Unit 2 Lecture 1: Model Complexity

STAT 471

September 16, 2021

Rolling into Unit 2

Unit 1: Intro to modern data mining

Unit 2: Tuning predictive models

Unit 3: Regression-based methods

Unit 4: Tree-based methods

Unit 5: Deep learning

Lecture 1: Model complexity

Lecture 2: Bias-variance trade-off

Lecture 3: Cross-validation

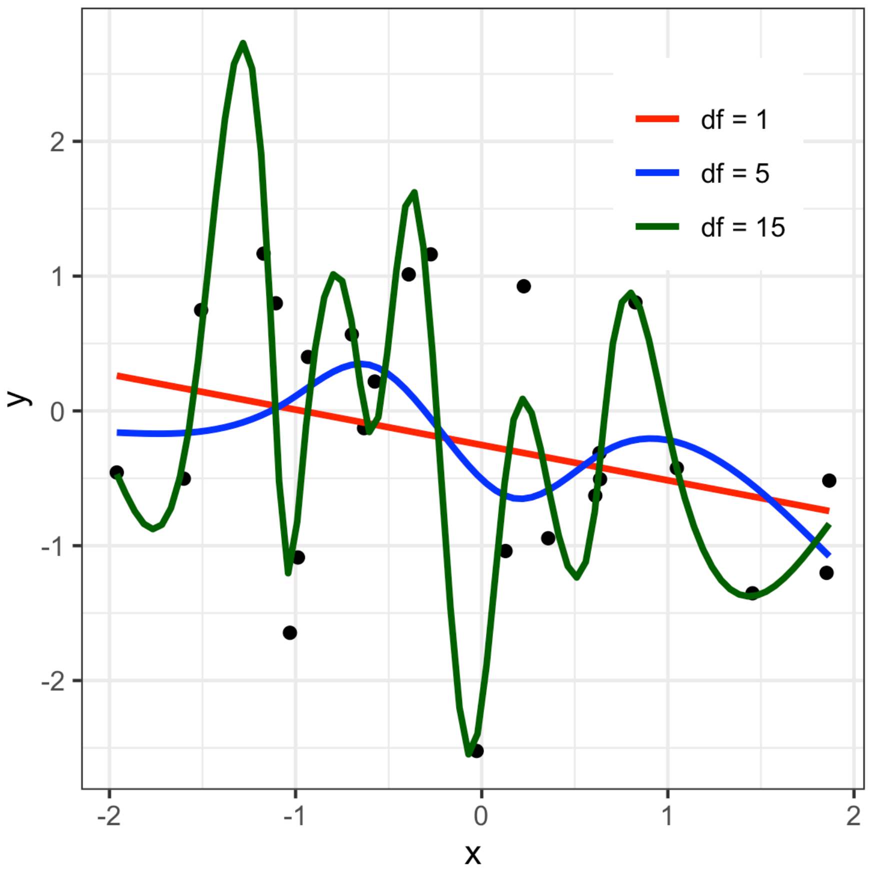
Lecture 4: Classification

Lecture 5: Unit review and quiz in class

Homework 1 due the following Sunday.

Today's lecture: Model complexity

- The same data can be fit with models of varying degrees of *flexibility* or *complexity*.
- Example: Smooth curve fits to data (called *splines*) are more flexible if they are more wiggly.
- Model complexity has an important effect on predictive performance:
    - Too flexible → too sensitive to noise in training data
    - Not flexible enough → can't capture the underlying trend



Fitting curves to data

Not all relationships are linear…

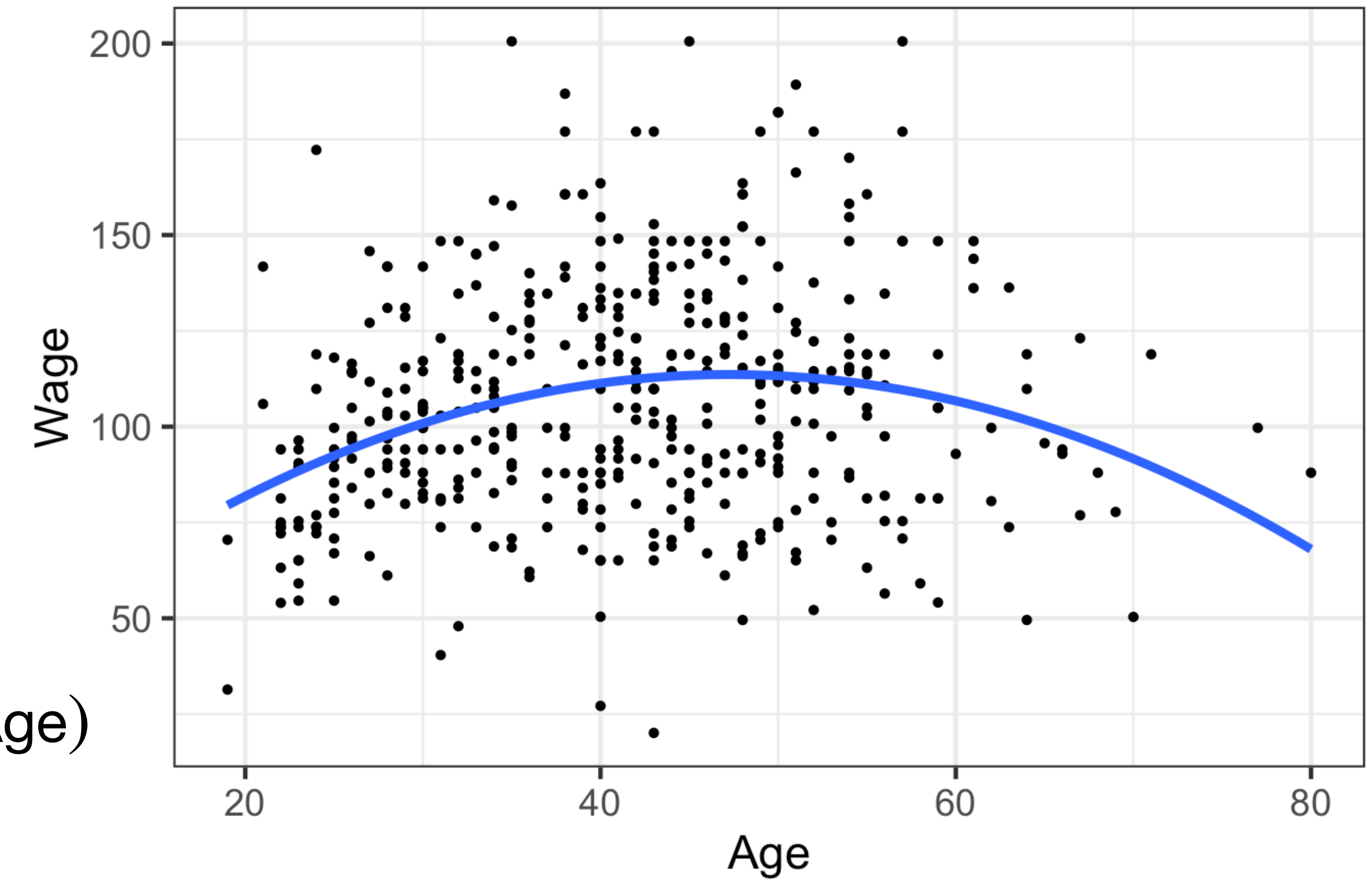
…at least in terms of the original variables.

Consider the *polynomial* model

$$\text{Wage} \approx \beta_0 + \beta_1 \text{Age} + \beta_2 \text{Age}^2.$$

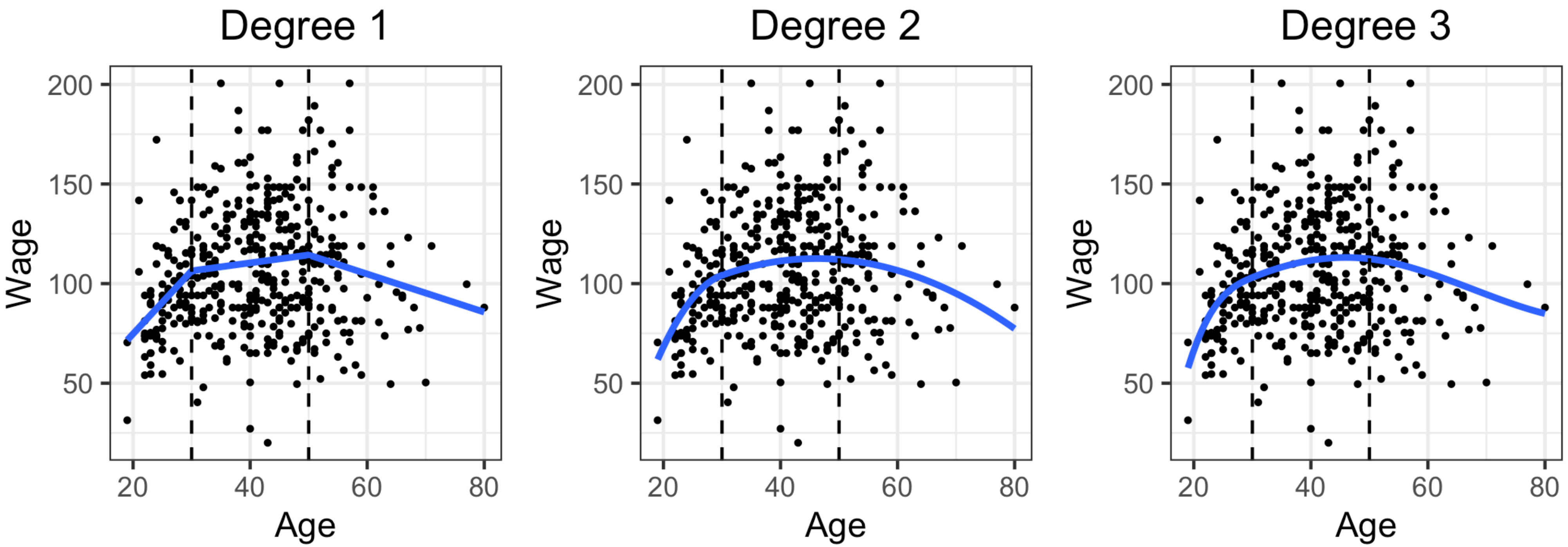
Or more generally:

$$\text{Wage} \approx \beta_1 f_1(\text{Age}) + \cdots + \beta_p f_p(\text{Age})$$



Splines: A fancier curve-fitting method

- Break range of Age into intervals separated by *knots*.
- Fit a polynomial of degree d to the data in each interval.
- “Stitch” the polynomials together to smooth transitions at knots.

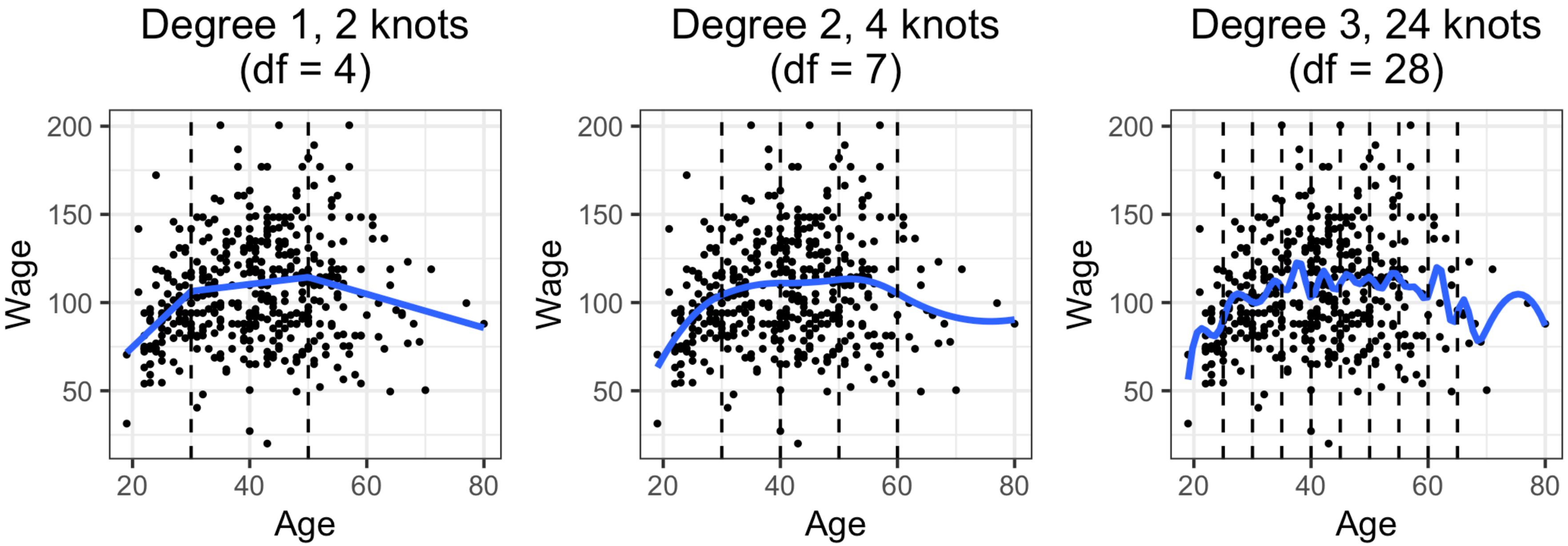


Degrees of freedom

Each spline fit can be represented in terms of a basis representation

$$\text{Wage} \approx \beta_1 f_1(\text{Age}) + \cdots + \beta_p f_p(\text{Age}).$$

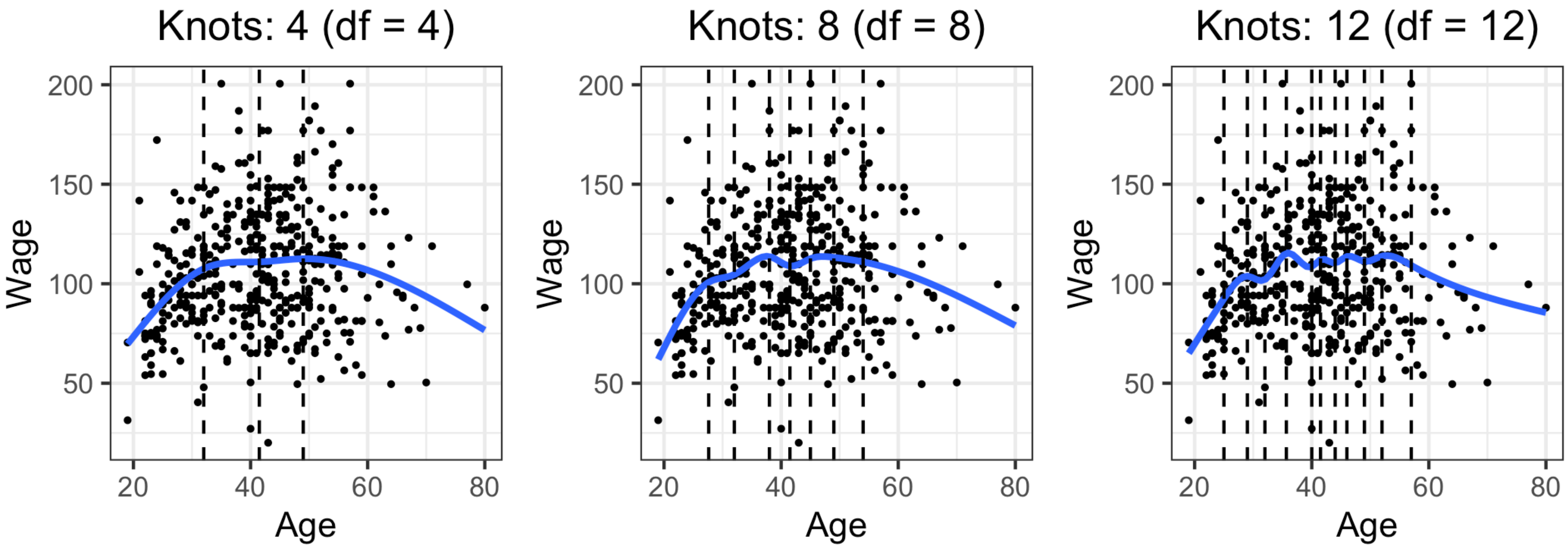
The higher p is, the more freedom the curve has to fit the data. Hence, we define *degrees of freedom (df)* $= p$.



Natural cubic splines

Like regular splines, except:

- Degree is always cubic
- Fit must be linear as you approach edges of the data
- Knots are chosen automatically at quantiles of the data



Recall: Prediction performance

You have training data $(X_1^{\text{train}}, Y_1^{\text{train}}), \ldots, (X_n^{\text{train}}, Y_n^{\text{train}})$, based on which you construct a predictive model $\hat{f}$ such that, hopefully, $Y \approx \hat{f}(X)$.

Will deploy $\hat{f}$ on test data $X_1^{\text{test}}, \ldots, X_N^{\text{test}}$ to guess $\hat{Y}_i^{\text{test}} = \hat{f}(X_i^{\text{test}})$ for each i.

Each X_i^{test} comes with a response Y_i^{test}, unknown to the predictive model.

Prediction quality: extent to which $Y_i^{\text{test}} \approx \hat{Y}_i^{\text{test}}$, e.g. mean squared test error:

$$\text{Test error of } \hat{f} = \frac{1}{N}\sum_{i=1}^{N}(Y_i^{\text{test}} - \hat{Y}_i^{\text{test}})^2.$$

Model complexity impacts prediction performance

Model complexity: how closely the model $\hat{f}$ fits the training data:

$$Y_i^{\text{train}} = f(X_i^{\text{train}}) + \epsilon_i.$$

During training, $\hat{f}$ picks up on patterns in both f (the signal) and ϵ_i (the noise).

Training error of $\hat{f}$ decreases as we increase model complexity, but test error will be high if model complexity is too low or too high.

Training error is an underestimate of the test error, especially as the model complexity increases (overfitting).