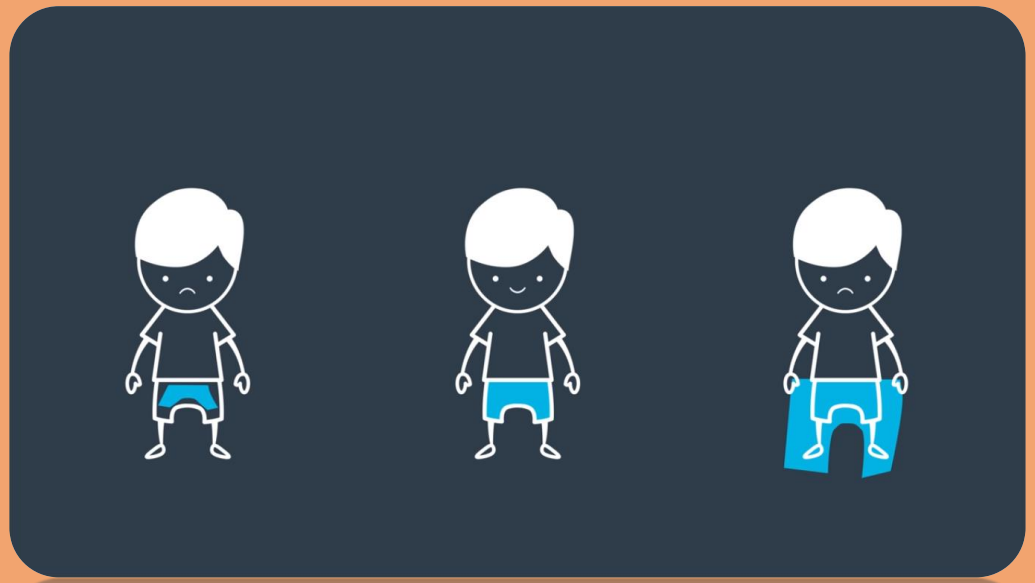
دوره‌ی آموزشی «علم داده»
Data Science Course



جلسه بیست و نهم

بیش برازش و روش‌های شناخت و رفع آن

Overfitting and tackling it

مدرس: محمد فزونی
عضو هیئت علمی دانشگاه گنبدکاووس


What's overfitting and how do we deal with it?


UNDERFITTING
OVERFITTING

UNDERFITTING

The model has not captured the underlying logic of the data

مدل ما نتونسته روابط منطقی موجود در داده‌ها رو خوب بفهمه

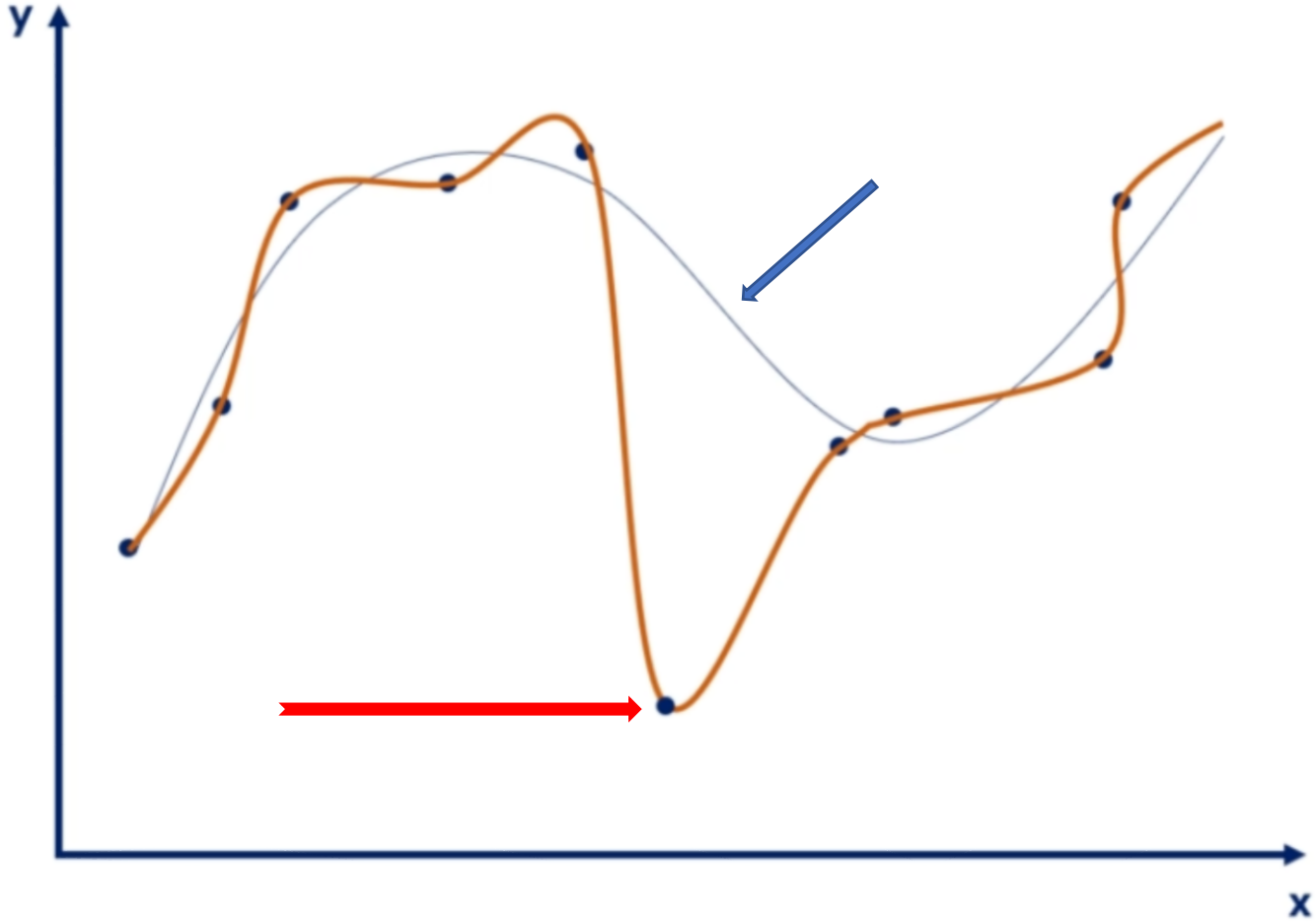
OVERFITTING

Our training has focused on the particular training set so much, it has "missed the point"

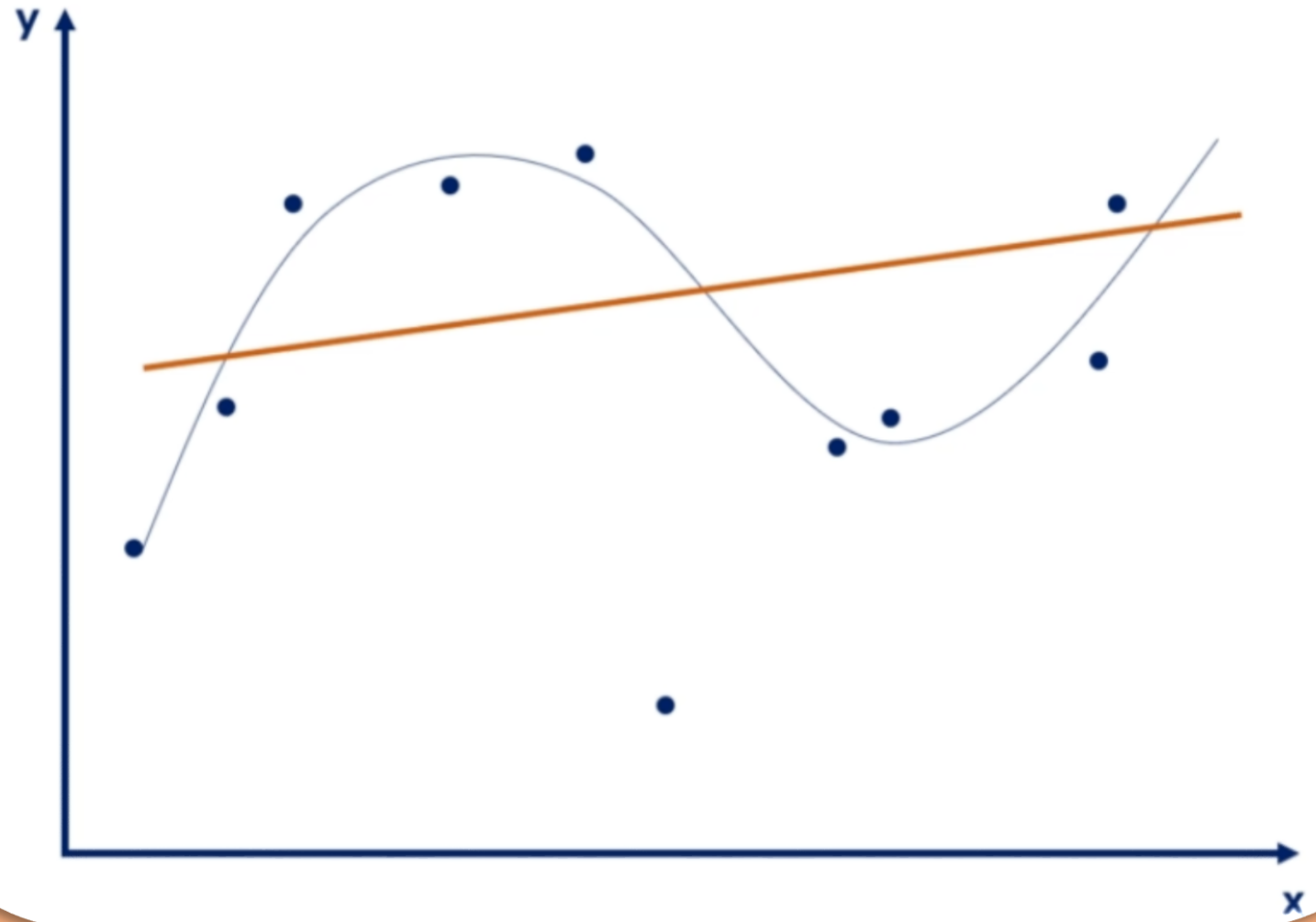
مدل بیش از حد، به روابط در دیتاست تمرینی دقت می‌کنه

High loss and low accuracy are indicators of a bad model

Underfitting and overfitting



Underfitting and overfitting



یک مثال ملموس از بیش‌برازش
در دنیای نرخ تبدیل ارز


TRAINING
Low cost
High accuracy
99.99%

OVERFITTING

The training has focused on the particular training set so much, it has "missed the point"

چه کسی مسئول است؟
الگوریتم یا کدنویس؟

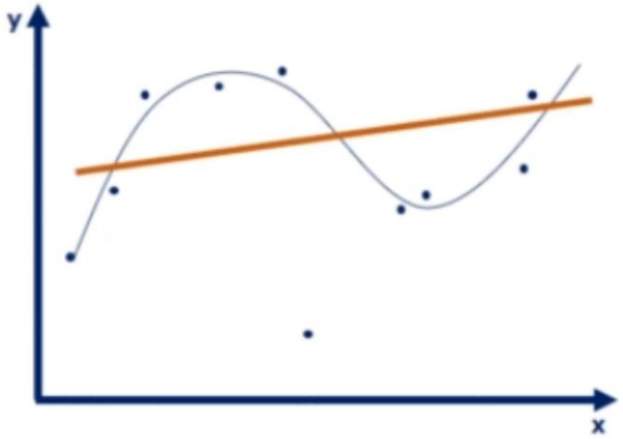
FIRST RULE OF PROGRAMMING

THE COMPUTER IS NEVER WRONG.
IT IS US, WHO MADE A MISTAKE

Underfitting and overfitting

یک مدل خوب، همیشه بین یک مدل بیش‌برازنده و یک مدل کم‌برازنده است

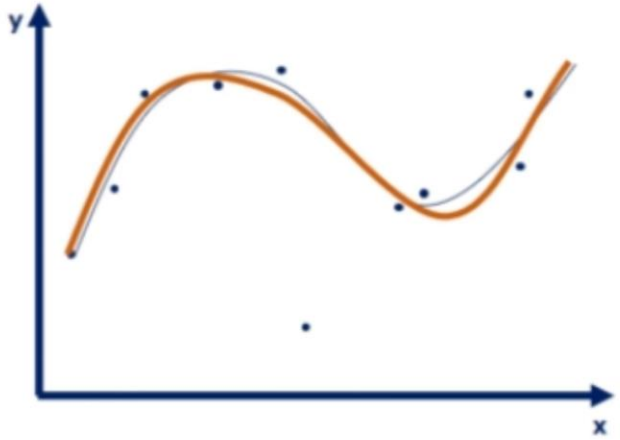
An **underfitted** model



Doesn't capture any logic

- High loss
- Low accuracy

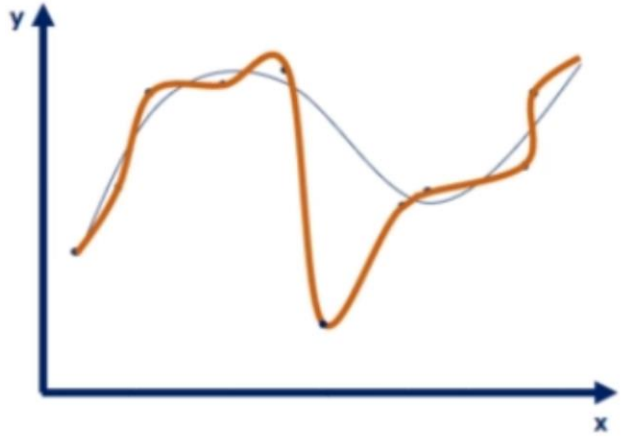
A **good** model



Captures the underlying logic of the dataset

- Low loss
- High accuracy

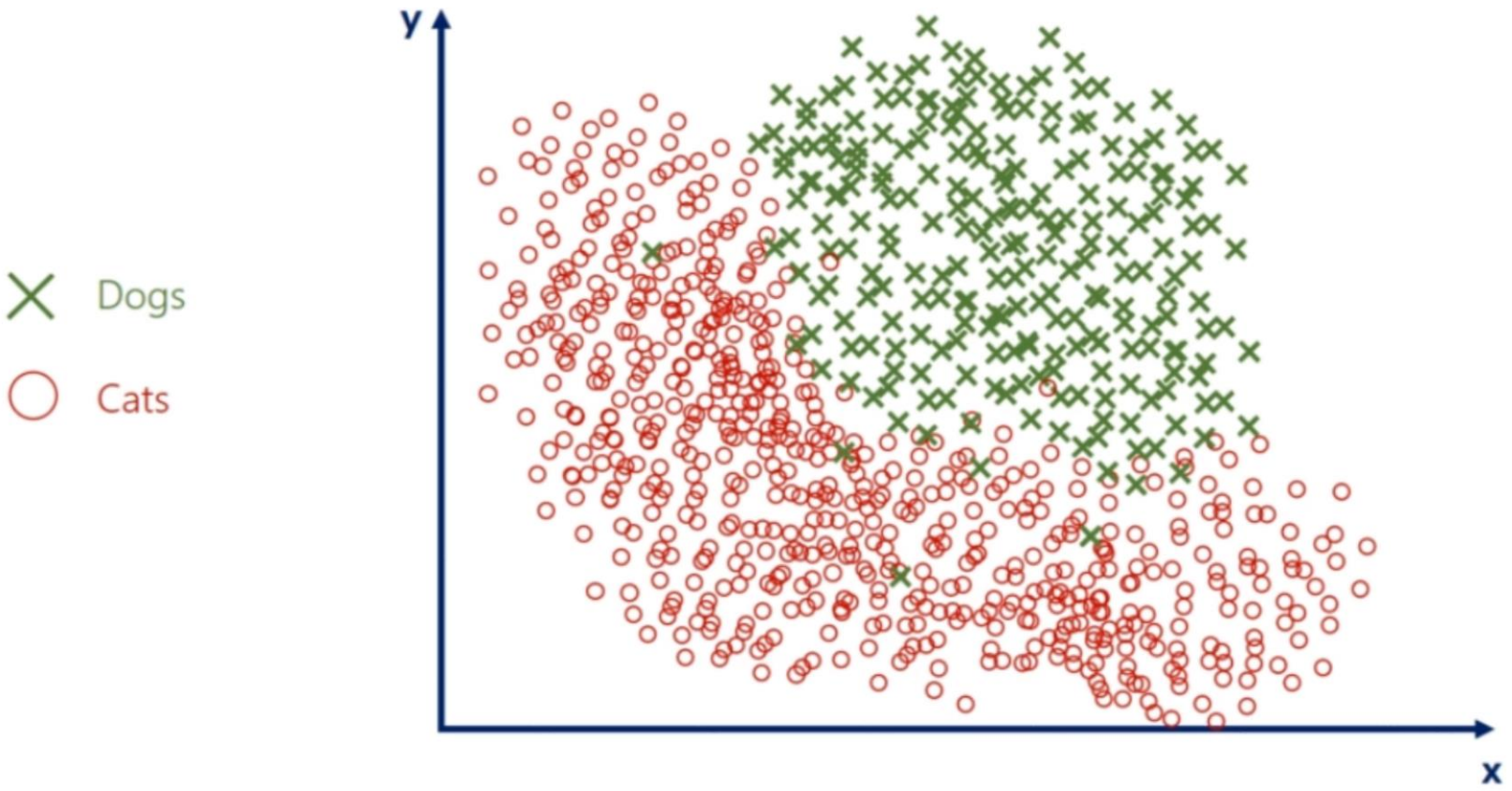
An **overfitted** model



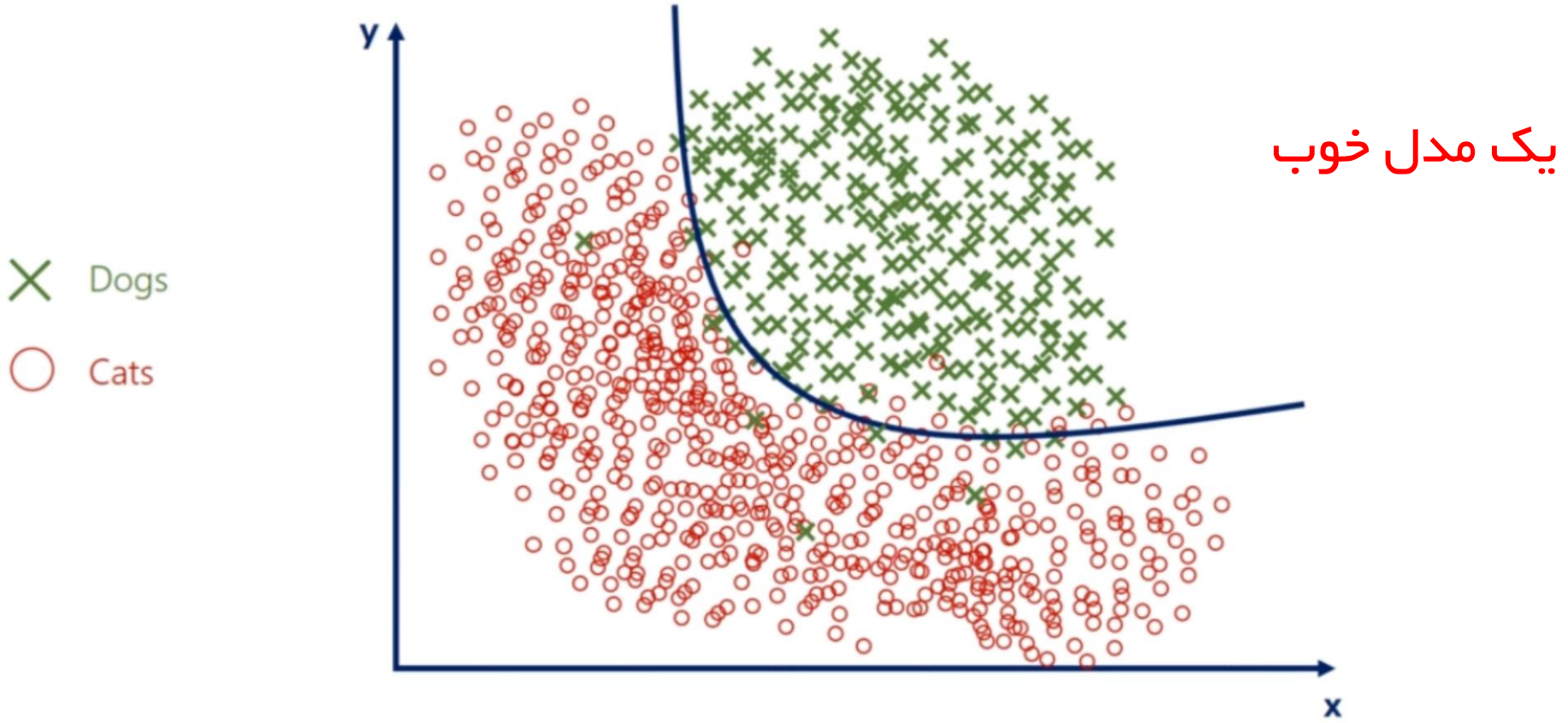
Captures all the noise, thus "missed the point"

- Low loss
- Low accuracy

Underfitting and overfitting. A classification example

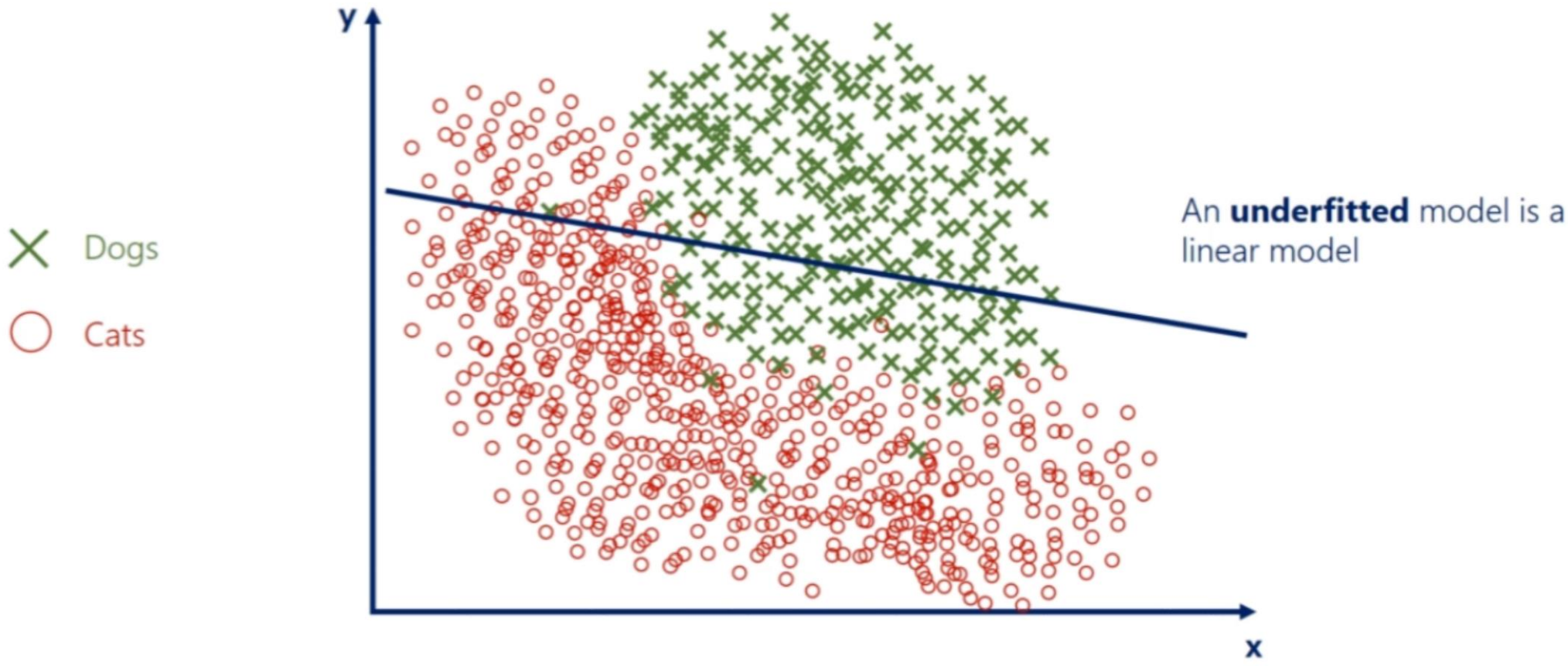


Underfitting and overfitting. A classification example



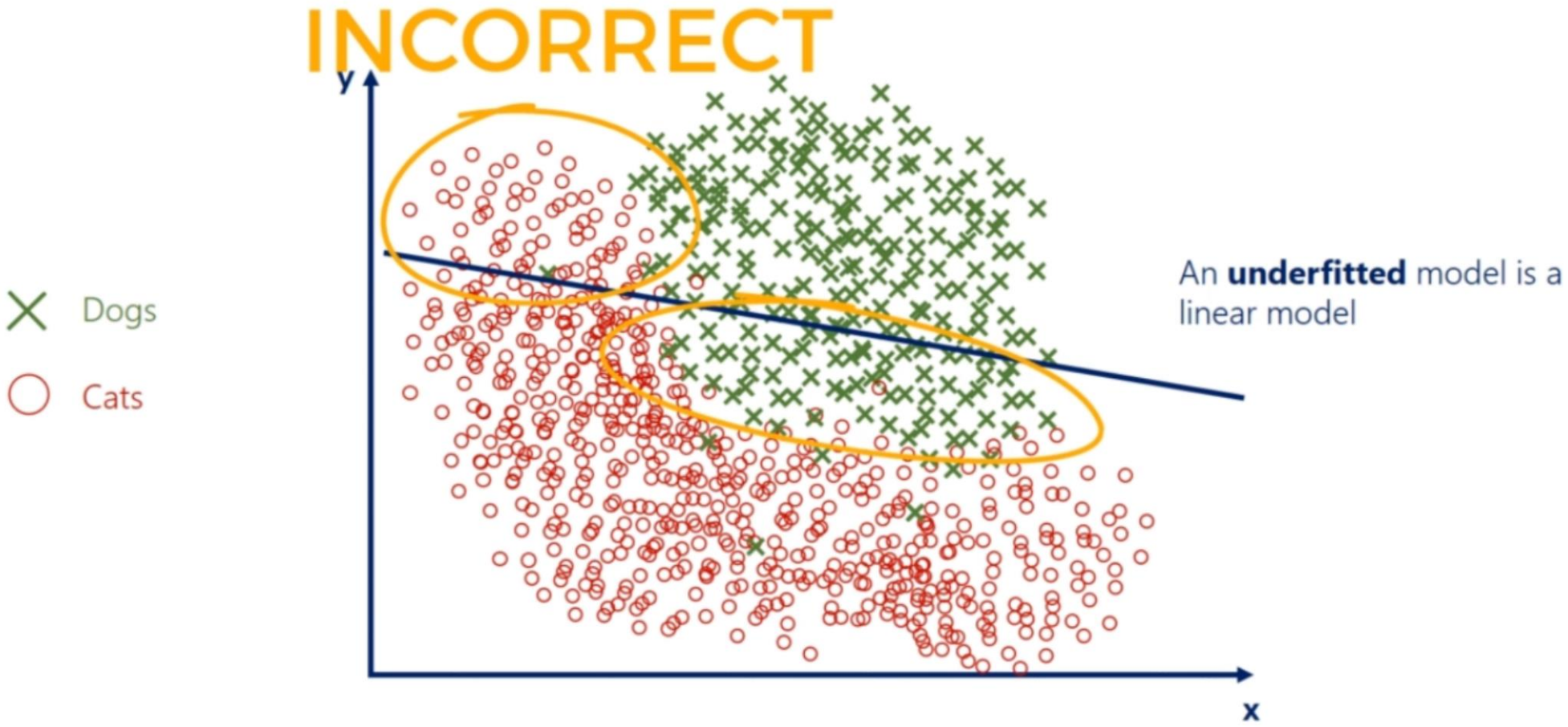
یک مدل خوب

Underfitting and overfitting. A classification example



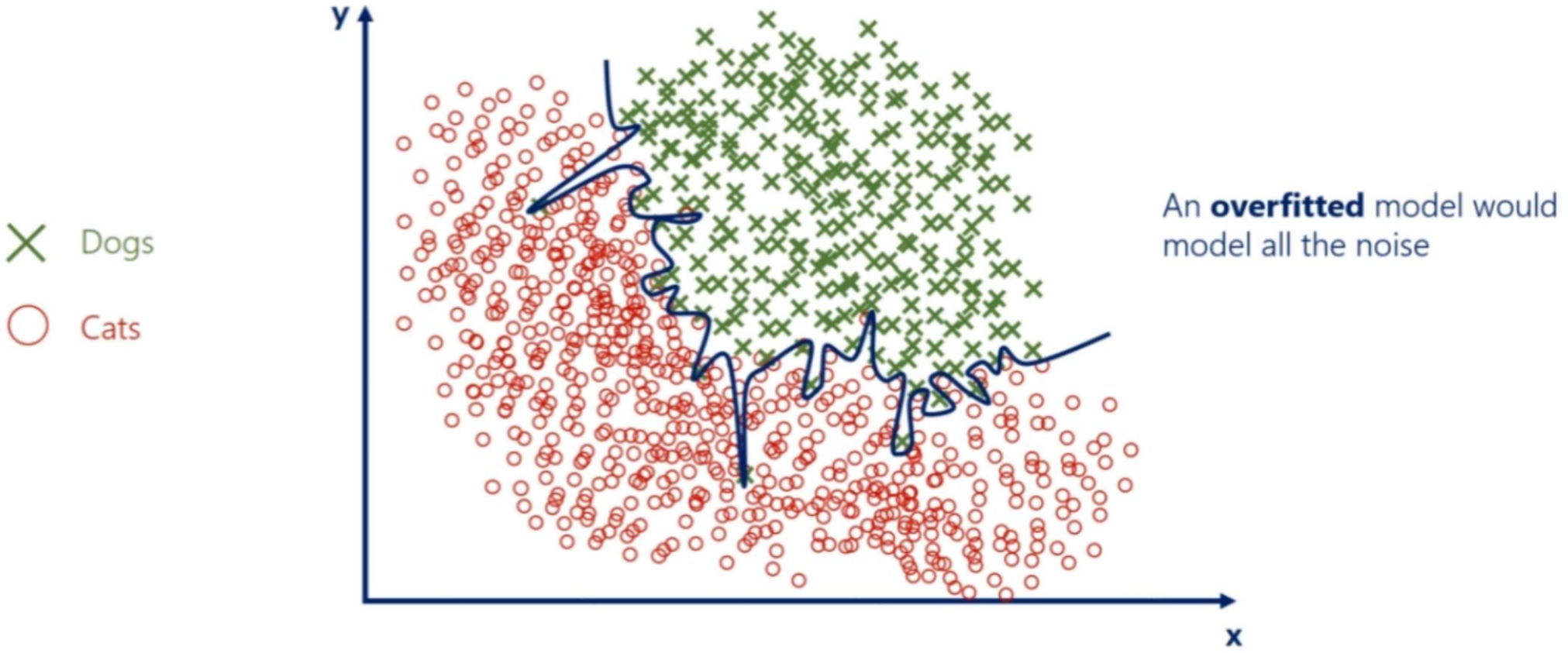
Underfitting and overfitting. A classification example

INCORRECT



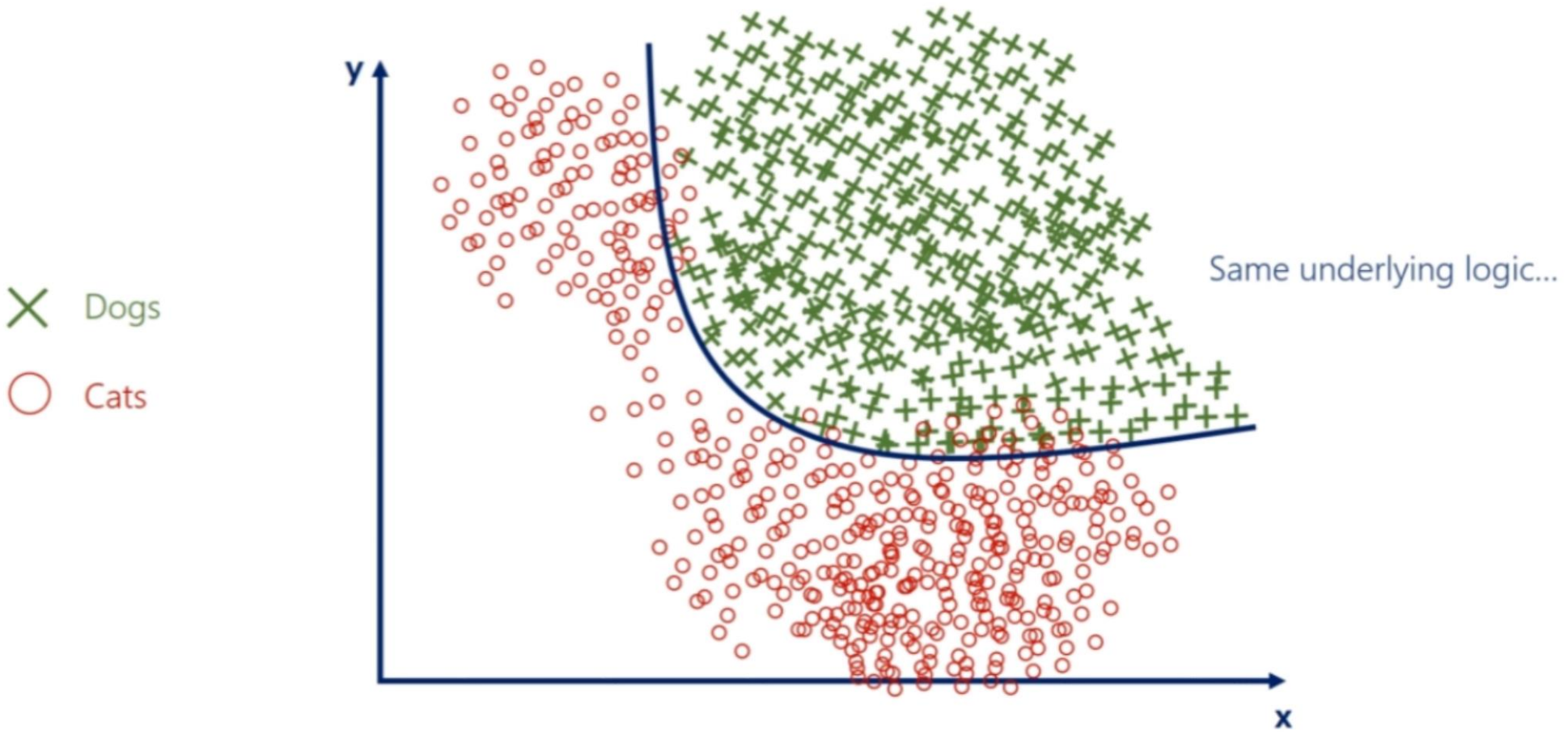
An **underfitted** model is a linear model

Underfitting and overfitting. A classification example

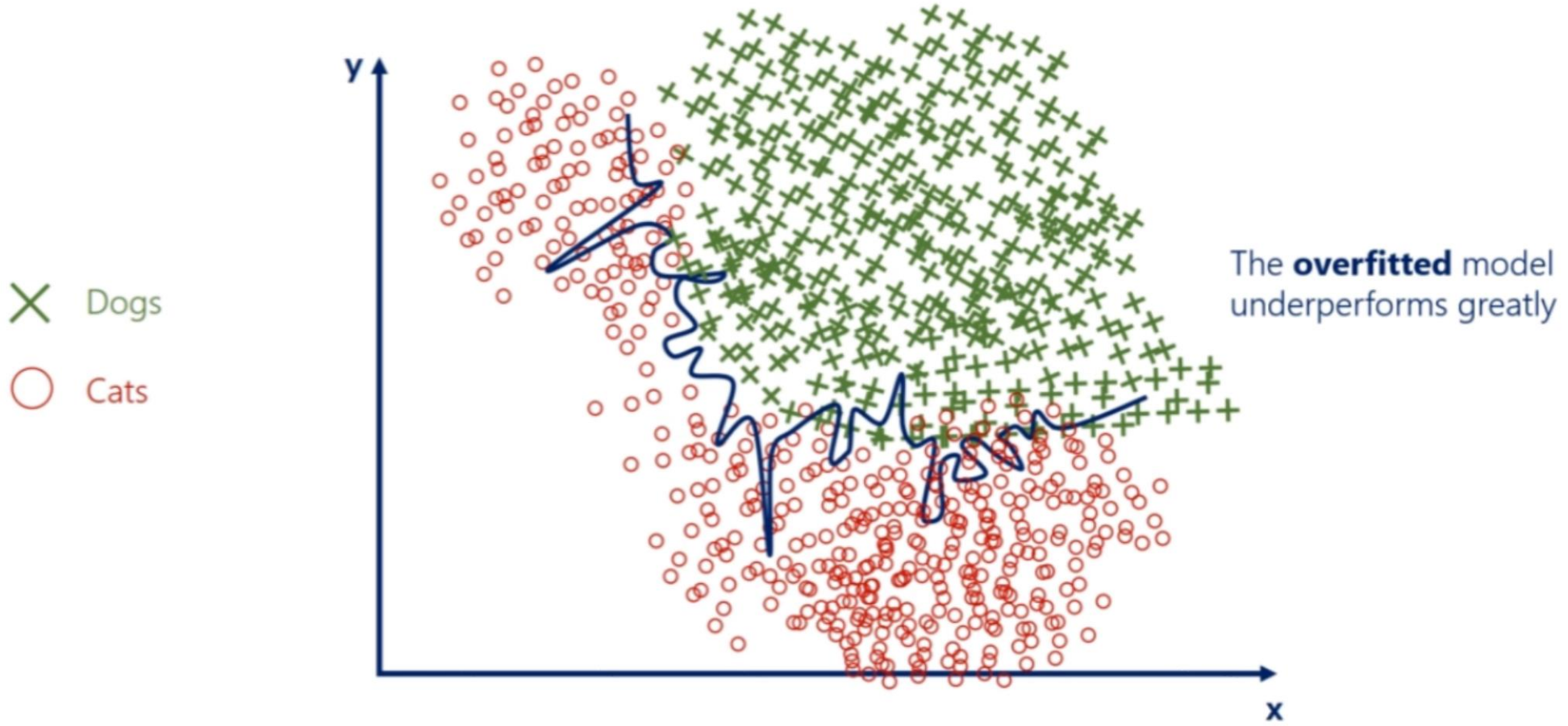


What if we got a different dataset with the same underlying relationship?

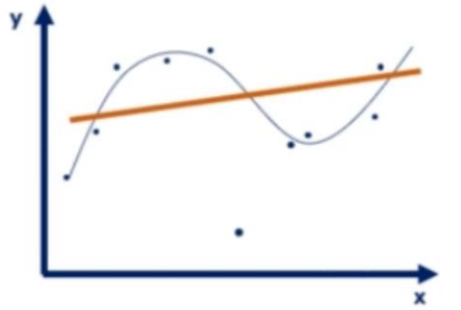
Underfitting and overfitting. A classification example

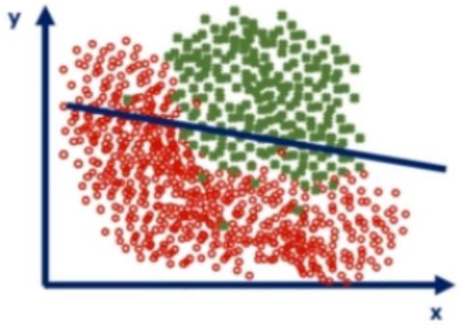


Underfitting and overfitting. A classification example



An **underfitted** model

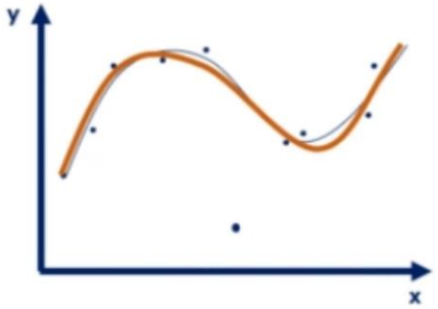


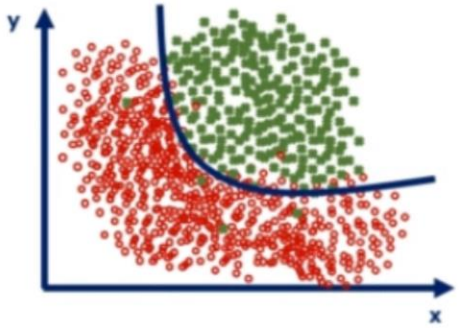


Doesn't capture any logic

- High loss
- Low accuracy

A **good** model

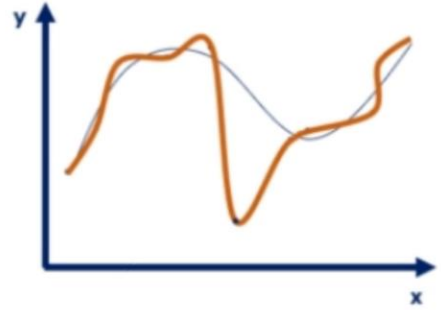


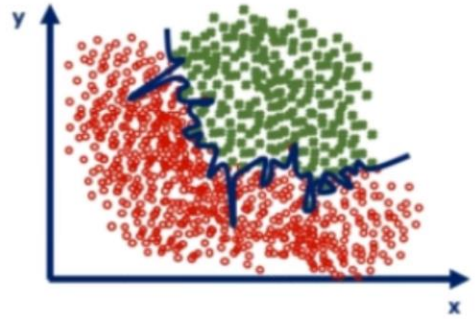


Captures the underlying logic of the dataset

- Low loss
- High accuracy

An **overfitted** model



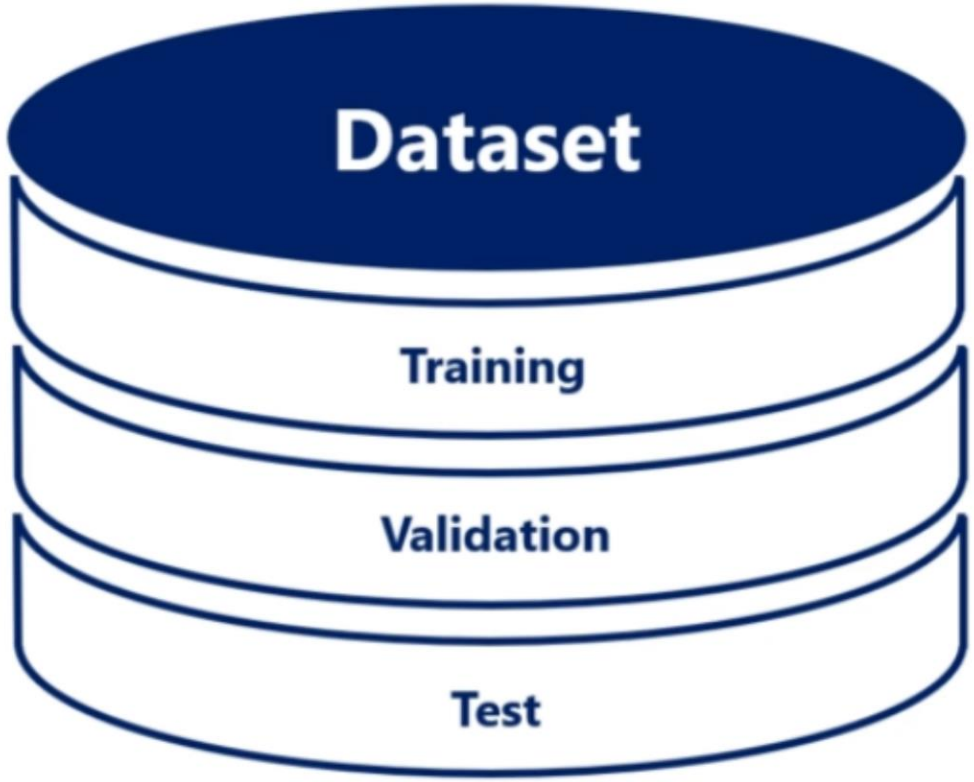


Captures all the noise, thus "missed the point"

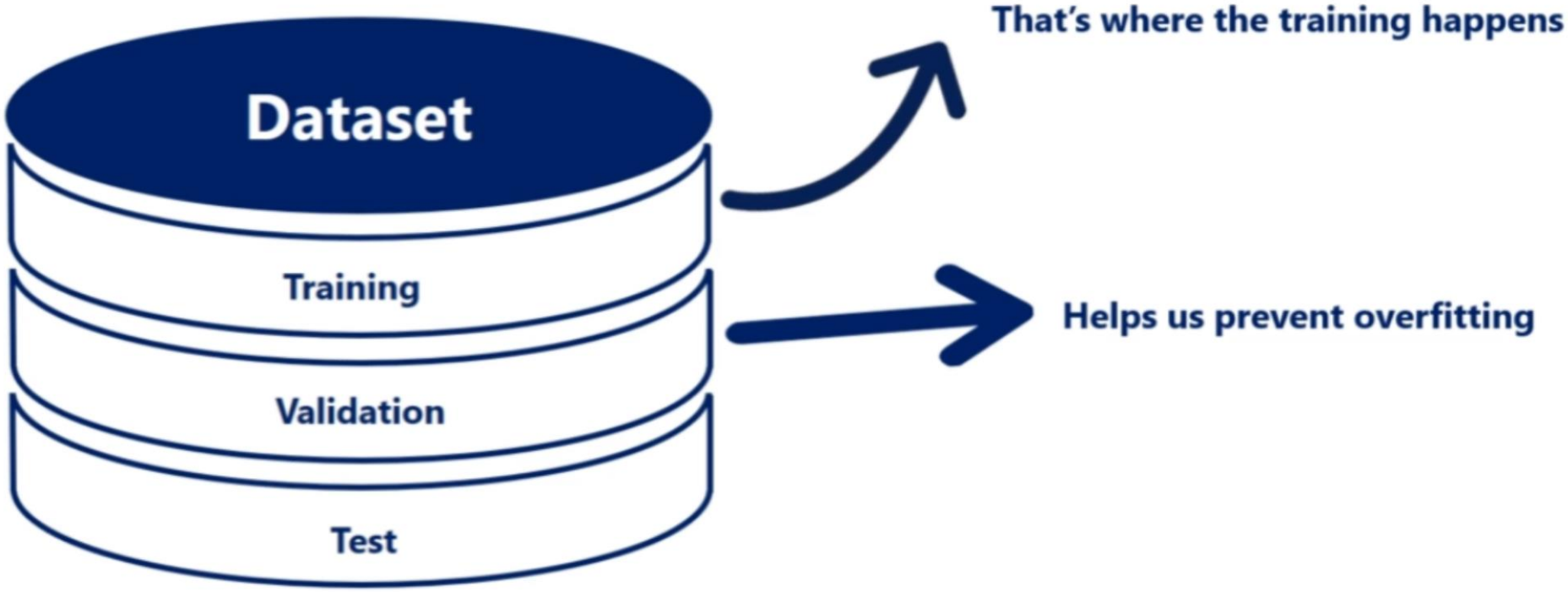
- Low loss
- Low accuracy

چطوری دیتاستمون رو تقسیم‌بندی
کنیم تا با مشکلات کمتری مثل
بیش‌برازش مواجه بشیم؟

Training, validation, and test

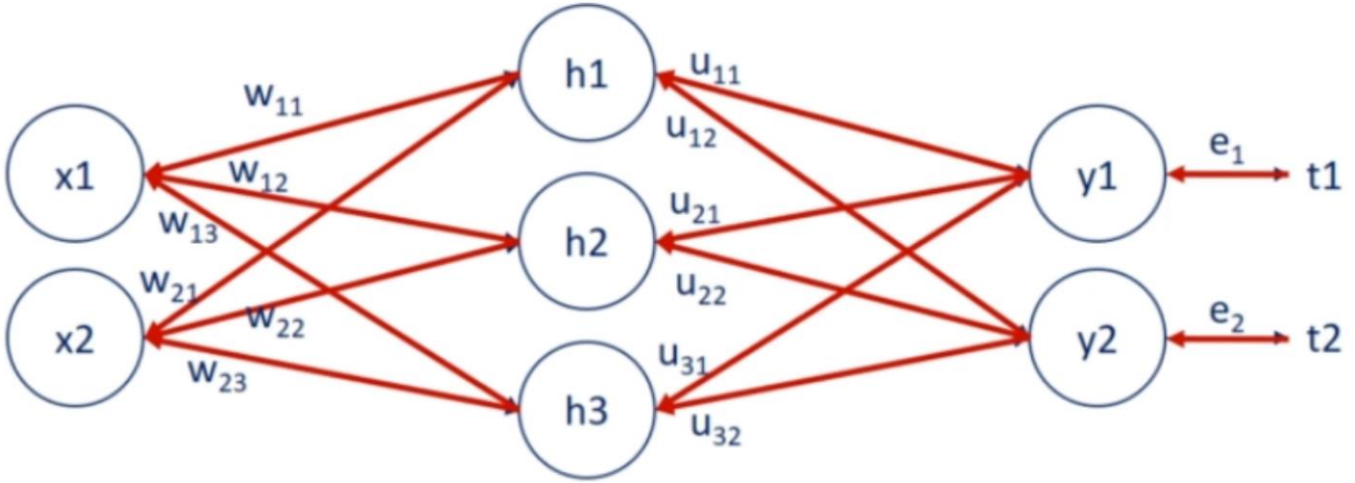


Training, validation, and test



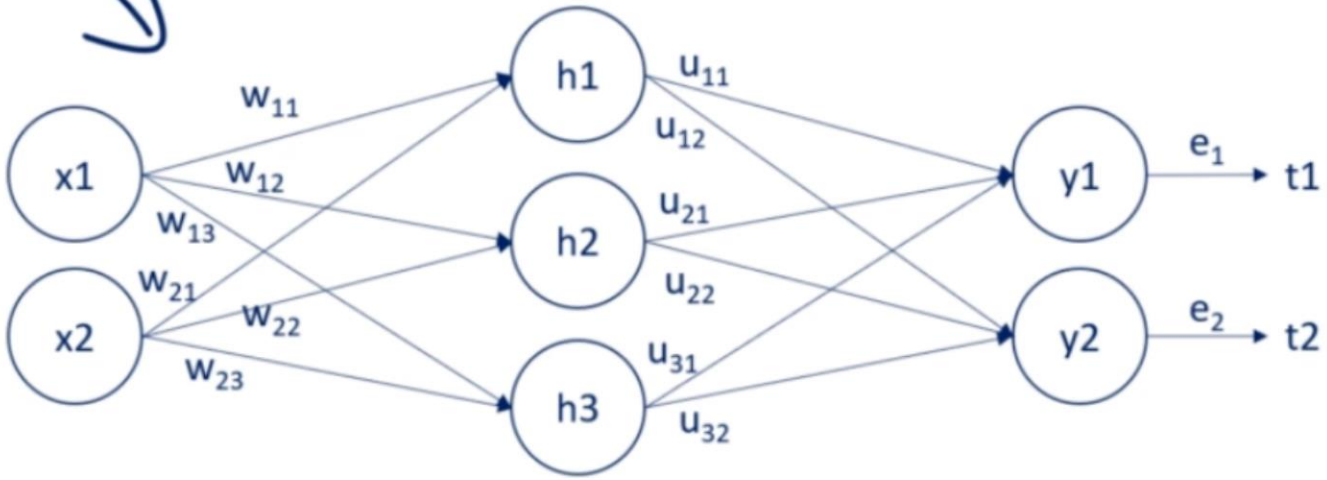
Training


Training

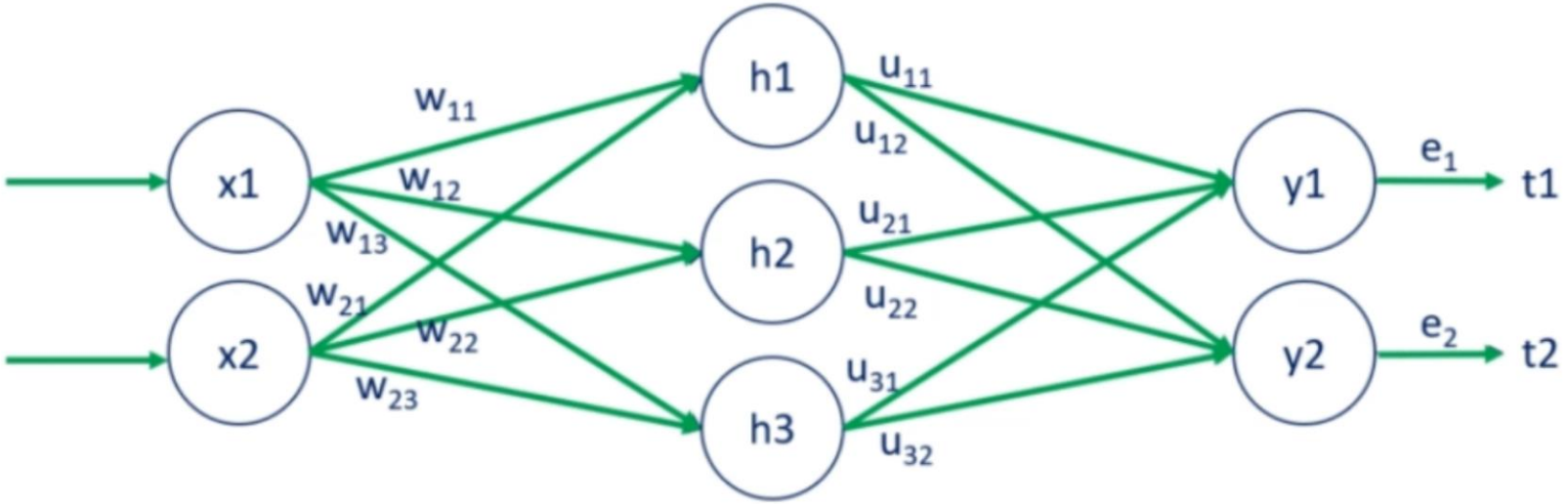

x1
x2
h1
h2
h3
y1
y2
w_{11}
w_{12}
w_{13}
w_{21}
w_{22}
w_{23}
u_{11}
u_{12}
u_{21}
u_{22}
u_{31}
u_{32}
e_1
e_2
t1
t2

Training

Somewhat trained



Validation



We just calculate the **validation loss.** On average it should equal the **training loss**

Training and validation

training_loss

validation_loss

```
Epoch 1. Training loss: 0.341. Validation loss: 0.192
Epoch 2. Training loss: 0.164. Validation loss: 0.143
Epoch 3. Training loss: 0.115. Validation loss: 0.106
Epoch 4. Training loss: 0.086. Validation loss: 0.093
Epoch 5. Training loss: 0.068. Validation loss: 0.087
Epoch 6. Training loss: 0.054. Validation loss: 0.084
Epoch 7. Training loss: 0.042. Validation loss: 0.074
Epoch 8. Training loss: 0.033. Validation loss: 0.077
Epoch 9. Training loss: 0.029. Validation loss: 0.090
Epoch 10. Training loss: 0.022. Validation loss: 0.072
Epoch 11. Training loss: 0.018. Validation loss: 0.074
Epoch 12. Training loss: 0.014. Validation loss: 0.075
Epoch 13. Training loss: 0.011. Validation loss: 0.081
Epoch 14. Training loss: 0.009. Validation loss: 0.085
Epoch 15. Training loss: 0.007. Validation loss: 0.085
Epoch 16. Training loss: 0.009. Validation loss: 0.091
Epoch 17. Training loss: 0.008. Validation loss: 0.099
Epoch 18. Training loss: 0.006. Validation loss: 0.100
Epoch 19. Training loss: 0.005. Validation loss: 0.092
Epoch 20. Training loss: 0.002. Validation loss: 0.084
Epoch 21. Training loss: 0.002. Validation loss: 0.092
Epoch 22. Training loss: 0.016. Validation loss: 0.085
Epoch 23. Training loss: 0.005. Validation loss: 0.094
Epoch 24. Training loss: 0.001. Validation loss: 0.085
Epoch 25. Training loss: 0.001. Validation loss: 0.088
Epoch 26. Training loss: 0.000. Validation loss: 0.087
Epoch 27. Training loss: 0.000. Validation loss: 0.089
Epoch 28. Training loss: 0.000. Validation loss: 0.091
Epoch 29. Training loss: 0.000. Validation loss: 0.092
Epoch 30. Training loss: 0.000. Validation loss: 0.092
```

Training and validation

training_loss

validation_loss

```
Epoch 1. Training loss: 0.341. Validation loss: 0.192
Epoch 2. Training loss: 0.164. Validation loss: 0.143
Epoch 3. Training loss: 0.115. Validation loss: 0.106
Epoch 4. Training loss: 0.086. Validation loss: 0.093
Epoch 5. Training loss: 0.068. Validation loss: 0.087
Epoch 6. Training loss: 0.054. Validation loss: 0.084
Epoch 7. Training loss: 0.042. Validation loss: 0.074
Epoch 8. Training loss: 0.033. Validation loss: 0.077
Epoch 9. Training loss: 0.029. Validation loss: 0.090
Epoch 10. Training loss: 0.022. Validation loss: 0.072
Epoch 11. Training loss: 0.018. Validation loss: 0.074
Epoch 12. Training loss: 0.014. Validation loss: 0.075
Epoch 13. Training loss: 0.011. Validation loss: 0.081
Epoch 14. Training loss: 0.009. Validation loss: 0.085
Epoch 15. Training loss: 0.007. Validation loss: 0.085
Epoch 16. Training loss: 0.009. Validation loss: 0.091
Epoch 17. Training loss: 0.008. Validation loss: 0.099
Epoch 18. Training loss: 0.006. Validation loss: 0.100
Epoch 19. Training loss: 0.005. Validation loss: 0.092
Epoch 20. Training loss: 0.002. Validation loss: 0.084
Epoch 21. Training loss: 0.002. Validation loss: 0.092
Epoch 22. Training loss: 0.016. Validation loss: 0.085
Epoch 23. Training loss: 0.005. Validation loss: 0.094
Epoch 24. Training loss: 0.001. Validation loss: 0.085
Epoch 25. Training loss: 0.001. Validation loss: 0.088
Epoch 26. Training loss: 0.000. Validation loss: 0.087
Epoch 27. Training loss: 0.000. Validation loss: 0.089
Epoch 28. Training loss: 0.000. Validation loss: 0.091
Epoch 29. Training loss: 0.000. Validation loss: 0.092
Epoch 30. Training loss: 0.000. Validation loss: 0.092
```

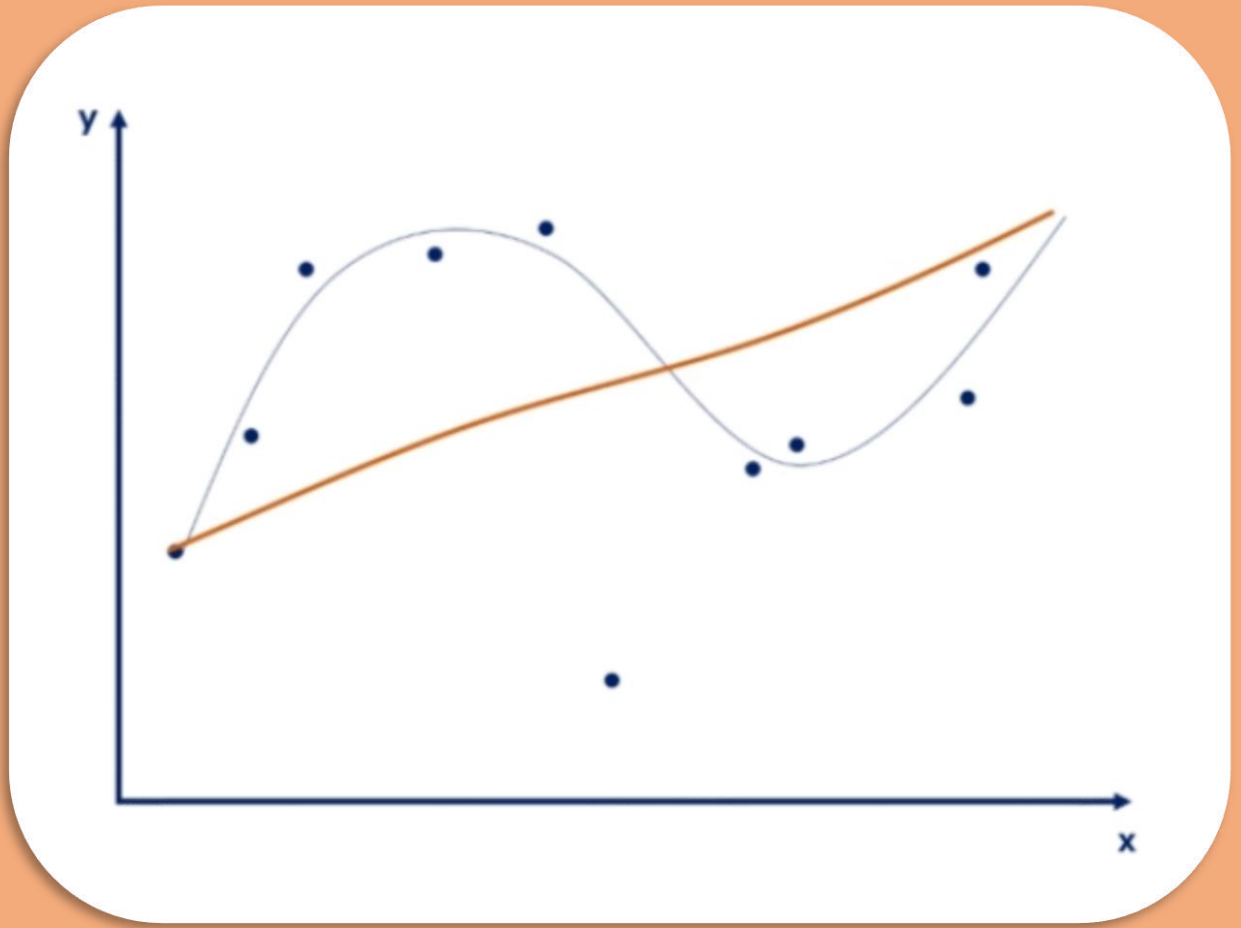
Training and validation

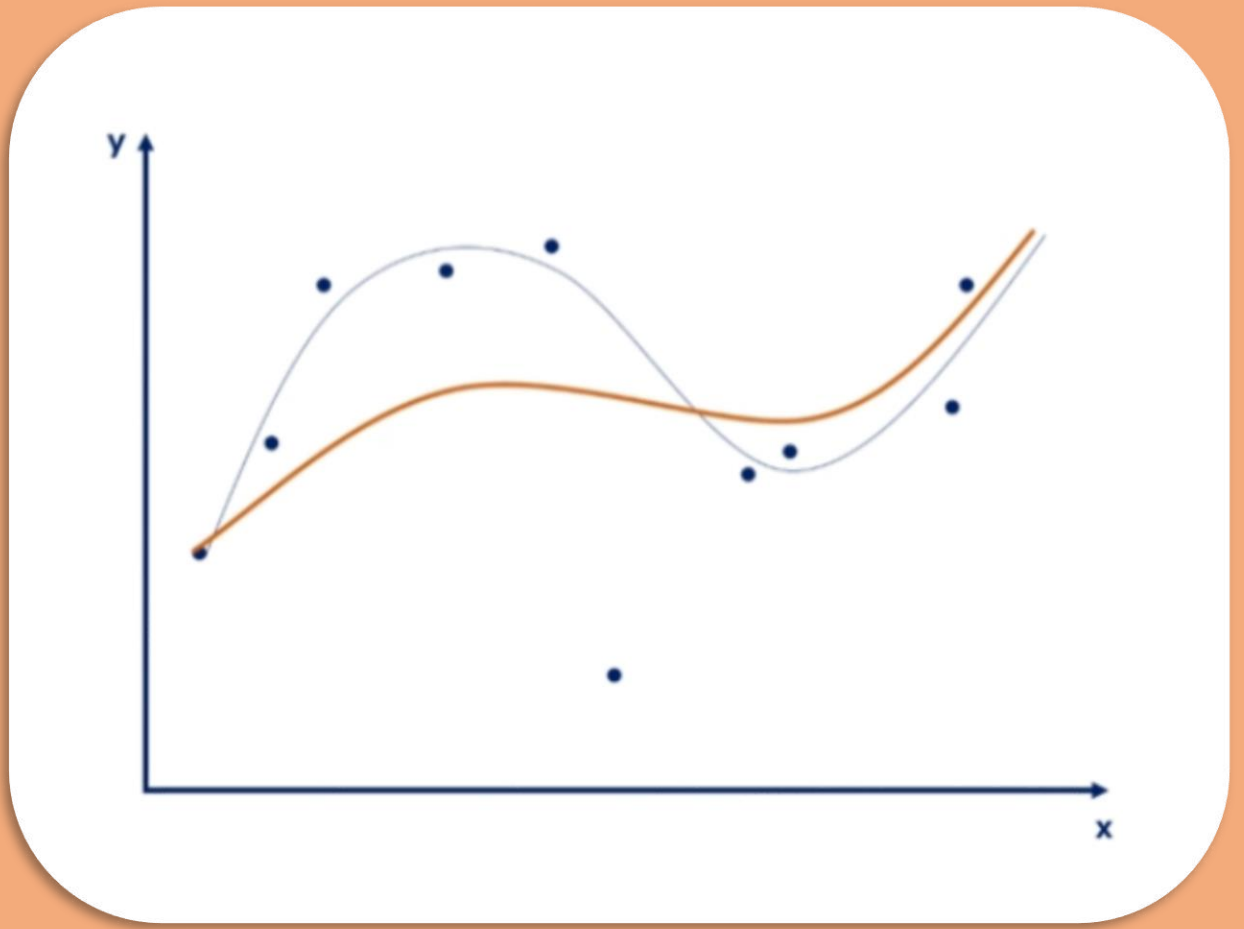
training_loss

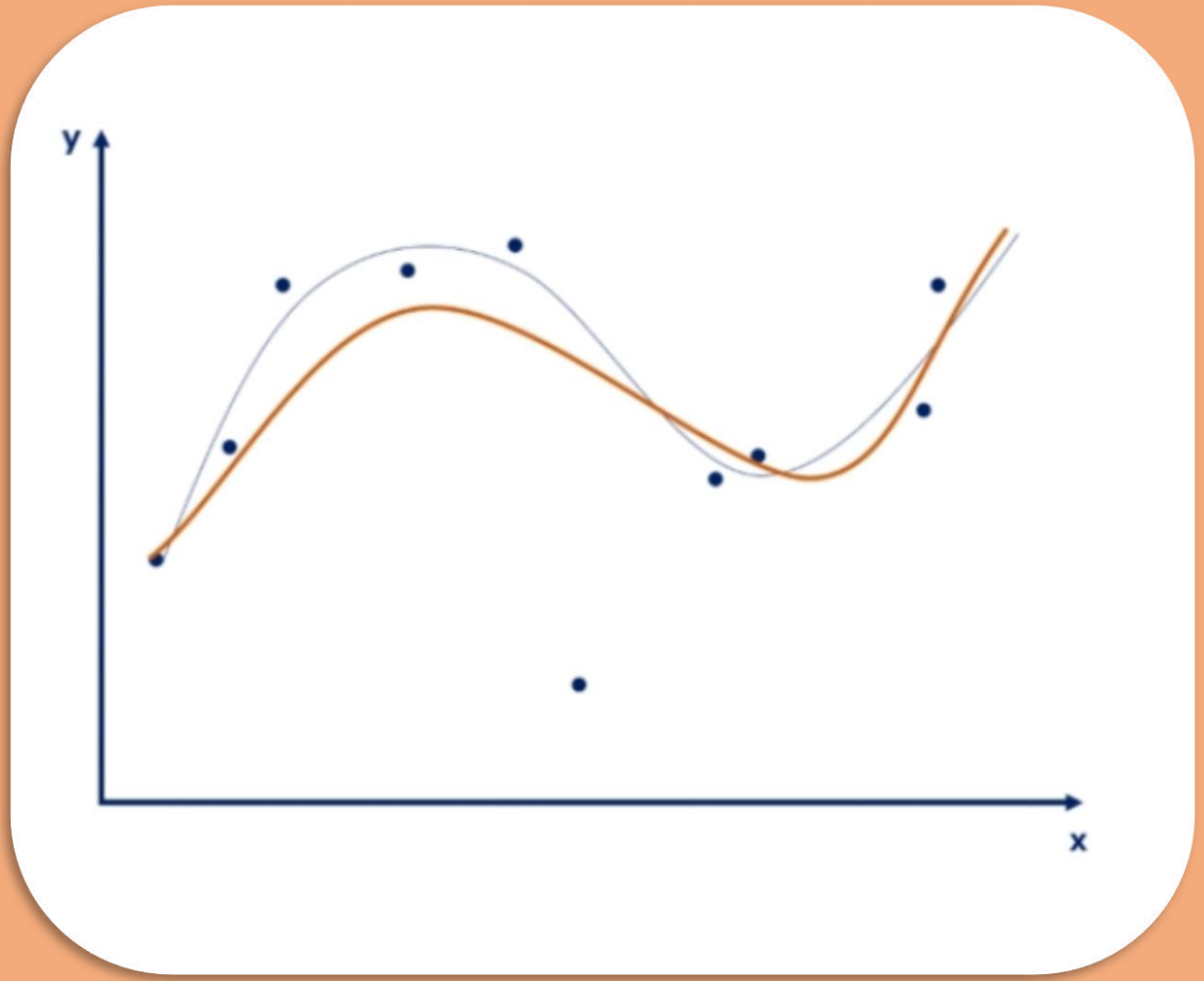
validation_loss

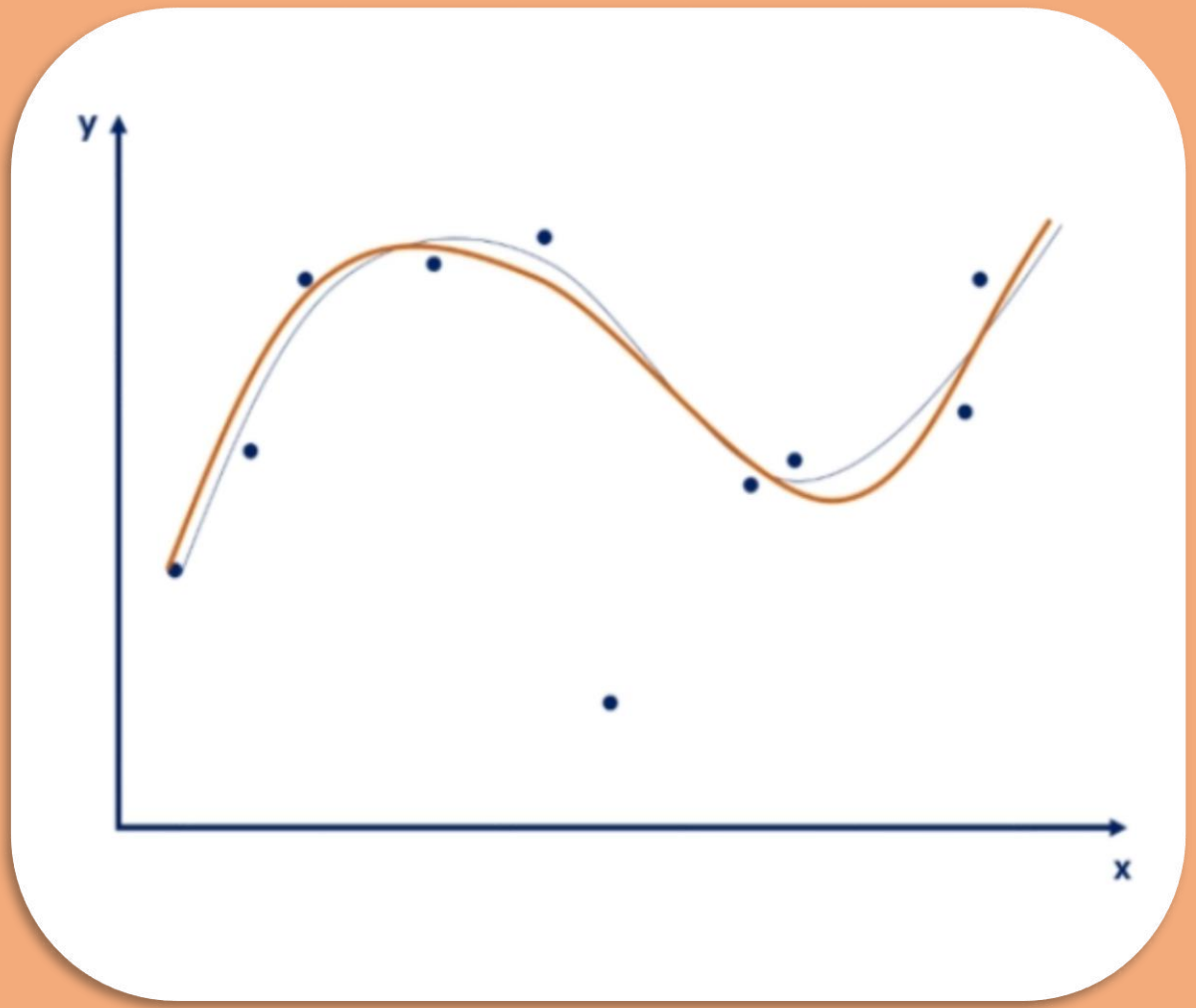
```
Epoch 1. Training loss: 0.341. Validation loss: 0.192
Epoch 2. Training loss: 0.164. Validation loss: 0.143
Epoch 3. Training loss: 0.115. Validation loss: 0.106
Epoch 4. Training loss: 0.086. Validation loss: 0.093
Epoch 5. Training loss: 0.068. Validation loss: 0.087
Epoch 6. Training loss: 0.054. Validation loss: 0.084
Epoch 7. Training loss: 0.042. Validation loss: 0.074
Epoch 8. Training loss: 0.033. Validation loss: 0.077
Epoch 9. Training loss: 0.029. Validation loss: 0.090
Epoch 10. Training loss: 0.022. Validation loss: 0.072
Epoch 11. Training loss: 0.018. Validation loss: 0.074
Epoch 12. Training loss: 0.014. Validation loss: 0.075
Epoch 13. Training loss: 0.011. Validation loss: 0.081
Epoch 14. Training loss: 0.009. Validation loss: 0.085
Epoch 15. Training loss: 0.007. Validation loss: 0.085
Epoch 16. Training loss: 0.009. Validation loss: 0.091
Epoch 17. Training loss: 0.008. Validation loss: 0.099
Epoch 18. Training loss: 0.006. Validation loss: 0.100
Epoch 19. Training loss: 0.005. Validation loss: 0.092
Epoch 20. Training loss: 0.002. Validation loss: 0.084
Epoch 21. Training loss: 0.002. Validation loss: 0.092
Epoch 22. Training loss: 0.016. Validation loss: 0.085
Epoch 23. Training loss: 0.005. Validation loss: 0.094
Epoch 24. Training loss: 0.001. Validation loss: 0.085
Epoch 25. Training loss: 0.001. Validation loss: 0.088
Epoch 26. Training loss: 0.000. Validation loss: 0.087
Epoch 27. Training loss: 0.000. Validation loss: 0.089
Epoch 28. Training loss: 0.000. Validation loss: 0.091
Epoch 29. Training loss: 0.000. Validation loss: 0.092
Epoch 30. Training loss: 0.000. Validation loss: 0.092
```

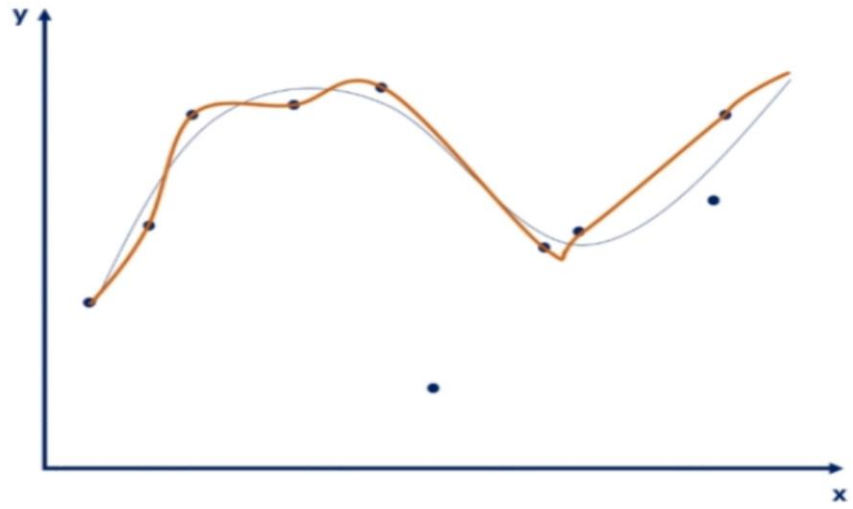
WE ARE OVERFITTING!!!

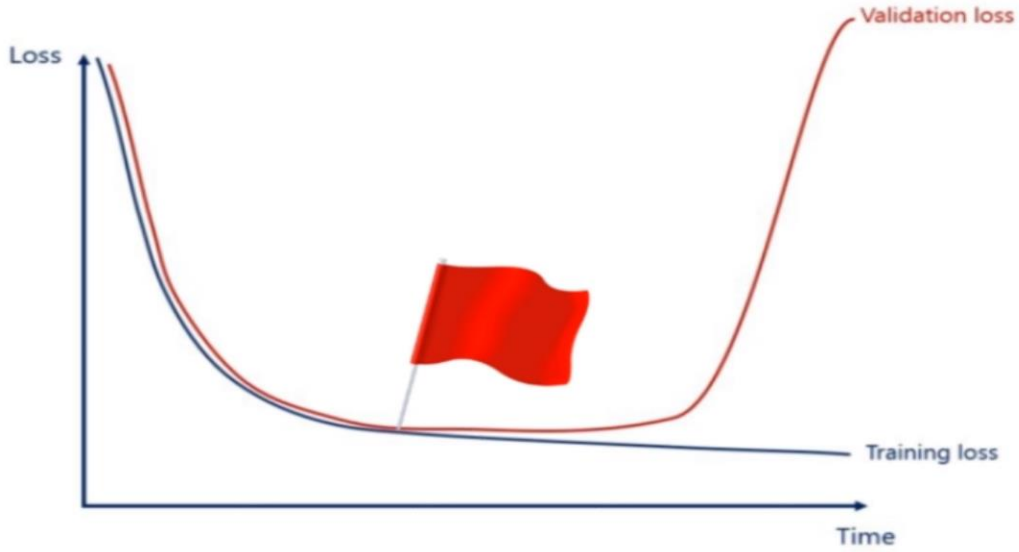

y
x

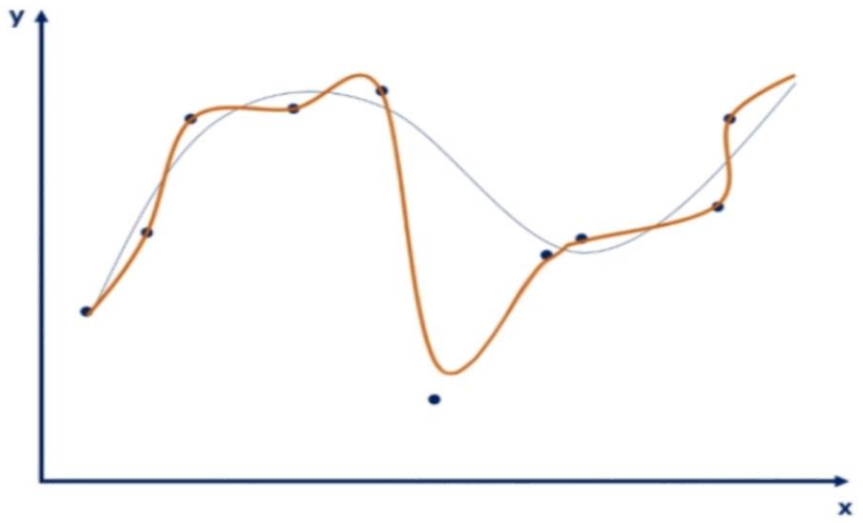

y
x

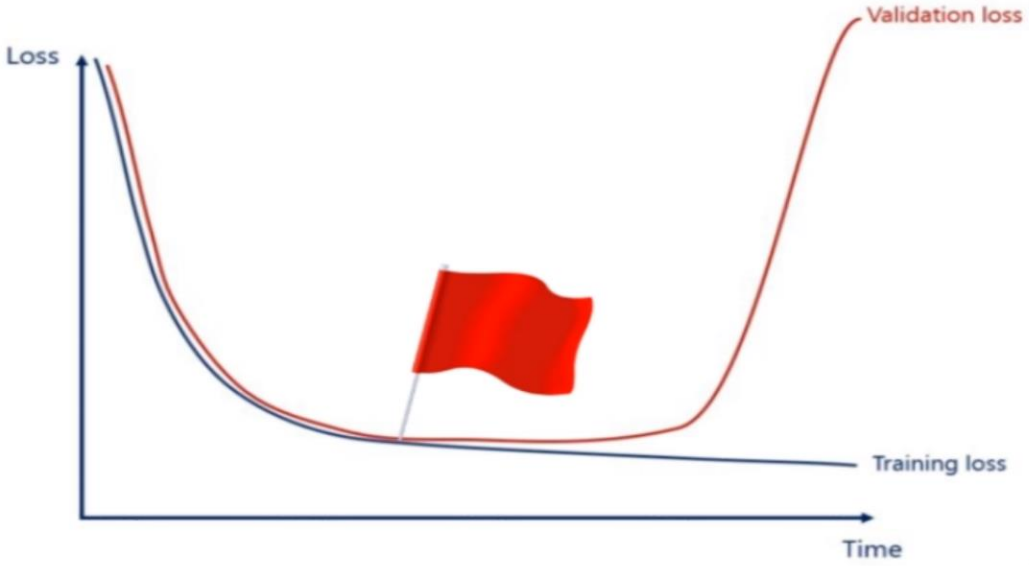

y
x


y
x

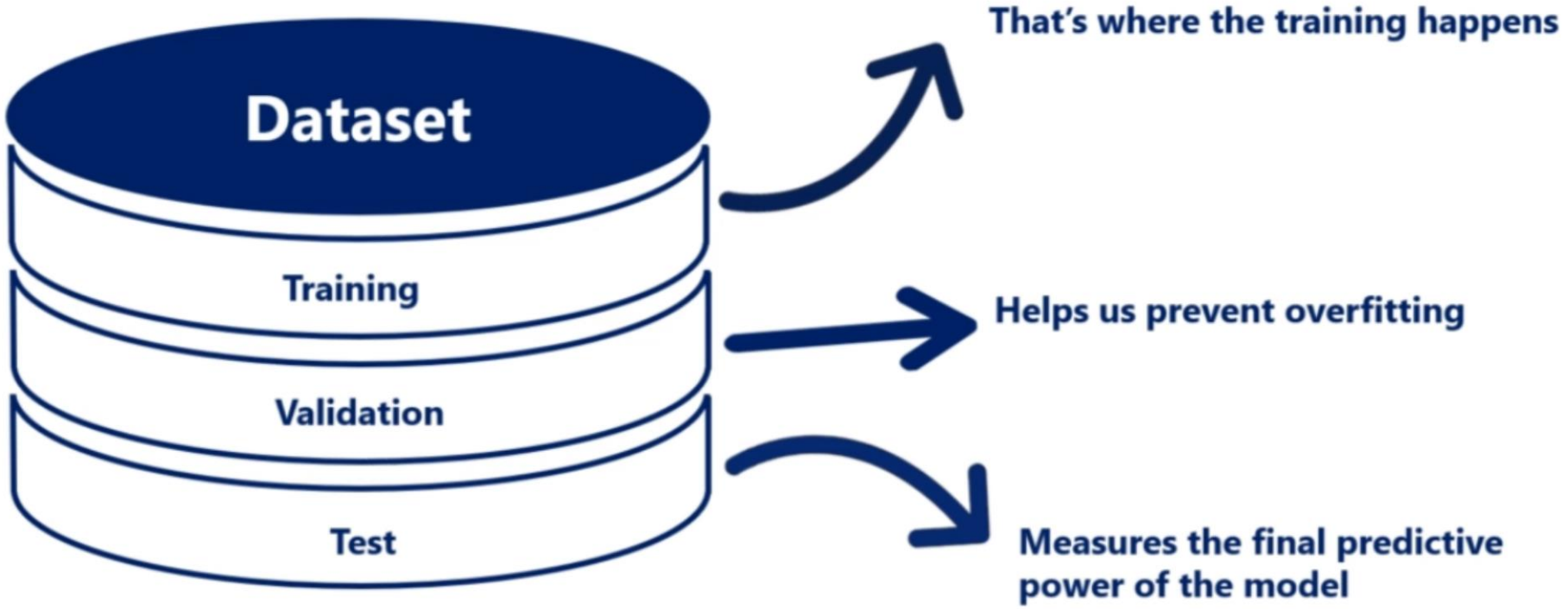

y
x


Loss
Validation loss
Training loss
Time

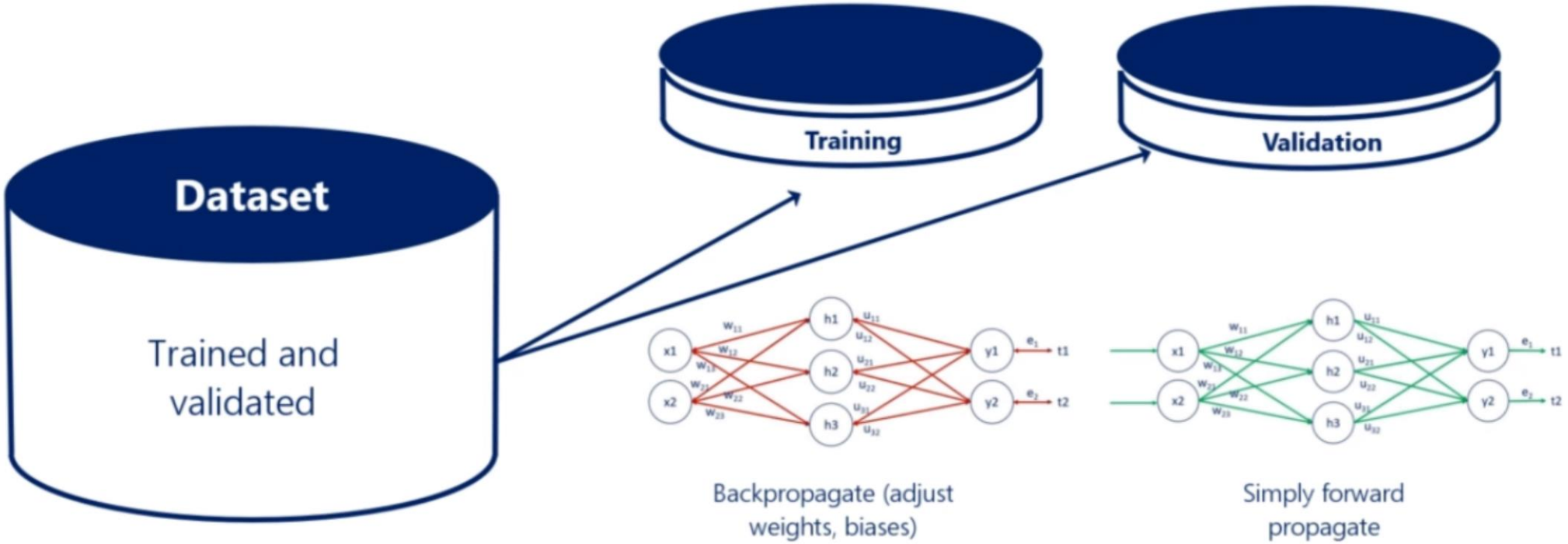

y
x

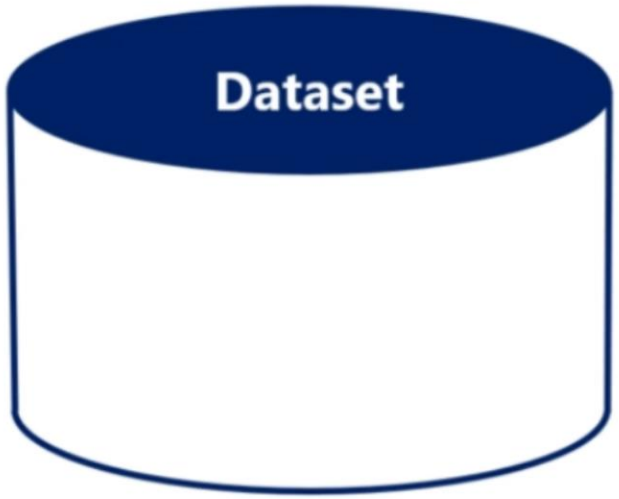

Loss
Validation loss
Training loss
Time

Training, validation, and test



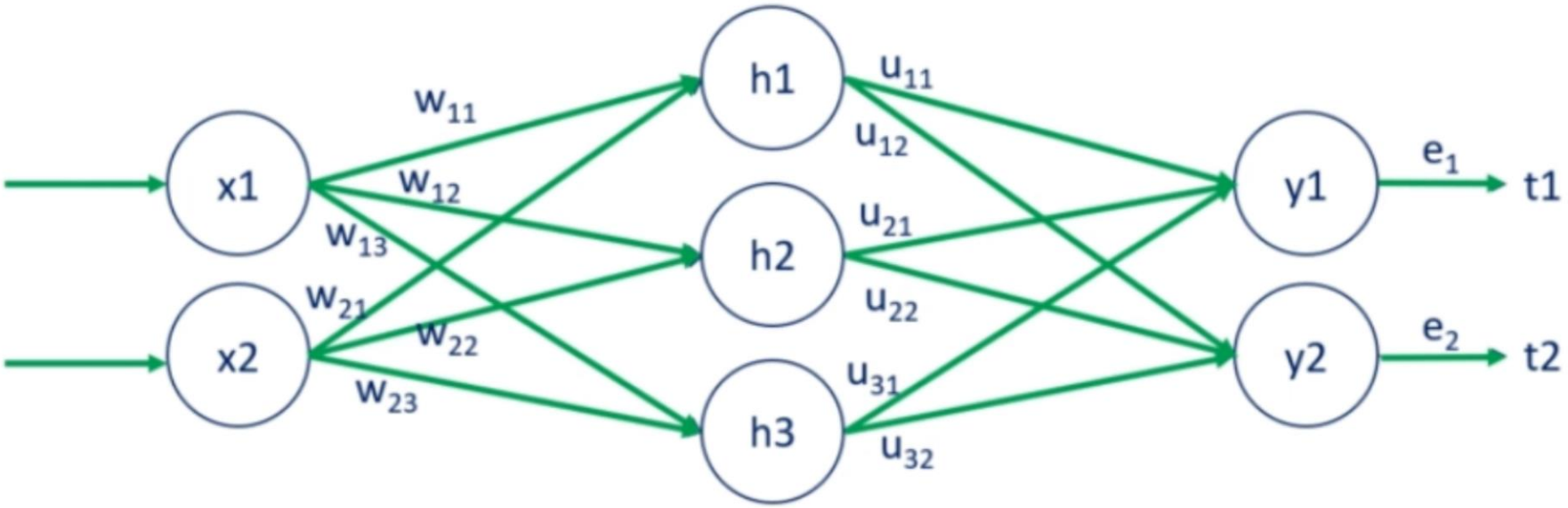
Training, validation, and test





Do we split the dataset evenly?
How do practitioners approach this?


Dataset
Training
Validation
Test
80%
Training
10%
Validation
10%
Test
70%
Training
20%
Validation
10%
Test



The accuracy we obtain at this stage is the accuracy of the algorithm

Training, validation, and test

Sometimes we have a small dataset

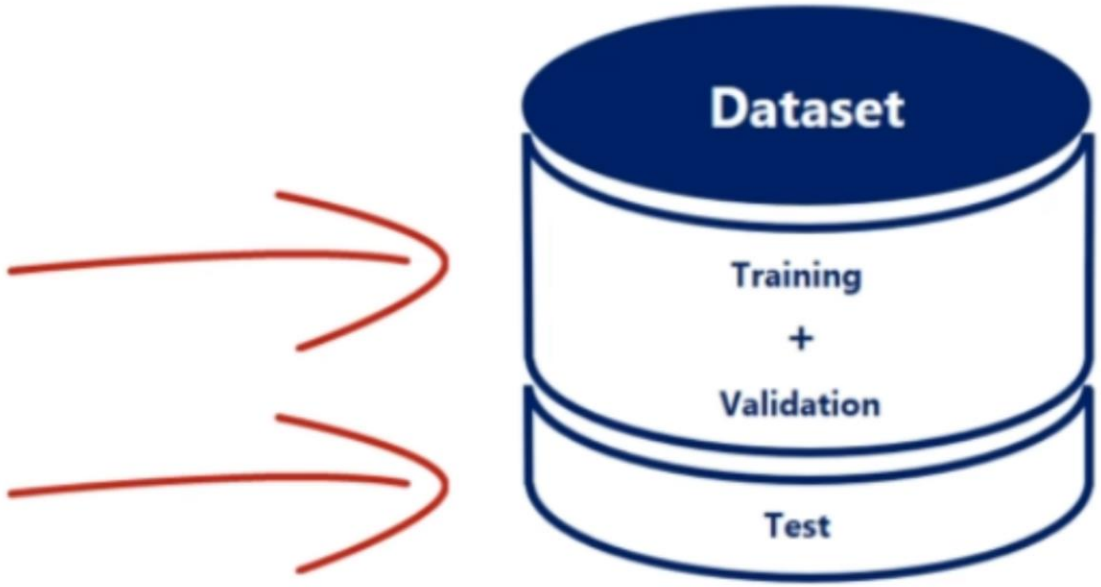
We can't afford to split it in three...

Because the algorithm may not **learn anything**

N-FOLD CROSS-VALIDATION

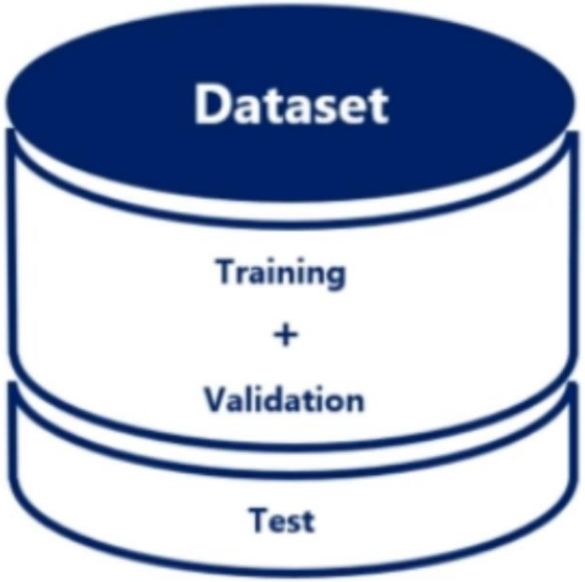
/ sometimes k-fold cross-validation /

N-FOLD CROSS-VALIDATION



N-FOLD CROSS-VALIDATION

11,000

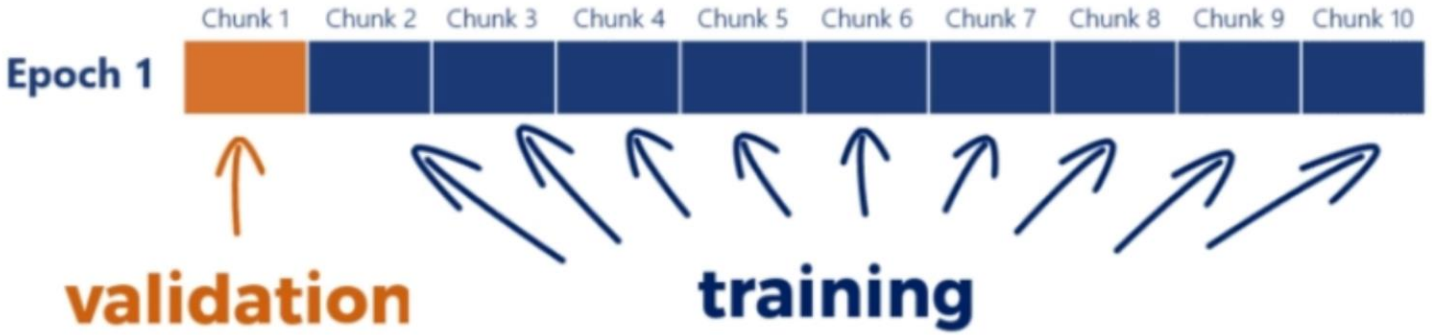


10,000

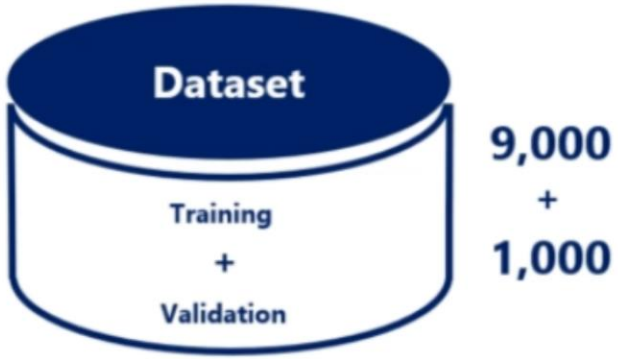
1,000

10-FOLD CROSS-VALIDATION

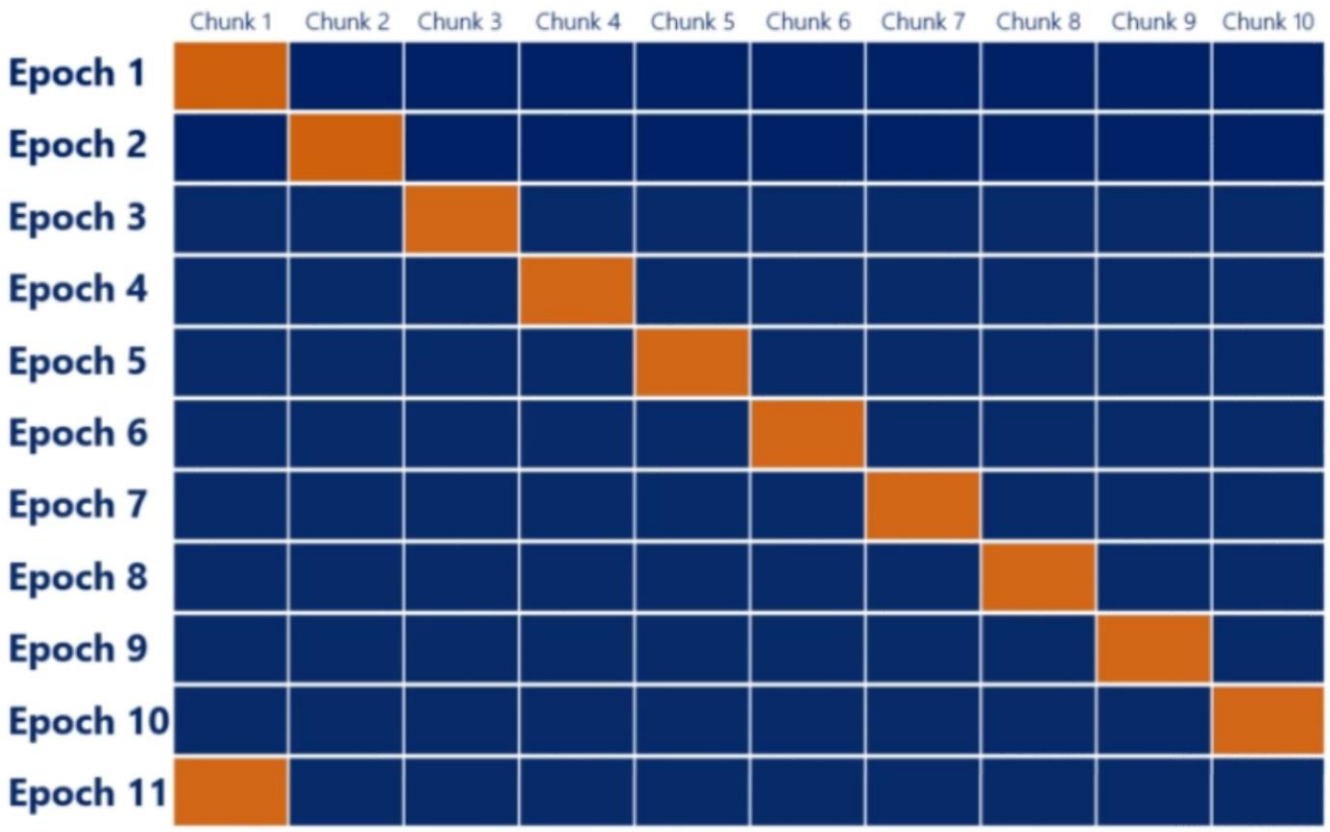
9,000
+
1,000



10-FOLD CROSS-VALIDATION



9,000
+
1,000



N-FOLD CROSS-VALIDATION

Pros

Utilized more data

We have a model

Cons

Possibly overfitted a bit

Early Stopping

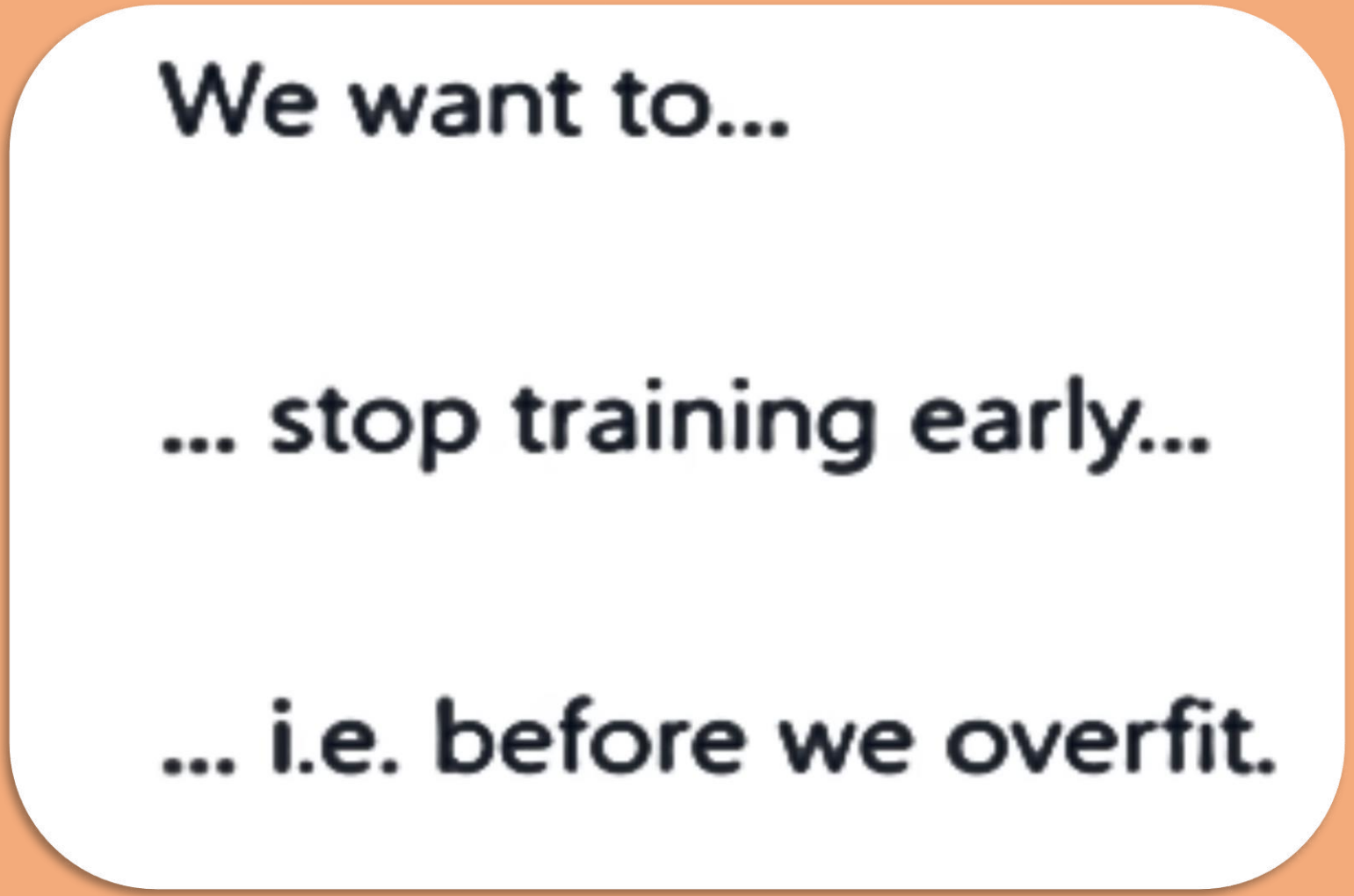
In machine learning, early stopping

used to avoid overfitting

when training a model with an iterative algorithm, such as gradient descent. Such methods update the learner so as to make it better fit the training data with each iteration.

توقف زودهنگام تکنیکی است که به منظور جلوگیری از بیش‌برازش بکار گرفته می‌شود

Early Stopping



Early stopping

1 Preset number of epochs

Learning

In [9]:
```
for e in range(100):

    _, curr_loss = sess.run([optimize, mean_loss],
                            feed_dict = {inputs: training_data['inputs'], targets: training_data['targets']})

    print (curr_loss)
```

Pros:

1. Eventually, solves the problem

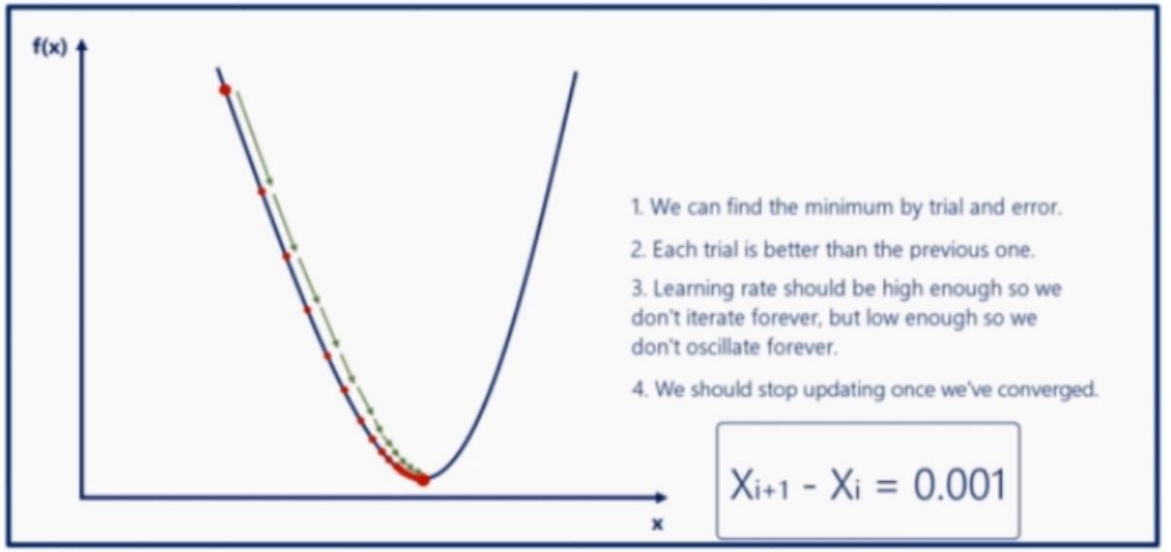
Cons:

1. No guarantee that the min is reached
2. Maybe doesn't minimize at all
3. Naive

Early stopping

2

Stop when updates become too small



Pros:

1. We are sure the loss is minimized
2. Saves computing power

Cons:

?

Early stopping

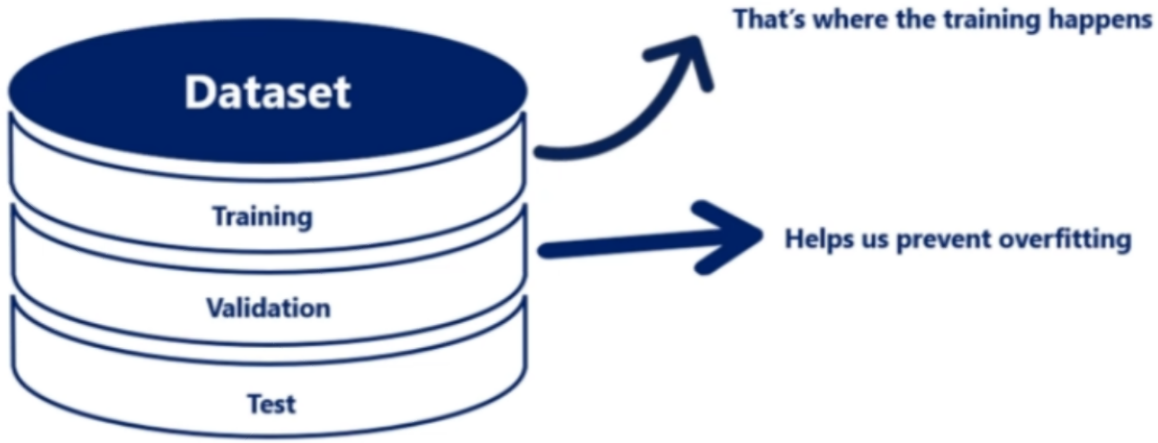
	Preset epochs	Updates too small	
Solves the problem	✓	✓	
Certain that loss is minimized	✗	✓	
Doesn't iterate uselessly	✗	✓	

Early stopping

	Preset epochs	Updates too small	
Solves the problem	✓	✓	
Certain that loss is minimized	✗	✓	
Doesn't iterate uselessly	✗	✓	
Prevents overfitting	✗	✗	

Early stopping

3 Validation set strategy



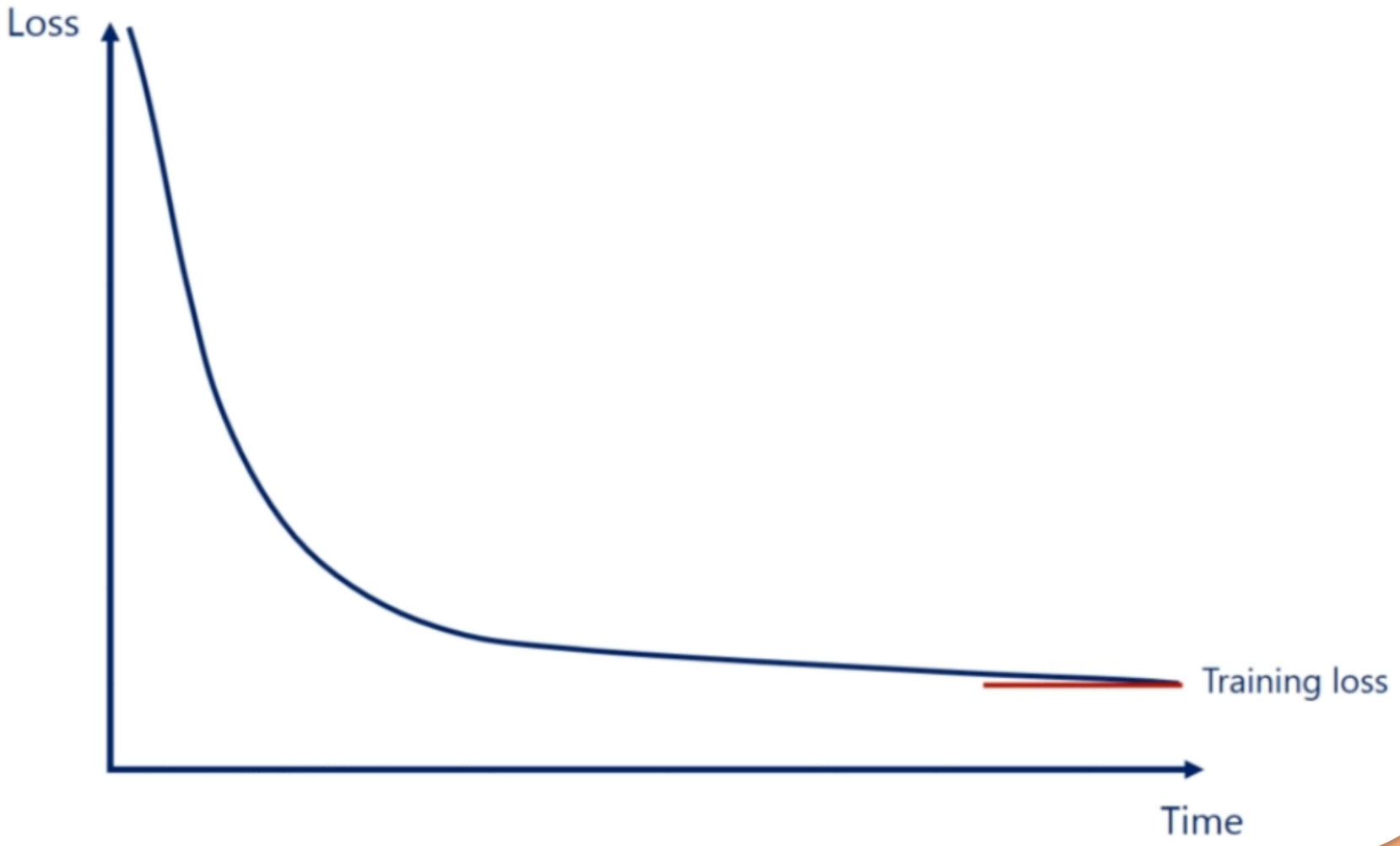
Pros:

1. We are sure the validation loss is minimized
2. Saves computing power
3. Prevents overfitting

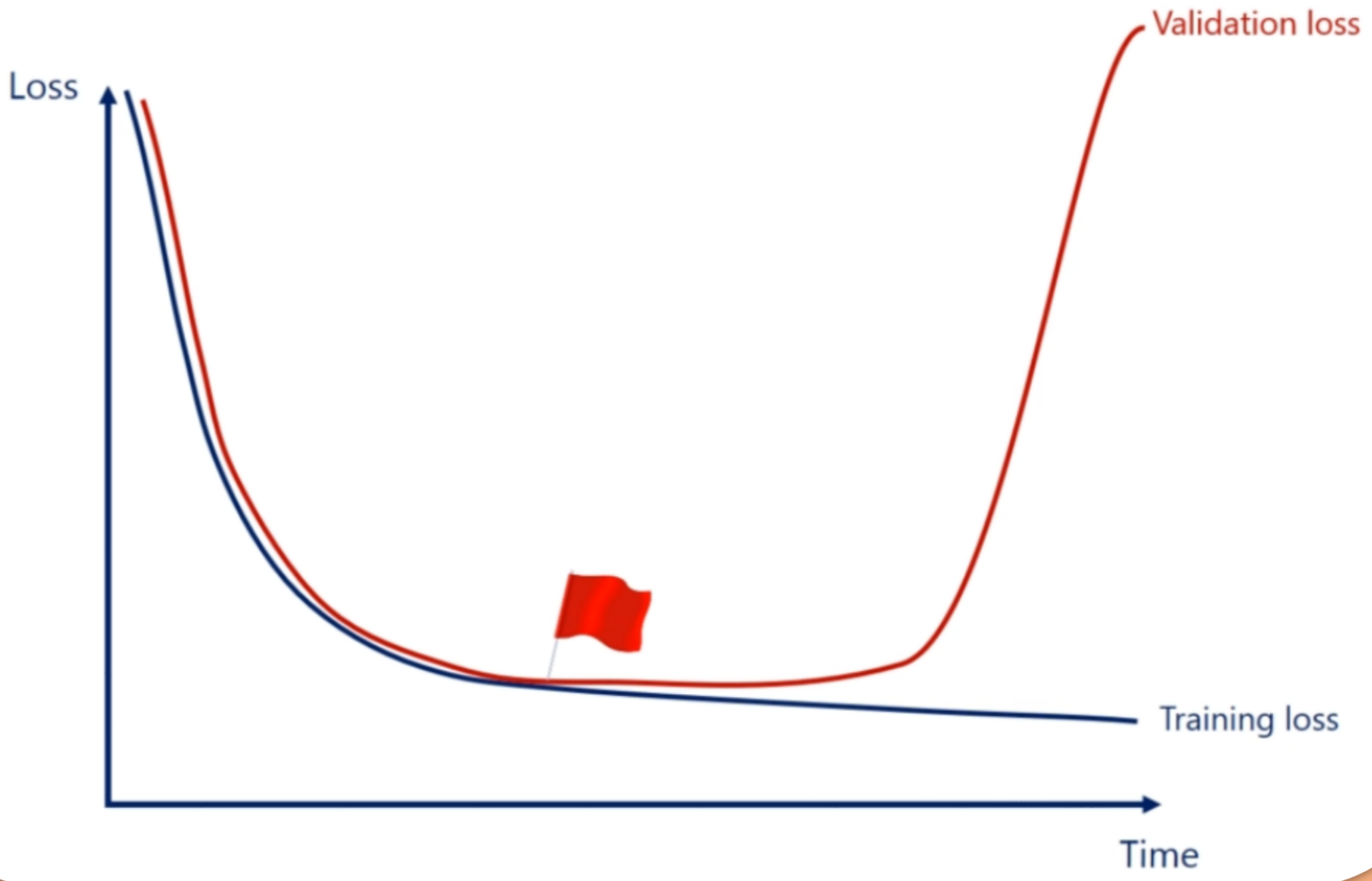
Cons:

?

Typical training



Typical training



Early stopping

	Preset epochs	Updates too small	Validation set
Solves the problem	✓	✓	✓
Certain that loss is minimized	✗	✓	✓
Doesn't iterate uselessly	✗	✓	✗
Prevents overfitting	✗	✗	✓

Early stopping

	Preset epochs	Updates too small	Validation set
Solves the problem	✓	✓	✓
Certain that loss is minimized	✗	✓	✓
Doesn't iterate uselessly	✗	✓	✗
Prevents overfitting	✗	✗	✓

Minimal example **Best practice**

Stay Tuned

Data Science course has a lot of fun for devoted students