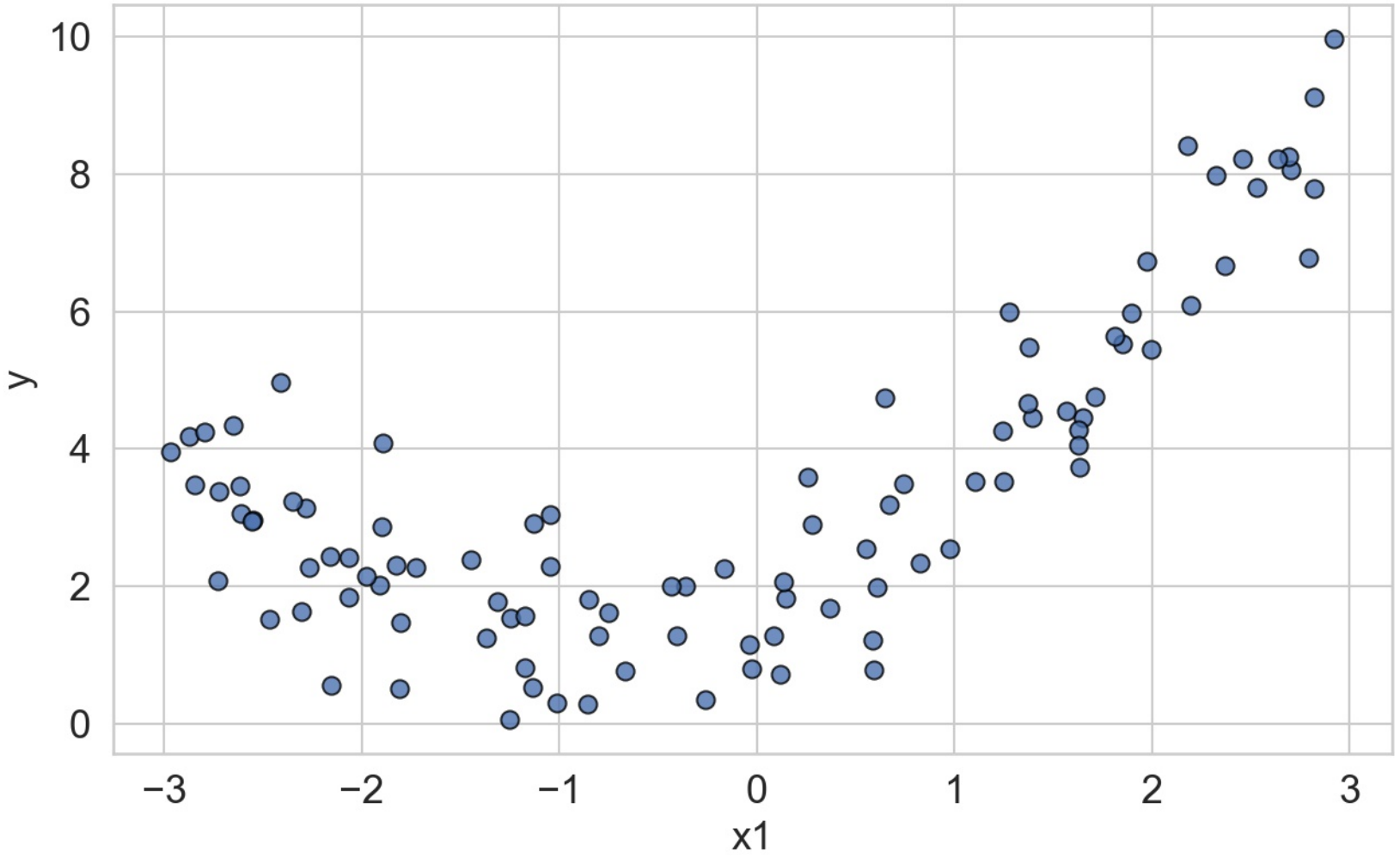
Aprendizado de Máquina e Reconhecimento de Padrões 2021.2

Polynomial Regression

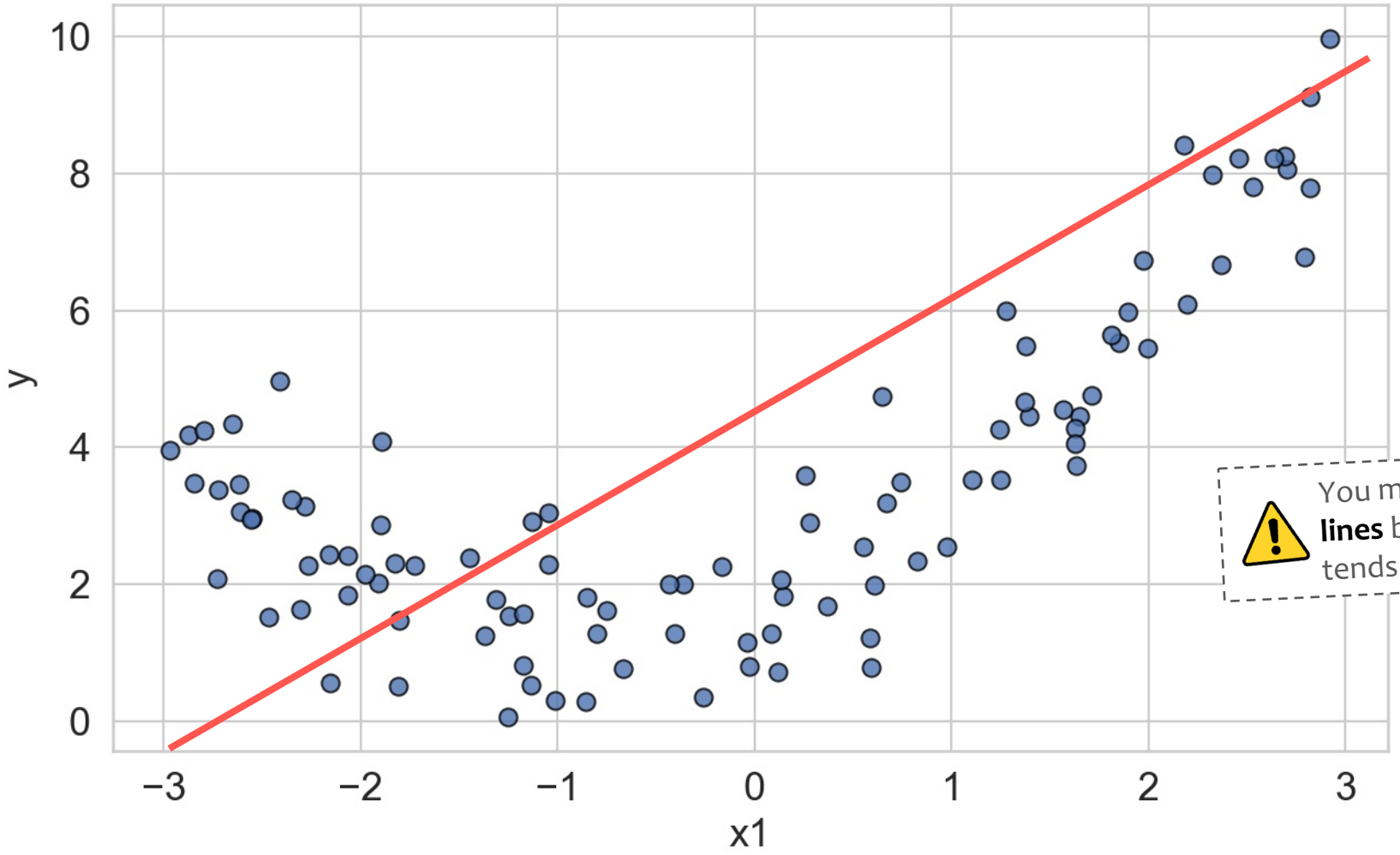
Prof. Dr. Samuel Martins (Samuka)
samuel.martins@ifsp.edu.br

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
São Paulo
Campus Campinas

What if your data is more complex than a straight line?

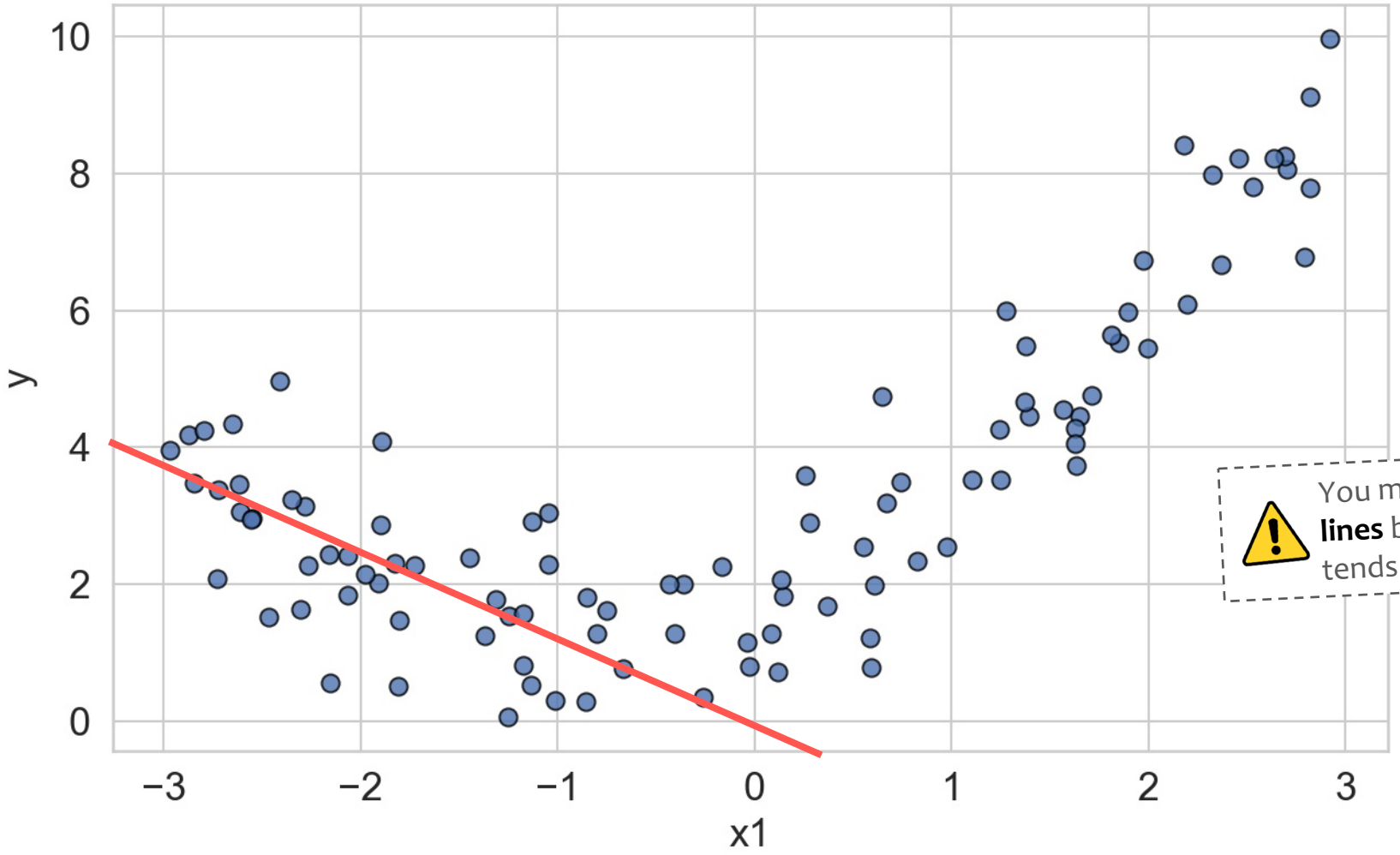


What if your data is more complex than a straight line?



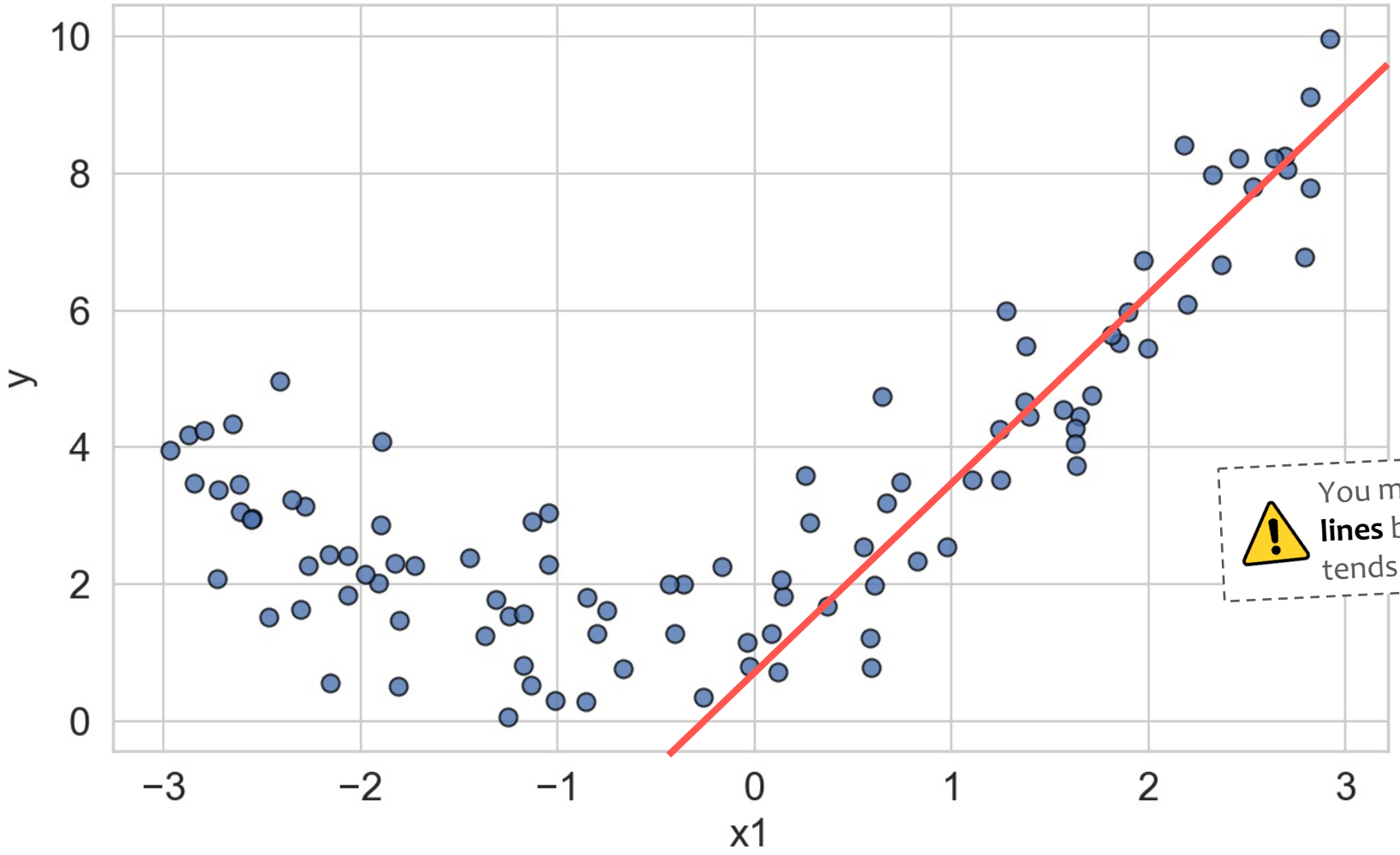
You may even try to **fit different lines** but the **prediction error** tends to be **high**.

What if your data is more complex than a straight line?



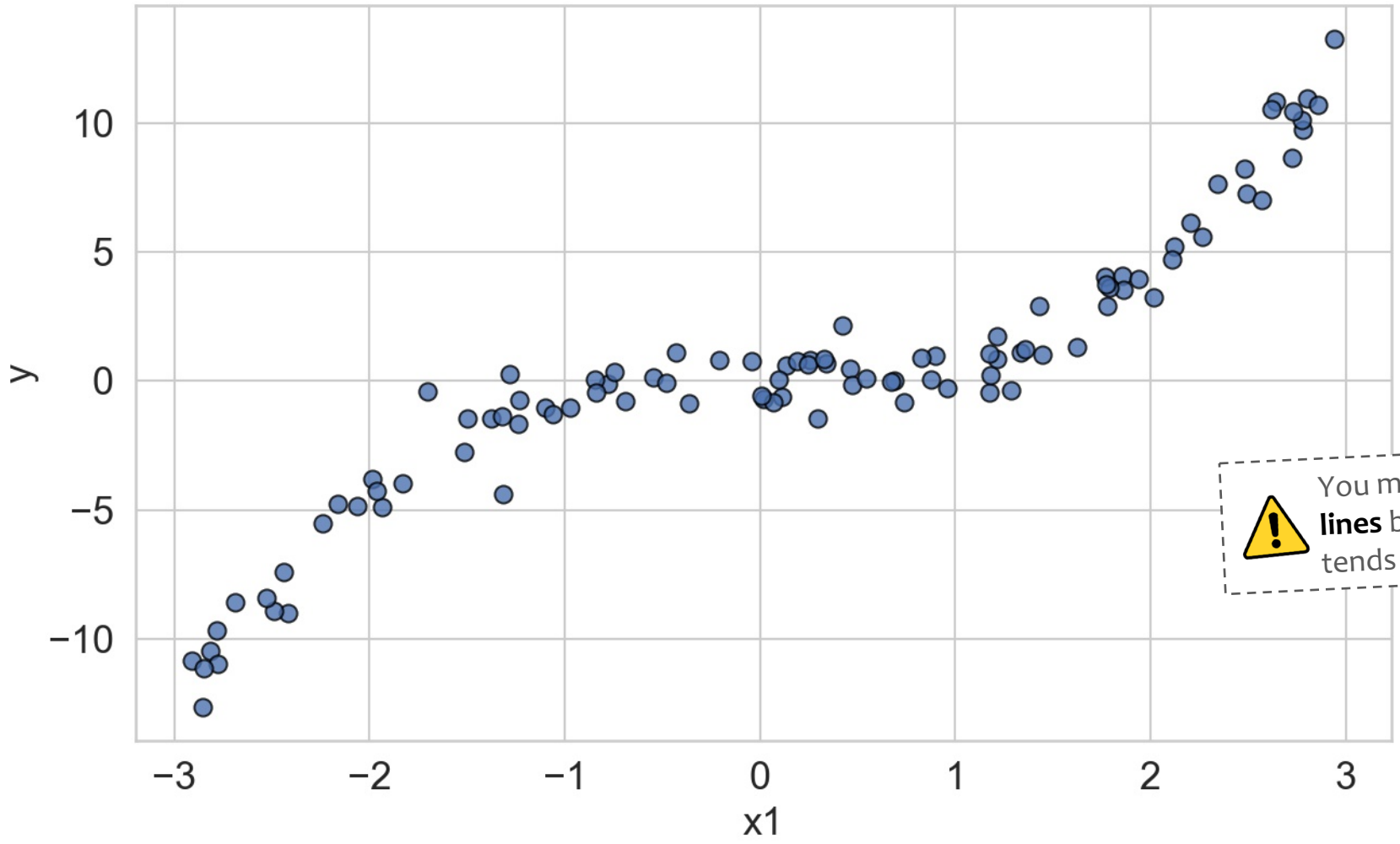
You may even try to **fit different lines** but the **prediction error** tends to be **high**.

What if your data is more complex than a straight line?



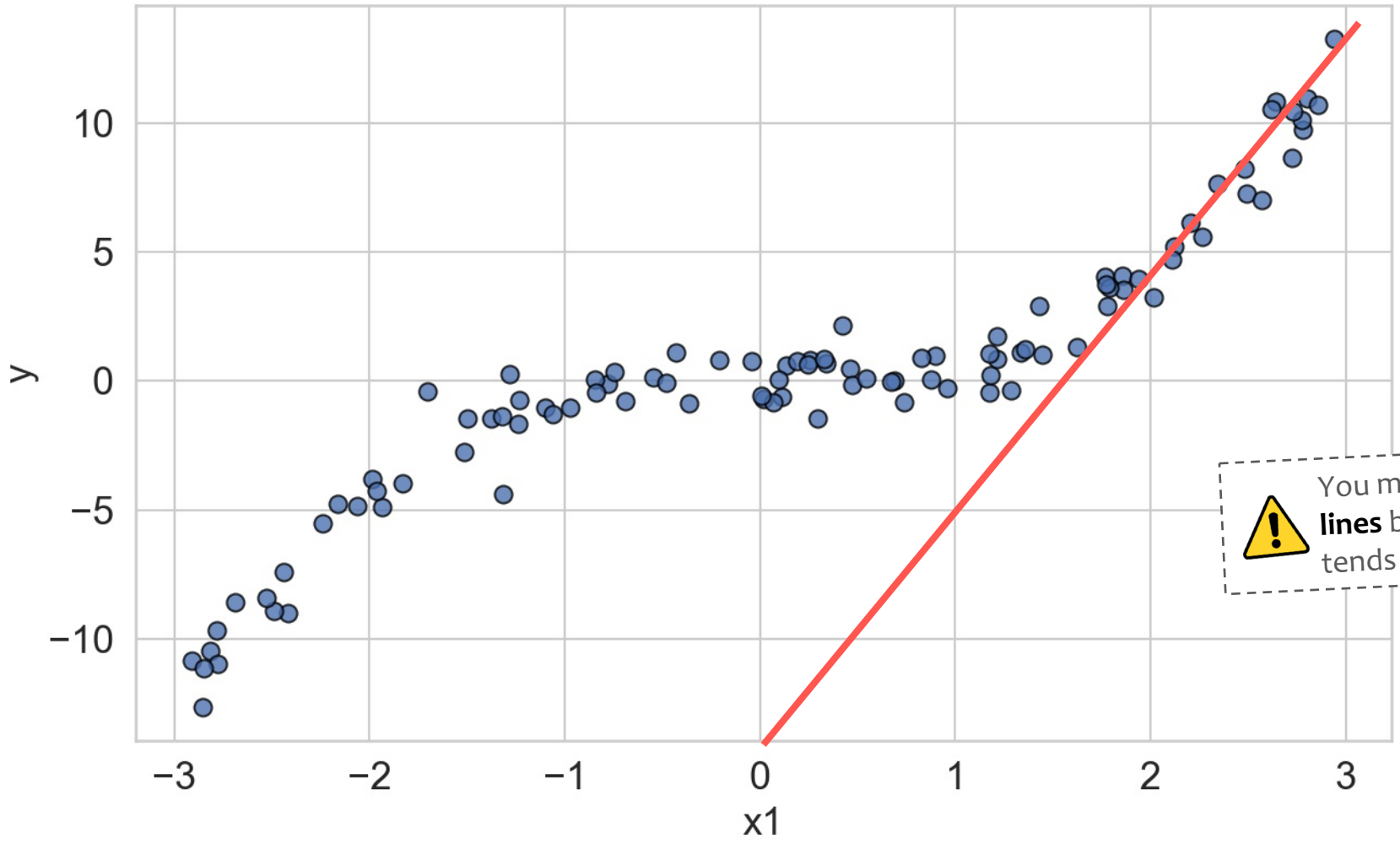
You may even try to **fit different lines** but the **prediction error** tends to be **high**.

What if your data is more complex than a straight line?



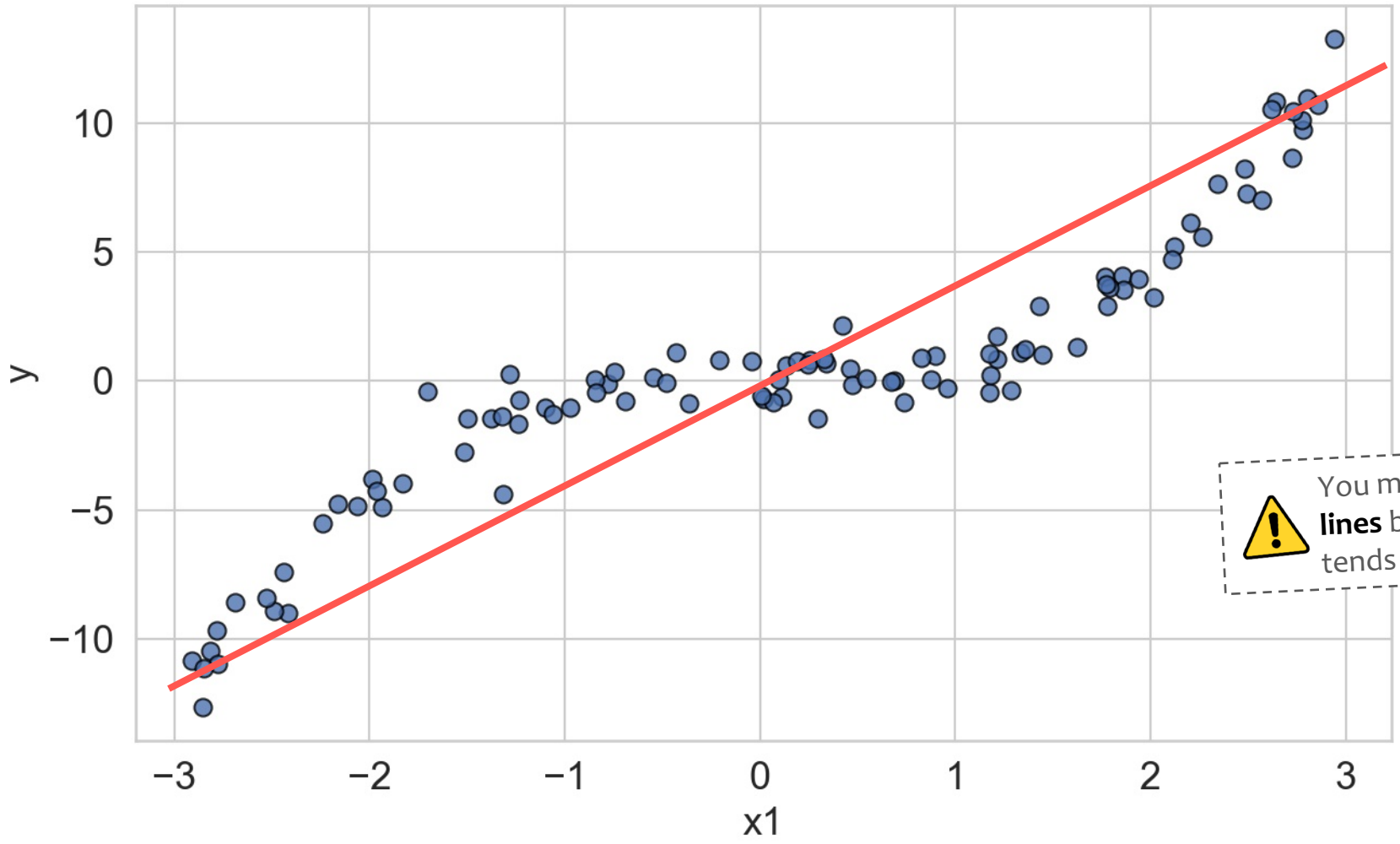
You may even try to **fit different lines** but the **prediction error** tends to be **high**.

What if your data is more complex than a straight line?



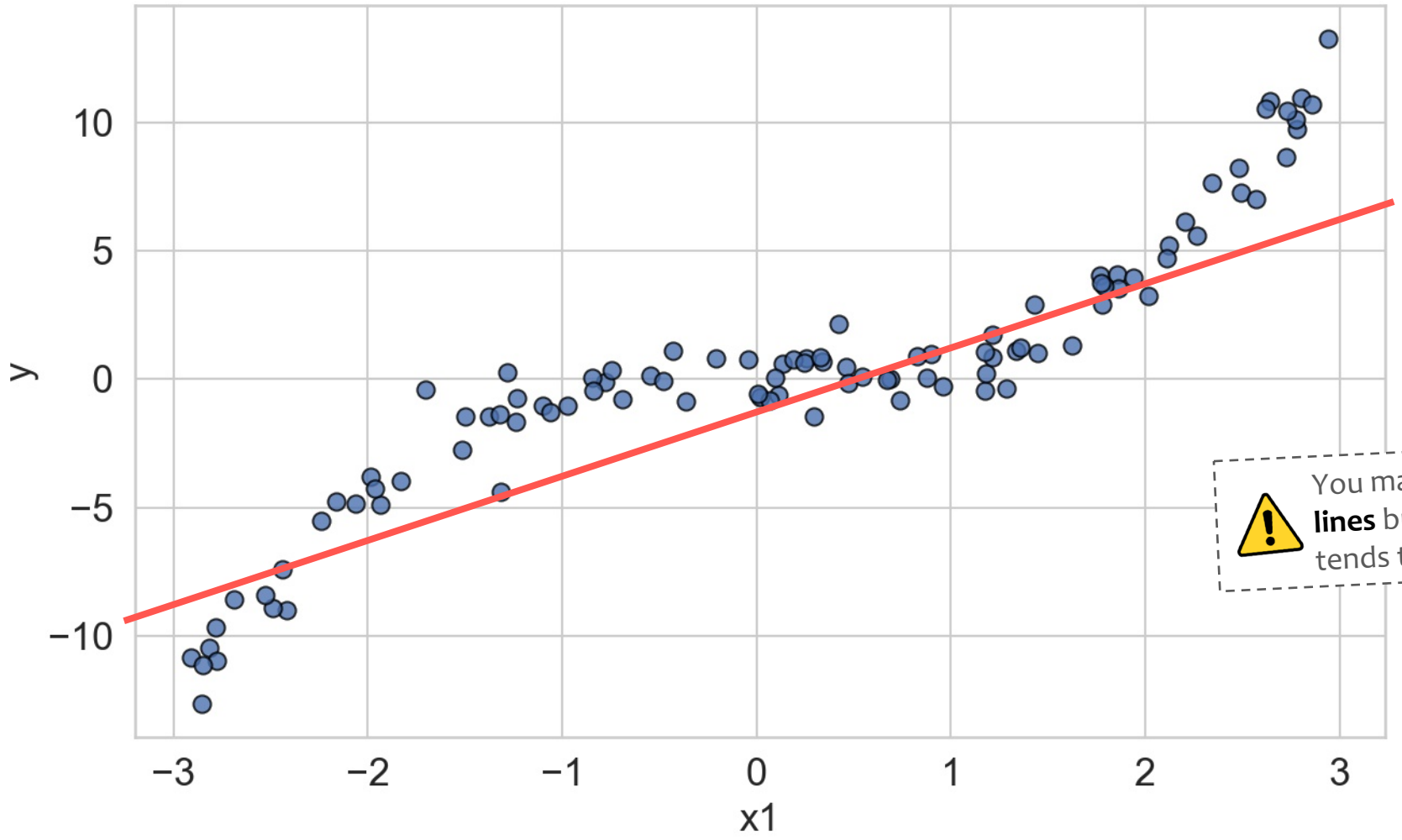
You may even try to **fit different lines** but the **prediction error** tends to be **high**.

What if your data is more complex than a straight line?



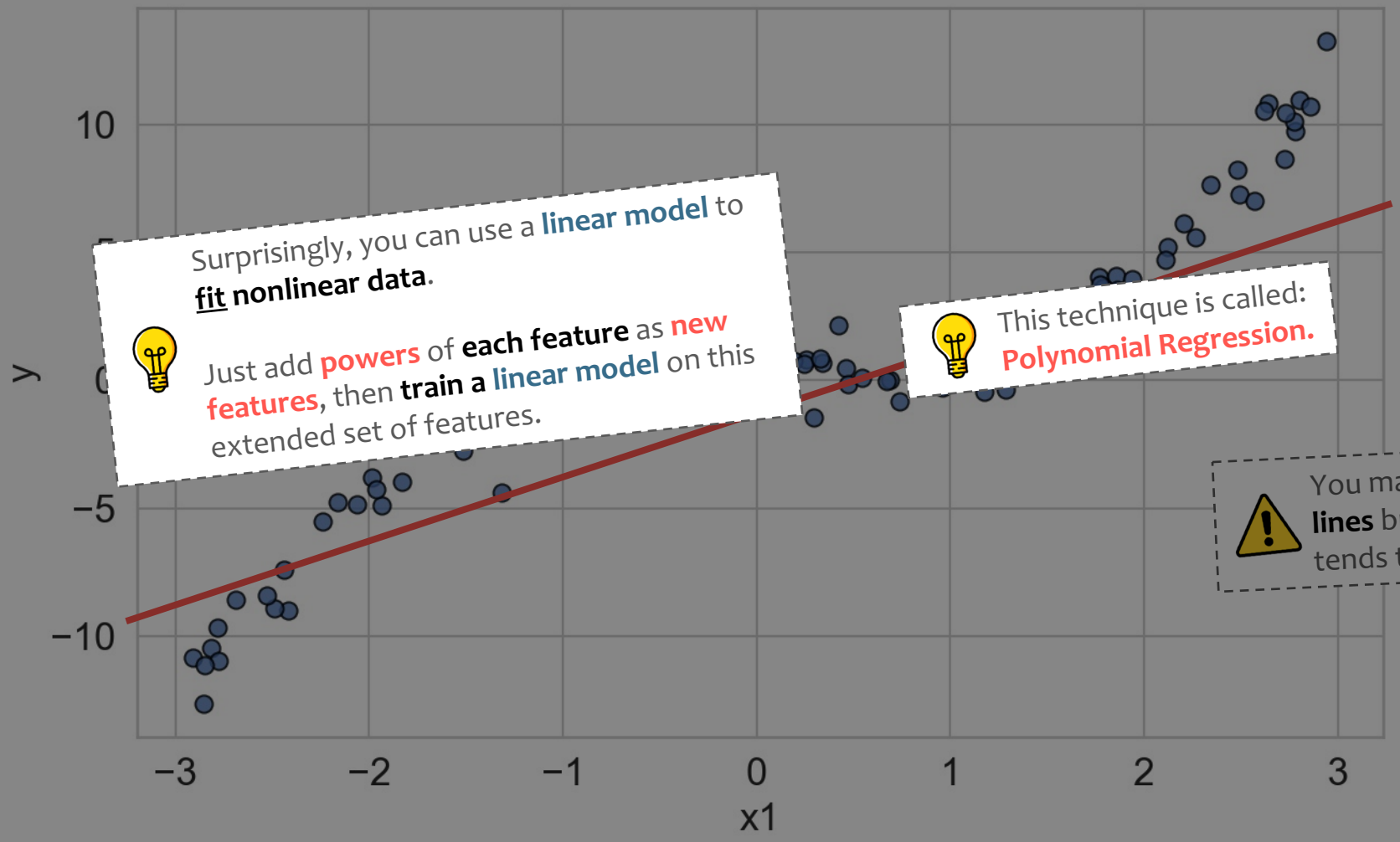
You may even try to **fit different lines** but the **prediction error** tends to be **high**.

What if your data is more complex than a straight line?



You may even try to **fit different lines** but the **prediction error** tends to be **high**.

What if your data is more complex than a straight line?



Surprisingly, you can use a **linear model** to **fit nonlinear data**.

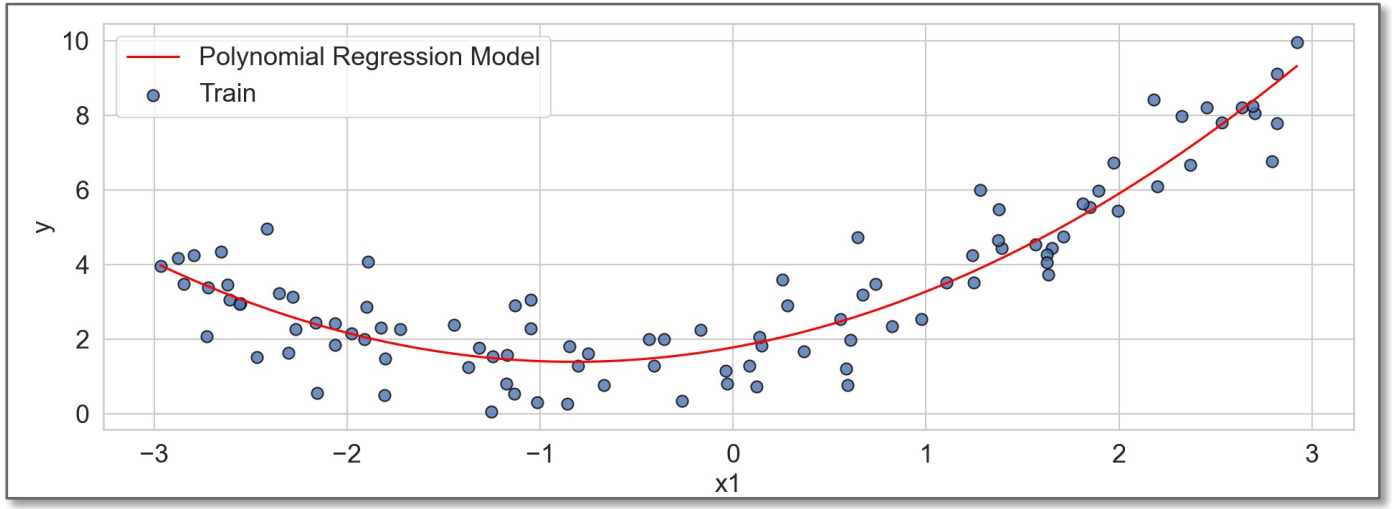
Just add **powers** of **each feature** as **new features**, then **train a linear model** on this extended set of features.

This technique is called: **Polynomial Regression.**

You may even try to **fit different lines** but the **prediction error** tends to be **high**.

Polynomial Regression

- It is a form of **regression analysis** in which **the relationship between** the **independent variables x** and the **dependent variable y** is **nonlinear** and can be modelled as an **n^{th} degree polynomial in x**.
- It is a special case of **linear regression** where we **fit** a **polynomial equation** on the data with a **curvilinear relationship** between the **dependent variable** and the **independent variables**.



Examples of nonlinear relationships between variables:

- The response to the dose of a drug is often nonlinear: doubling the dosage generally doesn't lead to a doubled response.
- The demand for a product isn't a linear function of marketing dollars spent; at some point, demand is likely to be saturated.

Simple Linear Regression

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1$$

Polynomial Regression

Simple Linear Regression

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1$$

Polynomial Regression

Degrees (d): 2

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2$$

Simple Linear Regression

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1$$

Polynomial Regression

Degrees (d): 2

polynomial features

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2$$

Polynomial features are those features created by **raising existing features to an exponent.**

Simple Linear Regression

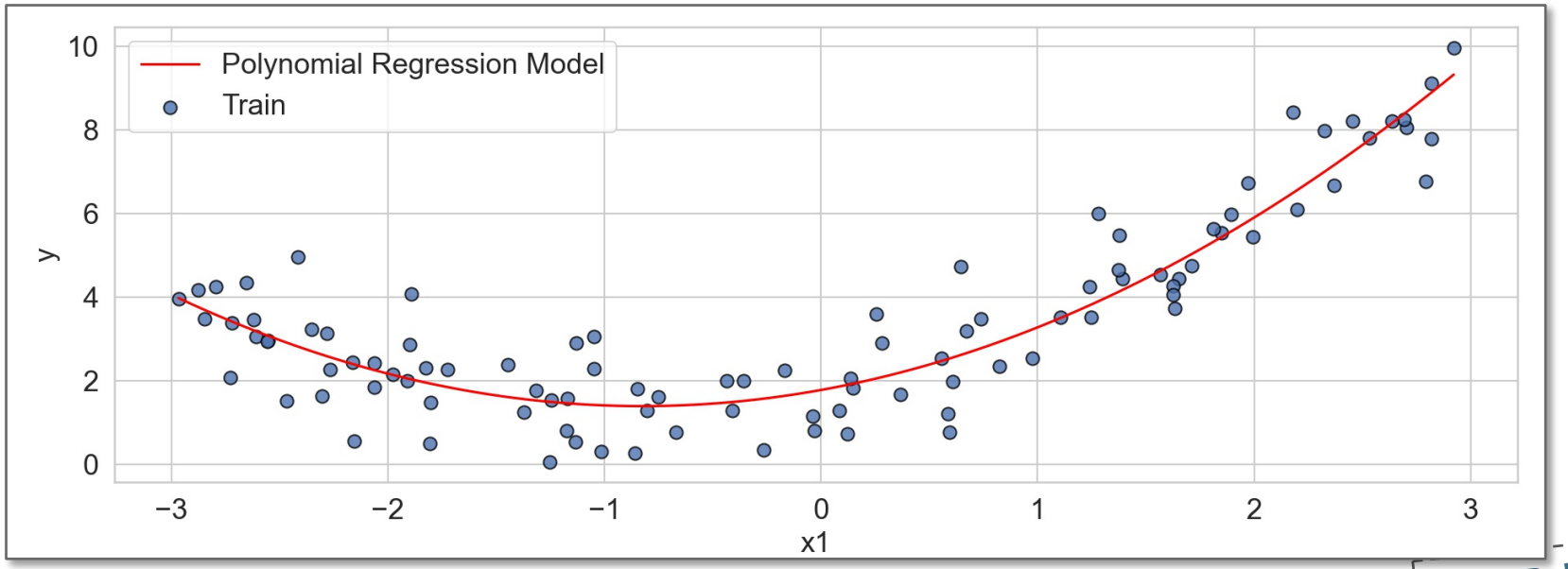
$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1$$

Polynomial Regression

Degrees (d): 2

polynomial features

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2$$



Polynomial features are those features created by **raising existing features to an exponent.**

Simple Linear Regression

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1$$

Polynomial Regression

Degrees (d): 2

polynomial features

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2$$

Degrees (d): 3

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_2 x_1^3$$

Polynomial features are those features created by **raising existing features to an exponent.**

Simple Linear Regression

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1$$

Polynomial Regression

Degrees (d): 2

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2$$

polynomial features

Degrees (d): 3

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_2 x_1^3$$

polynomial features

Polynomial features are those features created by **raising existing features to an exponent.**

Simple Linear Regression

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1$$

Polynomial Regression

Degrees (d): 2

polynomial features

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2$$

Degrees (d): 3

polynomial features

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_2 x_1^3$$

Degrees (d): d

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \cdots + \theta_d x_1^d$$

Polynomial features are those features created by **raising existing features to an exponent.**

Simple Linear Regression

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1$$

Polynomial Regression

Degrees (d): 2

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2$$

polynomial features

Degrees (d): 3

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_2 x_1^3$$

polynomial features

Degrees (d): d

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \cdots + \theta_d x_1^d$$

polynomial features

Polynomial features are those features created by **raising existing features to an exponent.**

Multiple Linear Regression

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

Polynomial Regression

Multiple Linear Regression

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

Polynomial Regression

Alternative 1

Select only **one promising independent variable x** whose relationship with the **dependent variable y** can be (or apparently be) modelled as an **n^{th} degree polynomial**.

Then, **fit** a **Linear Regression Model**.

Multiple Linear Regression

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

Polynomial Regression

Alternative 1

Select only **one promising independent variable x** whose relationship with the **dependent variable y** can be (or apparently be) modelled as an **n^{th} degree polynomial**.

Then, **fit** a **Linear Regression Model**.

Degrees (d): 2

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2$$

Degrees (d): 3

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_2 x_1^3$$

Multiple Linear Regression

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

Polynomial Regression

Alternative 2

Select a few **promising** **independent variables** and **fit** a **Linear Regression Model** for each one.

Pick that one with the **best score** in the **validation set**.

Multiple Linear Regression

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

Polynomial Regression

Alternative 2

Select a few **promising** **independent variables** and **fit** a **Linear Regression Model** for each one.

Pick that one with the **best score** in the **validation set**.

Degrees (d): 2

$$\hat{y} = h_{x1,\theta}(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2$$
$$\hat{y} = h_{x2,\theta}(\mathbf{x}) = \theta_0 + \theta_1 x_2 + \theta_2 x_2^2$$

Degrees (d): 3

$$\hat{y} = h_{x1,\theta}(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_2 x_1^3$$
$$\hat{y} = h_{x2,\theta}(\mathbf{x}) = \theta_0 + \theta_1 x_2 + \theta_2 x_2^2 + \theta_2 x_2^3$$

Multiple Linear Regression

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

Polynomial Regression

Alternative 3

Create a **new feature** by aggregating **other features** (if this makes sense, of course =D).

Then, **fit** a **Linear Regression Model** with this new feature.

For example:

$\mathbf{x1}$: frontage, $\mathbf{x2}$: depth

$\mathbf{x3} = \mathbf{x1} * \mathbf{x2}$ = frontage * depth (area)

Multiple Linear Regression

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

Polynomial Regression

Alternative 3

Create a **new feature** by aggregating **other features** (if this makes sense, of course =D).

Then, **fit** a **Linear Regression Model** with this new feature.

For example:

x1: frontage, **x2**: depth

$\mathbf{x3} = \mathbf{x1} * \mathbf{x2}$ = frontage * depth (area)

Degrees (d)**: 2**

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_3 + \theta_2 x_3^2$$

Degrees (d)**: 3**

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_3 + \theta_2 x_3^2 + \theta_2 x_3^3$$

Multiple Linear Regression

two features

$n = 2$

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

Multiple Polynomial Regression

`sklearn.preprocessing`.PolynomialFeatures

https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PolynomialFeatures.html

Alternative 4

Use **many or all features together**.

Multiple Linear Regression

two features

$n = 2$

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

Multiple Polynomial Regression

`sklearn.preprocessing`.PolynomialFeatures

https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PolynomialFeatures.html

Alternative 4

Use **many or all features together**.

Degrees (d): 2

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1 x_2 + \theta_5 x_2^2$$

Multiple Linear Regression

two features

$n = 2$

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

Multiple Polynomial Regression

`sklearn.preprocessing.PolynomialFeatures`

https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PolynomialFeatures.html

Alternative 4

Use **many or all features together**.

Degrees (d): 2

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1 x_2 + \theta_5 x_2^2$$

Degrees (d): 3

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1 x_2 + \theta_5 x_2^2 + \theta_6 x_1^3 + \theta_7 x_1^2 x_2 + \theta_8 x_1 x_2^2 + \theta_9 x_2^3$$

Multiple Linear Regression

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$

three features

$n = 3$

Multiple Polynomial Regression

Alternative 4

Use **many or all features together**.

`sklearn.preprocessing`.PolynomialFeatures

https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PolynomialFeatures.html

Multiple Linear Regression

three features

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$

$n = 3$

Multiple Polynomial Regression

Alternative 4

Use **many or all features together**.

`sklearn.preprocessing.PolynomialFeatures`

https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PolynomialFeatures.html

Degrees (d): 2

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_1^2 + \theta_5 x_1 x_2 + \theta_6 x_1 x_3 + \theta_7 x_2^2 + \theta_8 x_2 x_3 + \theta_9 x_3^2$$

Multiple Linear Regression

three features

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$

$n = 3$

Multiple Polynomial Regression

`sklearn.preprocessing.PolynomialFeatures`

https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PolynomialFeatures.html

Alternative 4

Use **many or all features together**.

Degrees (d): 2

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_1^2 + \theta_5 x_1 x_2 + \theta_6 x_1 x_3 + \theta_7 x_2^2 + \theta_8 x_2 x_3 + \theta_9 x_3^2$$

Degrees (d): 3

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_1^2 + \theta_5 x_1 x_2 + \theta_6 x_1 x_3 + \theta_7 x_2^2 +$$
$$\theta_8 x_2 x_3 + \theta_9 x_3^2 + \theta_{10} x_1^3 + \theta_{11} x_1^2 x_2 + \theta_{12} x_1^2 x_3 + \theta_{13} x_1 x_2^2 +$$
$$\theta_{14} x_1 x_2 x_3 + \theta_{15} x_1 x_3^2 + \theta_{16} x_2^3 + \theta_{17} x_2^2 x_3 + \theta_{18} x_2 x_3^2 + \theta_{19} x_3^3$$

Multiple Linear Regression

three features

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$

$n = 3$

Multiple Polynomial Regression

PolynomialFeatures(degree=d) transforms **an array** containing **n features** into an array containing $\frac{(n+d)!}{d!\,n!}$ **features** (including **a bias**).

Beware of the **combinatorial explosion** of the **number of features**!

Alternative 4

Use **many or all features together**.

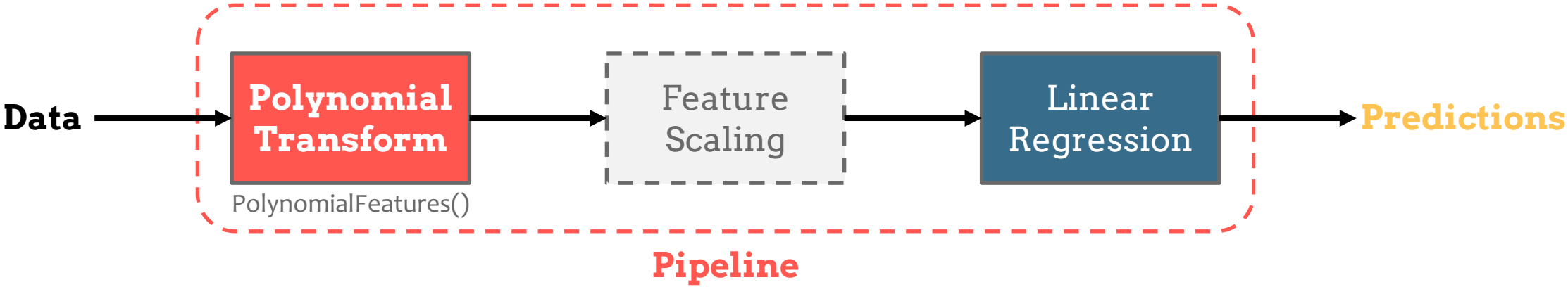
Degrees (d): 2

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_1^2 + \theta_5 x_1 x_2 + \theta_6 x_1 x_3 + \theta_7 x_2^2 + \theta_8 x_2 x_3 + \theta_9 x_3^2$$

Degrees (d): 3

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_1^2 + \theta_5 x_1 x_2 + \theta_6 x_1 x_3 + \theta_7 x_2^2 +$$
$$\theta_8 x_2 x_3 + \theta_9 x_3^2 + \theta_{10} x_1^3 + \theta_{11} x_1^2 x_2 + \theta_{12} x_1^2 x_3 + \theta_{13} x_1 x_2^2 +$$
$$\theta_{14} x_1 x_2 x_3 + \theta_{15} x_1 x_3^2 + \theta_{16} x_2^3 + \theta_{17} x_2^2 x_3 + \theta_{18} x_2 x_3^2 + \theta_{19} x_3^3$$

Polynomial Regression in Scikit-Learn



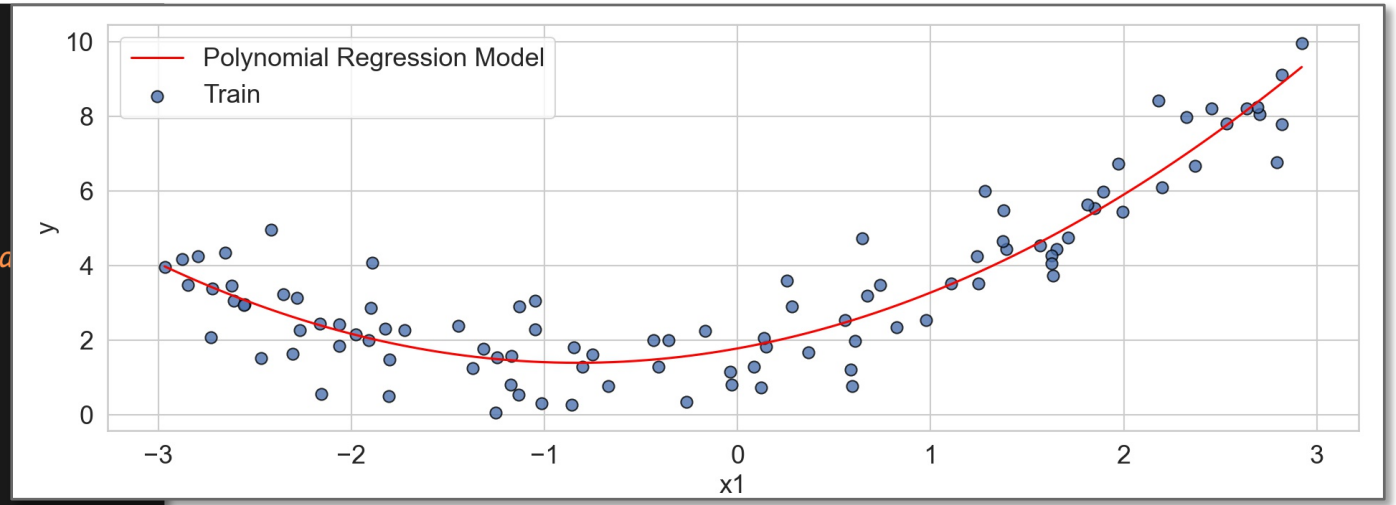
Polynomial Regression in Scikit-Learn

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression

poly_feat_transformer = PolynomialFeatures(degree=d, include_bias=False)
std_scaler = StandardScaler()
lin_reg = LinearRegression()

polynomial_regression = Pipeline([
    ("poly_feat_transformer", poly_feat_transformer),
    ("std_scaler", std_scaler),
    ("lin_reg", lin_reg),
])


# Fit all the transforms one after the other and transform the data,
# then fit the transformed data using the final estimator.
polynomial_regression.fit(X_train, y_train)

# apply transforms to the data, and predict with the final estimator
y_val_pred = polynomial_regression.predict(X_test)
```

Polynomial Regression in Scikit-Learn

$n = 1; d = 2$ quadratic data

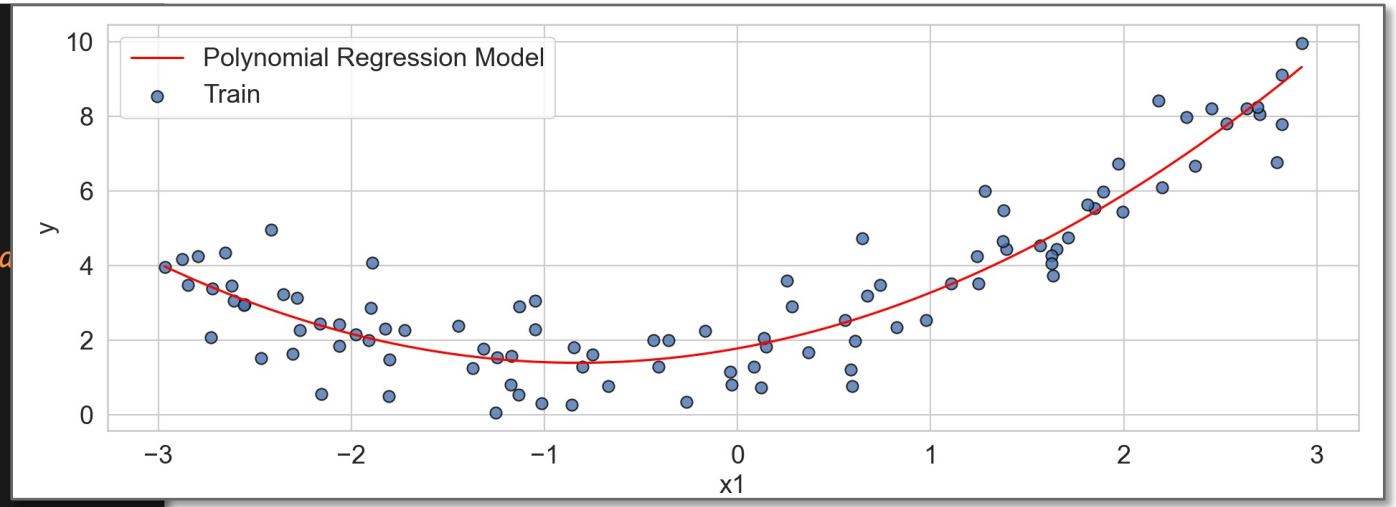
```python
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression

poly_feat_transformer = PolynomialFeatures(degree=d, include_bia
std_scaler = StandardScaler()
lin_reg = LinearRegression()

polynomial_regression = Pipeline([
    ("poly_feat_transformer", poly_feat_transformer),
    ("std_scaler", std_scaler),
    ("lin_reg", lin_reg),
])


# Fit all the transforms one after the other and transform the data,
# then fit the transformed data using the final estimator.
polynomial_regression.fit(X_train, y_train)

# apply transforms to the data, and predict with the final estimator
y_val_pred = polynomial_regression.predict(X_test)
```



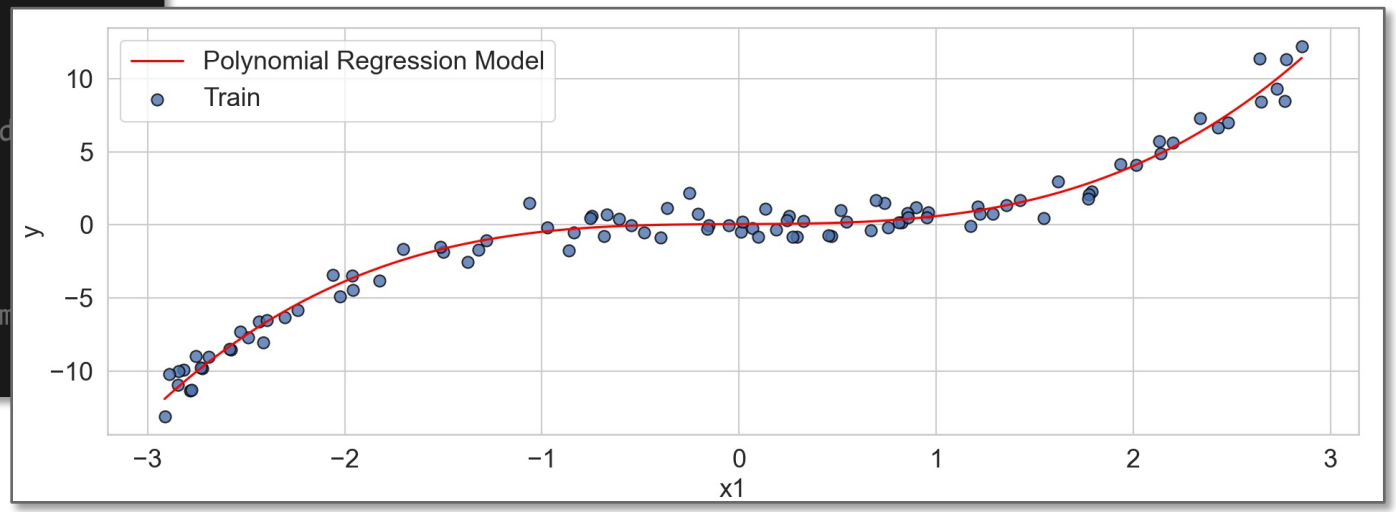
Polynomial Regression in Scikit-Learn

```python
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression

poly_feat_transformer = PolynomialFeatures(degree=d, include_bia
std_scaler = StandardScaler()
lin_reg = LinearRegression()

polynomial_regression = Pipeline([
    ("poly_feat_transformer", poly_feat_transformer),
    ("std_scaler", std_scaler),
    ("lin_reg", lin_reg),
])


# Fit all the transforms one after the other and transform the d
# then fit the transformed data using the final estimator.
polynomial_regression.fit(X_train, y_train)

# apply transforms to the data, and predict with the final estim
y_val_pred = polynomial_regression.predict(X_test)
```

$n = 1; d = 2$ quadratic data



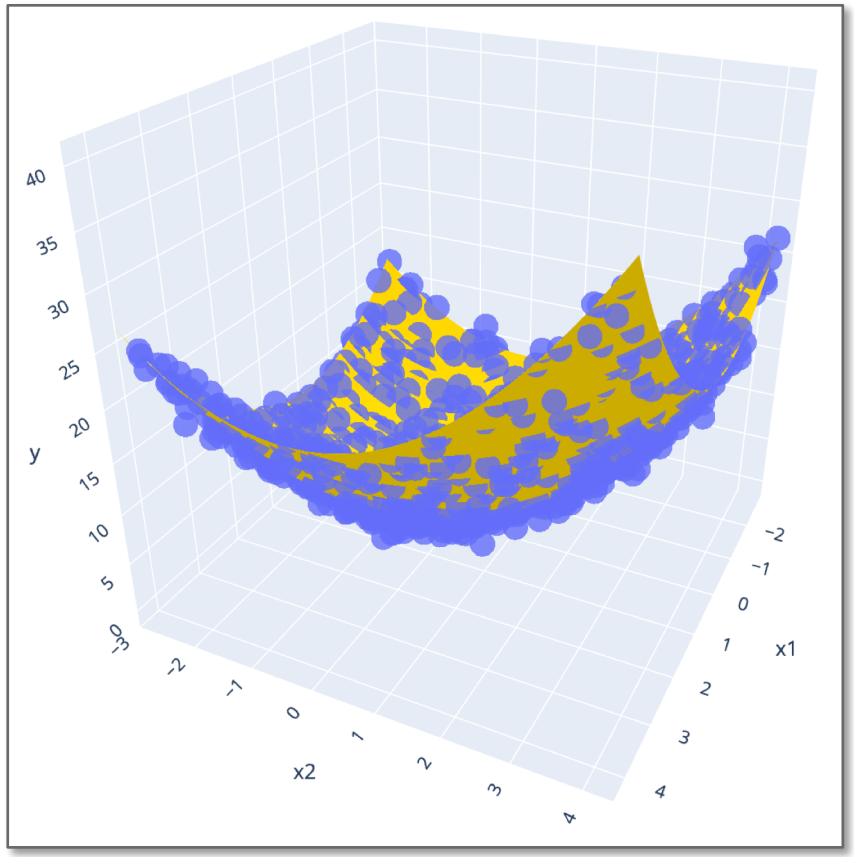
$n = 1; d = 3$ cubic data



Polynomial Regression in Scikit-Learn

```python
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression

poly_feat_transformer = PolynomialFeatures(degree=d, include_bias=False)
std_scaler = StandardScaler()
lin_reg = LinearRegression()

polynomial_regression = Pipeline([
    ("poly_feat_transformer", poly_feat_transformer),
    ("std_scaler", std_scaler),
    ("lin_reg", lin_reg),
])


# Fit all the transforms one after the other and transform the data,
# then fit the transformed data using the final estimator.
polynomial_regression.fit(X_train, y_train)

# apply transforms to the data, and predict with the final estimator
y_val_pred = polynomial_regression.predict(X_test)
```

$n = 2; d = 2$ quadratic data

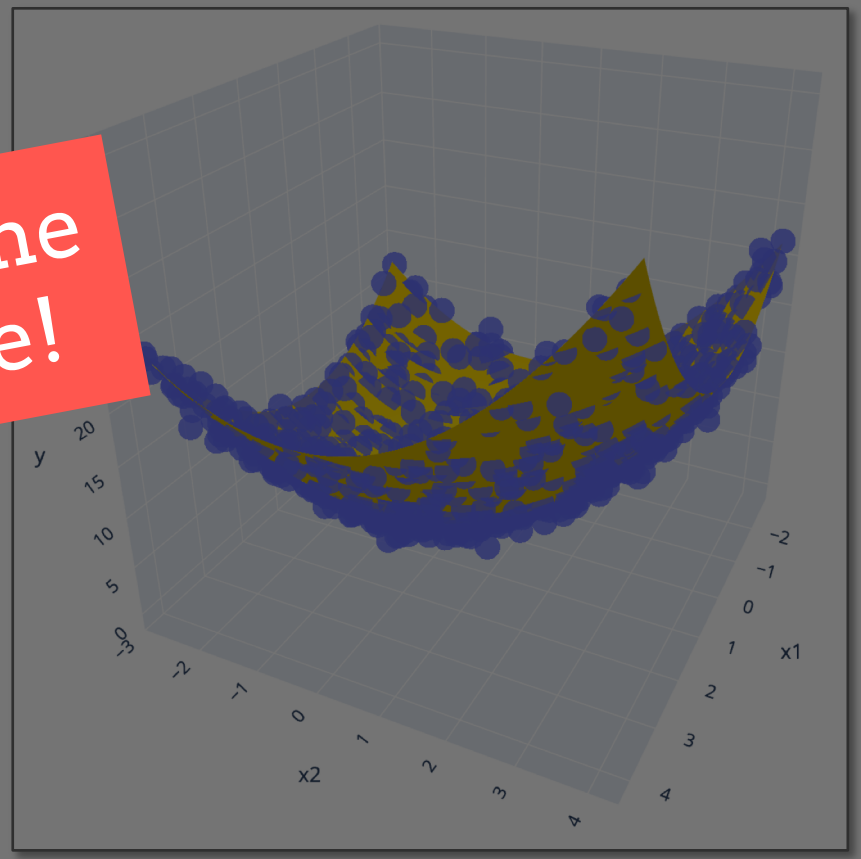


Polynomial Regression in Scikit-Learn

```python
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression

poly_feat_transformer = PolynomialFeatures(degree=d, include_bias=False)
std_scaler = StandardScaler()
lin_reg = LinearRegression()

polynomial_regression = Pipeline([
    ("poly_feat_transformer", poly_feat_transformer),
    ("std_scaler", std_scaler),
    ("lin_reg", lin_reg),
])


# Fit all the transforms one after the other and transform the data,
# then fit the transformed data using the final estimator.
polynomial_regression.fit(X_train, y_train)

# apply transforms to the data, and predict with the final estimator
y_val_pred = polynomial_regression.predict(X_test)
```

Show me the (live) code!

$n = 2; d = 2$ quadratic data



Effect of Polynomial Degree

- The **degree of the polynomial (d)** **dramatically increases** the number of **polynomial features** → $\frac{(n+d)!}{d!\,n!} - 1$
- For example: n=30 (input features)
  - **Degree** = 2, **Polynomial Feats =** 65
  - **Degree** = 3, **Polynomial Feats =** 285
  - …
  - **Degree** = 10, **Polynomial Feats =** 184755
  - **Degree** = 50, **Polynomial Feats =** 75394027565

Effect of Polynomial Degree

- The **degree of the polynomial (d)** **dramatically increases** the number of **polynomial features** → $\frac{(n+d)!}{d!\,n!} - 1$
- For example: n=30 (input features)
  - **Degree** = 2, **Polynomial Feats =** 65
  - **Degree** = 3, **Polynomial Feats =** 285
  - …
  - **Degree** = 10, **Polynomial Feats =** 184755
  - **Degree** = 50, **Polynomial Feats =** 75394027565

A problem with **Multivariate Polynomial Regression** is the **Multicollinearity**. When there are **multiple regression variables**, there are high chances that the **variables are dependent** on each other [1].

[1] Sinha, Priyanka. "Multivariate polynomial regression in data mining: methodology, problems and solutions." International Journal of Scientific and Engineering Research 4, no. 12 (2013): 962-965.

Effect of Polynomial Degree

- We need to choose it wisely!
- **High polynomial degrees** tends to **overfit** the data → bad results;
- **Low polynomial degrees** tends to **underfit** the data → bad results;

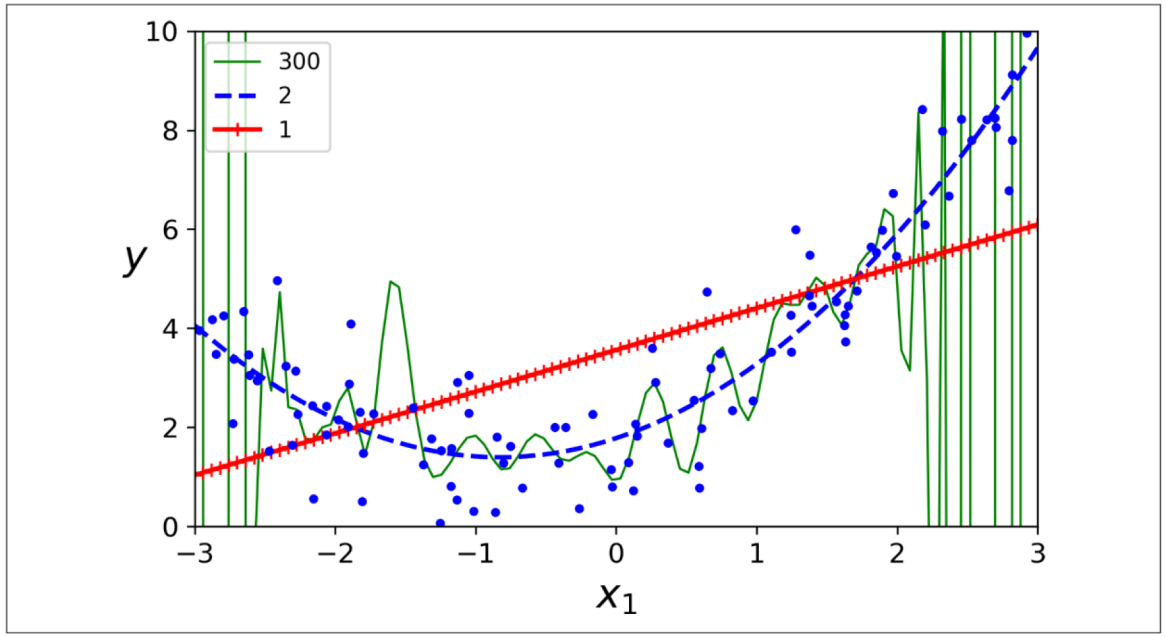
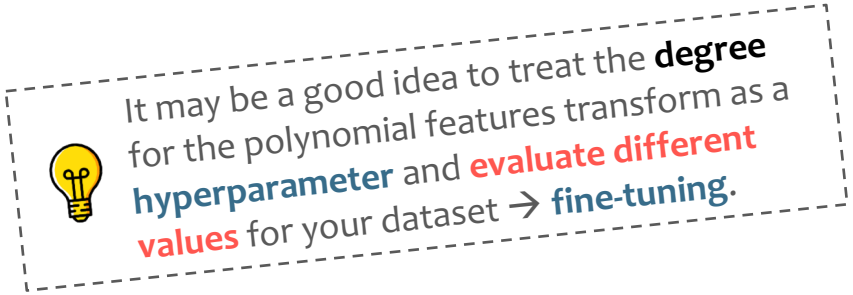


Figure 4-14. High-degree Polynomial Regression

Effect of Polynomial Degree

- We need to choose it wisely!
- **High polynomial degrees** tends to **overfit** the data → bad results;
- **Low polynomial degrees** tends to **underfit** the data → bad results;



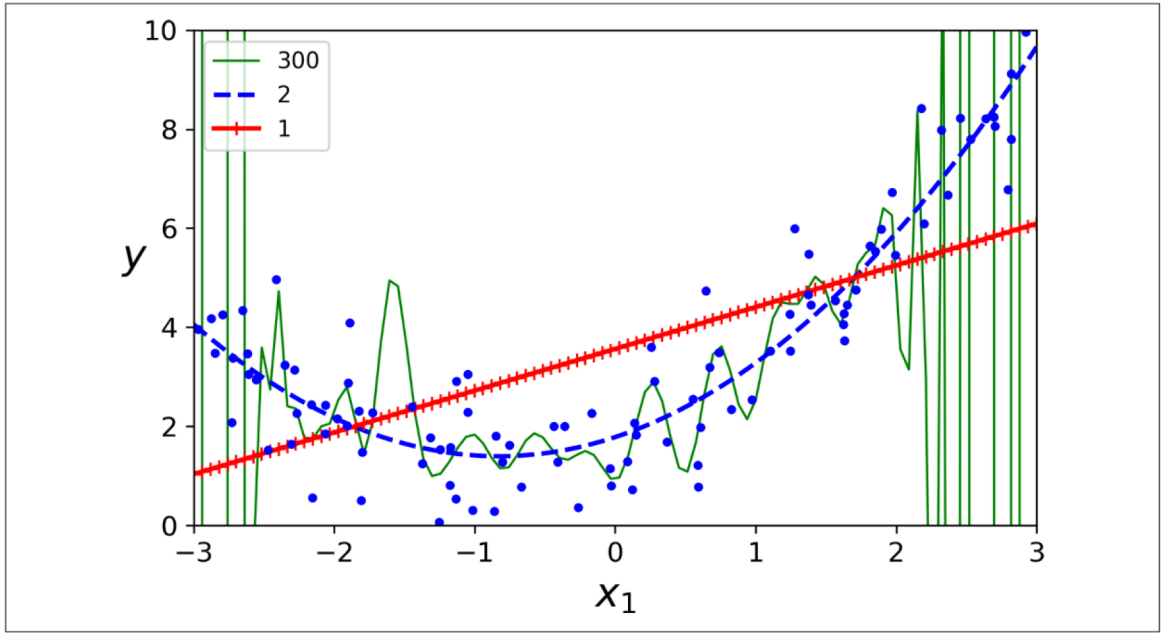


Figure 4-14. High-degree Polynomial Regression

Why Polynomial (Linear) Regression?

Adapted from the Wikipedia:

- Although **polynomial regression** fits a **nonlinear model** to the data, as a statistical estimation problem **it is linear** in the sense of **parameters** of the regression model.
- For this reason, **polynomial regression** is considered to be a *special case* of **multiple linear regression**.

Simple Linear Regression

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1$$

Polynomial Linear Regression

$$\hat{y} = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \cdots + \theta_d x_1^d$$

Pros and Cons of Polynomial Regression

Advantages

- It provides the **best approximation** of the **relationship** between the dependent and independent variables.
- A Broad range of function can be fit under it.
- Polynomial basically fits a wide range of curvature.

Disadvantages

- The presence of **outliers** in the data can **seriously affect** the results of the nonlinear analysis.

Aprendizado de Máquina e Reconhecimento de Padrões 2021.2

Polynomial Regression

Prof. Dr. Samuel Martins (Samuka)
samuel.martins@ifsp.edu.br

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
São Paulo
Campus Campinas