HW5

Trey Davidson

2024-09-27

### Question 1

A.) The probability of us picking the jth one is going to be 1 - (probability of us not picking jth one). Since there is n - 1 non jth terms. Then the probability of us not picking the jth terms is (n - 1) / n.

We then plug that into 1 - (probability of us not picking jth one). to get 1 - [(n - 1) / n] ]

B.) The first and second random ones are indpendent meaning that one sample could be repated therfore there is no differnce in probability compared to A just like how flipping a coin. 1 - [(n - 1) / n]

C.) The odds of not picking one one time is 1 - 1/n. If we do this n number of times and each are indpednent meaning the results of 1 does not effect the results of the sample we pick the next time then we just take it to the nth power leaving us with (1- 1/n)^n

# D   
  
nth\_chance <- function(x)  
{  
 return(1-(1-1/x)^x)  
}  
  
  
nth\_chance(5)

## [1] 0.67232

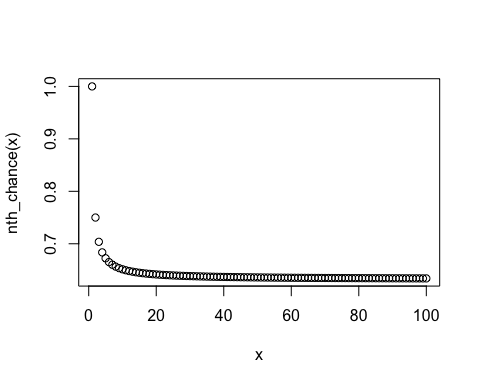
#E  
nth\_chance(100)

## [1] 0.6339677

#F  
nth\_chance(10000)

## [1] 0.632139

#G  
x <- (1:100)  
  
plot(x, nth\_chance(x))



store <- rep(NA, 10000)  
for (i in 1:10000)  
{  
 store[i] <- sum(sample(1:100, rep=TRUE) == 4) > 0   
   
}  
  
mean(store)

## [1] 0.6399

#I am consistnitly seeing aprox 0.63 which is roughly what I expected

### Questiion 5

library(ISLR2)  
  
log\_reg <- glm(default ~ income+balance, data=Default, family = "binomial")  
  
  
  
  
  
train\_proportion <- 0.5   
  
vald\_error <- numeric(3)  
  
for (i in 1:3) {  
# Generate a random sample of indexes  
train\_index <- sample(1:nrow(Default), size = train\_proportion \* nrow(Default))  
  
# Create training and testing datasets  
train <- Default[train\_index, ]  
test <- Default[-train\_index, ]  
  
train\_log\_reg <- glm(default ~ income + balance, data = train, family = "binomial")  
  
test$predicted\_probs <- predict(log\_reg, newdata = test, type = "response")  
  
test$predicted\_class <- ifelse(test$predicted\_probs > 0.5, "Yes", "No")  
  
confusion\_matrix <- table(test$default, test$predicted\_class)  
  
misclassified <- sum(confusion\_matrix) - sum(diag(confusion\_matrix))  
validation\_error <- misclassified / sum(confusion\_matrix)  
  
vald\_error[i] <- validation\_error  
  
}  
  
mean(vald\_error)

## [1] 0.02746667

# The test error looks to be quite low.   
  
  
model\_with\_student <- glm(default ~ income + balance + student, data = train, family = "binomial")  
  
  
for (i in 1:10) {  
# Generate a random sample of indexes  
train\_index <- sample(1:nrow(Default), size = train\_proportion \* nrow(Default))  
  
# Create training and testing datasets  
train <- Default[train\_index, ]  
test <- Default[-train\_index, ]  
  
train\_log\_reg <- glm(default ~ income + balance + student, data = train, family = "binomial")  
  
test$predicted\_probs <- predict(log\_reg, newdata = test, type = "response")  
  
test$predicted\_class <- ifelse(test$predicted\_probs > 0.5, "Yes", "No")  
  
confusion\_matrix <- table(test$default, test$predicted\_class)  
  
misclassified <- sum(confusion\_matrix) - sum(diag(confusion\_matrix))  
validation\_error <- misclassified / sum(confusion\_matrix)  
  
vald\_error[i] <- validation\_error  
  
}  
  
mean(vald\_error)

## [1] 0.0262

# Adding student doesn't seem to change error by a meaninful amount

### Question 7

# A.  
log\_model <- glm(Direction ~ Lag1 + Lag2, data=Weekly, family = "binomial")  
  
summary(log\_model)

##   
## Call:  
## glm(formula = Direction ~ Lag1 + Lag2, family = "binomial", data = Weekly)  
##   
## 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.  
log\_model\_minus\_one <- glm(Direction ~ Lag1 + Lag2, data= Weekly[-1, ], family = "binomial")  
  
  
#C.  
first\_obs <- data.frame(Lag1 = Weekly$Lag1[1], Lag2 = Weekly$Lag2[1])  
  
  
  
predicted\_prob <- predict(log\_model\_minus\_one, newdata = first\_obs, type = "response")  
  
predicted\_class <- ifelse(predicted\_prob > 0.5, "Up", "Down")  
  
actual\_class <- Weekly$Direction[1]  
  
predicted\_class

## 1   
## "Up"

Weekly$Direction[1]

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

# My moddel was correcrt  
 errors <- numeric(nrow(Weekly))  
 i <- -1  
# D  
for (i in 1:nrow(Weekly)) {  
 # i. Fit a model using all but the ith observation  
 model\_loocv <- glm(Direction ~ Lag1 + Lag2, data = Weekly[-i, ], family = "binomial")  
   
 # ii. Compute the posterior probability for the ith observation  
 new\_data <- data.frame(Lag1 = Weekly$Lag1[i], Lag2 = Weekly$Lag2[i])  
 predicted\_prob <- predict(model\_loocv, newdata = new\_data, type = "response")  
   
 # iii. Predict whether the market moves up  
 predicted\_class <- ifelse(predicted\_prob > 0.5, "Up", "Down")  
   
 # iv. Determine if an error was made  
 actual\_class <- Weekly$Direction[i]  
 errors[i] <- ifelse(predicted\_class == actual\_class, 0, 1) # 0 if correct, 1 if error  
}  
  
# E Calculate LOOCV error estimate  
loocv\_error <- mean(errors)  
print(paste("Test Error:", round(loocv\_error, 4)))

## [1] "Test Error: 0.45"

# 0.45 error rate implying that our model is slighlty better then a coin toss

### Question 9

# A.)  
mu\_hat <- mean(Boston$medv)  
print(mu\_hat)

## [1] 22.53281

# B.)  
  
# Estimate the standard deviation  
sd\_medv <- sd(Boston$medv)  
  
# Estimate the standard error  
n <- nrow(Boston)  
se\_mu\_hat <- sd\_medv / sqrt(n)  
print(se\_mu\_hat)

## [1] 0.4088611

#C.)  
  
# Bootstrap function to calculate mean  
set.seed(123) # For reproducibility  
B <- 1000 # Number of bootstrap samples  
bootstrap\_means <- numeric(B)  
  
for (i in 1:B) {  
 bootstrap\_sample <- sample(Boston$medv, n, replace = TRUE)  
 bootstrap\_means[i] <- mean(bootstrap\_sample)  
}  
  
# Estimate the standard error from bootstrap  
se\_bootstrap <- sd(bootstrap\_means)  
print(se\_bootstrap)

## [1] 0.4185474

#D.)  
  
# Confidence interval using bootstrap  
conf\_interval <- c(mu\_hat - 2 \* se\_bootstrap, mu\_hat + 2 \* se\_bootstrap)  
print(conf\_interval)

## [1] 21.69571 23.36990

# Compare with t-test  
t\_test <- t.test(Boston$medv)  
print(t\_test$conf.int)

## [1] 21.72953 23.33608  
## attr(,"conf.level")  
## [1] 0.95

#E.)  
  
# Estimate for the median  
mu\_med <- median(Boston$medv)  
print(mu\_med)

## [1] 21.2

#F.)  
  
# Bootstrap for median  
bootstrap\_medians <- numeric(B)  
  
for (i in 1:B) {  
 bootstrap\_sample <- sample(Boston$medv, n, replace = TRUE)  
 bootstrap\_medians[i] <- median(bootstrap\_sample)  
}  
  
# Estimate the standard error from bootstrap  
se\_bootstrap\_median <- sd(bootstrap\_medians)  
print(se\_bootstrap\_median)

## [1] 0.3779944

#G.)  
  
# Estimate for the 10th percentile  
mu\_0.1 <- quantile(Boston$medv, 0.1)  
print(mu\_0.1)

## 10%   
## 12.75

#H.)  
  
# Bootstrap for the 10th percentile  
bootstrap\_0.1 <- numeric(B)  
  
for (i in 1:B) {  
 bootstrap\_sample <- sample(Boston$medv, n, replace = TRUE)  
 bootstrap\_0.1[i] <- quantile(bootstrap\_sample, 0.1)  
}  
  
# Estimate the standard error from bootstrap  
se\_bootstrap\_0.1 <- sd(bootstrap\_0.1)  
print(se\_bootstrap\_0.1)

## [1] 0.5069193