Chapter 5: Assessing Model Accuracy

One of the key aims of this course is to introduce you to a wide range of statistical learning techniques. Why so many? Why not just the "best one"?

Hence, it's important to decide for any given set of data which method produces the best results.

https://xkcd.com/1838/

1 Measuring Quality of Fit

With linear regression we talked about some ways to measure fit of the model

In general, we need a way to measure fit and compare *across models*.

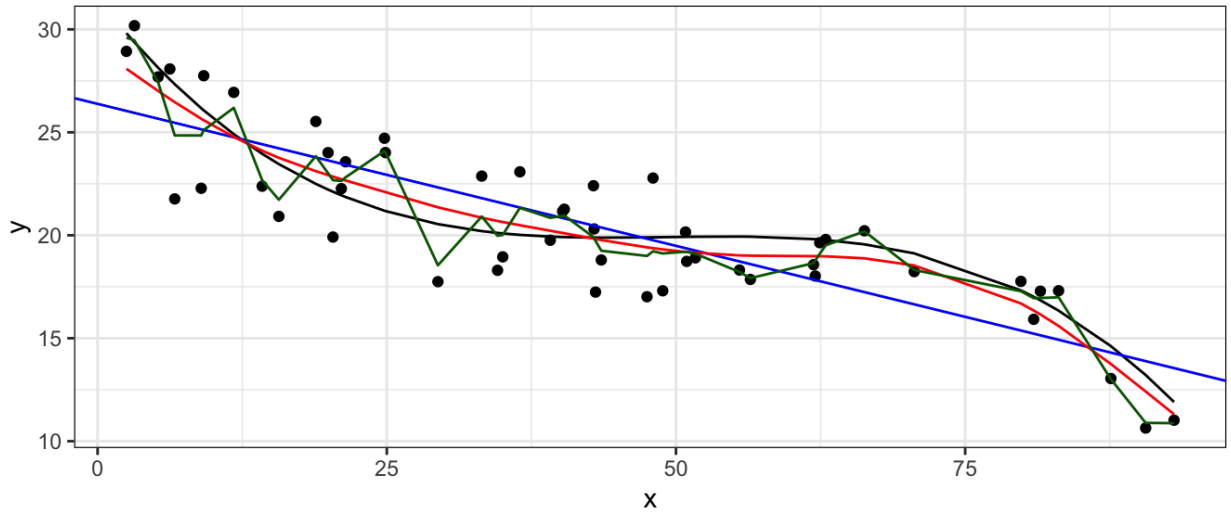
One way could be to measure how well its predictions match the observed data. In a regression session, the most commonly used measure is the *mean-squared error (MSE)*

We don't really care how well our methods work on the training data.

Instead, we are interested in the accuracy of the predictions that we obtain when we apply our method to previously unseen data. Why?

So how do we select a method that minimizes the test MSE?

But what if we don't have a test set available?



model	df	Test MSE	Train MSE
Linear Regression	2	36.0399	4.9654
Smoothing Spline	6	40.2160	3.5441
Smoothing Spline	25	38.8952	1.8645

1.1 Classification Setting

So far, we have talked about assessing model accuracy in the regression setting, but we also need a way to assess the accuracy of classification models.

Suppose we see to estimate f on the basis of training observations where now the response is categorical. The most common approach for quantifying the accuracy is the training error rate.

This is called the *training error rate* because it is based on the data that was used to train the classifier.

As with the regression setting, we are mode interested in error rates for data *not* in our training data.

1.2 Bias-Variance Trade-off

The U-shape in the test MSE curve compared with flexibility is the result of two competing properties of statistical learning methods. It is possible to show that the expected test MSE, for a given test value x_0, can be decomposed

This tells us in order to minimize the expected test error, we need to select a statistical learning method that siulatenously achieves *low variance* and *low bias*.

Variance –

Bias –

2 Cross-Validation

As we have seen, the test error can be easily calculated when there is a test data set available.

In contrast, the training error can be easily calculated.

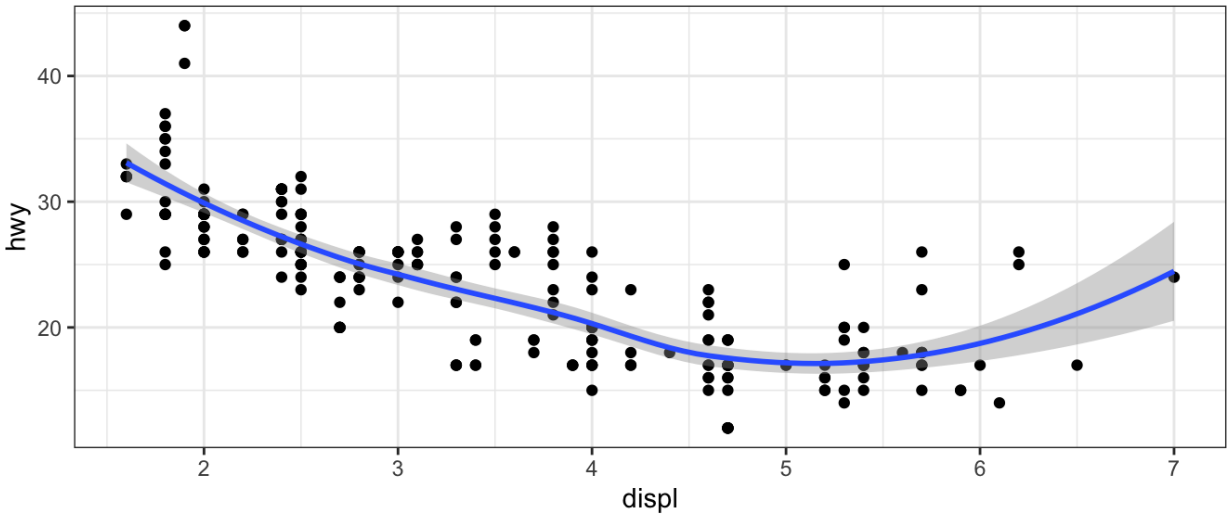
In the absense of a very large designated test set that can be used to estimate the test error rate, what to do?

For now we will assume we are in the regression setting (quantitative response), but concepts are the same for classification.

2.1 Validation Set

Suppose we would like to estimate the test error rate for a particular statistical learning method on a set of observations. What is the easiest thing we can think to do?

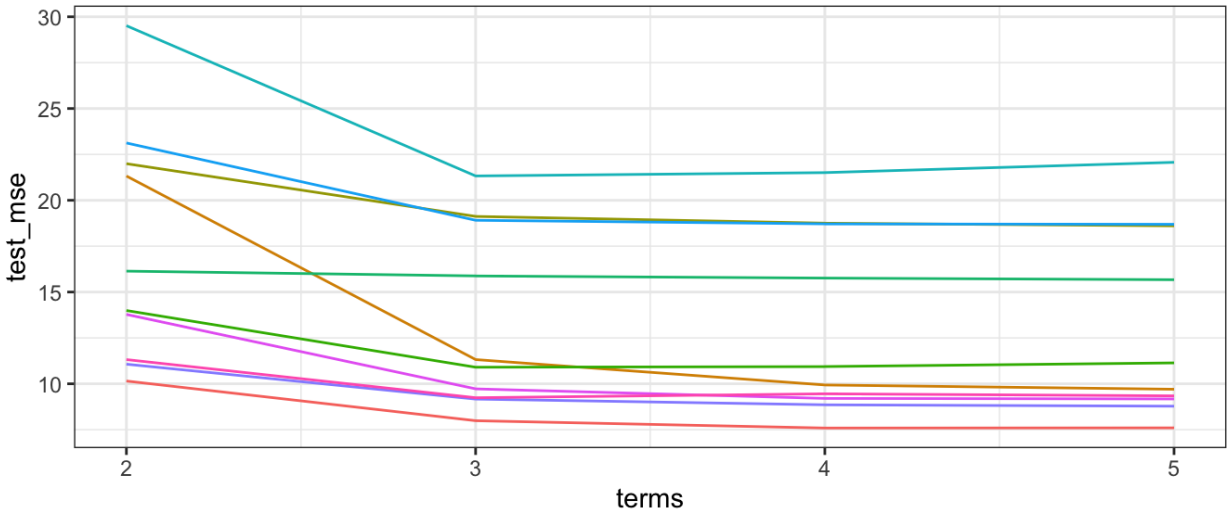
Let's do this using the `mpg` data set. Recall we found a non-linear relationship between `displ` and `hwy` mpg.



We fit the model with a squared term $\texttt{displ}^2$, but we might be wondering if we can get better predictive performance by including higher power terms!

```
## get index of training observations
# take 60% of observations as training and 40% for validation
n <- nrow(mpg)
trn <- seq_len(n) %in% sample(seq_len(n), round(0.6*n))

## fit models
m0 <- lm(hwy ~ displ, data = mpg[trn, ])
m1 <- lm(hwy ~ displ + I(displ^2), data = mpg[trn, ])
m2 <- lm(hwy ~ displ + I(displ^2) + I(displ^3), data = mpg[trn, ])
m3 <- lm(hwy ~ displ + I(displ^2) + I(displ^3) + I(displ^4), data =
  mpg[trn, ])

## predict on validation set
pred0 <- predict(m0, mpg[!trn, ])
pred1 <- predict(m1, mpg[!trn, ])
pred2 <- predict(m2, mpg[!trn, ])
pred3 <- predict(m3, mpg[!trn, ])

## estimate test MSE
true_hwy <-  mpg[!trn, ]$hwy # truth vector

data.frame(terms = 2, model = "linear", true = true_hwy, pred =
  pred0) %>%
  bind_rows(data.frame(terms = 3, model = "quadratic", true =
  true_hwy, pred = pred1)) %>%
  bind_rows(data.frame(terms = 4, model = "cubic", true = true_hwy,
  pred = pred2)) %>%
  bind_rows(data.frame(terms = 5, model = "quartic", true = true_hwy,
  pred = pred3)) %>% ## bind predictions together
  mutate(se = (true - pred)^2) %>% # squared errors
  group_by(terms, model) %>% # group by model
  summarise(test_mse = mean(se)) %>% ## get test mse
  kable() ## pretty table
```

terms	model	test_mse
2	linear	14.17119
3	quadratic	11.26710
4	cubic	11.08535
5	quartic	11.04907


30
25
20
15
10
test_mse
2
3
4
5
terms

2.2 Leave-One-Out Cross Validation

Leave-one-out cross-validation (LOOCV) is closely related to the validation set approach, but it attempts to address the method's drawbacks.

The LOOCV estimate for the test MSE is

LOOCV has a couple major advantages and a few disadvantages.

```
## perform LOOCV on the mpg dataset
res <- data.frame() ## store results
for(i in seq_len(n)) { # repeat for each observation
  trn <- seq_len(n) != i # leave one out

  ## fit models
  m0 <- lm(hwy ~ displ, data = mpg[trn, ])
  m1 <- lm(hwy ~ displ + I(displ^2), data = mpg[trn, ])
  m2 <- lm(hwy ~ displ + I(displ^2) + I(displ^3), data = mpg[trn, ])
  m3 <- lm(hwy ~ displ + I(displ^2) + I(displ^3) + I(displ^4), data =
  mpg[trn, ])

  ## predict on validation set
  pred0 <- predict(m0, mpg[!trn, ])
  pred1 <- predict(m1, mpg[!trn, ])
  pred2 <- predict(m2, mpg[!trn, ])
  pred3 <- predict(m3, mpg[!trn, ])

  ## estimate test MSE
  true_hwy <- mpg[!trn, ]$hwy # get truth vector

  res %>% ## store results for use outside the loop
    bind_rows(data.frame(terms = 2, model = "linear", true =
  true_hwy, pred = pred0)) %>%
    bind_rows(data.frame(terms = 3, model = "quadratic", true =
  true_hwy, pred = pred1)) %>%
    bind_rows(data.frame(terms = 4, model = "cubic", true = true_hwy,
  pred = pred2)) %>%
    bind_rows(data.frame(terms = 5, model = "quartic", true =
  true_hwy, pred = pred3)) %>% ## bind predictions together
    mutate(mse = (true - pred)^2) -> res
}

res %>%
  group_by(terms, model) %>%
  summarise(LOOCV_test_MSE = mean(mse)) %>%
  kable()
```

terms	model	LOOCV_test_MSE
2	linear	14.92437
3	quadratic	11.91775
4	cubic	11.78047
5	quartic	11.93978

2.3 k-Fold Cross Validation

An alternative to LOOCV is k-fold CV.

The k-fold CV estimate is computed by averaging

Why k-fold over LOOCV?

```r
## perform k-fold on the mpg dataset
res <- data.frame() ## store results

## get the folds
k <- 10
folds <- sample(seq_len(10), n, replace = TRUE) ## approximately
  equal sized

for(i in seq_len(k)) { # repeat for each observation
  trn <- folds != i # leave ith fold out

  ## fit models
  m0 <- lm(hwy ~ displ, data = mpg[trn, ])
  m1 <- lm(hwy ~ displ + I(displ^2), data = mpg[trn, ])
  m2 <- lm(hwy ~ displ + I(displ^2) + I(displ^3), data = mpg[trn, ])
  m3 <- lm(hwy ~ displ + I(displ^2) + I(displ^3) + I(displ^4), data =
  mpg[trn, ])

  ## predict on validation set
  pred0 <- predict(m0, mpg[!trn, ])
  pred1 <- predict(m1, mpg[!trn, ])
  pred2 <- predict(m2, mpg[!trn, ])
  pred3 <- predict(m3, mpg[!trn, ])

  ## estimate test MSE
  true_hwy <- mpg[!trn, ]$hwy # get truth vector

  data.frame(terms = 2, model = "linear", true = true_hwy, pred =
  pred0) %>%
    bind_rows(data.frame(terms = 3, model = "quadratic", true =
  true_hwy, pred = pred1)) %>%
    bind_rows(data.frame(terms = 4, model = "cubic", true = true_hwy,
  pred = pred2)) %>%
    bind_rows(data.frame(terms = 5, model = "quartic", true =
  true_hwy, pred = pred3)) %>% ## bind predictions together
    mutate(mse = (true - pred)^2) %>%
    group_by(terms, model) %>%
    summarise(mse = mean(mse)) -> test_mse_k

  res %>% bind_rows(test_mse_k) -> res
}
```

```
res %>%
  group_by(terms, model) %>%
  summarise(kfoldCV_test_MSE = mean(mse)) %>%
  kable()
```

terms	model	kfoldCV_test_MSE
2	linear	14.77098
3	quadratic	12.14423
4	cubic	11.94037
5	quartic	11.78830