Homework 1

Advanced Methods for Data Analysis (36-402)

Due Friday January 28, 2022 **at 3:00pm ET**

Solutions – not to be posted online or shared, even after the end of the semester.

You should **always show all your work** and submit both a writeup and *R* code.

- Assignments must be submitted through Gradescope as a PDF. Follow the instructions here: `https://www.cmu.edu/teaching/gradescope/`
- Gradescope will ask you to mark which parts of your submission correspond to each homework problem. This is mandatory; if you do not, grading will be slowed down, and your assignment will be penalized.
- Make sure your work is legible in Gradescope. You may not receive credit for work the TAs cannot read. **Note:** If you submit a PDF with pages much larger than 8.5×11", they will be blurry and unreadable in Gradescope.
- For questions involving R code, we strongly recommend using R Markdown. The relevant code should be included with each question, rather than in an appendix. A template Rmd file is provided on Canvas.

1. **Kernel regression.** Recall that we discussed kernel regression in Lecture 1. In this problem you will use a different data set. The data come from a sample of automobiles and consist of pairs of engine displacement and miles-per-gallon. The data are in the file `engine.Rdata` on the Canvas website. It is going to be helpful for you to look over the code `Demo1_intro.R` that we used in Lecture 1 carefully.

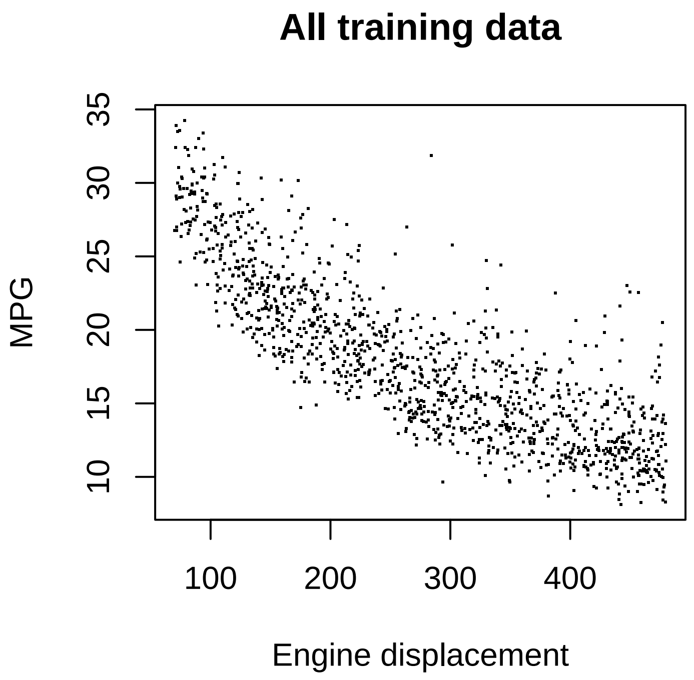
(4) (a) Download the file `engine.Rdata` from Canvas, and load it into your *R* session with `load("engine.Rdata")`. You can type `ls()` to see the R objects that have been loaded into memory. These will include four data matrices similar to the ones used in R Demo 1 plus a vector `x0` which has 411 equally-spaced values that cover the range of all of the observed predictor values. These should be used along the horizontal axis for plotting the fitted kernel regressions to reduce the effects of the gaps between the observed predictor values.

The matrices `engine.xtrain` and `engine.ytrain` are each 32×40, containing 40 training data sets each of 32 x or y points along its 32 columns. That is, the first column of `engine.xtrain` and the first column of `engine.ytrain` together make up a training data set of 32 (x, y) pairs. The x variable is engine displacement, and the y variable is miles-per-gallon.

Hence, amassing the data sets together, there are $40 \times 32 = 1280$ x points and 1280 y points *in total.* Plot these 1280 y points versus the corresponding 1280 x points, on a single plot, to get an idea of the trend, if any. (Hint: there is an easy way to do this with a single call to the `plot` function.) Is the trend linear?

Solution: First, we need to load the data and make the `np` library available. Next, we plot the combined training data. We can do this by converting the two matrices into columns. Put a `c()` around the name of each matrix to convert it into a column. Here is the *R* code:

```r
##############################################################
# Part a: Load and plot data
##############################################################
load("engine.Rdata")
suppressPackageStartupMessages(library(np))

# Plot all of the training data
plot(c(engine.xtrain), c(engine.ytrain), pch=".", xlab="Engine displacement",
     ylab = "MPG", main = "All training data")
```



This does not look linear.

(12) (b) For the next part, we will restrict our attention to just the first training set, i.e., the first columns of `engine.xtrain` and `engine.ytrain`. There is a standard *R* function that fits normal-kernel (Nadaraya-Watson) regression, but it has some serious issues. Instead, we will use the function `npreg`. To use this function you must first install the `np` package and then bring it into your workspace with `library(np)`.

As with `lm`, `npreg` requires that you fit your kernel regression using data from a `data.frame`, so you will need to put `engine.xtrain[,1]` and `engine.ytrain[,1]` into a `data.frame` first. If you call that data.frame `first`, with the two columns called `x` and `y`, the `npreg` syntax for fitting with bandwidth `h` is

```r
kregobj <- npreg(y ~ x, data=first, bws=h)
```

The output of `npreg` can then be used to create predictions. If you want predictions at the equally spaced points `x0`, use

```r
predict(kregobj, newdata = data.frame(x = x0))
```

If you want fitted values at the predictor values in `xtest[,1]`, use that instead of `x0` above. The normal kernel is the default.

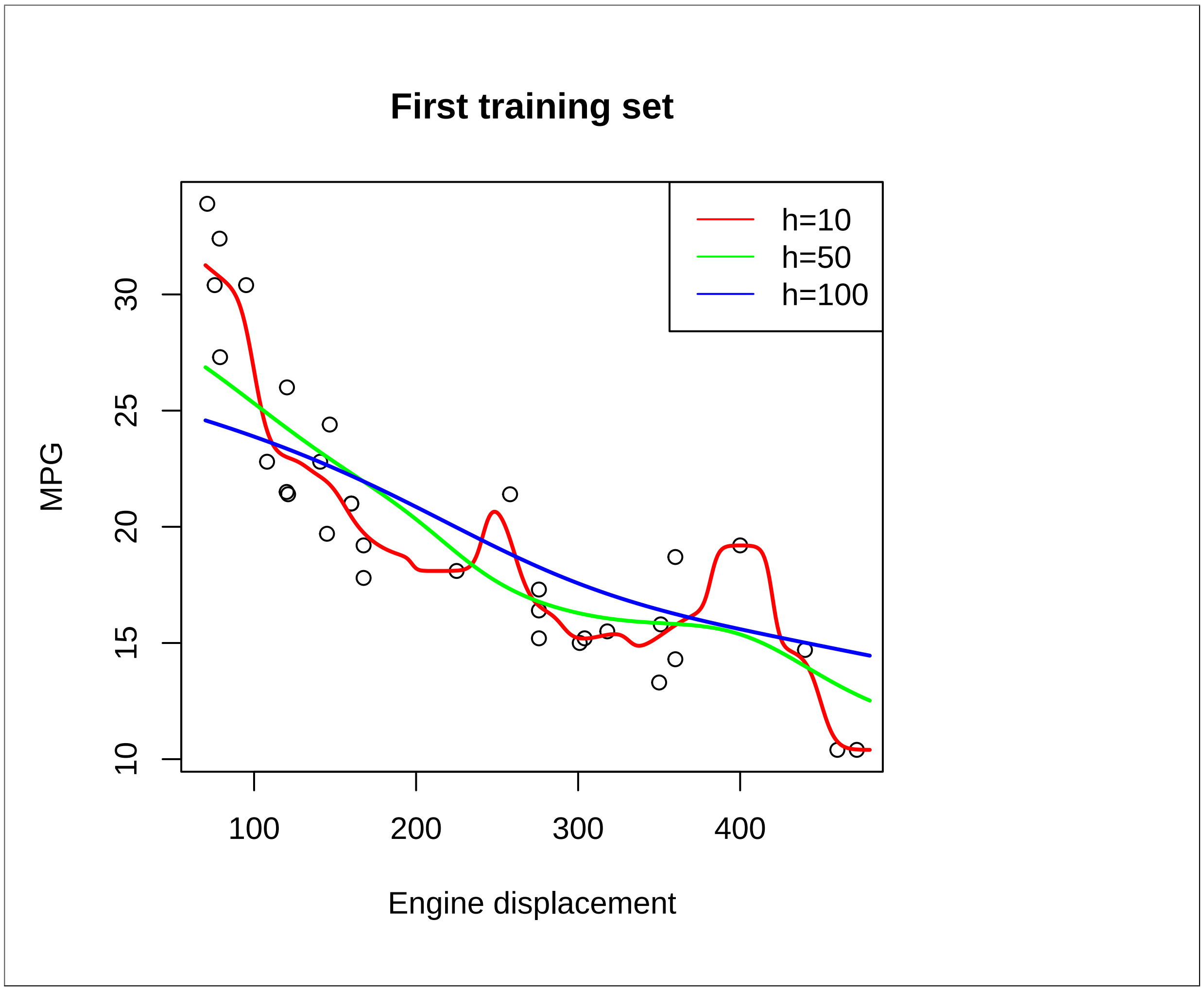
Fit kernel regressions with the normal kernel to the first set of training points, with 3 different values of the bandwidth parameter: 10, 50, and 100. For each bandwidth value, plot the estimated regression function from kernel regression as a line on top of the training points. To facilitate comparison, compute these lines using `x=x0` in the `predict` function. To be specific, if your command to compute the regression estimate is as stated earlier, then

`lines(x0, predict(kregobj, newdata = data.frame(x = x0)))`

will be a good way to *start* the appropriate command to add a line to the plot. Make sure that a reader can tell which line is which.

Solution: Since the `predict` function is expecting that the kernel regression was performed on data in `data.frame`, we put the first test set into a `data.frame`. Then we plot the first test set, run the three kernel regressions with the three bandwidths, and add the three curves. A `legend` should make it easy to tell the curves apart. Here is the *R* code:

```r
############################################################
# Part b: Fit first training set
############################################################

first <- data.frame(x = engine.xtrain[,1], y = engine.ytrain[,1])
par(mfrow=c(1,1))
x <- first$x
y <- first$y
plot(x, y, xlab="Engine displacement", ylab="MPG",
     main = "First training set")
h <- c(10, 50, 100)

# Fit the kernel regression with three bandwidths

smoothm1 <- npreg(y ~ x, data=first, bws=h[1])
smoothm2 <- npreg(y ~ x, data=first, bws=h[2])
smoothm3 <- npreg(y ~ x, data=first, bws=h[3])

# Plot curve (component 'y' in the output of npreg)

lines(x0, predict(smoothm1, newdata=data.frame(x=x0)),
      col="red", lwd=2)
lines(x0, predict(smoothm2, newdata=data.frame(x=x0)),
      col="green", lwd=2)
lines(x0, predict(smoothm3, newdata=data.frame(x=x0)),
      col="blue", lwd=2)

legend("topright", col=c("red", "green", "blue"), lty=c(1,1,1),
       legend=c("h=10", "h=50", "h=100"))
```



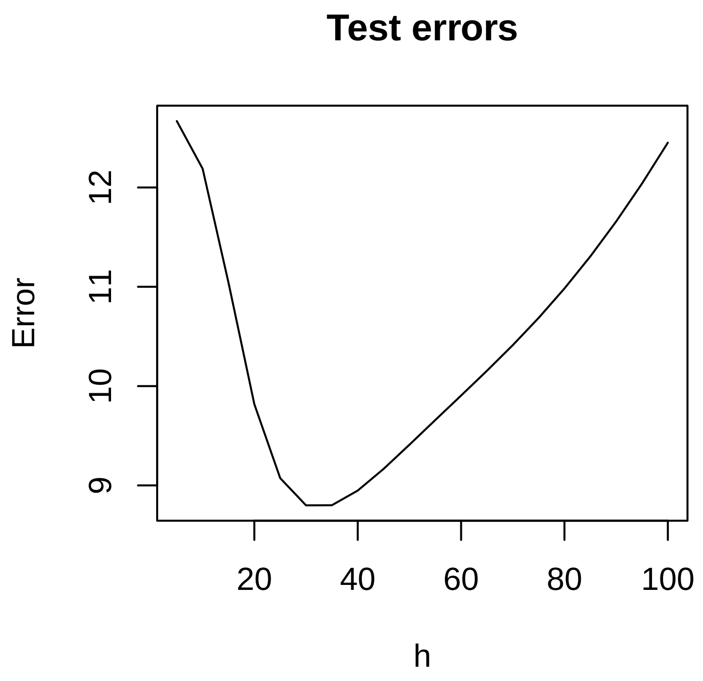
(10) (c) Describe in words what happens to the kernel regression fit as we vary the bandwidth parameter. What procedure does this remind you of (that we have already seen)? What happens as we increase the bandwidth parameter? Again, what procedure does this remind you of?

Solution: As the bandwidth decreases, the kernel regression starts overfitting by tracking the observations very closely. As the bandwidth increases, the kernel regression function begins to underfit, becoming nearly a straight line. This should be reminiscent of k-nearest neighbors from the first Demo (overfitting for small k, underfitting for large k).

(12) (d) Using the same training set, i.e., the first columns of `engine.xtrain` and `engine.ytrain`, we are going to investigate the predictive performance on the first test set, i.e., the first columns of `engine.xtest` and `engine.ytest`. For a set of 20 bandwidth values, equally spaced from 5 to 100 (5 apart), fit a kernel regression to the training points and predict the regression function at the test x points (not `x0`.) Evaluate its test error, measured in terms of average squared distance between the predicted values and the test y values. Hence, you will have a curve of 20 test errors. Plot this test error curve as a function of the underlying bandwidth values.

Solution: To get 20 points equally spaced from 5 to 100, we can use the `seq` function with `by=5`. We store the values into a vector and create a vector `errtest.ker` in which to store the test errors. Then we run a loop that fits the kernel regression for each bandwidth and stores the test error. At the end, `errtest.ker` has 20 average squared test error numbers, one for each bandwidth. Finally, we plot `errtest.ker` versus the bandwidth. Here is the *R* code:

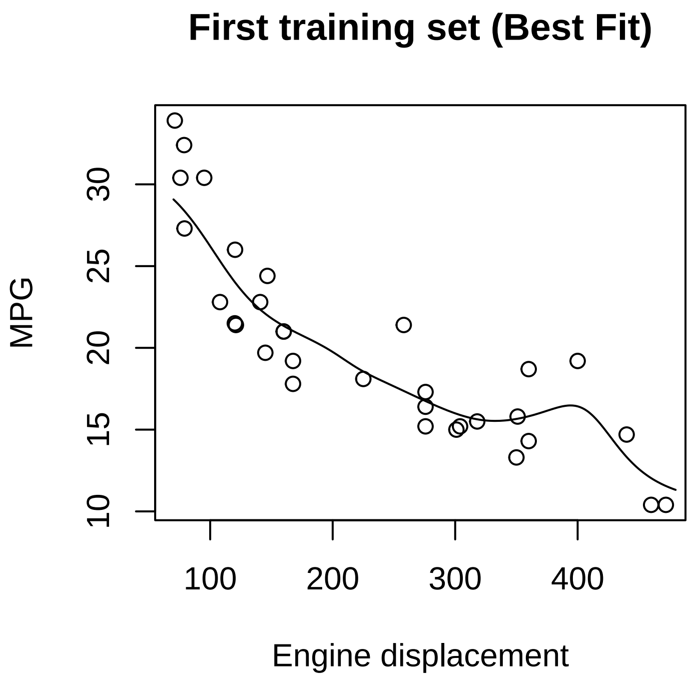
```
##########################################################
# Part d: Test errors as a function of h for the first data set.
##########################################################

# Let h=5,...,100 in steps of 5
hs <- seq(5, 100, by = 5)
nh <- length(hs)
errtest.ker <- numeric(nh)
for (i in 1:nh) {
  smoothm <- npreg(y ~ x, data=first, bws=hs[i])

  errtest.ker[i] <- mean(
  (engine.ytest[,1] - predict(smoothm, newdata=data.frame(x=engine.xtest[,1])))^2)
}

# Plot errors as function of h

plot(hs, errtest.ker, type="l", main="Test errors",
     xlab="h", ylab="Error")
```



(10) (e) According to this test error curve, what is the optimal bandwidth value? What is its associated test error? Plot the kernel regression fit, on top of the training points, using this optimal bandwidth value (and using `x0` on the horizontal axis.) Looking at the plot, does your eye agree that this is really a good bandwidth value? Why or why not?

Solution: We use the `which.min` function to see which element of `errtest.ker` has the smallest value. We then print the corresponding bandwidth and the test error. Here is the *R* code:

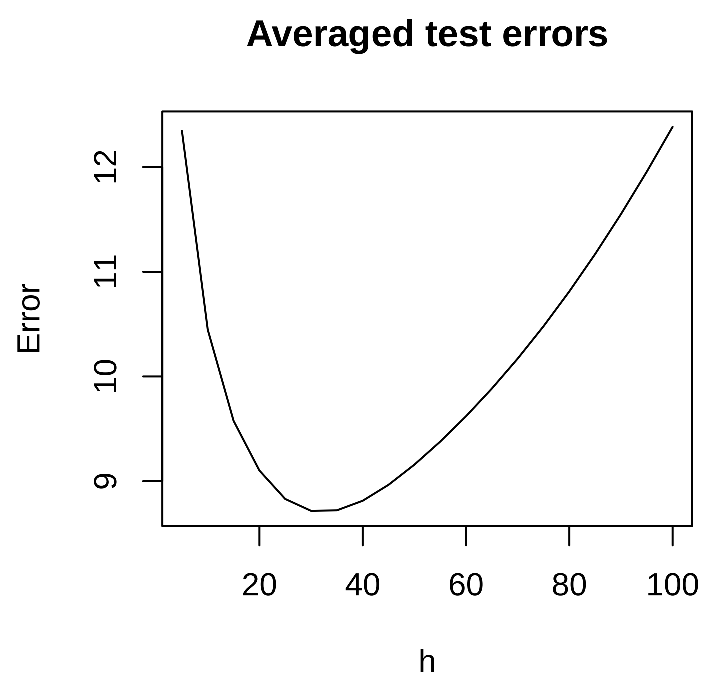
```
############################################################
# Part e: Find "optimal" bandwidth and plot data and fit
############################################################

hopt <- which.min(errtest.ker)

print(hs[hopt])
## [1] 30
print(errtest.ker[hopt])
## [1] 8.799357
#  Plot the fitted kernel regression with the "optimal" bandwidth over the data

plot(x, y, xlab="Engine displacement", ylab="MPG",
  main="First training set (Best Fit)")

smoothm = npreg(y ~ x, data=first, bws=hs[hopt])

lines(x0, predict(smoothm, newdata=data.frame(x=x0)))
```



(12) (f) Now repeat part (d), but do the same for each of the 40 training and test data sets in turn, and report the *average* test error curve at each bandwidth value over the 40 sets. Plot the average test error curve with respect to the bandwidth values. What do you see now? Has the optimal value of the bandwidth changed, and has its associated test error changed?

Solution: We repeat the loop from part (d) inside of another loop over the 40 training/test sets. Before doing that, we create the matrix `errtest.ker.all` to store all of the test errors. After finishing the outer loop, we find the average test error over the 40 data sets separately for each of the 20 bandwidths. In analogous manner as in part (d), we plot the average test error against bandwidth. Here is the *R* code:

```
############################################################
# Part f: Compute average test errors over the test data sets
############################################################

# Repeat for all 40 training/test sets.
p <- 40
errtest.ker.all <- matrix(0, nh, p)
for (j in 1:p) {
  x <- engine.xtrain[,j]
  y <- engine.ytrain[,j]
  alld <- data.frame(x=x, y=y)

  for (i in 1:nh) {
    smoothm <- npreg(y ~ x, data=alld, bws=hs[i])

    errtest.ker.all[i,j] <- mean((engine.ytest[,j] -
                             predict(smoothm, newdata = data.frame(x=engine.xtest[,j]
  }
}

# Average test errors over all 40 simulations (columns)

errtest.ker.ave <- rowMeans(errtest.ker.all)

# Plot results as a function of h

plot(hs, errtest.ker.ave, type="l",
     main="Averaged test errors", xlab="h", ylab="Error")
```



```r
theh <- which.min(errtest.ker.ave)

# Find the best bandwidth

print(hs[theh])
## [1] 30
print(errtest.ker.ave[theh])
## [1] 8.716932
```

2. **Optimism of the training error in linear regression.** We are given training data (X_i, Y_i), $i = 1, \dots, n$, and test data (X'_j, Y'_j), $j = 1, \dots, m$ which we will assume are *all* i.i.d., i.e., these observations are all independent and come from the same distribution.

   Assuming that the predictors are p-dimensional, let $\hat{\beta} \in \mathbb{R}^p$ denote estimated linear regression coefficients based on the training data,
   $$\hat{\beta} = (\mathbb{X}^\top \mathbb{X})^{-1} \mathbb{X}^\top Y,$$
   where $\mathbb{X}$ is the $n \times p$ matrix with ith row equal to X_i, and Y is an n-dimensional vector with ith component Y_i. (There's no intercept in this regression; instead, we can make the first column of $\mathbb{X}$ all ones.) We are going to prove that
   $$\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{\beta}^\top X_i)^2\right] \le \mathbb{E}\left[\frac{1}{m}\sum_{j=1}^{m}(Y'_j - \hat{\beta}^\top X'_j)^2\right],$$
   where the expectations above are over all that is random, i.e., over the training set on the left hand side, and over both the training and testing sets on the right hand side. In words, we're proving that the expected training error is always less than or equal to the expected testing error (without many assumptions at all on the true model), meaning that the training error is naïvely optimistic.

(12) (a) Argue that the expected test error is the same whether we have m test points or just 1 test point, i.e.,

$$\mathbb{E}\left[\frac{1}{m}\sum_{i=1}^{m}(Y_i' - \hat{\beta}^\top X_i')^2\right] = \mathbb{E}\left[(Y_1' - \hat{\beta}^\top X_1')^2\right].$$

Hence argue that indeed it doesn't matter whether we have m test points or n test points,

$$\mathbb{E}\left[\frac{1}{m}\sum_{i=1}^{m}(Y_i' - \hat{\beta}^\top X_i')^2\right] = \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}(Y_i' - \hat{\beta}^\top X_i')^2\right],$$

and so we may assume without a loss of generality that $m = n$ (the testing and training sets have the same number of samples).

(*Hint:* For each $j = 1, \ldots, m$, think about the conditional mean of $(Y_j' - \hat{\beta}^\top X_j')^2$ given the training data $(X_1, Y_1), \ldots, (X_n, Y_n)$. What does this tell you about $\mathbb{E}[(Y_j' - \hat{\beta}^\top X_j')^2]$ for each j?)

Solution: To prove this result, we make use of the following lemmas. Convince yourself that at least the first two lemmas are conformal with our intuition.

Lemma 1. *Let X_1 and X_2 be two independent random variables ($X_1 \perp\!\!\!\perp X_2$) and define $Y_1 := g(X_1)$ and $Y_2 := g(X_2)$. Then, $Y_1 \perp\!\!\!\perp Y_2$.*

Lemma 2 (Linearity of Expectation)**.** *For random variables $X_1, \ldots, X_n$ and constants $c_1, \ldots, c_n$, it holds that $E\left(\sum_i c_i X_i\right) = \sum_i c_i \mathbb{E}\left(X_i\right)$.*

Lemma 3 (Tower rule)**.** *Let X be a random variable with expected value, $\mathbb{E}(X)$, well defined. Let Y be any random variable on the same probability space. Then $\mathbb{E}(X) = \mathbb{E}(\mathbb{E}(X|Y))$.*

Notice that we can express $\hat{\beta}$ as a function of the training observations $(Y_{1:n}, \mathbf{X}_{1:n})$: $\hat{\beta} = g(Y_{1:n}, \mathbf{X}_{1:n})$. So, by Lemma 1, $\hat{\beta} \perp\!\!\!\perp (Y_j^{'}, \mathbf{X}_j^{'})$ for any observation j in the test set. Let f be the density of $(Y^{'}, \mathbf{X}^{'})$ (it must be the same function for any j because obs are iid), then

$$f\left(Y_j^{'}, \mathbf{X}_j^{'} \;\middle|\; \hat{\beta}\right) = f\left(Y_j^{'}, \mathbf{X}_j^{'}\right) = f\left(Y_1^{'}, \mathbf{X}_1^{'}\right)$$

where the first equality follows by independence and the second equality follows because observations are iid. In particular,

$$\begin{aligned}
\mathbb{E}\left[\left(Y_j^{'} - \hat{\beta}^T \mathbf{X}_j^{'}\right)^2 \middle| \hat{\beta} = b\right] &= \int \left(y_j^{'} - \hat{\beta}^T \mathbf{x}_j^{'}\right)^2 f\left(y_j^{'}, \mathbf{x}_j^{'} \;\middle|\; \hat{\beta}\right) d(y_j^{'}, \mathbf{x}_j^{'}) \\
&= \int \left(y_1^{'} - \hat{\beta}^T \mathbf{x}_1^{'}\right)^2 f\left(y_1^{'}, \mathbf{x}_1^{'}\right) d(y_1^{'}, \mathbf{x}_1^{'}) \\
&= \int \left(y_1^{'} - \hat{\beta}^T \mathbf{x}_1^{'}\right)^2 f\left(y_1^{'}, \mathbf{x}_1^{'} \;\middle|\; \hat{\beta}\right) d(y_1^{'}, \mathbf{x}_1^{'}) \\
&= \mathbb{E}\left[\left(Y_1^{'} - \hat{\beta}^T \mathbf{X}_1^{'}\right)^2 \middle| \hat{\beta} = b\right]
\end{aligned}$$

Now we conclude that

$$\begin{aligned}
\mathbb{E}\left(\frac{1}{m}\sum_{j=1}^{m}\left(Y_j^{'}-\hat{\beta}^T\mathbf{X}_j^{'}\right)^2\right) &= \frac{1}{m}\sum_{j=1}^{m}\mathbb{E}\left(\left(Y_j^{'}-\hat{\beta}^T\mathbf{X}_j^{'}\right)^2\right)\\
&= \frac{1}{m}\sum_{j=1}^{m}\mathbb{E}\left[\mathbb{E}\left(\left(Y_j^{'}-\hat{\beta}^T\mathbf{X}_j^{'}\right)^2\Big|\hat{\beta}\right)\right]\\
&= \frac{1}{m}\sum_{j=1}^{m}\mathbb{E}\left[\mathbb{E}\left(\left(Y_1^{'}-\hat{\beta}^T\mathbf{X}_1^{'}\right)^2\Big|\hat{\beta}\right)\right]\\
&= \mathbb{E}\left[\left(Y_1^{'}-\hat{\beta}^T\mathbf{X}_1^{'}\right)^2\right]
\end{aligned}$$

where the first equality follows by Lemma 2, the second by Lemma 3 and the third one by the result above.

(12) (b) Now it is our task to compare the sizes of $\mathbb{E}[\frac{1}{n}\sum_{i=1}^{n}(Y_i-\hat{\beta}^\top X_i)^2]$ and $\mathbb{E}[\frac{1}{n}\sum_{i=1}^{n}(Y_i'-\hat{\beta}^\top X_i')^2]$. First consider the random variables

$$A=\frac{1}{n}\sum_{i=1}^{n}(Y_i-\hat{\beta}^\top X_i)^2 \quad\text{and}\quad B=\frac{1}{n}\sum_{j=1}^{n}(Y_j'-\tilde{\beta}^\top X_j')^2,$$

where $\tilde{\beta}\in\mathbb{R}^p$ denotes the estimated linear regression coefficients but fit from the *test set*. Argue that A and B have the same distribution, so $\mathbb{E}(A)=\mathbb{E}(B)$.

Solution: Since $\tilde{\beta}$ is the linear regression coefficients estimated on the test set, we know that

$$\tilde{\beta}=(\mathbb{X}'^\top\mathbb{X}')^{-1}\mathbb{X}'^\top\vec{Y}'.$$

Since the training set and the test set are iid, A and B have the same distribution.
More formally, let $f(\mathbb{X},\vec{Y})=\frac{1}{n}\sum_{i=1}^{n}(Y_i-((\mathbb{X}^\top\mathbb{X})^{-1}\mathbb{X}^\top\vec{Y})^\top X_i)^2$. We can write $A=f(\mathbb{X},\vec{Y})$ and $B=f(\mathbb{X}',\vec{Y}')$, i.e. A and B are the same deterministic function of $(\mathbb{X},\vec{Y})$ and $(\mathbb{X}',\vec{Y}')$ respectively and therefore have the same distribution because the corresponding arguments are identically distributed.

(10) (c) Argue that the random variable B defined in part (b) is always less than or equal to the observed test error,

$$B=\frac{1}{n}\sum_{i=1}^{n}(Y_i'-\tilde{\beta}^\top X_i')^2\le\frac{1}{n}\sum_{i=1}^{n}(Y_i'-\hat{\beta}^\top X_i')^2.$$

(*Hint:* don't try to plug in a formula for $\tilde{\beta}$, just recall that it is characterized by least squares ...)

Solution: We know that $\hat{\beta}$ is the value of β that minimizes the sum of squared errors $\sum_{i=1}^{n}(Y_i-\beta^\top X_i)^2$. By the same token, $\tilde{\beta}=\text{argmin}_\beta\sum_{i=1}^{n}(Y_i'-\beta^\top X_i')^2$.
Therefore,

$$B=\frac{1}{n}\sum_{i=1}^{n}(Y_i'-\tilde{\beta}^\top X_i')^2\le\frac{1}{n}\sum_{i=1}^{n}(Y_i'-\beta^\top X_i')^2,$$

for any β, in particular for $\beta=\hat{\beta}$.

(6) (d) Use the result of part (c) to conclude that

$$\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{\beta}^\top X_i)^2\right] \le \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}(Y_i' - \hat{\beta}^\top X_i')^2\right],$$

as desired.

Solution: By the above results,

$$\begin{aligned}\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{\beta}^\top X_i)^2\right] &= \mathbb{E}(A) = \mathbb{E}(B) \\ &= \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}(Y_i' - \tilde{\beta}^\top X_i')^2\right] \le \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}(Y_i' - \hat{\beta}^\top X_i')^2\right]\end{aligned}$$