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 one for every situation!

exception: if you know the TRUE model that your data comes from
you won't know this.

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

How to decide?

this is NOT how we will decide!

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)*

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

small if predictions are close to the responses.

true response for ith observation; squared error; prediction for ith observation

This is based on training data (data used to fit the model). "training MSE"

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

kind of care about relationships

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 response values in our training data!

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

We can compute $\hat{f}(x_i)$. If these are close to our response y_i => small training MSE.

But we care about:

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

Want to choose the model w/ lowest test MSE

$\text{Ave}((y_0 - \hat{f}(x_0))^2)$ over a large # of test observations (x_0, y_0).

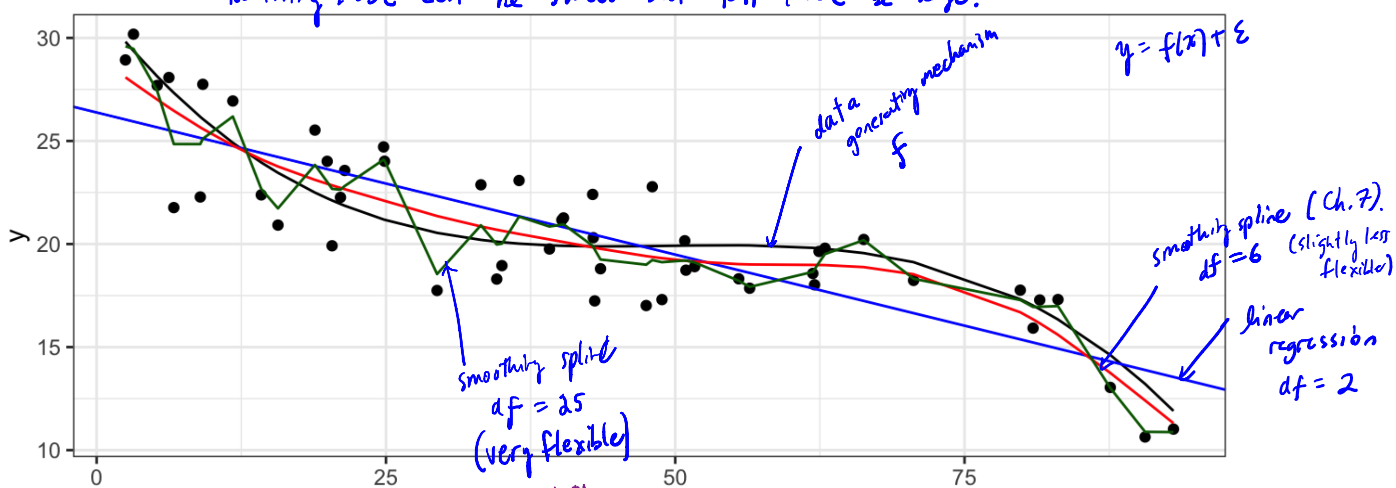
So how do we select a method that minimizes the test MSE?

Sometimes we have a test data set available to us based on scientific problem.
↳ access to set of obs. that were not used to fit 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 statistical learning methods are fit to lower training MSE.
⇒ training MSE can be small but test MSE be large!



estimate using a large # of draws from f(x) + ε not used to fit the model. ← predict each training point $\hat{y}_i$, train MSE = $\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$

least flexible ↓ most flexible.

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

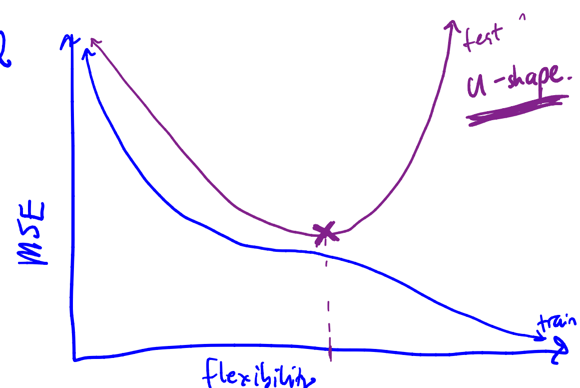
linear regression has worst training MSE.

best training MSE fits data the best!

linear best test MSE

How to choose the best model? need to estimate test MSE! (next).

In general



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.

quantitative response.

categorical responses.

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.

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

true label for ith observation

predicted value for ith observation

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 more interested in error rates for data *not* in our training data; i.e. test data (x_0, y_0)

Test error rate is

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

$\hat{y}_0$: predicted class for test observation w/ predictor x_0 $= \hat{f}(x_0)$.

A good classifier is one for which this quantity is small.

If we want a good estimate of test error, we should use many test data points.

1.2 Bias-Variance Trade-off

The U-shape in the test MSE curve compared with flexibility (or in test error) 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

$$E\left[(y_0 - \hat{f}(x_0))^2\right] = \mathrm{Var}(\hat{f}(x_0)) + \left[\mathrm{Bias}(\hat{f}(x_0))\right]^2 + \mathrm{Var}\,\varepsilon$$

($\mathrm{Var}(\hat{f}(x_0)) \geq 0$, $[\mathrm{Bias}(\hat{f}(x_0))]^2 \geq 0$, $\mathrm{Var}\,\varepsilon$: irreducible error)

$E[(y_0 - \hat{f}(x_0))^2]$: average test MSE we would obtain if we repeatedly measure f at many training data sets and predict x_0.

Overall expected test MSE obtained by averaging $E[(y_0 - \hat{f}(x_0))^2]$ over many test points (x_0, y_0).

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

Variance – the amount by which $\hat{f}$ would change if we estimated it using different training data.
In general, more flexible methods have higher variance because the fit data so closely
$\Rightarrow$ new data would result in bigger change in $\hat{f}$

Bias – the error that is introduced by approximating a real life problem by a much simpler model.

ex: linear regression assumes linear form. It is unlikely that any real-world problems are actually linear $\Rightarrow$ there will introduce bias.

In general: $\uparrow$ flexibility $\Rightarrow$ $\downarrow$ bias & $\uparrow$ variance

how much these change determine test MSE

Similar ideas hold for classification setting and test error.

2 Cross-Validation

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

Unfortunately, this is not usually the case.

In contrast, the training error can be easily calculated.

But training error can wildly underestimate test error rate.

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

Brainstorm:

Split the training set and use part of it as the "test set"

as long as we are careful about not systematically biasing our test vs. training.

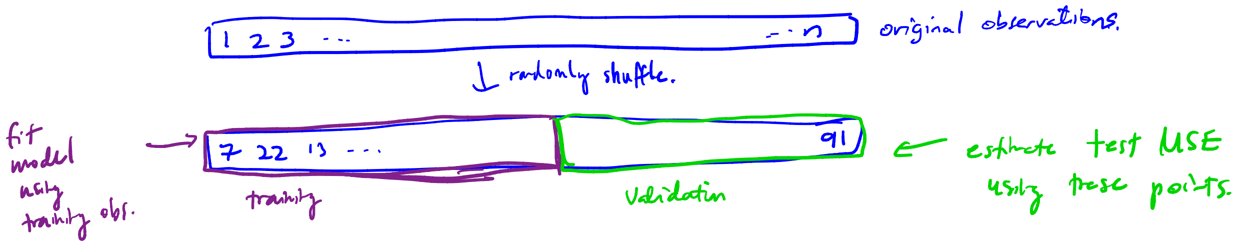
For now we will assume we are in the regression setting (quantitative response), but concepts are the same for classification.

categorical response.

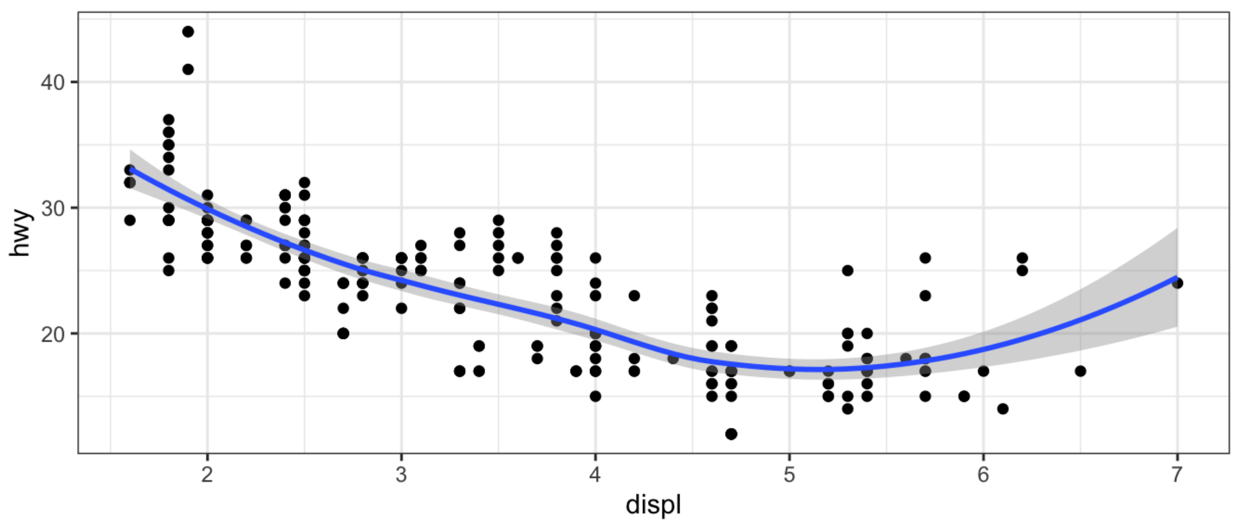
2.1 Validation Set

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

We could randomly divide the available data set into two parts: training and validation



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

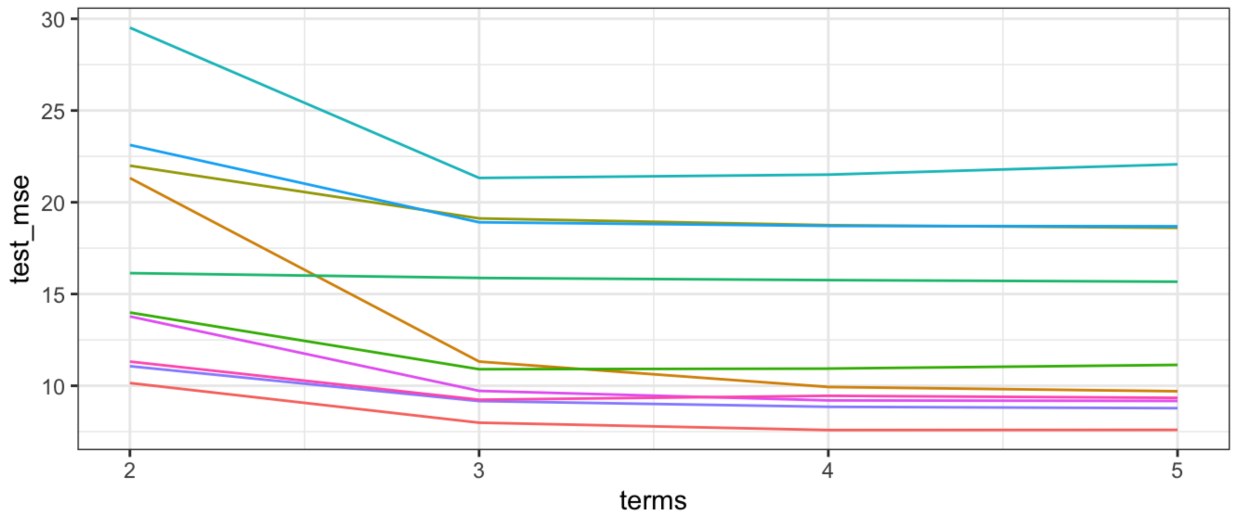


We fit the model with a squared term $\texttt{displ}^2$, but we might be wondering if we can get better predictive performance by including higher power terms!

displ^3, displ^4

```r
## 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.

```r
## 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.

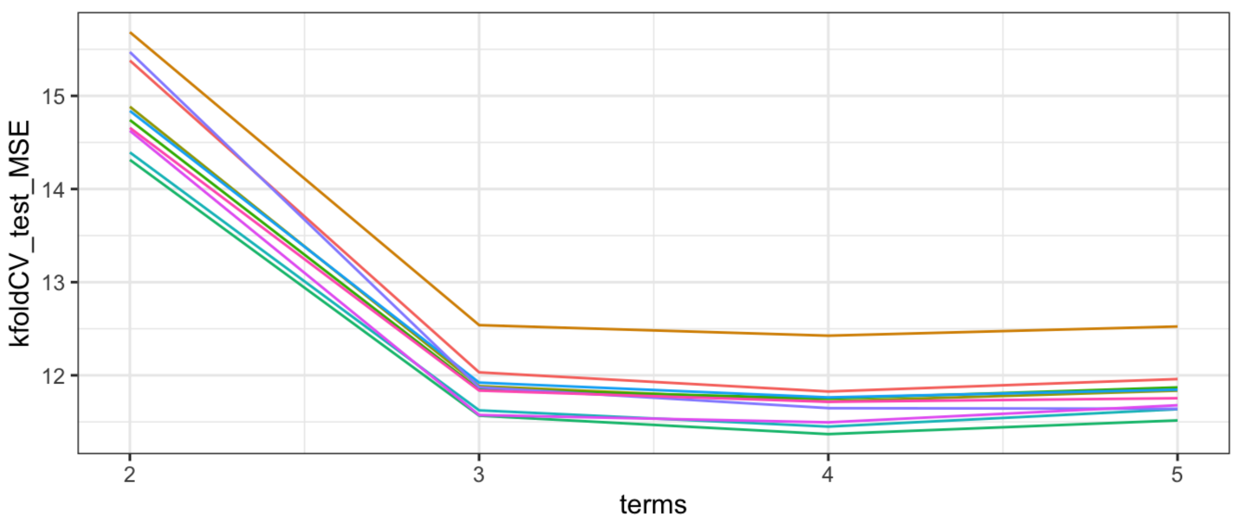
The k-fold CV estimate is computed by averaging

Why k-fold over LOOCV?

```
## 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



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.

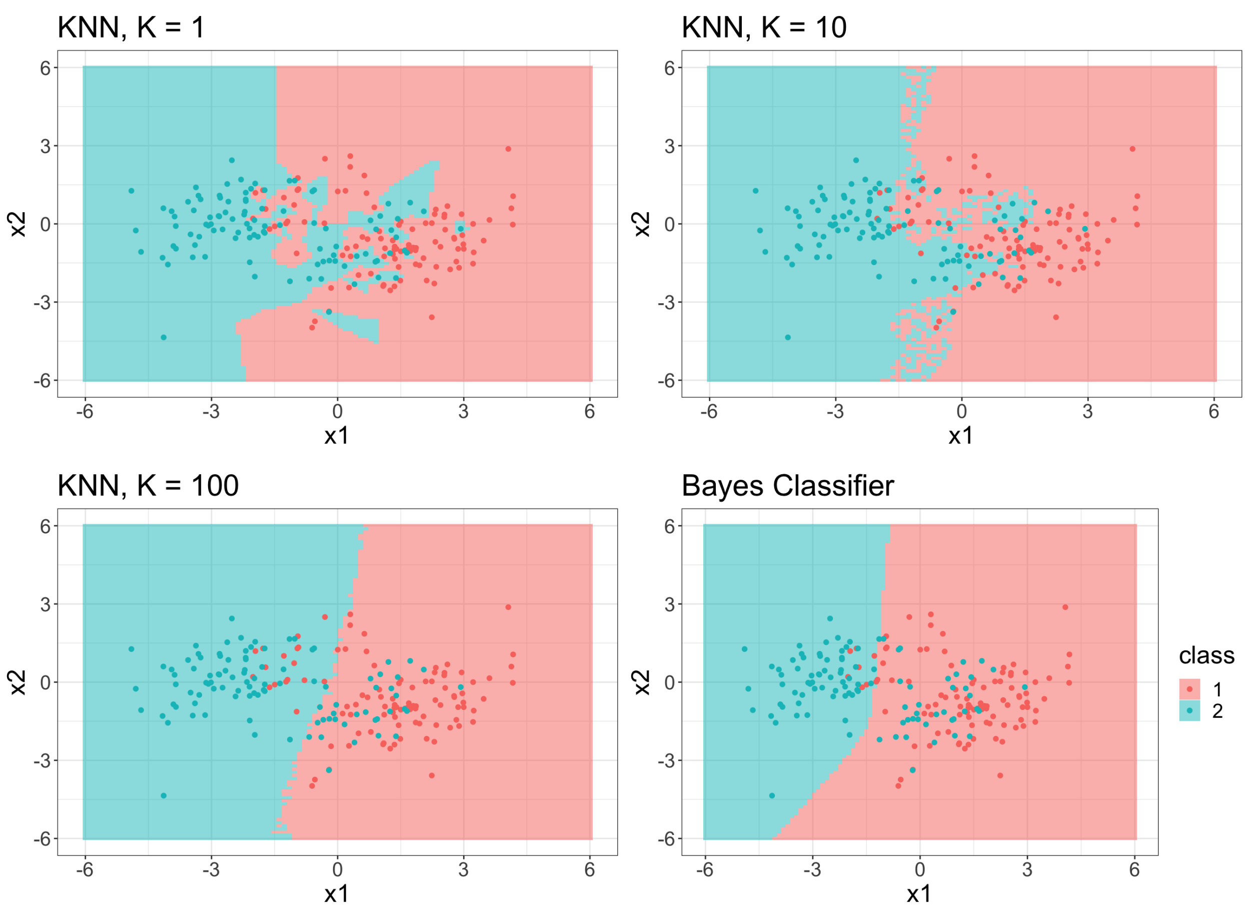
But we know that bias is only half the story! We also need to consider the procedure's variance.

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.

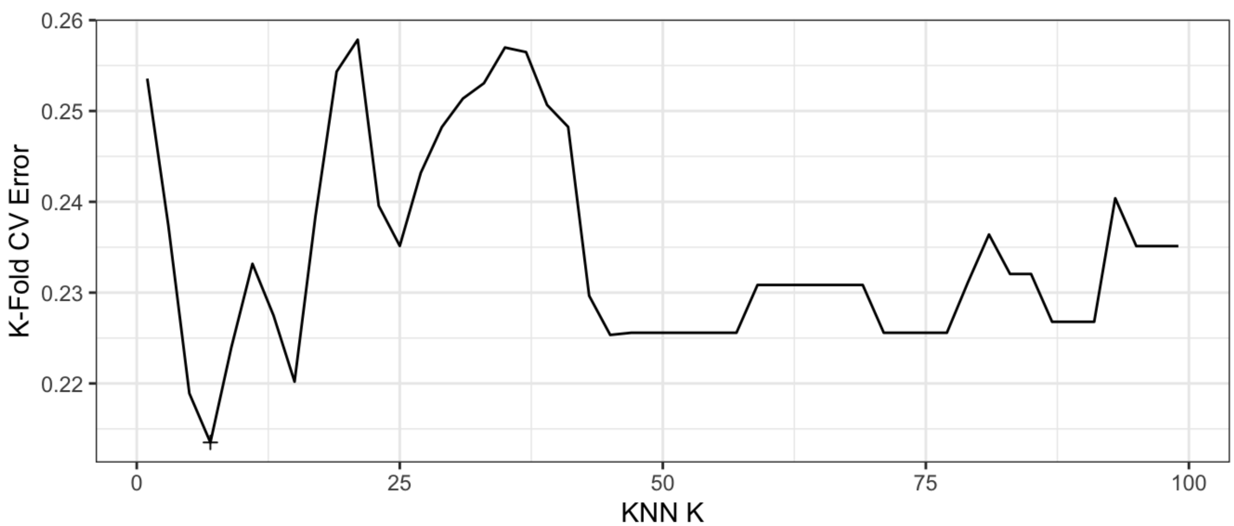
2.5 Cross-Validation for Classification Problems

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
cv_label <- sample(seq_len(k_fold), nrow(train), replace = TRUE)
err <- rep(NA, k) # store errors for each flexibility level

for(k in seq(1, 100, by = 2)) {
  err_cv <- rep(NA, k_fold) # store error rates for each fold
  for(ell in seq_len(k_fold)) {
    trn_vec <- cv_label != ell # fit model on these
    tst_vec <- cv_label == ell # estimate error on these

    ## fit knn
    knn_fit <- knn(train[trn_vec, -1], train[tst_vec, -1],
  train[trn_vec, ]$class, k = k)
    ## error rate
    err_cv[ell] <- mean(knn_fit != train[tst_vec, ]$class)
  }
  err[k] <- mean(err_cv)
}
err <- na.omit(err)
```



Minimum CV error of 0.2135 found at $K = 7$.