Math448: Chapter 5 HW

Zahraa Alshalal

2023-04-18

## Conceptual Questions:

Exercise 3:

* + K-fold cross-validation is used to evaluate the performance of a model on a given data-set.
  + It works by randomly splitting the data-set into k equal-sized subsets or folds.
  + Then the algorithm iterates through k rounds of training and evaluation. In each round, one fold is held out as a validation set, while the remaining k-1 folds are used for training the model.
  + The validation set is used evaluate the model.
  + The test error is estimated by averaging the k resulting MSE estimates.
    1. Advantages compared to The validation set approach:
  + k-fold cross-validation provides a more accurate estimate of the model’s performance because it uses all available data for training and testing.
  + k-fold cross-validation reduces the risk of over-fitting because it trains and evaluates the model on multiple subsets of the data rather than just one.
  + Disadvantages compared to The validation set approach:
  + k-fold cross-validation is computationally more expensive.
  + k-fold cross-validation may not be suitable for small data-sets when the data-set has high variance,
    1. Advantages compared to LOOCV:
  + k-fold cross-validation is less computationally expensive.
  + k-fold cross-validation can provide a more accurate estimate of the model’s performance.
  + Disadvantages compared to LOOCV:
  + LOOCV can provide a more accurate estimate of the model’s performance than k-fold cross-validation when the data-set has a small sample size.
  + LOOCV can be less biased when the data-set has a small sample size.

## Applied Questions:

### Exercise 5:

library(ISLR)  
summary(Default)

## default student balance income   
## No :9667 No :7056 Min. : 0.0 Min. : 772   
## Yes: 333 Yes:2944 1st Qu.: 481.7 1st Qu.:21340   
## Median : 823.6 Median :34553   
## Mean : 835.4 Mean :33517   
## 3rd Qu.:1166.3 3rd Qu.:43808   
## Max. :2654.3 Max. :73554

attach(Default)

#### a.

set.seed(1)  
fit.glm = glm(default ~ income + balance, data = Default, family = "binomial")  
summary(fit.glm)

##   
## Call:  
## glm(formula = default ~ income + balance, family = "binomial",   
## data = Default)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.4725 -0.1444 -0.0574 -0.0211 3.7245   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.154e+01 4.348e-01 -26.545 < 2e-16 \*\*\*  
## income 2.081e-05 4.985e-06 4.174 2.99e-05 \*\*\*  
## balance 5.647e-03 2.274e-04 24.836 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 2920.6 on 9999 degrees of freedom  
## Residual deviance: 1579.0 on 9997 degrees of freedom  
## AIC: 1585  
##   
## Number of Fisher Scoring iterations: 8

#### b.

# i.  
trainset = sample(dim(Default)[1], dim(Default)[1] / 2)  
  
# ii.   
fit.trainset = glm(default ~ income + balance, data = Default, family = "binomial", subset = trainset)  
  
# iii.   
glm.pred = rep("No", dim(Default)[1]/2)  
  
glm.probs = predict(fit.trainset, Default[-trainset, ], type = "response")  
  
glm.pred[glm.probs > 0.5] = "Yes"  
  
# iv.   
et = mean(glm.pred != Default[-trainset, "default"])  
cat("The test error rate:", et\*100, "% from validation set approach")

## The test error rate: 2.54 % from validation set approach

#### c.

# 1  
# i.  
trainset = sample(dim(Default)[1], dim(Default)[1] / 2)  
  
# ii.   
fit.trainset = glm(default ~ income + balance, data = Default, family = "binomial", subset = trainset)  
  
# iii.   
glm.pred = rep("No", dim(Default)[1]/2)  
  
glm.probs = predict(fit.trainset, Default[-trainset, ], type = "response")  
  
glm.pred[glm.probs > 0.5] = "Yes"  
  
# iv.   
et = mean(glm.pred != Default[-trainset, "default"])  
cat("The test error rate:", et\*100, "% from validation set approach")

## The test error rate: 2.74 % from validation set approach

# 2  
# i.  
trainset = sample(dim(Default)[1], dim(Default)[1] / 2)  
  
# ii.   
fit.trainset = glm(default ~ income + balance, data = Default, family = "binomial", subset = trainset)  
  
# iii.   
glm.pred = rep("No", dim(Default)[1]/2)  
  
glm.probs = predict(fit.trainset, Default[-trainset, ], type = "response")  
  
glm.pred[glm.probs > 0.5] = "Yes"  
  
# iv.   
et = mean(glm.pred != Default[-trainset, "default"])  
cat("The test error rate:", et\*100, "% from validation set approach")

## The test error rate: 2.44 % from validation set approach

# 3  
# i.  
trainset = sample(dim(Default)[1], dim(Default)[1] / 2)  
  
# ii.   
fit.trainset = glm(default ~ income + balance, data = Default, family = "binomial", subset = trainset)  
  
# iii.   
glm.pred = rep("No", dim(Default)[1]/2)  
  
glm.probs = predict(fit.trainset, Default[-trainset, ], type = "response")  
  
glm.pred[glm.probs > 0.5] = "Yes"  
  
# iv.   
et = mean(glm.pred != Default[-trainset, "default"])  
cat("The test error rate:", et\*100, "% from validation set approach")

## The test error rate: 2.44 % from validation set approach

* The test error rate hovers around 2.7%.

#### d.

# i.  
trainset = sample(dim(Default)[1], dim(Default)[1] / 2)  
  
# ii.   
fit.trainset = glm(default ~ income + balance + student, data = Default, family = "binomial", subset = trainset)  
  
# iii.   
glm.pred = rep("No", dim(Default)[1]/2)  
  
glm.probs = predict(fit.trainset, Default[-trainset, ], type = "response")  
  
glm.pred[glm.probs > 0.5] = "Yes"  
  
# iv.   
et = mean(glm.pred != Default[-trainset, "default"])  
cat("The test error rate:", et\*100, "% from validation set approach")

## The test error rate: 2.78 % from validation set approach

* Adding the “student” dummy variable does not lead to a reduction in the validation set estimate of the test error rate.

### Exercise 6:

#### a.

set.seed(1)  
fit.glm = glm(default ~ income + balance, data = Default, family = "binomial")  
summary(fit.glm)

##   
## Call:  
## glm(formula = default ~ income + balance, family = "binomial",   
## data = Default)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.4725 -0.1444 -0.0574 -0.0211 3.7245   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.154e+01 4.348e-01 -26.545 < 2e-16 \*\*\*  
## income 2.081e-05 4.985e-06 4.174 2.99e-05 \*\*\*  
## balance 5.647e-03 2.274e-04 24.836 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 2920.6 on 9999 degrees of freedom  
## Residual deviance: 1579.0 on 9997 degrees of freedom  
## AIC: 1585  
##   
## Number of Fisher Scoring iterations: 8

* The standard error for income:
* The standard error for balance:

#### b.

boot.fn <- function(data, index) {  
 fit <- glm(default ~ income + balance, data = data, family = "binomial", subset = index)  
 return (coef(fit))  
}

#### c.

library(boot)  
boot(Default, boot.fn, 1000)

##   
## ORDINARY NONPARAMETRIC BOOTSTRAP  
##   
##   
## Call:  
## boot(data = Default, statistic = boot.fn, R = 1000)  
##   
##   
## Bootstrap Statistics :  
## original bias std. error  
## t1\* -1.154047e+01 -3.945460e-02 4.344722e-01  
## t2\* 2.080898e-05 1.680317e-07 4.866284e-06  
## t3\* 5.647103e-03 1.855765e-05 2.298949e-04

* The standard error:  
  // //

#### d.

* The estimated standard errors are almost exactly the same as the calculated standard errors. This shows the practical uses of the bootstrap.

### Exercise 7:

summary(Weekly)

## Year Lag1 Lag2 Lag3   
## Min. :1990 Min. :-18.1950 Min. :-18.1950 Min. :-18.1950   
## 1st Qu.:1995 1st Qu.: -1.1540 1st Qu.: -1.1540 1st Qu.: -1.1580   
## Median :2000 Median : 0.2410 Median : 0.2410 Median : 0.2410   
## Mean :2000 Mean : 0.1506 Mean : 0.1511 Mean : 0.1472   
## 3rd Qu.:2005 3rd Qu.: 1.4050 3rd Qu.: 1.4090 3rd Qu.: 1.4090   
## Max. :2010 Max. : 12.0260 Max. : 12.0260 Max. : 12.0260   
## Lag4 Lag5 Volume Today   
## Min. :-18.1950 Min. :-18.1950 Min. :0.08747 Min. :-18.1950   
## 1st Qu.: -1.1580 1st Qu.: -1.1660 1st Qu.:0.33202 1st Qu.: -1.1540   
## Median : 0.2380 Median : 0.2340 Median :1.00268 Median : 0.2410   
## Mean : 0.1458 Mean : 0.1399 Mean :1.57462 Mean : 0.1499   
## 3rd Qu.: 1.4090 3rd Qu.: 1.4050 3rd Qu.:2.05373 3rd Qu.: 1.4050   
## Max. : 12.0260 Max. : 12.0260 Max. :9.32821 Max. : 12.0260   
## Direction   
## Down:484   
## Up :605   
##   
##   
##   
##

set.seed(1)  
attach(Weekly)

#### a.

Weekly.fit = glm(Direction ~ Lag1 + Lag2, data = Weekly, family = binomial)  
summary(Weekly.fit)

##   
## Call:  
## glm(formula = Direction ~ Lag1 + Lag2, family = binomial, data = Weekly)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.623 -1.261 1.001 1.083 1.506   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 0.22122 0.06147 3.599 0.000319 \*\*\*  
## Lag1 -0.03872 0.02622 -1.477 0.139672   
## Lag2 0.06025 0.02655 2.270 0.023232 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 1496.2 on 1088 degrees of freedom  
## Residual deviance: 1488.2 on 1086 degrees of freedom  
## AIC: 1494.2  
##   
## Number of Fisher Scoring iterations: 4

#### b.

butthefirstobservation.fit = glm(Direction ~ Lag1 + Lag2, data = Weekly[-1, ], family = binomial)  
summary(butthefirstobservation.fit)

##   
## Call:  
## glm(formula = Direction ~ Lag1 + Lag2, family = binomial, data = Weekly[-1,   
## ])  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.6258 -1.2617 0.9999 1.0819 1.5071   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 0.22324 0.06150 3.630 0.000283 \*\*\*  
## Lag1 -0.03843 0.02622 -1.466 0.142683   
## Lag2 0.06085 0.02656 2.291 0.021971 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 1494.6 on 1087 degrees of freedom  
## Residual deviance: 1486.5 on 1085 degrees of freedom  
## AIC: 1492.5  
##   
## Number of Fisher Scoring iterations: 4

#### c.

predict(butthefirstobservation.fit, newdata = Weekly[1,], type = "response") > 0.5

## 1   
## TRUE

Weekly$Direction[1]

## [1] Down  
## Levels: Down Up

* Prediction was UP, true Direction was DOWN.

#### d.

count = rep(0, dim(Weekly)[1])  
for (i in 1:(dim(Weekly)[1])) {  
 glm.fit = glm(Direction ~ Lag1 + Lag2, data = Weekly[-i, ], family = binomial)  
 is\_up = predict.glm(glm.fit, Weekly[i, ], type = "response") > 0.5  
 is\_true\_up = Weekly[i, ]$Direction == "Up"  
 if (is\_up != is\_true\_up)   
 count[i] = 1  
}  
sum(count)

## [1] 490

* 490 errors.

#### e.

mean(count)

## [1] 0.4499541

* LOOCV estimates a test error rate of 45%.