Chapter-5-Figures-Code.R

Stephanie

2025-04-02

#### Chapter 5 Coding Exercises ####  
  
#### Table 5.1: Prediction Error Metrics from a Model for Toyota Car Prices, Training and Validation ####  
  
# package forecast is required to evaluate performance  
library(forecast)

## Registered S3 method overwritten by 'quantmod':  
## method from  
## as.zoo.data.frame zoo

# load file  
toyota.corolla.df <- read.csv("ToyotaCorolla.csv")  
  
# randomly generate training and validation sets  
training <- sample(toyota.corolla.df$Id, 600)  
validation <- sample(setdiff(toyota.corolla.df$Id, training), 400)  
  
# run linear regression model  
reg <- lm(Price~., data=toyota.corolla.df[,-c(1,2,8,11)], subset=training,  
 na.action=na.exclude)  
pred\_t <- predict(reg, na.action=na.pass)  
pred\_v <- predict(reg, newdata=toyota.corolla.df[validation,-c(1,2,8,11)],  
 na.action=na.pass)

## Warning in predict.lm(reg, newdata = toyota.corolla.df[validation, -c(1, :  
## prediction from rank-deficient fit; attr(\*, "non-estim") has doubtful cases

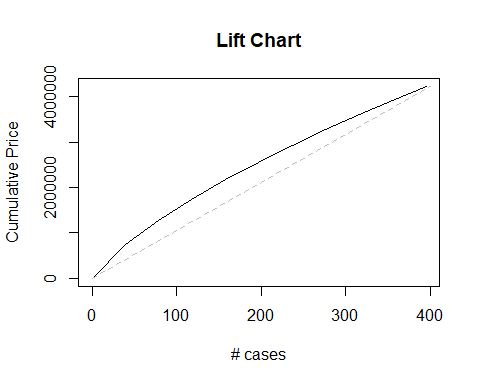
## evaluate performance  
# training  
accuracy(pred\_t, toyota.corolla.df[training,]$Price)

## ME RMSE MAE MPE MAPE  
## Test set 1.381312e-11 1094.674 824.7784 -0.961128 8.297118

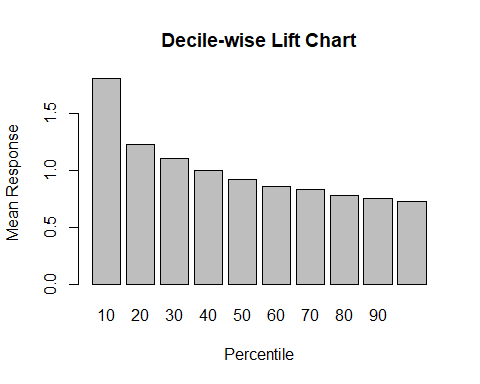
# validation  
accuracy(pred\_v, toyota.corolla.df[validation,]$Price)

## ME RMSE MAE MPE MAPE  
## Test set 49.7653 1157.767 842.6724 -0.126014 8.238515

#### Figure 5.2: Lift Chart and Decile Lift Chart for Continuous Outcome Variable ####  
  
toyota.corolla.df <- read.csv("ToyotaCorolla.csv")  
  
# remove missing Price data  
toyota.corolla.df <-  
 toyota.corolla.df[!is.na(toyota.corolla.df[validation,]$Price),]  
  
# generate random Training and Validation sets  
training <- sample(toyota.corolla.df$Id, 600)  
validation <- sample(toyota.corolla.df$Id, 400)  
  
# regression model based on all numerical predictors  
reg <- lm(Price~., data = toyota.corolla.df[,-c(1,2,8,11)], subset = training)  
  
# predictions  
pred\_v <- predict(reg, newdata = toyota.corolla.df[validation, -c(1,2,8,11)])  
  
# load package gains, compute gains (we will use package caret for categorical y later)  
library(gains)  
gain <- gains(toyota.corolla.df[validation,]$Price[!is.na(pred\_v)],  
 pred\_v[!is.na(pred\_v)])  
  
# cumulative lift chart  
options(scipen = 999) # avoid scientific notation  
# we will compute the gain relative to price  
price <- toyota.corolla.df[validation,]$Price[!is.na(toyota.corolla.df[validation,]$Price)]  
plot(c(0, gain$cume.pct.of.total\*sum(price))~c(0, gain$cume.obs),  
 xlab="# cases", ylab="Cumulative Price", main="Lift Chart", type="l")  
  
# baseline  
lines(c(0,sum(price))~c(0,dim(toyota.corolla.df[validation,])[1]),  
 col="gray", lty=2)



# Decile-wise Lift Chart  
barplot(gain$mean.resp/mean(price), names.arg = gain$depth,  
 xlab = "Percentile", ylab = "Mean Response",   
 main = "Decile-wise Lift Chart")



#### Table 5.5: Confusion Matrices Based on Cutoffs of 0.5, 0.25, and 0.75 ####  
  
# install "caret" package first  
# install.packages("caret")  
  
library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

owner.df <- read.csv("ownerExample.csv")  
## cutoff = 0.5  
# confusionMatrix(ifelse(owner.df$Probability>0.5, 'owner', 'nonowner'),  
 # owner.df$Class)  
# note: "reference" = "actual"  
  
# have to code again to fix coding errors  
owner.df <- read.csv("ownerExample.csv")  
## cutoff = 0.5  
# Create predicted classes as a factor  
predicted\_classes <- factor(ifelse(owner.df$Probability > 0.5, 'owner', 'nonowner'),  
 levels = c('nonowner', 'owner'))  
# Convert actual classes to a factor, ensuring levels match if possible  
actual\_classes <- factor(owner.df$Class, levels = c('nonowner', 'owner'))  
# Now run the confusionMatrix  
confusionMatrix(data = predicted\_classes, reference = actual\_classes)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction nonowner owner  
## nonowner 10 1  
## owner 2 11  
##   
## Accuracy : 0.875   
## 95% CI : (0.6764, 0.9734)  
## No Information Rate : 0.5   
## P-Value [Acc > NIR] : 0.0001386   
##   
## Kappa : 0.75   
##   
## Mcnemar's Test P-Value : 1.0000000   
##   
## Sensitivity : 0.8333   
## Specificity : 0.9167   
## Pos Pred Value : 0.9091   
## Neg Pred Value : 0.8462   
## Prevalence : 0.5000   
## Detection Rate : 0.4167   
## Detection Prevalence : 0.4583   
## Balanced Accuracy : 0.8750   
##   
## 'Positive' Class : nonowner   
##

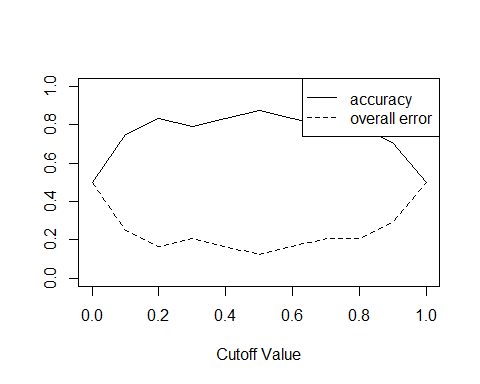
# note: "reference" = "actual"  
  
# coding for the 0.25 cutoff  
owner.df <- read.csv("ownerExample.csv")  
## cutoff = 0.25  
# Create predicted classes as a factor  
predicted\_classes <- factor(ifelse(owner.df$Probability > 0.25, 'owner', 'nonowner'),  
 levels = c('nonowner', 'owner'))  
# Convert actual classes to a factor, ensuring levels match if possible  
actual\_classes <- factor(owner.df$Class, levels = c('nonowner', 'owner'))  
# Now run the confusionMatrix  
confusionMatrix(data = predicted\_classes, reference = actual\_classes)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction nonowner owner  
## nonowner 8 1  
## owner 4 11  
##   
## Accuracy : 0.7917   
## 95% CI : (0.5785, 0.9287)  
## No Information Rate : 0.5   
## P-Value [Acc > NIR] : 0.003305   
##   
## Kappa : 0.5833   
##   
## Mcnemar's Test P-Value : 0.371093   
##   
## Sensitivity : 0.6667   
## Specificity : 0.9167   
## Pos Pred Value : 0.8889   
## Neg Pred Value : 0.7333   
## Prevalence : 0.5000   
## Detection Rate : 0.3333   
## Detection Prevalence : 0.3750   
## Balanced Accuracy : 0.7917   
##   
## 'Positive' Class : nonowner   
##

# note: "reference" = "actual"  
  
# coding for the 0.75 cutoff  
owner.df <- read.csv("ownerExample.csv")  
## cutoff = 0.75  
# Create predicted classes as a factor  
predicted\_classes <- factor(ifelse(owner.df$Probability > 0.75, 'owner', 'nonowner'),  
 levels = c('nonowner', 'owner'))  
# Convert actual classes to a factor, ensuring levels match if possible  
actual\_classes <- factor(owner.df$Class, levels = c('nonowner', 'owner'))  
# Now run the confusionMatrix  
confusionMatrix(data = predicted\_classes, reference = actual\_classes)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction nonowner owner  
## nonowner 11 5  
## owner 1 7  
##   
## Accuracy : 0.75   
## 95% CI : (0.5329, 0.9023)  
## No Information Rate : 0.5   
## P-Value [Acc > NIR] : 0.01133   
##   
## Kappa : 0.5   
##   
## Mcnemar's Test P-Value : 0.22067   
##   
## Sensitivity : 0.9167   
## Specificity : 0.5833   
## Pos Pred Value : 0.6875   
## Neg Pred Value : 0.8750   
## Prevalence : 0.5000   
## Detection Rate : 0.4583   
## Detection Prevalence : 0.6667   
## Balanced Accuracy : 0.7500   
##   
## 'Positive' Class : nonowner   
##

# note: "reference" = "actual"  
  
  
#### Figure 5.4: Plotting Accuracy and Overall Error as a Function of the Cutoff Value ####  
  
# create empty accuracy table  
accT = c()  
  
# compute accuracy per cutoff  
for (cut in seq(0,1,0.1)){  
 predicted\_classes <- factor(ifelse(owner.df$Probability > cut, 'owner', 'nonowner'),  
 levels = c('nonowner', 'owner'))  
 actual\_classes <- factor(owner.df$Class, levels = c('nonowner', 'owner'))  
 cm <- confusionMatrix(data = predicted\_classes, reference = actual\_classes)  
 accT = c(accT, cm$overall[1])  
}  
  
# plot accuracy  
plot(accT ~ seq(0,1,0.1),   
 xlab = "Cutoff Value",  
 ylab = "",  
 type = "l",   
 ylim = c(0,1))  
lines(1-accT ~ seq(0,1,0.1),  
 type = "l",  
 lty = 2)  
legend("topright",  
 c("accuracy", "overall error"),  
 lty = c(1,2),  
 merge = TRUE)



#### Figure 5.5: ROC Curve for Riding Mowers Example ####  
# install PROC package before using the library  
# install.packages("pROC")  
library(pROC)

## Type 'citation("pROC")' for a citation.

##   
## Attaching package: 'pROC'

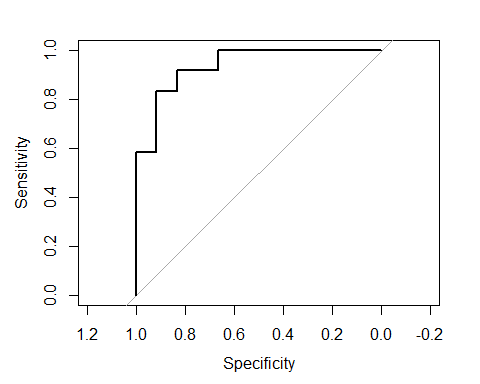
## The following objects are masked from 'package:stats':  
##   
## cov, smooth, var

r <- roc(owner.df$Class, owner.df$Probability)

## Setting levels: control = nonowner, case = owner

## Setting direction: controls < cases

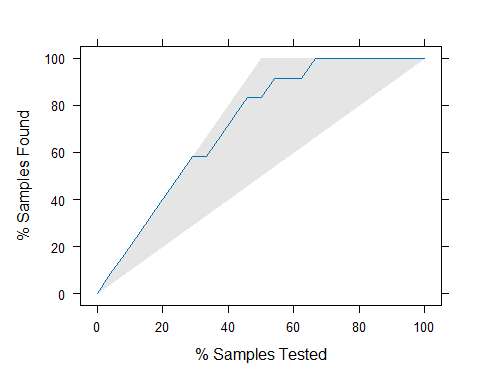
plot.roc(r)



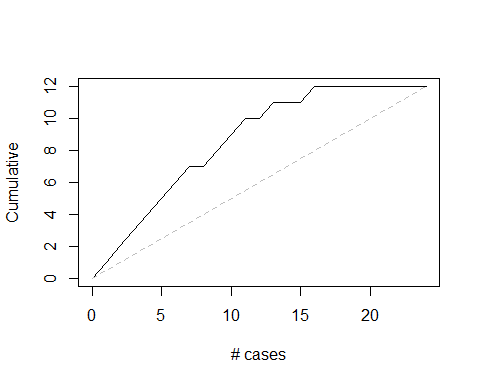
# compute auc  
auc(r)

## Area under the curve: 0.9375

#### Figure 5.6: Lift Chart for the Mower Example Using Caret Package and Gains Package ####  
  
# first option with 'caret' library:  
library(caret)  
actual\_classes <- factor(owner.df$Class,   
 levels = c('nonowner', 'owner'))  
  
lift.example <- lift(relevel(as.factor(actual\_classes), ref="owner") ~ owner.df$Probability, data = owner.df)  
xyplot(lift.example, plot = "gain")



# Second option with 'gains' library:  
library(gains)  
lift.df <- read.csv("liftExample.csv")  
gain <- gains(lift.df$actual, lift.df$prob, groups=dim(lift.df)[1])  
plot(c(0, gain$cume.pct.of.total\*sum(lift.df$actual)) ~ c(0, gain$cume.obs),  
 xlab = "# cases", ylab = "Cumulative", type="l")  
lines(c(0,sum(lift.df$actual))~c(0,dim(lift.df)[1]),  
 col="gray", lty=2)



#### Figure 5.7: Decile Lift Chart ####  
  
# use gains() to compute deciles  
# when using the caret package, deciles must be computed manually.  
  
lift.df <- read.csv("liftExample.csv")  
gain <- gains(lift.df$actual, lift.df$prob,)  
barplot(gain$mean.resp / mean(lift.df$actual),  
 names.arg = gain$depth,   
 xlab = "Precentile",  
 ylab = "Mean Response",  
 main = "Decile-wise lift chart")

