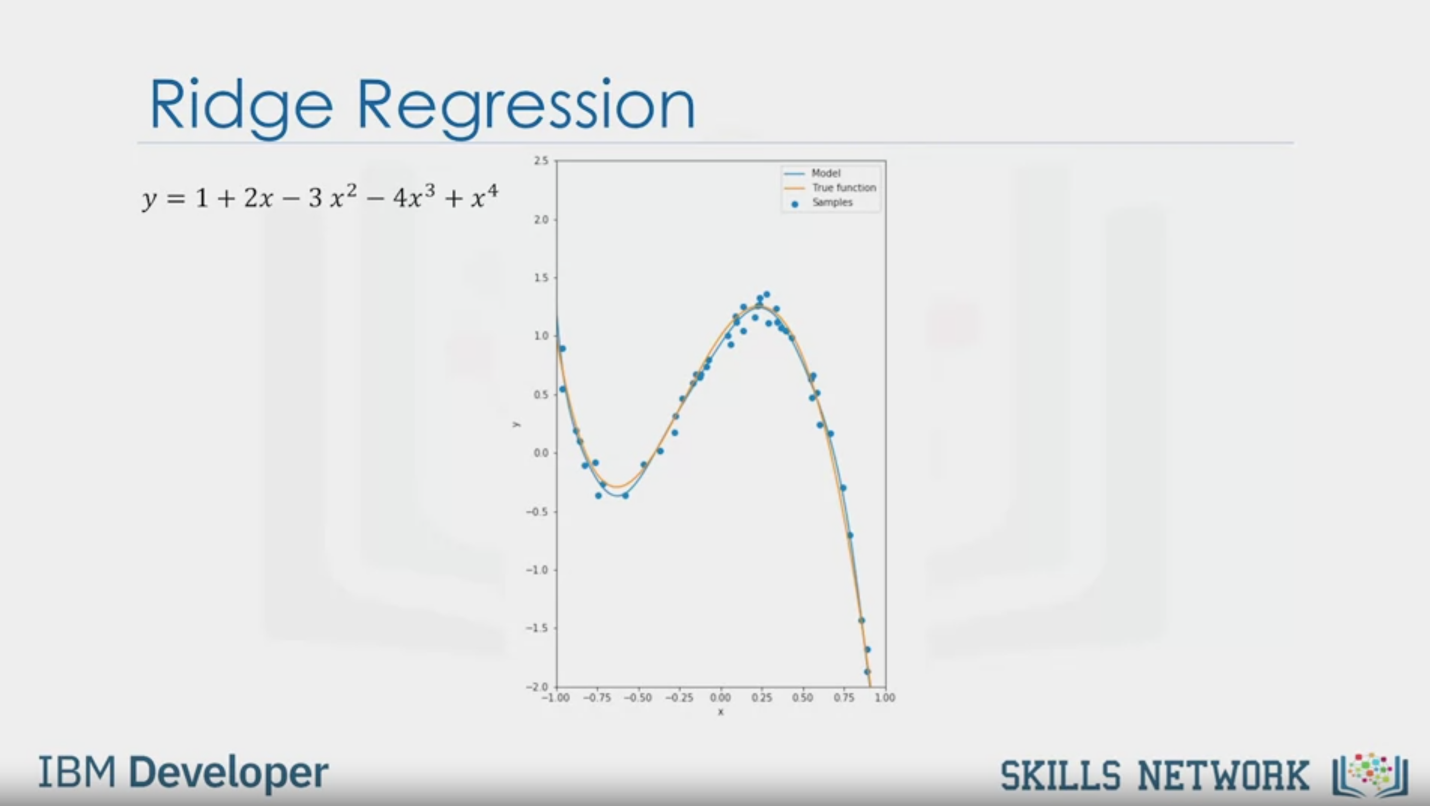


Ridge Regression

IBM Developer

SKILLS NETWORK

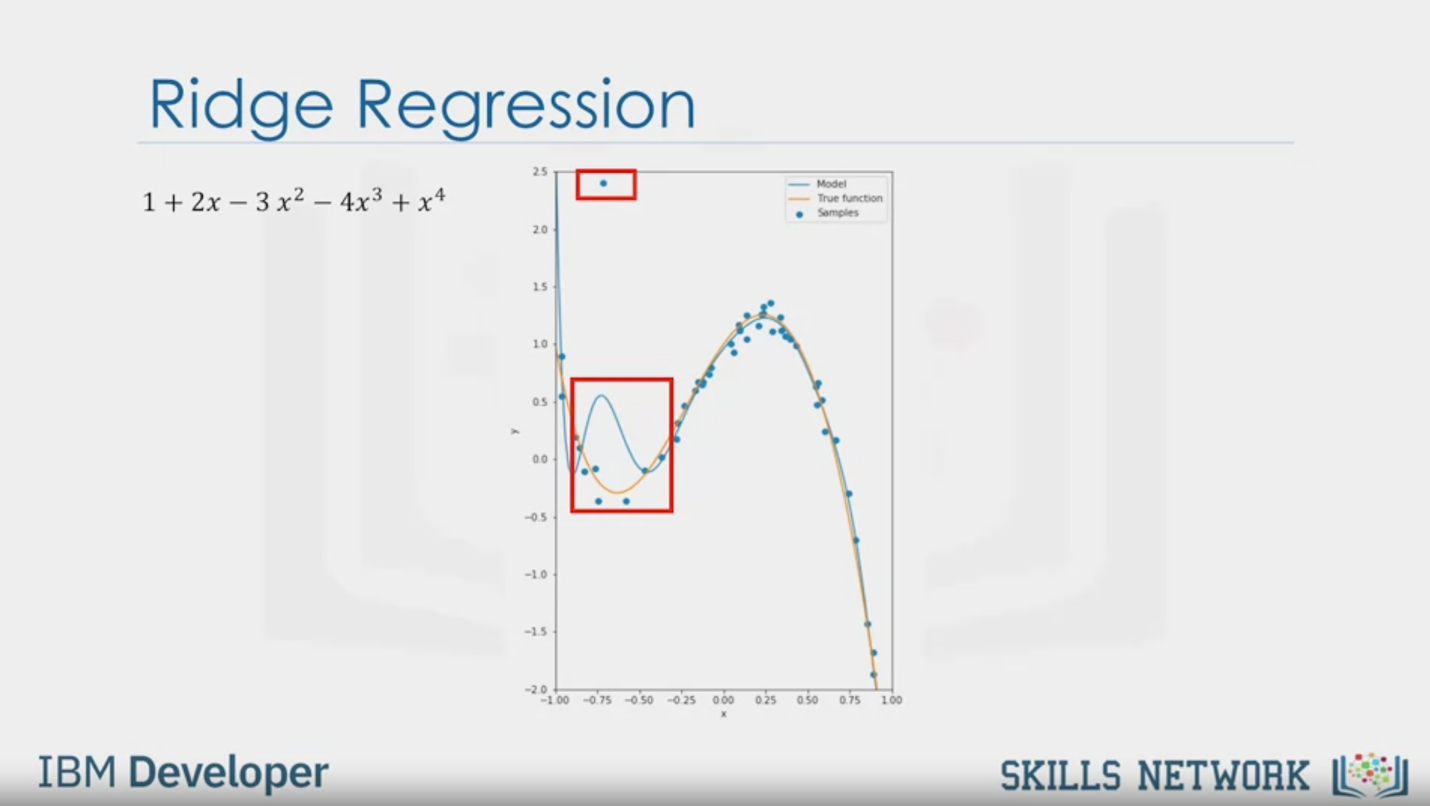


Ridge Regression

y =1 + 2x - 3x2- 4x3 + x4

IBM Developer

SKILLS NETWORK

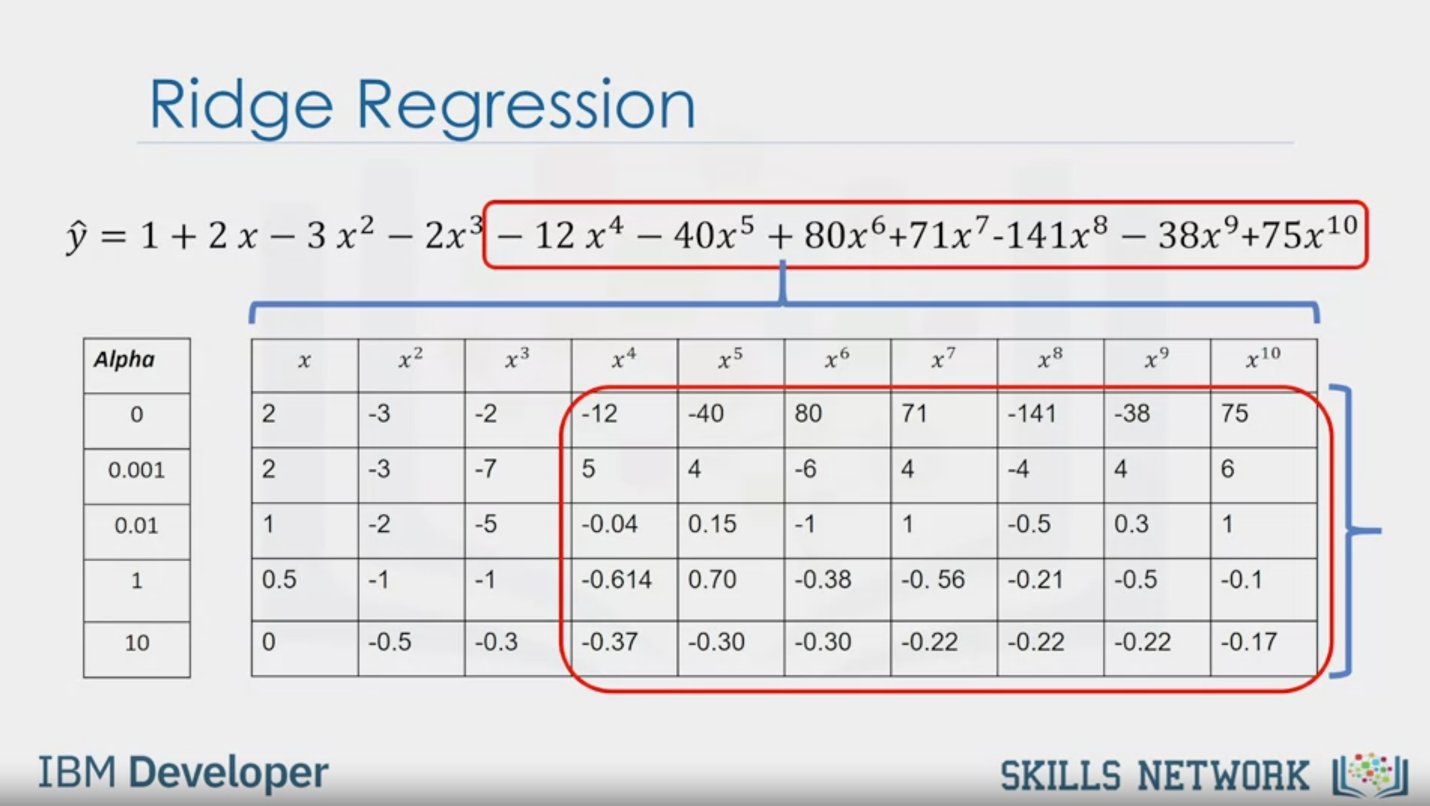


Ridge Regression

1 + 2x - 3x2- 4x3 + x4

IBM Developer

SKILLS NETWORK



Ridge Regression

Alpha

0

0.001

0.01

1

10

X

2

2

1

0.5

0

-3

-3

-2

-1

-0.5

-2

-7

-5

-1

-0.3

-12

5

-0.04

-0.614

-0.37

-40

4

0.15

0.70

-0.30

80

-6

-1

-0.38

-0.30

71

4

1

-0.56

-0.22

-141

-44

-0.5

-0.21

-0.22

-38

4

0.3

-0.5

-0.22

75

6

1

-0.1

-0.17

IBM Developer

SKILLS NETWORK



Ridge Regression

alpha

0

0.001

0.01

1

10

IBM Developer

SKILLS NETWORK



Ridge Regression

alpha

0

0.001

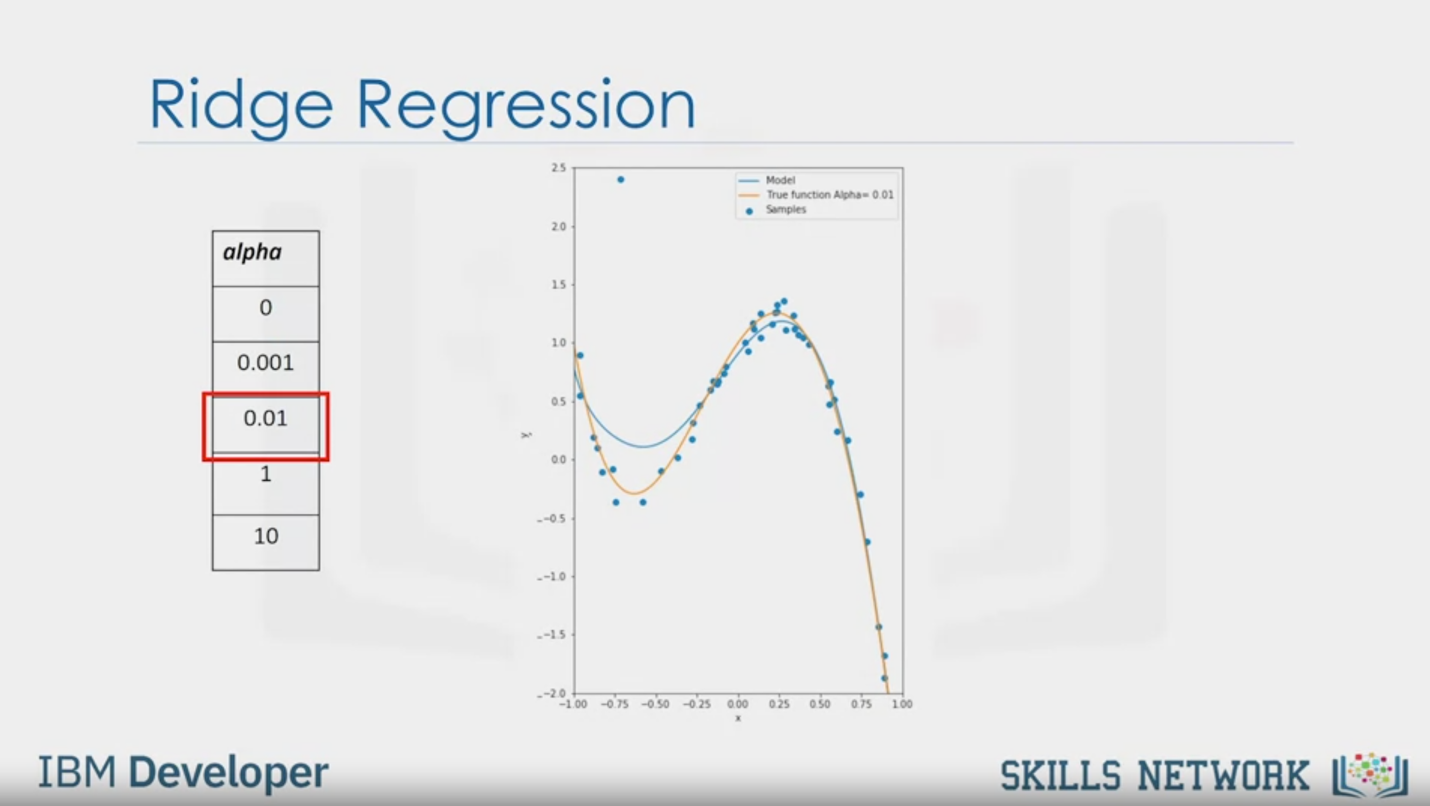
0.01

1

10

IBM Developer

SKILLS NETWORK



Ridge Regression

alpha

0

0.001

0.01

1

10

IBM Developer

SKILLS NETWORK



Ridge Regression

alpha

0

0.001

0.01

1

10

IBM Developer

SKILLS NETWORK



Ridge Regression

alpha

0

0.001

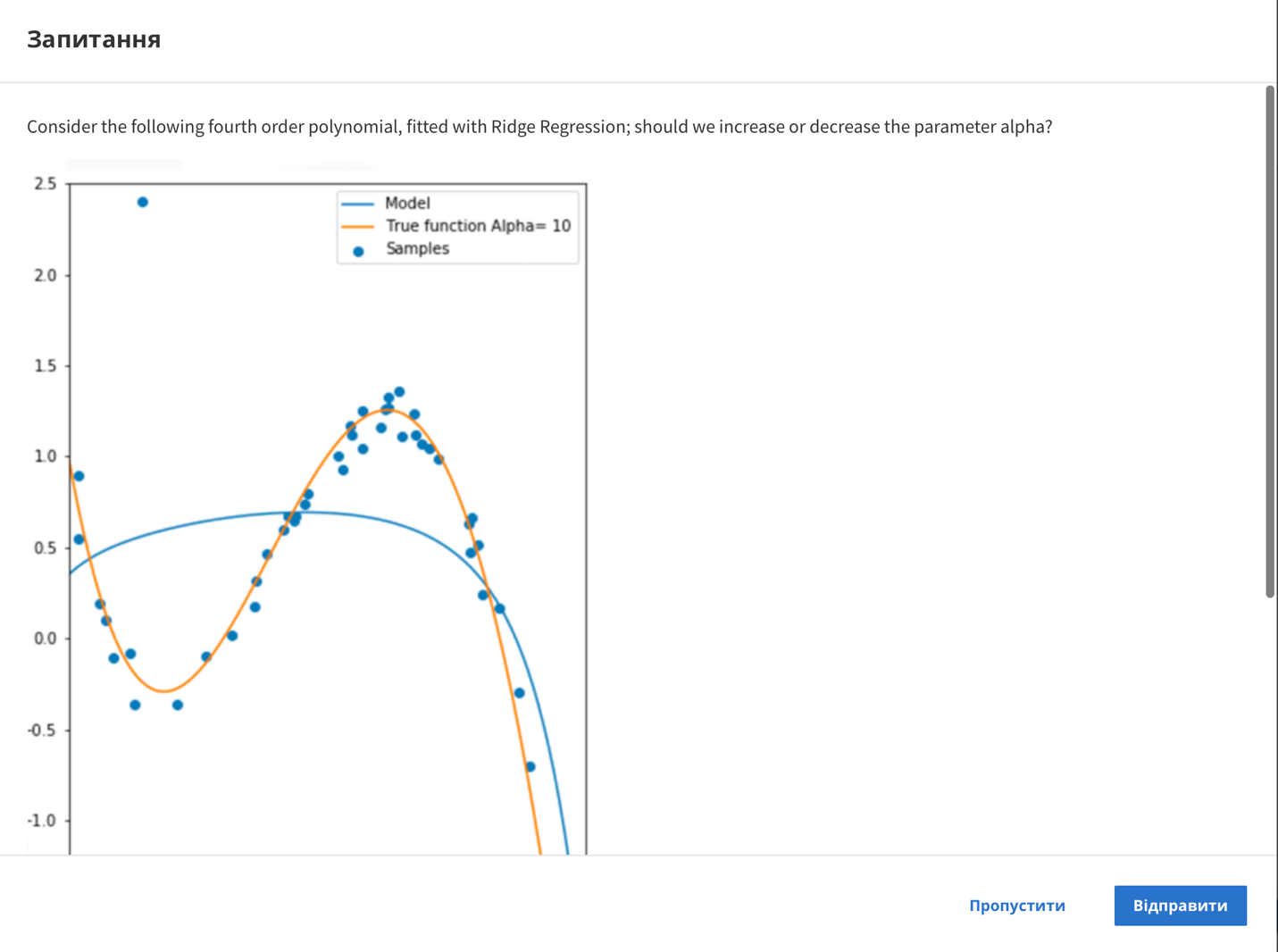
0.01

1

10

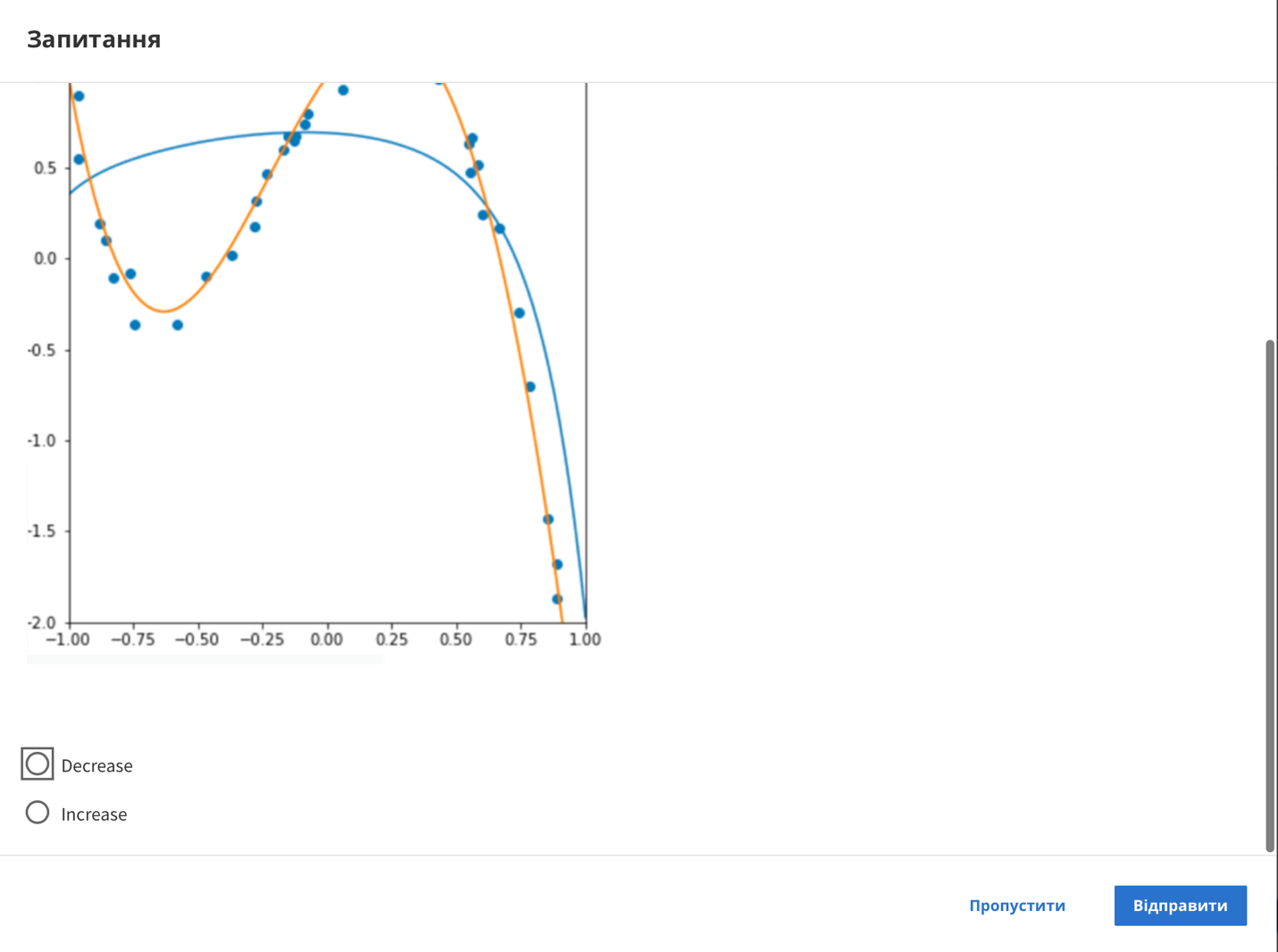
IBM Developer

SKILLS NETWORK



Question

Consider the following fourth order polynomial, fitted with Ridge Regression; should we increase or decrease the parameter alpha?

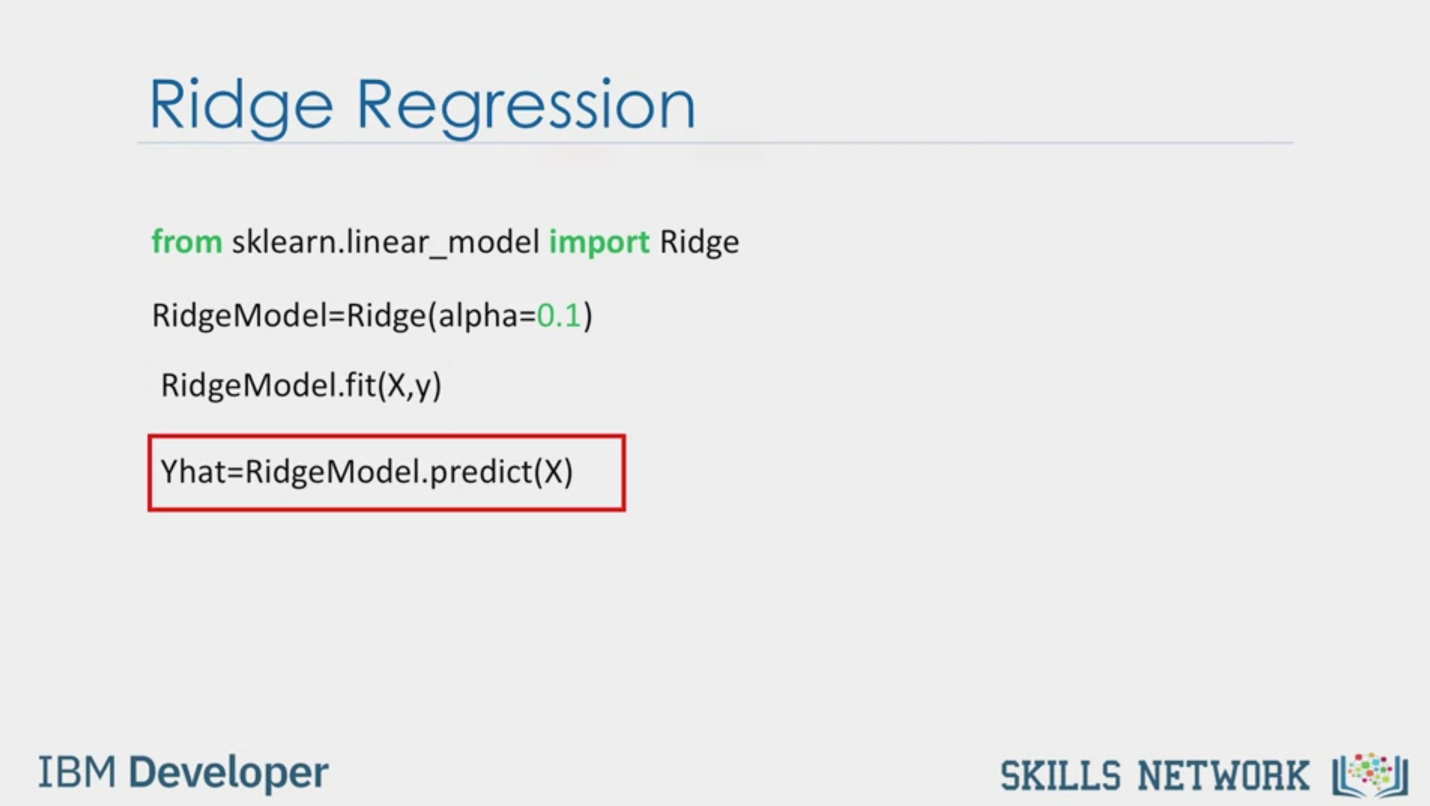


Decrease

Increase

Skip

Send



Ridge Regression

from sklearn.linear\_model import Ridge

RidgeModel=Ridge(alpha=0.1)

RidgeModel.fit(X,y)

Yhat=RidgeModel.predict(X)

IBM Developer

SKILLS NETWORK



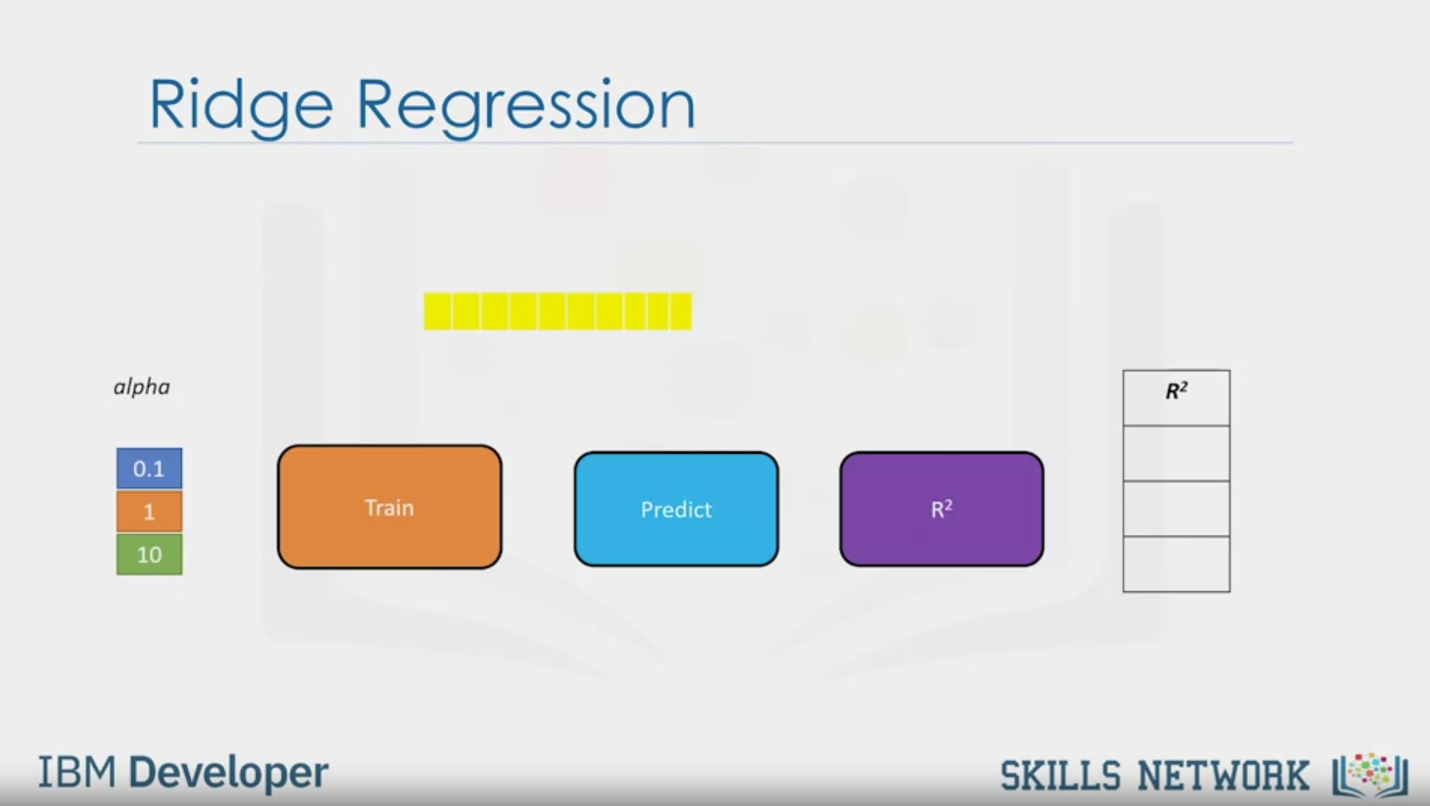
Question

Create a Ridge object using the constructor; set the parameter alpha to 1

Enter your answer here

Skip

Send



Ridge Regression

alpha

0.1

1

10

Train

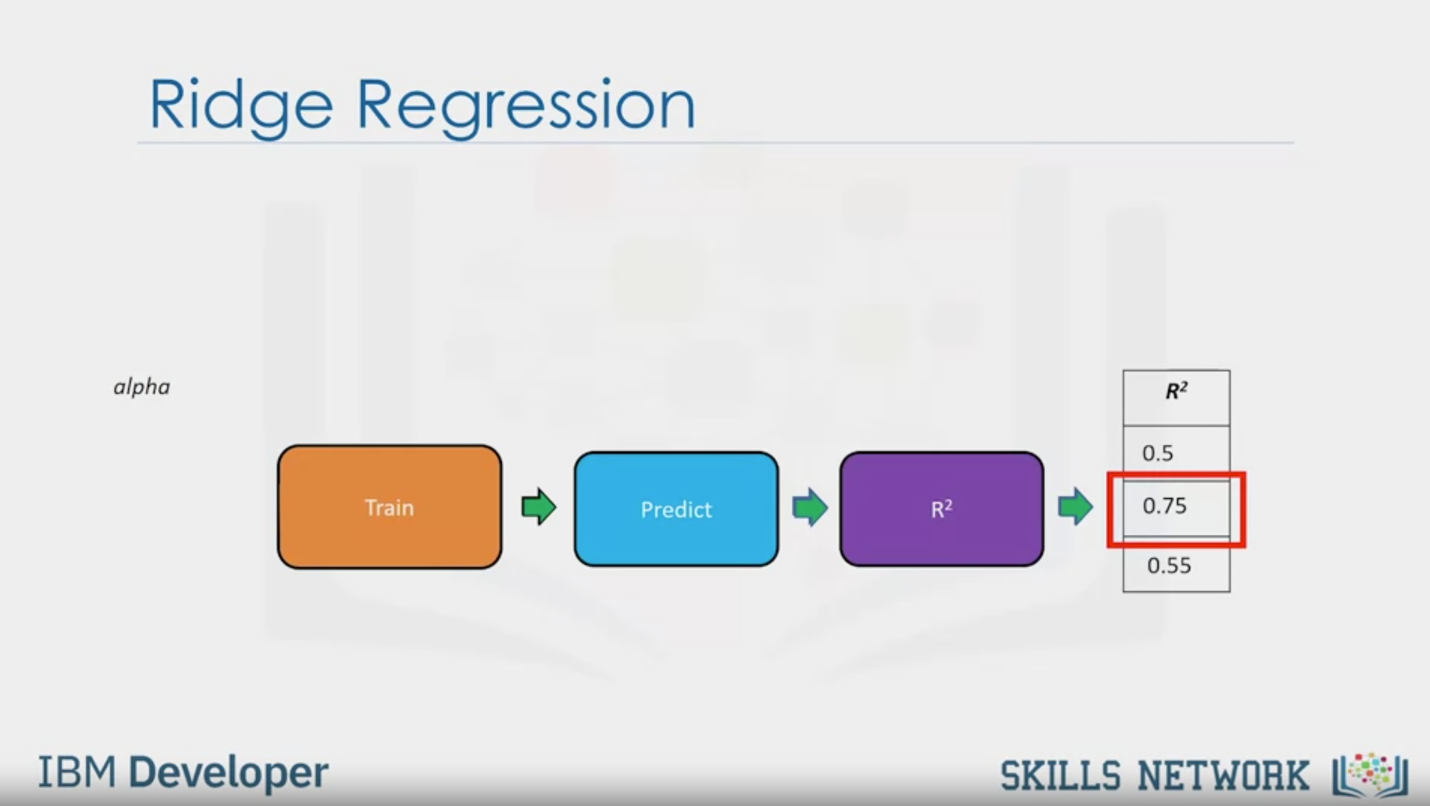
Predict

R2

R2

IBM Developer

SKILLS NETWORK



Ridge Regression

Alpha

Train

Predict

R2

R2

0.5

0.75

0.55

IBM Developer

SKILLS NETWORK



Ridge Regression

IBM Developer

SKILLS NETWORK



SKILLS NETWORK

IBM Developer

In this video, we'll discuss ridge regression.

Ridge regression prevents overfitting.

In this video, we will focus on polynomial regression for visualization,

but overfitting is also a big problem when

you have multiple independent variables, or features.

Consider the following fourth order polynomial in orange.

The blue points are generated from this function.

We can use a tenth order polynomial to fit the data.

The estimated function in blue does a good job at approximating the true function.

In many cases real data has outliers.

For example, this point shown here does not appear to come from the function in orange.

If we use a tenth order polynomial function to fit the data,

the estimated function in blue is incorrect,

and is not a good estimate of the actual function in orange.

If we examine the expression for the estimated function,

we see the estimated polynomial coefficients have a very large magnitude.

This is especially evident for the higher order polynomials.

Ridge regression controls the magnitude of

these polynomial coefficients by introducing the parameter alpha.

Alpha is a parameter we select before fitting or training the model.

Each row in the following table represents an increasing value of alpha.

Let's see how different values of alpha change the model.

This table represents the polynomial coefficients for different values of alpha.

The column corresponds to the different polynomial coefficients,

and the rows correspond to the different values of alpha.

As alpha increases, the parameters get smaller.

This is most evident for the higher order polynomial features.

But Alpha must be selected carefully.

If alpha is too large,

the coefficients will approach zero and underfit the data.

If alpha is zero,

the overfitting is evident.

For alpha equal to 0.001,

the overfitting begins to subside.

For Alpha equal to 0.01,

the estimated function tracks the actual function.

When alpha equals one,

we see the first signs of underfitting.

The estimated function does not have enough flexibility.

At alpha equals to 10,

we see extreme underfitting.

It does not even track the two points.

In order to select alpha,

we use cross validation.

To make a prediction using ridge regression,

import ridge from sklearn.linear\_models.

Create a ridge object using the constructor.

The parameter alpha is one of the arguments of the constructor.

We train the model using the fit method.

To make a prediction, we use the predict method.

In order to determine the parameter alpha,

we use some data for training.

We use a second set called validation data.

This is similar to test data,

but it is used to select parameters like alpha.

We start with a small value of alpha.

We train the model, make a prediction using the validation data,

then calculate the R-squared and store the values.

Repeat the value for a larger value of alpha.

We train the model again,

make a prediction using the validation data,

then calculate the R-squared and store the values of R-squared.

We repeat the process for a different alpha value,

training the model, and making a prediction.

We select the value of alpha that maximizes the R-squared.

Note that we can use other metrics to select the value of alpha,

like mean squared error.

The overfitting problem is even worse if we have lots of features.

The following plot shows the different values of R-squared on the vertical axis.

The horizontal axis represents different values for alpha.

We use several features from our used car data

set and a second order polynomial function.

The training data is in red and validation data is in blue.

We see as the value for alpha increases,

the value of R-squared increases and converges at approximately 0.75.

In this case, we select the maximum value of alpha because

running the experiment for higher values of alpha have little impact.

Conversely, as alpha increases,

the R-squared on the test data decreases.

This is because the term alpha prevents overfitting.

This may improve the results in the unseen data,

but the model has worse performance on the test data.

See the lab on how to generate this plot.