Supervised Learning

Vishal Patel

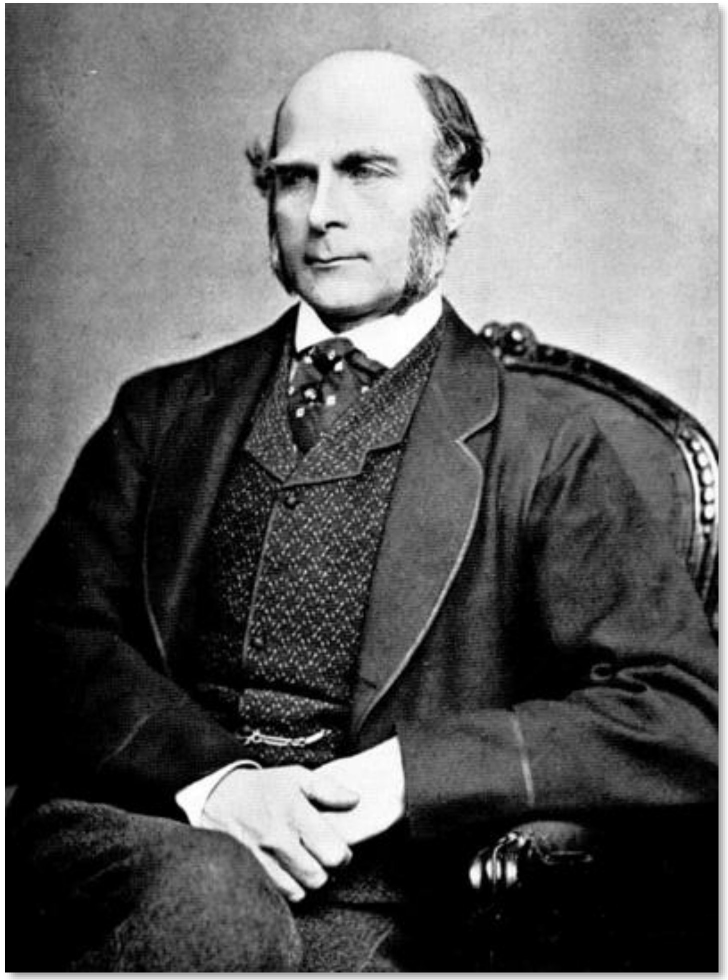
Spring 2024


THE NEW YORK TIMES BESTSELLER
THINKING,
FAST AND SLOW
DANIEL
KAHNEMAN
WINNER OF THE NOBEL PRIZE IN ECONOMICS
669
LN

Course Outline

1. ~~Introduction~~
2. ~~The Data Science Process~~
3. **Supervised Learning**
4. Unsupervised Learning
5. The Grunt Work
6. Wrap Up

Regression to Mediocrity



Sir Francis Galton
1822 – 1911

An English Victorian era statistician,
progressive,
polymath,
sociologist,
psychologist,
anthropologist,
eugenicist,
tropical explorer,
geographer,
inventor,
meteorologist,
proto-geneticist, and
psychometrician.

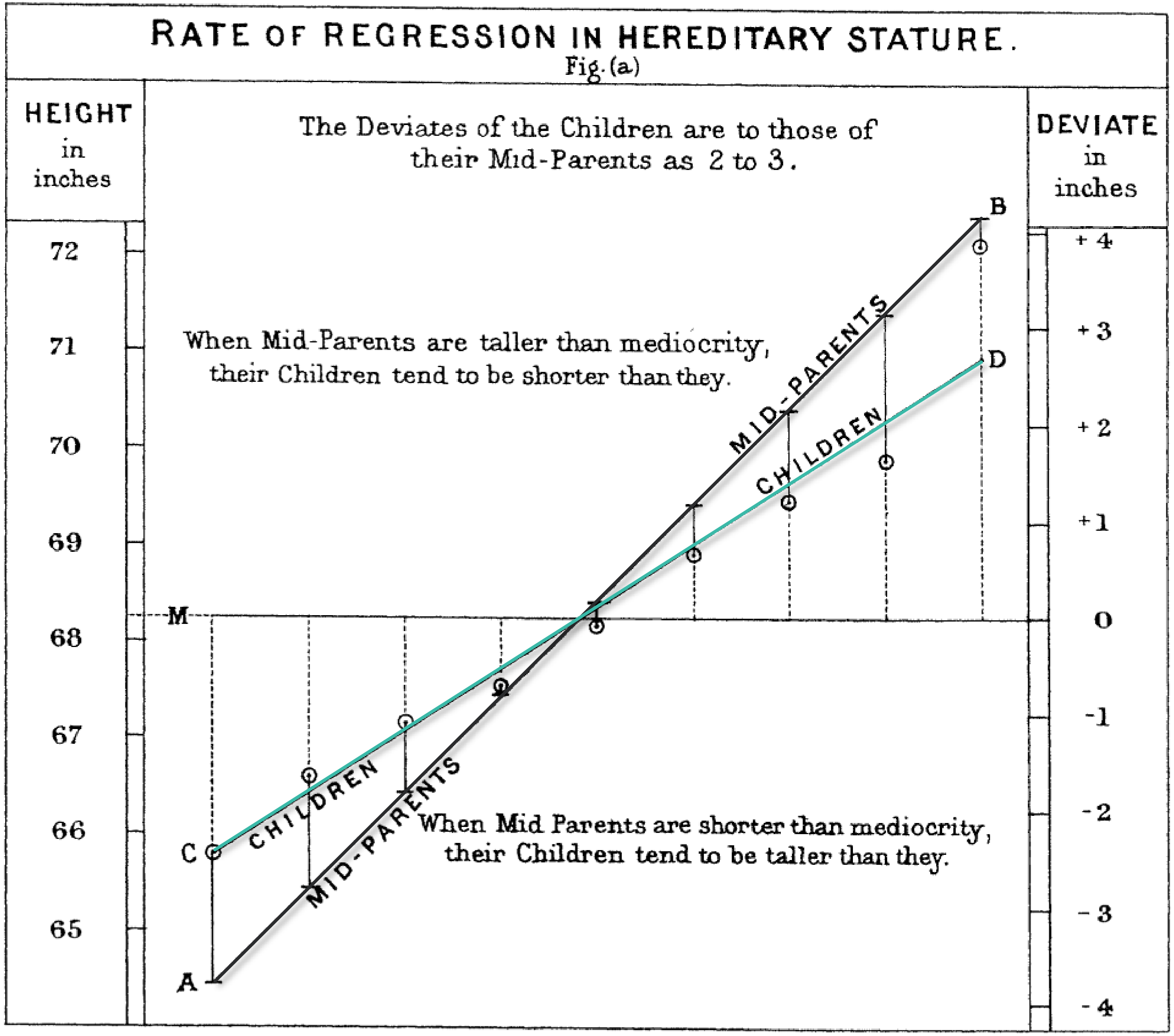
Regression towards Mediocrity

ANTHROPOLOGICAL MISCELLANEA.

REGRESSION *towards* MEDIOCRITY *in* HEREDITARY STATURE.

By FRANCIS GALTON, F.R.S., &c.

1886


RATE OF REGRESSION IN HEREDITARY STATURE.
Fig. (a)
HEIGHT in inches
DEVIATE in inches
The Deviates of the Children are to those of their Mid-Parents as 2 to 3.
When Mid-Parents are taller than mediocrity, their Children tend to be shorter than they.
When Mid Parents are shorter than mediocrity, their Children tend to be taller than they.
MID-PARENTS
CHILDREN
MID-PARENTS
CHILDREN
A
B
C
D
M
72
71
70
69
68
67
66
65
+4
+3
+2
+1
0
-1
-2
-3
-4

Regression to the Mean

- In Kahneman’s example, if a cadet did something exceptional, his/her next attempt is unlikely to be as good (whether he/she was praised or not.)
- Similarly, a baseball player's batting average in the second half of the season can be expected to be closer to the mean (for all players) than his batting average in the first half of the season. And so on.
- The key word here is "expected".

Linear Regression

Supervised Learning

1 Train (a model)

Observations + Labels

Ground Truth

2 Make Predictions

Observations → Predictions

Linear Regression

- **Variable:** A quantity that may vary across observations (either measurements taken across different times or across different subjects, e.g., people).
- When we fit a linear model, we assume (or hope) that one variable (e.g., y) do not vary randomly, but varies as a straight-line function of another variable (e.g., x). In other words, y is dependent on x.
- How do we measure this dependence?
- **Variance:** A measure of the amount of variability in a variable, and it's defined as average squared deviations (fluctuations) from its mean.
  - Alternatively, we can measure variability in terms of standard deviation, which is defined as the square root of variance.
- The goal of a liner model is to find out how much of the variation in y (fluctuations from its mean) can be explained by variation in x (fluctuations from its mean).

A simple linear regression model can be estimated based on only **three** statistics:

1. Mean

2. Variance
   Standard Deviation

3. Correlation Coefficient
   r_{xy}

If we know the correlation coefficient, we know the extent to which fluctuations of one variable (x) from its mean can be used to predict the fluctuations of other variable (y) from its mean.

Correlation Coefficient

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

It measures the strength of the linear relationship between y and x on a scale of -1 to $+1$.

In order to calculate the correlation coefficient,

we should first standardize the variables

by taking out the mean and dividing by standard deviation.

How much does each point
fluctuate from its mean

Z Score $$x_i^* = \frac{x_i - AVERAGE(x)}{STDEV(x)}$$

... compared to
how much this variable fluctuates overall.

$$x_i^* = \frac{x_i - AVERAGE(x)}{STDEV(x)}$$

$$x_i^* = \frac{(x_i - \bar{x})}{\frac{1}{n}\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}}$$

$$y_i^* = \frac{y_i - AVERAGE(y)}{STDEV(y)}$$

$$y_i^* = \frac{(y_i - \bar{y})}{\frac{1}{n}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

$$r_{xy} = \frac{1}{n}\sum_{i=1}^{n} x_i^* y_i^*$$

$$r_{xy} = \frac{1}{n}\sum_{i=1}^{n} x_i^* y_i^*$$

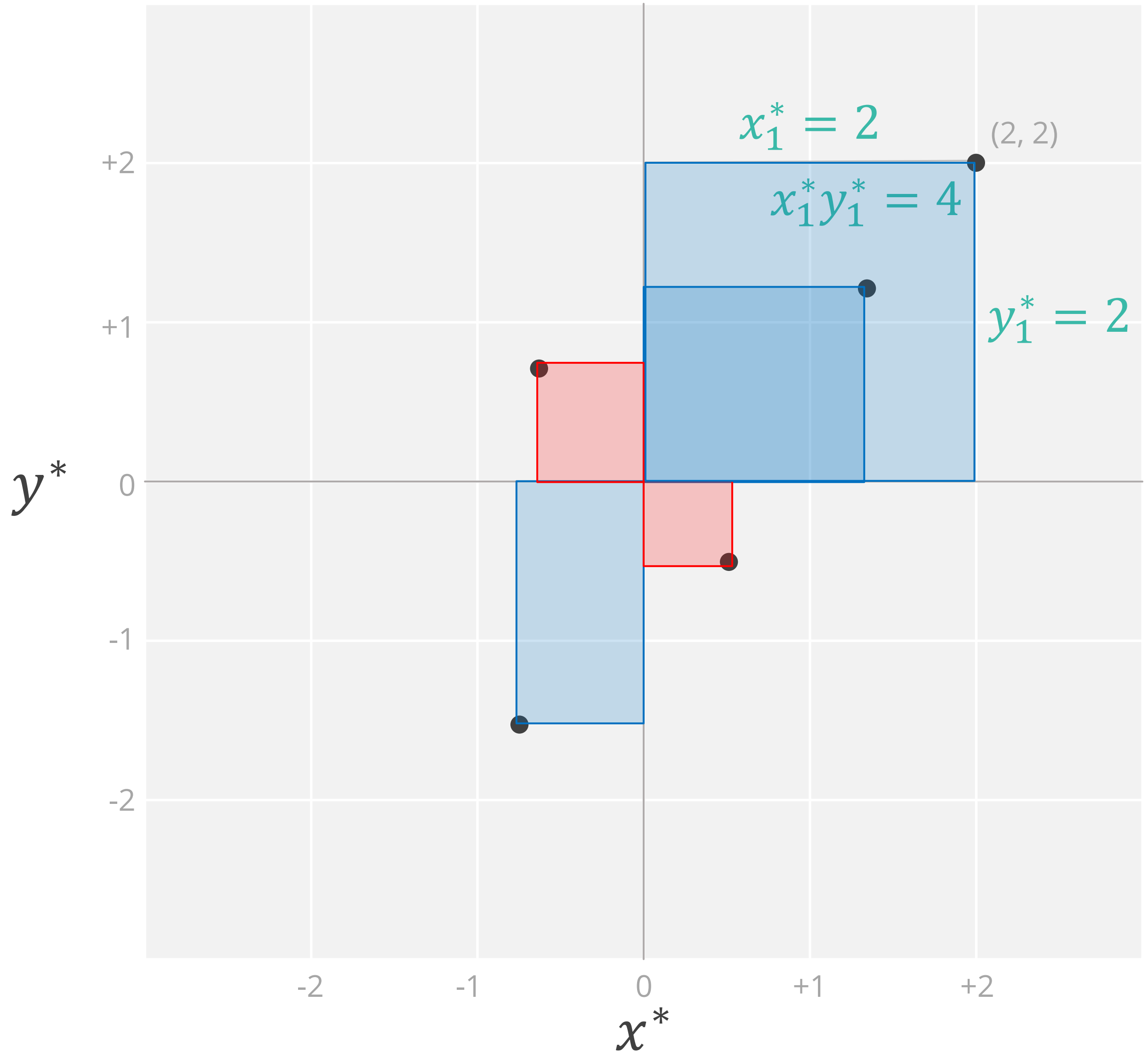
$$r_{xy} = \frac{(x_1^* y_1^* + x_2^* y_2^* + \cdots + x_n^* y_n^*)}{n}$$

The correlation coefficient is equal to

the average product of the standardized values of the two variables.

$$x_i^* = \frac{(x_i - \bar{x})}{\frac{1}{n}\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}} \qquad y_i^* = \frac{(y_i - \bar{y})}{\frac{1}{n}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

$$r_{xy} = \frac{(x_1^* y_1^* + x_2^* y_2^* + x_3^* y_3^* + x_5^* y_5^* + x_5^* y_5^*)}{5}$$



The sum of those five rectangles would add up to a positive number, hence we would get a positive correlation coefficient between x and y.

The correlation coefficient

measures the strength of the linear relationship

between y and x.

The linear equation for predicting y^* from x^*

that minimizes mean squared error is simply:

$$\boxed{\hat{y}_i^* = r_{xy}\, x_i^*}$$

Thus, if x is observed to be 1 standard deviation above its own mean,

then we should predict that y will be r_{xy} standard deviations above its own mean.

$$\hat{y}_i^* = r_{xy}\, x_i^*$$

1 Means 2 Standard deviations 3 Correlation coefficient

$$\frac{(\hat{\boldsymbol{y}}_i - \bar{\boldsymbol{y}})}{\boldsymbol{\sigma}_{\boldsymbol{y}}} = r_{xy} \frac{(\boldsymbol{x}_i - \bar{\boldsymbol{x}})}{\boldsymbol{\sigma}_{\boldsymbol{x}}}$$

Where:

$$\beta_1 = r_{xy} \frac{\sigma_y}{\sigma_x}$$

$$\beta_0 = \bar{y} - \beta_1 \bar{x}$$

$$(\hat{y}_i - \bar{y}) = r_{xy} \frac{\sigma_y}{\sigma_x} (x_i - \bar{x})$$

$$(\hat{y}_i - \bar{y}) = r_{xy} \frac{\sigma_y}{\sigma_x} (\boldsymbol{x}_i) - r_{xy} \frac{\sigma_y}{\sigma_x} (\bar{\boldsymbol{x}})$$

$$\hat{y}_i = \beta_0 + \beta_1 x_i$$

$$\hat{y}_i = r_{xy} \frac{\sigma_y}{\sigma_x} (x_i) - r_{xy} \frac{\sigma_y}{\sigma_x} (\bar{x}) + \bar{\boldsymbol{y}}$$

$$\hat{y}_i = \bar{y} - \boxed{r_{xy} \frac{\sigma_y}{\sigma_x}} (\bar{x}) + \boxed{r_{xy} \frac{\sigma_y}{\sigma_x}} (x_i)$$

$$\hat{y}_i = \boxed{\bar{y} - \boldsymbol{\beta}_1 \bar{x}} + \boldsymbol{\beta}_1 x_i$$

R Squared

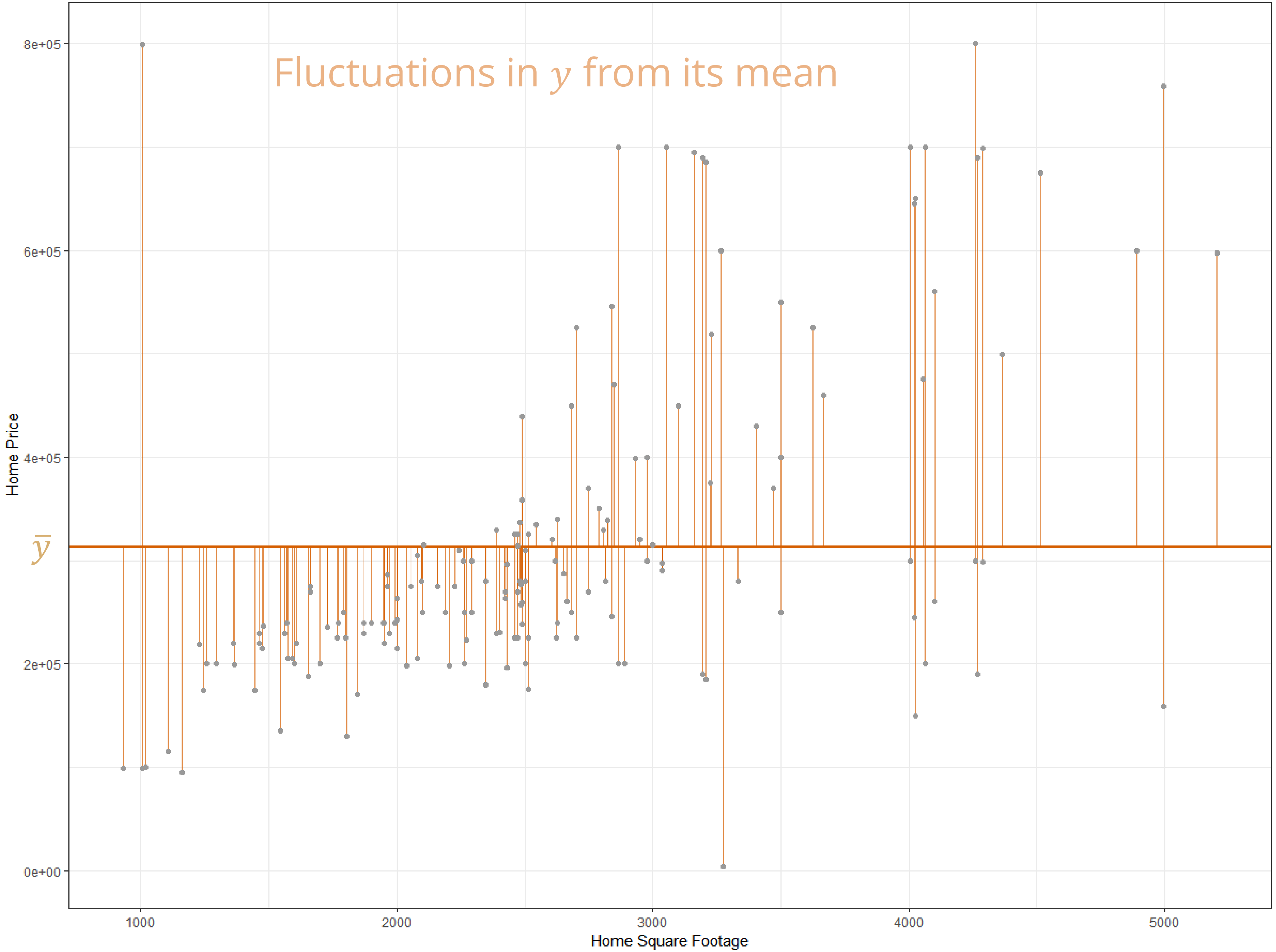
Residual (Error) Sum of Squares **SSE**

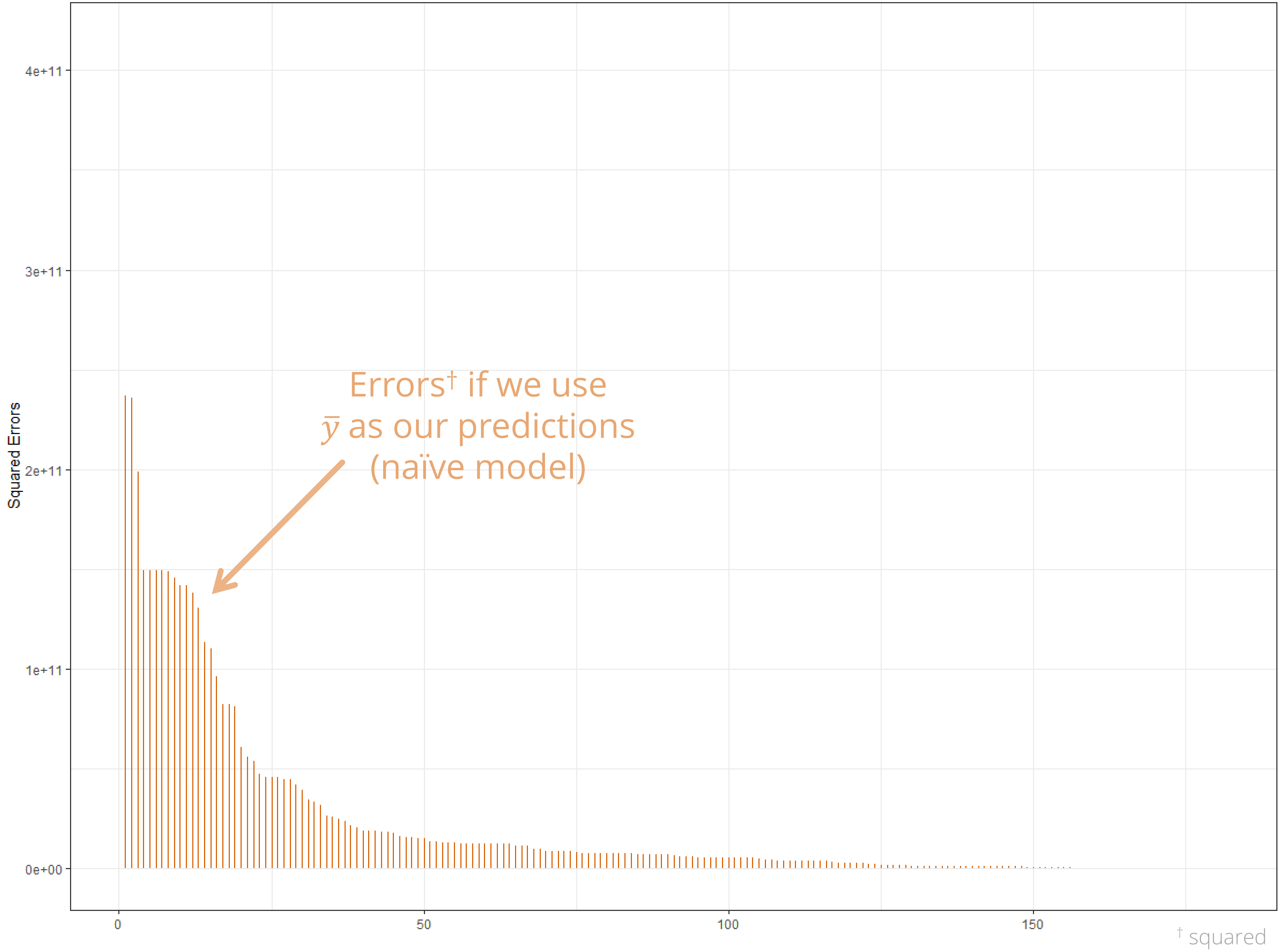
Difference between actual and predicted values of y

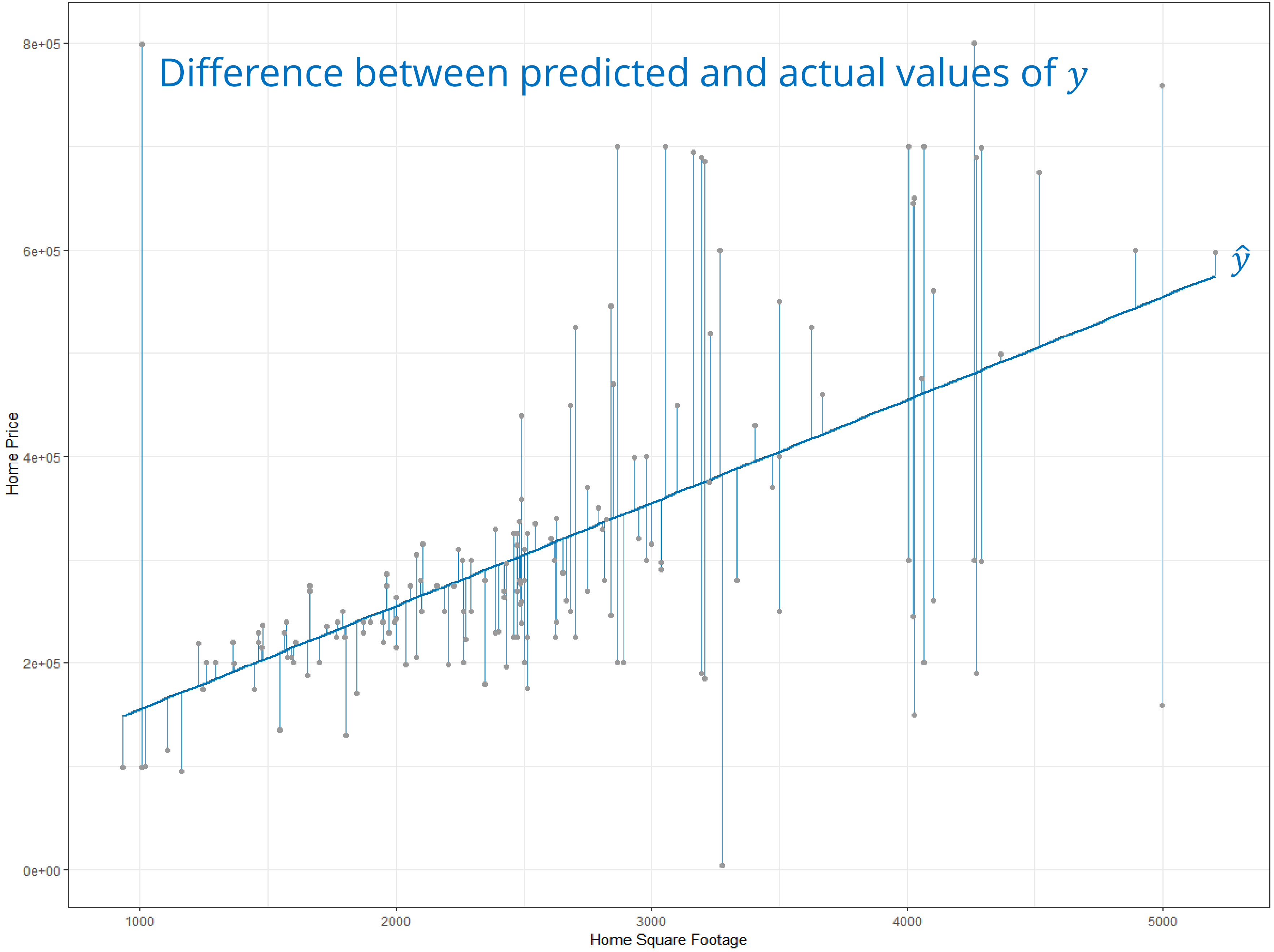
$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Fluctuations in y from its mean

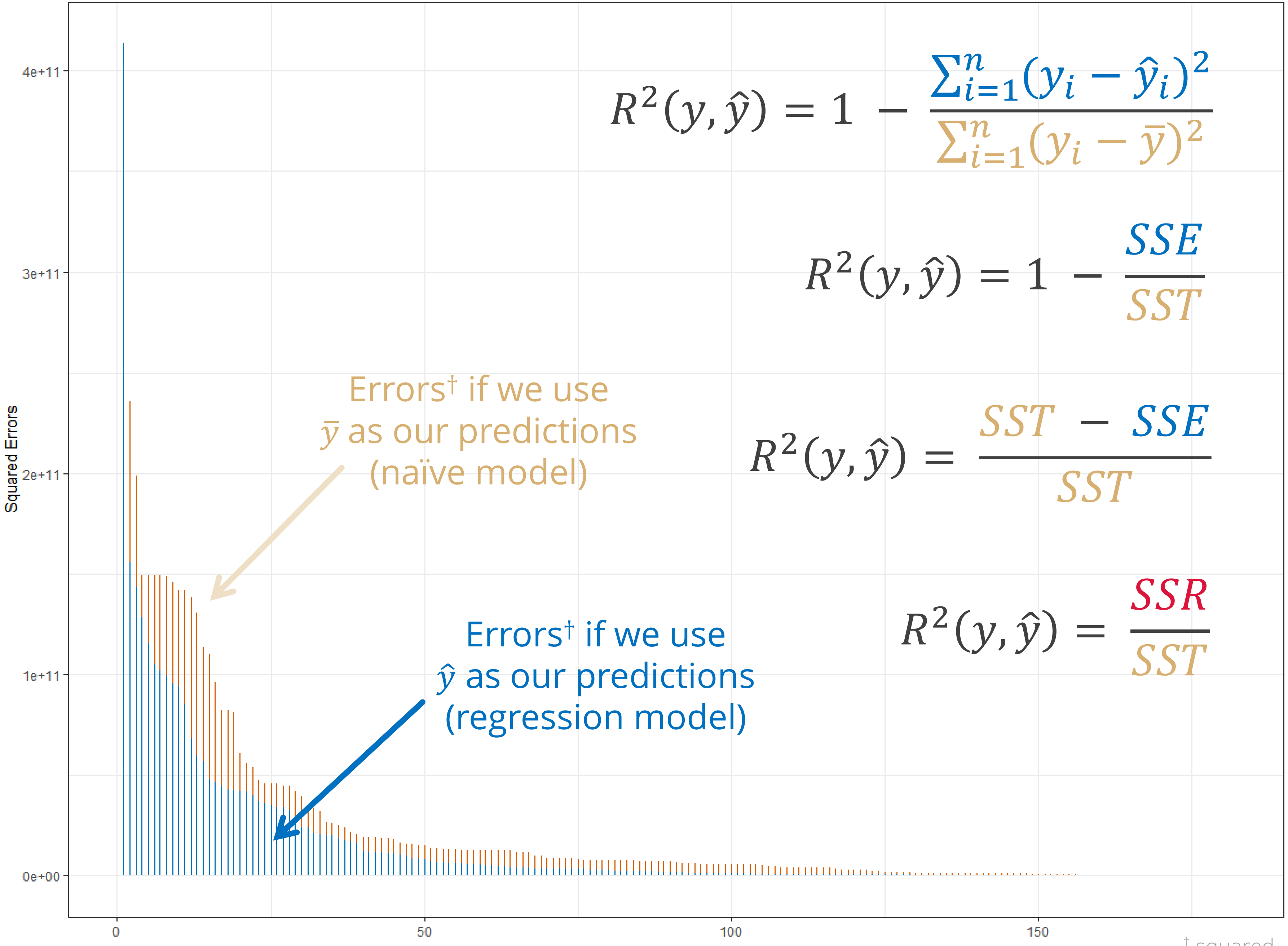
Total Sum of Squares **SST**


Fluctuations in y from its mean
Home Price
Home Square Footage
$\bar{y}$
8e+05
6e+05
4e+05
2e+05
0e+00
1000
2000
3000
4000
5000


Errors† if we use
$\bar{y}$ as our predictions
(naïve model)
Squared Errors
4e+11
3e+11
2e+11
1e+11
0e+00
0
50
100
150
† squared



Difference between predicted and actual values of y
$\hat{y}$
8e+05
6e+05
4e+05
2e+05
0e+00
Home Price
1000
2000
3000
4000
5000
Home Square Footage



$$R^2(y,\hat{y}) = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

$$R^2(y,\hat{y}) = 1 - \frac{SSE}{SST}$$

$$R^2(y,\hat{y}) = \frac{SST - SSE}{SST}$$

$$R^2(y,\hat{y}) = \frac{SSR}{SST}$$

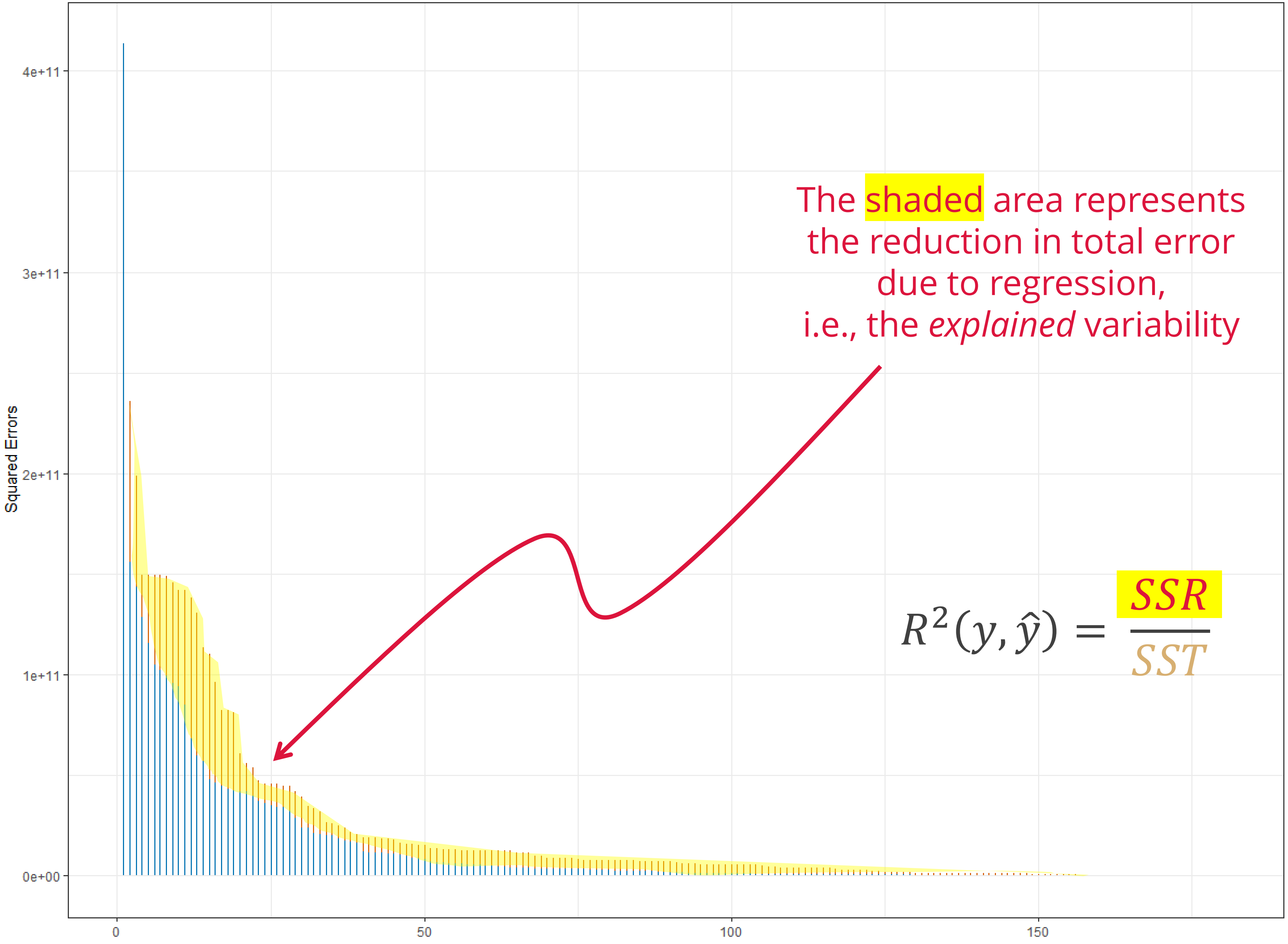
† squared

Variability

SST = Total

SSR = Explained

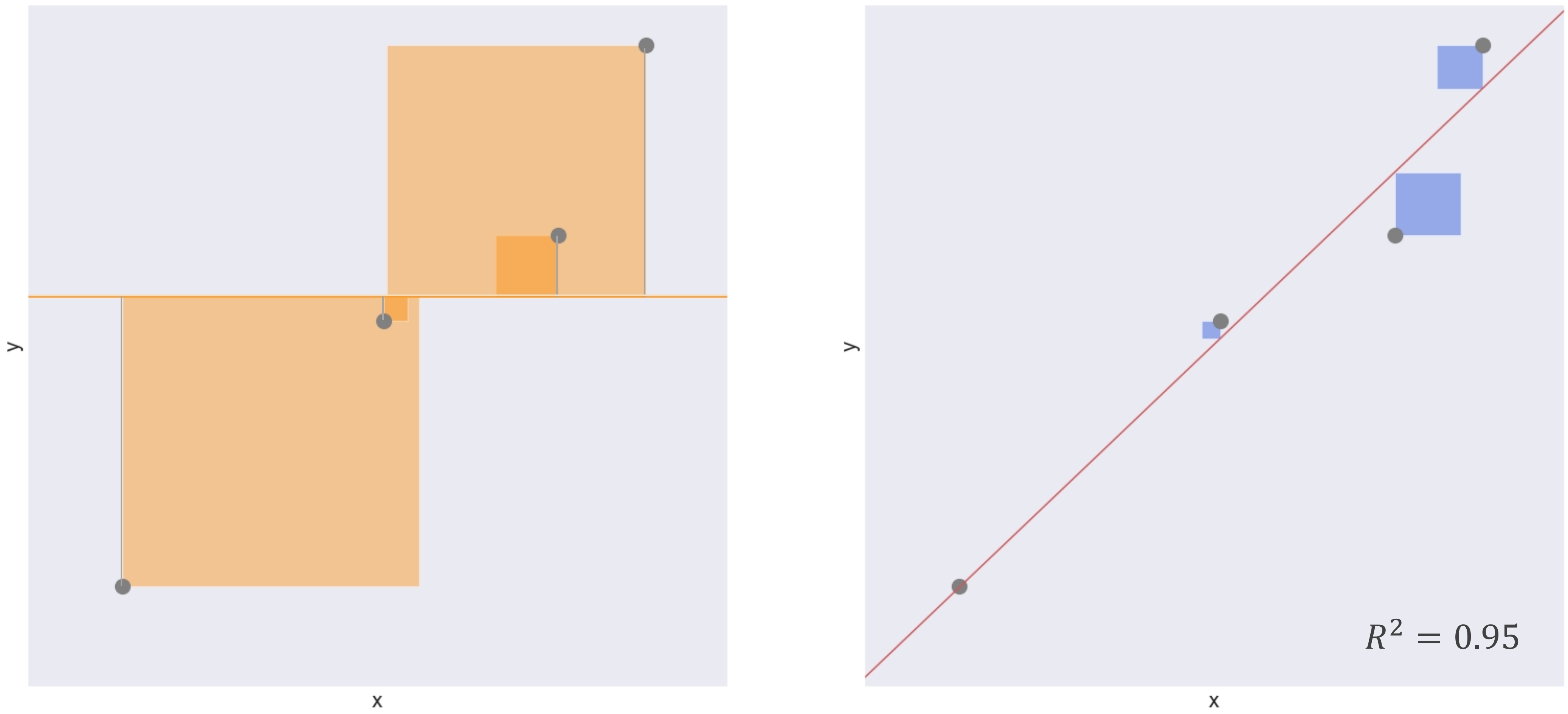
SSE = Unexplained



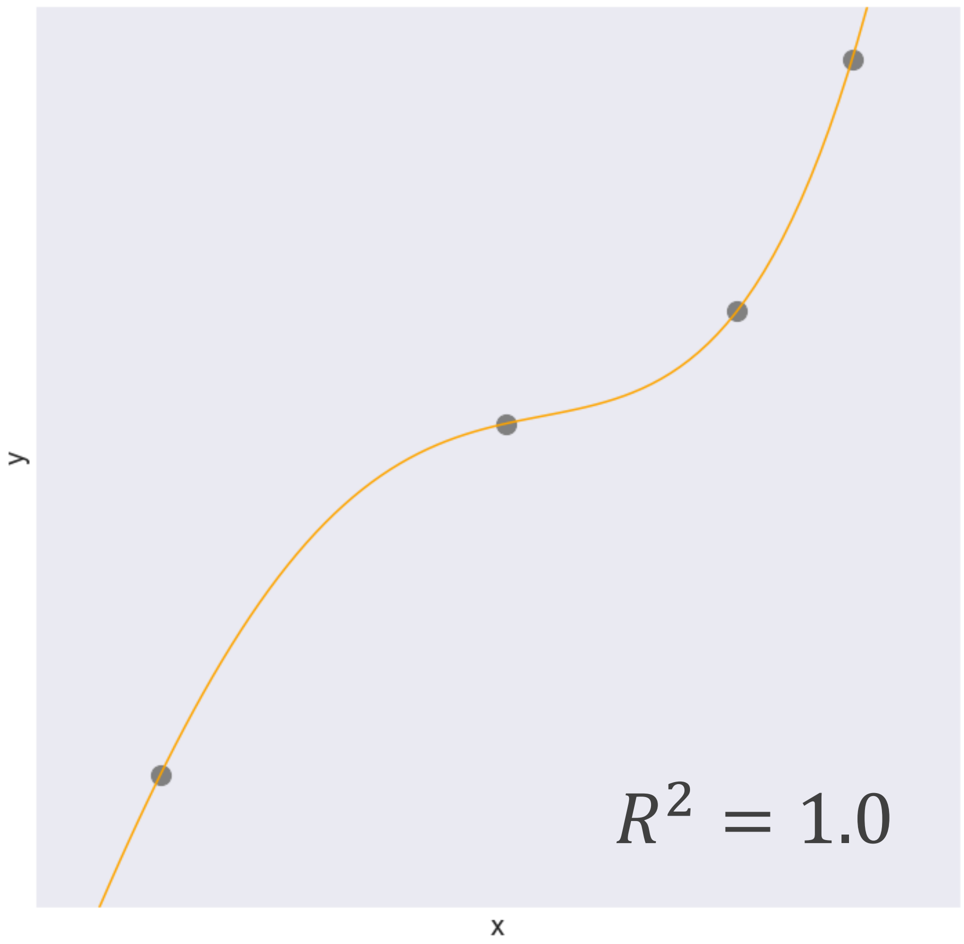
The shaded area represents
the reduction in total error
due to regression,
i.e., the *explained* variability

$$R^2(y, \hat{y}) = \frac{SSR}{SST}$$

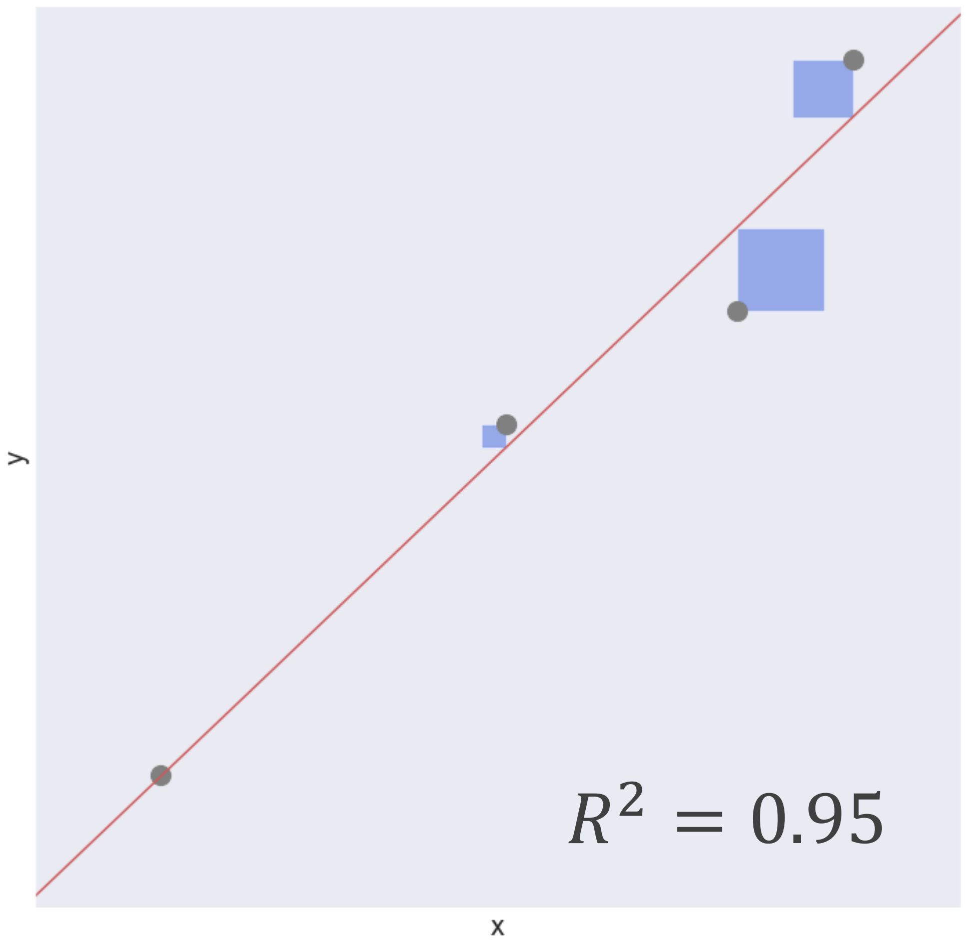
SST $\quad$ SSE



$$R^2 = 1 - \frac{SSE}{SST}$$



Polynomial Regression (n=4)



Linear Regression


THE BEST WAY TO
EXPLAIN OVERFITTING
imgflip.com
ADDTEXT.COM

t Statistic

t-statistic

$$t = \frac{b_1 - \beta_1^{(0)}}{SE_{b_1}}$$

Student's t-distribution

Variety

vs.

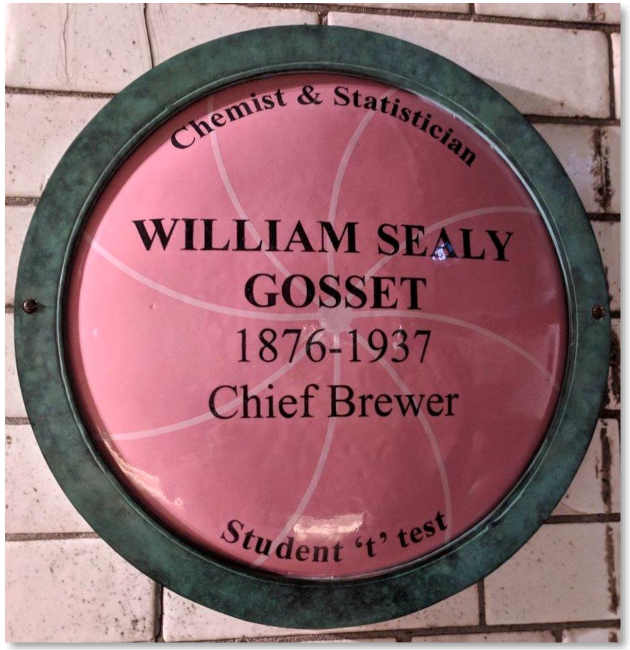
Consistency

↓

Statistical Quality Control

William Sealy Gosset

A 19th century English statistician



The Probable Error of a Mean (1908), published by an anonymous "Student"

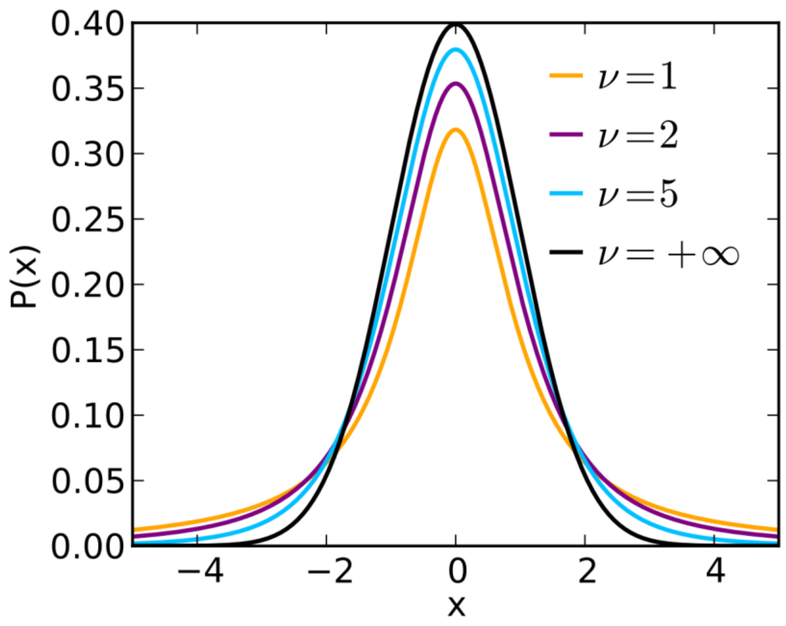
t-statistic

$$t = \frac{\hat{\beta} - 0}{SE_{\hat{\beta}}}$$

It measures

how many standard deviations away from zero

the estimated coefficient is.



The p-value is the probability

of observing a t-statistic that large or larger in magnitude

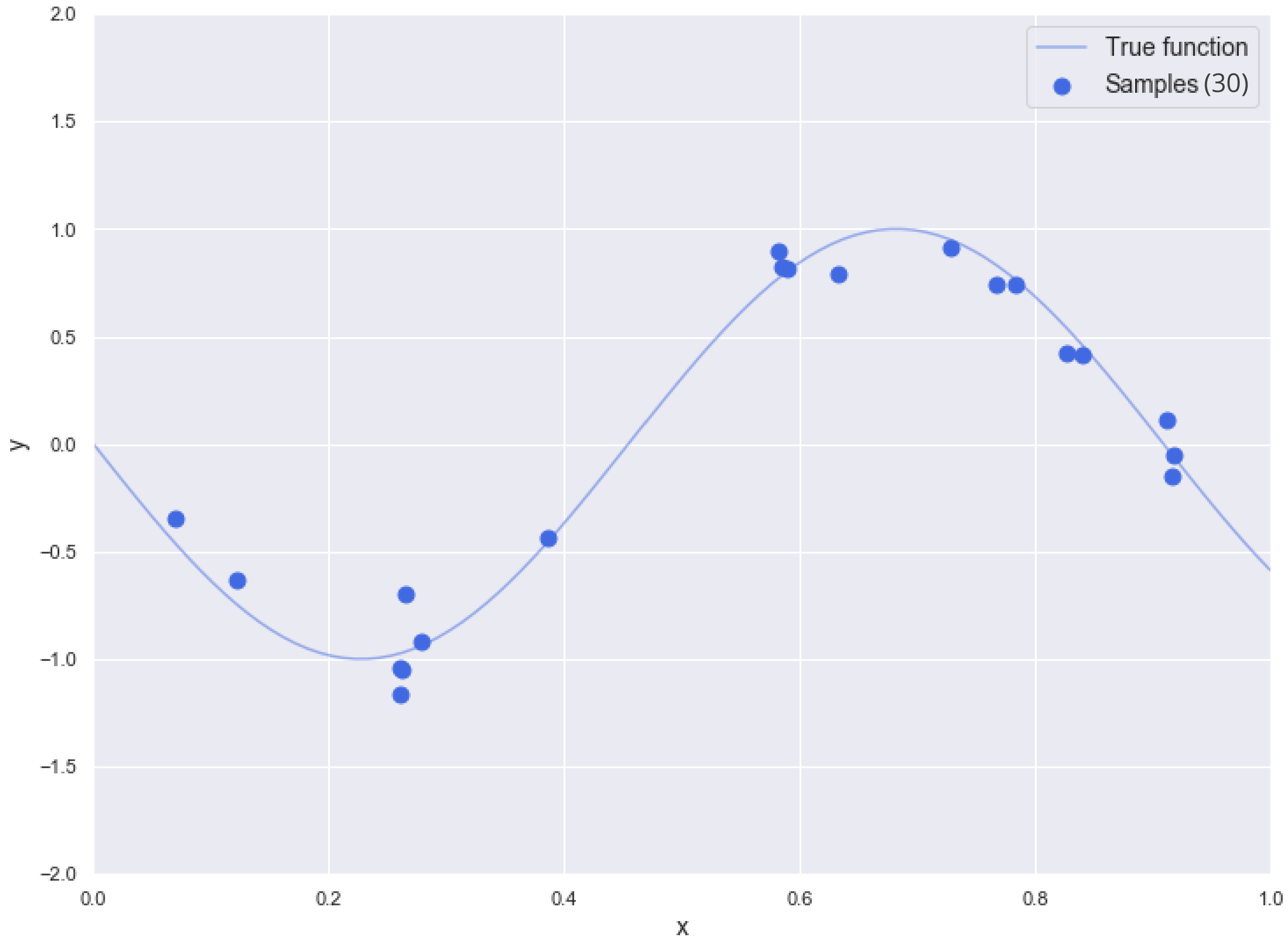
given the null hypothesis that the true coefficient value is zero.

***p*-value > 0.05** → the variable is "accidentally" significant

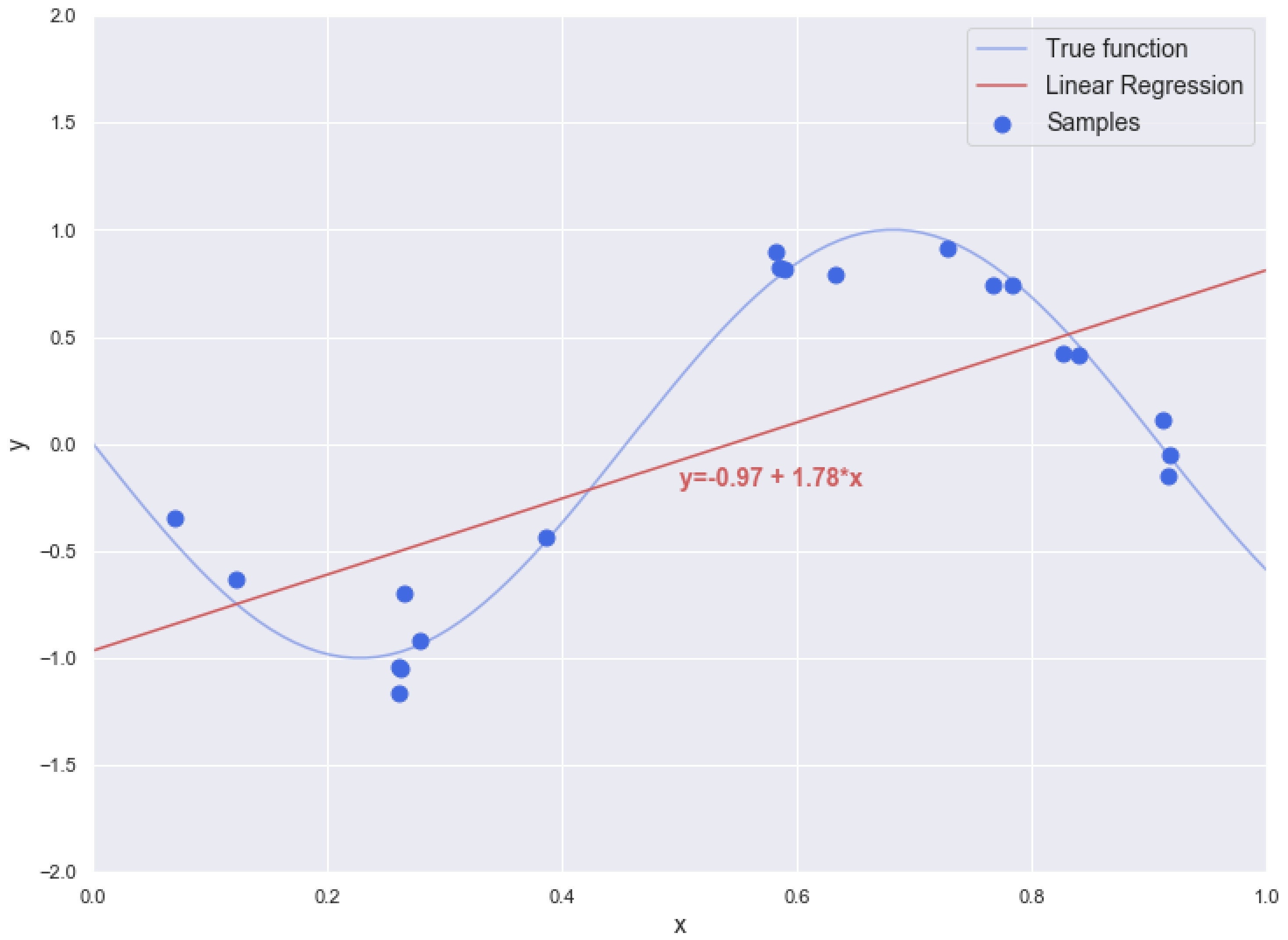
Useful for Feature Selection.


SUI 89:26
p=0.0501
SUI 90:58
p=0.0499

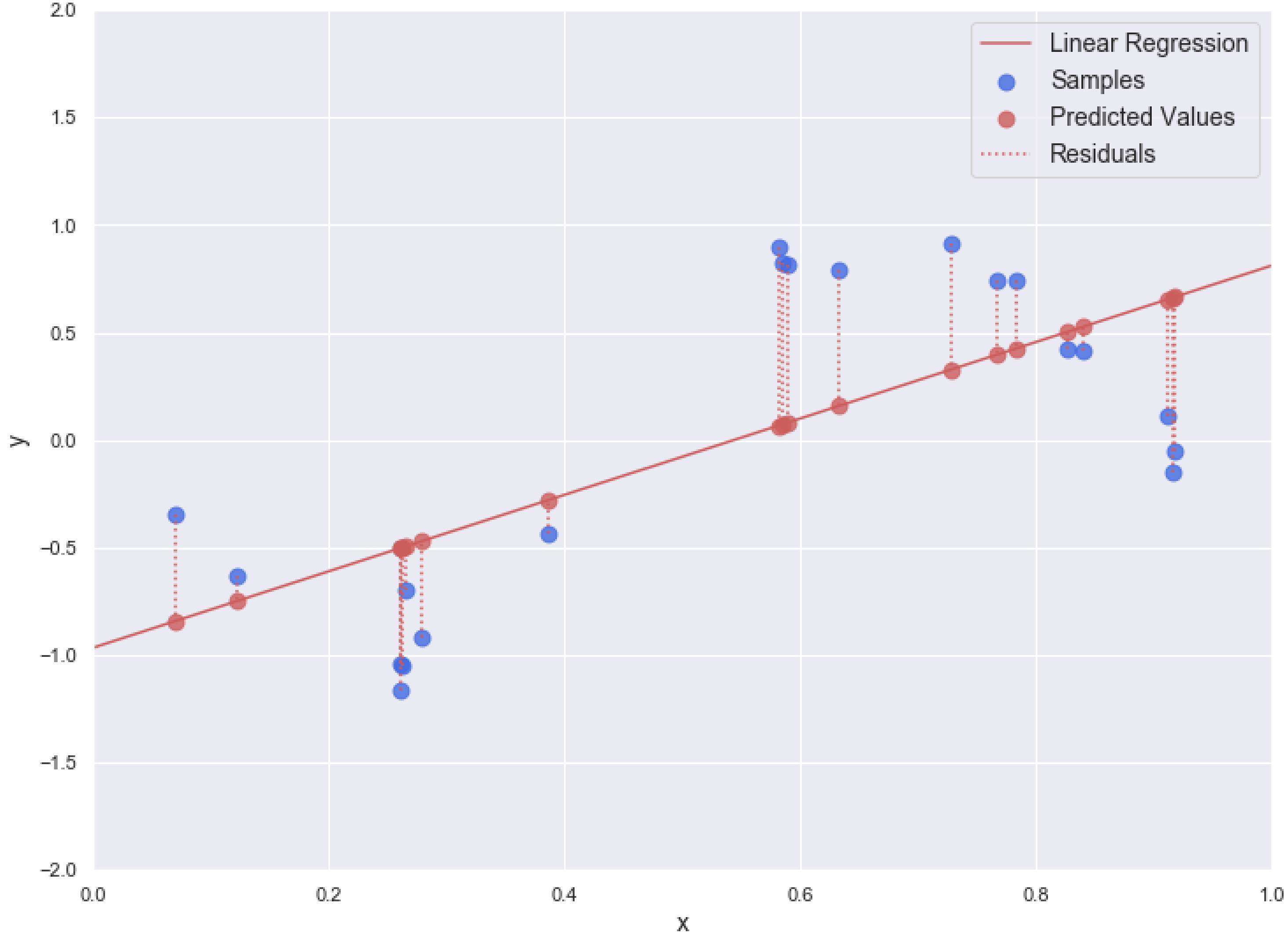
Simple Linear Regression (Example)

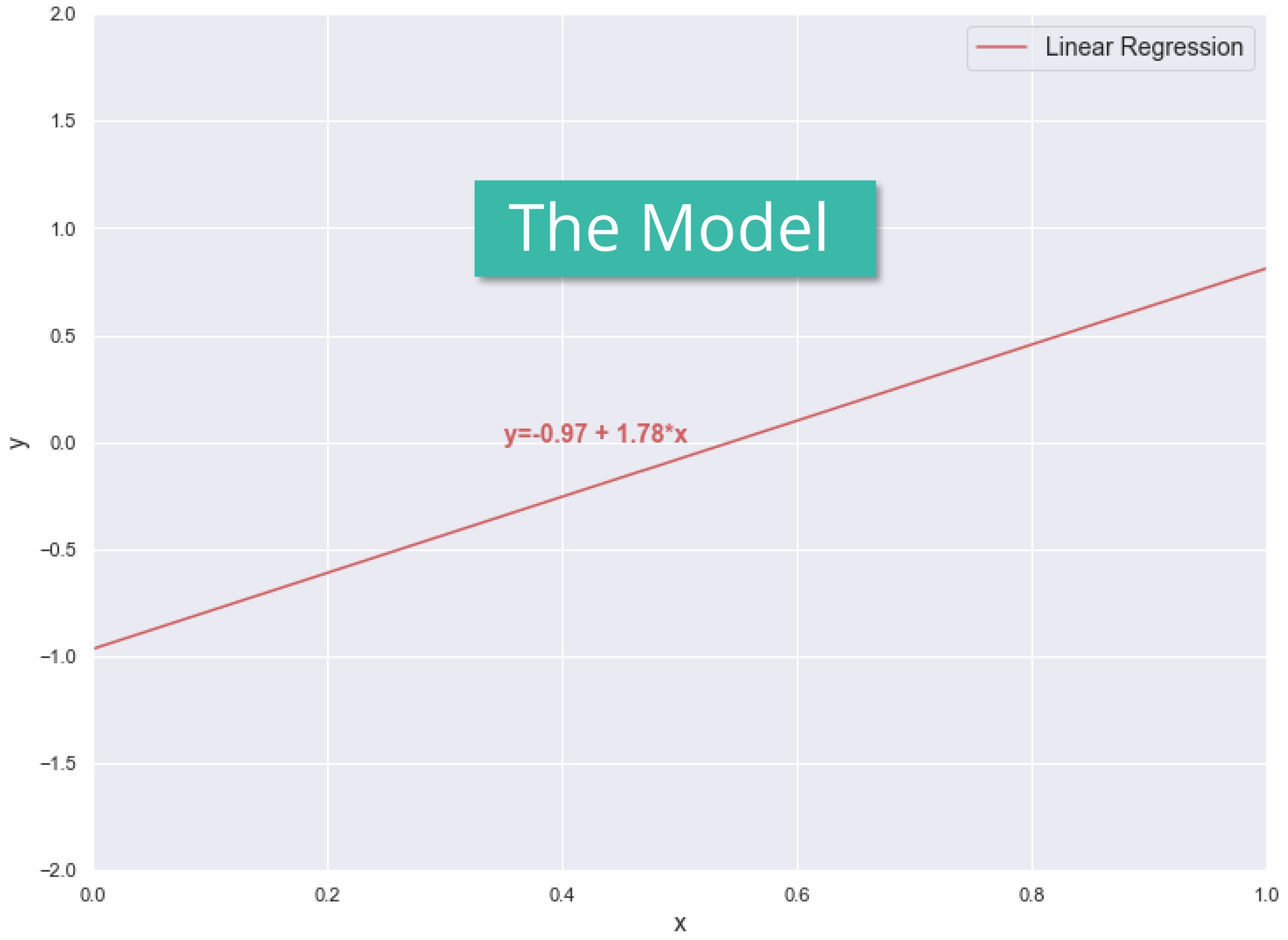


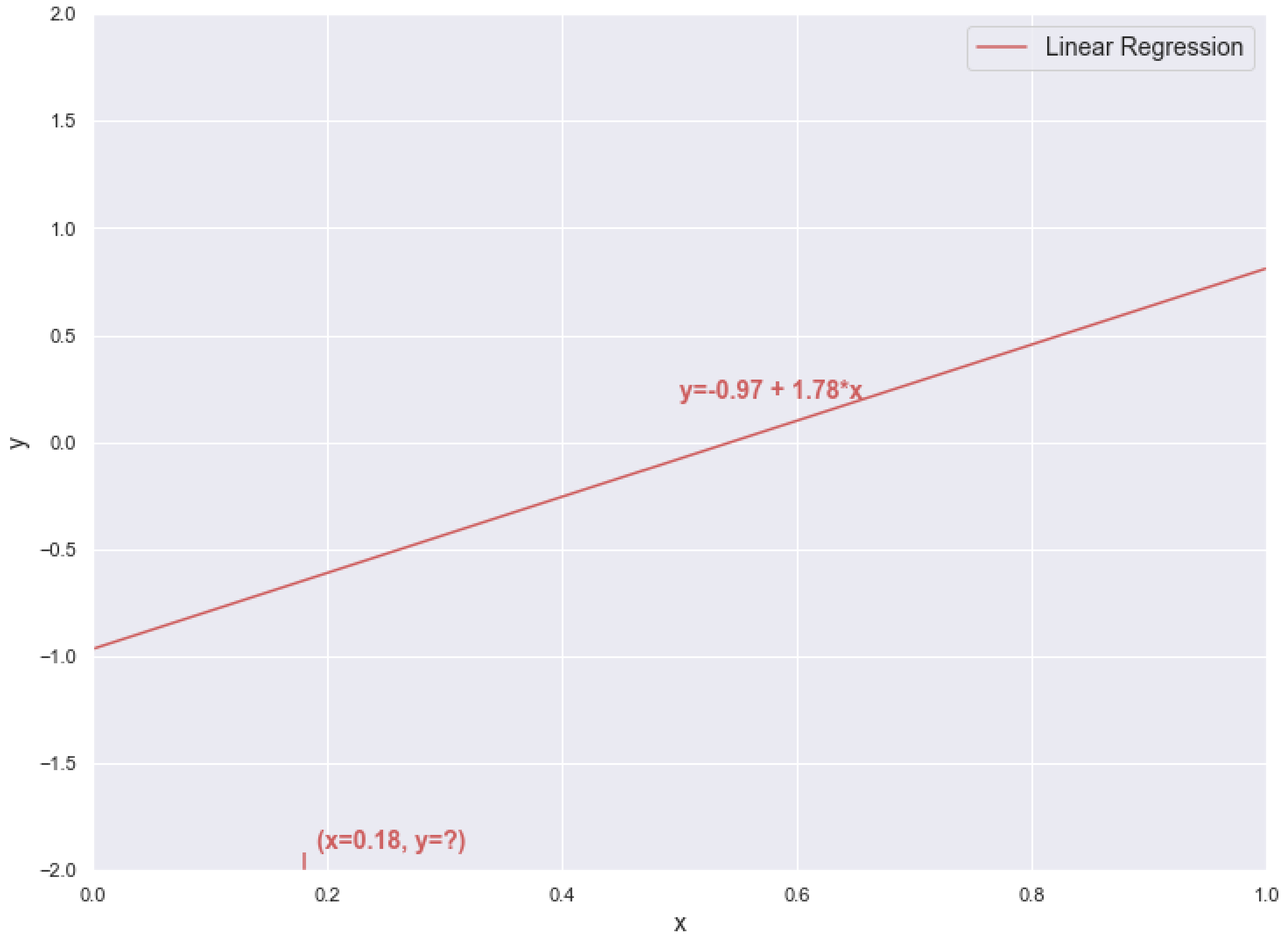
$$y = -\sin(2.2 * \pi * X) + \varepsilon$$



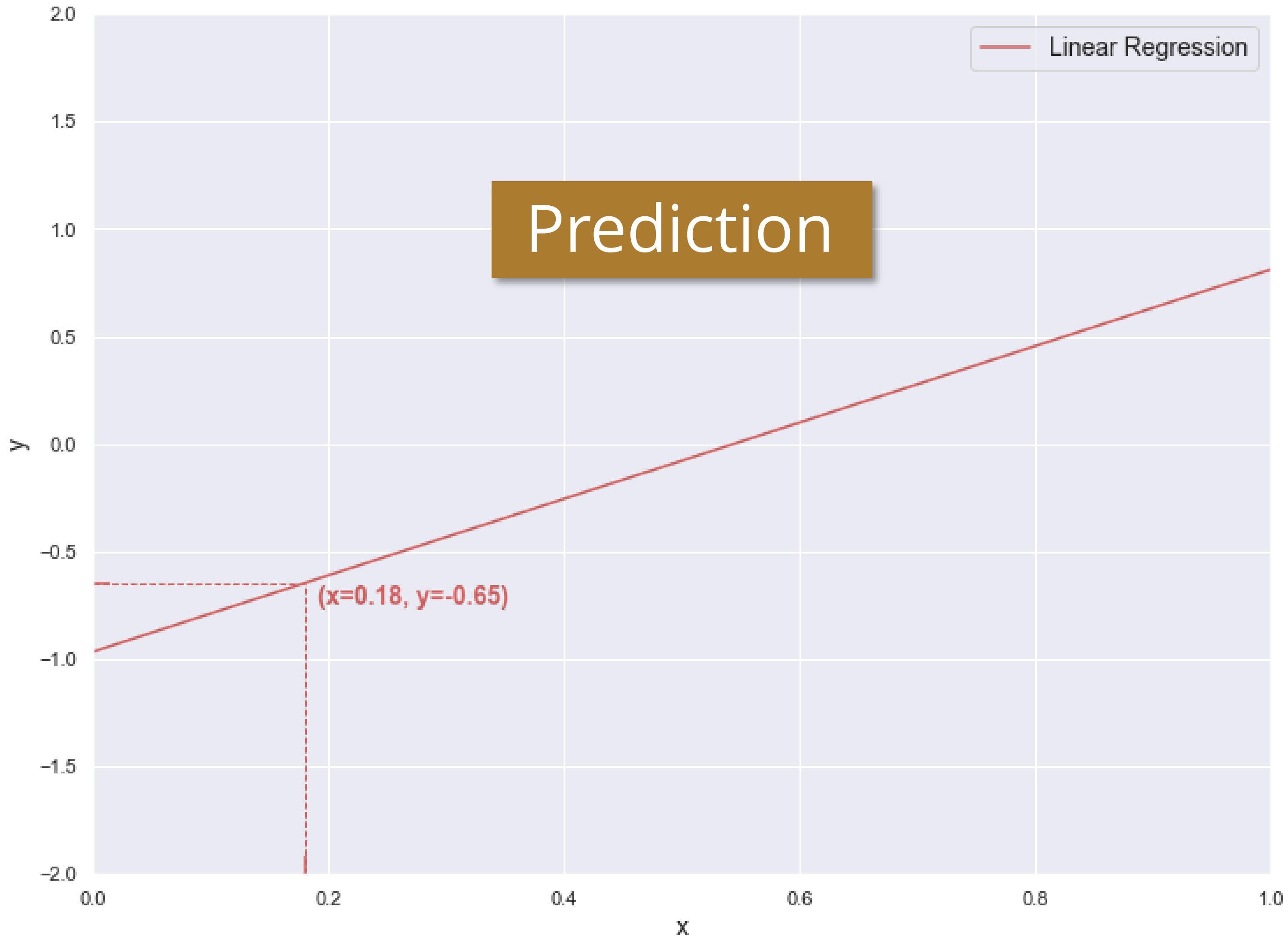
True function
Linear Regression
Samples
y=-0.97 + 1.78*x
x
y

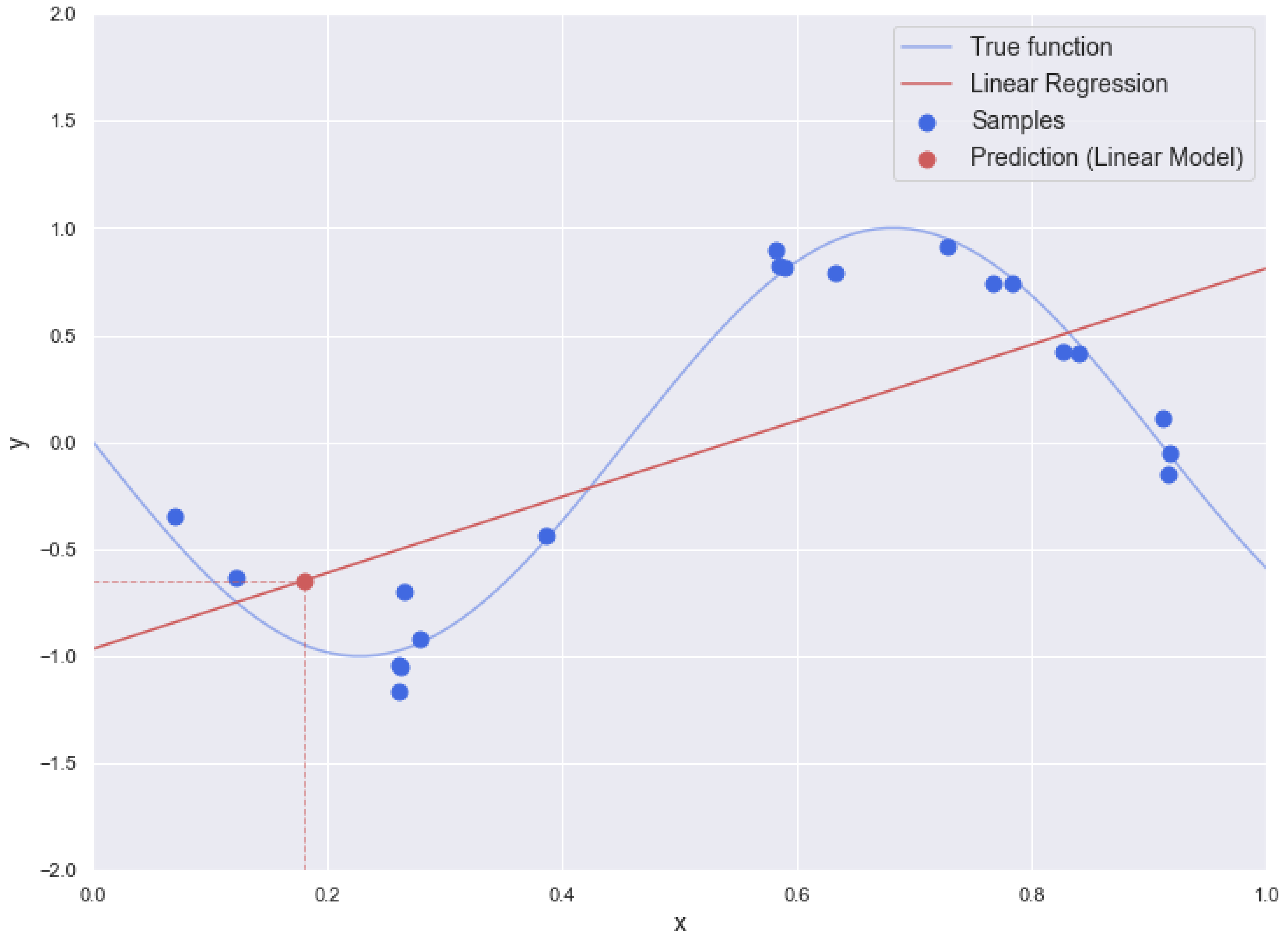

Linear Regression
Samples
Predicted Values
Residuals
x
y
0.0
0.2
0.4
0.6
0.8
1.0
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0

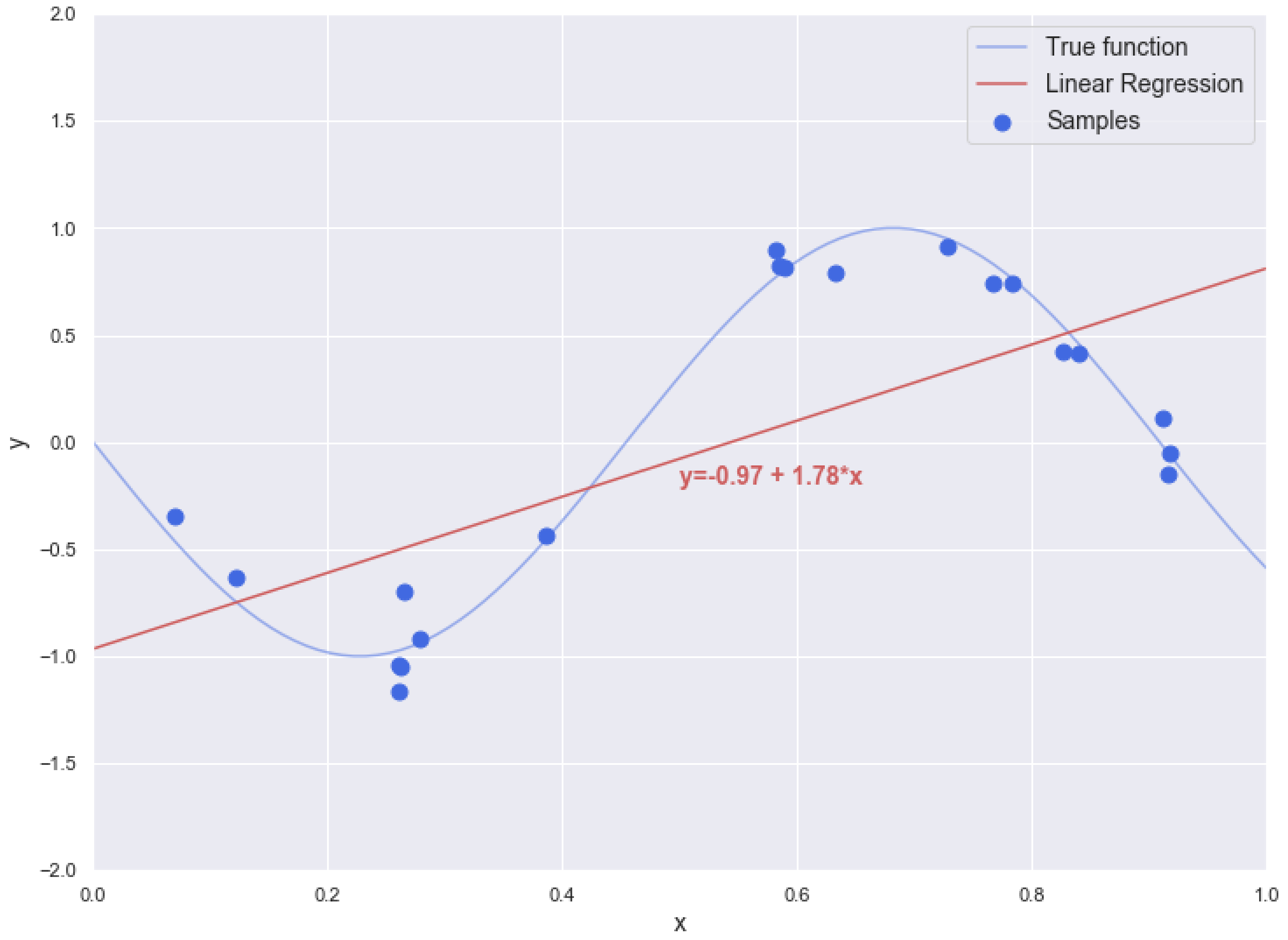

The Model
Linear Regression
y=-0.97 + 1.78*x
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
y
0.0
0.2
0.4
0.6
0.8
1.0
x



Linear Regression
y=-0.97 + 1.78*x
(x=0.18, y=?)
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
y
0.0
0.2
0.4
0.6
0.8
1.0
x


Linear Regression
Prediction
(x=0.18, y=-0.65)
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
y
0.0
0.2
0.4
0.6
0.8
1.0
x


True function
Linear Regression
Samples
Prediction (Linear Model)
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
y
0.0
0.2
0.4
0.6
0.8
1.0
x



MSE = **0.29**

R-Squared = **0.31**

Linear Regression

DATA SET	$\{y_i, x_{i1}, \dots, x_{ij}\}_{i=1}^{n}$
EQUATION	$y = \underbrace{X^T\beta} + \varepsilon$ The model is linear in its parameters.
ASSUMPTION	$\varepsilon \sim N(0, \sigma^2)$ The error is a Gaussian random variable with expectation zero and variance σ^2.

$$y = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \cdot \\ \cdot \\ \cdot \\ y_n \end{pmatrix} \qquad X = \begin{pmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1j} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2j} \\ x_{31} & x_{32} & x_{33} & \dots & x_{3j} \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ x_{n1} & x_{n2} & x_{n3} & \dots & x_{nj} \end{pmatrix}$$

Supervised Learning

Linear Regression in scikit-learn

Install User Guide API Examples More

Go

scikit-learn

Machine Learning in Python

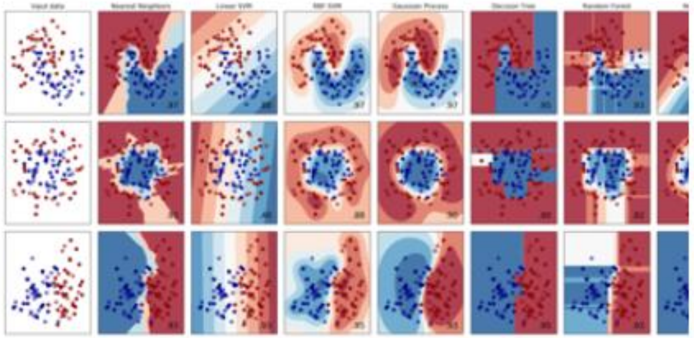
Getting Started | What's New in 0.22.1 | GitHub

- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.
Algorithms: SVM, nearest neighbors, random forest, and more...

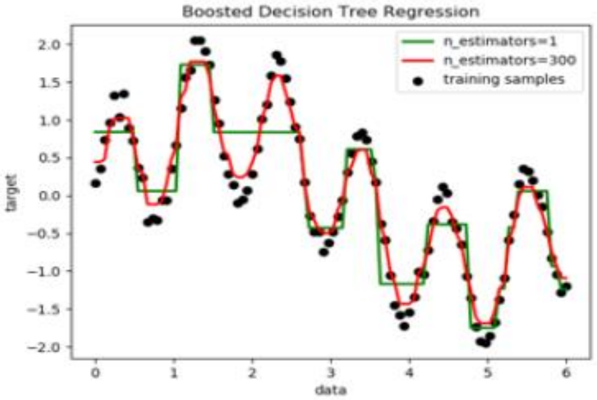


Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: SVR, nearest neighbors, random forest, and more...



Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, and more...

Examples

```
class sklearn.linear_model.LinearRegression(
    fit_intercept=True,
    normalize=False,
    copy_X=True,
    n_jobs=None)
```

Package Module Class

1
```
# Import
from sklearn.linear_model import LinearRegression
```

2
```
# Define
linreg = LinearRegression()
```

3
```
# Fit
linreg.fit(X_train, y)
```

4
```
# Predict
linreg.predict(X_test)
```

Functions

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

```
class sklearn.linear_model.LinearRegression(
    fit_intercept=True,
    normalize=False,
    copy_X=True,
    n_jobs=None)
```

Whether to calculate the intercept for this model.

If set to `False`,

no intercept will be used in calculations

(e.g. data is expected to be already centered)

Recommendation: `fit_intercept = True` (default)

```
class sklearn.linear_model.LinearRegression(
    fit_intercept=True,
    normalize=False,
    copy_X=True,
    n_jobs=None)
```

If `True`, the regressors X will be normalized before regression by subtracting the mean and dividing by the l2-norm.

This parameter is ignored when `fit_intercept` is set to `False`.

This parameter has been deprecated.

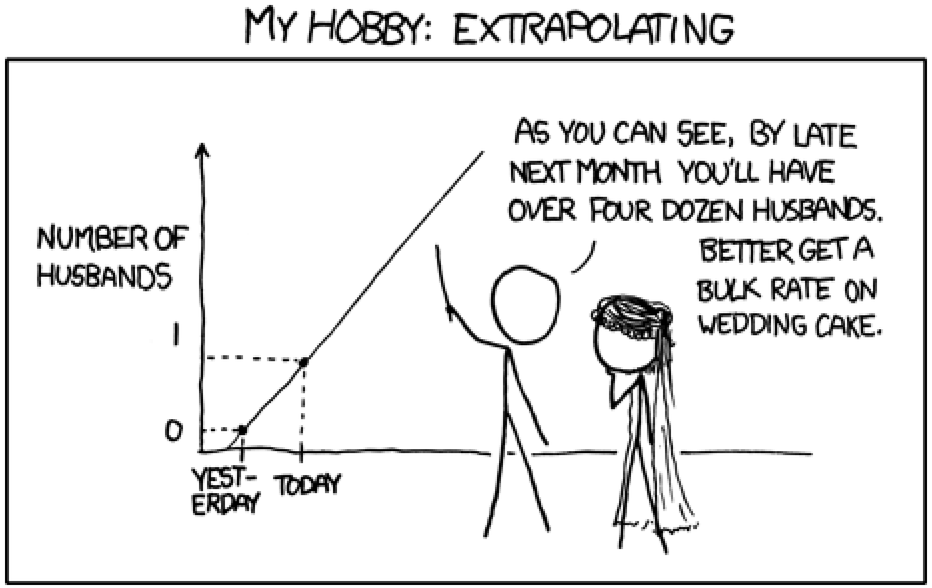
Recommendation: `normalize = False` (default)

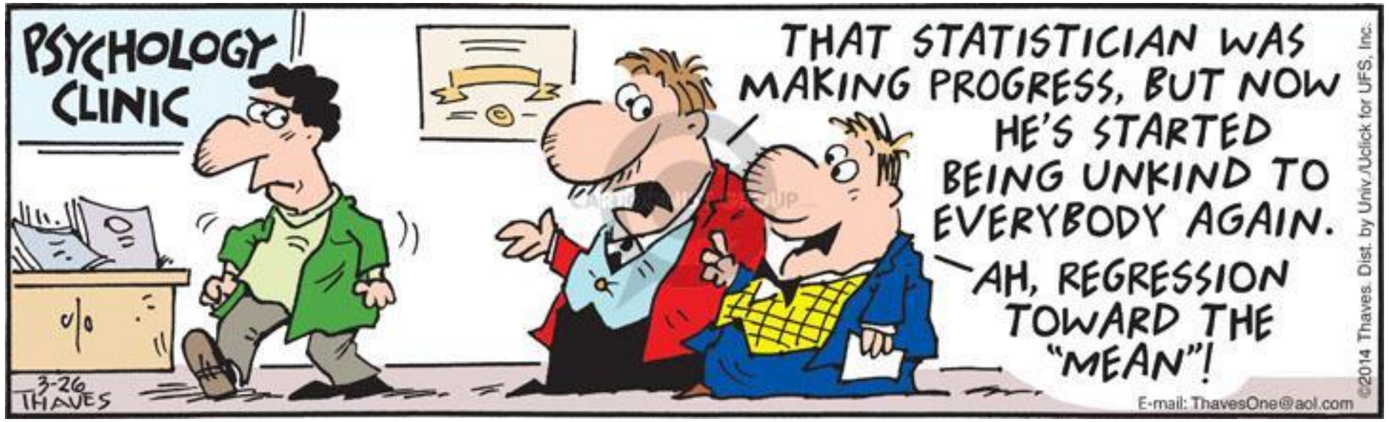
Normalize the data prior to training a model.

Linear Regression Tutorial

07_linear_reg_intro.ipynb

If she loves you more each and every day,
by linear regression she hated you before you met.





Curve Fitting: https://xkcd.com/2048/
Linear Regression: https://xkcd.com/1725/

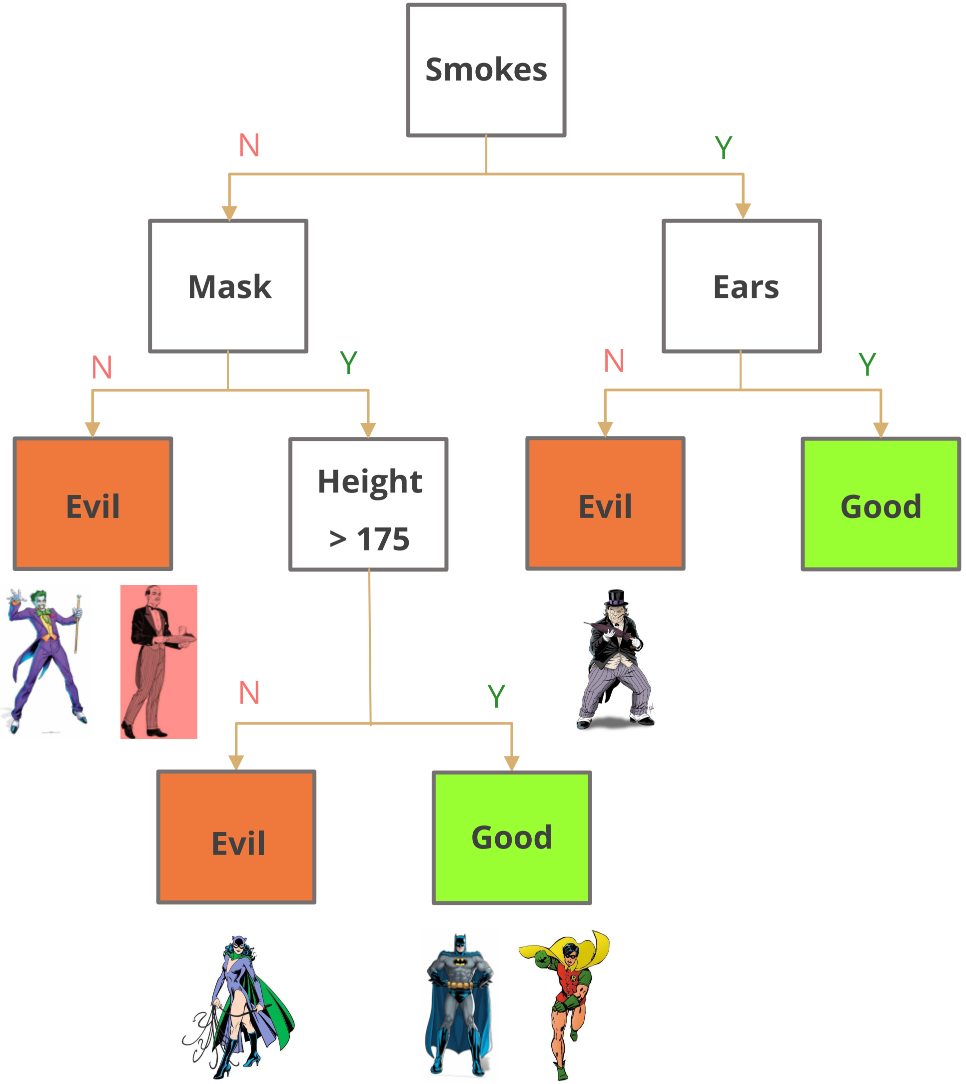





	Mask	Cape	Tie	Ears	Smokes	Height	Class
Batman	Y	Y	N	Y	N	180	**Good**
Robin	Y	Y	N	N	N	176	**Good**
Alfred	N	N	Y	N	N	185	**Good**
Penguin	N	N	Y	N	Y	140	**Evil**
Catwoman	Y	N	N	Y	N	170	**Evil**
Joker	N	N	N	N	N	179	**Evil**

Question:

Is this a good tree? Would it misclassify anybody?

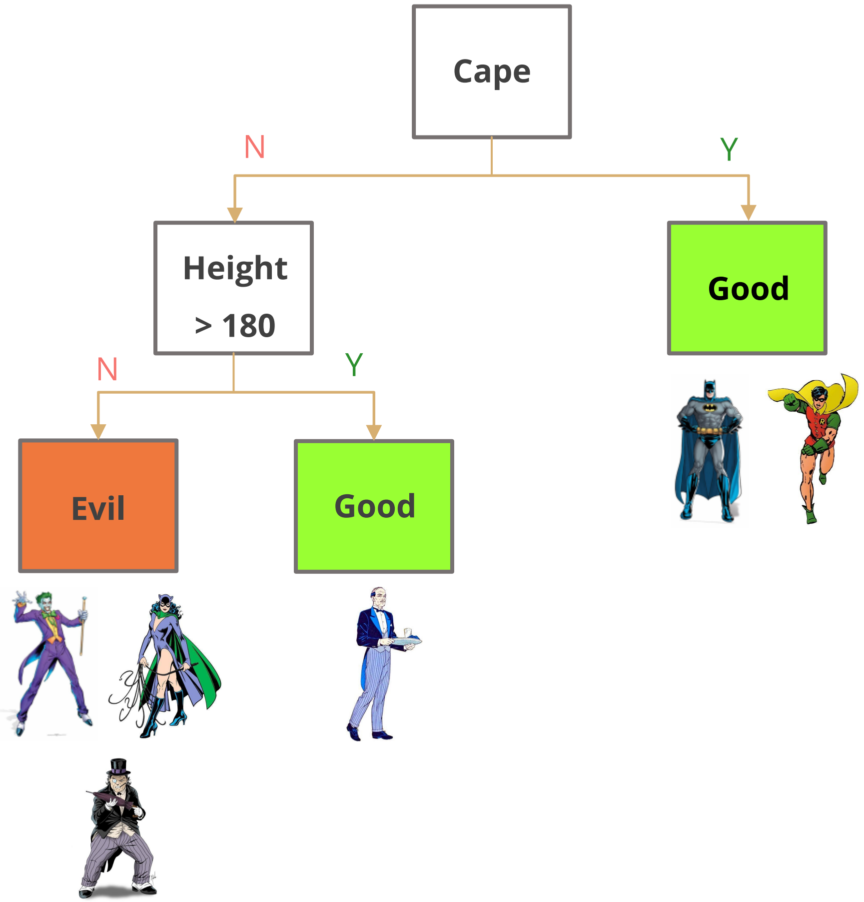


Source: ML Lecture 29 / Cornell CS4780 (YouTube)

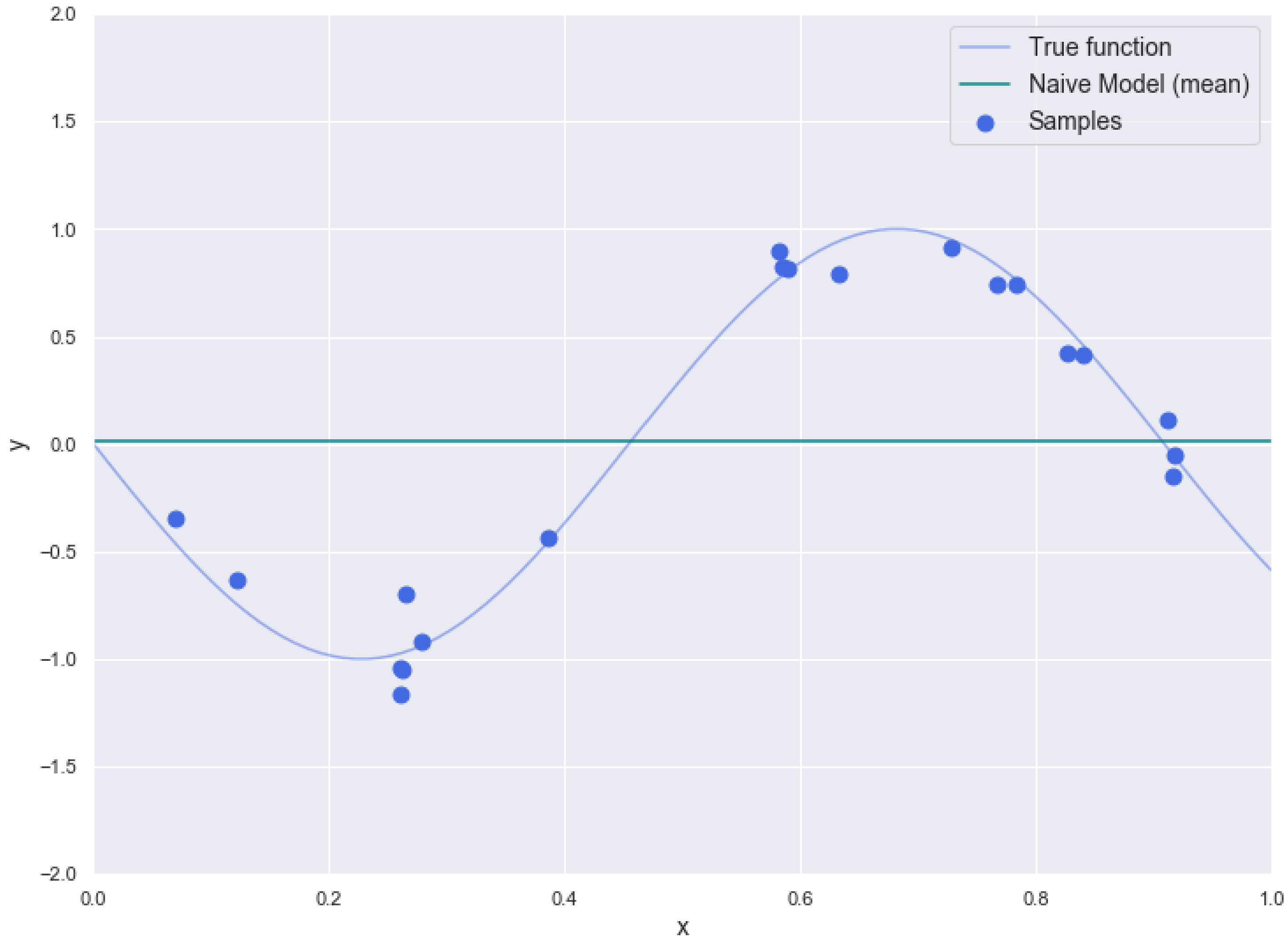
	Mask	Cape	Tie	Ears	Smokes	Height	Class
Batman	Y	Y	N	Y	N	180	**Good**
Robin	Y	Y	N	N	N	176	**Good**
Alfred	N	N	Y	N	N	185	**Good**
Penguin	N	N	Y	N	Y	140	**Evil**
Catwoman	Y	N	N	Y	N	170	**Evil**
Joker	N	N	N	N	N	179	**Evil**

Question:

What's the smallest possible tree that doesn't misclassify anybody?

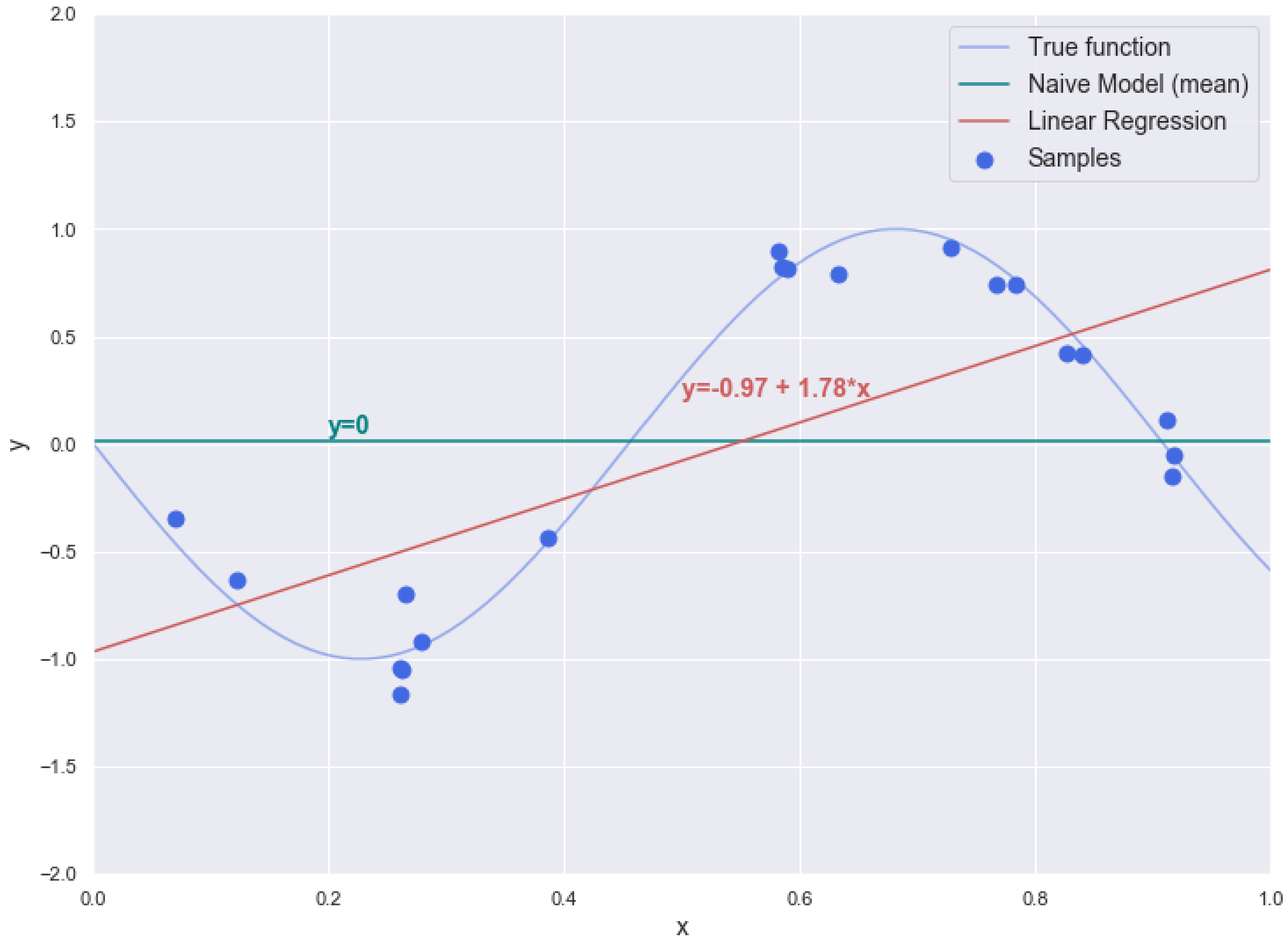


Source: ML Lecture 29 / Cornell CS4780 (YouTube)



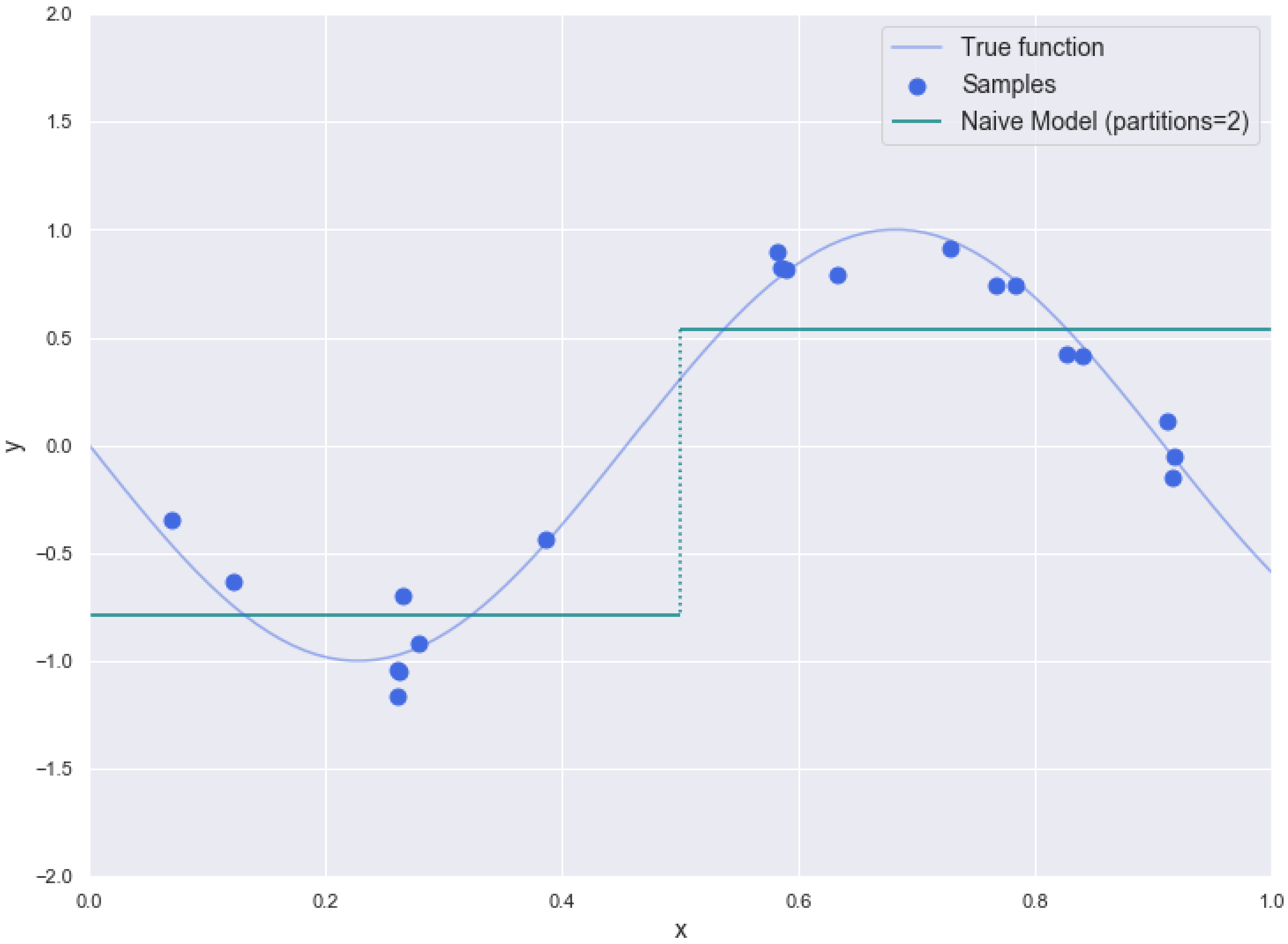
MSE = **0.53**

R-Squared = **0.00**



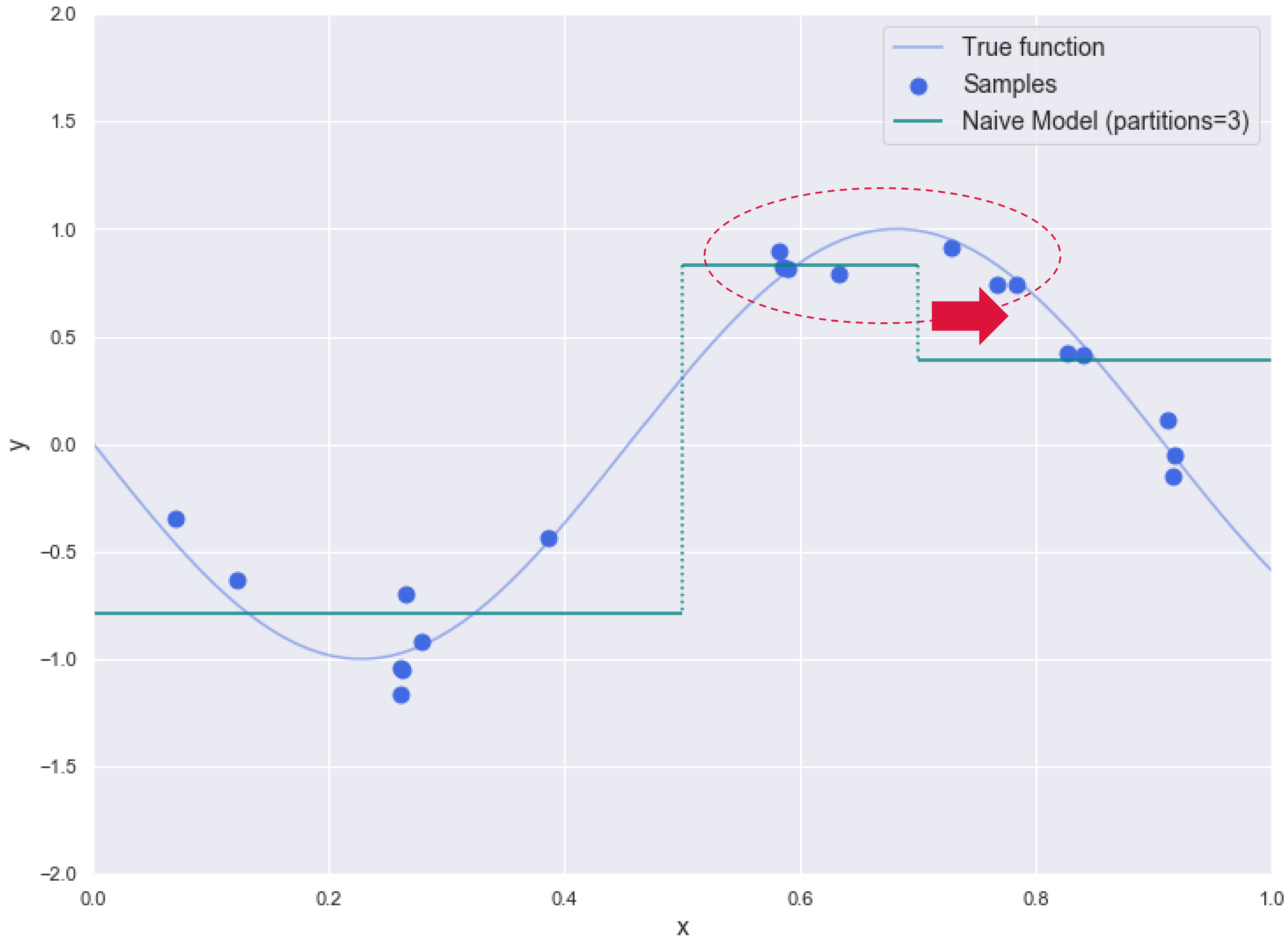
MSE = **0.29**

MSE = **0.53**



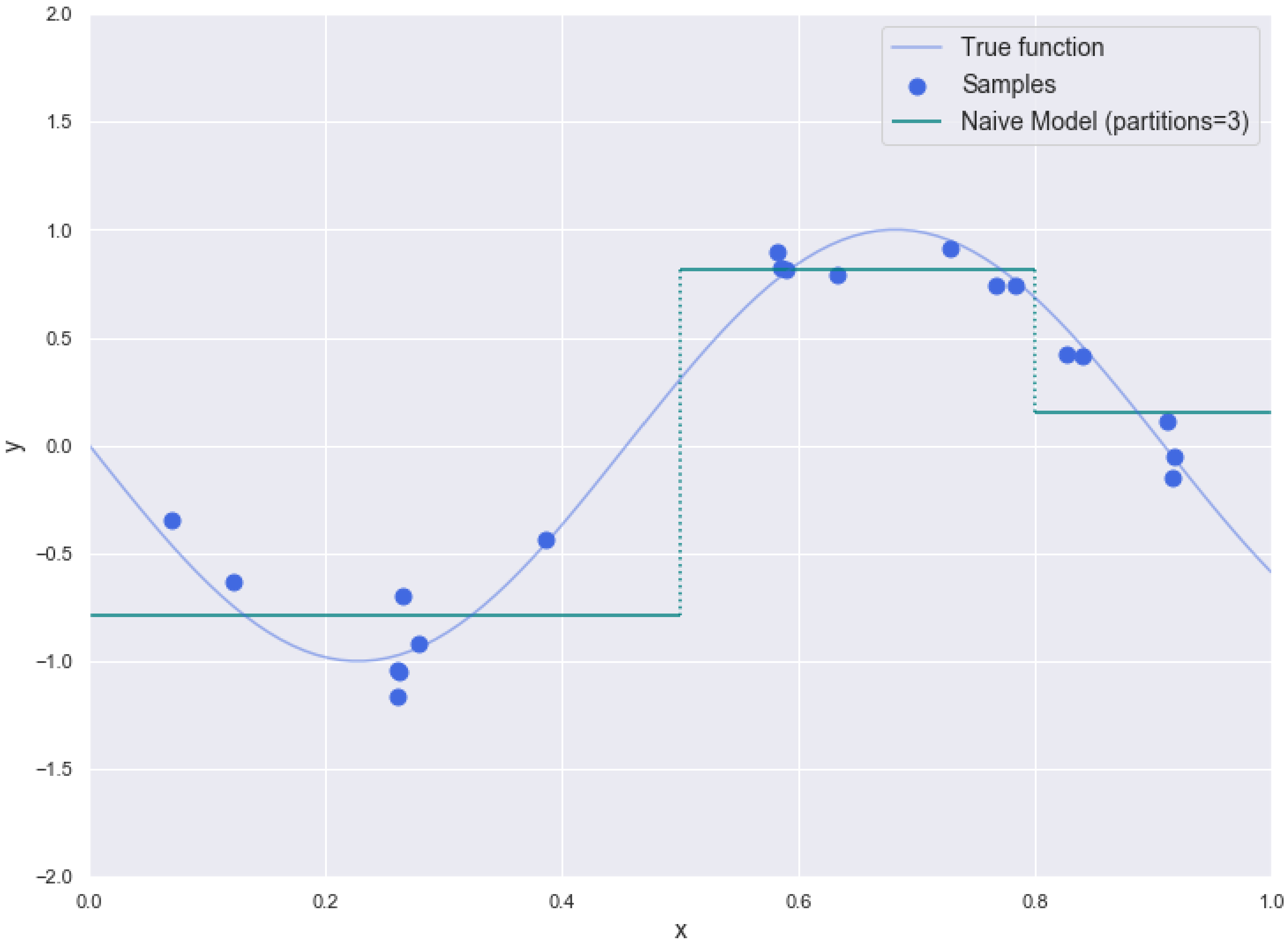
True function
Samples
Naive Model (partitions=2)
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
y
0.0
0.2
0.4
0.6
0.8
1.0
x

MSE = **0.11**

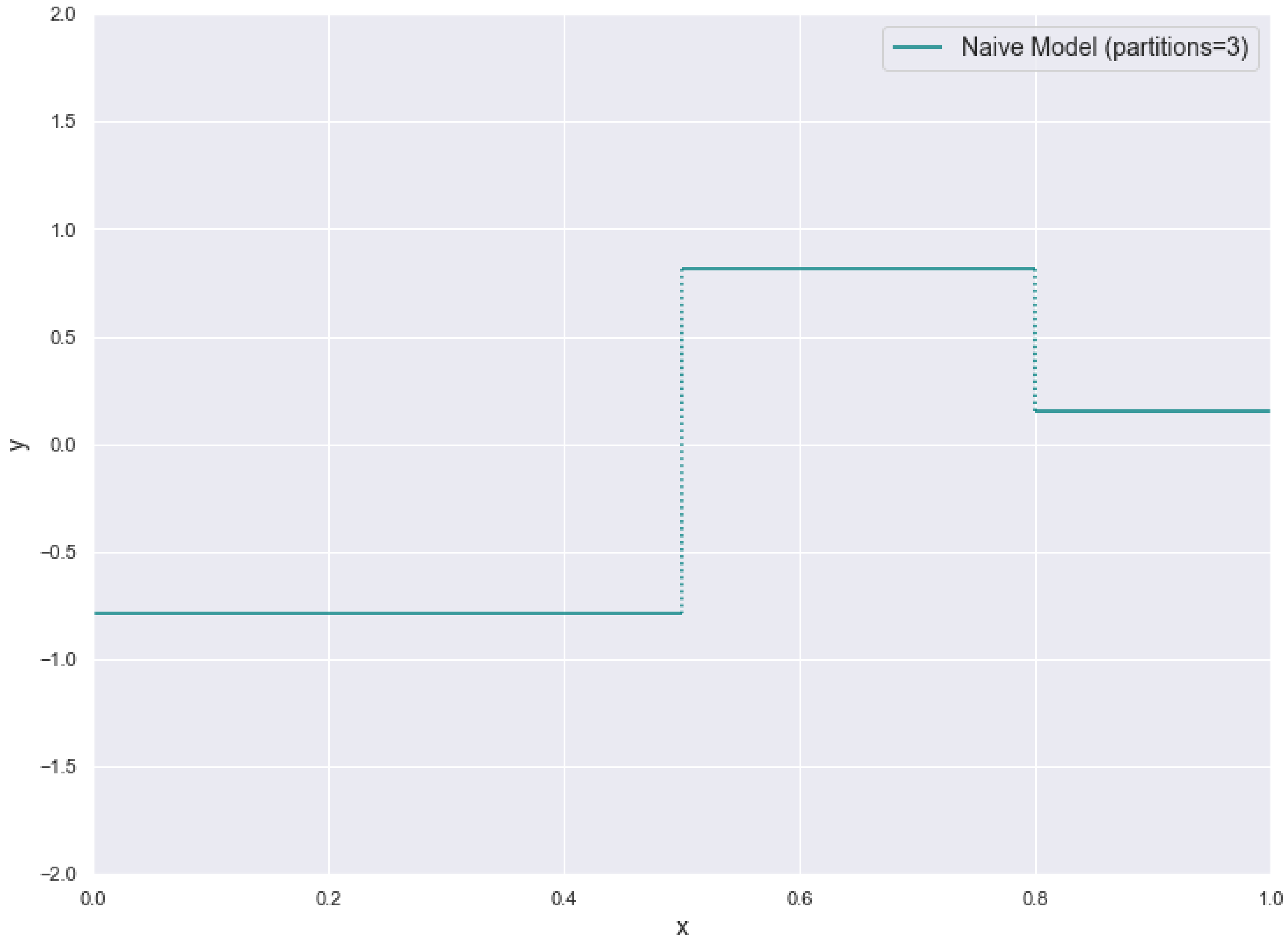


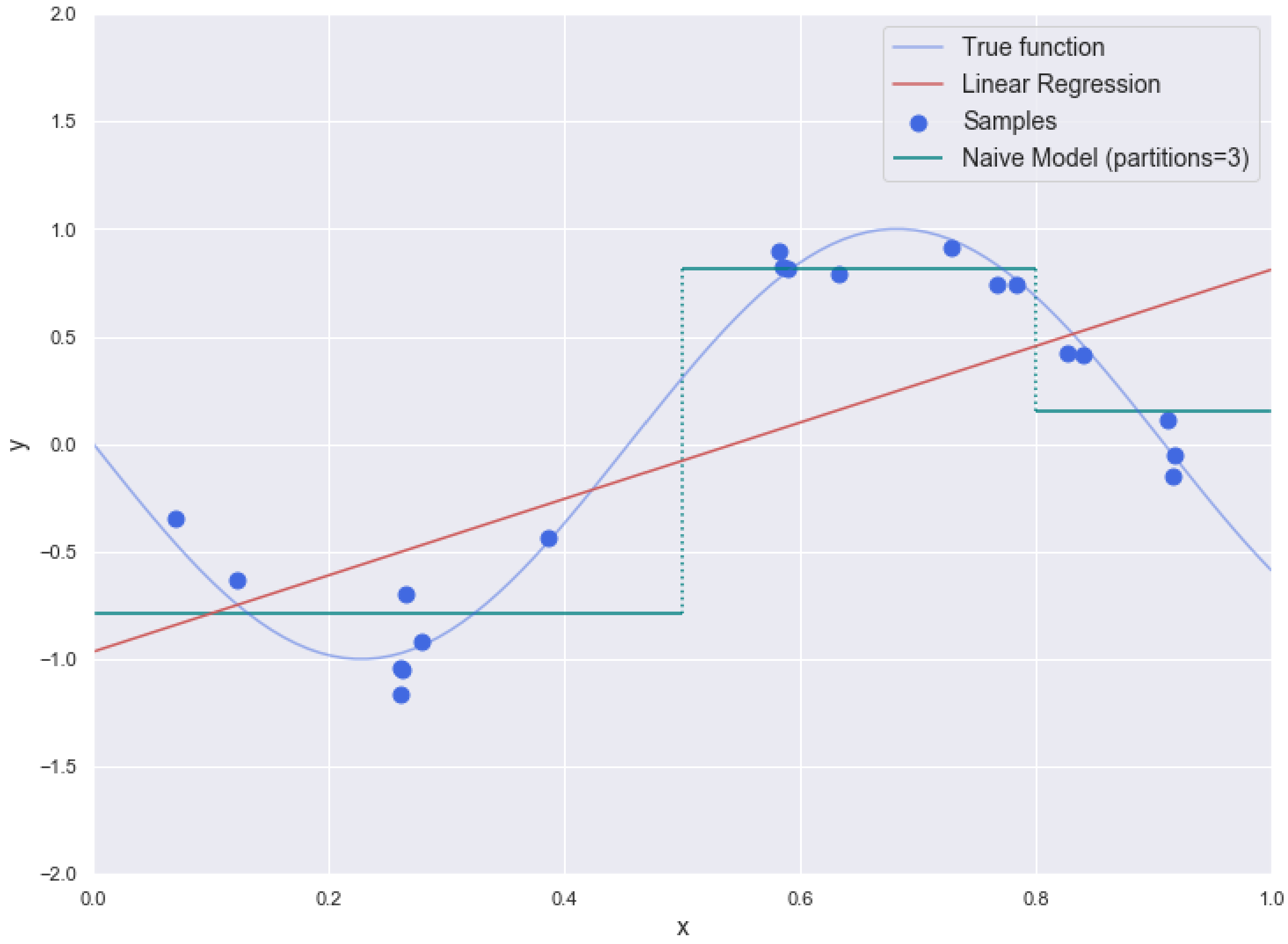
True function
Samples
Naive Model (partitions=3)
y
x
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0.0
0.2
0.4
0.6
0.8
1.0

MSE = **0.09**


True function
Samples
Naive Model (partitions=3)
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
y
0.0
0.2
0.4
0.6
0.8
1.0
x

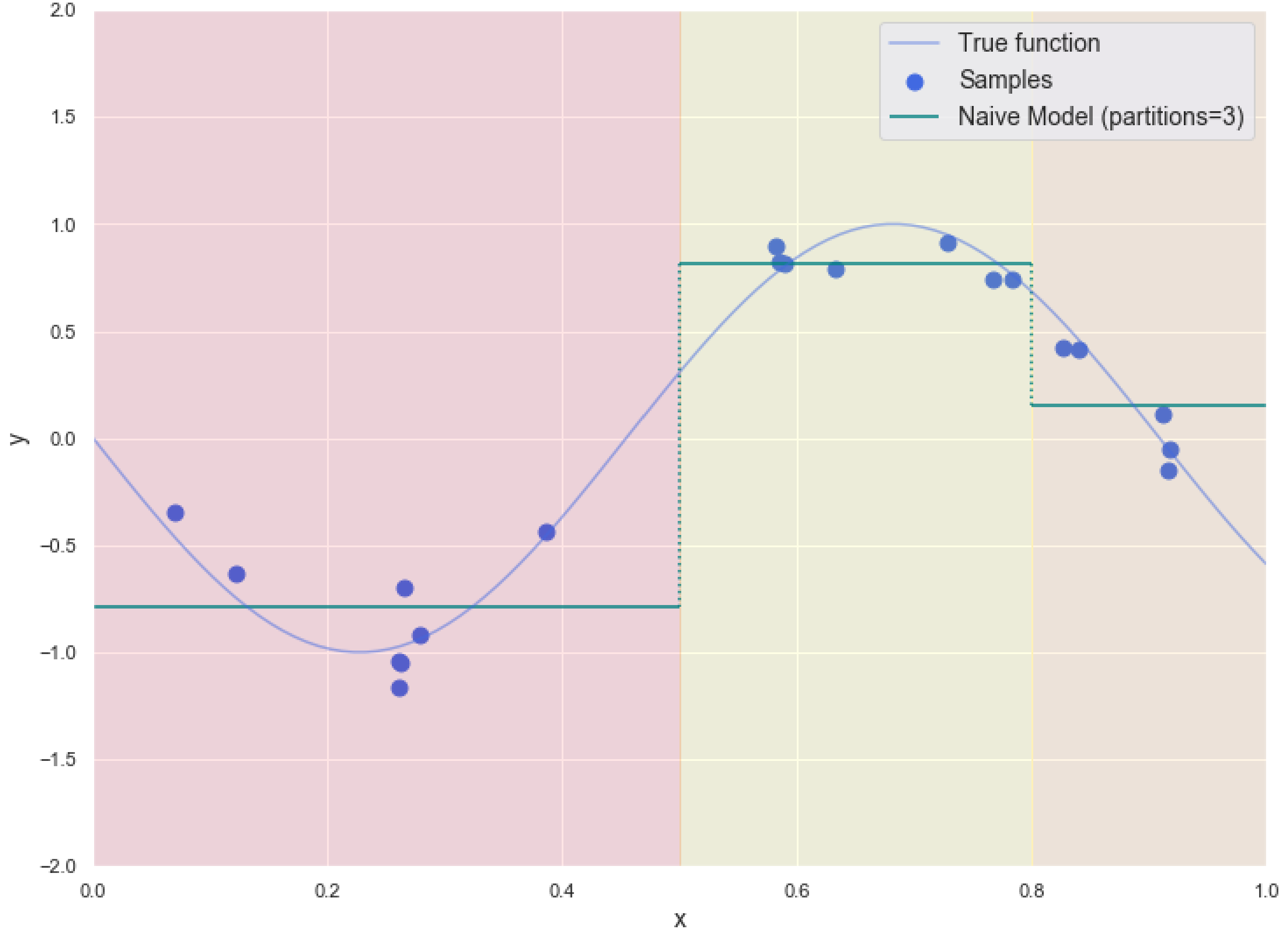
MSE = **0.05**

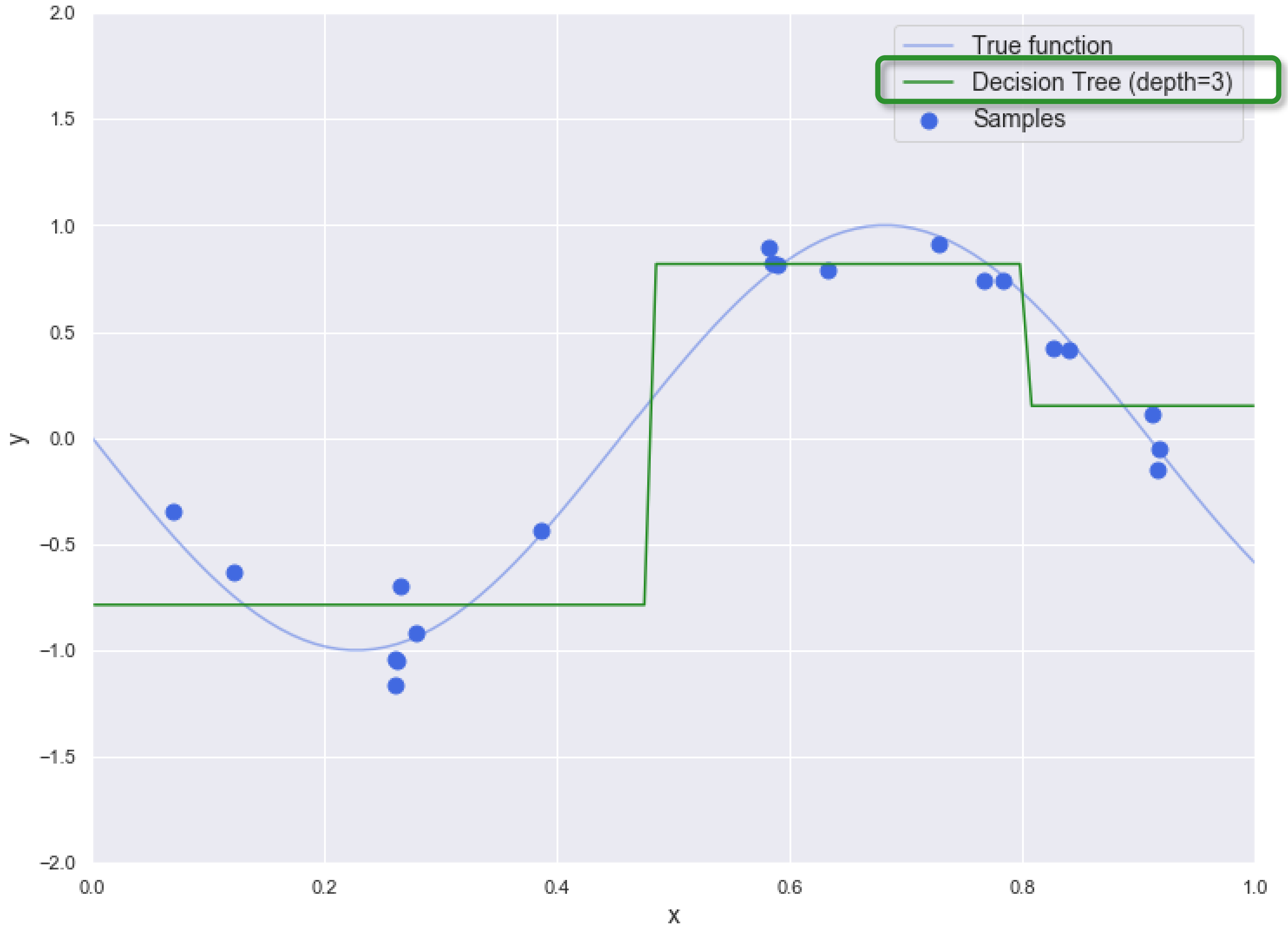

Naive Model (partitions=3)
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
y
0.0
0.2
0.4
0.6
0.8
1.0
x



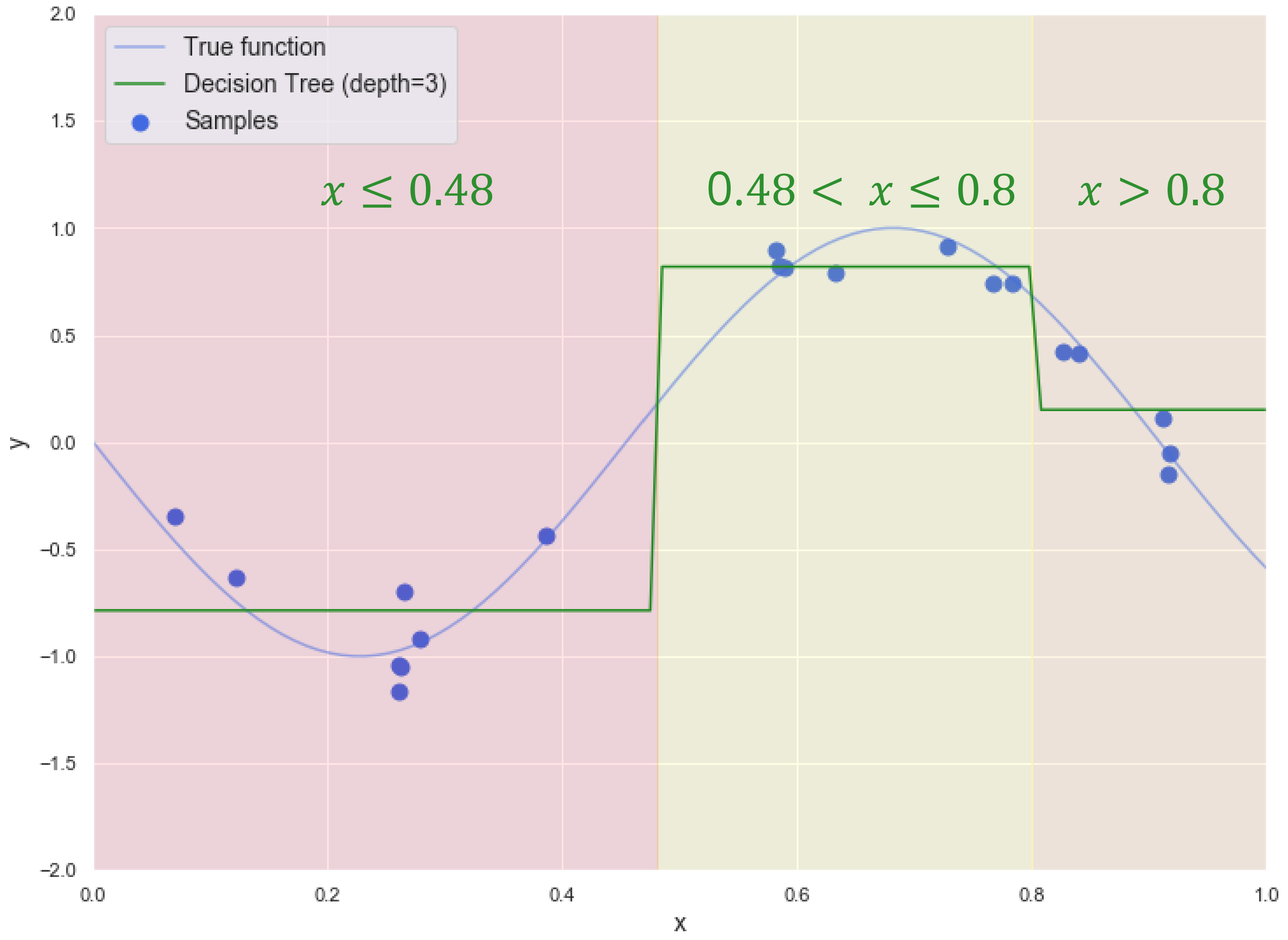
MSE = **0.29**

MSE = **0.05**

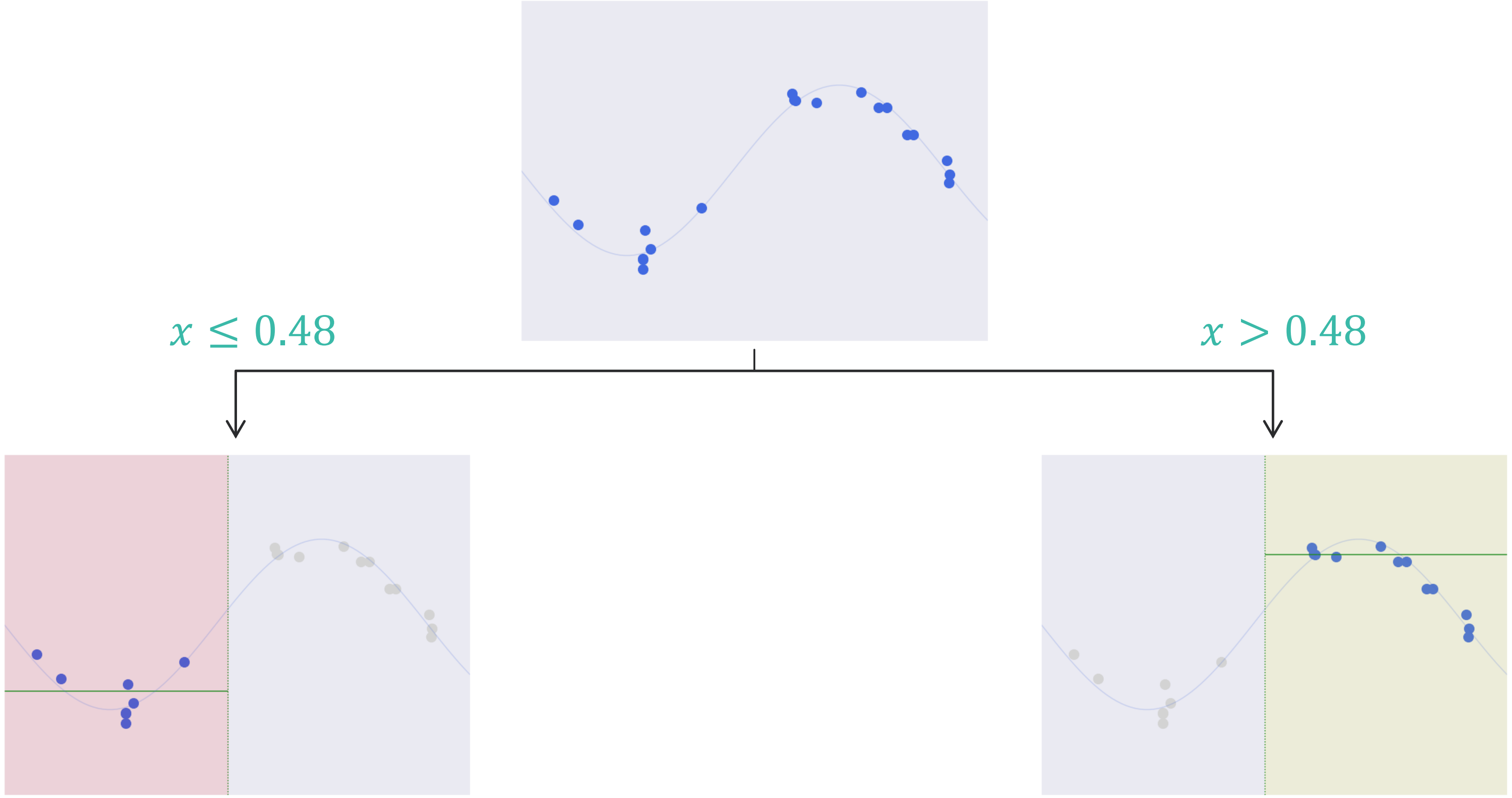

True function
Samples
Naive Model (partitions=3)
y
x
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0.0
0.2
0.4
0.6
0.8
1.0

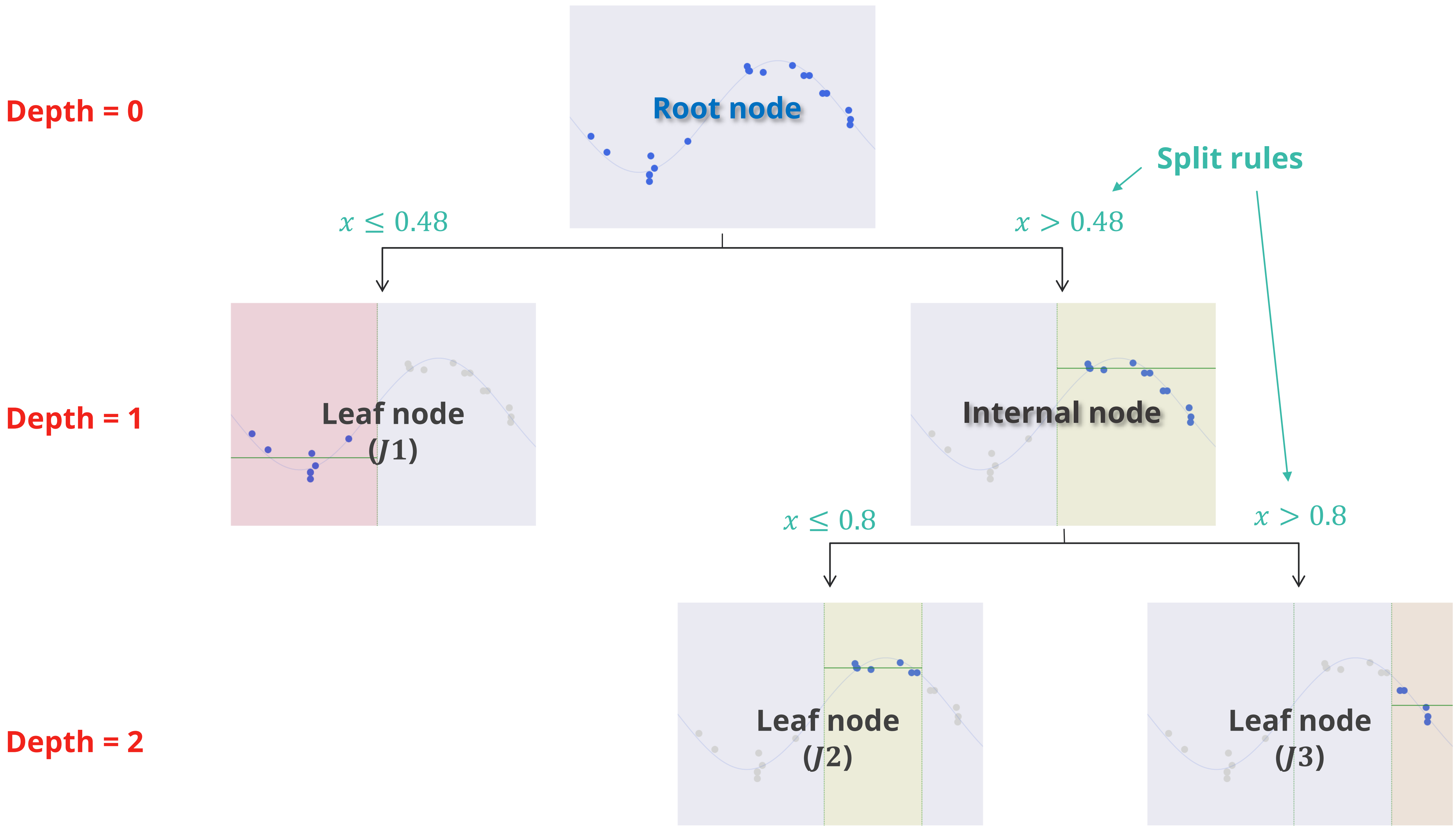


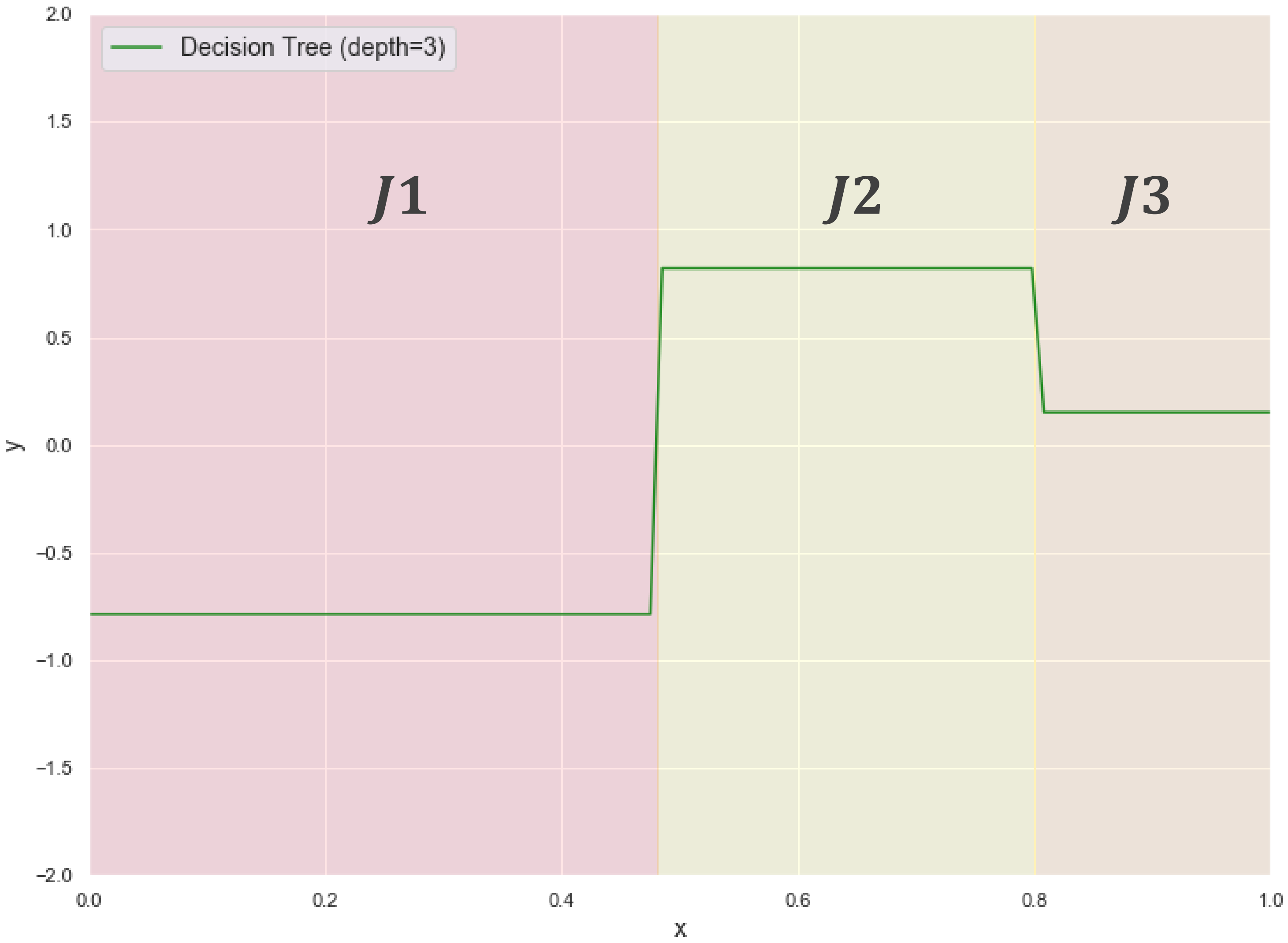
Decision Tree Regressor



Recursive Partitioning


$x \leq 0.48$
$x > 0.48$

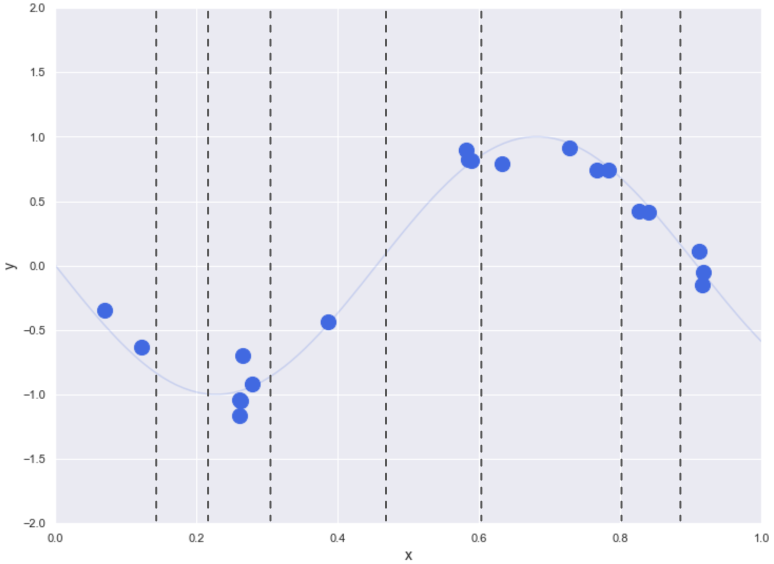

Depth = 0
Root node
Split rules
$x \leq 0.48$
$x > 0.48$
Depth = 1
Leaf node
$(J1)$
Internal node
$x \leq 0.8$
$x > 0.8$
Depth = 2
Leaf node
$(J2)$
Leaf node
$(J3)$

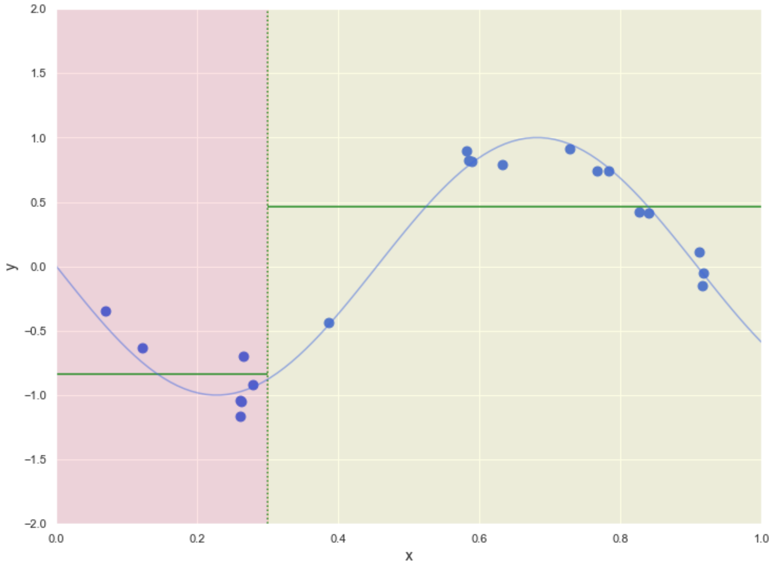
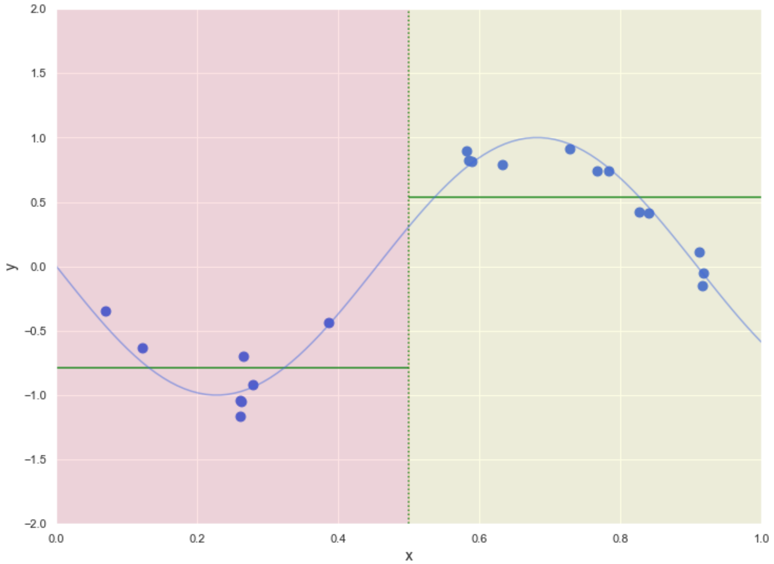
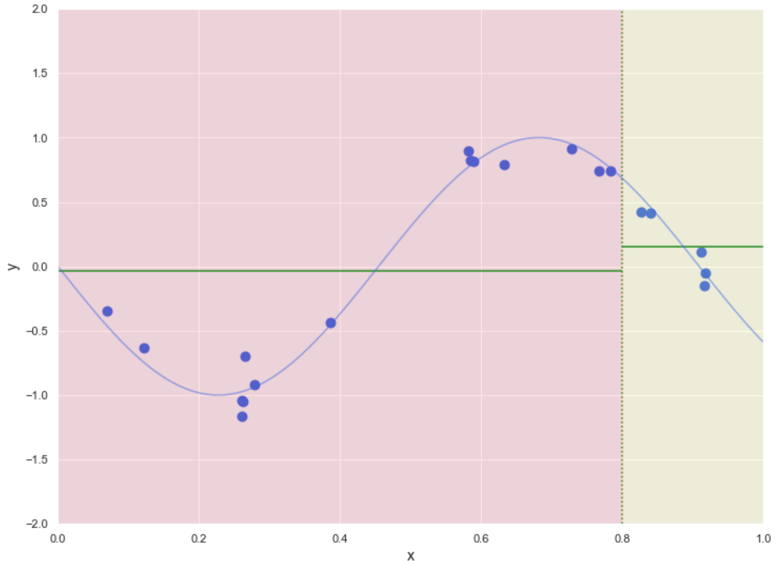


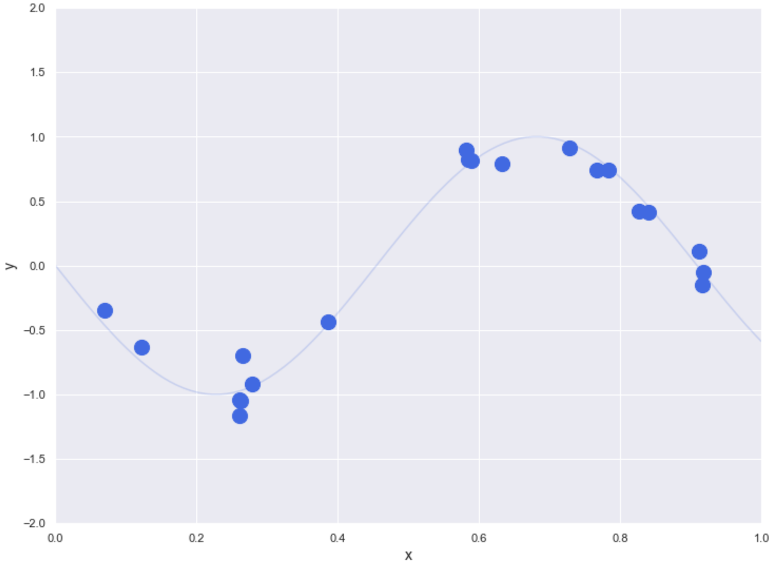
$J1, J2, J3$ = Leaf (terminal) nodes

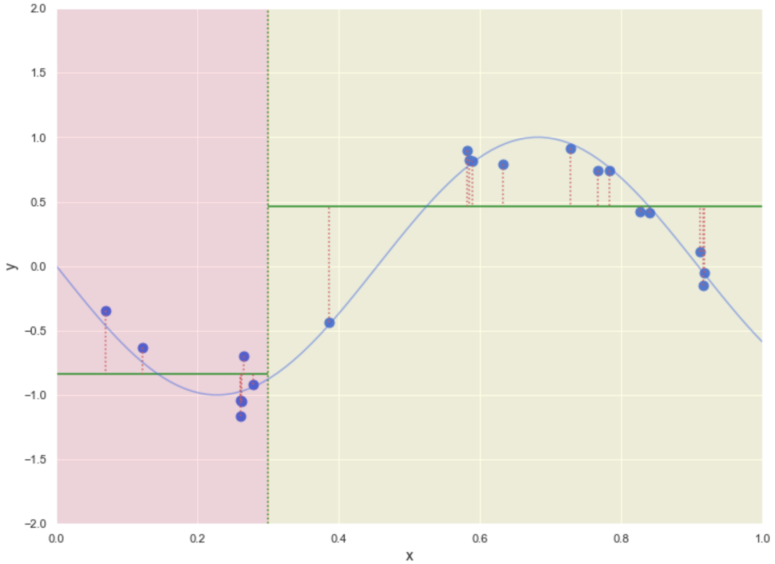
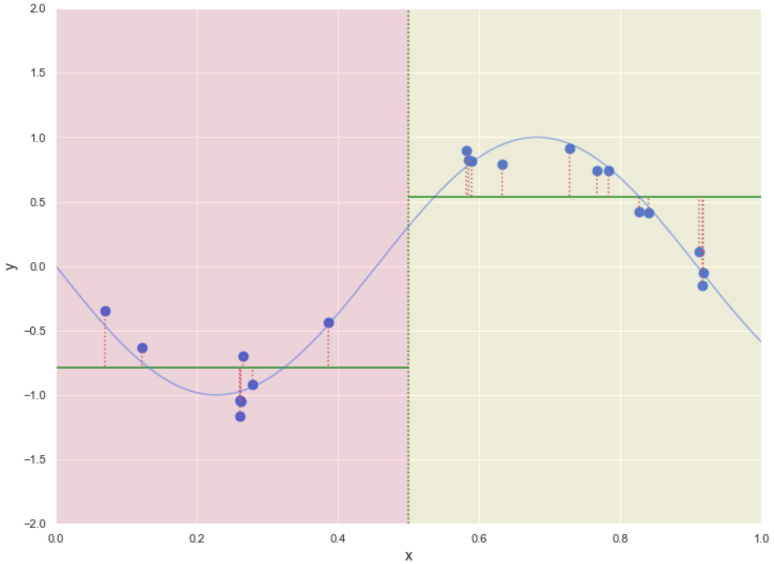
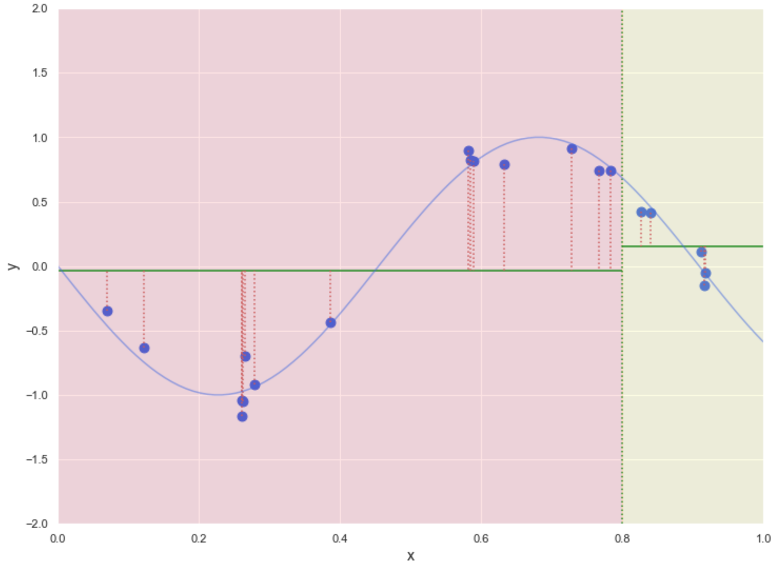
1. How to partition the data?
2. When to stop?

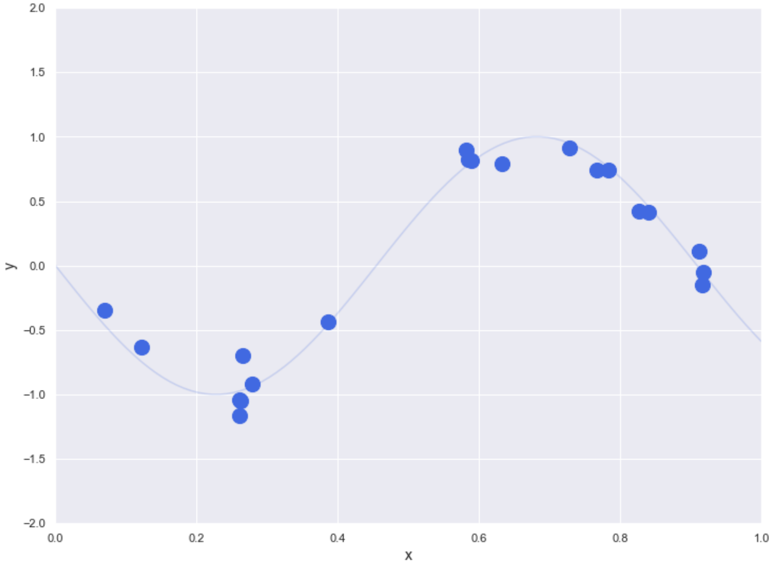
1. How to partition the data?
2. When to stop?



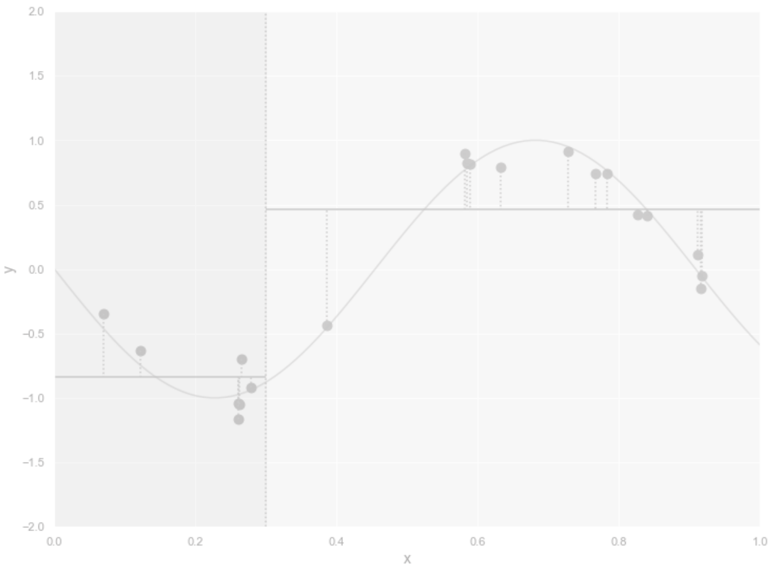
Option 1	Option 2	...	Option n
			



Option 1	Option 2	…	Option n
			
MSE = 0.15	**MSE = 0.11**		**MSE = 0.53**

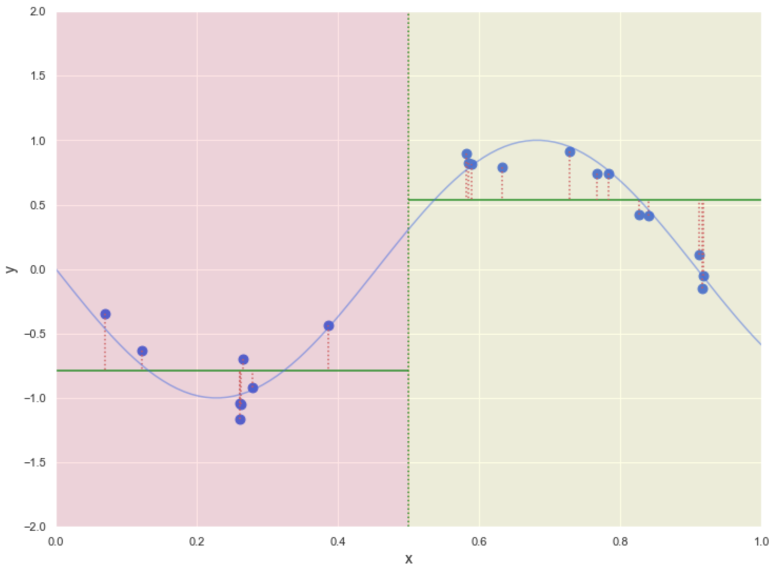


Option 1



MSE = 0.15

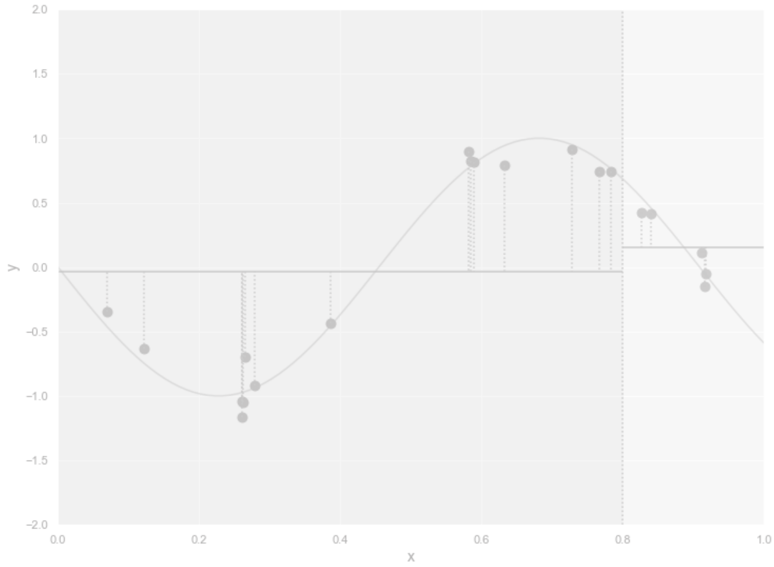
Option 2



MSE = 0.11

...

Option n



MSE = 0.53

LINEAR REGRESSION

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

CISION TREE REGRESSION

$$MSE = \frac{1}{n}\sum_{j=1}^{J}\sum_{i\in R_j}(y_i - \hat{y}_{R_j})^2$$

class sklearn.tree.DecisionTreeRegressor(

criterion='squared_error',

splitter='best',

max_depth=None,

min_samples_split=2,

min_samples_leaf=1,

min_weight_fraction_leaf=0.0,

max_features=None,

random_state=None,

max_leaf_nodes=None,

min_impurity_decrease=0.0,

ccp_alpha=0.0)

1

```
# Import
from sklearn.tree import DecisionTreeRegressor
```

2

```
# Define
tree = DecisionTreeRegressor
```

3

```
# Fit
tree.fit(X_train, y)
```

4

```
# Predict
tree.predict(X_test)
```

https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html

```
class sklearn.tree.DecisionTreeRegressor(
    criterion='squared_error',
    splitter='best',
    max_depth=None,
    min_samples_split=2,
    min_samples_leaf=1,
    min_weight_fraction_leaf=0.0,
    max_features=None,
    random_state=None,
    max_leaf_nodes=None,
    min_impurity_decrease=0.0,
    ccp_alpha=0.0)
```

The function to measure the quality of a split.

'squared_error' = Mean Squared Error

$$MSE = \frac{1}{n}\sum_{j=1}^{J}\sum_{i\in R_j}(y_i - \hat{y}_{R_j})^2$$

1. How to partition the data?
2. When to stop?

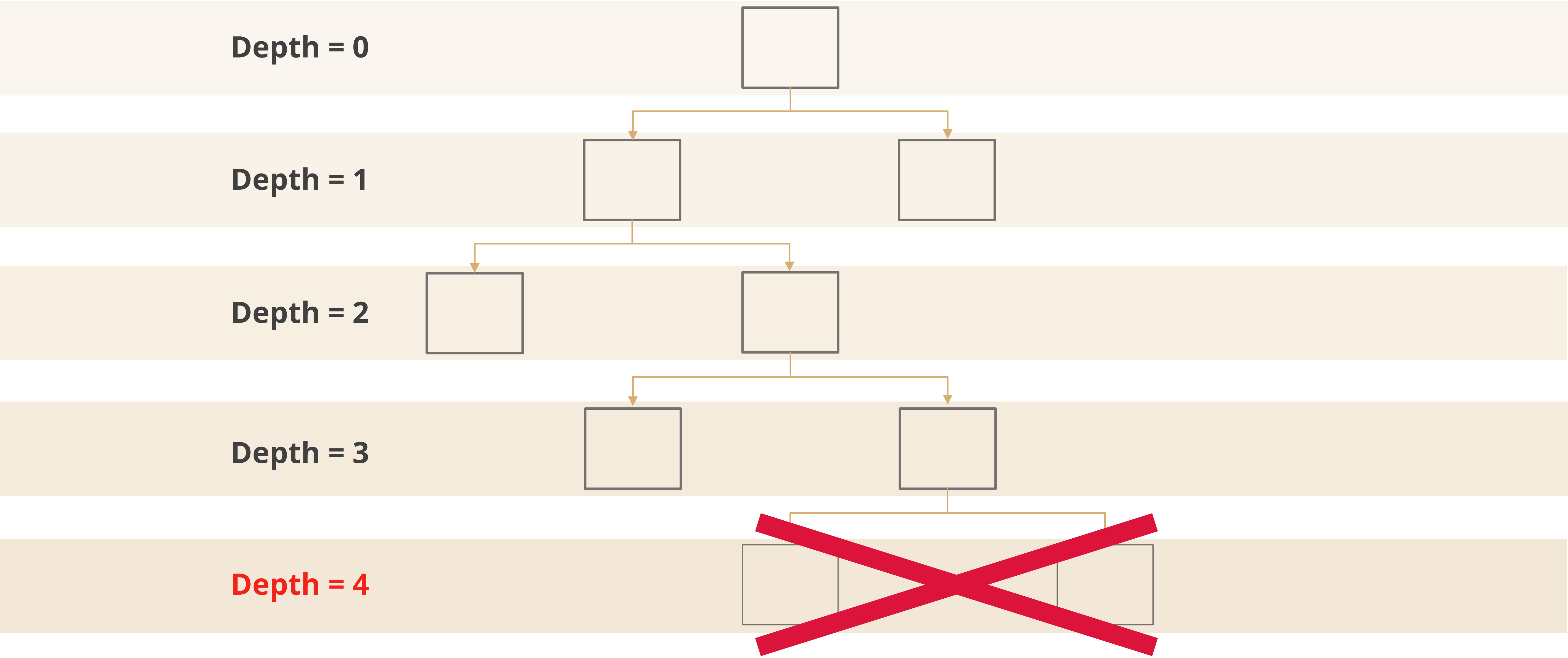
```
class sklearn.tree.DecisionTreeRegressor(
    criterion='squared_error',
    splitter='best',
    max_depth=None,
    min_samples_split=2,
    min_samples_leaf=1,
    min_weight_fraction_leaf=0.0,
    max_features=None,
    random_state=None,
    max_leaf_nodes=None,
    min_impurity_decrease=0.0,
    ccp_alpha=0.0)
```

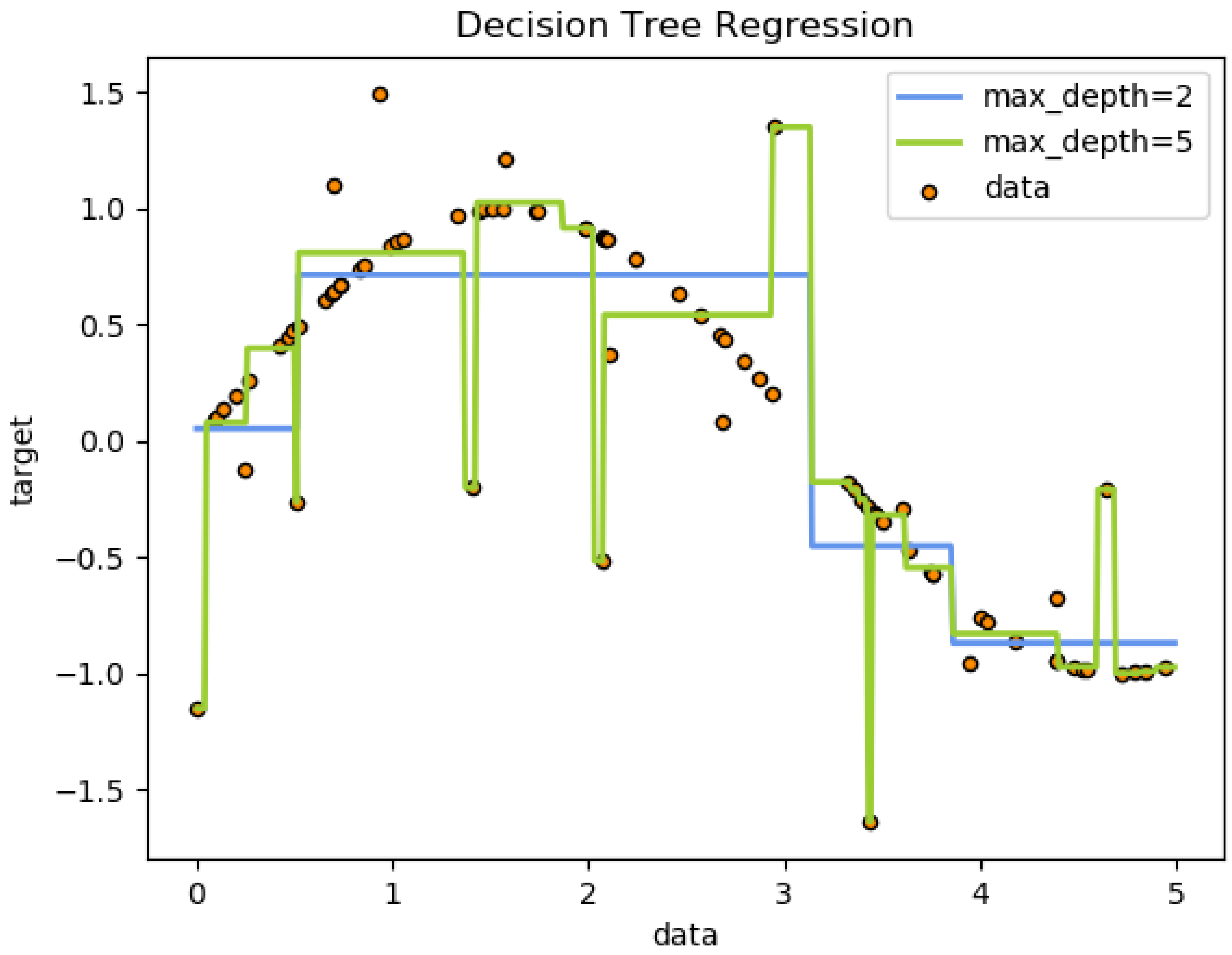
The maximum depth of the tree.

If None, then nodes are expanded until all leaves are pure or until all leaves contain less than `min_samples_split` samples.

Recommendation: `max_depth` start between 6 and 10

If `max_depth` is set to 3...





Decision Tree Regression
max_depth=2
max_depth=5
data
target
data
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0
1
2
3
4
5

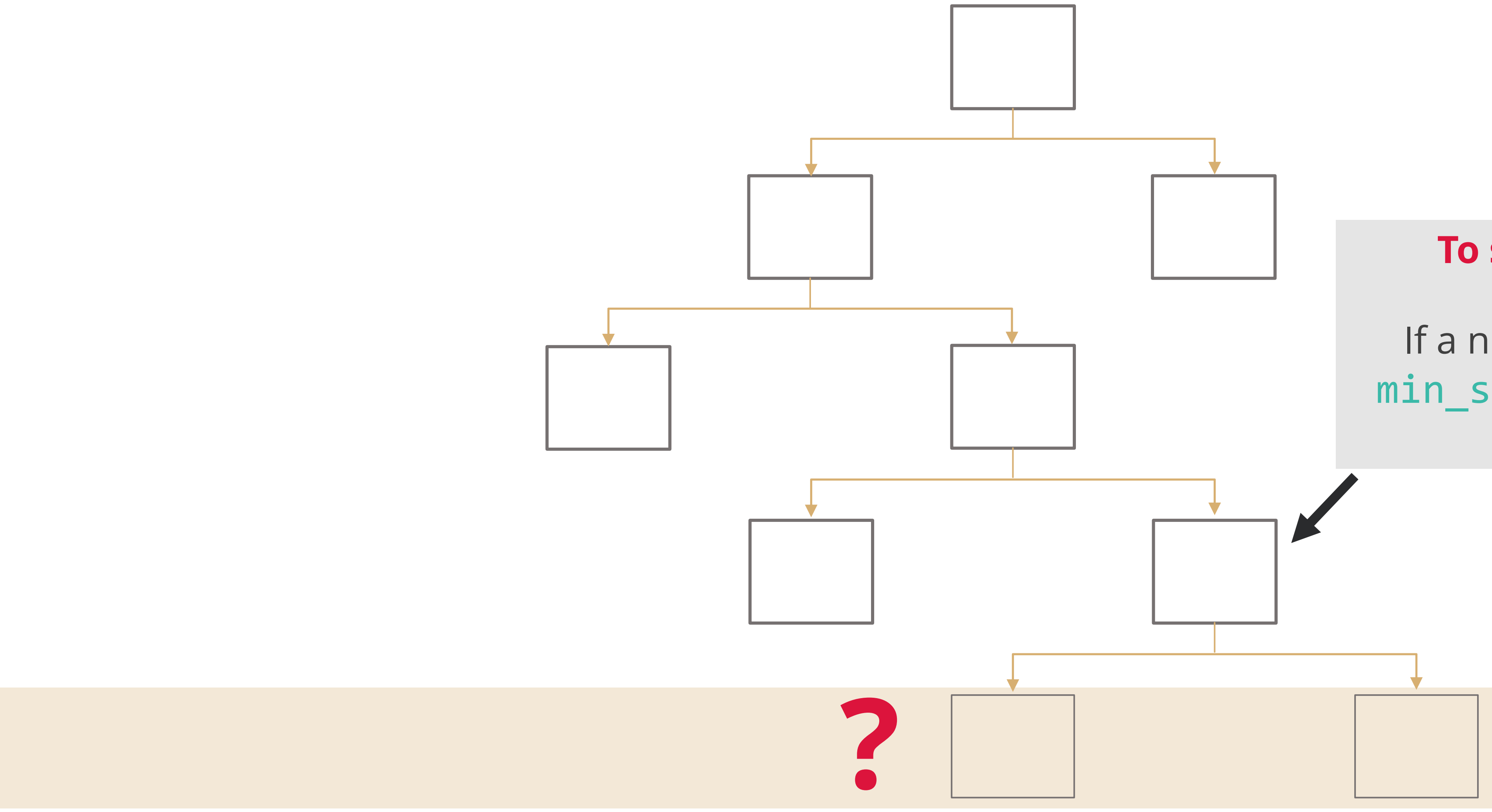
class sklearn.tree.DecisionTreeRegressor(
criterion='squared_error',
splitter='best',
max_depth=None,
min_samples_split=2,
min_samples_leaf=1,
min_weight_fraction_leaf=0.0,
max_features=None,
random_state=None,
max_leaf_nodes=None,
min_impurity_decrease=0.0,
ccp_alpha=0.0)

The minimum number of samples required to split an internal node:

If `int`, then consider `min_samples_split` as the minimum number.

If `float`, then `ceil(min_samples_split * n_samples)` are the minimum number of samples for each split.

Recommendation: `min_samples_split = 0.04`



To split or not to split.

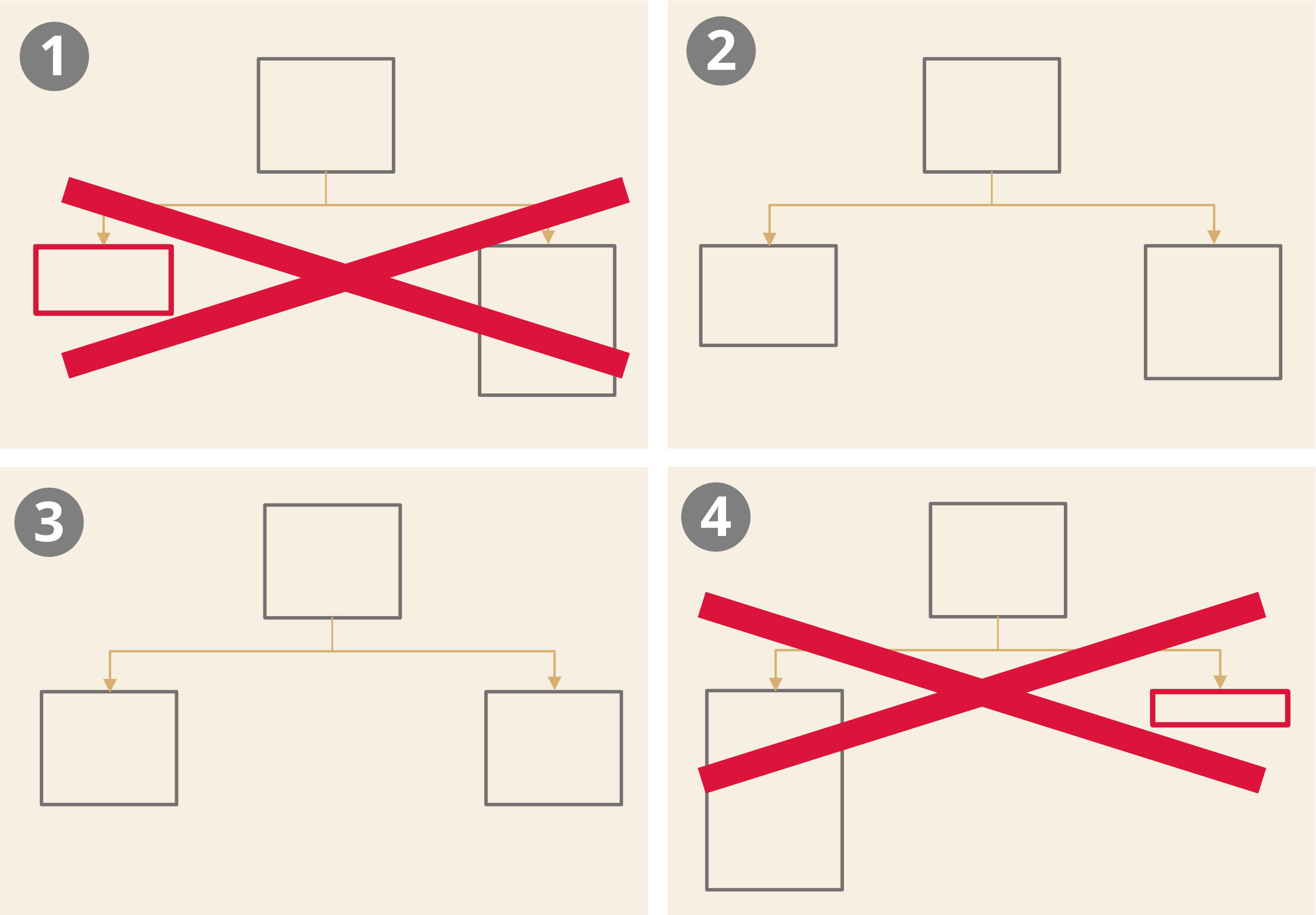
If a node contains less than `min_sample_split` records, then don't split.

```
class sklearn.tree.DecisionTreeRegressor(
    criterion='squared_error',
    splitter='best',
    max_depth=None,
    min_samples_split=2,
    min_samples_leaf=1,
    min_weight_fraction_leaf=0.0,
    max_features=None,
    random_state=None,
    max_leaf_nodes=None,
    min_impurity_decrease=0.0,
    ccp_alpha=0.0)
```

The minimum number of samples required to be at a leaf node.

A split point at any depth will only be considered if it leaves at least `min_samples_leaf` training samples in each of the left and right branches.

Recommendation: `min_samples_leaf = 0.02`



Should a split be considered?

If a split results
in a children node with less than
`min_samples_leaf` records,
then discard that split.

```
class sklearn.tree.DecisionTreeRegressor(
    criterion='squared_error',
    splitter='best',
    max_depth=None,
    min_samples_split=2,
    min_samples_leaf=1,
    min_weight_fraction_leaf=0.0,
    max_features=None,
    random_state=None,
    max_leaf_nodes=None,
    min_impurity_decrease=0.0,
    ccp_alpha=0.0)
```

The number of features to consider when looking for the best split.

- If int, then consider max_features features at each split.
- If float, then max_features is a fraction
  and int(max_features * n_features) features are considered at each split.
- If "auto", then max_features=n_features.
- If "sqrt", then max_features=sqrt(n_features).
- If "log2", then max_features=log2(n_features).
- If None, then max_features=n_features.

Recommendation: Consider using 'sqrt' or 'log2' if training on a large dataset; otherwise, leave to None.

```
class sklearn.tree.DecisionTreeRegressor(
    criterion='squared_error',
    splitter='best',
    max_depth=None,
    min_samples_split=2,
    min_samples_leaf=1,
    min_weight_fraction_leaf=0.0,
    max_features=None,
    random_state=None,
    max_leaf_nodes=None,
    min_impurity_decrease=0.0,
    ccp_alpha=0.0)
```

Set a user-defined seed for reproducible results.

If `int`, `random_state` is the seed used by the random number generator.

Recommendation: Always set a seed (e.g., 314) to ensure reproducible results.

Decision Tree Tutorial

`05_decision_tree_intro.ipynb`

Decision Tree Algorithm

PSEUDOCODE

1. Start at the root node.
2. For each feature:
   - Identify the best split that minimizes MSE.
3. Identify the feature that generates the lowest MSE.
4. Split the node using that feature and its best split.
5. Repeat steps 2 thru 4 until a stopping criterion is met.

ID3
Iterative Dichotomizer

C4.5

CART
Classification and Regression Trees

sklearn

Decision Trees

- Simple and intuitive
- Can handle non-linear relationships
- Can handle both numeric and categorical variables†
- Not influenced heavily by outliers

† However, in scikit-learn, you still need to convert them into a numeric form.

However…

- **Decision tree** is a greedy algorithm; it tends to overfit on data with large number of features.

Recommendations:

1. Perform **feature reduction** before training a model.
2. Always use `min_samples_leaf` to control the amount of over-fitting.
3. Try a small tree first, using `max_depth`, and then grow further if necessary.

Data Science / ML Term	Multidisciplinary Synonyms
Label	• Dependent variable • Response variable • Target • Output
Features	• Independent variables • Explanatory variables • Attributes • Inputs • Predictors
(Regression) Coefficients	• Parameter estimates • Slopes
Noise	• Random error • Residuals
Cases	• Observations • Records • Rows
Train	• Fit • Build

Next Up

1. ~~Introduction~~
2. ~~The Data Science Process~~
3. **Supervised Learning: Classification**
4. Unsupervised Learning
5. The Grunt Work
6. Wrap Up