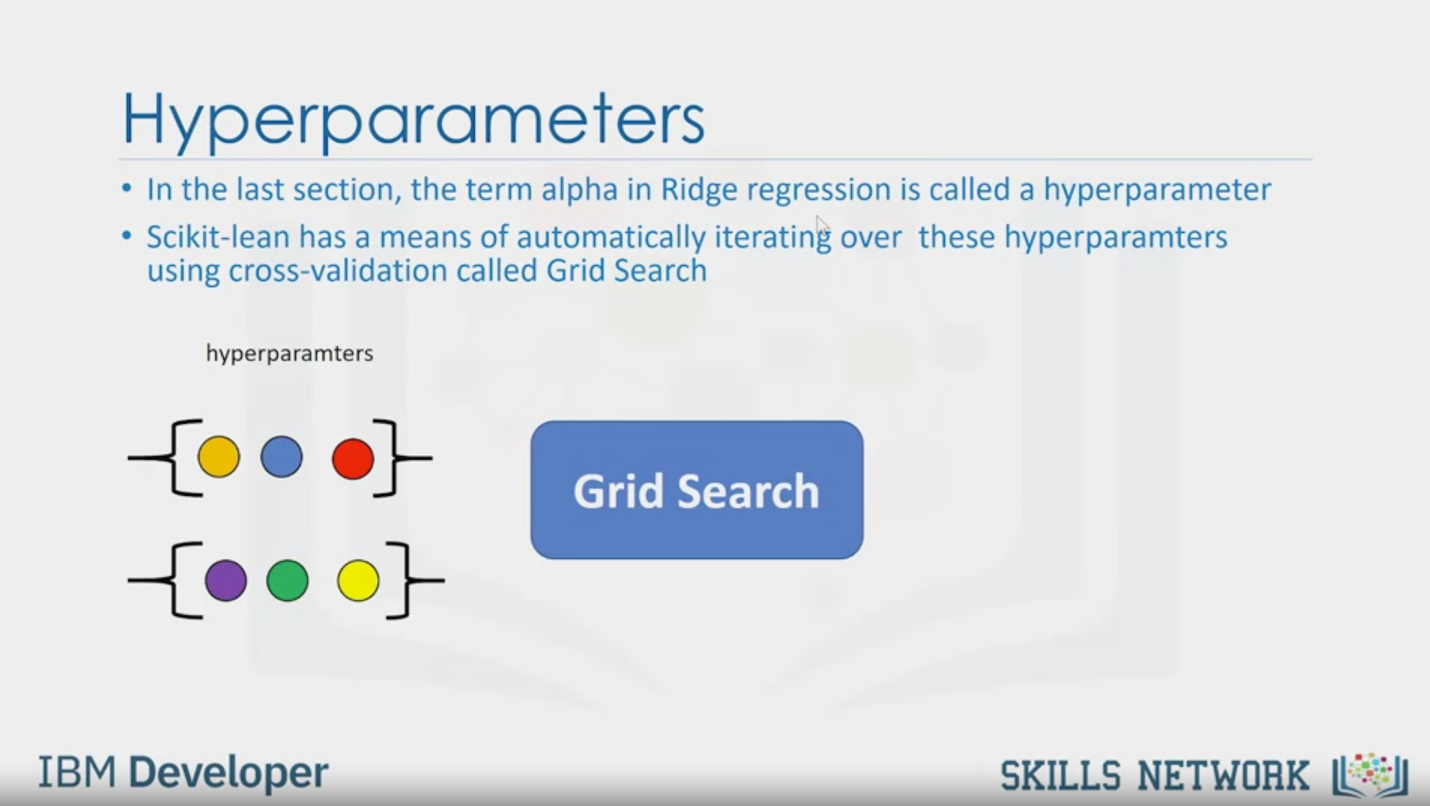


Grid Search

IBM Developer

SKILLS NETWORK



Hyperparameters

• In the last section, the term alpha in Ridge regression is called a hyperparameter

• Scikit-lean has a means of automatically iterating over these hyperparamters

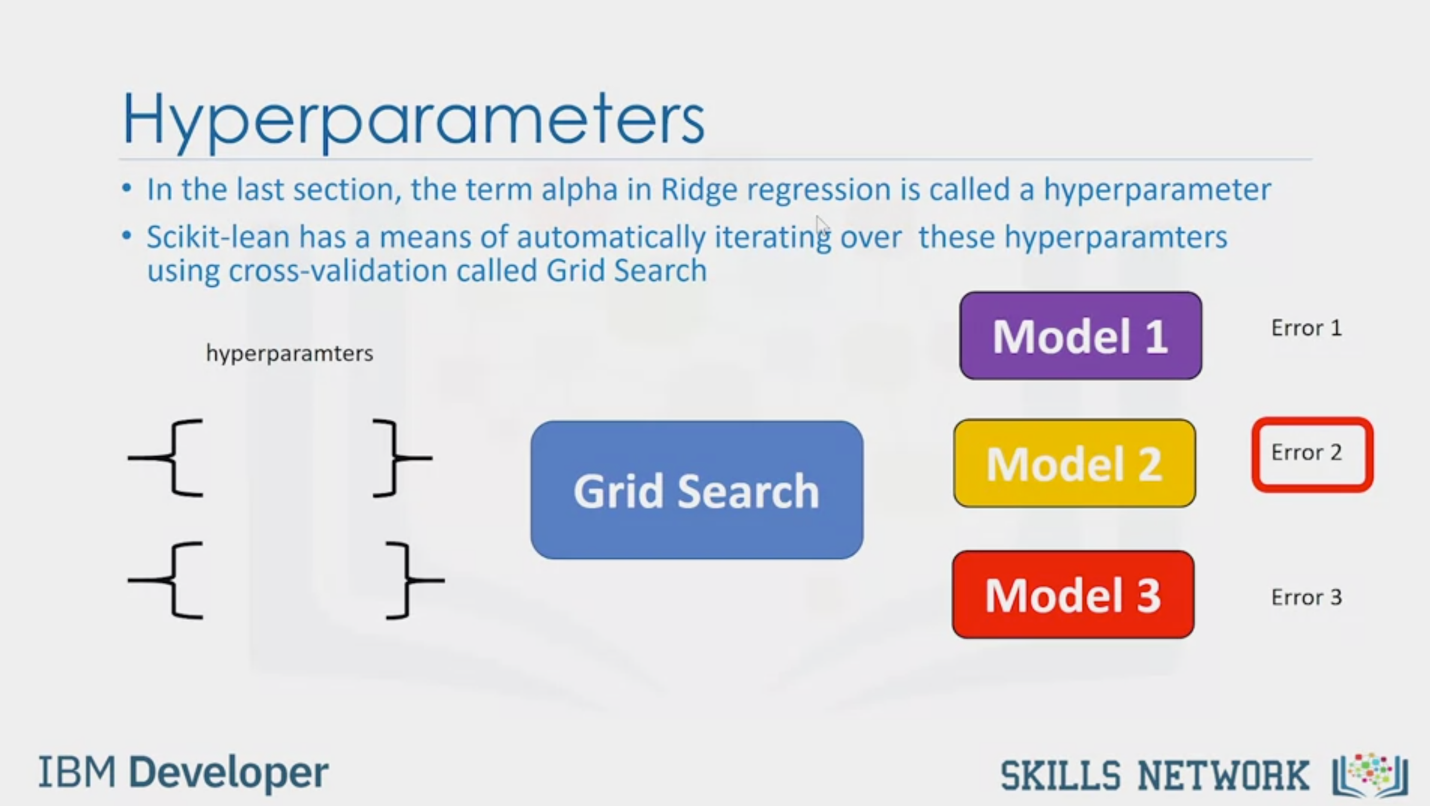
using cross-validation called Grid Search

hyperparamters

Grid Search

IBM Developer

SKILLS NETWORK



Hyperparameters

• In the last section, the term alpha in Ridge regression is called a hyperparameter

• Scikit-lean has a means of automatically iterating over these hyperparamters

using cross-validation called Grid Search

hyperparamters

Grid Search

Model 1

Error 1

Model 2

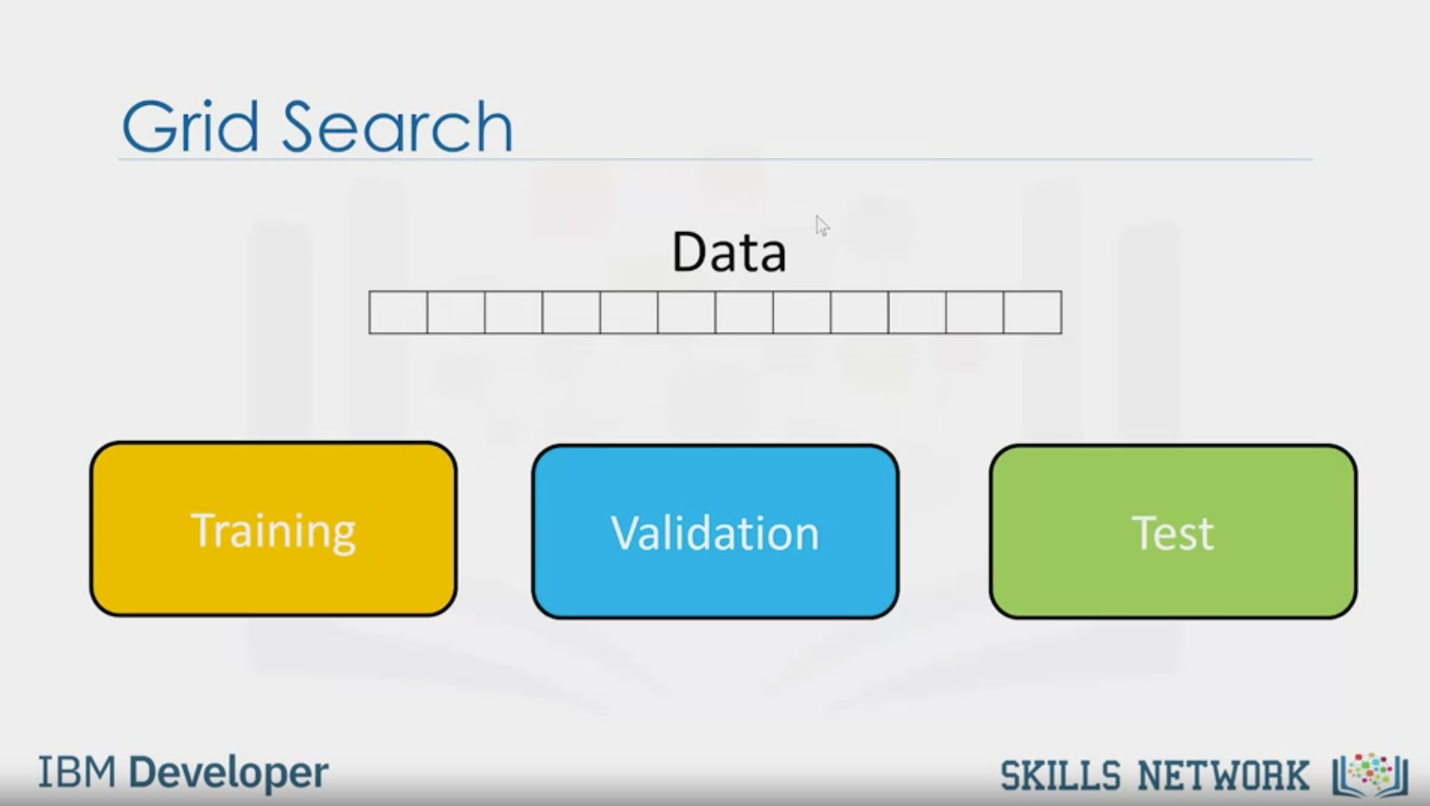
Error 2

Model 3

Error 3

IBM Developer

SKILLS NETWORK



Grid Search

Data

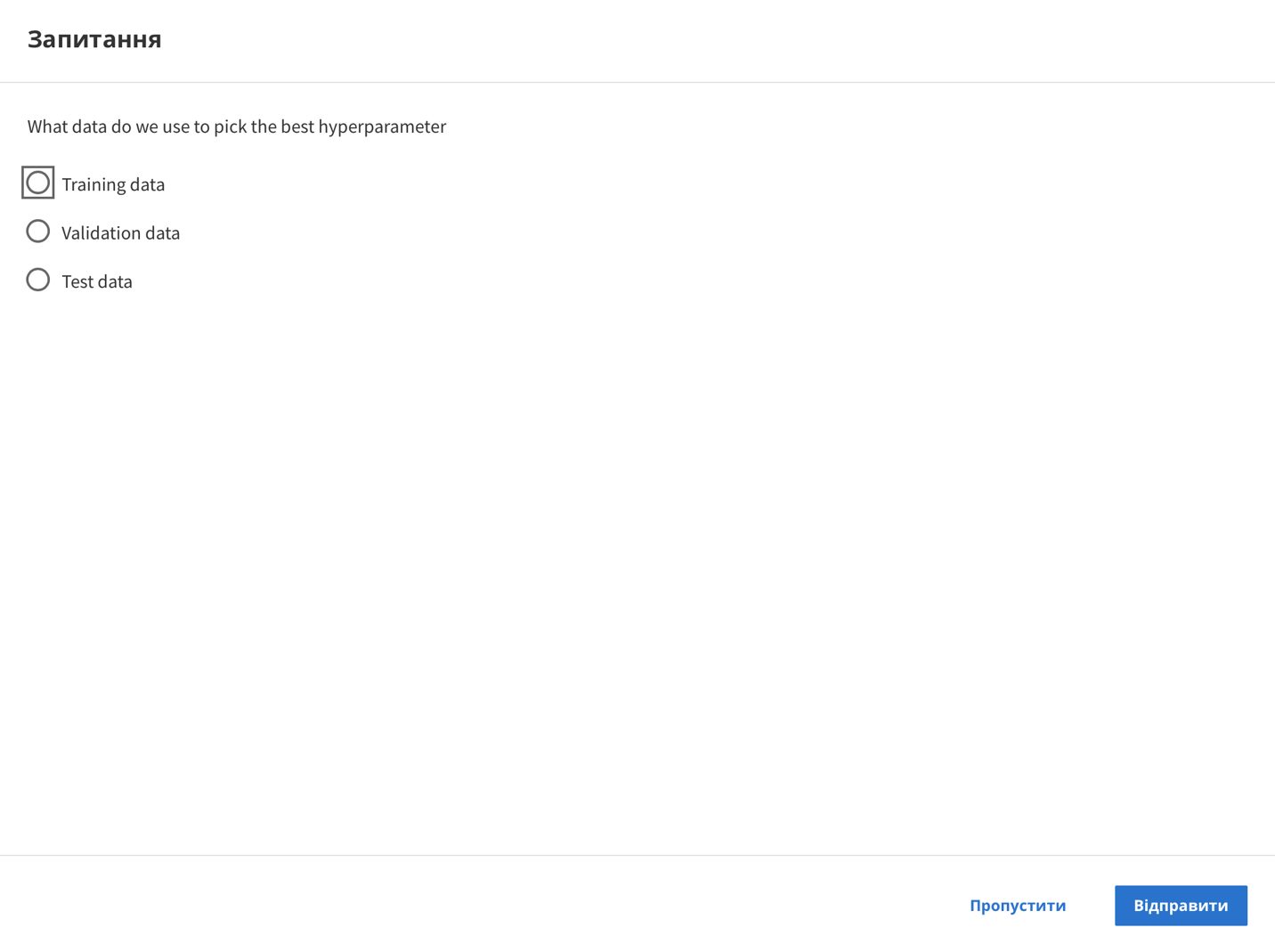
Training

Validation

Test

IBM Developer

SKILLS NETWORK



Question

What data do we use to pick the best hyperparameter

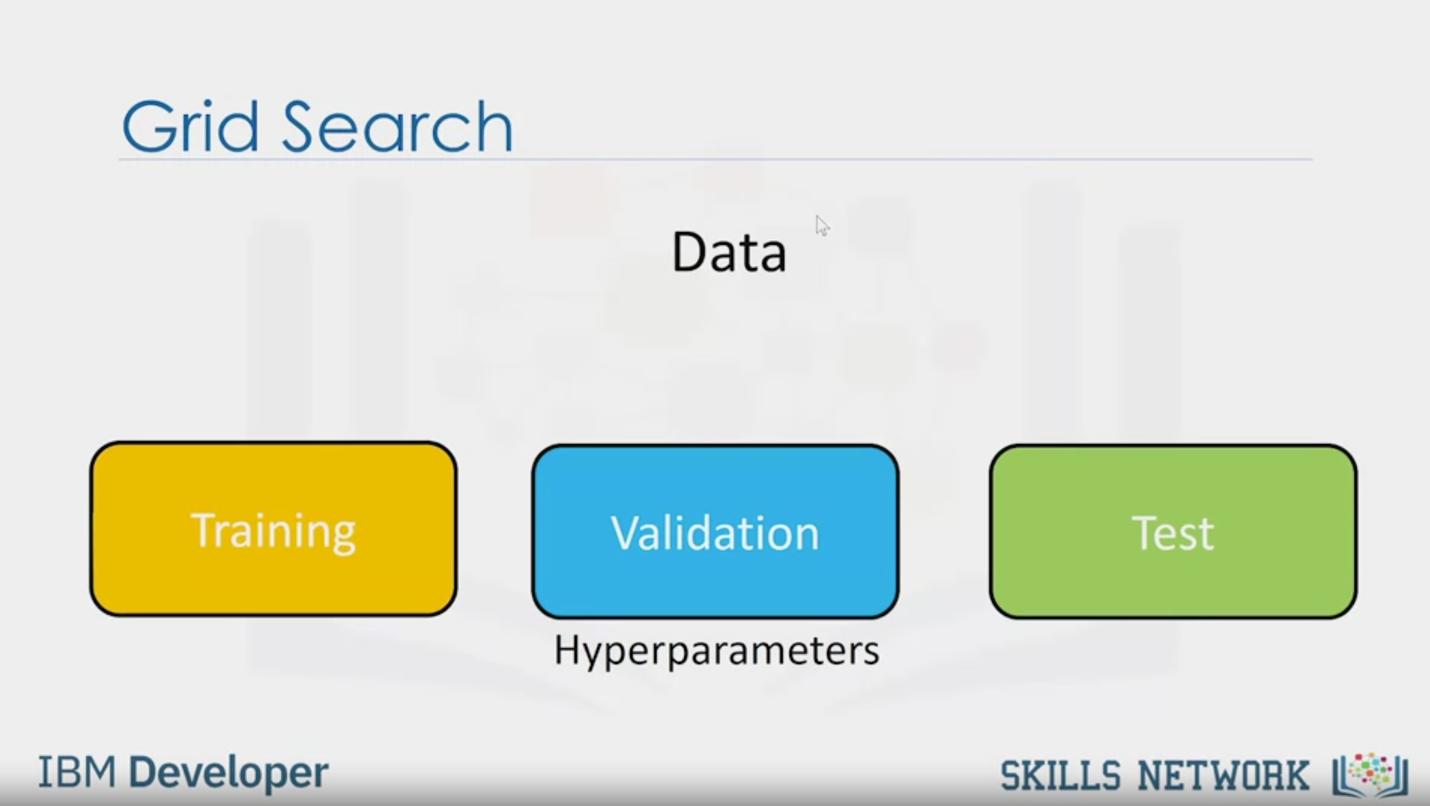
Training data

Validation data

Test data

Skip

Send



Grid Search

Data

Training

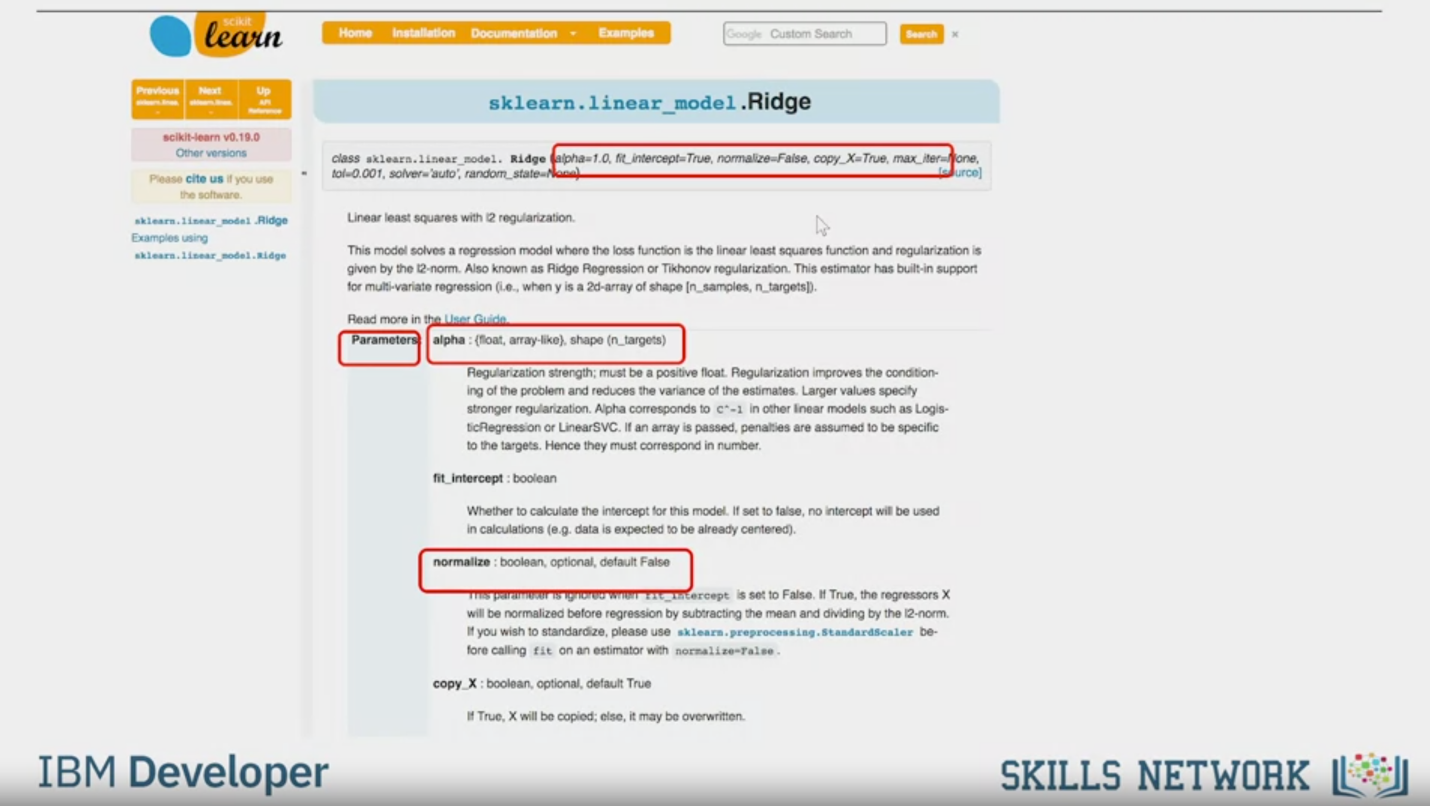
Validation

Test

Hyperparameters

IBM Developer

SKILLS NETWORK



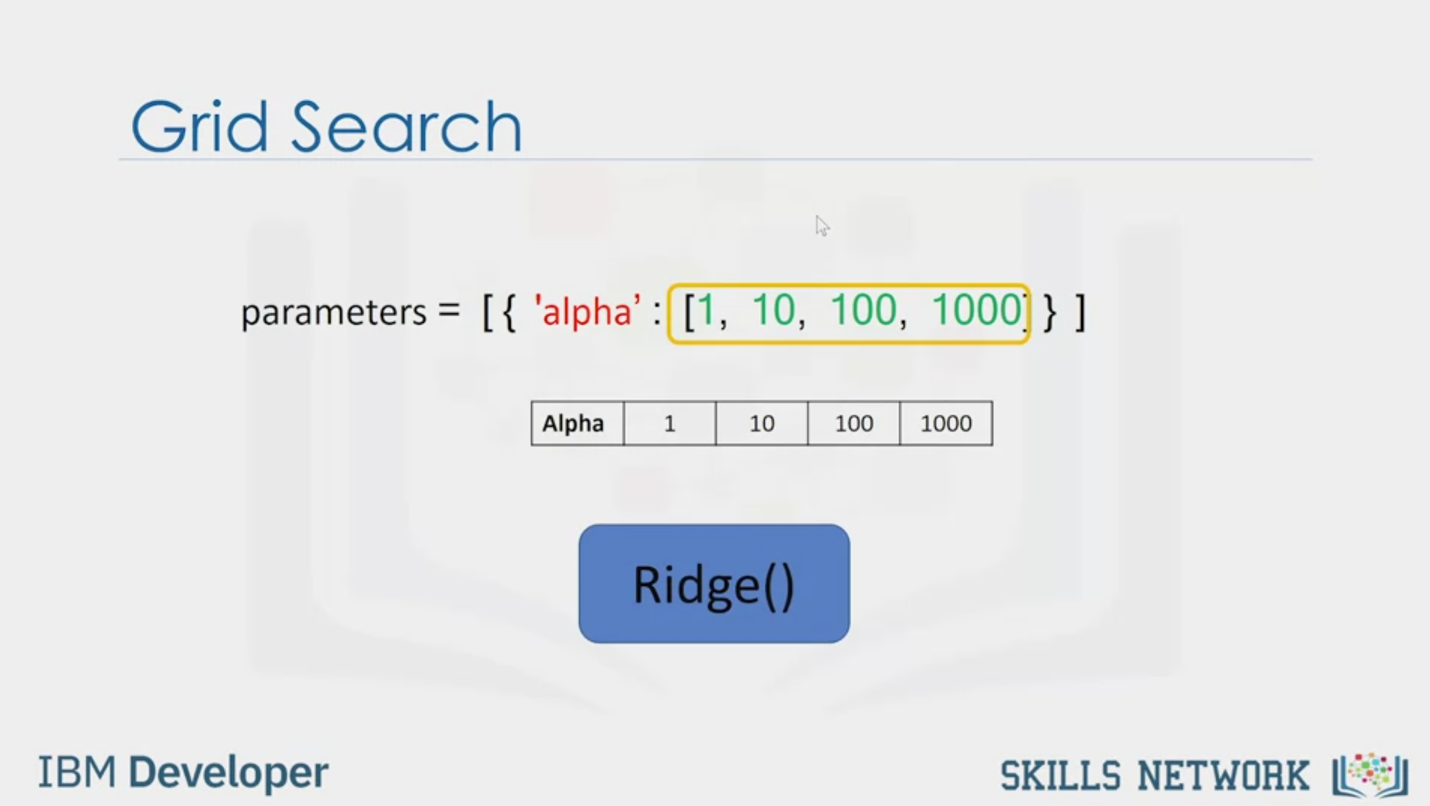
Ridge (alpha = 1.0, fit\_intercept=True, normalize=False, copy\_X=True, max\_iter=None,

tol=0.001, solver=’auto’, random\_state=None)  
Parameters: alpha: {float, arrey-like}, shape (n\_targets)

normalize: boolean, optional, default False

IBM Developer

SKILLS NETWORK



Grid Search

parameters = [ { 'alpha' : [1, 10, 100, 1000] } ]

Alpha

1

10

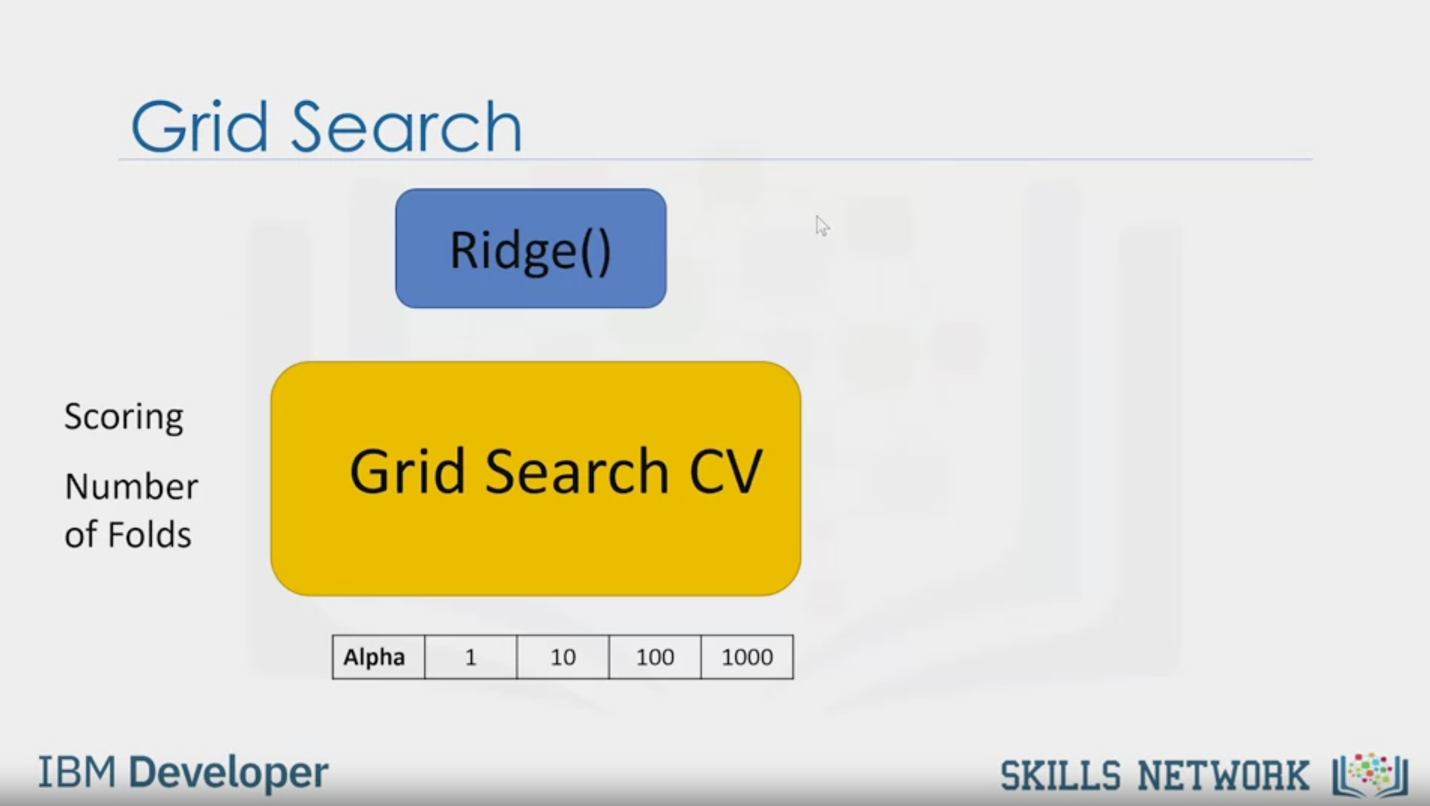
100

1000

Ridge()

IBM Developer

SKILLS NETWORK



Grid Search

Ridge()

Scoring

Number

of Folds

Grid Search CV

Alpha

1

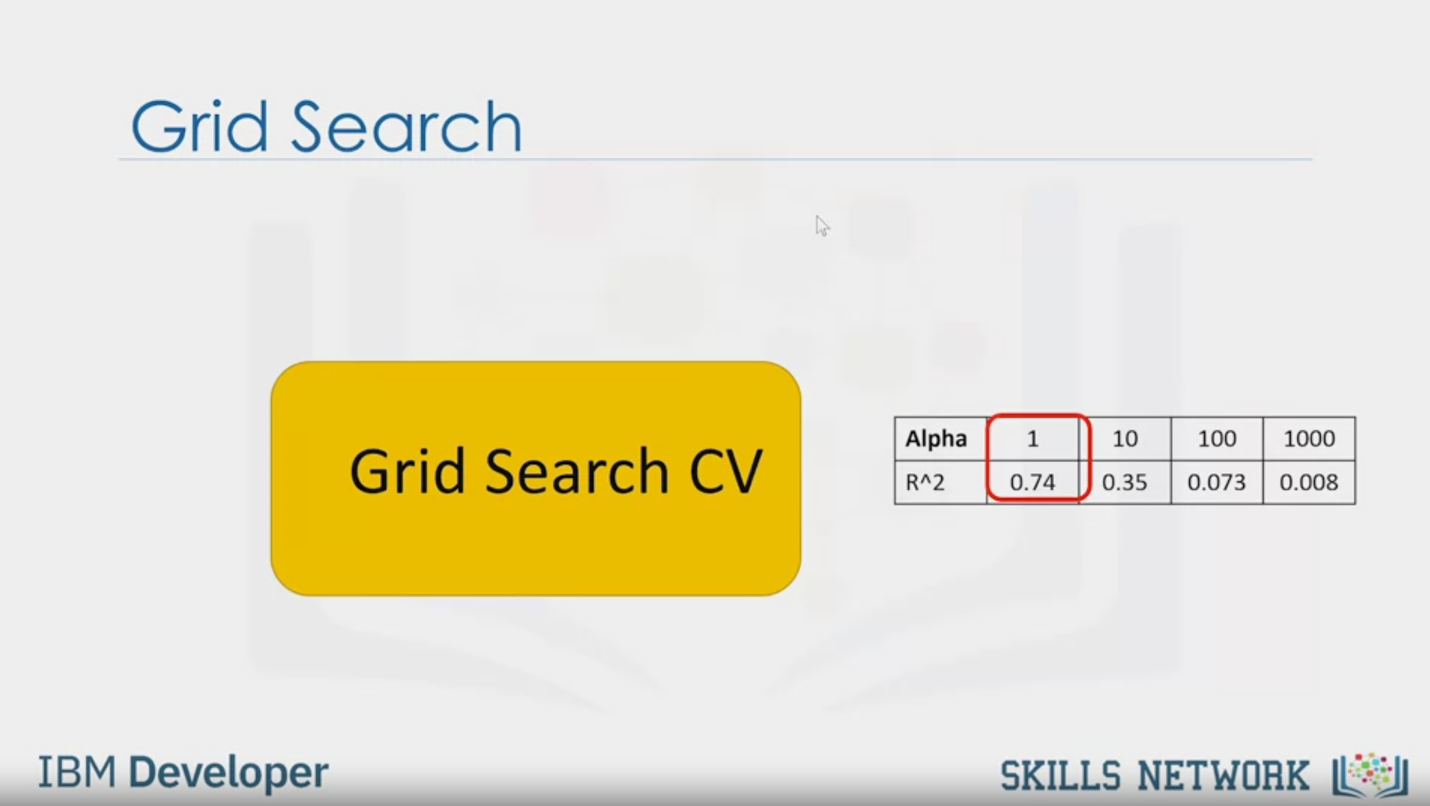
10

100

1000

IBM Developer

SKILLS NETWORK



Grid Search

Grid Search CV

Alpha

1

10

100

1000

R^2

0.74

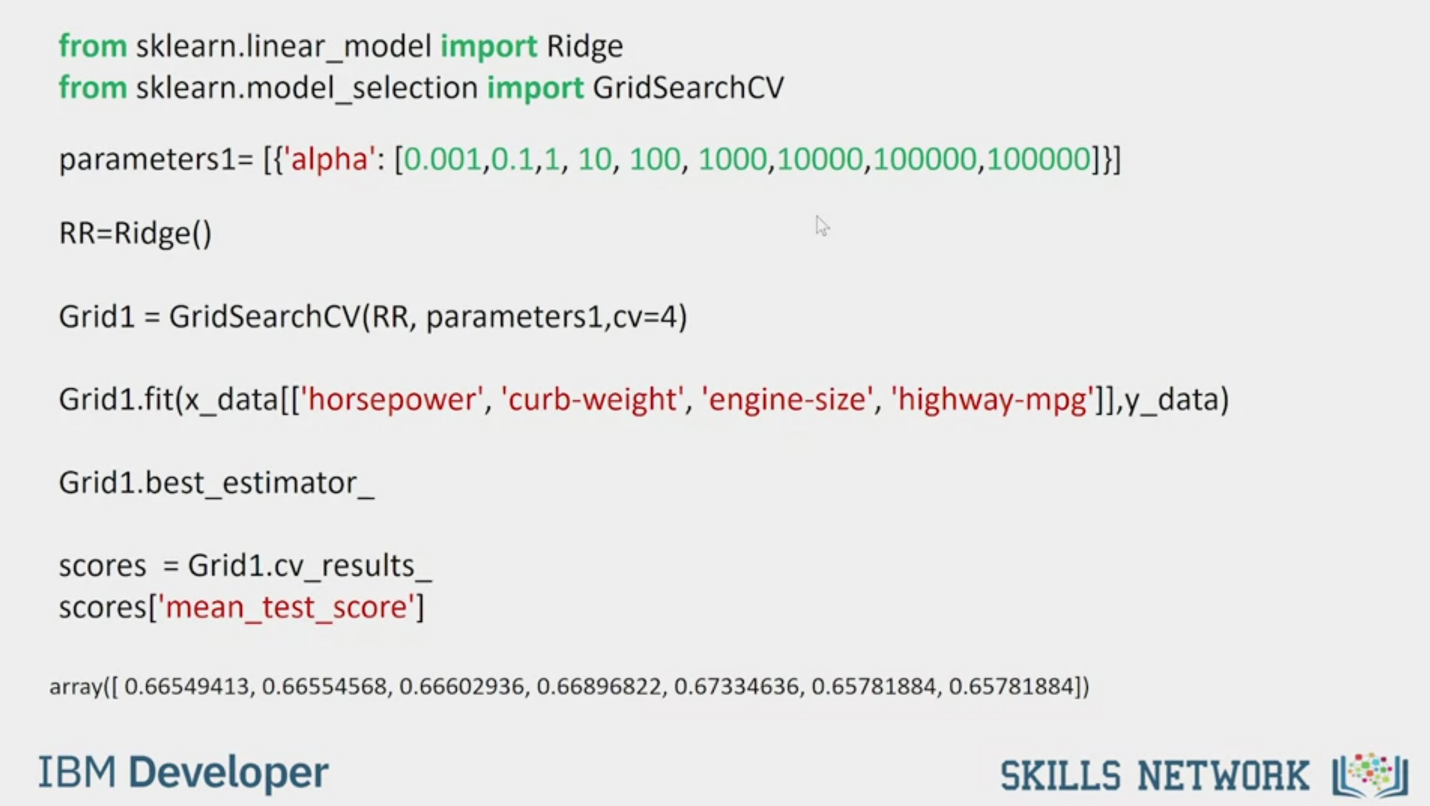
0.35

0.073

0.008

IBM Developer

SKILLS NETWORK



from sklearn.linear\_model import Ridge

from sklearn.model\_selection import GridSearchCV

parameters1= [{'alpha': [0.001,0.1,1, 10, 100, 1000,10000,100000,100000]}]

RR=Ridge()

Grid1 = GridSearchCV(RR, parameters1, cv=4)

Grid1.fit(x\_data[['horsepower', 'curb-weight', 'engine-size', 'highway-mpg']],y\_data)

Grid1.best\_estimator\_

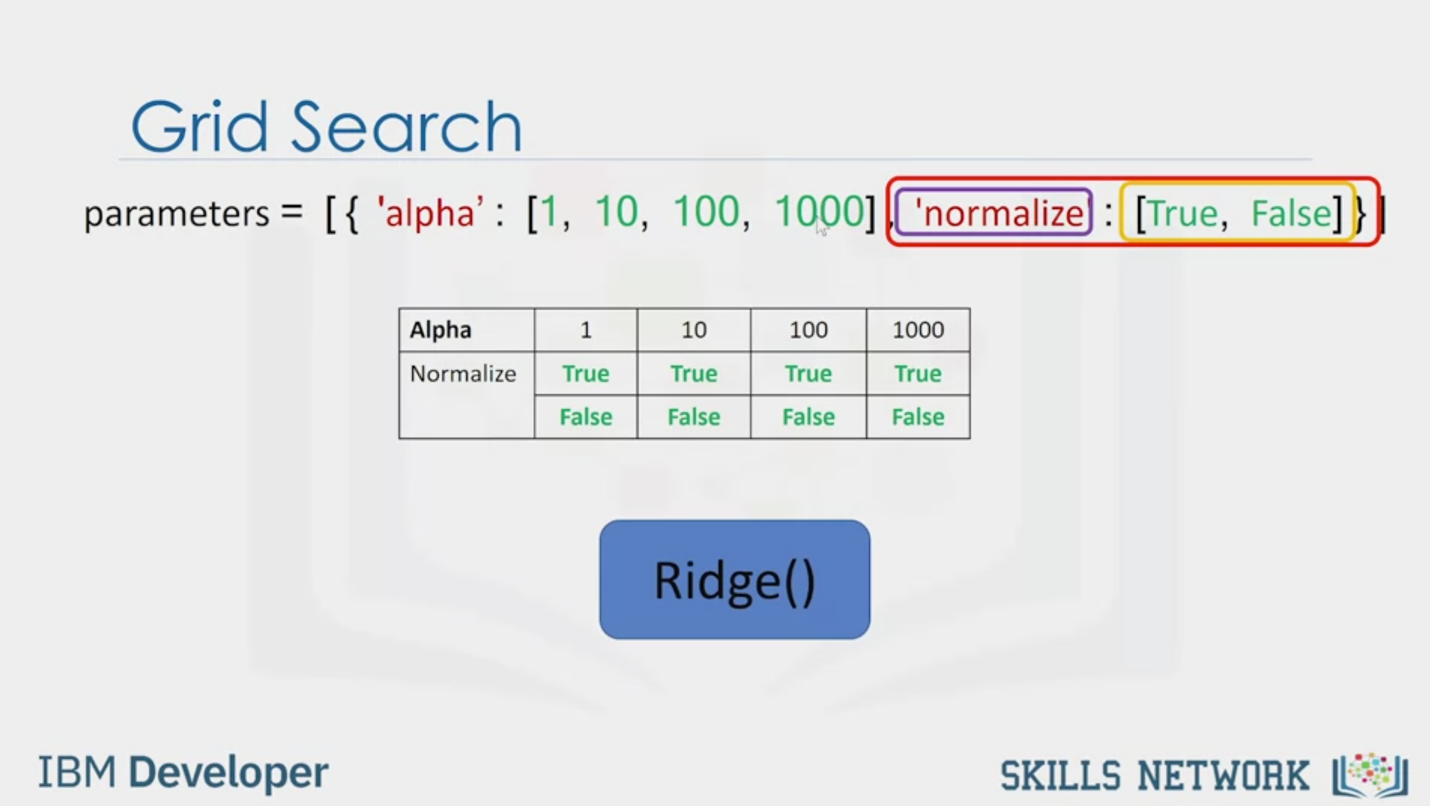
scores = Grid1.cv\_results\_

scores['mean\_test\_ score'l

array([ 0.66549413, 0.66554568, 0.66602936, 0.66896822, 0.67334636, 0.65781884, 0.657818841])

IBM Developer

SKILLS NETWORK



Grid Search

parameters = [{ 'alpha': [1, 10, 100, 1000], ‘normalize’ : [True, False]}]

Alpha

Normalize

1

True

False

10

True

False

100

True

False

1000

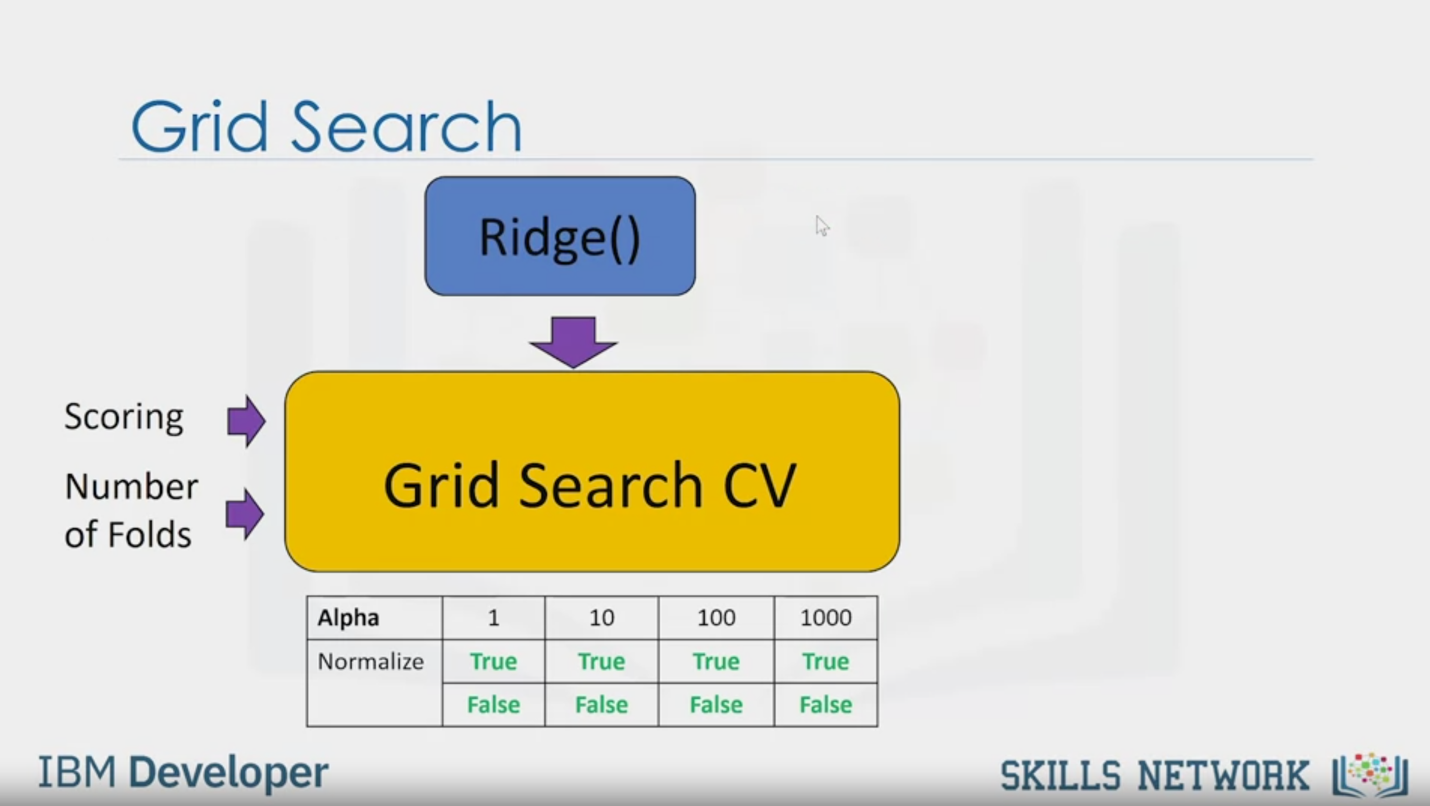
True

False

Ridge()

IBM Developer

SKILLS NETWORK



Grid Search

Ridge()

Scoring

Number

of Folds

Grid Search CV

Alpha

Normalize

1

True

False

10

True

False

100

True

False

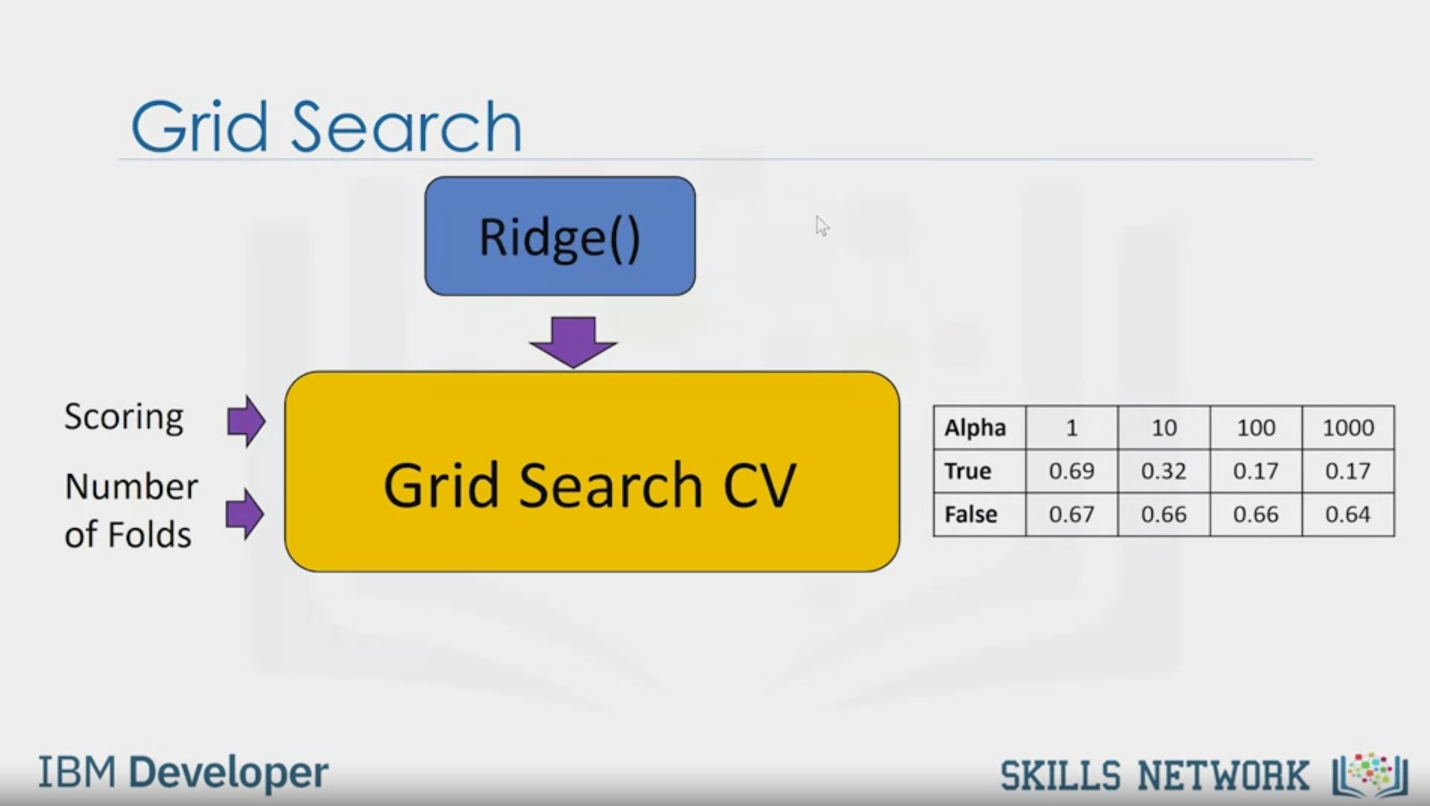
1000

True

False

IBM Developer

SKILLS NETWORK



Grid Search

Ridge()

Scoring

Number

of Folds

Grid Search CV

Alpha

True

False

1

0.69

0.67

10

0.32

0.66

100

0.17

0.66

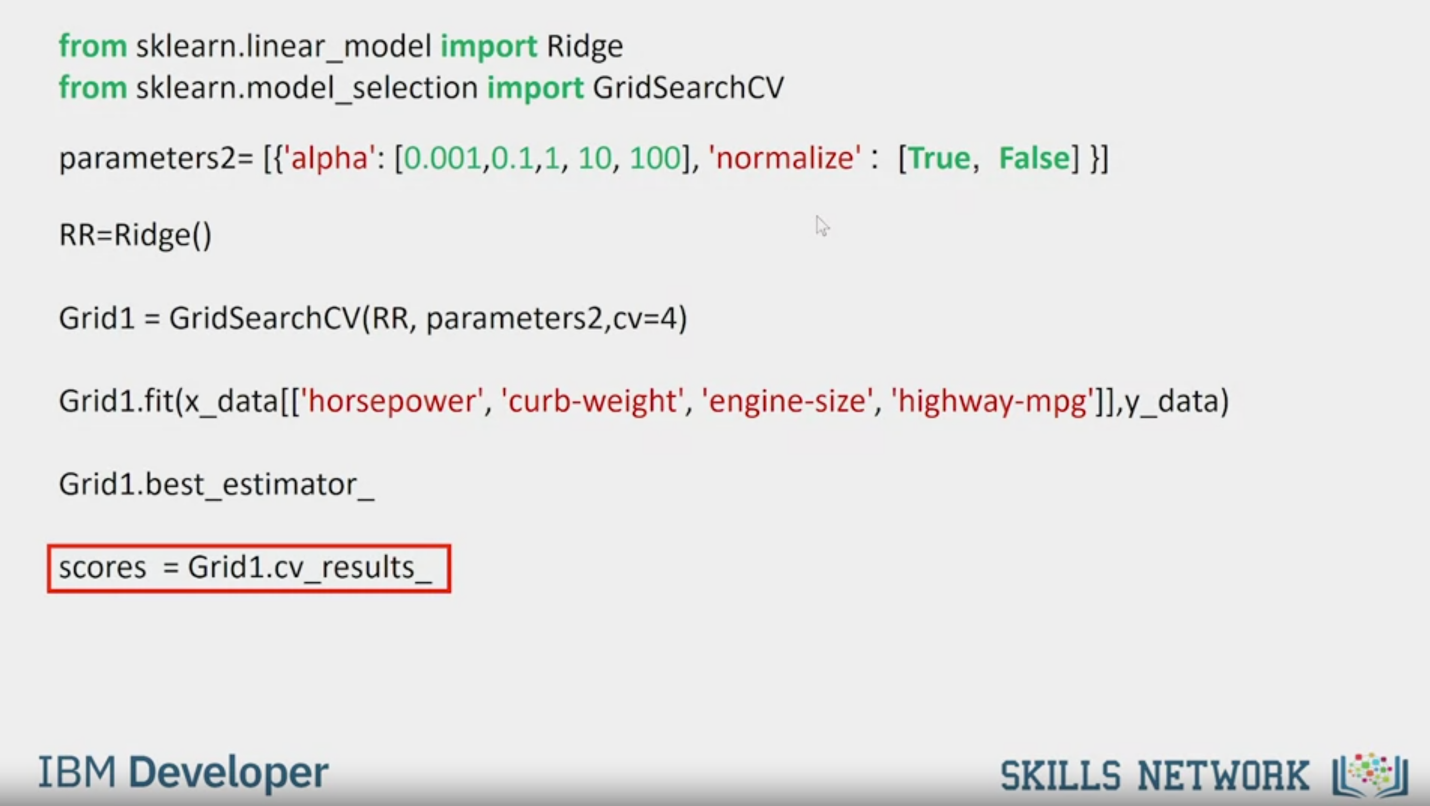
1000

0.17

0.64

IBM Developer

SKILLS NETWORK



from sklearn.linear\_model import Ridge

from sklearn.model\_selection import GridSearchCV

parameters2= [{'alpha': [0.001,0.1,1, 10, 100], 'normalize' : [True, False] }]

RR=Ridge()

Grid1 = GridSearchCV(RR, parameters2, cv=4)

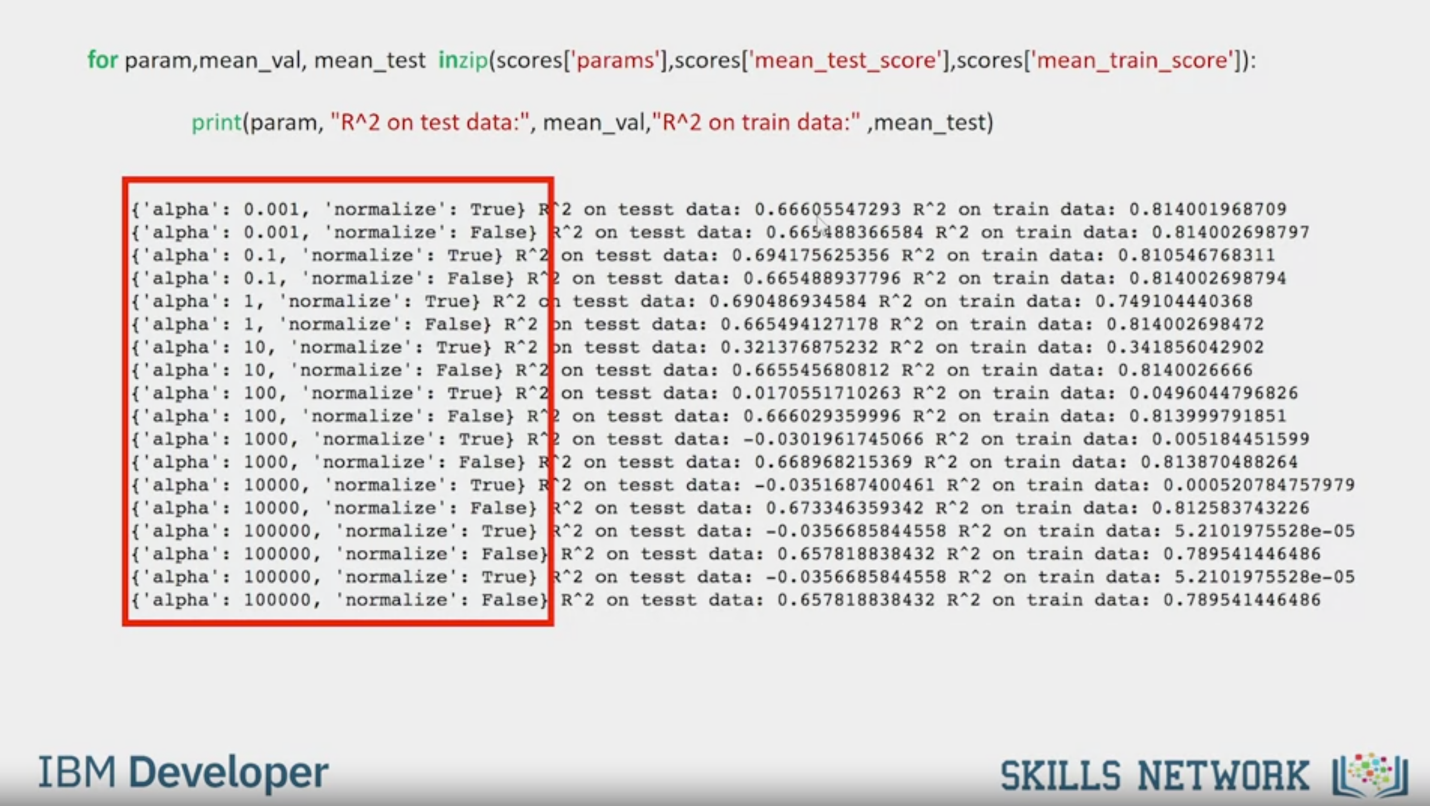
Grid1.fit(x\_data[['horsepower', 'curb-weight', 'engine-size', 'highway-mpg']],y\_data)

Grid1.best\_estimator\_

scores = Grid1.cv\_results\_

IBM Developer

SKILLS NETWORK



for param,mean\_val, mean\_test inzip(scores['params'],scores['mean\_ test\_score'],scores['mean\_train\_score']):

print(param, "R^2 on test data:", mean\_val, "R^2 on train data:" mean\_test)

('alpha': 0.001,

'normalize': True} R^2 on tesst data: 0.66605547293 R^2 on train data: 0.814001968709

('alpha': 0.001,

'normalize': False} R^2 on test data: 0.665488366584 R^2 on train data: 0.814002698797

"alpha: 0.1,

'normalize': True) R^2 on tesst data: 0.694175625356 R^2 on train data: 0.810546768311

('alpha':

0.1,

'normalize': False) R^2 on tesst data:

0.665488937796 R^2 on train data:

0.814002698794

('alpha':

'normalize': True} R^2 on tesst data: 0.690486934584 R^2 on train data: 0.749104440368

l'alpha'

1

'normalize': False) R^2 on tesst data: 0.665494127178 R^2 on train data: 0.814002698472

('alpha'

10,

"normalize': True) R^2

on tesst data: 0.321376875232 R^2

on train data: 0.341856042902

('alpha': 10,

"normalize': False} R^2 on tesst data: 0.665545680812 R^2 on train data: 0.8140026666

('alpha': 100,

'normalize': Truel R^2 on tesst data:

0.0170551710263 R^2 on train data: 0.0496044796826

('alpha': 100,

'normalize': False) R^2 on tesst data: 0.666029359996 R^2 on traín data: 0.813999791851

('alpha': 1000,

'normalize'

: Truel

R^2

on

tesst data:

.0301961745066 R^2

on train data:

0.005184451599

('alpha': 1000.

'normalize': Palse} R^2 on tesst data: 0.668968215369 R^2 on train data: 0.813870488264

('alpha': 10000,

'normalize': True! R^2 on tesst data:

-0.0351687400461 R^2 on train data: 0.000520784757979

('alpha': 10000,

"normalize

": False} R^2

An teget rata:

0.673346359342 R^2

on train data: 0.812583743226

('alpha': 100000,

'normalize': True} R^2 on tesst data:

-0.0356685844558 R^2 on train

data: 5.2101975528e-05

('alpha': 100000,

"normalize': Palse) R^2 on tesst data: 0.657818838432 R^2 on train data: 0.789541446486

('alpha': 100000,

'normalize': True} R^2 on tesst data:

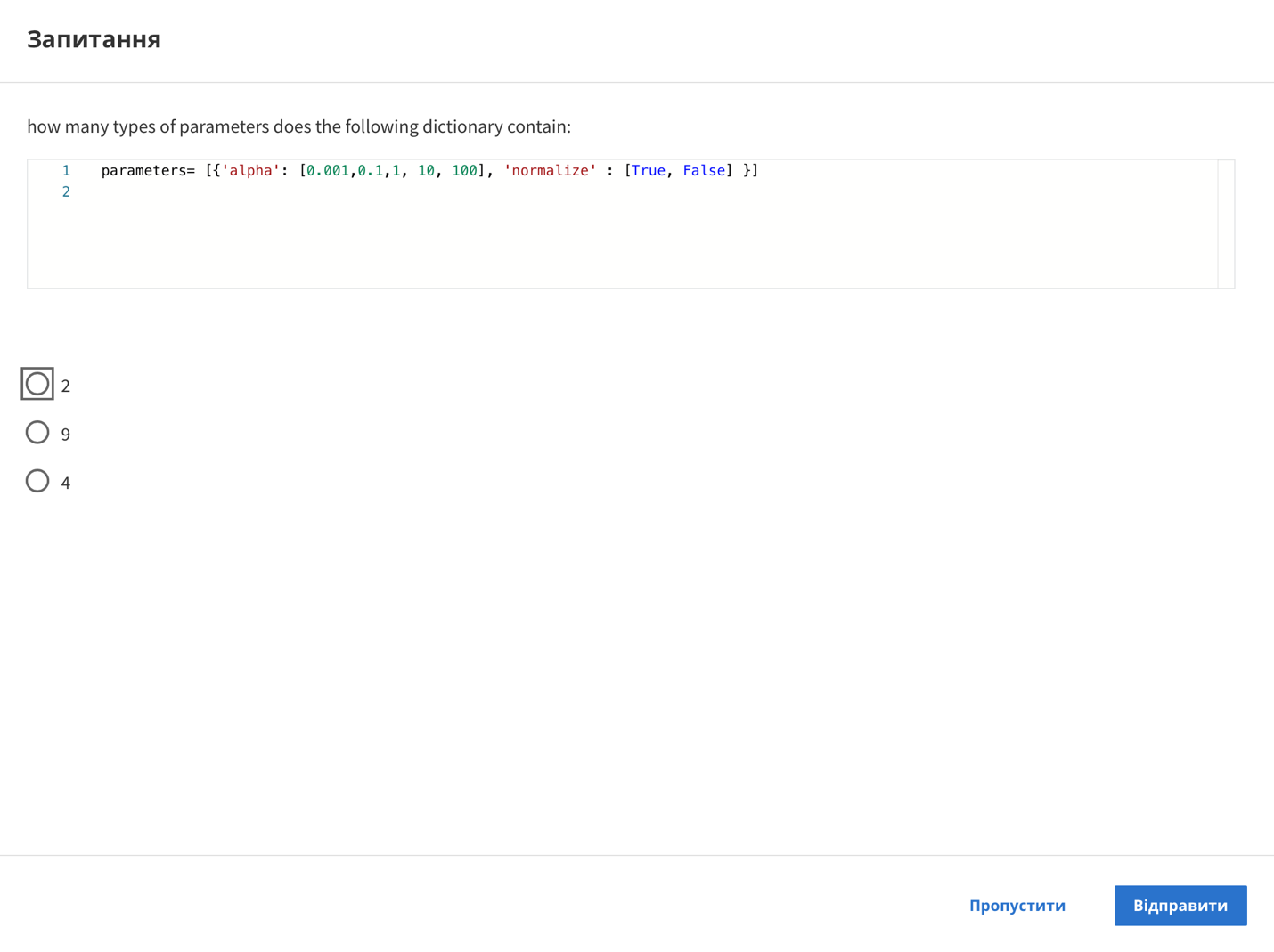
-0.0356685844558 R^2 on train data: 5.2101975528e-05

('alpha': 100000,

\*normalize': Palse! R^2 on tesst data: 0.657818838432 R^2 on train data: 0.789541446486

IBM Developer

SKILLS NETWORK



Question

how many types of parameters does the following dictionary contain:

1 parameters= [{'alpha': [0.001,0.1,1, 10, 100], 'normalize': [True, False] }]

2

2

9

4

Skip

Send



SKILLS NETWORK

IBM Developer

Grid Search allows us to scan through multiple free parameters with few lines of code.

Parameters like the alpha term discussed in

the previous video are not part of the fitting or training process.

These values are called hyperparameters.

Scikit-learn has a means of automatically

iterating over these hyperparameters using cross-validation.

This method is called Grid Search.

Grid Search takes the model or objects you would like to

train and different values of the hyperparameters.

It then calculates the mean square error or R-squared for various hyperparameter values,

allowing you to choose the best values.

Let the small circles represent different hyperparameters.

We start off with one value for hyperparameters and train the model.

We use different hyperparameters to train the model.

We continue the process until we have exhausted the different free parameter values.

Each model produces an error.

We select the hyperparameter that minimizes the error.

To select the hyperparameter,

we split our dataset into three parts,

the training set, validation set, and test set.

We train the model for different hyperparameters.

We use the R-squared or mean square error for each model.

We select the hyperparameter that minimizes the mean squared

error or maximizes the R-squared on the validation set.

We finally test our model performance using the test data.

This is the scikit-learn web page,

where the object constructor parameters are given.

It should be noted that the attributes of an object are also called parameters.

We will not make the distinction even though some of

the options are not hyperparameters per se.

In this module, we will focus on

the hyperparameter alpha and the normalization parameter.

The value of your Grid Search is a Python list that contains a Python dictionary.

The key is the name of the free parameter.

The value of the dictionary is the different values of the free parameter.

This can be viewed as a table with various free parameter values.

We also have the object or model.

The Grid Search takes on the scoring method.

In this case, R-squared the number of folds,

the model or object,

and the free parameter values.

Some of the outputs include the different scores for different free parameter values.

In this case, the R-squared along with a free parameter values that have the best score.

First, we import the libraries we need,

including GridSearchCV, the dictionary of parameter values.

We create a ridge regression object or model.

We then create a GridSearchCV object.

The inputs are the ridge regression object,

the parameter values, and the number of folds.

We will use R-squared.

This is the default scoring method. We fit the object.

We can find the best values for the free parameters using the attribute best estimator.

We can also get information like the mean score on

the validation data using the attribute CV result.

What are the advantages of Grid Search is how quickly we can test multiple parameters.

For example, ridge regression has the option to normalize the data.

To see how to standardize, see module four.

The term alpha is the first element in the dictionary.

The second element is the normalized option.

The key is the name of the parameter.

The value is the different options in

this case because we can either normalize the data or not.

The values are True or False respectively.

The dictionary is a table or grid that contains two different values.

As before, we need the ridge regression object or model.

The procedure is similar except that we have

a table or grid of different parameter values.

The output is the score for all the different combinations of parameter values.

The code is also similar.

The dictionary contains the different free parameter values.

We can find the best value for the free parameters.

The resulting scores of the different free parameters are stored in

this dictionary, Grid1.cv\_results \_.

We can print out the score for the different free parameter values.

The parameter values are stored as shown here.

See the course labs for more examples.