STA 478 Assignment #4 Solutions

Dr. Robert Buscaglia

September 22, 2025

ISLR Chapter 4

Problem 1 (ISLR : Exercise 4)

a

Suppose that we have a set of observations, each with measurements on $p = 1$ feature, X. We assume that X is uniformly (evenly) distributed on $[0, 1]$. Associated with each observation is a response value. Suppose that we wish to predict a test observation's response using only observations that are within 10% of the range of X closest to that test observation. For instance, in order to predict the response for a test observation with $X = 0.6$, we will use observations in the range $[0.55, 0.65]$. On average, what fraction of the available observations will we use to make the prediction?

Solution.

It suffices to recognize that the length of the vector $[0.55, 0.65]$ is 0.1. It can be shown by integration that

$$\int_{0.55}^{0.65} 1 \quad dx = 0.1$$

which is the solution of the probability of finding a point on the given interval for a uniform distribution. Further thought with this problem may evaluate the edges of the distribution, but this solution will suffice for the learning objective.

b

Now suppose that we have a set of observations, each with measurements on $p = 2$ features, X_1 and X_2. We assume that (X_1, X_2) are uniformly distributed on $[0, 1]$x$[0, 1]$. We wish to predict a test observation's response using only observations that are within 10% of the range of X_1 and within 10% of the range of X_2 closest to that test observation. For instance, in order to predict the response for a test observation with $X_1 = 0.6$ and $X_2 = 0.35$, we will use observations in the range $[0.55, 0.65]$ for X_1 and in the range $[0.3, 0.4]$ for X_2. On average, what fraction of the available observations will we use to make the prediction?

Solution.

The position of the points is rather arbitrary, as long as they are not within 0.05 of the boundary. We can likely see that each dimension (variable) will have a probability of 0.1 to be drawn on the given interval. We can quickly assert that the probability here for finding something within 10% in both dimensions will be $(0.10)^2 = 0.01$. This could also be shown through calculus:

$$\int_{0.3}^{0.4} \int_{0.55}^{0.65} 1 \quad dx_1 dx_2 = 0.01$$

c

Now suppose that we have a set of observations on $p = 100$ features. Again the observations are uniformly distributed on each feature, and again each feature ranges in value from 0 to 1. We wish to predict a test observation's response using observations within the 10% of each feature's range that is closest to that test observation. What fraction of the available observations will we use to make the prediction?

Solution.

We can generalize our solution quickly that for observations not near the boundary, each dimensions has a probability of 0.1 for finding another draw within 10% of the first, and thus for p dimensions the probability of a point being within 10% in all dimensions becomes $(0.1)^p$. We can see this will quickly converge to zero as $p \to \infty$. At $p = 100$, the probability is beyond machine precision zero $(0.1)^{100} = 10^{-100}$. (For modern 64-bit machines, machine precision is approximately $2^{-63} \approx 10^{-19}$)

d

Using your answers to parts (a)–(c), argue that a drawback of KNN when p is large is that there are very few training observations "near" any given test observation.

Solution.

KNN is constructed around the concept of observations being 'near' one another. The above illustrates that as p grows, the probability of objects being near to one another in all dimensions converges to zero. Thus, while two objects may be close in most dimensions, by simply being further apart in one dimension, they have a chance of being classified differently. For KNN, this becomes a drawback as it can significantly effect the distance calculation and lead to models with poor performance. This concept is known as the **Curse of Dimensionality**, and is a field of study in statistics. The field that I study frequently, Functional Data Analysis, is capable of reducing the issues with Curse of Dimensionality by integrating areas rather than using point-wise estimation of a distance.

Problem 2 (ISLR : Exercise 6)

Suppose we collect data from a group of students in a statistics class with variables $X1 =$ hours studied, $X2 =$ undergrad GPA, and $Y =$ received an A. We fit a logistic regression and produce estimated coefficients, $\hat{\beta}_0 = -6$, $\hat{\beta}_1 = 0.05$, $\hat{\beta}_2 = 1$.

Setup.

We can state that our estimated model as:

$$log\left(\frac{\hat{p}}{1-\hat{p}}\right) = -6 + 0.05X1 + X2$$

a

Estimate the probability that a student who studies 40 hours and has an undergrad GPA of 3.5 gets an A in the class.

Solution.

We use that $X1 = 40$ and $X2 = 3.5$. Plugging into the above and solving for $\hat{p}$, we obtain

$$
\begin{aligned}
log\left(\frac{\hat{p}}{1-\hat{p}}\right) &= -6 + 0.05X1 + X2 \\
&= -6 + 0.05 * (40) + (3.5) \\
&= -0.5 \\
&\iff \\
\frac{\hat{p}}{1-\hat{p}} &= e^{-0.5} \\
&\iff \\
\hat{p} &= \frac{e^{-0.5}}{1+e^{-0.5}} \approx 0.3775
\end{aligned}
$$

The above equation uses the inverse-logit for the solution of $\hat{p}$. Under the given set of predictors, the probability of this student scoring an A is 0.3775.

b

How many hours would the student in part (a) need to study to have a 50% chance of getting an A in the class?

Solution.

We now let $\hat{p} = 0.5$, and solve for $X1$:

$$
\begin{aligned}
log\left(\frac{0.5}{1-0.5}\right) &= -6 + 0.05X1 + 3.5 \\
&\iff \\
log\,(1) = 0 &= -2.5 + 0.05X1 \\
&\iff \\
2.5 &= 0.05X1 \\
&\iff \\
X1 &= \frac{2.5}{0.05} = 50 \text{ hours studied}
\end{aligned}
$$

Thus, a student with a GPA of 3.5 needs to study 50 hours to have a 50% probability to score an A in this particular class.

Exercise 3

The `fuel2001` data set from the `alr4` package gives a description of fuel consumption in 2001 including the `FuelC` or gasoline sold in 1000s of gallons. We will investigate some properties of cross-validation while reviewing linear models.

a

Load the `fuel2001` data set and select the variables `FuelC`, `Pop`, and `Drivers`.

```
fuel2001 <- alr4::fuel2001 %>% select(FuelC, Pop, Drivers)
```

b

Investigate the correlation of the three selected variables within the data frame. Which variable is expected to produce a better predictive model?

```
cor(fuel2001)
```

```
##              FuelC       Pop   Drivers
## FuelC   1.0000000 0.9792750 0.9850793
## Pop     0.9792750 1.0000000 0.9949681
## Drivers 0.9850793 0.9949681 1.0000000
```

Solution.

Well this is certainly not a desirable correlation analysis. It is clear that the two variables `Pop` and `Drivers` have strong positive correlations with `FuelC`. Specifically, `Drivers` has a correlation of 0.985 and `Pop` a correlation of 0.979. Based on linear modeling, we would expect either to produce a significant linear model and expect `Drivers` to be capable of explaining more variation in `FuelC` due to higher correlation. We also expect that a two-predictor model will not be very useful, as the two predictors are likely to explain the same variations in `FuelC`.

c

Run cross-validation using $n = 34$ samples to train and the remaining samples to test. Compare three models to predict `FuelC`: a) using only `Pop`, b) using only `Drivers`, c) using both predictors. Display a summary result of your cross-validation findings. Which model is preferred? Relate this to the correlation analysis above.

Solution.

This exercise was to ensure you can write the for-loops that will be required as we continue to work through details of statistical learning. We will frequently need to write a cross-validation loop in our studies; although I expect many of you will use `caret` in the future, the goal is to understand the black-box. We produce a subset cross-validation loop that tests the three requested models, using a training set size of 34.

```
set.seed(10)
iter <- 1000
n.total <- nrow(fuel2001)
model.mse <- matrix(ncol = 3, nrow = iter)
for(j in 1:iter)
{
  index <- sample(1:n.total, 34)
  train <- fuel2001 %>% slice(index)
```

```
  test <- fuel2001 %>% slice(-index)

  ### All models train and test on the same splits
  lm.1 <- lm(FuelC ~ Pop, train)
  lm.2 <- lm(FuelC ~ Drivers, train)
  lm.3 <- lm(FuelC ~ Pop + Drivers, train)

  ### Append the testing MSE to rows
  model.mse[j,1] <- mean((test$FuelC - predict(lm.1, test))^2)
  model.mse[j,2] <- mean((test$FuelC - predict(lm.2, test))^2)
  model.mse[j,3] <- mean((test$FuelC - predict(lm.3, test))^2)
}
```

Visualize and summarize the results to make meaningful interpretations.

```
model.mse.long <- data.frame(model.mse) %>%
  rename('Pop' = X1, 'Drivers' = X2, 'Both' = X3) %>%
  pivot_longer(names_to = 'Model', values_to = 'MSE', 1:3)

model.mse.long %>%
  group_by(Model) %>%
  summarize(Mean = mean(MSE), Std.Dev = sd(MSE),
            Median = median(MSE), IQR = IQR(MSE)) %>%
  kable(align='c',
        caption = 'Summary Metrics of Test Set MSE resulting from Cross-Validation.')
```

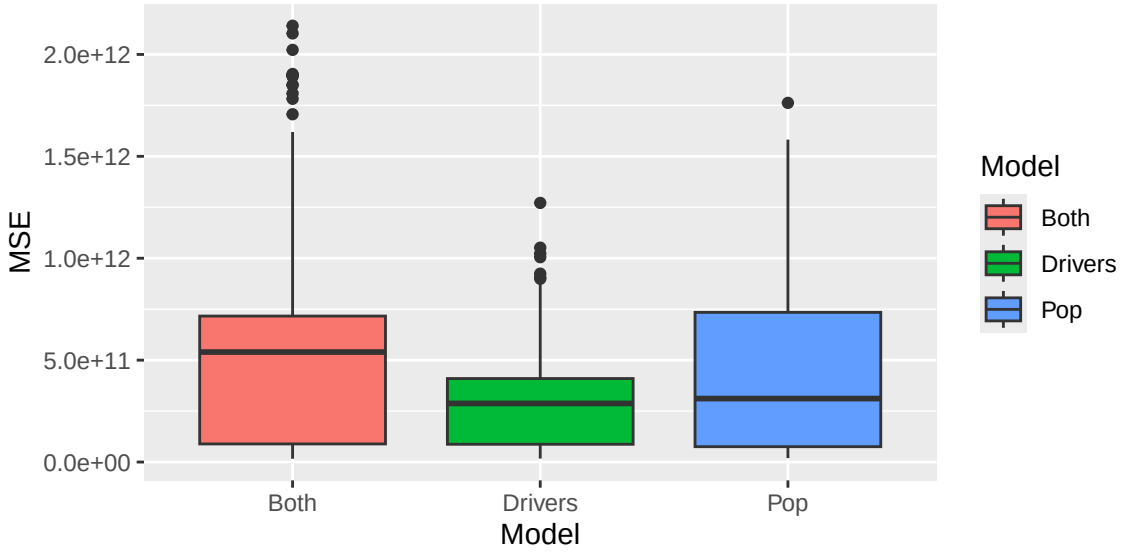
Table 1: Summary Metrics of Test Set MSE resulting from Cross-Validation.

Model	Mean	Std.Dev	Median	IQR
Both	504659424008	358552168354	539759673590	627539562313
Drivers	282343185134	209021610520	287609441874	322407482866
Pop	413634947527	353092231641	311437917561	659118514506

A summary of the mean and standard deviation for the mean test set MSE is given in the above table. We observe that the minimum mean test set MSE is the `Drivers` only model, which also has the least variance in the estimate of the irreducible error. This matches exactly our correlation analysis, in that the variable with the highest correlation is ultimately chosen by cross-validation. It is good to see things that we know well match the tools we are constructing. The `Pop` model estimated error is about 40% larger than that of the `Drivers` model, even though it had only a slightly lower correlation than that of `Drivers`. The two-predictor model has the highest estimated error rate and is unfavorable, matching what we have learned in earlier courses about using highly-correlated predictors in linear models.

We can also visualize the resulting boxplots to evaluate the distribution of error rates.

```
ggplot(model.mse.long, aes(x = Model, y = MSE, fill=Model)) +
  geom_boxplot( )
```



The visualization also shows details of the discussion above. Useful is that boxplots are based on robust statistics, such as quantiles. We see that the median performance of the two models is nearly equivalent although the mean performance clearly favored the `Drivers` model. The cause of this is evident in the visualization by the significantly larger IQR of the `Pop` model, as well as a significantly larger *maximum* observed test set MSE. The `Drivers` model is also favored because it gives more reproducible estimation (less variation).

d

Using the preferred model from part (c), we will evaluate how the choice of training and testing sizes can impact cross-validation. Run validation by subset using training proportions of 40 to 70% using every 5%. Evaluate changes to the mean test set MSE and variance of the test set MSE. Explain what you think is an optimal training set proportion.

Solution.

There is an interesting result to be produced here, but again a code exercise. We are tasked with evaluating a set of training proportions while performing subset cross-validation. We can then evaluate how the proportion effects the results. Let us work on the code. We need a second loop now that controls the proportions evaluated. Our setup from before is nearly everything we need, with a few changes. To accentuate the behavior here, I show a larger grid of proportions with a large number of iterations.

```r
set.seed(10)
iter <- 1e4
proportions <- seq(0.1, 0.9, 0.05)
n.total <- nrow(fuel2001)
prop.mse <- matrix(ncol = length(proportions), nrow = iter)
for(k in 1:length(proportions))
{
  prop.now <- proportions[k]
  for(j in 1:iter)
  {
    index <- sample(1:n.total, n.total*prop.now)
    train <- fuel2001 %>% slice(index)
    test <- fuel2001 %>% slice(-index)

    ### All models train and test on the same splits
    lm.1 <- lm(FuelC ~ Drivers, train)

    ### Append the testing MSE to rows
    prop.mse[j,k] <- mean((test$FuelC - predict(lm.1, test))^2)
  }
}
```

We can then evaluate summary metrics.

```r
prop.mse <- data.frame(prop.mse)
colnames(prop.mse) <- proportions

prop.mse.long <- prop.mse %>%
  pivot_longer(names_to = 'Proportion',
               values_to = 'MSE', 1:length(proportions)) %>%
  mutate(shade =
           ifelse(Proportion=='0.55' | Proportion == '0.6' | Proportion == '0.65',
                  'Shade', NA))
```

```
prop.mse.long %>%
  group_by(Proportion) %>%
  summarize(Mean = mean(MSE), Std.Dev = sd(MSE),
            Median = median(MSE), IQR = IQR(MSE)) %>%
  kable(align='c',
        caption = 'Summary Metrics of Test Set MSE based on training proportion.')
```

Table 2: Summary Metrics of Test Set MSE based on training proportion.

Proportion	Mean	Std.Dev	Median	IQR
0.1	435941517825	277047214658	315810119037	304720069237
0.15	398010042443	214930493216	297587006561	230057211582
0.2	374474412073	189739894349	291513568725	196428979048
0.25	366265523134	180473117181	292063920404	187659981704
0.3	350868106549	174168101754	297458376075	164973308898
0.35	344835561246	169744745766	305999479288	179110887139
0.4	333385297380	169342640304	319786380769	199845097699
0.45	323785939111	172021274056	318881339941	195275591037
0.5	313868526104	176361066783	312710239435	202697058343
0.55	300828031812	184039248550	283054890499	286076655698
0.6	294856733807	189267162888	281533825091	302263407826
0.65	288560878342	204528397926	289781386116	327464449723
0.7	284706807405	215989512387	294770988833	328229723075
0.75	275875299805	237895971657	173291056062	357203845882
0.8	276146296373	259888859464	163566367305	411289672252
0.85	265014745380	303211045896	95670316396	495924616543
0.9	262633078037	348172040143	78207181524	256407044443

The table is difficult without adding some additional details to highlight values. What might be the most important take away is that the median estimate is fairly consistent across the board (there is a trend in the mean estimate, but this is because it is chasing the significant number of outliers observed at these proportions). We notice the most symmetric estimates occur near proportion sizes of 0.5 to 0.7, which includes the commonly used `two-thirds/one-third` validation scheme. An important takeaway from this analysis is that the estimate of the irreducible error is fairly stable, until you increase the training size too high and the predictions become 'unreliably good'. When we use proportions like 0.8 or 0.9, we train models effectively a large portion of the time and see a majority of small estimates. The problem becomes what happens when the model misses, and when trained on too high a proportion, large misses can are observed (see graph below). This makes large proportions sizes unreliable and they should not be used. There exists another useful estimate known as the leave-one-out cross-validation error. We will discuss this more later in the semester, but let us calculate it for this model.

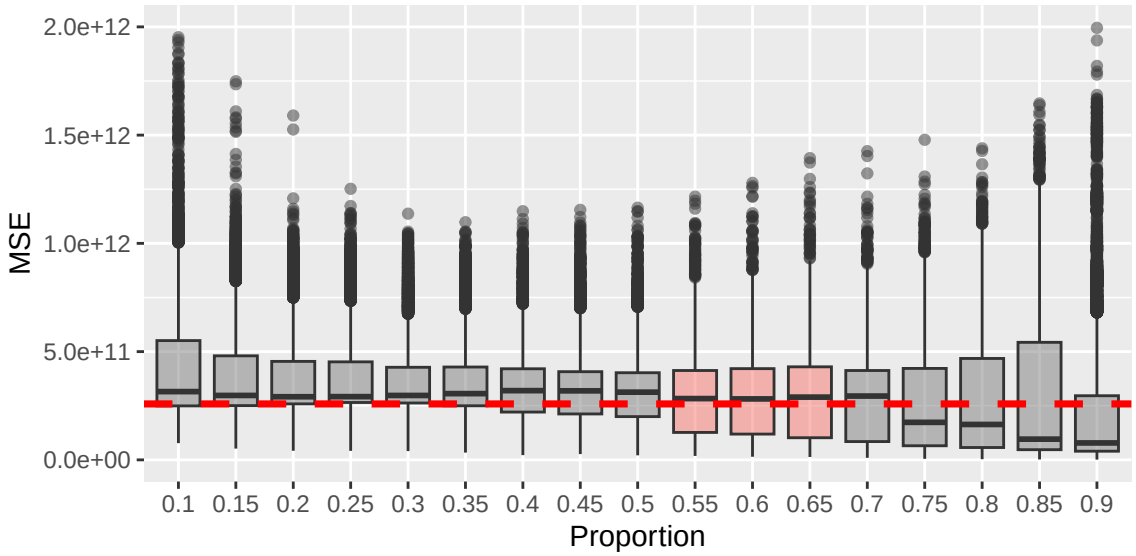
```
loocv <- numeric()
for(j in 1:nrow(fuel2001))
{
   train <- fuel2001 %>% slice(-j)
   test <- fuel2001 %>% slice(j)
   lm.1 <- lm(FuelC ~ Drivers, train)
   loocv[j] <- mean((test$FuelC - predict(lm.1, test))^2)
```

```
}
mean(loocv)
```

```
## [1] 258393575656
```

The LOOCV estimate is often considered a lower bound for the irreducible error (or conservative estimate). Notice it is lower than any of the mean estimates from the table above. It also shows why the high proportion estimates are erroneous, as their median estimates fall well below this value. The final thing we might like to evaluate are the distributions of the estimated errors.

```
ggplot(prop.mse.long, aes(x = Proportion, y = MSE)) +
  geom_boxplot(aes(fill = shade), alpha=0.5) +
  ylim(c(0, 2e12)) + theme(legend.position="none") +
  geom_hline(yintercept = mean(loocv), col='red', size=1.25, linetype='dashed')
```



This figure highlights the three proportions that are the most effective under the model here. The low proportions also have a very poor feature in that they are highly skewed distributions. There is a significant frequency of seeing models that perform either at the LOOCV level or higher. You never see values below the LOOCV error, and the distribution becomes highly skewed. This is why the mean estimates are so much higher in the table above. Notice though the three shaded boxes show an excellent distribution around the mean estimate (nearly that of the LOOCV error). In the $1e4$ iterations run, there are some outliers observed at these proportion sizes, but the distribution shows the most consistent symmetric estimates. Based on information such as this and abundant amount of empirical studies, the most commonly used split is 66%.