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"?

There is no BEST model for every situation!.
↳ unless you know the true model the data comes from (which you won't).

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

How to decide?

Not like this!

https://xkcd.com/1838/

1 Measuring Quality of Fit

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

R^2, Residual standard error.

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

not just linear regression.

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)*

small if predictions are close to responses

Sometimes talk about root MSE (RMSE)

$RMSE = \sqrt{MSE}$

(same scale as response $\Rightarrow$ more interpretable)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2$$

response for i^{th} training observation

prediction for i^{th} training observation

based on training data (used to fit model) "training 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?

test data

We already know the response values for the training data!

Suppose we fit our learning method on training data $\{(x_1,y_1),\dots,(x_n,y_n)\}$ and get an estimate $\hat{f}$.

Can compute $\hat{f}(x_1),\dots,\hat{f}(x_n)$ if these are close to $y_1,\dots,y_n \Rightarrow$ small training MSE.

But we care more about:

$\hat{f}(x_0) \approx y_0$ for (x_0,y_0) unseen data not used to fit the model.

Compute $\underbrace{\text{Ave}(y_0 - \hat{f}(x_0))^2}_{\text{test MSE}}$ for large # of test observations (x_0,y_0).

Want to choose model with lowest test MSE.

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

Sometimes we have a test data set available to us based on scientific problem.

↳ access to a set of observations that were not used to fit the model.

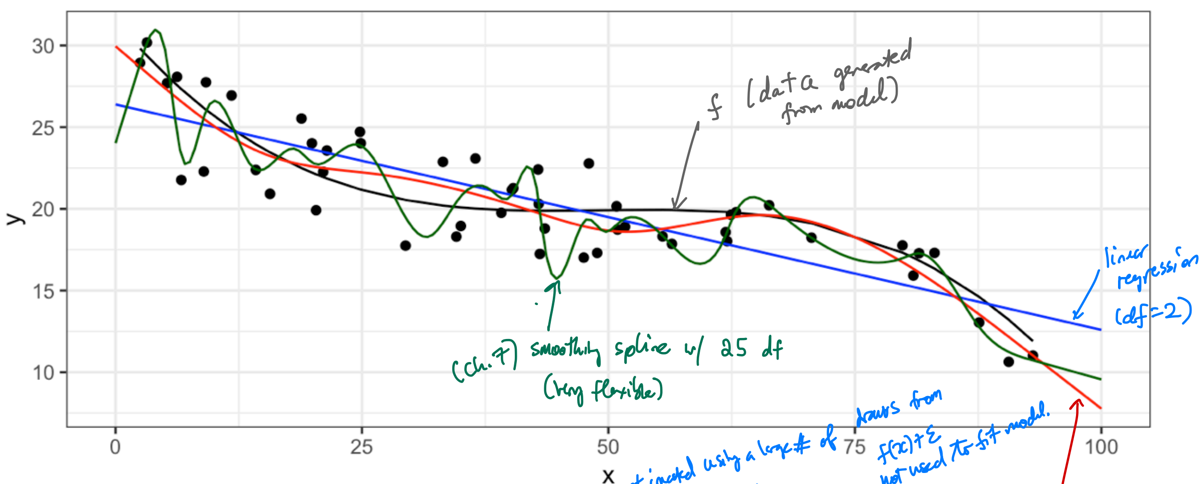
But what if we don't have a test set available?

Maybe we just minimize train MSE?

Problem: there is no guarantee lowering training MSE lowers test MSE!

Because many stat learning methods estimate coef's to lower train MSE

⟹ train MSE can be small but test MSE large!



estimated using a large # of draws from $f(x)+\varepsilon$ not used to fit model.

best test MSE!

smoothing spline w/ df=6

least flexible ↓ most flexible.

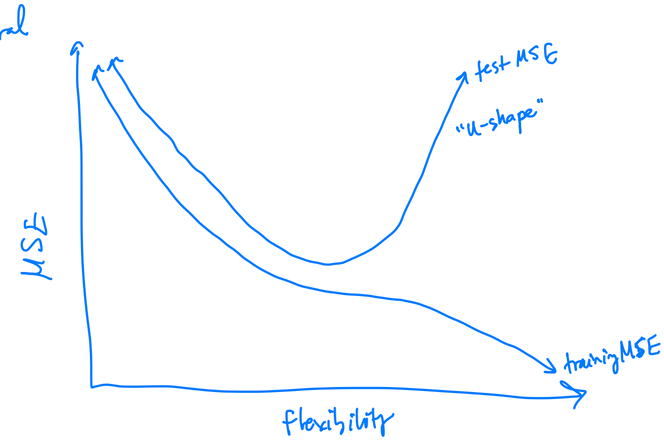
model	df	Test MSE	Train MSE
Linear Regression	2	34.4168	$\boxed{4.9654}$
Smoothing Spline	6	38.9525	3.5248
Smoothing Spline	25	39.9288	2.3107

worst training MSE

best training MSE! fits training data the best.

not the best test MSE

In general



How to choose the proper model?

Need to estimate test MSE! (next).

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.

categorical response.

Suppose we seek 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.

$$\frac{1}{n}\sum_{i=1}^{n} \mathbb{I}(y_i \neq \hat{y}_i) \quad \text{where} \quad \mathbb{I}(y_i \neq \hat{y}_i) = \begin{cases} 1 & \text{if } y_i \neq \hat{y}_i \\ 0 & \text{otherwise (correctly classified).} \end{cases}$$

true label for i^{th} training obs.

predicted label for i^{th} training obs.

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

Could also talk about "training accuracy" = 1 − training error rate.

As with the regression setting, we are mode interested in error rates for data *not* in our training data, i.e. test data (x_0, y_0)

The test error rate is

$$\text{Ave}\left(\mathbb{I}(y_0 \neq \hat{y}_0)\right)$$

predicted class for test obs w/ predictor x_0.

A good classifier is one for which the test error rate is small.

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.

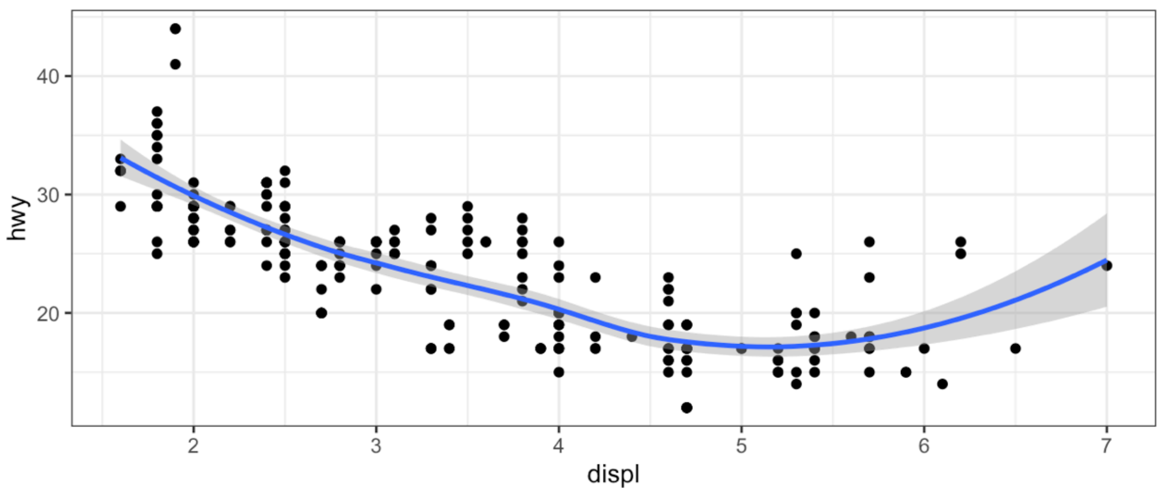
In the absense of a very large designated test set that can be used to estimate the test error rate, what to do?

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?

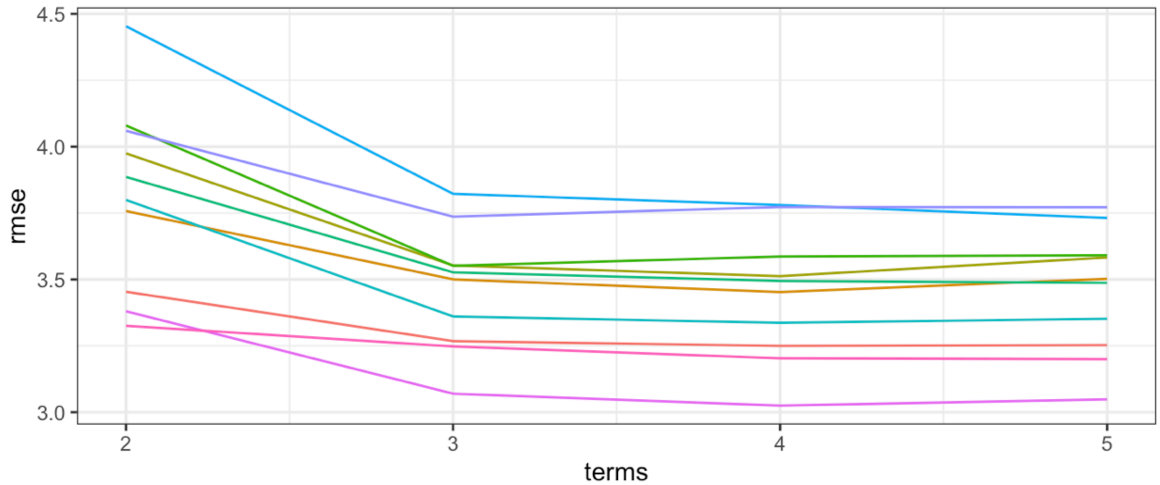
Let's do this using the `mpg` data set. Recall we found a non-linear relationship between `displ` and `hwy` mpg.



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

```r
## get index of training observations
# take 60% of observations as training and 40% for validation
mpg_val <- validation_split(mpg, prop = 0.6)

## models
lm_spec <- linear_reg()

linear_recipe <- recipe(hwy ~ displ, data = mpg)
quad_recipe <- linear_recipe |> step_mutate(displ2 = displ^2)
cubic_recipe <- quad_recipe |> step_mutate(displ3 = displ^3)
quart_recipe <- cubic_recipe |> step_mutate(displ4 = displ^4)

m0 <- workflow() |> add_model(lm_spec) |> add_recipe(linear_recipe) |>
        fit_resamples(resamples = mpg_val)
m1 <- workflow() |> add_model(lm_spec) |> add_recipe(quad_recipe) |>
        fit_resamples(resamples = mpg_val)
m2 <- workflow() |> add_model(lm_spec) |> add_recipe(cubic_recipe) |>
        fit_resamples(resamples = mpg_val)
m3 <- workflow() |> add_model(lm_spec) |> add_recipe(quart_recipe) |>
        fit_resamples(resamples = mpg_val)

## estimate test MSE
collect_metrics(m0) |> mutate(model = "linear") |>
  bind_rows(collect_metrics(m1) |> mutate(model = "quadratic")) |>
  bind_rows(collect_metrics(m2) |> mutate(model = "cubic")) |>
  bind_rows(collect_metrics(m3) |> mutate(model = "quartic")) |>
  select(model, .metric, mean) |>
  pivot_wider(names_from = .metric, values_from = mean) |>
  select(-rsq) |>
  kable()
```

model	rmse
linear	4.318968
quadratic	3.882112
cubic	3.866194
quartic	3.860612


4.5
4.0
3.5
3.0
rmse
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.

```r
## perform LOOCV on the mpg dataset
mpg_loocv <- vfold_cv(mpg, v = nrow(mpg))

## models
m0 <- workflow() |> add_model(lm_spec) |> add_recipe(linear_recipe) |>
         fit_resamples(resamples = mpg_loocv)
m1 <- workflow() |> add_model(lm_spec) |> add_recipe(quad_recipe) |>
         fit_resamples(resamples = mpg_loocv)
m2 <- workflow() |> add_model(lm_spec) |> add_recipe(cubic_recipe) |>
         fit_resamples(resamples = mpg_loocv)
m3 <- workflow() |> add_model(lm_spec) |> add_recipe(quart_recipe) |>
         fit_resamples(resamples = mpg_loocv)

## estimate test MSE
collect_metrics(m0) |> mutate(model = "linear") |>
  bind_rows(collect_metrics(m1) |> mutate(model = "quadratic")) |>
  bind_rows(collect_metrics(m2) |> mutate(model = "cubic")) |>
  bind_rows(collect_metrics(m3) |> mutate(model = "quartic")) |>
  select(model, .metric, mean) |>
  pivot_wider(names_from = .metric, values_from = mean) |>
  select(-rsq) |>
  kable()
```

model	rmse
linear	2.808356
quadratic	2.675896
cubic	2.615363
quartic	2.643536

2.3 k-Fold Cross Validation

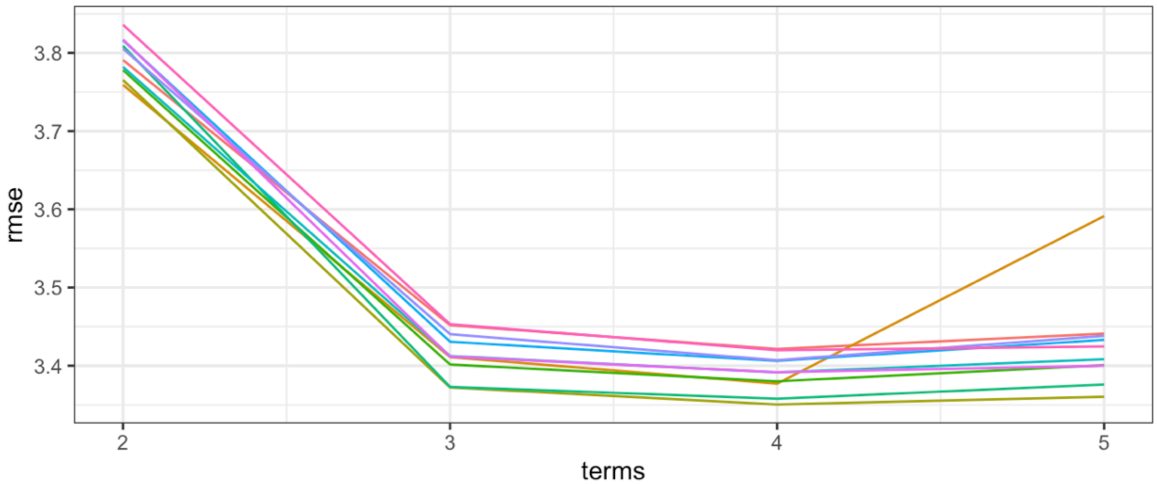
An alternative to LOOCV is k-fold CV.

The k-fold CV estimate is computed by averaging

Why k-fold over LOOCV?

```r
## perform k-fold on the mpg dataset
mpg_10foldcv <- vfold_cv(mpg, v = 10)

## models
m0 <- workflow() |> add_model(lm_spec) |> add_recipe(linear_recipe) |>
        fit_resamples(resamples = mpg_10foldcv)
m1 <- workflow() |> add_model(lm_spec) |> add_recipe(quad_recipe) |>
        fit_resamples(resamples = mpg_10foldcv)
m2 <- workflow() |> add_model(lm_spec) |> add_recipe(cubic_recipe) |>
        fit_resamples(resamples = mpg_10foldcv)
m3 <- workflow() |> add_model(lm_spec) |> add_recipe(quart_recipe) |>
        fit_resamples(resamples = mpg_10foldcv)

## estimate test MSE
collect_metrics(m0) |> mutate(model = "linear") |>
  bind_rows(collect_metrics(m1) |> mutate(model = "quadratic")) |>
  bind_rows(collect_metrics(m2) |> mutate(model = "cubic")) |>
  bind_rows(collect_metrics(m3) |> mutate(model = "quartic")) |>
  select(model, .metric, mean) |>
  pivot_wider(names_from = .metric, values_from = mean) |>
  select(-rsq) |>
  kable()
```

model	rmse
linear	3.805566
quadratic	3.432052
cubic	3.409391
quartic	3.408420


rmse
3.8
3.7
3.6
3.5
3.4
2
3
4
5
terms

2.4 Bias-Variance Trade-off for k-Fold Cross Validation

k-Fold CV with $k < n$ has a computational advantace to LOOCV.

We know the validation approach can overestimate the test error because we use only half of the data to fit the statistical learning method.

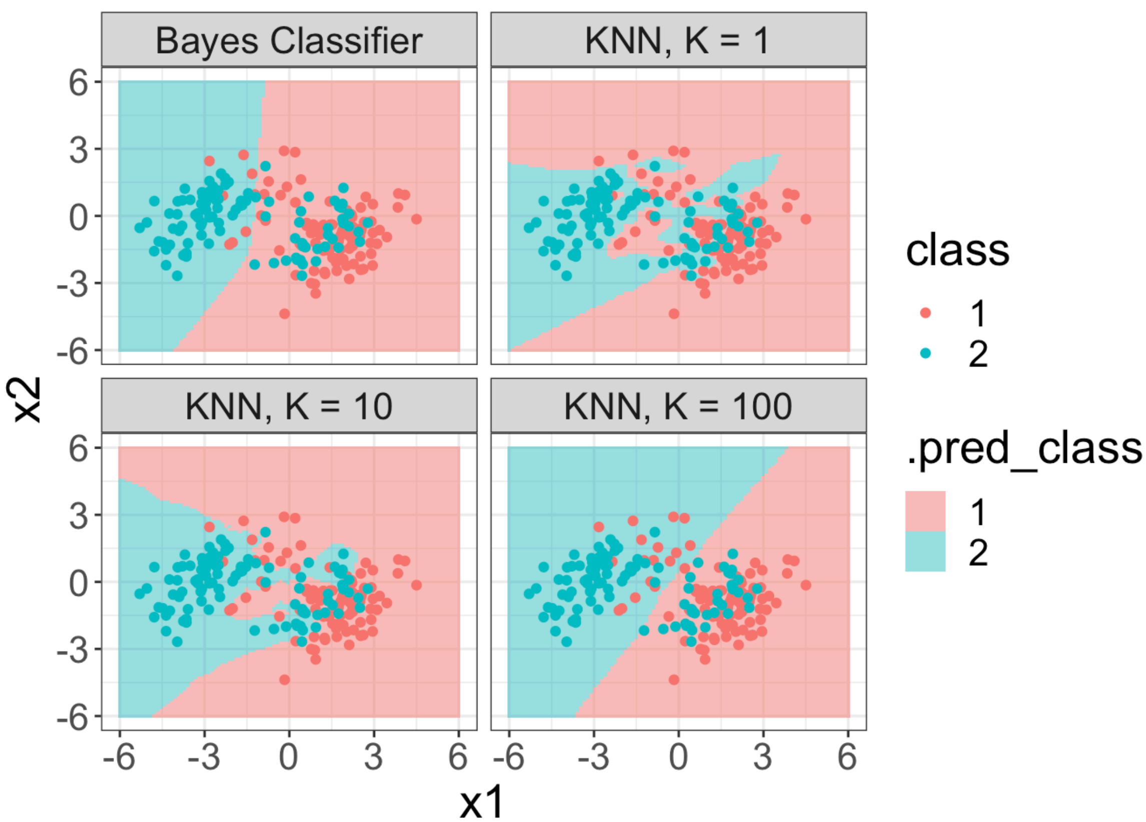
But we know that bias is only half the story! We also need to consider the procedure's variance.

To summarise, there is a bias-variance trade-off associated with the choice of k in k-fold CV. Typically we use $k = 5$ or $k = 10$ because these have been shown empirically to yield test error rates closest to the truth.

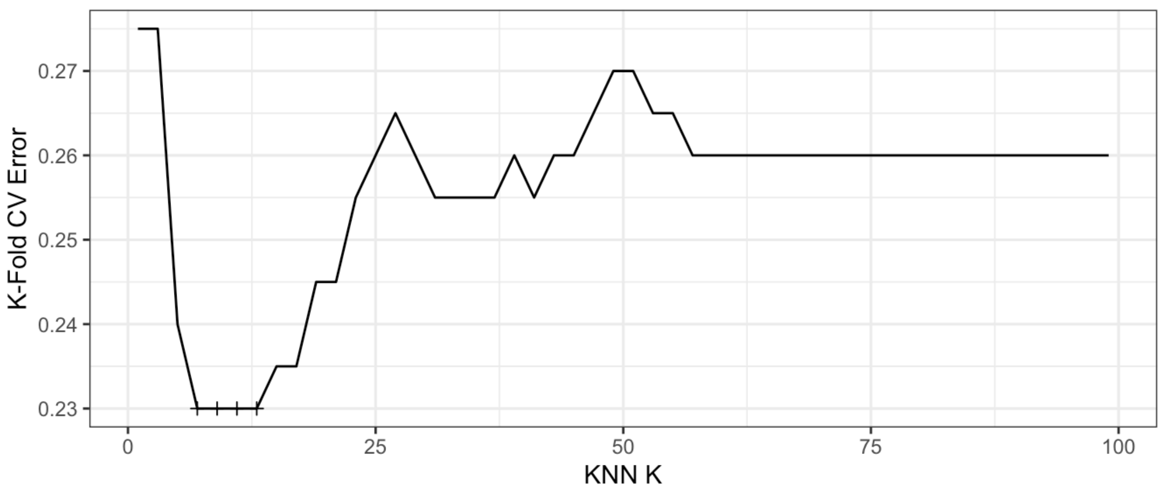
2.5 Cross-Validation for Classification Problems

So far we have talked only about CV for regression problems.

But CV can also be very useful for classification problems! For example, the LOOCV error rate for classification problems takes the form



```
k_fold <- 10
train_cv <- vfold_cv(train, v = k_fold)

grid_large <- tibble(neighbors = seq(1, 100, by = 2))

knn_spec <- nearest_neighbor(mode = "classification", neighbors =
         tune("neighbors"))
knn_spec |>
  tune_grid(class ~ x1 + x2, resamples = train_cv, grid = grid_large)
         |>
  collect_metrics() |>
  filter(.metric == "accuracy") |>
  mutate(error = 1 - mean) -> knn_err
```



Minimum CV error of 0.23 found at $K = 7$.