

Prediction

Utrecht University Winter School: Regression in R



**Utrecht
University**

Kyle M. Lang

Department of Methodology & Statistics
Utrecht University

2022-02-02

Outline

Generating Predictions

Interval Estimates for Prediction

Evaluating Predictive Models

Cross-Validation



Prediction

So far, we've focused primarily on inferences about the estimated regression coefficients.

- Asking questions about how \mathbf{X} is related to Y

We can also use linear regression for *prediction*.

- Given a new observation, X_m , what outcome value, \hat{Y}_m , does our model attribute to the m th observation?



Prediction Example

To fix ideas, let's reconsider the *diabetes* data and the following model:

$$Y_{LDL} = \beta_0 + \beta_1 X_{BP} + \beta_2 X_{gluc} + \beta_3 X_{BMI} + \varepsilon$$

Training this model on the first $N = 400$ patients' data produces the following fitted model:

$$\hat{Y}_{LDL} = 22.135 + 0.089X_{BP} + 0.498X_{gluc} + 1.48X_{BMI}$$



Prediction Example

To fix ideas, let's reconsider the *diabetes* data and the following model:

$$Y_{LDL} = \beta_0 + \beta_1 X_{BP} + \beta_2 X_{gluc} + \beta_3 X_{BMI} + \varepsilon$$

Training this model on the first $N = 400$ patients' data produces the following fitted model:

$$\hat{Y}_{LDL} = 22.135 + 0.089X_{BP} + 0.498X_{gluc} + 1.48X_{BMI}$$

Suppose a new patient presents with $BP = 121$, $gluc = 89$, and $BMI = 30.6$. We can predict their *LDL* score by:

$$\begin{aligned}\hat{Y}_{LDL} &= 22.135 + 0.089(121) + 0.498(89) + 1.48(30.6) \\ &= 122.463\end{aligned}$$

Prediction with Centered X Variables

Suppose we fit the preceding model with *BP* centered at 90, *gluc* centered at 100, and *BMI* centered at 30.

- We'd get the following fitted model:

$$\hat{Y}_{LDL} = 124.289 + 0.089X_{BP.90} + 0.498X_{gluc.100} + 1.48X_{BMI.30}$$



Prediction with Centered X Variables

Suppose we fit the preceding model with *BP* centered at 90, *gluc* centered at 100, and *BMI* centered at 30.

- We'd get the following fitted model:

$$\hat{Y}_{LDL} = 124.289 + 0.089X_{BP.90} + 0.498X_{gluc.100} + 1.48X_{BMI.30}$$

Now, let's generate predictions for our patient with *BP* = 121, *gluc* = 89, and *BMI* = 30.6:

$$\begin{aligned}\hat{Y}_{LDL} &= 124.289 + 0.089(121) + 0.498(89) + 1.48(30.6) \\ &= 224.617\end{aligned}$$



Prediction with Centered X Variables

Suppose we fit the preceding model with *BP* centered at 90, *gluc* centered at 100, and *BMI* centered at 30.

- We'd get the following fitted model:

$$\hat{Y}_{LDL} = 124.289 + 0.089X_{BP.90} + 0.498X_{gluc.100} + 1.48X_{BMI.30}$$

Now, let's generate predictions for our patient with *BP* = 121, *gluc* = 89, and *BMI* = 30.6:

$$\begin{aligned}\hat{Y}_{LDL} &= 124.289 + 0.089(121) + 0.498(89) + 1.48(30.6) \\ &= 224.617\end{aligned}$$

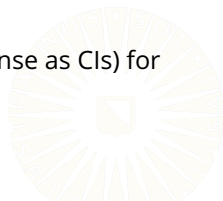
To get the correct prediction, we need to plug-in the centered X values:

$$\begin{aligned}\hat{Y}_{LDL} &= 124.289 + 0.089(121 - 90) + 0.498(89 - 100) + 1.48(30.6 - 30) \\ &= 122.463\end{aligned}$$

Interval Estimates for Prediction

To quantify uncertainty in our predictions, we want to use an appropriate interval estimate.

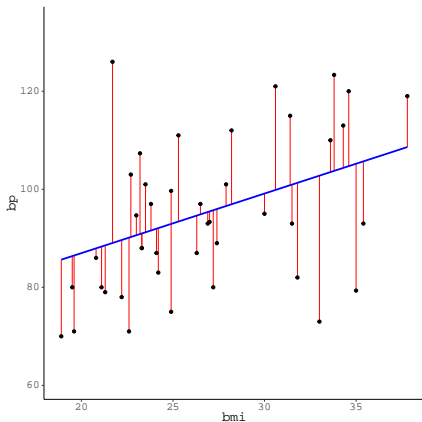
- Two flavors of interval are applicable to predictions:
 1. Confidence intervals for \hat{Y}
 2. Prediction intervals for a specific observation
- CIs for \hat{Y} give a likely range (in the sense of coverage probability and “confidence”) for the true conditional mean of Y , $\mu_{Y|X^*}$.
 - They only account for uncertainty in the estimated regression coefficients, $\{\hat{\beta}_0, \hat{\beta}_p\}$.
- Prediction intervals give a likely range (in the same sense as CIs) for future outcome values, Y^* .
 - They also account for the regression errors, ε .



Confidence vs. Prediction Intervals

Let's visualize the predictions from a simple model:

$$Y_{BP} = \hat{\beta}_0 + \hat{\beta}_1 X_{BMI} + \hat{\epsilon}$$

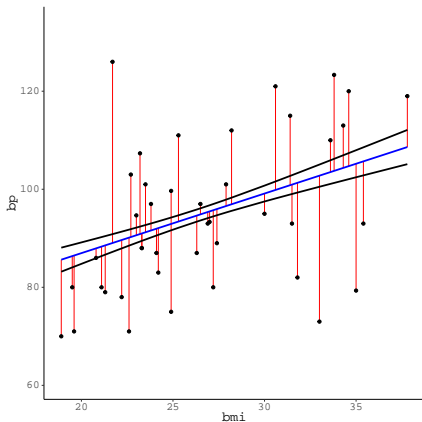


Confidence vs. Prediction Intervals

Let's visualize the predictions from a simple model:

$$Y_{BP} = \hat{\beta}_0 + \hat{\beta}_1 X_{BMI} + \hat{\epsilon}$$

- CIs for \hat{Y} ignore the errors, ϵ .
 - They only care about the best-fit line, $\beta_0 + \beta_1 X_{BMI}$.

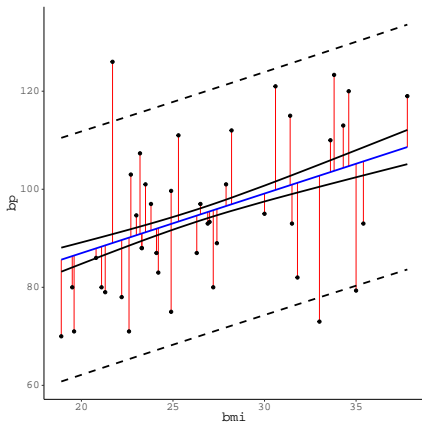


Confidence vs. Prediction Intervals

Let's visualize the predictions from a simple model:

$$Y_{BP} = \hat{\beta}_0 + \hat{\beta}_1 X_{BMI} + \hat{\epsilon}$$

- CIs for \hat{Y} ignore the errors, ϵ .
 - They only care about the best-fit line, $\beta_0 + \beta_1 X_{BMI}$.
- Prediction intervals are wider than CIs.
 - They account for the additional uncertainty contributed by ϵ .



Interval Estimates Example

Going back to our hypothetical “new” patient, we get the following 95% interval estimates:

$$95\%CI_{\hat{Y}_{LDL}} = [115.599; 129.327]$$

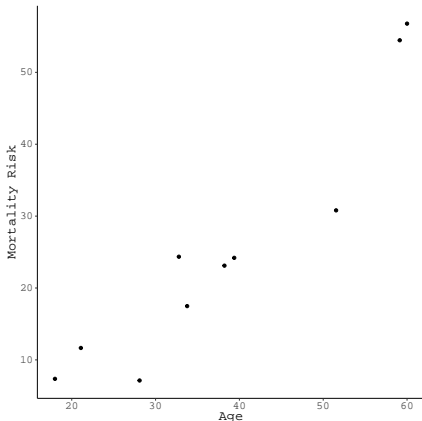
$$95\%PI = [66.559; 178.368]$$

- We can be 95% confident that the average LDL of patients with $BP = 121$, $gluc = 89$, and $BMI = 30.6$ will be somewhere between 115.599 and 129.327.
- We can be 95% confident that the LDL of a specific patient with $BP = 121$, $gluc = 89$, and $BMI = 30.6$ will be somewhere between 66.559 and 178.368.

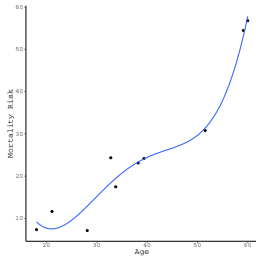
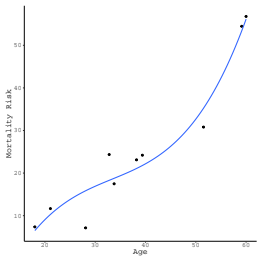
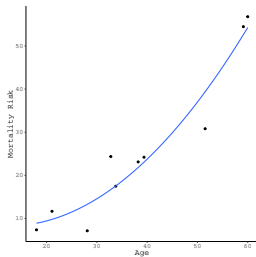
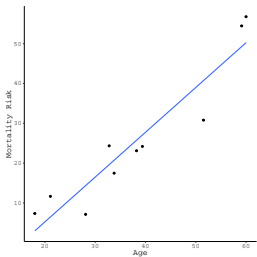
Evaluating Predictive Performance

How do we assess “good” prediction?

- Can we simply find the model that best predicts the data used to train the model?
- What are we trying to do when building a predictive model?
- Can we quantify this objective with some type of fit measure?



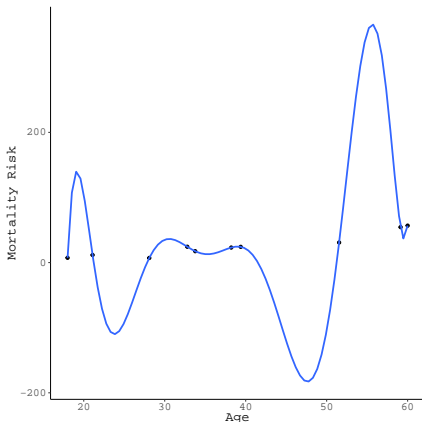
Different Possible Models



Over-fitting

We can easily go too far.

- Enough polynomial terms will exactly replicate any data.
- Is this what we're trying to do?
- What kind of issues arise in the extreme case?



Consequences of Over-fitting

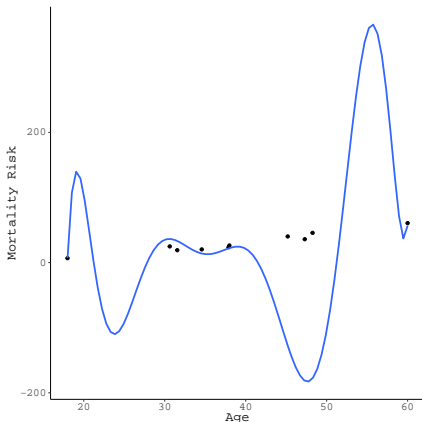
Should we be pleased to be able to perfectly predict mortality risk?

- Is our model useful?
- What happens if we try to apply our fitted model to new data?

Consequences of Over-fitting

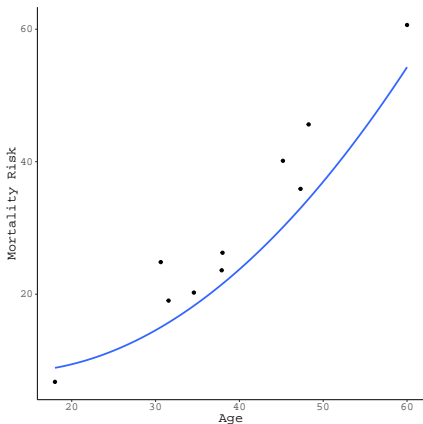
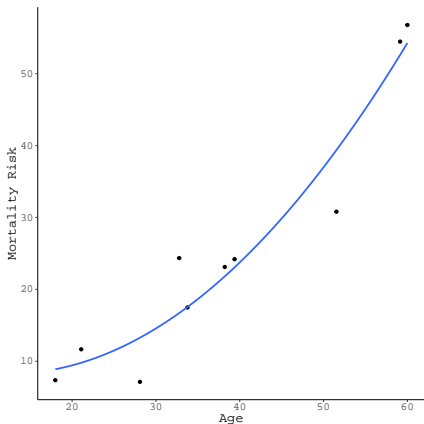
Should we be pleased to be able to perfectly predict mortality risk?

- Is our model useful?
- What happens if we try to apply our fitted model to new data?



Correct Fit

Let's try something a bit more reasonable.



A Sensible Goal

Our goal is to train a model that can best predict *new data*.

- The predictive performance on the training data is immaterial.
- We can always fit the training data arbitrarily well.
- Fit to the training data will always be at-odds with fit to future data.

This conflict is the driving force behind the *bias-variance trade-off*



Model Fit for Prediction

When assessing predictive performance, we will most often use the *mean squared error* (MSE) as our criterion.

$$\begin{aligned}MSE &= \frac{1}{N} \sum_{n=1}^N \left(Y_n - \hat{Y}_n \right)^2 \\&= \frac{1}{N} \sum_{n=1}^N \left(Y_n - \hat{\beta}_0 - \sum_{p=1}^P \hat{\beta}_p X_{np} \right)^2 \\&= \frac{RSS}{N}\end{aligned}$$



Training vs. Test MSE

The MSE on the preceding slide (i.e., the only MSE we've considered, so far) is computing entirely from training data.

- *Training MSE*

What we want is a measure of fit to new, *testing* data.

- *Test MSE*
- Given M new observations $\{Y_m, X_{m1}, X_{m2}, \dots, X_{mP}\}$, and a fitted regression model, $f(\mathbf{X})$, defined by the coefficients $\{\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2, \dots, \hat{\beta}_p\}$, the *Test MSE* is given by:

$$MSE = \frac{1}{M} \sum_{m=1}^M \left(Y_m - \hat{\beta}_0 - \sum_{p=1}^P \hat{\beta}_p X_{mp} \right)^2$$



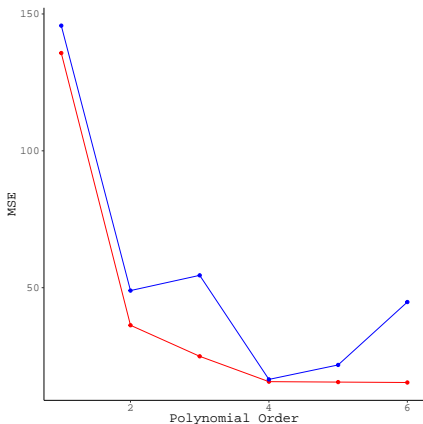
Training vs. Test MSE

Training MSE will always decrease in response to increased model complexity.

- Note the red line in the plot.

Test MSE will reach a minimum at some “optimal” level of model complexity.

- Further complicating the model will increase Test MSE.
- Note the blue line.



Training vs. Testing MSE

At the end of our model building example, we compared the following two models:

$$Y_{BP} = \beta_0 + \beta_1 X_{age} + \beta_2 X_{LDL} + \beta_3 X_{HDL} + \beta_4 X_{BMI} + \varepsilon \quad (1)$$

$$Y_{BP} = \beta_0 + \beta_1 X_{age} + \beta_2 X_{BMI} + \varepsilon \quad (2)$$

- The ΔR^2 test suggested that the loss in fit between Model 1 and Model 2 was trivial.
- The AIC and BIC both suggested that Model 2 should be preferred over Model 1.
- The training MSE values suggested that Model 1 should be preferred.

What happens when we do the comparison based on testing MSE instead of training MSE?

Training vs. Test MSE

```
set.seed(235711)

## Read in the data:
dataDir <- "../.../data/"
dDat    <- readRDS(paste0(dataDir, "diabetes.rds"))

## Split data into training and testing sets:
ind  <- sample(1 : nrow(dDat))
dat0 <- dDat[ind[1 : 400], ] # Training data
dat1 <- dDat[ind[401 : 442], ] # Testing data

## Fit the models:
outF <- lm(bp ~ age + bmi + ldl + hdl, data = dat0)
outR <- lm(bp ~ age + bmi, data = dat0)

## Compute training MSEs:
trainMseF <- MSE(y_pred = predict(outF), y_true = dat0$bp)
trainMseR <- MSE(y_pred = predict(outR), y_true = dat0$bp)
```

Training vs. Test MSE

```
## Compute testing MSEs:
```

```
testMseF <- MSE(y_pred = predict(outF, newdata = dat1),  
                y_true = dat1$bp)  
testMseR <- MSE(y_pred = predict(outR, newdata = dat1),  
                y_true = dat1$bp)
```

Compare the two approaches:

| | Full | Restricted |
|-------|--------|------------|
| Train | 147.72 | 148.44 |
| Test | 141.25 | 138.02 |

Table: MSE Values



Cross-Validation

To train a model that best predicts new data, we can use *cross-validation* to evaluate the expected predictive performance on new data.

1. Split the sample into two, disjoint sub-samples
 - *Training* data
 - *Testing* data
2. Estimate a candidate model, $f(\mathbf{X})$, on the training data.
3. Check the predictive performance of $\hat{f}(\mathbf{X})$ on the testing data.



Cross-Validation

To train a model that best predicts new data, we can use *cross-validation* to evaluate the expected predictive performance on new data.

1. Split the sample into two, disjoint sub-samples
 - *Training* data
 - *Testing* data
2. Estimate a candidate model, $f(\mathbf{X})$, on the training data.
3. Check the predictive performance of $\hat{f}(\mathbf{X})$ on the testing data.

We can use this idea to select the best model from a pool of candidate models, $\mathcal{F} = \{f_1(X), f_2(X), \dots, f_J(X)\}$

1. Repeat Steps 2 and 3 for all candidate models in \mathcal{F} .
2. Pick the $\hat{f}_j(\mathbf{X})$ that best predicts the testing data.



Different Flavors of Cross-Validation

In practice, the split-sample cross-validation procedure describe above can be highly variable.

- The solution is highly sensitive to the way the sample is split because each model is only training once.

Split-sample cross-validation can also be wasteful.

- We don't need to set aside an entire chunk of data for validation.

In most cases, we will want to employ a slightly more complex flavor of cross-validation:

- *K-fold cross-validation*



K-Fold Cross-Validation

1. Partition the data into K disjoint subsets $C_k = C_1, C_2, \dots, C_K$.



K -Fold Cross-Validation

1. Partition the data into K disjoint subsets $C_k = C_1, C_2, \dots, C_K$.
2. Conduct K training replications.
 - For each training replication, collapse $K - 1$ partitions into a set of training data, and use this training data to estimate the model.
 - Compute the test MSE for the k th partition, MSE_k , by using subset C_k as the test data for the k th fitted model.



K-Fold Cross-Validation

1. Partition the data into K disjoint subsets $C_k = C_1, C_2, \dots, C_K$.
2. Conduct K training replications.
 - For each training replication, collapse $K - 1$ partitions into a set of training data, and use this training data to estimate the model.
 - Compute the test MSE for the k th partition, MSE_k , by using subset C_k as the test data for the k th fitted model.
3. Compute the overall K-fold cross-validation error as:

$$CVE = \sum_{k=1}^K \frac{N_k}{N} MSE_k,$$

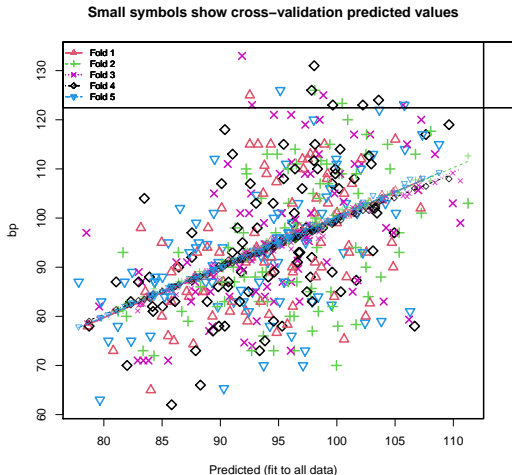


Applying K-Fold CV to our Example

```
cvOutF <- CVlm(data = diabetes,  
  form.lm = outF,  
  m = 5,  
  printit = FALSE,  
  seed = 235711)
```

```
## Estimated CVE:  
attr(cvOutF, "ms")
```

```
[1] 150.8718
```



Applying K-Fold CV to our Example

```
cvOutR <- CVlm(data = diabetes,  
  form.lm = outR,  
  m = 5,  
  printit = FALSE,  
  seed = 235711)
```

```
## Estimated CVE:  
attr(cvOutR, "ms")
```

```
[1] 149.6954
```

