate(fit\_xgb, cmethod = "hare", method = "boot", B = 1000)

par(mar = c(5.1, 4.1, 4.1, 2.1))

plot(cal\_xgb, xlim = c(0, 1), ylim = c(0, 1),

xlab = "Predicted Probability",

ylab = "Actual Probability",

col = "blue", lwd = 2)

abline(0, 1, lty = 2, col = "red")

y\_train\_num <- as.numeric(as.character(train$diagnosis))

p\_hat\_train <- train\_predictions

qlogis(0/1) = ±Inf

p\_hat\_train <- pmin(pmax(p\_hat\_train, 1e-6), 1 - 1e-6)

keep <- complete.cases(y\_train\_num, p\_hat\_train)

y\_train\_num <- y\_train\_num[keep]

p\_hat\_train <- p\_hat\_train[keep]

brier\_xgb <- mean((p\_hat\_train - y\_train\_num)^2)

df\_cal <- data.frame(

logit\_p = qlogis(p\_hat\_train),

y = y\_train\_num

)

fit\_int <- glm(y ~ offset(logit\_p),

family = binomial(link = "logit"),

data = df\_cal)

intercept\_xgb <- coef(fit\_int)[1]

fit\_slope <- glm(y ~ logit\_p,

family = binomial(link = "logit"),

data = df\_cal)

slope\_xgb <- coef(fit\_slope)[2]

if (!requireNamespace("ResourceSelection", quietly = TRUE))

install.packages("ResourceSelection")

library(ResourceSelection)

hl\_xgb <- hoslem.test(y\_train\_num, p\_hat\_train, g = 10)

eo\_xgb <- sum(p\_hat\_train) / sum(y\_train\_num)

cat("\n--- XGBoost ---\n")

cat("Intercept (CITL) :", round(intercept\_xgb, 3), "\n")

cat("Slope :", round(slope\_xgb, 3), "\n")

cat("Brier Score :", round(brier\_xgb, 3), "\n")

cat("HL Chi-square :", round(hl\_xgb$statistic, 2), "\n")

cat("HL P-value :", signif(hl\_xgb$p.value, 2), "\n")

cat("E/O Ratio :", round(eo\_xgb, 2), "\n")

calibration\_data\_test <- data.frame(

diagnosis = test$diagnosis,

predictions = test\_predictions

)

dd <- datadist(calibration\_data\_test)

options(datadist = 'dd')

fit\_xgb\_test <- lrm(diagnosis ~ predictions, data = calibration\_data\_test, x = TRUE, y = TRUE)

cal\_xgb\_test <- calibrate(fit\_xgb\_test, cmethod = "hare", method = "boot", B = 1000)

par(mar = c(5.1, 4.1, 4.1, 2.1))

plot(cal\_xgb\_test, xlim = c(0, 1), ylim = c(0, 1),

xlab = "Predicted Probability",

ylab = "Actual Probability",

col = "blue", lwd = 2)

abline(0, 1, lty = 2, col = "red")

y\_val <- as.numeric(as.character(test$diagnosis))

p\_hat <- test\_predictions

p\_hat <- pmin(pmax(p\_hat, 1e-6), 1 - 1e-6)

keep <- complete.cases(y\_val, p\_hat)

y\_val <- y\_val[keep]

p\_hat <- p\_hat[keep]

brier\_val <- mean((p\_hat - y\_val)^2)

df\_cal\_val <- data.frame(

logit\_p = qlogis(p\_hat),

y = y\_val

)

fit\_int\_val <- glm(

y ~ offset(logit\_p),

family = binomial(link = "logit"),

data = df\_cal\_val

)

intercept\_val <- coef(fit\_int\_val)[1]

fit\_slope\_val <- glm(

y ~ logit\_p,

family = binomial(link = "logit"),

data = df\_cal\_val

)

slope\_val <- coef(fit\_slope\_val)[2]

if (!requireNamespace("ResourceSelection", quietly = TRUE))

install.packages("ResourceSelection")

library(ResourceSelection)

hl\_val <- hoslem.test(y\_val, p\_hat, g = 10)

eo\_val <- sum(p\_hat) / sum(y\_val)

cat("\n--- XGBoost ---\n")

cat("Intercept (CITL) :", round(intercept\_val, 3), "\n")

cat("Slope :", round(slope\_val, 3), "\n")

cat("Brier Score :", round(brier\_val, 3), "\n")

cat("HL Chi-square :", round(hl\_val$statistic, 2), "\n")

cat("HL P-value :", signif(hl\_val$p.value, 2), "\n")

cat("E/O Ratio :", round(eo\_val, 2), "\n")

## SHAP####

install.packages("shapviz")

library(shapviz)

shap <- shapviz(xgb\_model\_final,X\_pred =data.matrix(dat\_train[,-1]))

sv\_waterfall(shap,

row\_id = 4,

fill\_colors=c("#E31F1CD7", "#246EE3"))

sv\_force(shap,

row\_id = 45,

max\_display = 10,

fill\_colors=c("#E31F1CD7", "#246EE3"))

sv\_waterfall(shap,

row\_id = 2,

fill\_colors=c("#E31F1CD7", "#246EE3"))

sv\_force(shap,

row\_id = 2,

max\_display = 10,

fill\_colors=c("#E31F1CD7", "#246EE3"))

sv\_importance(shap, kind = "beeswarm")

sv\_dependence(shap,

v = c("sapsii","admissionage","sodium","mbp"))

sv\_dependence(shap,

v = "sapsii")

sv\_dependence(shap,

v = "admissionage")

sv\_dependence(shap,

v = "sodium")

sv\_dependence(shap,

v = "mbp")

#XGBOOST的训练集的决策曲线

# 加载必要的包

library(rmda) # 决策曲线分析包

library(ggplot2) # 绘图包

# 准备训练集和验证集数据

train$prob <- train\_predictions # XGBoost 在训练集上的预测概率

test$prob <- test\_predictions # XGBoost 在验证集上的预测概率

# 确保诊断结果为二元变量（0 和 1）

train$diagnosis <- as.numeric(train$diagnosis)

test$diagnosis <- as.numeric(test$diagnosis)

# 决策曲线分析（训练集）

dca\_model\_train <- decision\_curve(

formula = diagnosis ~ prob, # 使用预测概率

data = train,

study.design = "cohort",

bootstraps = 100 # 引导计算置信区间

)

# 决策曲线分析（验证集）

dca\_model\_test <- decision\_curve(

formula = diagnosis ~ prob, # 使用预测概率

data = test,

study.design = "cohort",

bootstraps = 100 # 引导计算置信区间

)

# 绘制训练集的决策曲线

plot\_decision\_curve(

dca\_model\_train,

curve.names = "XGBoost Model",

xlim = c(0, 1), # 设置x轴范围

legend.position = "topright", # 图例放置在右上角

cost.benefit.axis = TRUE, # 显示成本收益轴

confidence.intervals = TRUE # 显示置信区间

) +

theme(

legend.title = element\_blank(), # 去除图例标题

legend.text = element\_text(size = 8), # 调整图例字体大小

legend.key.size = unit(0.6, "lines") # 调整图例方框大小

) +

ggtitle("Decision Curve Analysis (Test)") +

labs(x = "High Risk Threshold", y = "Net Benefit")

# 绘制验证集的决策曲线

plot\_decision\_curve(

dca\_model\_test,

curve.names = "XGBoost Model",

xlim = c(0, 1), # 设置x轴范围

legend.position = "topright", # 图例放置在右上角

cost.benefit.axis = TRUE, # 显示成本收益轴

confidence.intervals = TRUE # 显示置信区间

) +

theme(

legend.title = element\_blank(), # 去除图例标题

legend.text = element\_text(size = 8), # 调整图例字体大小

legend.key.size = unit(0.6, "lines") # 调整图例方框大小

) +

ggtitle("Decision Curve Analysis (Test)") +

labs(x = "High Risk Threshold", y = "Net Benefit")