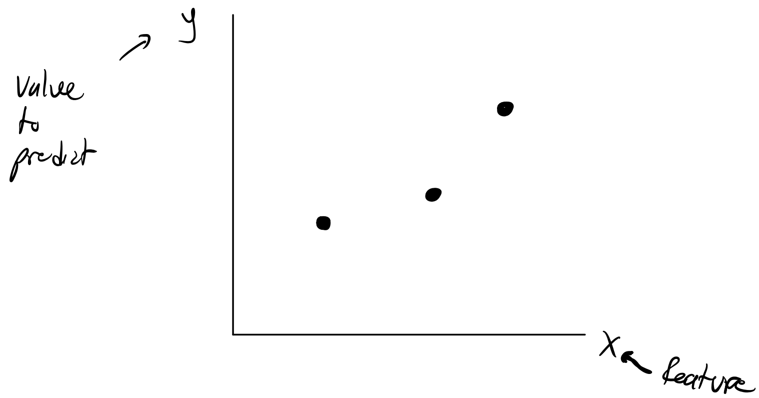
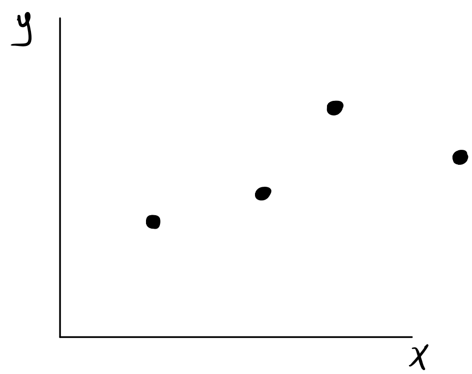
Occam's Razor

Task: regression (not necessarily linear)





Overfitting

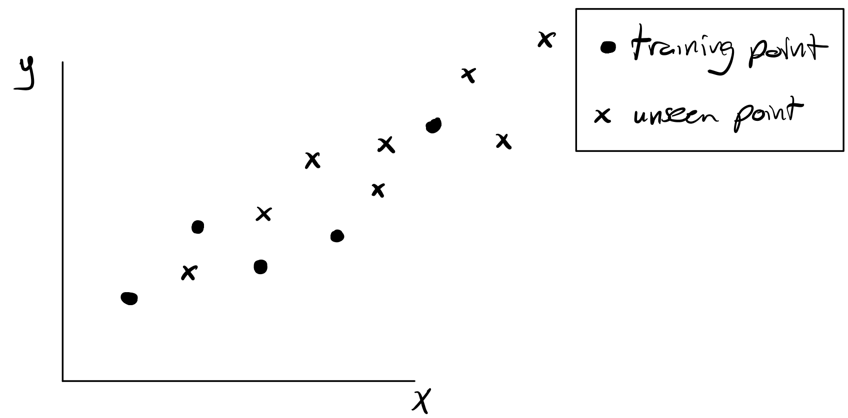
Goal: make predictions for <u>unseen</u> data.

Overfitting:

model mistakes noise for signal

Underfitting

model doesn't (can't) fit signal

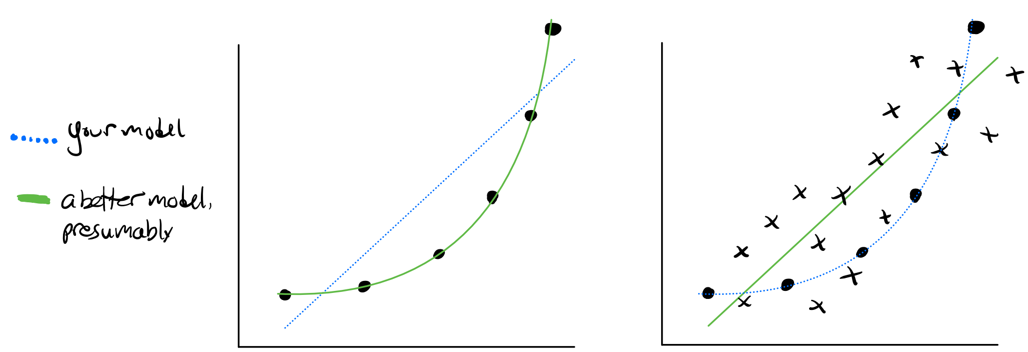


"Simplest possible explanation for the data"
needs to take into account <u>noise</u>/sampling error

Bias vs. Variance

<u>Bias:</u> modeling error - your model can't fit the underlying phenomenon

<u>Variance:</u> sampling error - your model mistakes noise for the underlying phenomenon



Ex.: Which is which?

Tools in the fight against overfitting

How can we fit a model and convince ourselves it isn't overfitting?

1. Data Splits

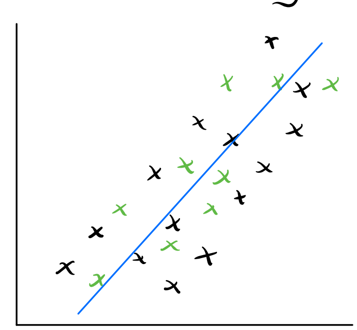
Idea: Hold back some training data

Train on ⊗, validate on ⊗

× training set
× Validation set

Scenario: accuracy is measured as distance from x to /

All available training data:



		Accuracy on validation set	
		Bad	Good
Accuracy on training set	Bad		
	Good		

Pseudocode for MLBot 1.0:

```
make modeling assumptions
While True:
    train
    validate
    if train == val == good:
        break
  else:
      revisit modeling assumptions
```

Problem?

Solution? Hold out another set of available data:

test set

Use only once to see how model does on truly unseen data.

Data split best practice:

Labeled data

Train	Val	Test

What %? Depends on size of data and amount of noise.

Smaller $\rightarrow$ higher variance

larger $\rightarrow$ less data for other splits

For small data, take full advantage of as much data as possible:

Val_1	Val_2	Val_3	...	Val_k	Test

"k-fold cross-validation":

train on each subset of k-1 chunks, val on the last

avg val accuracy across all k trials

\+ better training, lower-variance val accuracy

\- need to train k times

"leave-one-out cross-validation":

$k = n$

Tools in the fight against overfitting

2. Regularization: build a "simplicity prior" into your model.