Jeffery Watson **DS_740**
Assignment Lesson_2 due 02/04/2023 at 11:59pm CST

Problem 1. **(1 point)**

Important Note: Since we have a numeric response variable, we will be using the *knn.reg* function from the **FNN** package. Thus, please use

```
library(FNN)
```

for any *k*-nearest-neighbors predictions, throughout this assignment.

We will be continuing (from Webwork Lesson 1) to analyze a data set about neighborhoods in Boston. We will begin by using these data to fit the model to the full data set.

a. In Webwork Lesson 1, you opened the **Boston** data set in the library **MASS** and standardized the two predictor variables. For numerical accuracy, we repeat that process here:

- standardize the original variables *age* and *rad* based on the mean and standard deviation of these variables in the whole data set.
- Create a data frame called *BostonStd*, containing *age*, *rad*, their standardized versions, and the (unstandardized) *crim* variable.
- Include column names on your data frame.
- Your matrix should have 506 rows and 5 columns.
- You may use either **base**-R or **dplyr**.

- A. Show me how
- B. OK, got it!

Hint:

```
library(dplyr)
library(MASS)
BostonStd <- Boston %>%
  mutate(age.std = scale(age),rad.std = scale(rad)) %>%
  select(age,rad,age.std,rad.std,crim)
```

Note: (if working in a .Rmd) in the last line, you will need to specify

```
  dplyr::select(age,rad,age.std,rad.std,crim)
```

to identify which package the *select* function is coming from.

Solution:

```
library(dplyr)
library(MASS)
BostonStd <- Boston %>%
  mutate(age.std = scale(age),rad.std = scale(rad)) %>%
  select(age,rad,age.std,rad.std,crim)
```

b. We want to fit the model for future use in making predictions. How many data points are going to be used to fit the model?

data points for fitting = ___

Hint: The full data set can be used to fit the model (thus not losing any information and getting the most accurate model).

```
dim(Boston)
```

Later, model assessment will split the data multiple times, to proceed with cross validation.

Solution: The full data set can be used to fit the model (thus not losing any information and getting the most accurate model).

```
dim(Boston)
```

Later, model assessment will split the data multiple times, to proceed with cross validation.

c. Use *BostonStd* to define a matrix *x.std* containing the variables *age.std* and *rad.std*, and define *y* to be the variable *crim*.

- A. Show me how
- B. OK, got it!

Hint:

```
x.std <- BostonStd %>%
  select(age.std,rad.std)
y <- BostonStd$crim
```

Note: (if working in a .Rmd) in the last line, you will need to specify

```
  dplyr::select(....)
```

to identify which package the *select* function is coming from.

Solution:

```
x.std <- BostonStd %>%
  select(age.std,rad.std)
y <- BostonStd$crim
```

d. Fit the 25-nearest-neighbors model on the full data set, using the two variables in *x.std* (*age.std* and *rad.std*) to predict *crim*. Compute MSE (using predicted values for the full data set) and report that value (out to 5 decimal places) here:

MSE = ___

Hint:

```
library(FNN)
predictions = knn.reg(x.std, x.std, y, k = 25)
mean( (y - predictions$pred)^2 )
```

Solution:

```
library(FNN)
predictions = knn.reg(x.std, x.std, y, k = 25)
mean( (y - predictions$pred)^2 )
```

Correct Answers:

- B
- 506
- B
- 37.65

Problem 2. **(2 points)**

We will now use the standardized Boston data to assess the model (via cross-validation). Two measures for model assessment are the mean-squared error, MSE, for all the data and leave-one-out cross-validated measure $CV_{(n)}$.

***IMPORTANT NOTE for any model requiring data standardization**: Since the entire model-fitting process includes both standardizing the data and fitting the model to the data, every time we split the data, we must standardize *both the training and the test set* according to the training data, for **each** split.

a. Which model assessment measure do you expect to have the lower value?

- A. MSE
- B. '$CV_{(n)}$'

Hint: For model assessment, recall that the MSE (as in problem 1) is computed based on predictions that reuse the same data to assess the model as were used to fit the model, while cross-validation is computed based on predicting truly new data within each split.

Solution: For model assessment, recall that the MSE (as in problem 1) is computed based on predictions that reuse the same data to assess the model as were used to fit the model, while cross-validation is computed based on predicting truly new data within each split.

b. Which model assessment measure do you expect to be a more honest measure of the model's predictive abilities?

- A. MSE
- B. '$CV_{(n)}$'

Hint: For model assessment, recall that the MSE (as in problem 1) is computed based on predictions that reuse the same data to assess the model as were used to fit the model, while cross-validation is computed based on predicting truly new data within each split.

Solution: For model assessment, recall that the MSE (as in problem 1) is computed based on predictions that reuse the same data to assess the model as were used to fit the model, while cross-validation is computed based on predicting truly new data within each split.

c. Start with the original variables:

```
x <- BostonStd %>% select(age,rad)
```

Note: (if working in a .Rmd) you will need to specify

```
x <- BostonStd %>% dplyr::select(age,rad)
```

to identify which package the *select* function is coming from.

Compute the leave-one-out cross-validated (LOOCV) measure, $CV_{(506)}$, for 25-nearest-neighbors model, reporting that value (out to 5 decimal places) here: (**remember to standardize the training and test sets *within each fold***)

$CV_{(506)}$ = _____

Hint:

```
x <- BostonStd %>% select(age,rad)
n=dim(BostonStd)[1]
LOOCVpredictions = rep(NA,n)
cvgroups = 1:n; nfolds=n
for (ii in 1:  nfolds) {    # ii is an easier string to search fo
  groupii = (cvgroups == ii)     # logical vector for group ii
  train.x = x[!groupii,]
  train.x.std = scale(train.x)
  train.y = y[!groupii]
  test.x = x[groupii,]
  test.x.std = scale(test.x,
                     center = attr(train.x.std, "scaled:center"),
                     scale = attr(train.x.std, "scaled:scale"))
  predictions = knn.reg(train.x.std, test.x.std, train.y, k = 25)
  LOOCVpredictions[groupii] = predictions$pred
}
mean( (y - LOOCVpredictions)^2 )
```

Solution:

```
x <- BostonStd %>% select(age,rad)
n=dim(BostonStd)[1]
LOOCVpredictions = rep(NA,n)
cvgroups = 1:n; nfolds=n
for (ii in 1:  nfolds) {    # ii is an easier string to search fo
  groupii = (cvgroups == ii)     # logical vector for group ii
  train.x = x[!groupii,]
  train.x.std = scale(train.x)
  train.y = y[!groupii]
  test.x = x[groupii,]
  test.x.std = scale(test.x,
                     center = attr(train.x.std, "scaled:center"),
                     scale = attr(train.x.std, "scaled:scale"))
  predictions = knn.reg(train.x.std, test.x.std, train.y, k = 25)
  LOOCVpredictions[groupii] = predictions$pred
}
mean( (y - LOOCVpredictions)^2 )
```

d. When doing the LOO computation, how many different model fits are we performing?

- A. 5060
- B. 506
- C. 1
- D. 10
- E. 12650

Hint: Since we are producing a different fit, cycling through and leaving out each observation as a test set, we are fitting the model *n* times. But we are only computing the LOOCV for one such model (the 25-nearest-neighbors model).

Solution: Since we are producing a different fit, cycling through and leaving out each observation as a test set, we are fitting the model *n* times. But we are only computing the LOOCV for one such model (the 25-nearest-neighbors model).

Correct Answers:

- A

- B
- 40.69515
- B

Problem 3. **(2 points)**

A third measure for model assessment is m-fold** cross-validated measure $CV_{(m)}$. We will use 10-fold cross-validation to compute the value of the measure $CV_{(10)}$.

***IMPORTANT NOTE for any model requiring data standardization**: Since the entire model-fitting process includes both standardizing the data and fitting the model to the data, every time we split the data, we must standardize *both the training and the test set* according to the training data, for **each** fold.

a. Along with

```
set.seed(2)
```

use the sample function to come up with cvgroups, containing the labels for a 10-fold split of the data.

- A. Show me how
- B. OK, got it!

Hint:

```
nfolds=10; n = dim(Boston)[1]
groups = rep(1:nfolds,length=n)
set.seed(2)
cvgroups = sample(groups,n)
```

Solution:

```
nfolds=10; n = dim(Boston)[1]
groups = rep(1:nfolds,length=n)
set.seed(2)
cvgroups = sample(groups,n)
```

b. Compute the 10-fold cross-validated measure, $CV_{(10)}$, for 25-nearest-neighbors model, reporting that value (out to 5 decimal places) here: (**remember to standardize the training and test sets *within each fold***, according to the training mean and standard deviation)

$CV_{(10)}$ = _____

Hint:

```
tenfoldCVpredictions = rep(NA,n)
for (ii in 1:  nfolds) {    # ii is an easier string to search fo
  groupii = (cvgroups == ii)     # logical vector for group ii
  train.x = x[!groupii,]
  train.x.std = scale(train.x)
  train.y = y[!groupii]
  test.x = x[groupii,]
  test.x.std = scale(test.x,
                     center = attr(train.x.std, "scaled:center"),
                     scale = attr(train.x.std, "scaled:scale"))
  predictions = knn.reg(train.x.std, test.x.std, train.y, k = 25)
  tenfoldCVpredictions[groupii] = predictions$pred
}
mean( (y - tenfoldCVpredictions)^2 )
```

Solution:

```
tenfoldCVpredictions = rep(NA,n)
for (ii in 1:  nfolds) {    # ii is an easier string to search fo
  groupii = (cvgroups == ii)     # logical vector for group ii
```

```
  train.x = x[!groupii,]
  train.x.std = scale(train.x)
  train.y = y[!groupii]
  test.x = x[groupii,]
  test.x.std = scale(test.x,
                     center = attr(train.x.std, "scaled:center"),
                     scale = attr(train.x.std, "scaled:scale"))
  predictions = knn.reg(train.x.std, test.x.std, train.y, k = 25)
  tenfoldCVpredictions[groupii] = predictions$pred
}
mean( (y - tenfoldCVpredictions)^2 )
```

c. Identify the correct choice and reason for selecting between LOOCV and *m*-fold CV for model assessment.

- A. We should use LOOCV, since it results in a smaller value of the CV measure.
- B. We prefer to use LOOCV, since it results in a less variable estimate of error than 10-fold cross-validation.
- C. We use 10-fold cross-validation, as a good compromise for bias-variance trade-off.

Hint: There is a subtle point - since LOOCV and m-fold CV measure the same model (just using slightly different processes), we cannot compare their values directly.

Generally, LOOCV results in less biased estimate of error (since each training set is almost size n), but more variable (since each test set is only size 1).

Solution: There is a subtle point - since LOOCV and m-fold CV measure the same model (just using slightly different processes), we cannot compare their values directly.

Generally, LOOCV results in less biased estimate of error (since each training set is almost size n), but more variable (since each test set is only size 1).

d. To find a more optimal model, what else might you suggest?

- A. Measure according to MSE rather than ‘$\mathrm{CV}_{(m)}$‘$B.Computethecross-validationmeasurefordifferentpossiblevaluesofk.$
- C. Remove half the data points, using only 253 observations.

Hint: We can adjust the complexity of the model by adjust *k*; this balances variance and bias of the model (fewer neighbors means higher variance / more complex, while more neighbors means higher bias / less complex).

Solution: We can adjust the complexity of the model by adjust *k*; this balances variance and bias of the model (fewer neighbors means higher variance / more complex, while more neighbors means higher bias / less complex).

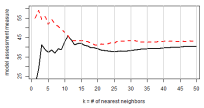
Correct Answers:

- B
- 41.18455
- C
- B

Problem 4. **(1 point)**

We now focus on the using of m-fold cross-validation for selecting between models.

a. Using the same type of loops as in problem 3, we could compute assessment measure $CV_{(10)}$ for each model fit using number of nearest neighbors, k = 1, 2, ..., 50.
Doing so results in the information displayed on the following graph.



One line represents the MSE values for using the fitted model to predict the same data originally used to fit the model; the other line represents the CV(10) values calculated via 10-fold cross-validation (which predicts truly new data). Which measure's values are represented by the black solid line?

- A. MSE
- B. '$CV_{(10)}$'

Hint: Data used to fit the model will, by necessity, be well-predicted by the model.

Solution: Data used to fit the model will, by necessity, be well-predicted by the model.

b. Using the graph from part a., choose the preferred value for the number of nearest neighbors k to make the best predictions.

- A. 5
- B. 10
- C. 20
- D. 25
- E. 50

Hint: Select the number of nearest neighbors k both to minimize $CV_{(10)}$ (the estimated mean-square-error using prediction of *truly new* data) as well as to keep the model simple.

Solution: Select the number of nearest neighbors k both to minimize $CV_{(10)}$ (the estimated mean-square-error using prediction of *truly new* data) as well as to keep the model simple.

c. In order to compute the $CV_{(10)}$ for all 50 models, how many different model fits would we need to perform?

- A. 50
- B. 25300
- C. 500
- D. 10
- E. 5060
- F. 506

Hint: Since we are producing a different fit, cycling through and leaving out each of m=10 test sets, we are fitting each model ten times. Now, we are repeating this process for *each* of 50 different

Problem 5. **(2 points)**

We now examine some regression models, using 10-fold cross-validation to compute the value of the measure $CV_{(10)}$.

***IMPORTANT NOTE regarding using cross-validation for model-selection**: Since the CV measure for numeric data is tied to the magnitude of the response, we can only compare models fit with the same response values.

Thus, we will start with two models:

Model 1. $crim = \beta_0 + \beta_1 \cdot rad$

Model 2. $crim = \beta_0 + \beta_1 \cdot rad + \beta_2 \cdot age$, and

Model 3. *crim* modeled on all variables other than *crim*

Since linear regression does not require predictors to be standardized, we will fit using data from the original **Boston** data frame.

a. Along with

```
set.seed(2)
```

use the sample function to come up with cvgroups, containing the labels for a 10-fold split of the data. (This should produce the same split of the data that you initialized in problem 3)

- A. OK, got it!
- B. Show me how

Hint:

```
nfolds=10; n = dim(Boston)[1]
groups = rep(1:nfolds,length=n)
set.seed(2)
cvgroups = sample(groups,n)
```

Solution:

```
nfolds=10; n = dim(Boston)[1]
groups = rep(1:nfolds,length=n)
set.seed(2)
cvgroups = sample(groups,n)
```

b. Compute the 10-fold cross-validated measure, $CV_{(10)}$, for Model **1**. Report that value (out to 5 decimal places) here:

$CV_{(10)}$ = _____

Hint: After defining the model (call it **ModelUsed**), apply the following:

```
tenfoldCVpredictions = rep(NA,n)
for (ii in 1:nfolds) {    # ii is an easier string to search for index
  groupii = (cvgroups == ii)     # logical vector for group ii
  trainset = Boston[!groupii,]  # all data EXCEPT for group ii
  testset = Boston[groupii, ]   # data in group ii
  modelfit = lm(ModelUsed, data=trainset) # fit to train set
  predicted = predict(modelfit, newdata = testset)   # predict for test set
  tenfoldCVpredictions[groupii] = predicted
}
mean( (y - tenfoldCVpredictions)^2 )
```

Solution:

```
ModelUsed <- (crim ~ rad)
tenfoldCVpredictions = rep(NA,n)
for (ii in 1:nfolds) {    # ii is an easier string to search for
  groupii = (cvgroups == ii)     # logical vector for group ii
  trainset = Boston[!groupii,]  # all data EXCEPT for group ii
  testset = Boston[groupii, ]   # data in group ii
  modelfit = lm(ModelUsed, data=trainset) # fit to train set
  predicted = predict(modelfit, newdata = testset)   # predict fo
  tenfoldCVpredictions[groupii] = predicted
}
mean( (y - tenfoldCVpredictions)^2 )
```

c. Compute the 10-fold cross-validated measure, $CV_{(10)}$, for Model **2**. Report that value (out to 5 decimal places) here:

$CV_{(10)}$ = _____

Hint: Start with:

```
ModelUsed <- (crim ~ rad + age)
```

and use precisely the same code for the cross-validation as in part b.

Solution:

```
ModelUsed <- (crim ~ rad + age)
tenfoldCVpredictions = rep(NA,n)
for (ii in 1:  nfolds) {    # ii is an easier string to search fo
  groupii = (cvgroups == ii)     # logical vector for group ii
  trainset = Boston[!groupii,]  # all data EXCEPT for group ii
  testset = Boston[groupii, ]   # data in group ii
  modelfit = lm(ModelUsed, data=trainset) # fit to train set
  predicted = predict(modelfit, newdata = testset)   # predict fo
  tenfoldCVpredictions[groupii] = predicted
}
mean( (y - tenfoldCVpredictions)^2 )
```

d. Compute the 10-fold cross-validated measure, $CV_{(10)}$, for Model **3**. Report that value (out to 5 decimal places) here:

$CV_{(10)}$ = _____

Hint: Start with:

```
ModelUsed <- (crim ~ .)
```

and use precisely the same code for the cross-validation as in part b.

Solution:

```
ModelUsed <- (crim ~ .)
tenfoldCVpredictions = rep(NA,n)
for (ii in 1:  nfolds) {    # ii is an easier string to search fo
  groupii = (cvgroups == ii)     # logical vector for group ii
  trainset = Boston[!groupii,]  # all data EXCEPT for group ii
  testset = Boston[groupii, ]   # data in group ii
  modelfit = lm(ModelUsed, data=trainset) # fit to train set
  predicted = predict(modelfit, newdata = testset)   # predict fo
  tenfoldCVpredictions[groupii] = predicted
}
mean( (y - tenfoldCVpredictions)^2 )
```

e. We now have $CV_{(10)}$ measures for each of the three regression models, using *crim* as the response.

Which regression model is preferred, according to the $CV_{(10)}$ criterion?

- A. Model 2
- B. Model 1
- C. Model 3

Hint: Pick the model with the lowest $CV_{(10)}$ measure.

Solution: Model **3** has the lowest $CV_{(10)}$ measure.

f. Surprisingly, linear regression Model **3** performs *worse* (according to the $CV_{(10)}$ criterion) than the 25-nearest-neighbors model. Select appropriate reasoning for this performance.

- A. The CV measure computed for the linear regression Models uses fewer data points.
- B. The linear regression models are more flexible than the 25-nearest-neighbors model.
- C. The fit from the linear regression Model 3 has a few extremely large residuals (due to a very-skewed response variable).

Hint: Look at the following:

```
hist(Boston$crim)
```

or

```
modelfit = lm(crim ~ ., data=Boston)
plot(modelfit)
```

Solution: Look at the following:

```
hist(Boston$crim)
```

or

```
modelfit = lm(crim ~ ., data=Boston)
plot(modelfit)
```

Note the very large residual values in the plot of residuals versus fitted values.

Note: We will work with the log-ransformed response in the next problem. You may find it informative to look at the following plots about the usage of the log-transform:

```
hist(log(Boston$crim))
```

and / or

```
logmodelfit = lm(log(crim) ~ ., data=Boston)
plot(logmodelfit)
```

Correct Answers:

- A
- 45.29722
- 44.9109
- 43.69566
- C
- C

Problem 6. **(2 points)**

We will continue using the data in from **Boston**. We will be using these data to fit a multiple linear regression model to an appropriately-transformed response and to compute standard errors via bootstrapping.

a. Define a new data frame, **BostonT**, that contains the variables *age*, *rad*, and *log.crim* (the natural log transformation of the variable crim). Note that *log* function refers to the natural log transformation.

- A. Show me how
- B. OK, got it!

Hint:

```
BostonT <- Boston %>%
  mutate(log.crim = log(crim)) %>%
  select(age,rad,log.crim)
```

Note: (if working in a .Rmd) in the last line, you will need to specify

```
  dplyr::select(....)
```

to identify which package the *select* function is coming from.

Solution:

```
BostonT <- Boston %>%
  mutate(log.crim = log(crim)) %>%
  select(age,rad,log.crim)
```

b. Fit the multiple linear regression model of *log.crim* on the predictors *age* and *rad*. Produce a summary of the regression model, and report the standard errors (out to 6 decimal places) of the coefficients:

SE(intercept) = ____

SE(coefficient for age) = ____

SE(coefficient for rad) = ____

Hint:

```
lmfit = lm(log.crim ~ ., data = BostonT)
summary(lmfit)  # see column Std.  Error
```

Solution:

```
lmfit = lm(log.crim ~ ., data = BostonT)
summary(lmfit)  # see column Std.  Error
```

c. Define a function called *beta.fn* that takes *inputdata* (the data set to use for model-fitting) and *index* (of which data points to use) as inputs and returns the coefficients fit for the multiple linear regression model of *log.crim* on predictors *age* and *rad*.

- A. Show me how
- B. OK, got it!

Hint:

```
beta.fn = function(inputdata,index) {
  lmfitboot = lm(formula = log.crim ~., data= inputdata[index,])
  return(lmfitboot$coef)
}
beta.fn(BostonT,1:n)  # outputs coefficients of model fit on full
```

Solution:

```
beta.fn = function(inputdata,index) {
  lmfitboot = lm(formula = log.crim ~., data= inputdata[index,])
  return(lmfitboot$coef)
}
beta.fn(BostonT,1:n)  # outputs coefficients of model fit on full
```

d. Use the *boot* function from the library **boot** to compute standard error based on bootstrap sample estimates of the coefficients. Set the seed to be 2, and use 5000 bootstrap samples. Report the bootstrapped standard errors (rounded to 6 decimal places):

SE_B(intercept) = ____

SE_B(coefficient for age) = ____

SE_B(coefficient for rad) = ____

Hint:

```
library(boot)
set.seed(2)
bootoutput = boot(BostonT,beta.fn,R=5000)
print(bootoutput)

# standard error as estimated via simulation
round(sd((bootoutput$t)[,1]),6); hist((bootoutput$t)[,1])

round(sd((bootoutput$t)[,2]),6); hist((bootoutput$t)[,2])

round(sd((bootoutput$t)[,3]),6); hist((bootoutput$t)[,3])
```

Solution:

```
library(boot)
set.seed(2)
bootoutput = boot(BostonT,beta.fn,R=5000)
print(bootoutput)

# standard error as estimated via simulation
round(sd((bootoutput$t)[,1]),6); hist((bootoutput$t)[,1])

round(sd((bootoutput$t)[,2]),6); hist((bootoutput$t)[,2])

round(sd((bootoutput$t)[,3]),6); hist((bootoutput$t)[,3])
```

Correct Answers:

- B
- 0.10777
- 0.001634
- 0.005282
- B
- 0.111243
- 0.001794
- 0.004872