Module 8: Beyond Linearity

Content based on ISLR Chapter 7.1 – 7.7

Moving Beyond Linearlity

- The two types of regression functions that we have looked at (linear and logit) are surprisingly effective with many types of models and have advantages of interpretability.
- However, the underlying structure of the data is sometimes too non-linear to be effectively modeled with these equations and more complex models must be considered.

Moving Beyond Linearity

Outline

- Non-linear Regression Overview
  - Basis Functions
  - Polynomial Regression
  - Piecewise-Constant Regression Models (Step Functions)
  - Regression splines
  - Local regression
  - General Additive Models

Basis Functions

- A general approach to extending the linear regression and classification approaches involves the use of *basis functions*
- Instead of calculating regression coefficients on the explanatory attributes, we apply them to a function of the explanatory attributes:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \cdots + \varepsilon$$

$$Y = \beta_0 + \beta_1 b_1(X_1) + \beta_2 b_2(X_2) + \beta_3 b_3(X_3) + \cdots + \varepsilon$$

where $b_1(\cdot)$ is the basis function applied to the explanatory attribute

Polynomial Regression

- Already covered in Module 3 as a simple extension of the linear model

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 {X_1}^2 + \beta_3 {X_1}^3 + \cdots + \varepsilon$$

- As previously noted, this is still a linear model and we can use all of the linear regression techniques by simply creating a new transformed attributes $({X_1}^2, {X_1}^3, \ldots)$
- Polynomial regression is one example of the use of a basis function approach where the basis function is given as:

$$b_j(X_1) = X_1^j$$

Wage Example

Polynomial Regression

	year	age	sex	maritl	race	education	region	jobclass	health	health_ins	logwage	wage
0	2006	18	1. Male	1. Never Married	1. White	1. < HS Grad	2. Middle Atlantic	1. Industrial	1. <=Good	2. No	4.318063	75.043154
1	2004	24	1. Male	1. Never Married	1. White	4. College Grad	2. Middle Atlantic	2. Information	2. >=Very Good	2. No	4.255273	70.476020
2	2003	45	1. Male	2. Married	1. White	3. Some College	2. Middle Atlantic	1. Industrial	1. <=Good	1. Yes	4.875061	130.982177
3	2003	43	1. Male	2. Married	3. Asian	4. College Grad	2. Middle Atlantic	2. Information	2. >=Very Good	1. Yes	5.041393	154.685293
4	2005	50	1. Male	4. Divorced	1. White	2. HS Grad	2. Middle Atlantic	2. Information	1. <=Good	1. Yes	4.318063	75.043154
...	...	...	...	...	...	...	...	...	...	...	...	...
2995	2008	44	1. Male	2. Married	1. White	3. Some College	2. Middle Atlantic	1. Industrial	2. >=Very Good	1. Yes	5.041393	154.685293
2996	2007	30	1. Male	2. Married	1. White	2. HS Grad	2. Middle Atlantic	1. Industrial	2. >=Very Good	2. No	4.602060	99.689464
2997	2005	27	1. Male	2. Married	2. Black	1. < HS Grad	2. Middle Atlantic	1. Industrial	1. <=Good	2. No	4.193125	66.229408
2998	2005	27	1. Male	1. Never Married	1. White	3. Some College	2. Middle Atlantic	1. Industrial	2. >=Very Good	1. Yes	4.477121	87.981033
2999	2009	55	1. Male	5. Separated	1. White	2. HS Grad	2. Middle Atlantic	1. Industrial	1. <=Good	1. Yes	4.505150	90.481913

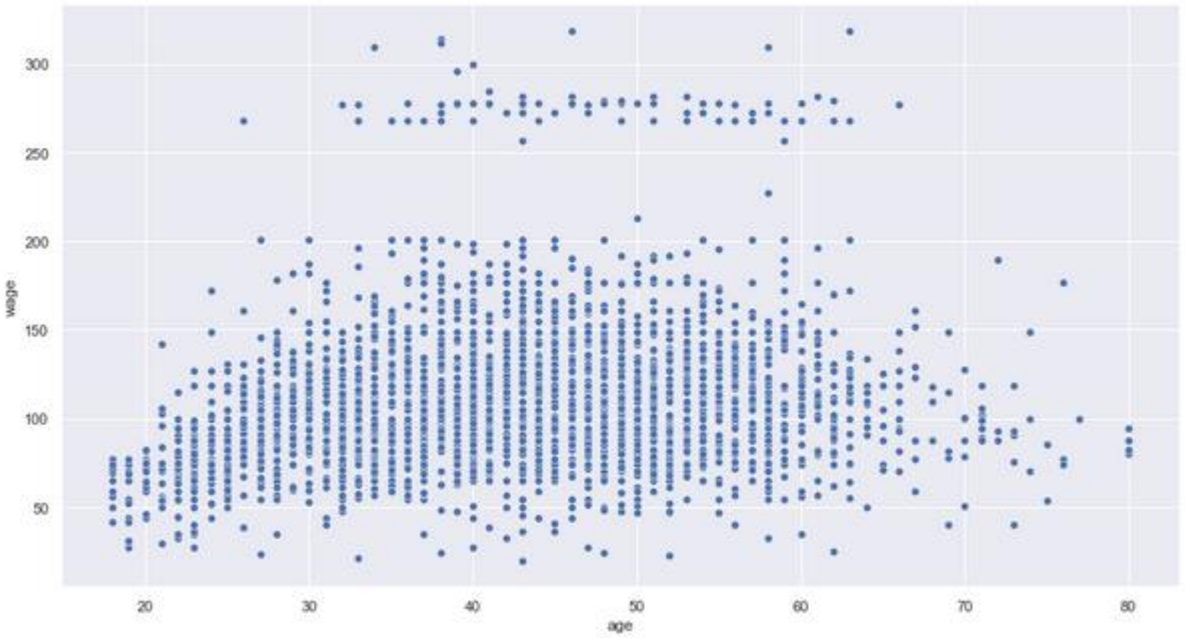
3000 rows × 12 columns

Wage Example

Polynomial Regression

```
import seaborn as sns
sns.set(rc = {'figure.figsize':(15,8)})
sns.scatterplot(data = wage, x = 'age', y = 'wage')
```

```
<AxesSubplot:xlabel='age', ylabel='wage'>
```



Wage Example

Polynomial Regression

Define sklearn polynomial regregession model using Sklearn pipeline function

```
]:
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import Pipeline

# Pipeline chains together multiple steps
poly_model = Pipeline([('poly', PolynomialFeatures(degree=4)), ('linear', LinearRegression())])
poly_model.fit(wage_data[['age']],wage_data['wage'])
```

```
]: Pipeline(steps=[('poly', PolynomialFeatures(degree=4)),
                ('linear', LinearRegression())])
```

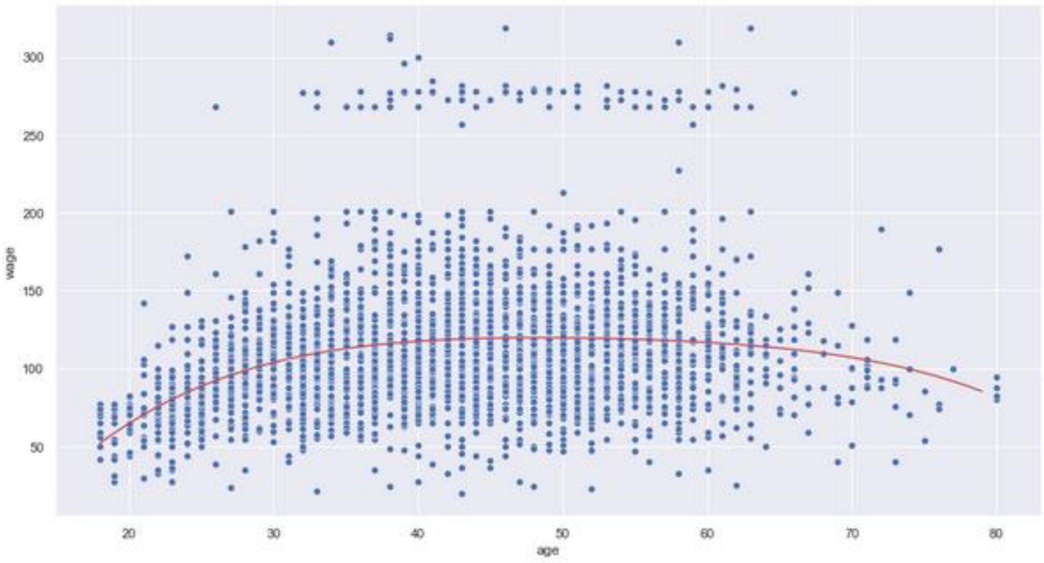
Wage Example

Polynomial Regression

Create array of ages and use model to predict wages to create curve and plot

```
ages = np.arange(wage['age'].min(), wage['age'].max(), 1)
y_hat = poly_model.predict(ages.reshape(-1,1))
sns.set(rc = {'figure.figsize':(15,8)})
sns.scatterplot(data = wage, x = 'age', y = 'wage')
sns.lineplot(x=ages, y=y_hat, color = 'r')
```

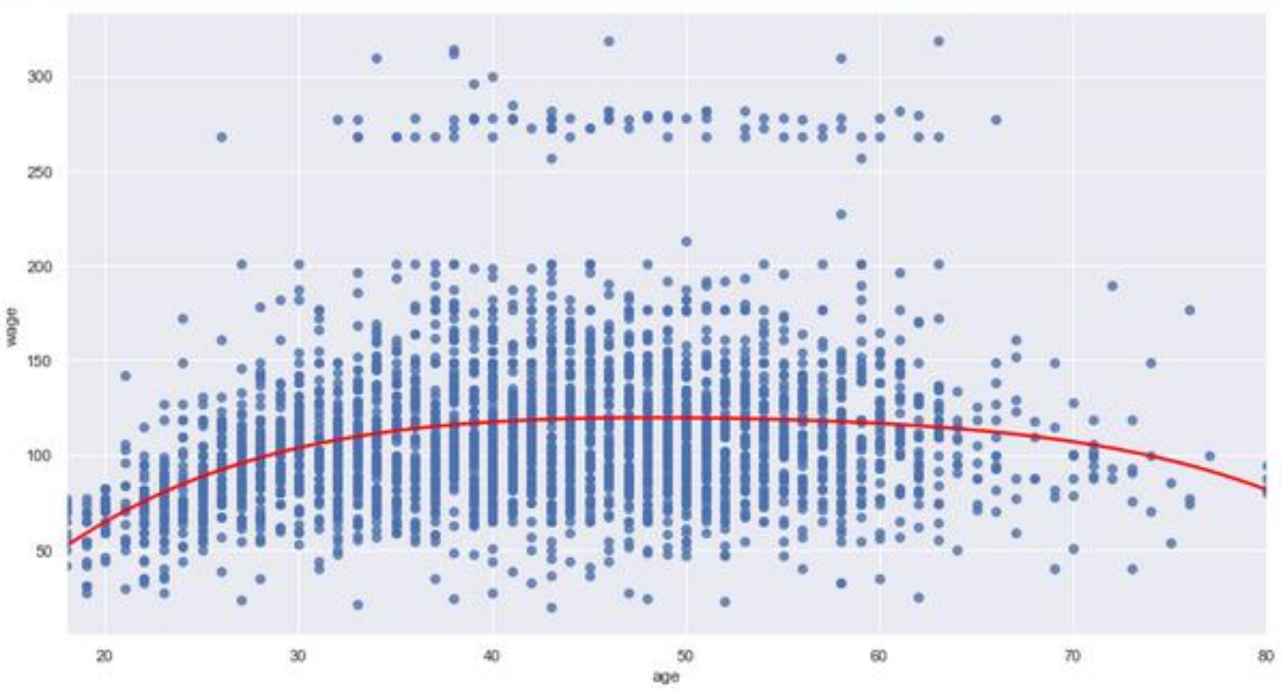
```
<AxesSubplot:xlabel='age', ylabel='wage'>
```



Wage Example

Polynomial Regression

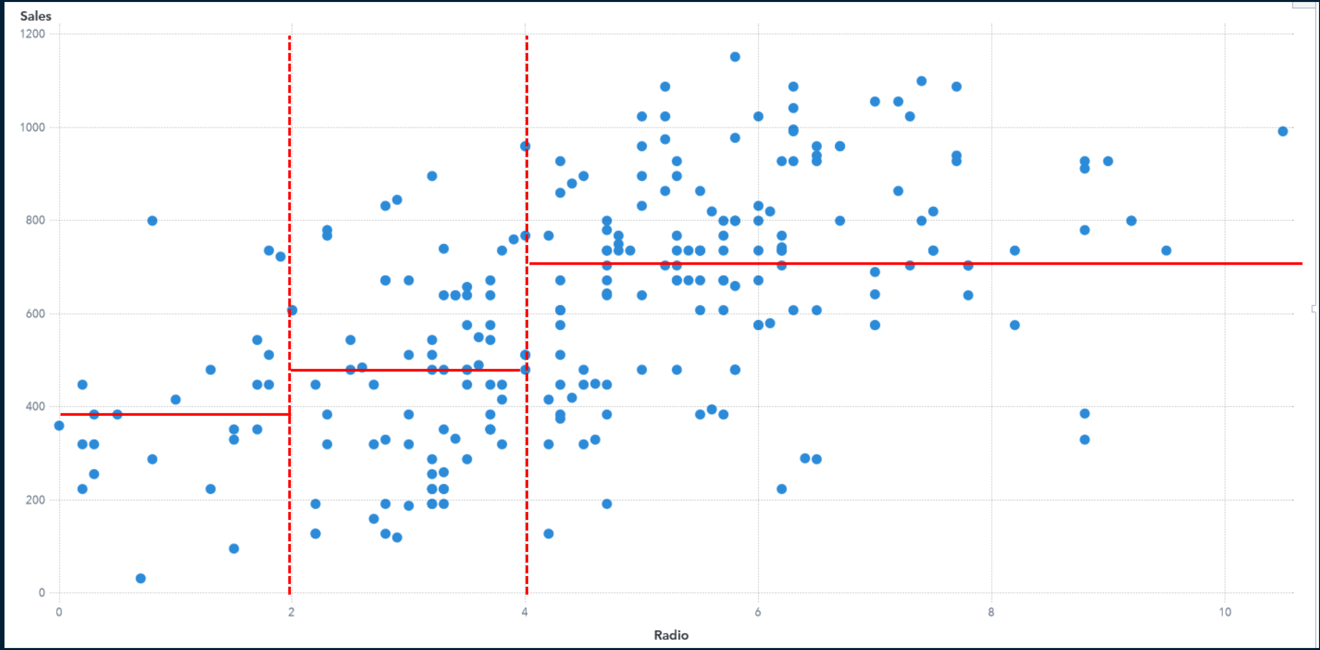
```
sns.regplot(x=wage['age'], y=wage['wage'], ci=None, order=4, line_kws={"color": "red"});
```



Piecewise-Constant Regression Models

"Step Functions"

- Instead of imposing a global polynomial structure on model, we cut the variable into distinct regions and fit with piecewise constants:



Step Function

- Model is easily fit by creating *Indicator Functions* $I(\cdot)$ that return 1 if the condition is true and 0 otherwise)
- To create the step function model in our example, we need three step functions:
  - $C_1(Radio) = I(Radio < 2)$
  - $C_2(Radio) = I(2 \leq\ Radio \leq 4)$
  - $C_3(Radio) = I(Radio > 4)$
- Then, we use standard least squares to fit a linear model with these the indicators as predictors:

$$Sales = \ \beta_0\ + \beta_1 C_1(Radio) + \beta_2 C_2(Radio) + \beta_3 C_3(Radio)$$

Step Function

Wage Example

Step Function

```
df_cut, bins = pd.cut(wage['age'], 4, retbins=True, right=True)
bins
```

```
array([17.938, 33.5  , 49.   , 64.5  , 80.   ])
```

```
df_cut
```

```
0       (17.938, 33.5]
1       (17.938, 33.5]
2         (33.5, 49.0]
3         (33.5, 49.0]
4         (49.0, 64.5]
             ...
2995      (33.5, 49.0]
2996    (17.938, 33.5]
2997    (17.938, 33.5]
2998    (17.938, 33.5]
2999      (49.0, 64.5]
Name: age, Length: 3000, dtype: category
Categories (4, interval[float64]): [(17.938, 33.5] < (33.5, 49.0] < (49.0, 64.5] < (64.5, 80.0]]
```

Step Function

Wage Example

```
df_steps = pd.concat([wage['age'], df_cut, wage['age']], keys=['age','age_cuts','wage'], axis=1)
df_steps
```

	age	age_cuts	wage
0	18	(17.938, 33.5]	18
1	24	(17.938, 33.5]	24
2	45	(33.5, 49.0]	45
3	43	(33.5, 49.0]	43
4	50	(49.0, 64.5]	50
...	...	...	...
2995	44	(33.5, 49.0]	44
2996	30	(17.938, 33.5]	30
2997	27	(17.938, 33.5]	27
2998	27	(17.938, 33.5]	27
2999	55	(49.0, 64.5]	55

3000 rows × 3 columns

Step Function

Wage Example

```python
# Create dummy variables for the age groups
X = pd.get_dummies(df_steps, drop_first = True).drop(['wage', 'age'],1)
X
```

	age_cuts_(33.5, 49.0]	age_cuts_(49.0, 64.5]	age_cuts_(64.5, 80.0]
0	0	0	0
1	0	0	0
2	1	0	0
3	1	0	0
4	0	1	0
...	...	...	...
2995	1	0	0
2996	0	0	0
2997	0	0	0
2998	0	0	0
2999	0	1	0

3000 rows × 3 columns

Step Function

Wage Example

```
bin_mapping = np.digitize(ages, bins)
bin_mapping
```

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4], dtype=int64)
```

```
ages_steps = pd.get_dummies(bin_mapping).drop(1, axis=1)
ages_steps
```

	2	3	4
0	0	0	0
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
...	...	...	...
57	0	0	1
58	0	0	1
59	0	0	1
60	0	0	1
61	0	0	1

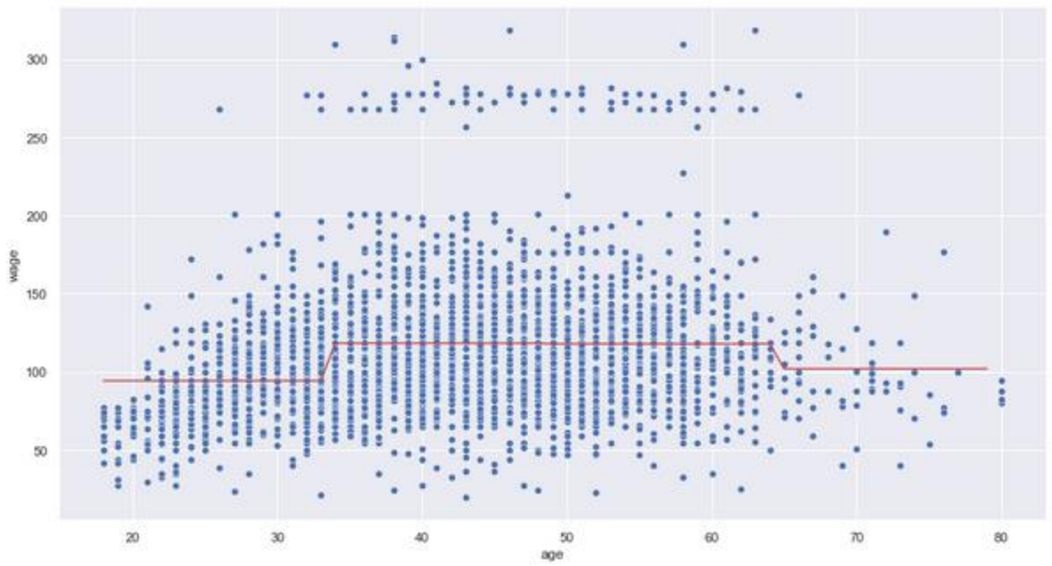
62 rows × 3 columns

Step Function

Wage Example

```
lregmodel = LinearRegression(fit_intercept = True)
lregmodel.fit(X,y = wage['wage'])
y_hat = lregmodel.predict(ages_steps)
sns.set(rc = {'figure.figsize':(15,8)})
sns.scatterplot(data = wage, x = 'age', y = 'wage')
sns.lineplot(x=ages, y=y_hat, color = 'r')
```

```
<AxesSubplot:xlabel='age', ylabel='wage'>
```



Step Functions - Summary

- Special case of a basis function where:

$$b_j(X_1) = I(c_j \leq X_1 \leq c_{j+1})$$

- Easy to work with and interpret
- Works well when there are natural cutpoints (or *knots*) to work with
- The discontinuous step function can introduce significant issues

Piecewise Polynomials

Overview

Instead of a single polynomial in X over its whole domain, we can rather use different polynomials in regions defined by knots. For example, a piecewise cubic polynomial with one knot at c:

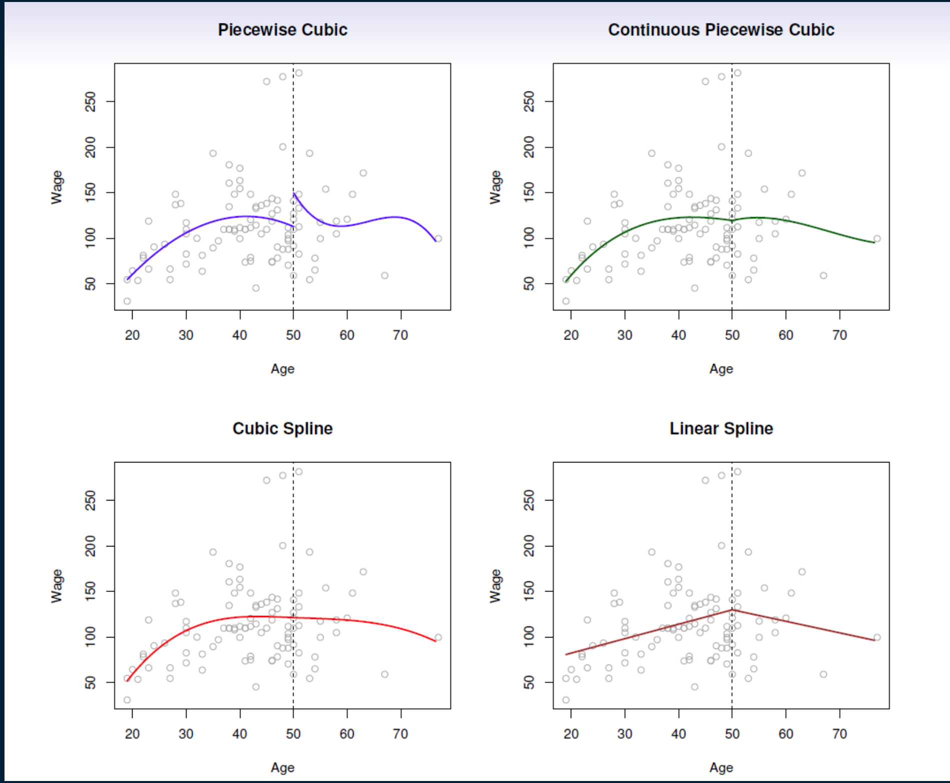
$$y_i = \begin{cases} \beta_{01} + \beta_{11}x_i + \beta_{21}{x_i}^2 + \beta_{31}x^3 + \epsilon_i & if \;\; x_i < c \\ \beta_{01} + \beta_{11}x_i + \beta_{21}{x_i}^2 + \beta_{31}x^3 + \epsilon_i & if \;\; x_i \geq c \end{cases}$$

- In general, we also add constraints to the polynomials for continuity

Piecewise Polynomial Examples

Discontinuity at the cut-point is undesirable

Adding a constraint that curve must be continuous



Adding constraints that first and second derivatives must be continuous

Piecewise polynomial of order 1

Piecewise Polynomials

Overview

- Piecewise polynomials improve on the "global" polynomial regression because we are able to use lower-order polynomials resulting in less risk of overfitting
  - *Note: the step function is thus a piecewise polynomial of degree 0*

Linear Splines

A linear spline with knots at $\xi_k, k = 1, \ldots K$ is a piecewise linear polynomial continuous at each knot

- We can represent this model with basis functions:

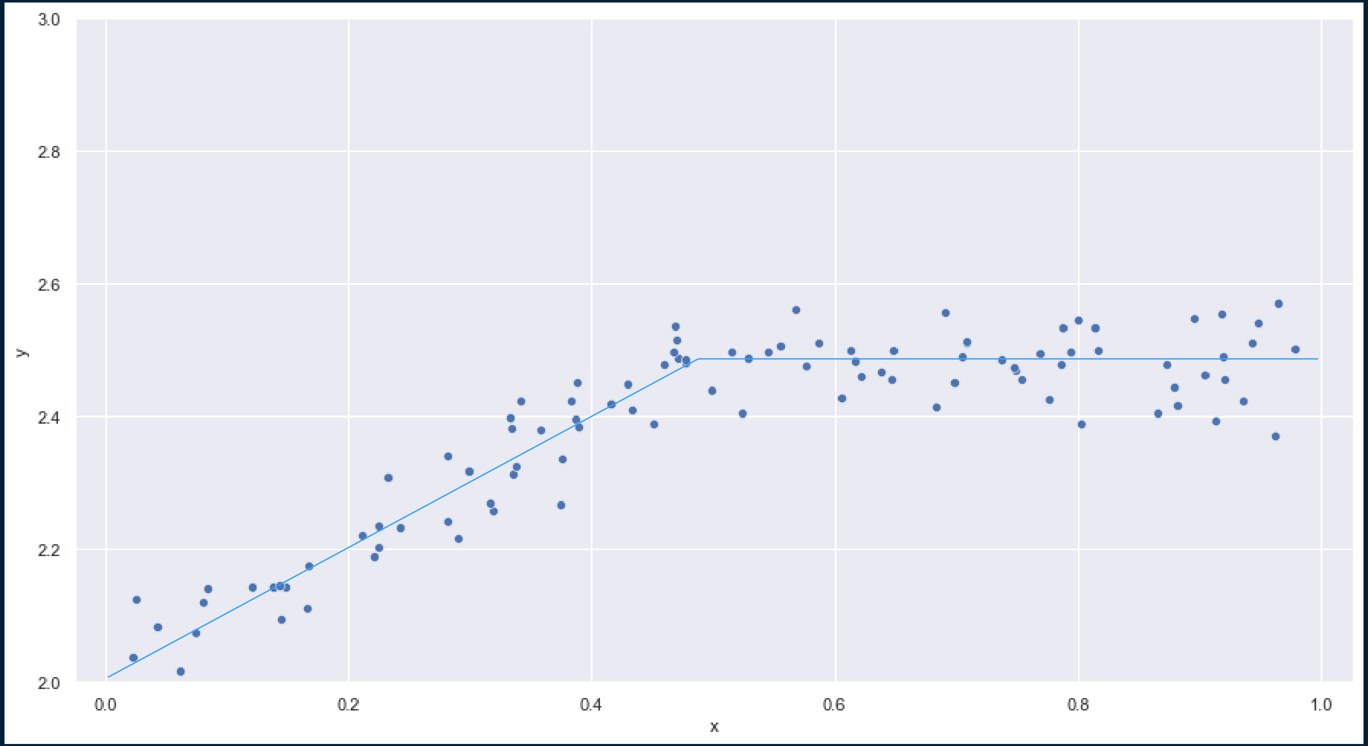
$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \cdots + \beta_{K+1} b_{K+1}(x_i) + \epsilon_i$$

where the b_k functions are basis functions:

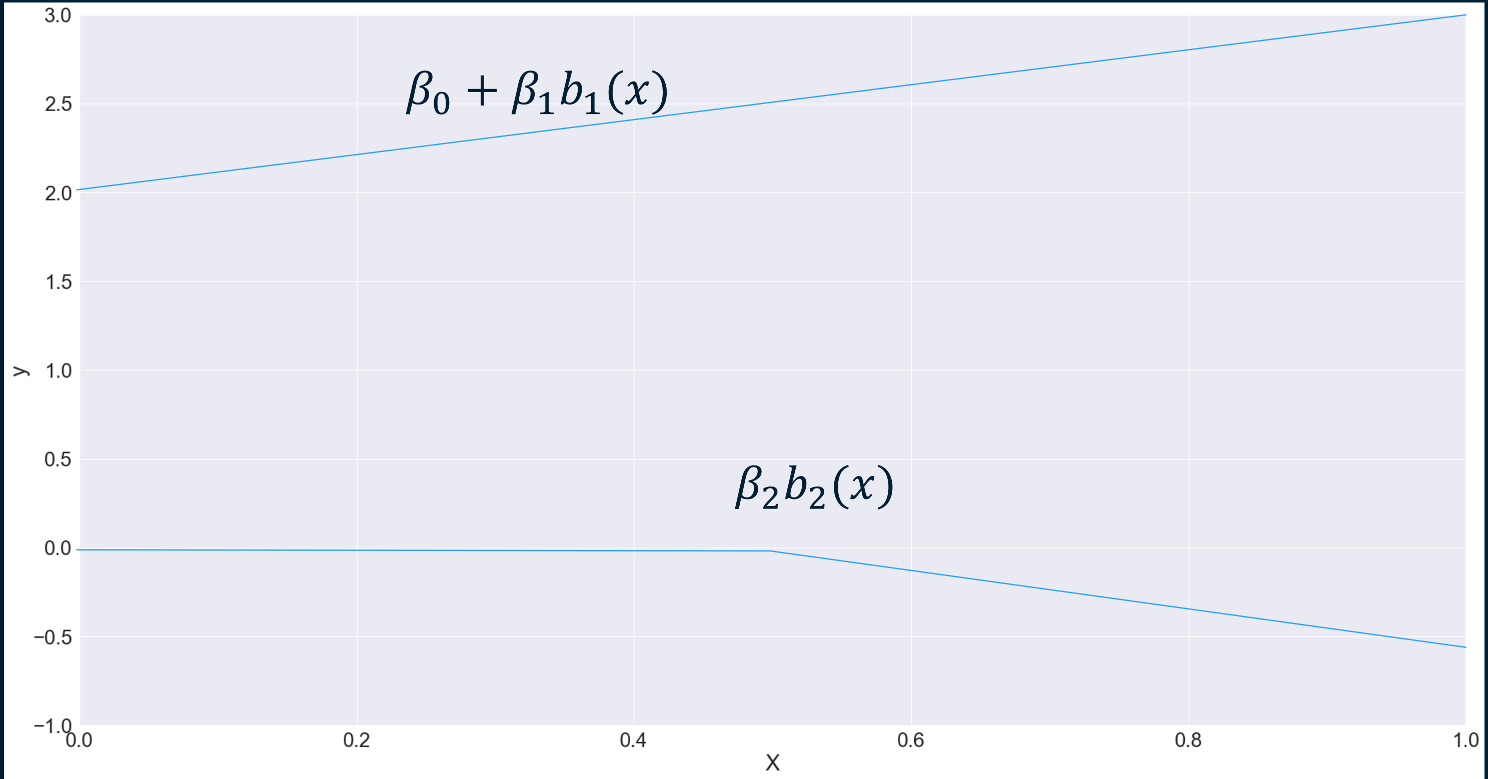
$$\begin{aligned} b_1(x_i) &= \ x_i \\ b_{k+1}(x_i) &= \ (x_i - \xi_k)_+, \qquad k = 1, \ldots, K \end{aligned}$$

"Positive part": returns 0 if value is negative

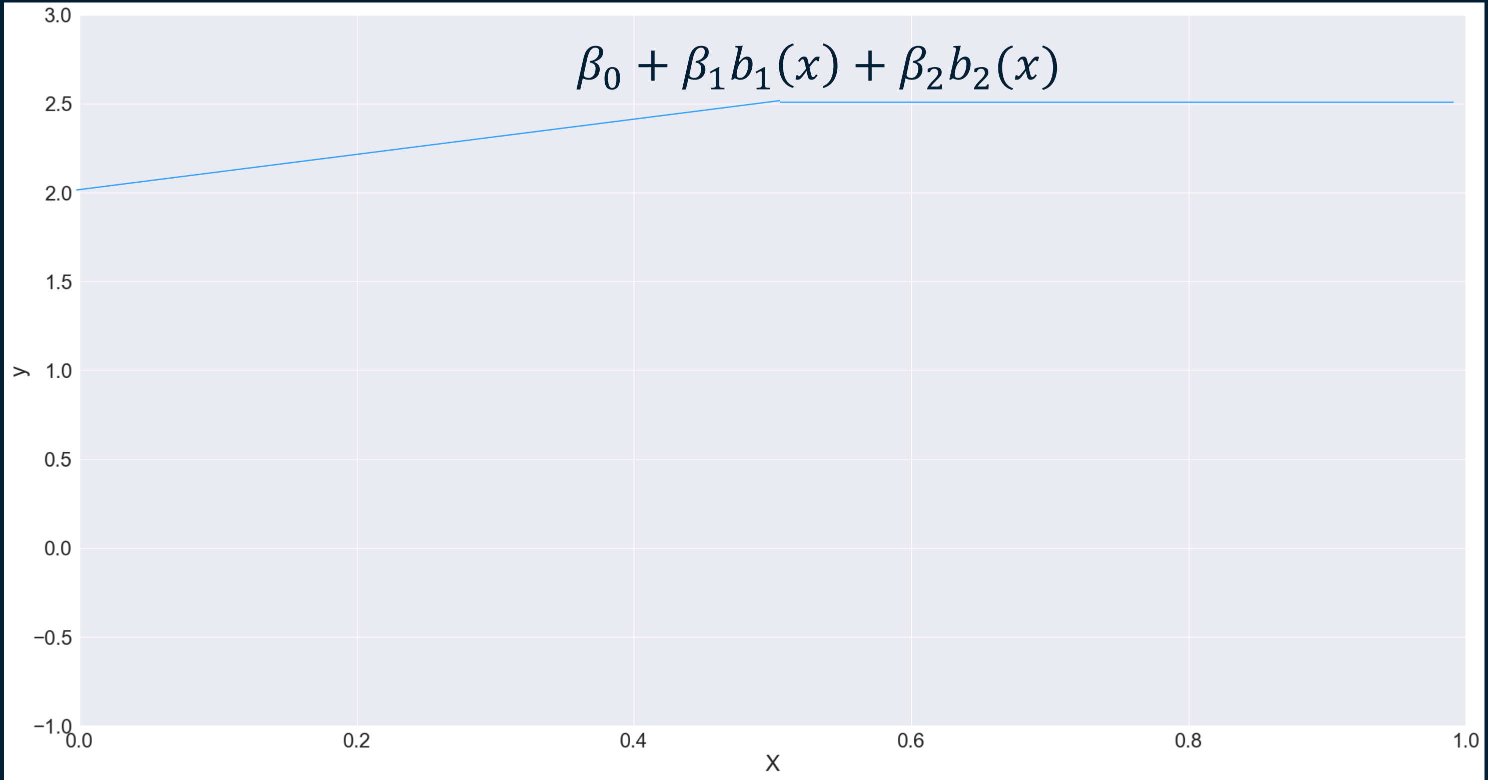
Linear Splines



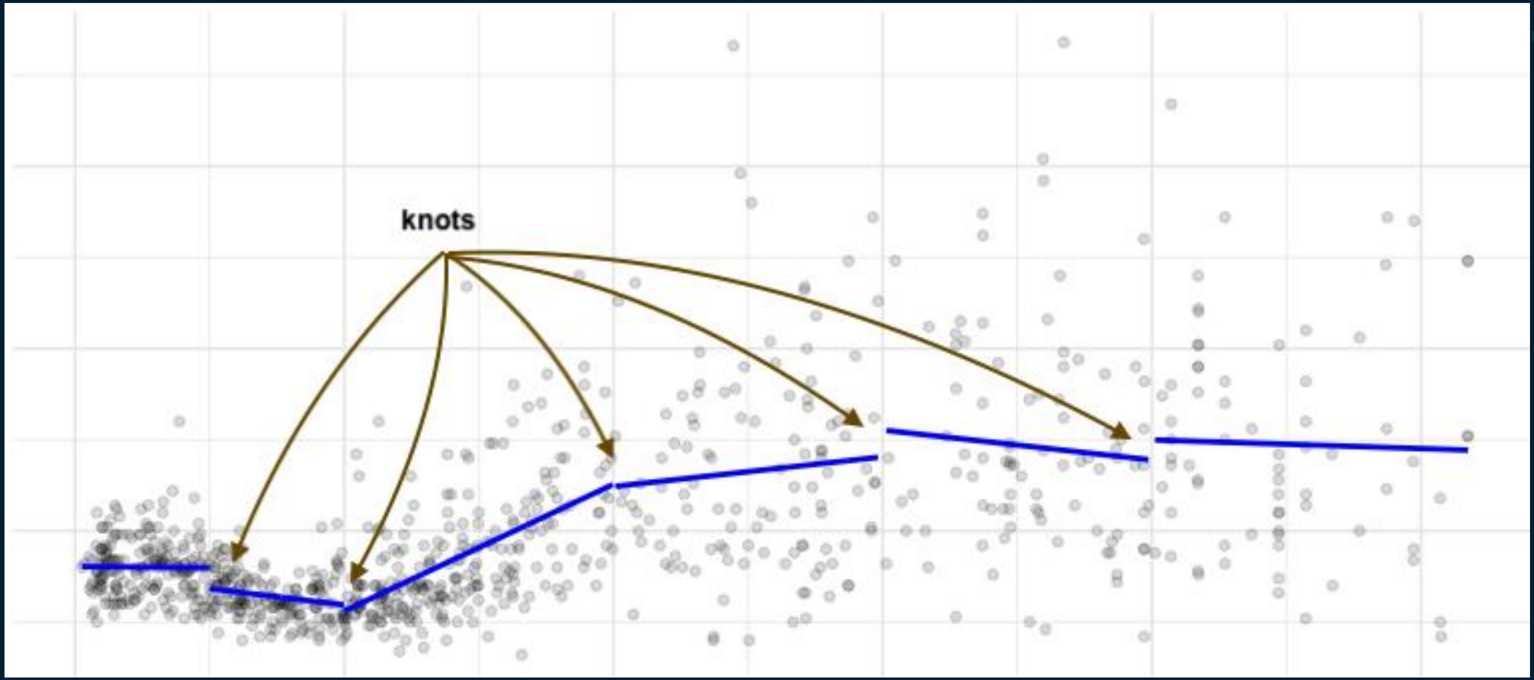
Linear Splines



Linear Splines



Piecewise Linear Regression Models



Cubic Splines

A linear cubic with knots at $\xi_k, k = 1, \dots K$ is a piecewise cubic polynomial with continuous derivatives up to order 2 at each knot

- Again, we can represent this model with truncated basis functions:

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \cdots + \beta_{K+3} b_{K+3}(x_i) + \epsilon_i$$

where:

$$b_1(x_i) = x_i$$
$$b_2(x_i) = {x_i}^2$$
$$b_3(x_i) = {x_i}^3$$
$$b_{k+3}(x_i) = {(x_i - \xi_k)_+}^3, \qquad k = 1, \dots, K$$

Cubic Splines

Wage Example

```
from statsmodels.gam.api import GLMGam, BSplines
bs = BSplines(wage[['age']], df=7, degree=3)
reg = GLMGam.from_formula('wage ~ age', wage, smoother=bs).fit()
reg.summary()
```

Generalized Linear Model Regression Results

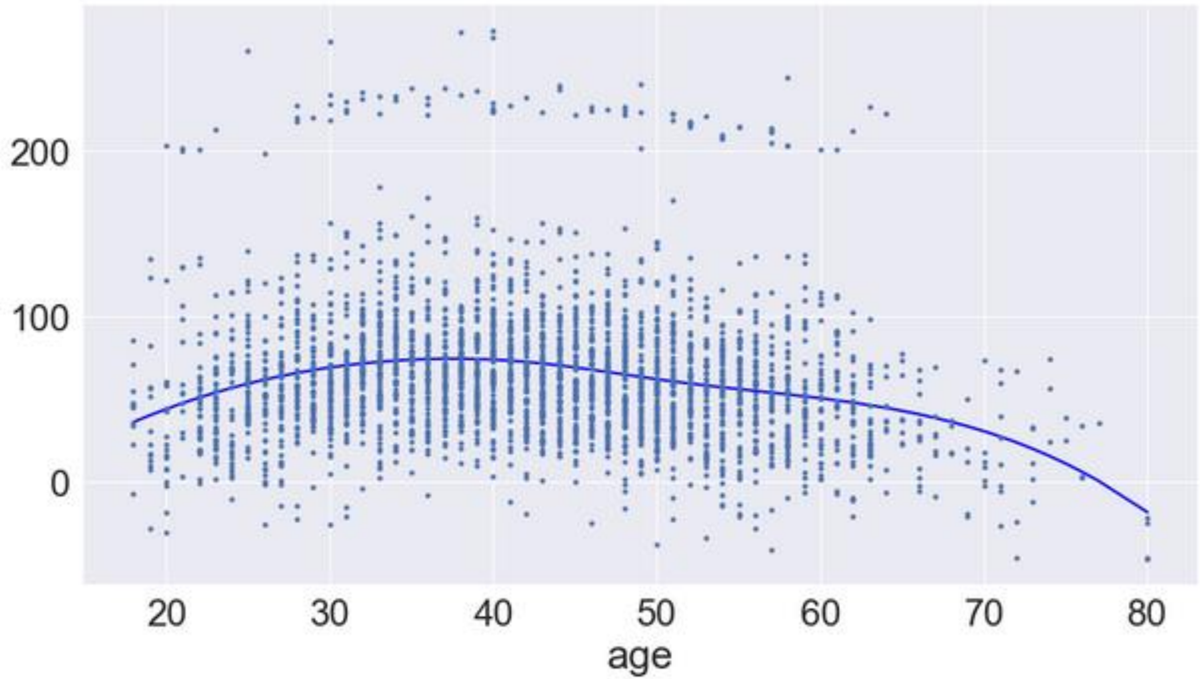
Dep. Variable:	wage	**No. Observations:**	3000
Model:	GLMGam	**Df Residuals:**	2993
Model Family:	Gaussian	**Df Model:**	6.00
Link Function:	identity	**Scale:**	1592.5
Method:	PIRLS	**Log-Likelihood:**	-15313.
Date:	Sun, 31 Jul 2022	**Deviance:**	4.7662e+06
Time:	15:10:45	**Pearson chi2:**	4.77e+06
No. Iterations:	3		
Covariance Type:	nonrobust		

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	36.1720	9.878	3.662	0.000	16.812	55.532
age	1.1190	0.161	6.945	0.000	0.803	1.435
age_s0	21.9493	11.786	1.862	0.063	-1.150	45.049
age_s1	39.2359	5.849	6.708	0.000	27.772	50.700
age_s2	38.6929	5.676	6.817	0.000	27.569	49.817
age_s3	11.4261	6.444	1.773	0.076	-1.204	24.056
age_s4	13.5710	12.406	1.094	0.274	-10.745	37.887
age_s5	-54.6265	10.772	-5.071	0.000	-75.739	-33.514

Cubic Splines

Wage Example

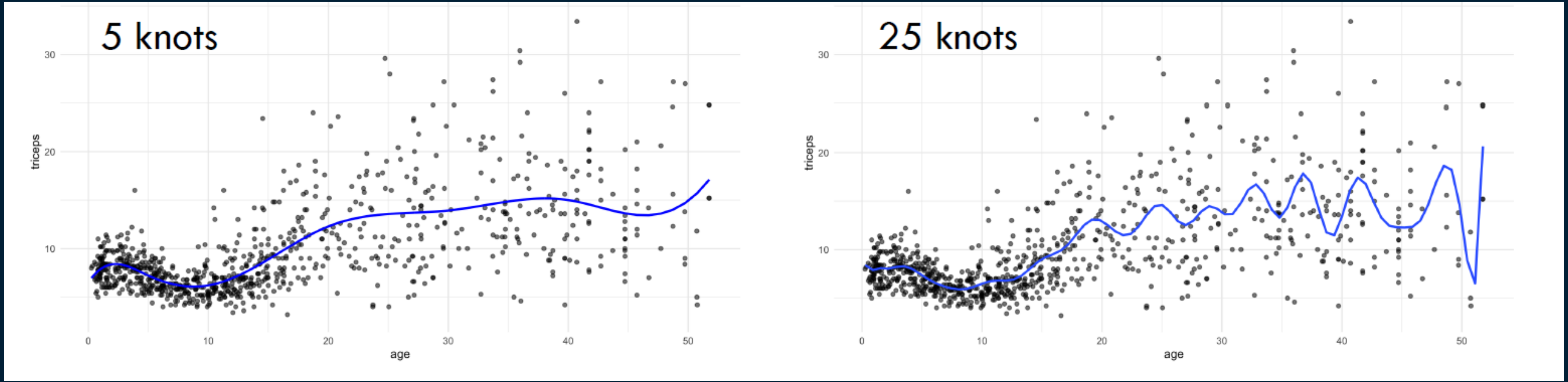
```
reg.plot_partial(0, cpr=True, plot_se=False)
```



Regression Splines

- Regression splines are defined to be piecewise polynomials there are continuous at the *knots* (cut-points). That is, the end points where segments meet are constrained to be equal (upper right)
- We can also add constraints that the first and second derivatives at the end points meet are also constrained to be equal, giving a smooth curve

Cubic Splines – Knot Selection



Knot Placement

- One strategy is to decide K, the number of knots, and then place them at appropriate quantiles of the observed X
- A cubic spline with K knots has K+4 parameters (or degrees of freedom)

Generalized Additive Models

Generalized Additive Models

Overview

- Extension of multiple linear regression models which replaces each linear component ($\beta_j x_{ij}$) with a smooth nonlinear function $f_j(x_{ij})$.
- Linear regression model then is generalized from this:

$$y_i = \beta_0 + \sum_{j=1}^{p} \beta_j x_{ij} + \epsilon_i$$

to this:

$$y_i = \beta_0 + \sum_{j=1}^{p} f_j(x_{ij}) + \epsilon_i$$

Generalized Additive Models

Overview

- Allows combination of regression splines, smoothing splines, and local regression models to deal with multiple predictors in one model
  - Modeler can decide which method to use for every feature

General Additive Models

Extending and Generalizing to Multi-Variable Models

$$y_i = \beta_0 + f_1(x_{i1}) + f_2(x_{i2}) + \cdots + f_p(x_{ip}) + \epsilon_i.$$