foundations of data science for everyone

IV: Machine Learning & Linear Regression

dr.federica bianco | *fbb.space* |  *fedhere* |  *fedhere*

this slide deck: https://slides.com/federicabianco/fdsfe_04

what is machine learning?

what is a model?

the best way to think about it
in the ML context:

a model is a low
dimensional representation
of a higher dimensionality
datase

what is machine learning?

[Machine Learning is the] field of study that gives computers the ability to learn without being explicitly programmed.

Arthur Samuel, 1959

what is machine learning?

ML: any model with parameters learnt from the data

https://miro.medium.com/max/960/1*imhEKEpzX24CC_LIIureBw.gif

what is machine learning?

Machine Learning models are parametrized representation of "reality" where the parameters are learned from finite sets of realizations of that reality

(note: learning by instance, e.g. nearest neighbours, may not comply to this definition)

Machine Learning is the disciplines that conceptualizes, studies, and applies those models.

Key Concept

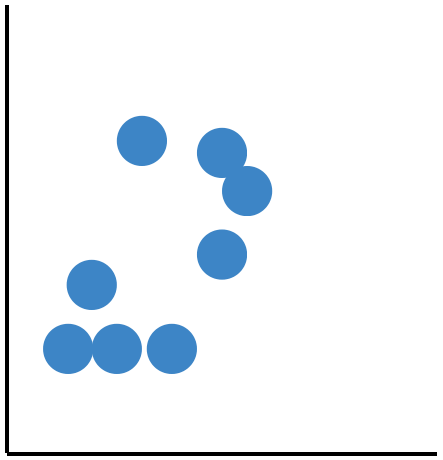
General ML points

used to:

- understand the structure of a feature space
- regression : predict values based on conditions *based on examples*
- classify *based on examples*
- understand which features are important for the success of the model (to get close to causality)

unsupervised vs supervised learning

understand the structure of a feature space



Clustering

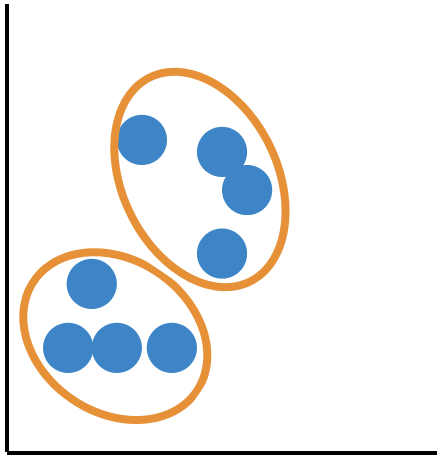
partitioning the feature space so that the existing data is grouped (according to some target function!)

unsupervised vs supervised learning

understand the structure of a feature space

Unsupervised learning

- understanding structure
- anomaly detection
- dimensionality reduction



Clustering

partitioning the feature space so that the existing data is grouped (according to some target function!)

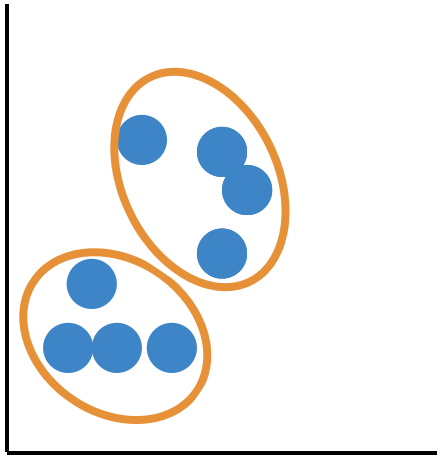
All features are observed for all datapoints

unsupervised vs supervised learning

prediction and classification based on examples

Unsupervised learning

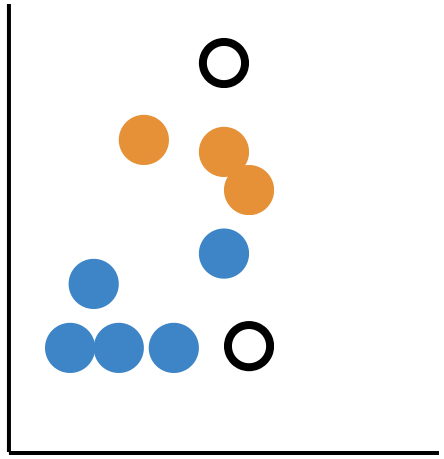
- understanding structure
- anomaly detection
- dimensionality reduction



Clustering

partitioning the feature space so that the existing data is grouped (according to some target function!)

All features are observed for all datapoints



Classifying & regression

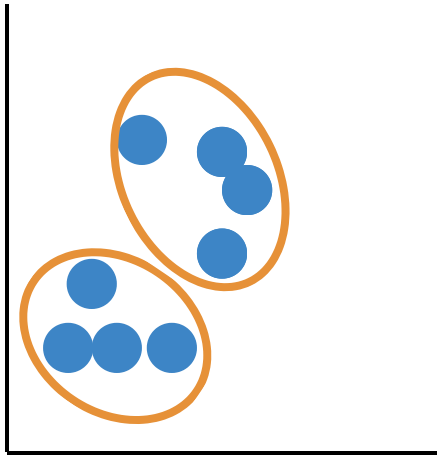
finding functions of the variables that allow to predict unobserved properties of new observations

unsupervised vs supervised learning

prediction and classification based on examples

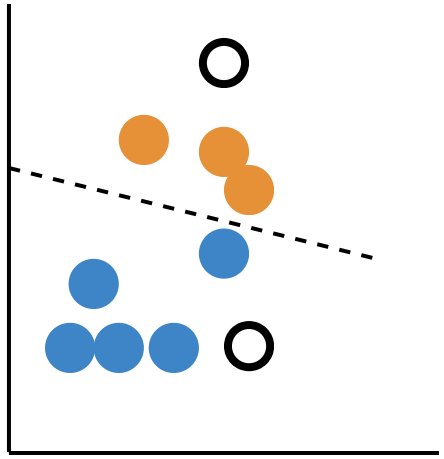
Unsupervised learning

- understanding structure
- anomaly detection
- dimensionality reduction



Clustering

partitioning the feature space so that the existing data is grouped (according to some target function!)



Classifying & regression

finding functions of the variables that allow to predict unobserved properties of new observations

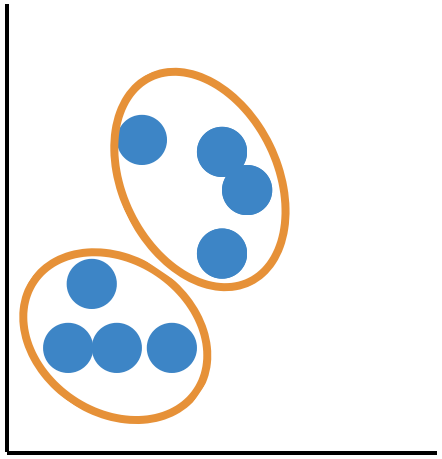
All features are observed for all datapoints

unsupervised vs supervised learning

prediction and classification based on examples

Unsupervised learning

- understanding structure
- anomaly detection
- dimensionality reduction



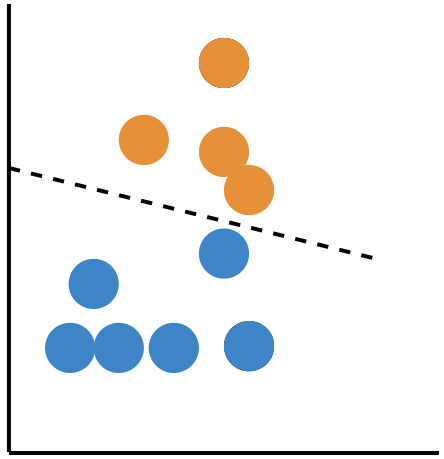
Clustering

partitioning the feature space so that the existing data is grouped (according to some target function!)

All features are observed for all datapoints

Supervised learning

- classification
- prediction
- feature selection



Classifying & regression

finding functions of the variables that allow to predict unobserved properties of new observations

Some features are not observed for some data points we want to predict them.

unsupervised vs supervised learning

Unsupervised learning

All features are observed for all datapoints

and we are looking for structure in the feature space

Supervised learning

Some features are not observed for some data points we want to predict them.

The datapoints for which the target feature is observed are said to be "*labeled*"

also...

Semi-supervised learning

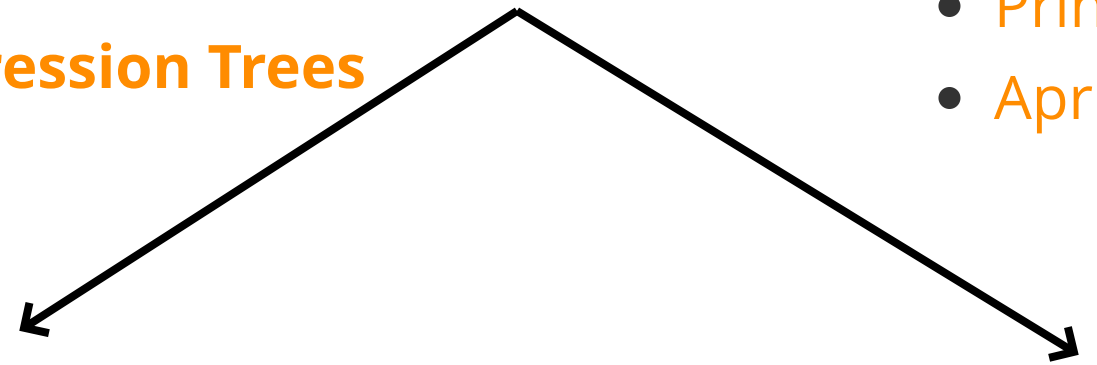
A small amount of labeled data is available. Data is cluster and clusters inherit labels

Active learning

The code can interact with the user to update labels and update model.

what is machine learning?

- k-Nearest Neighbors
- **Regression**
- Support Vector Machines
- **Classification/Regression Trees**
- **Neural networks**



- **clustering**
- Principle Component Analsysis
- Apriori (association rule)

supervised learning

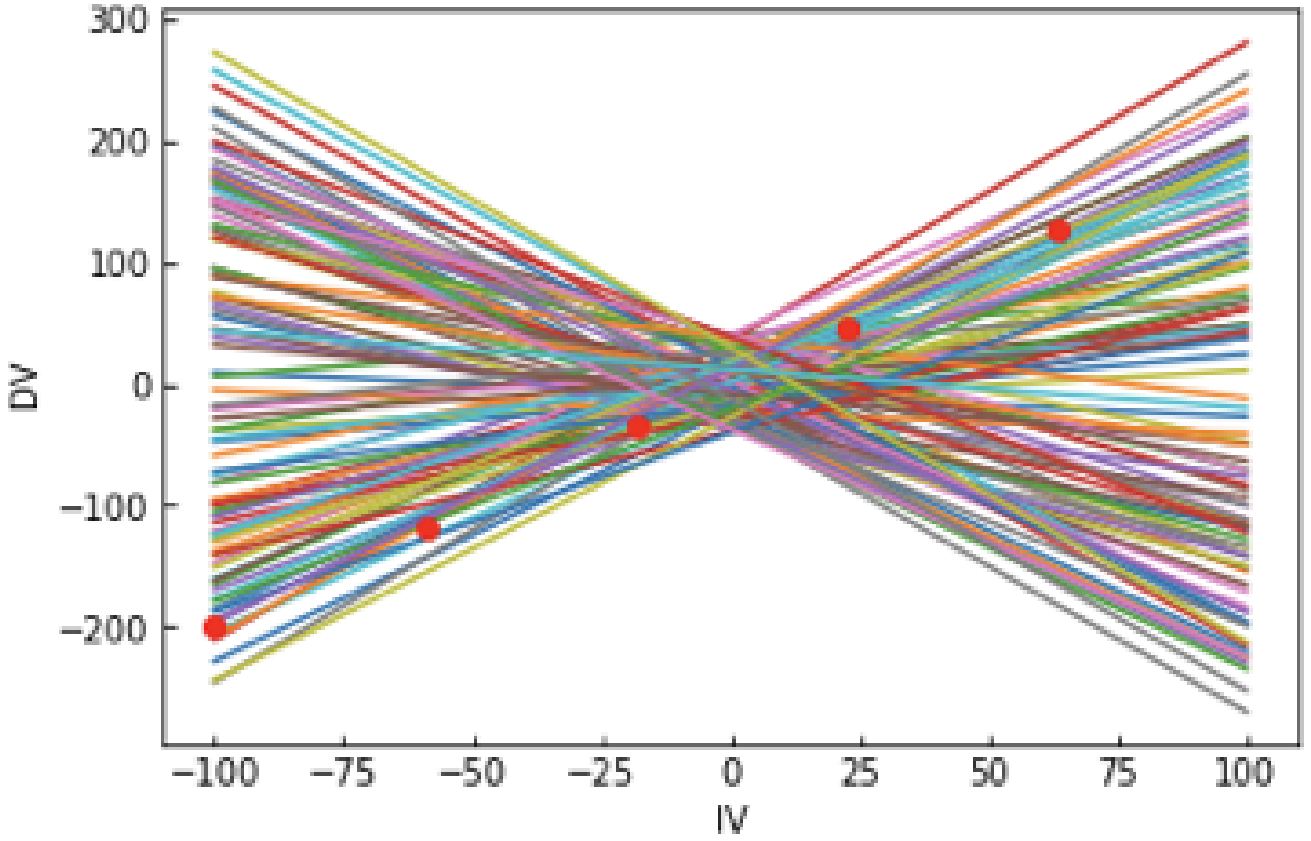
extract features and create models that allow prediction where the correct answer is known for a subset of the data

unsupervised learning

identify features and create models that allow to understand structure in the data

2 what is a model?

line model: $ax+b$



1

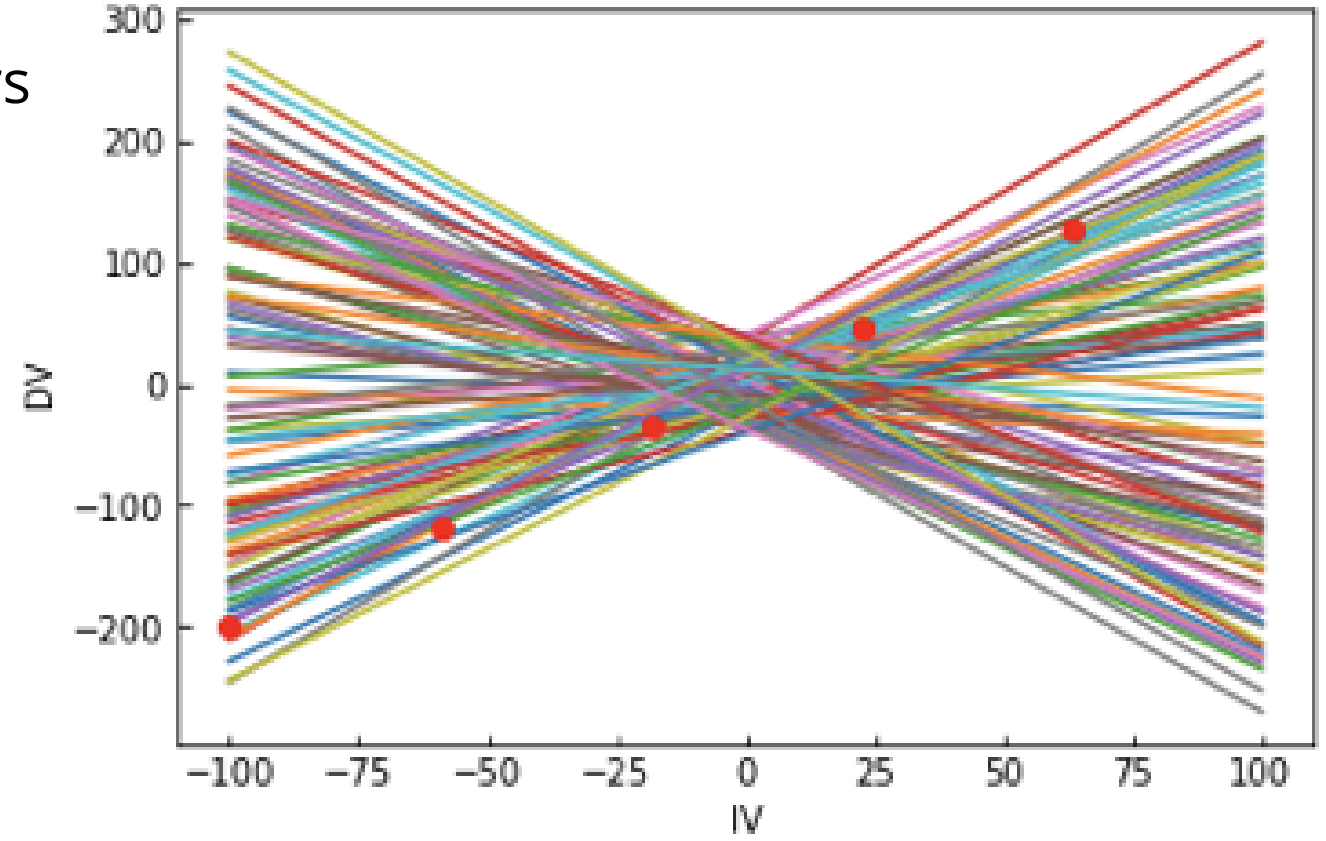
choose your model :

choose a mathematical formula to represent the behavior you see/expect in the data

line model: $ax+b$

a mathematical formula describes a family of shapes. The parameters define the exact shape: in a line fit the parameters are...

- a: slope
- b: intercept



1

choose your model :

choose a mathematical formula to represent the behavior you see/expect in the data

generalizatoin to a polynomial fit:

- the *degree N* of the polynomial is a *hyperparameter* of the model

we choose hyperparameters,

we fit parameters

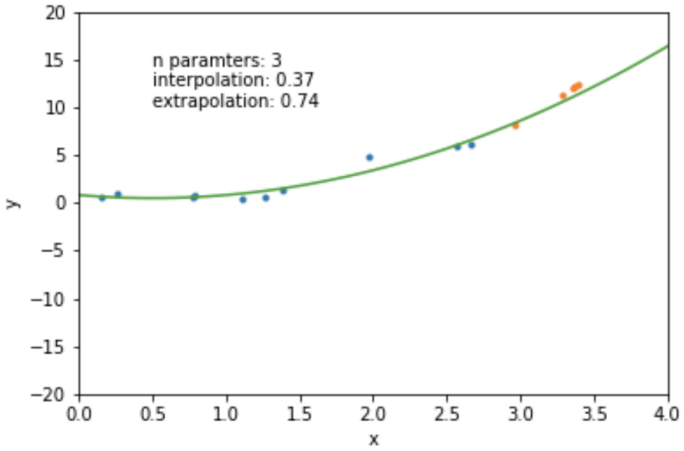
1

choose your model :

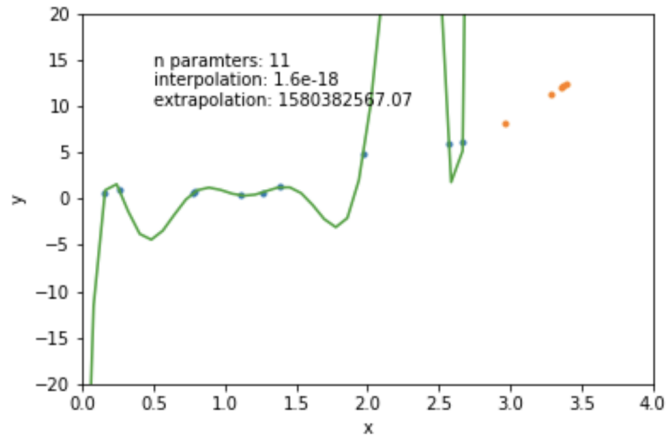
choose a mathematical formula to represent the behavior you see/expect in the data

polynomial model:

$$polyn = \sum_{i=0}^{N} c_i x^i$$



N=2: a conservative hyperparameter choice



N=12: goes exactly through each data point but it has too many parameters- N = number of data points

https://github.com/fedhere/PUS2022_FBianco/blob/master/classdemo/overfit_animation.ipynb

generalizatoin to a polynomial fit:

- the *degree N* of the polynomial is a *hyperparameter* of the model

we choose hyperparameters,

we fit parameters

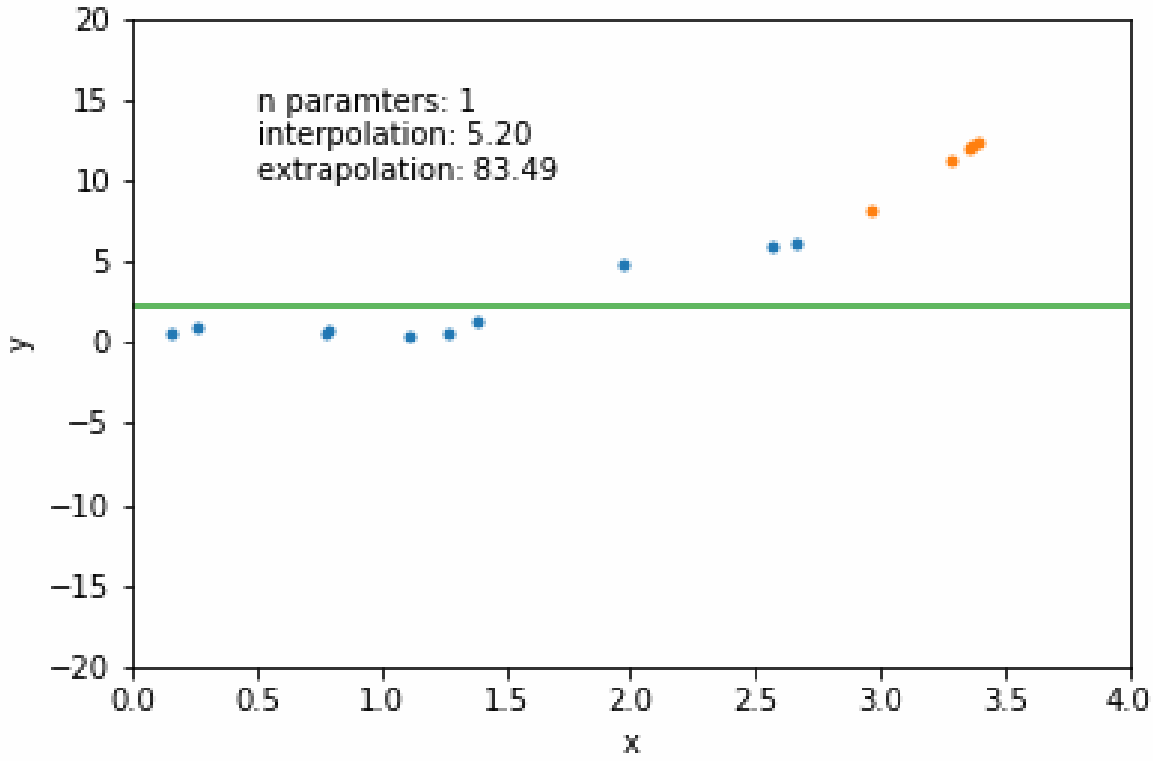
1

choose your model :

choose a mathematical formula to represent the behavior you see/expect in the data

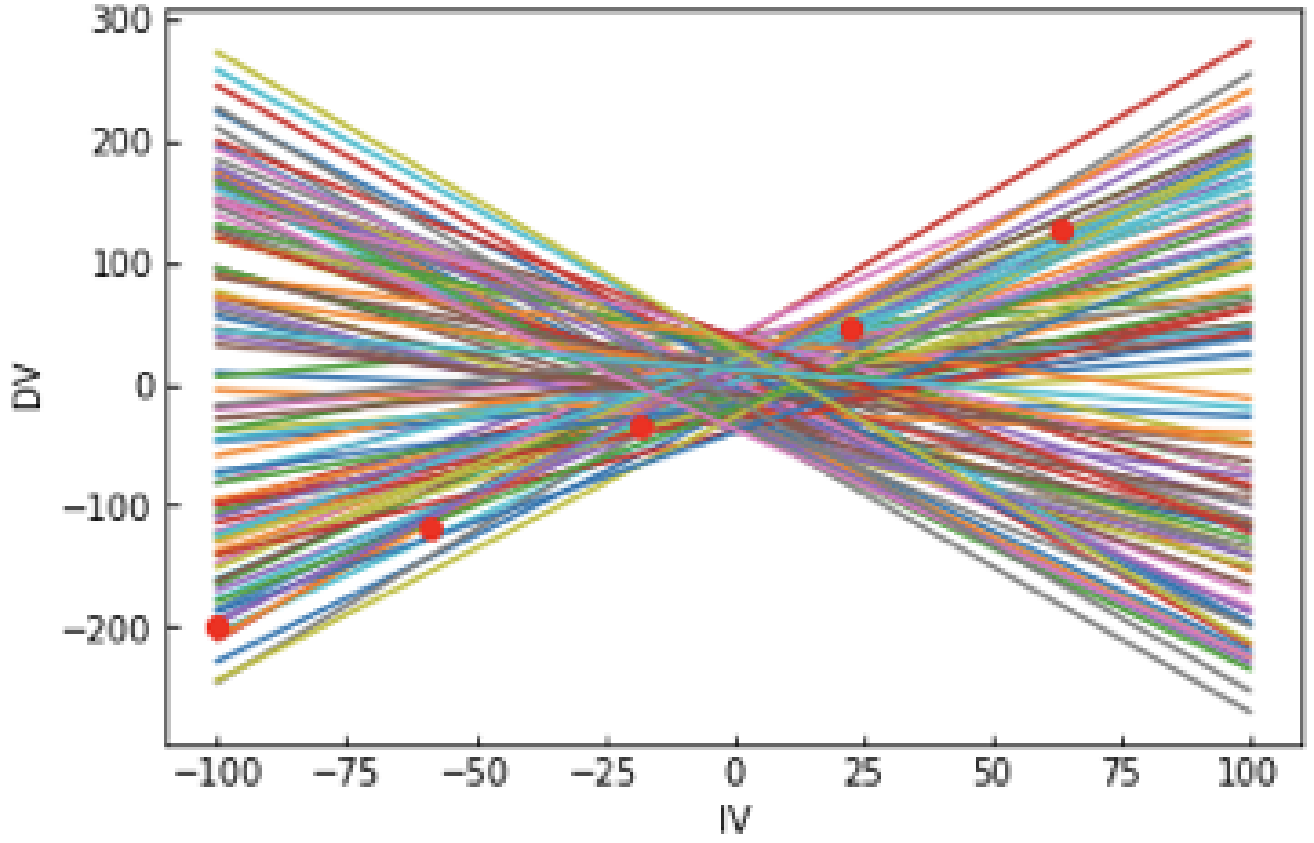
polynomial model:

$$polyn = \sum_{i=0}^{N} c_i x^i$$



https://github.com/fedhere/PUS2022_FBianco/blob/master/classdemo/overfit_animation.ipynb

line model: $ax+b$



2

choose an objective function :

you need a plan to choose the parameters of the model: to "optimize" the model.
You need to choose something to be MINIMIZED or MAXIMIZED

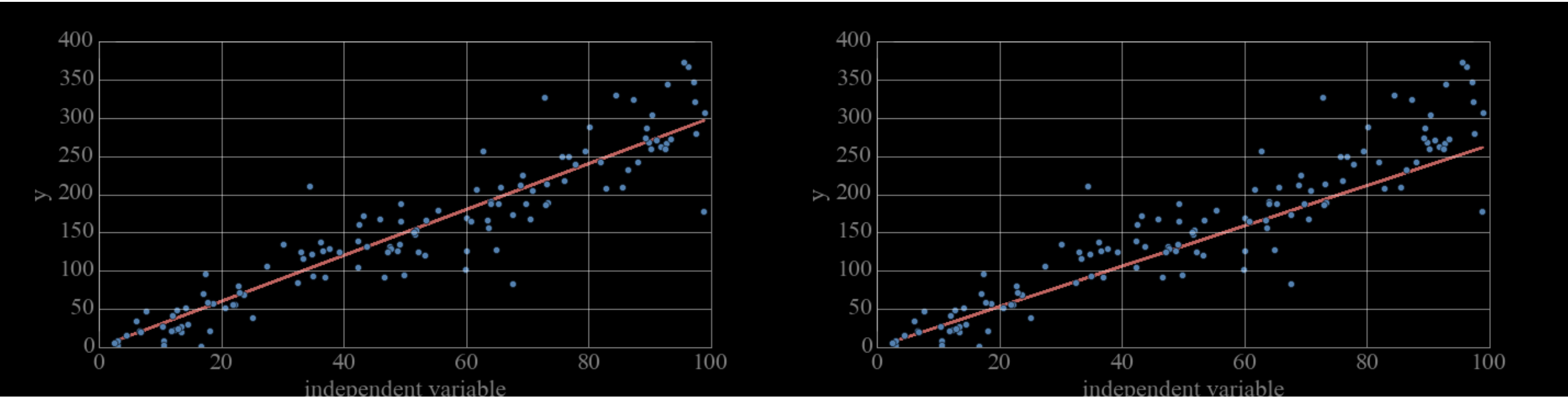
objective function:

what you want to optimize for

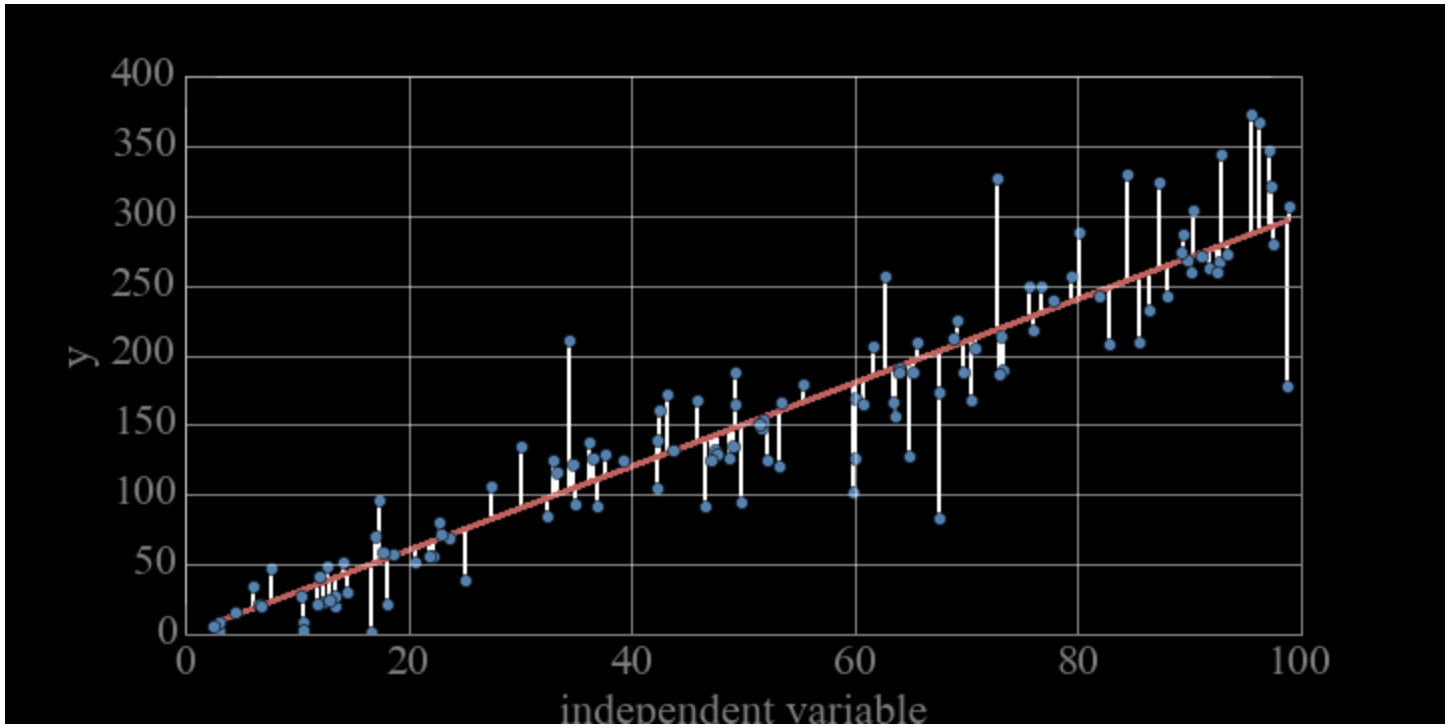
In principle, there are many choices for objective function. But the only procedure that is truly justified—in the sense that it leads to interpretable probabilistic inference, is to make a generative model for the data.

objective function:

what you want to optimize for



objective function:



objective function:

what you want to optimize for

e.g. Sum of residual squared (*least square fit method*)

$$SSE = \sum (y_i - (mx_i + b))^2$$

yi: i-th observation

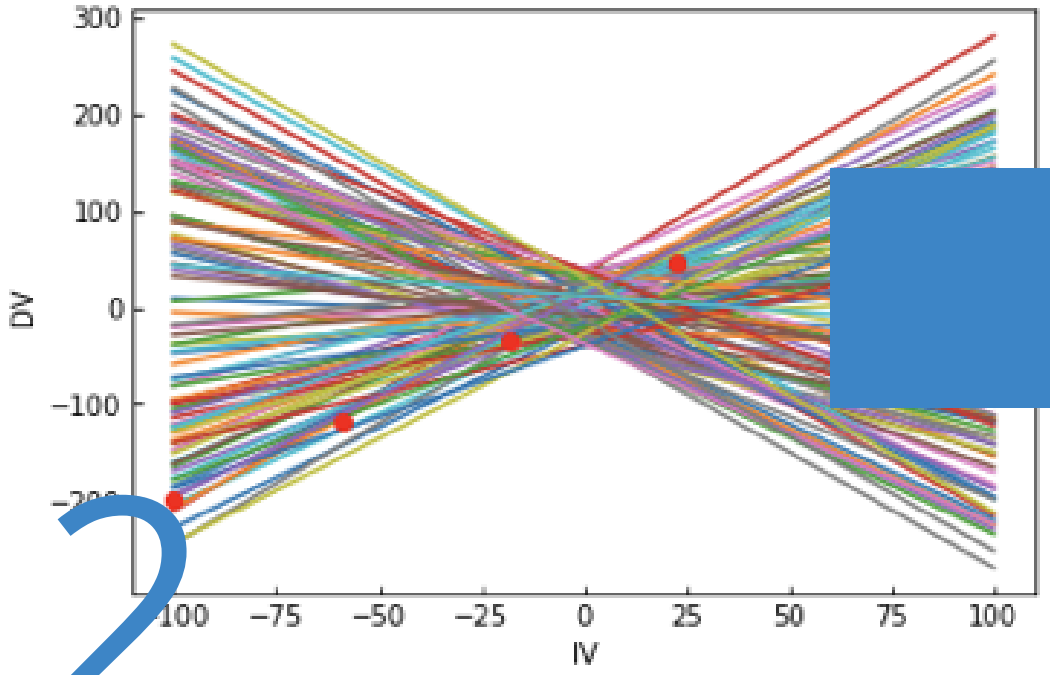
xi: i-th measurement "location"

$$SSE = \sum (y_{i,observed} - y_{i,predicted})^2$$

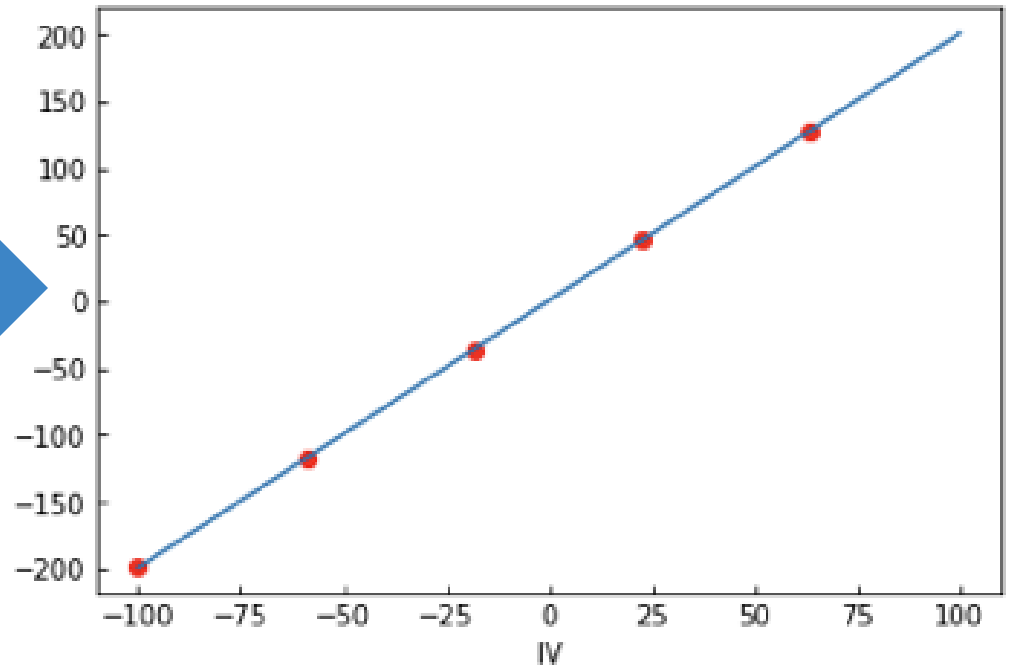
Fit model parameters <==> minimize the Sum of residuals squared

line model: *ax+b*

a line is a family of models



a line with set parameters is a models



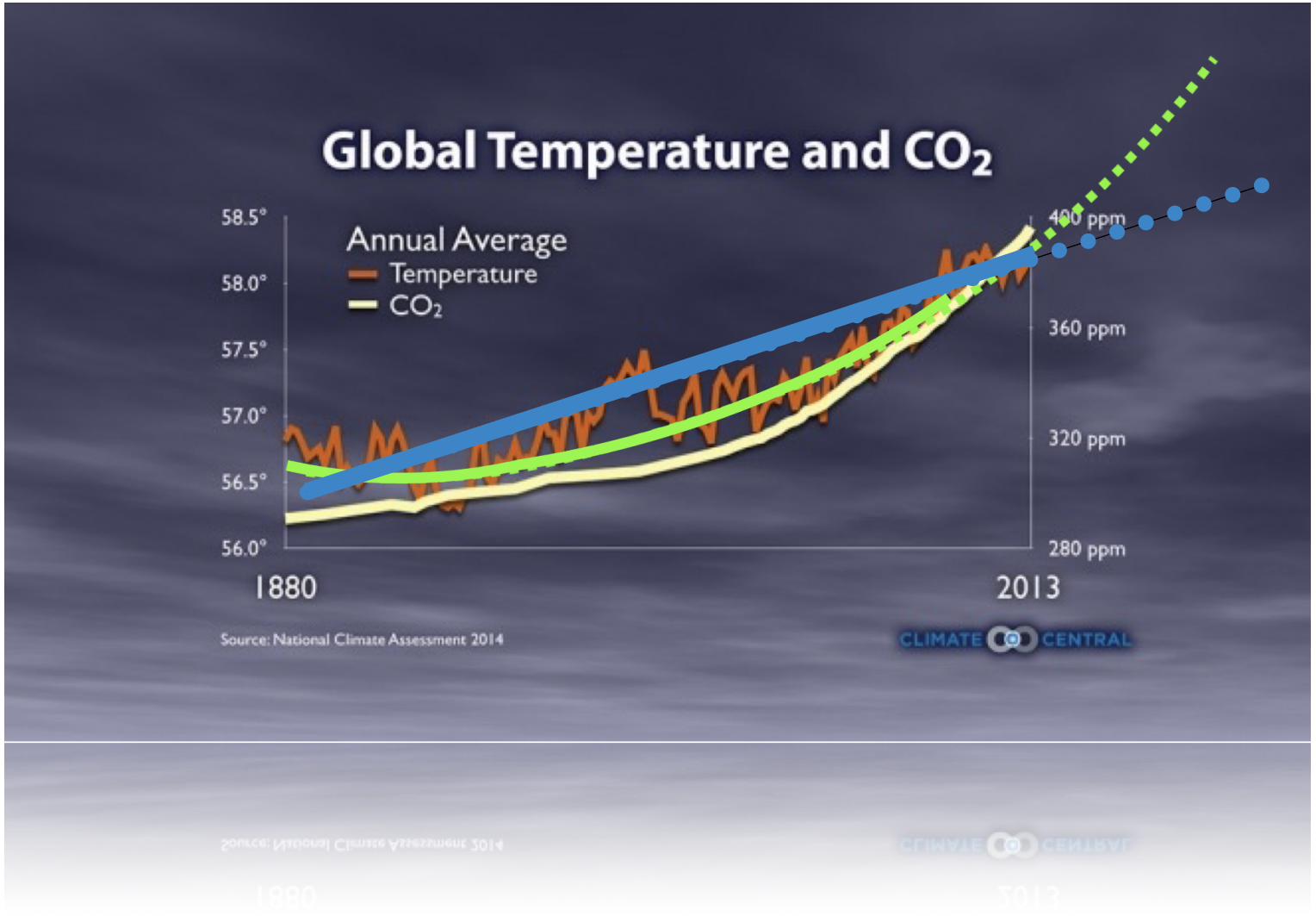
2

choose an objective function :

you need a plan to choose the parameters of the model: to "optimize" the model.
You need to choose something to be MINIMIZED or MAXIMIZED

21 why do we model?

why do we model?



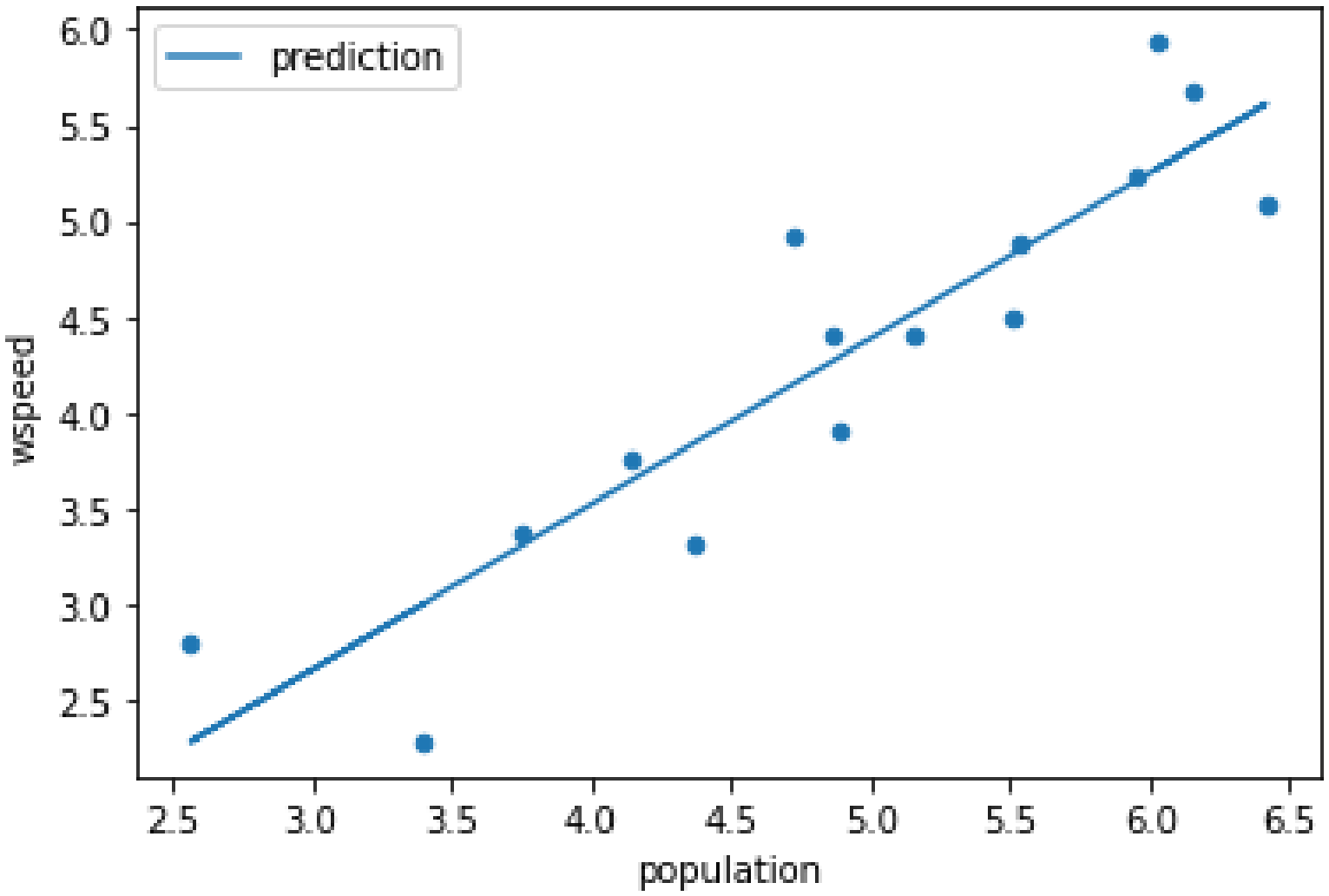
to explain

to predict

3 fitting a line to data (the longish story)

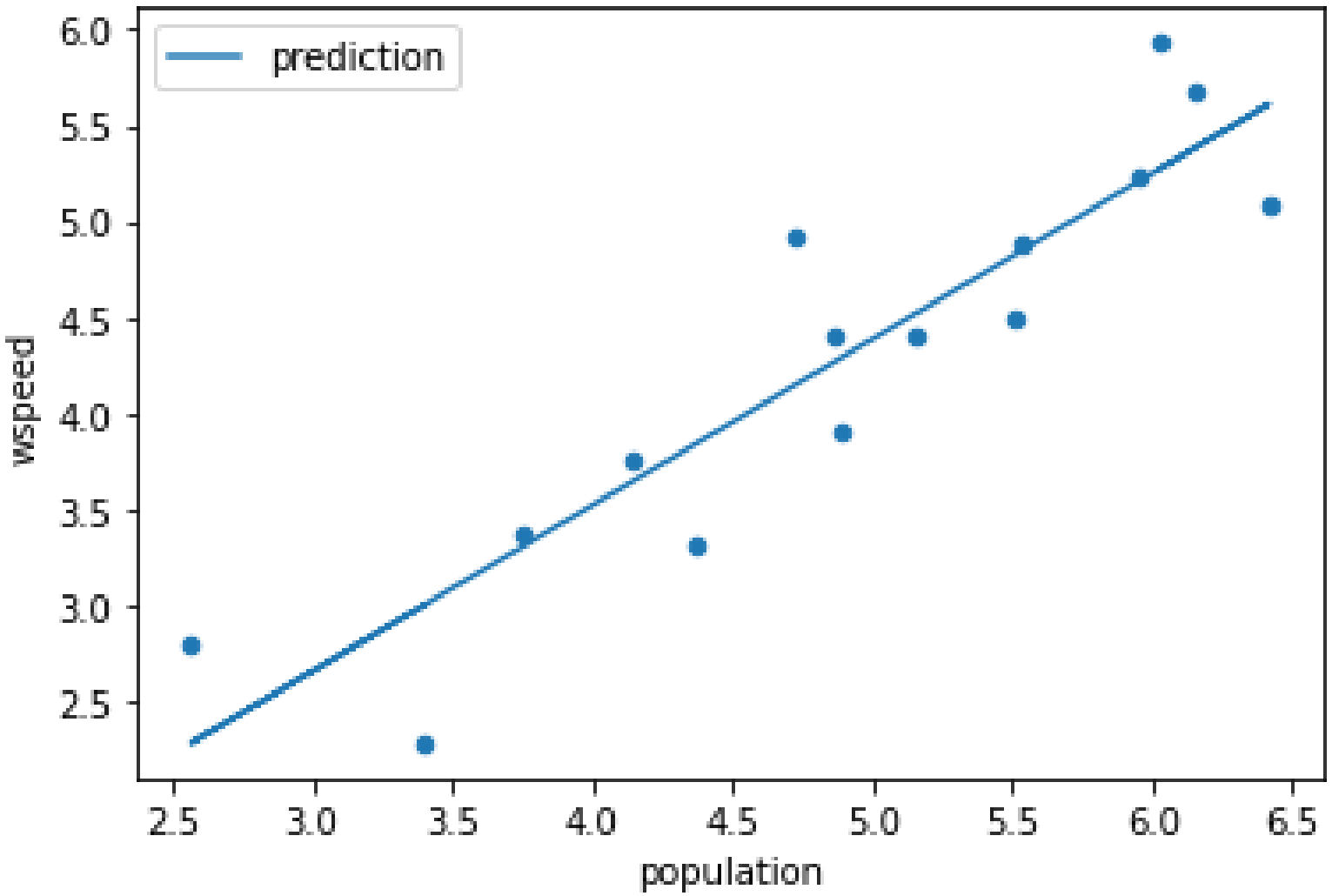
notebook ...

data
https://raw.githubusercontent.com/fedhere/PUS2022_FBianco/master/data/walkingspeed_Bettencourt07.csv

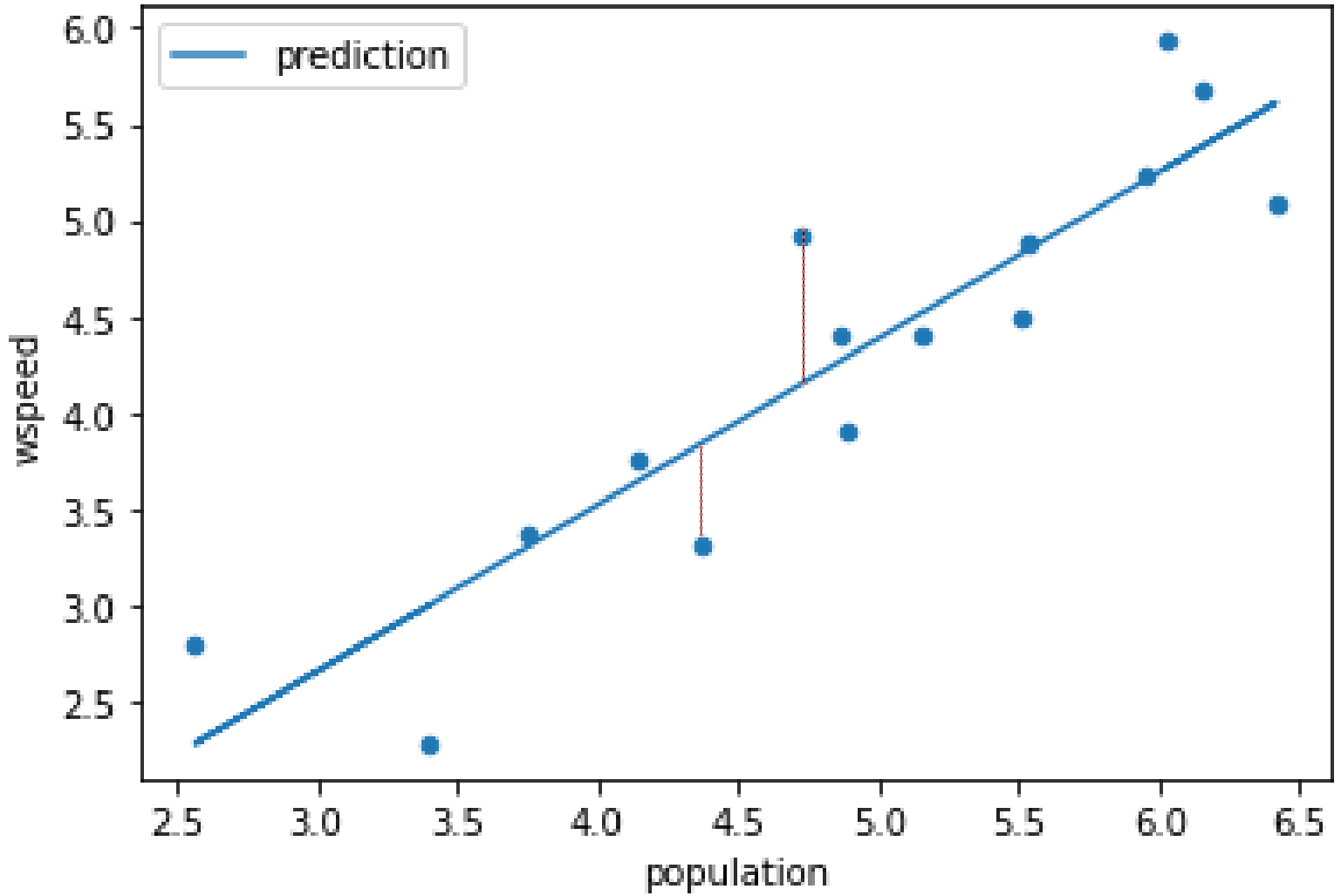


We are trying to find the "best" line that goes through the data... but "best" is a judgment call

Choosing the objective function



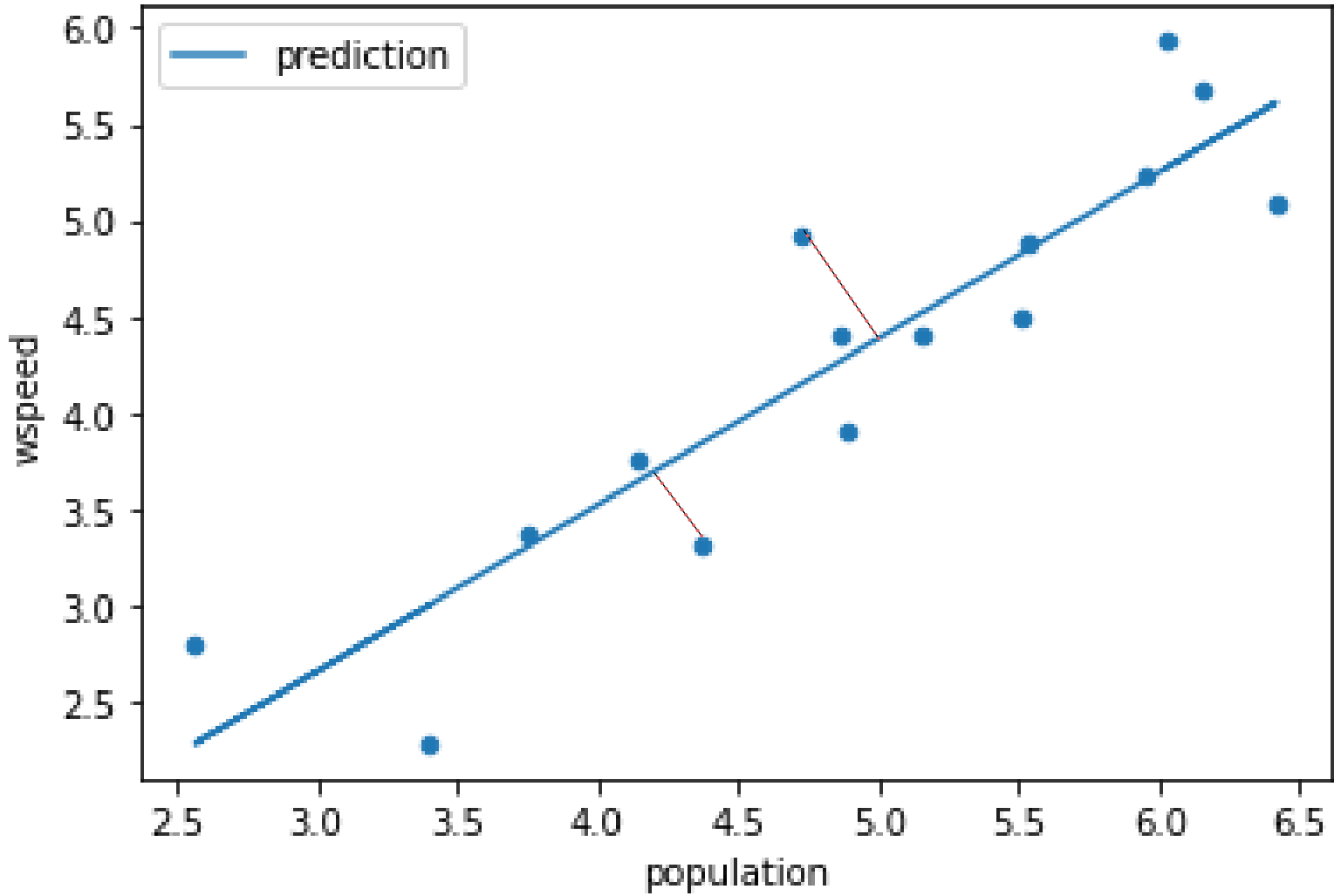
We are trying to find the "best" line that goes through the data... but "best" is a judgment call



We are trying to find the "best" line that goes through the data... but "best" is a judgment call

minimize *something*:

$$\text{Sum of errors} : \sum_{i=0}^{N} y_i - y_{i,predicted}$$



We are trying to find the "best" line that goes through the data... but "best" is a judgment call

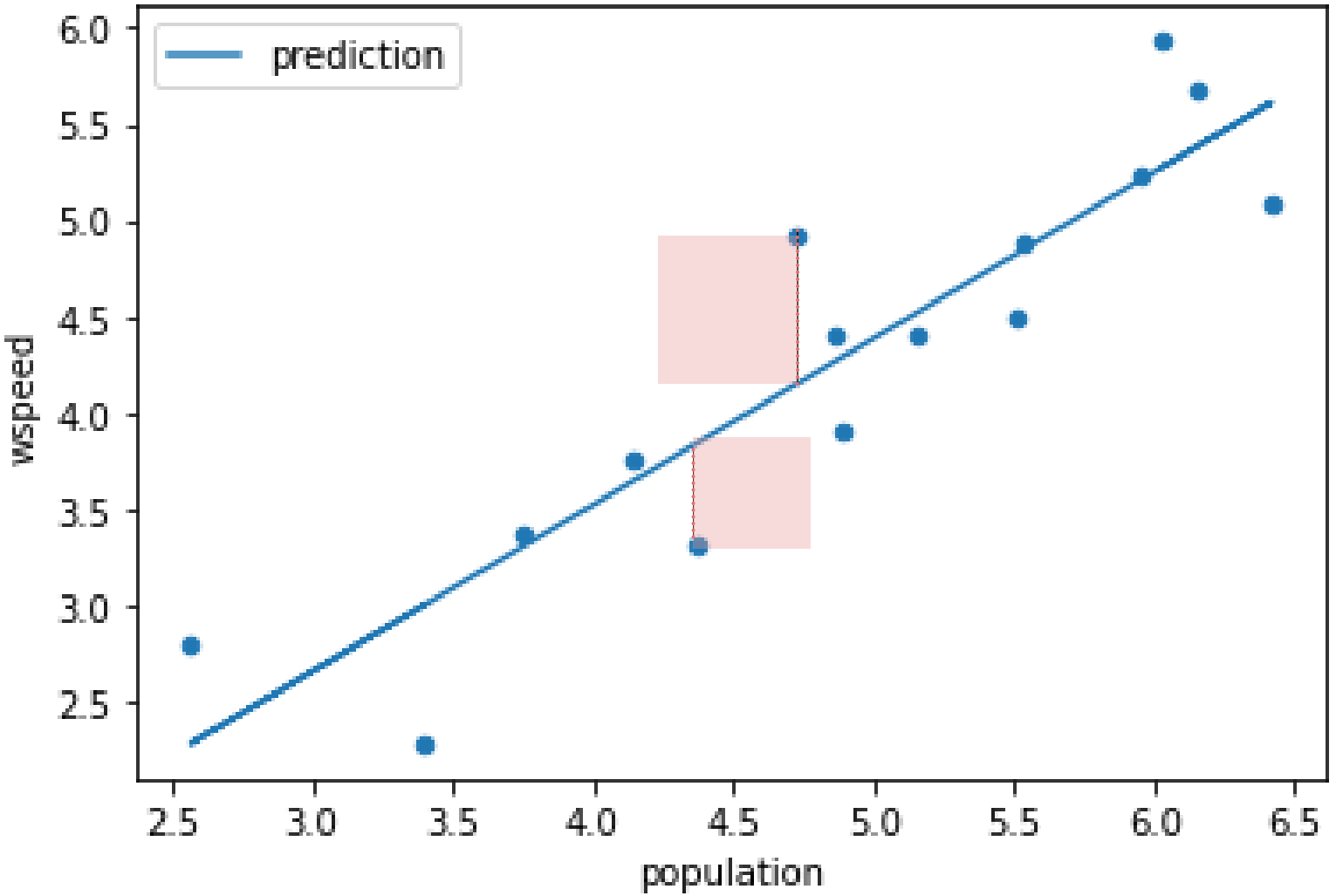
minimize *something*:

$$\text{Sum of errors} : \sum_{i=0}^{N} y_i - y_{i,predicted}$$

why square?

- So that the errors do not cancel themselves out

- To add more weight to predictions that are worse



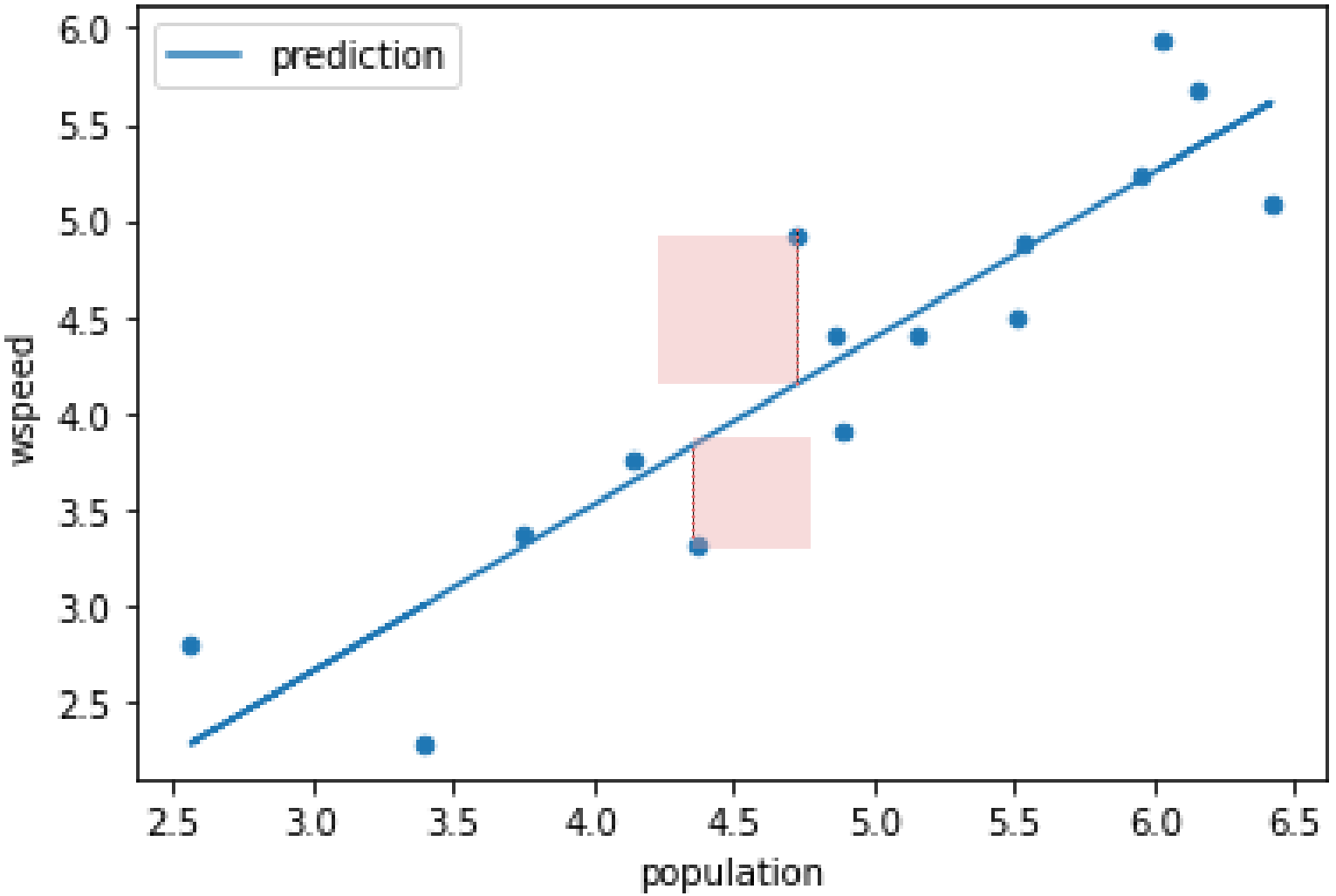
Ordinary least square

minimize *something*:

$$\text{Sum of quared errors} : SSE = \sum_{i=0}^{N} (y_i - y_{i,predicted})^2$$

we can minimize manually...

```
def sumsqerror(y, yp):
  ''' objective function squared error
  y: vector of observations
  yp: vector of predictions
  return: sum squared difference
  '''
  return ((y - yp) ** 2).sum()

minnow = 1e7
for s in np.arange(0, 3, 0.01):
  for i in np.arange(0, 2.5, 0.01):
    prediction = df['population'] * s + i
    sse = sumsqerror(df.wspeed, prediction)
    if sse < minnow:
      minnow = sse
```



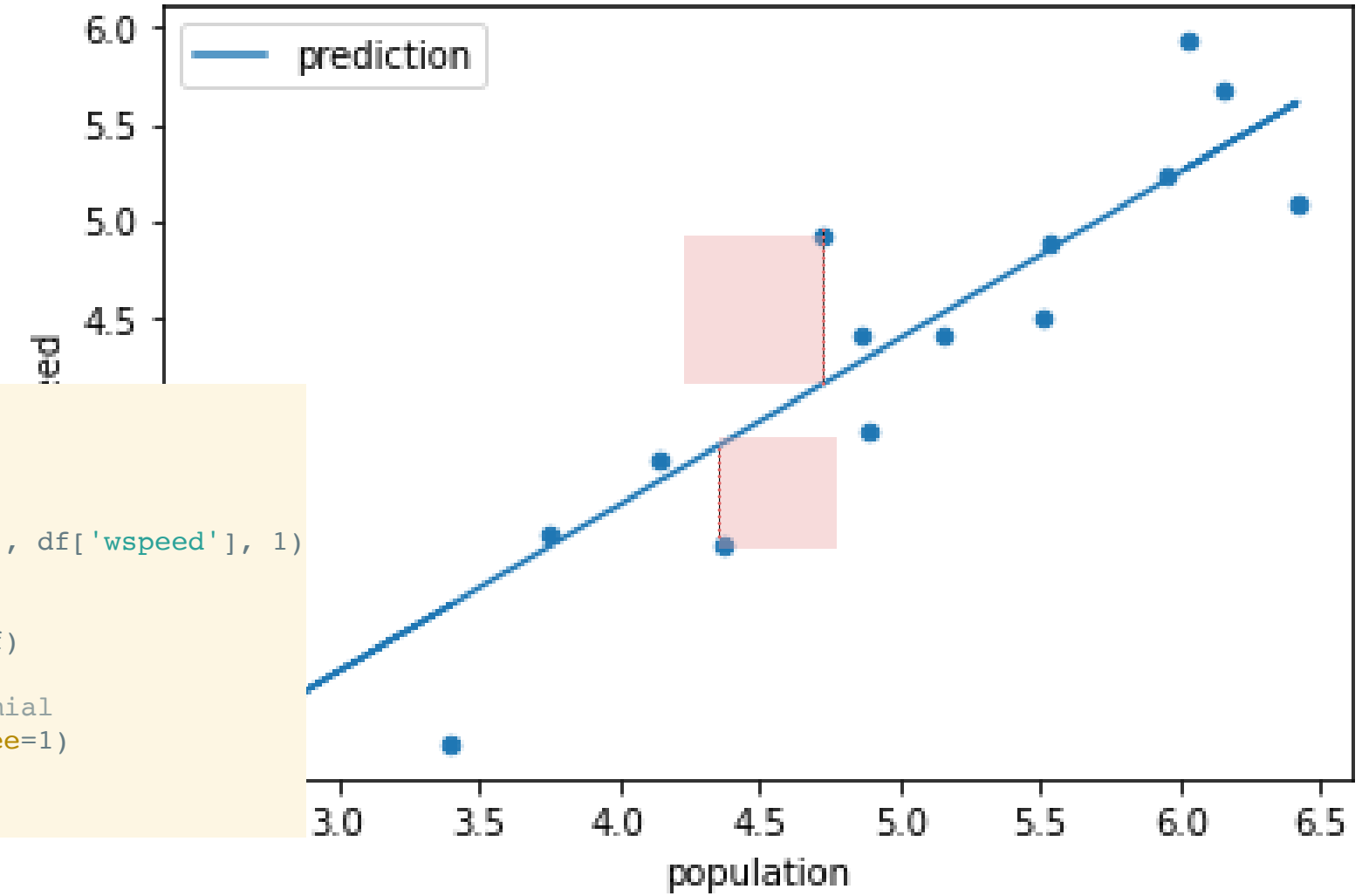
Ordinary least square

minimize *something*:

$$\text{Sum of quared errors} : SSE = \sum_{i=0}^{N} (y_i - y_{i,predicted})^2$$

but its a lot easier to use built in functions

```
# seaborn (just plots)
sns.regplot(df['population'], df['wspeed'])

# numpy
slope, intercept = np.polyfit(df['population'], df['wspeed'], 1)

# statsmodels formula
smf.ols(formula='wspeed ~ population', data=df)

# statsmodels OLS works for any degree polynomial
polynomial_features = PolynomialFeatures(degree=1)
xp1 = polynomial_features.fit_transform(x)
model = sm.OLS(y[:10], xp1[:10]).fit()
```



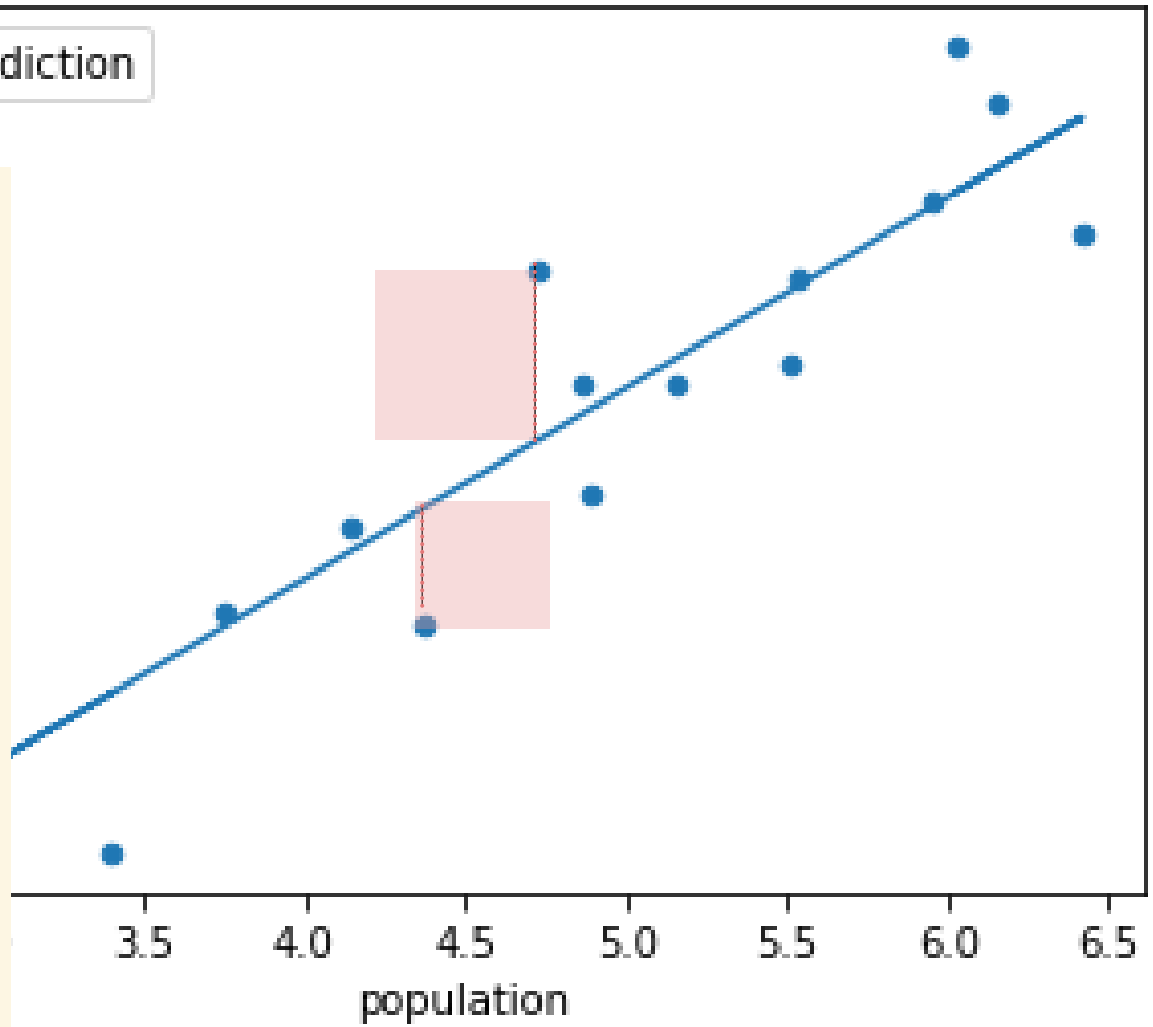
We are trying to find the "best" line that goes through the data... but "best" is a judgment call

minimize *something*:

$$\text{Sum of quared errors} : SSE = \sum_{i=0}^{N} (y_i - y_{i,predicted})^2$$

but its a lot easier to use built in functions

```python
# seaborn (just plots)
import seaborn as sns
sns.regplot(df['population'], df['wspeed'])

# numpy
import numpy as np
slope, intercept = np.polyfit(df['population'], df['wspeed'], 1)

# statsmodels formula
import statsmodels.formula.api as smf
smf.ols(formula='wspeed ~ population', data=df)

# statsmodels OLS works for any degree polynomial
from sklearn.preprocessing import PolynomialFeatures
polynomial_features = PolynomialFeatures(degree=1)
xp1 = polynomial_features.fit_transform(x)
model = sm.OLS(y[:10], xp1[:10]).fit()

# sklearn
from sklearn.linear_model import LinearRegression
lm = LinearRegression().fit(X_train, y_train, sample_weight=None)
```



ine that goes through the data... but "best" is a judgment call

minimize *something*:

$$\text{Sum of quared errors} : SSE = \sum_{i=0}^{N} (y_i - y_{i,predicted})^2$$

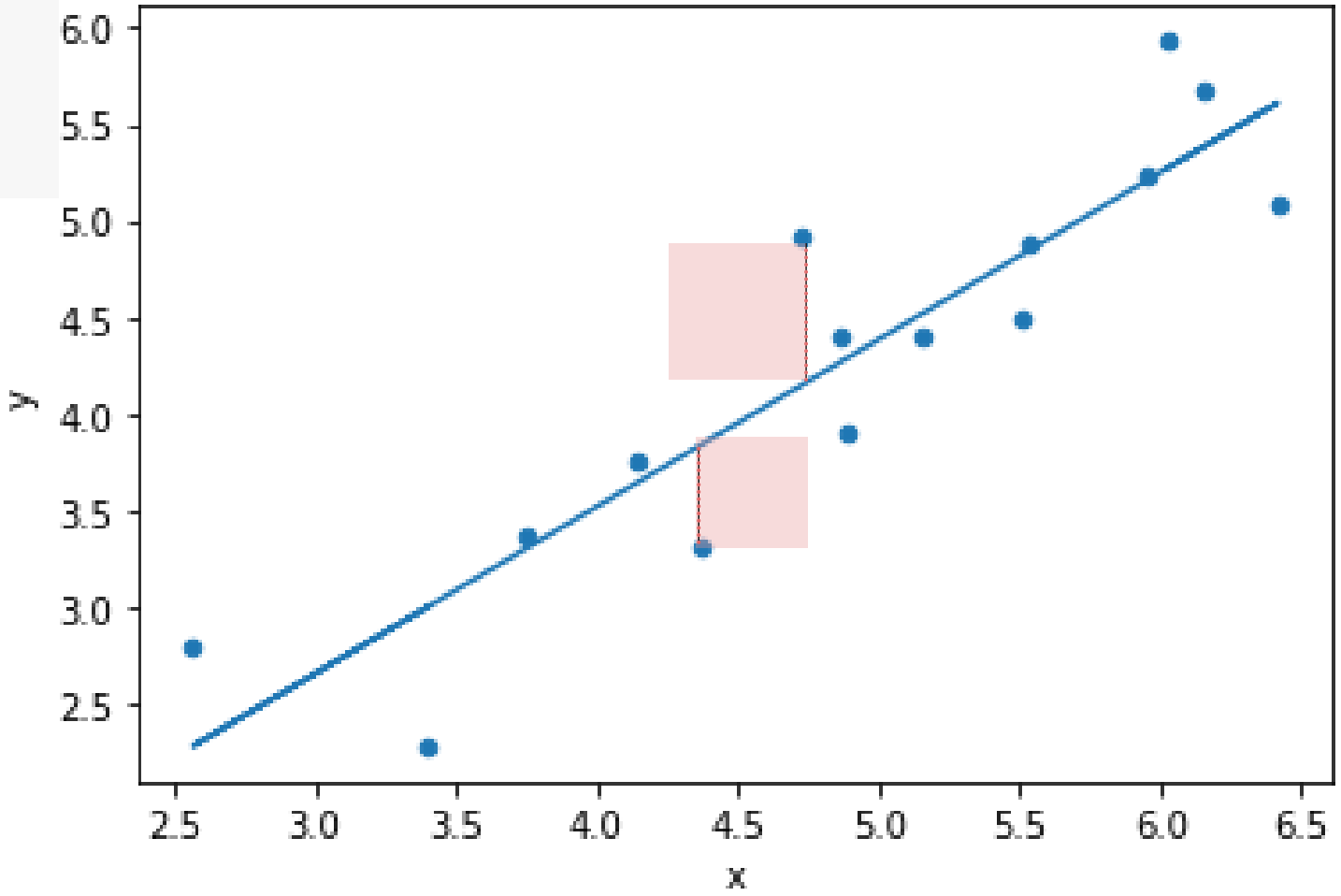
```
mod = smf.ols(formula='wspeed ~ population', data=df)

res = mod.fit()

res.summary()
```

OLS Regression Results

Dep. Variable:	wspeed	**R-squared:**	0.822
Model:	OLS	**Adj. R-squared:**	0.808
Method:	Least Squares	**F-statistic:**	59.90
Date:	Wed, 30 Sep 2020	**Prob (F-statistic):**	3.21e-06
Time:	13:06:28	**Log-Likelihood:**	-8.5829
No. Observations:	15	**AIC:**	21.17
Df Residuals:	13	**BIC:**	22.58
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.0566	0.560	0.101	0.921	-1.154	1.267
population	0.8653	0.112	7.740	0.000	0.624	1.107

Omnibus:	0.456	**Durbin-Watson:**	2.120
Prob(Omnibus):	0.796	**Jarque-Bera (JB):**	0.531
Skew:	0.115	**Prob(JB):**	0.767
Kurtosis:	2.107	**Cond. No.**	24.5



how good is the model?

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y}_i)^2}$$

← SSE ($\hat{y}_i$ prediction)

← variance ($\bar{y}_i$ mean)

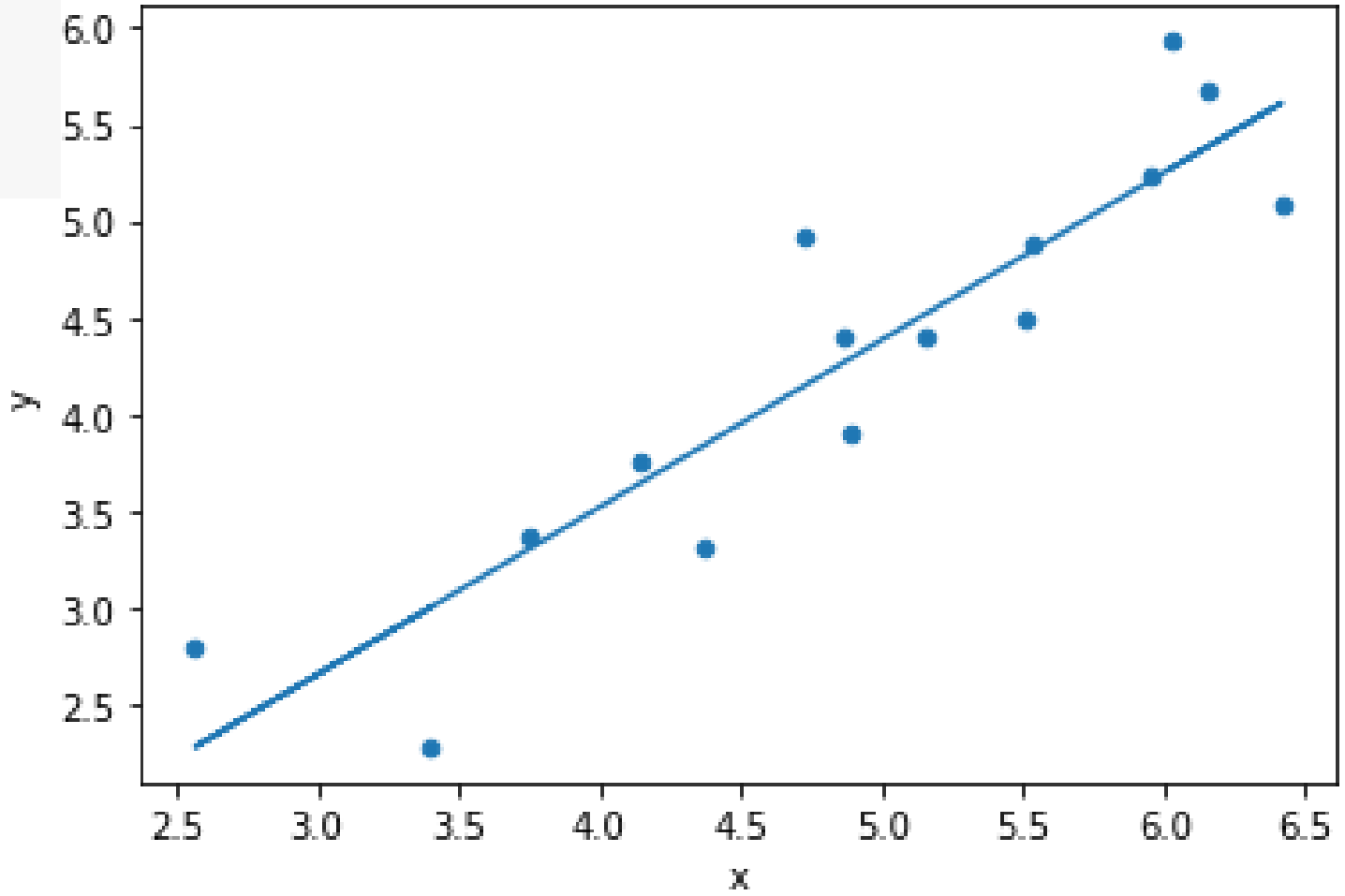
```
mod = smf.ols(formula='wspeed ~ population', data=df)

res = mod.fit()

res.summary()
```

OLS Regression Results

Dep. Variable:	wspeed	**R-squared:**	0.822
Model:	OLS	**Adj. R-squared:**	0.808
Method:	Least Squares	**F-statistic:**	59.90
Date:	Wed, 30 Sep 2020	**Prob (F-statistic):**	3.21e-06
Time:	13:06:28	**Log-Likelihood:**	-8.5829
No. Observations:	15	**AIC:**	21.17
Df Residuals:	13	**BIC:**	22.58
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.0566	0.560	0.101	0.921	-1.154	1.267
population	0.8653	0.112	7.740	0.000	0.624	1.107

Omnibus:	0.456	**Durbin-Watson:**	2.120
Prob(Omnibus):	0.796	**Jarque-Bera (JB):**	0.531
Skew:	0.115	**Prob(JB):**	0.767
Kurtosis:	2.107	**Cond. No.**	24.5



how good is the model?

$$R^2\,adjusted = 1 - \frac{(1 - R^2)(N - 1)}{N - p - 1}$$

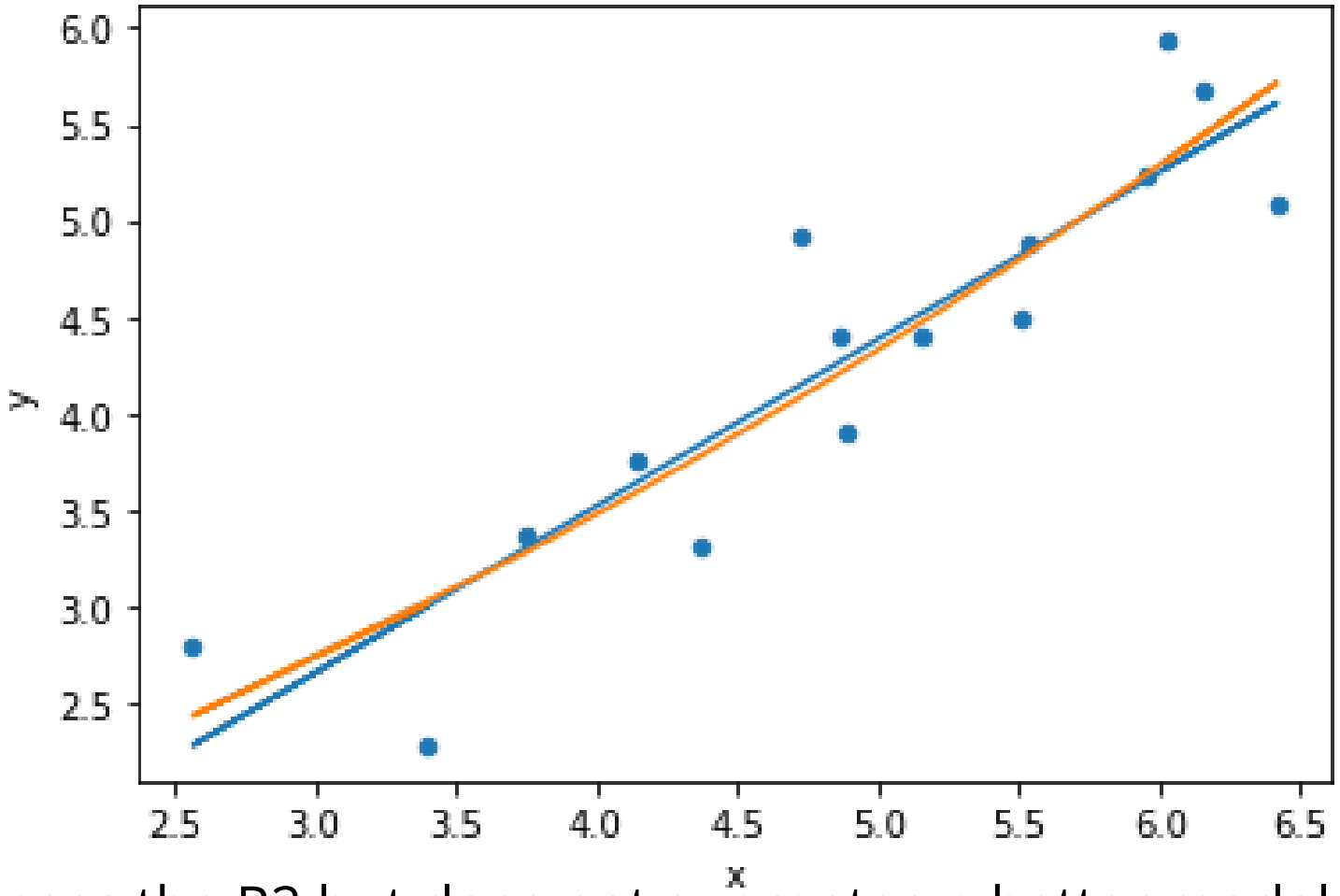
Adjusted R2 takes into account how many *datapoints* you have and how many *paramters* you have

```
mod = smf.ols(formula='wspeed ~ I(population**2) + population',
              data=df)
res = mod.fit()

res.summary()
```

Dep. Variable:	wspeed	**R-squared:**	0.826
Model:	OLS	**Adj. R-squared:**	0.797
Method:	Least Squares	**F-statistic:**	28.41
Date:	Wed, 30 Sep 2020	**Prob (F-statistic):**	2.81e-05
Time:	13:19:50	**Log-Likelihood:**	-8.4158
No. Observations:	15	**AIC:**	22.83
Df Residuals:	12	**BIC:**	24.96
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.0889	2.067	0.527	0.608	-3.415	5.593
I(population ** 2)	0.0513	0.099	0.520	0.613	-0.164	0.266
population	0.3916	0.918	0.426	0.677	-1.609	2.392

Omnibus:	0.026	**Durbin-Watson:**	2.071
Prob(Omnibus):	0.987	**Jarque-Bera (JB):**	0.238
Skew:	0.051	**Prob(JB):**	0.888
Kurtosis:	2.392	**Cond. No.**	507.



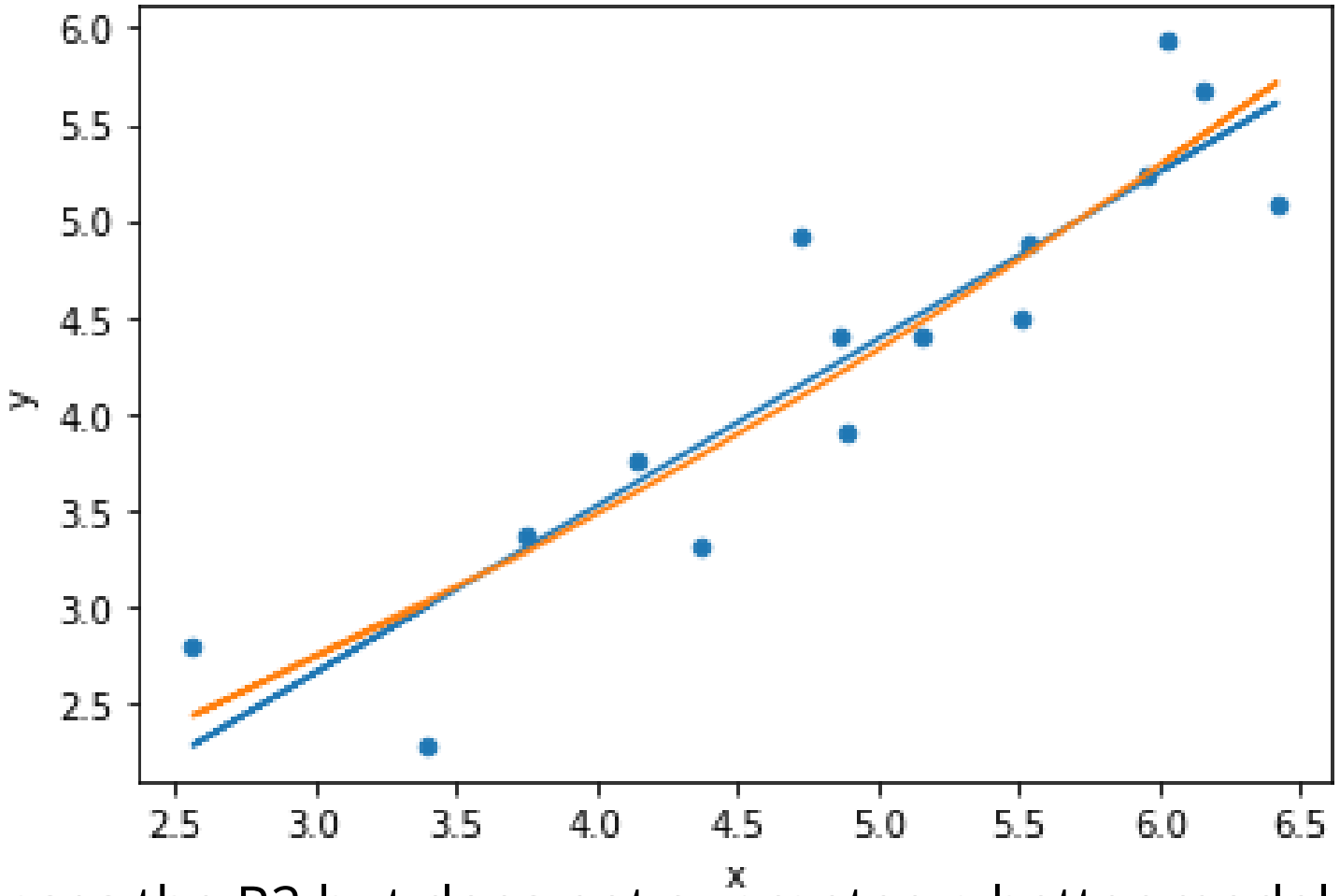
- increasing the model complexity increases the R2 but does not guarantee a better model
- the likelihood ratio is a way to assess if the more complex model is better in a NHRT sense

```
mod = smf.ols(formula='wspeed ~ I(population**2) + population',
              data=df)
res = mod.fit()

res.summary()
```

Dep. Variable:	wspeed	**R-squared:**	0.826
Model:	OLS	**Adj. R-squared:**	0.797
Method:	Least Squares	**F-statistic:**	28.41
Date:	Wed, 30 Sep 2020	**Prob (F-statistic):**	2.81e-05
Time:	13:19:50	**Log-Likelihood:**	-8.4158
No. Observations:	15	**AIC:**	22.83
Df Residuals:	12	**BIC:**	24.96
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.0889	2.067	0.527	0.608	-3.415	5.593
I(population ** 2)	0.0513	0.099	0.520	0.613	-0.164	0.266
population	0.3916	0.918	0.426	0.677	-1.609	2.392

Omnibus:	0.026	**Durbin-Watson:**	2.071
Prob(Omnibus):	0.987	**Jarque-Bera (JB):**	0.238
Skew:	0.051	**Prob(JB):**	0.888
Kurtosis:	2.392	**Cond. No.**	507.



- increasing the model complexity increases the R2 but does not guarantee a better model
- the likelihood ratio is a way to assess if the more complex model is better in a NHRT sense

NH: the more complex model is better

under the NH the LR statistics is distributed like a chi^2 distribution with number of dof = number of extra parameters

p-value of LR in a chi^2 distribution with dof

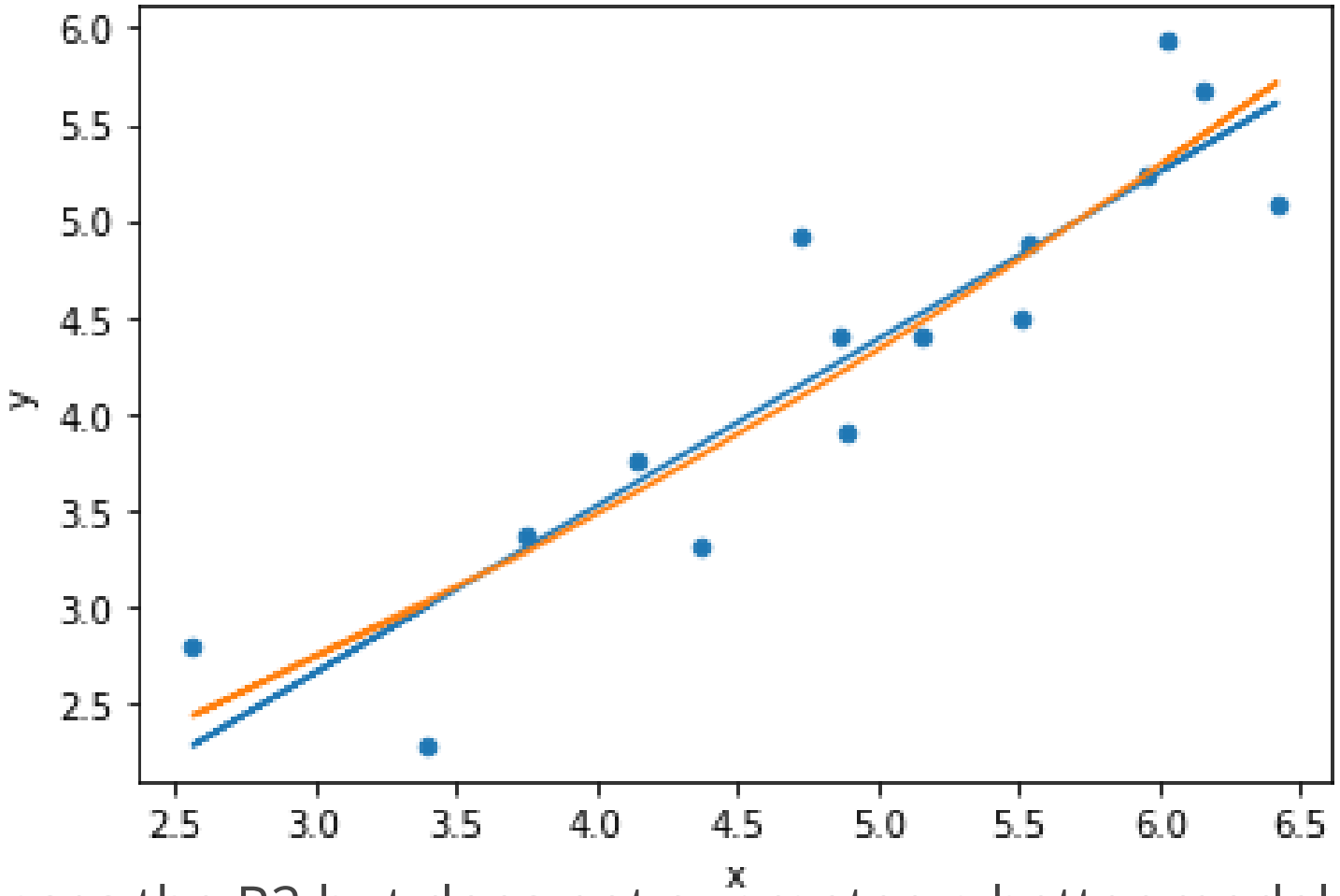
```
res2.compare_lr_test(res)
(0.3342540144461843, 0.5631648421666332, 1.0)
```

```
mod = smf.ols(formula='wspeed ~ I(population**2) + population',
              data=df)
res = mod.fit()

res.summary()
```

Dep. Variable:	wspeed	**R-squared:**	0.826
Model:	OLS	**Adj. R-squared:**	0.797
Method:	Least Squares	**F-statistic:**	28.41
Date:	Wed, 30 Sep 2020	**Prob (F-statistic):**	2.81e-05
Time:	13:19:50	**Log-Likelihood:**	-8.4158
No. Observations:	15	**AIC:**	22.83
Df Residuals:	12	**BIC:**	24.96
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.0889	2.067	0.527	0.608	-3.415	5.593
I(population ** 2)	0.0513	0.099	0.520	0.613	-0.164	0.266
population	0.3916	0.918	0.426	0.677	-1.609	2.392

Omnibus:	0.026	**Durbin-Watson:**	2.071
Prob(Omnibus):	0.987	**Jarque-Bera (JB):**	0.238
Skew:	0.051	**Prob(JB):**	0.888
Kurtosis:	2.392	**Cond. No.**	507.



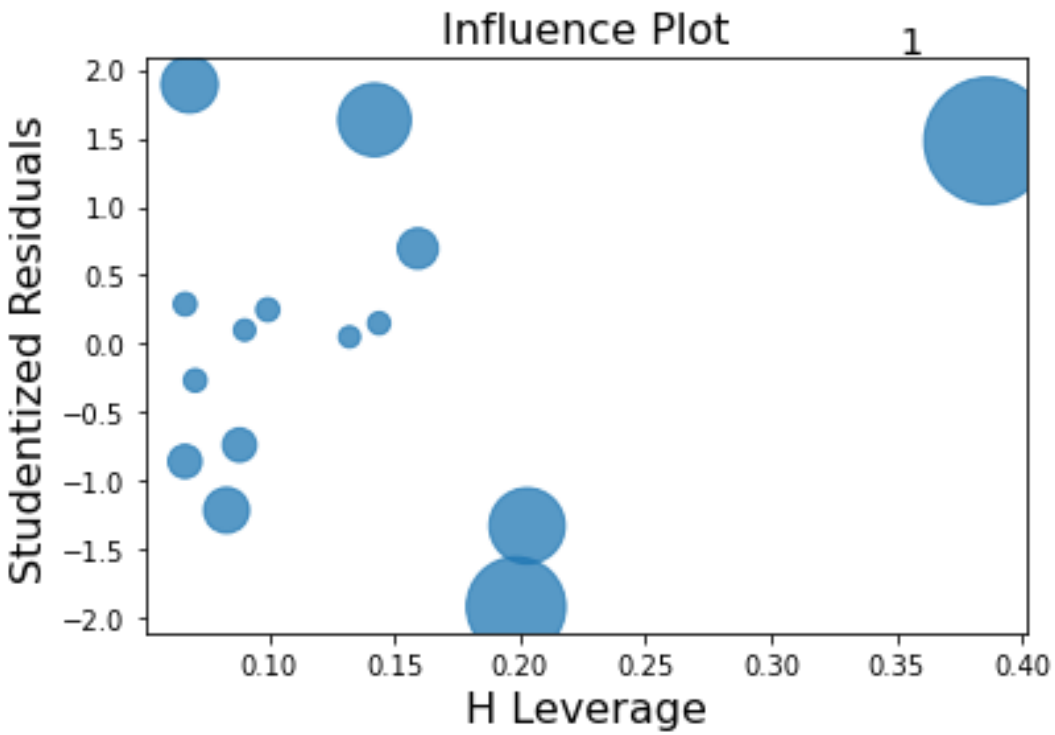
- increasing the model complexity increases the R2 but does not guarantee a better model
- the likelihood ratio is a way to assess if the more complex model is better in a NHRT sense

```
likelihood ration test - set alpha to alpha=0.05

 the NH is that the more complex model is better than the simpler one

 the likelihood ration statistics is 0.33, which corresponds to a p-value of 0.56

since the likelishood ration statistics is chi square distributed
with DoF the difference in the number of parameters in the 2 models (=1 here)
this corresponds not being able to reject the NH at alpha 0.05
```

```
res2.compare_lr_test(res)
```

```
(0.3342540144461843, 0.5631648421666332, 1.0)
```

not all points are equal!

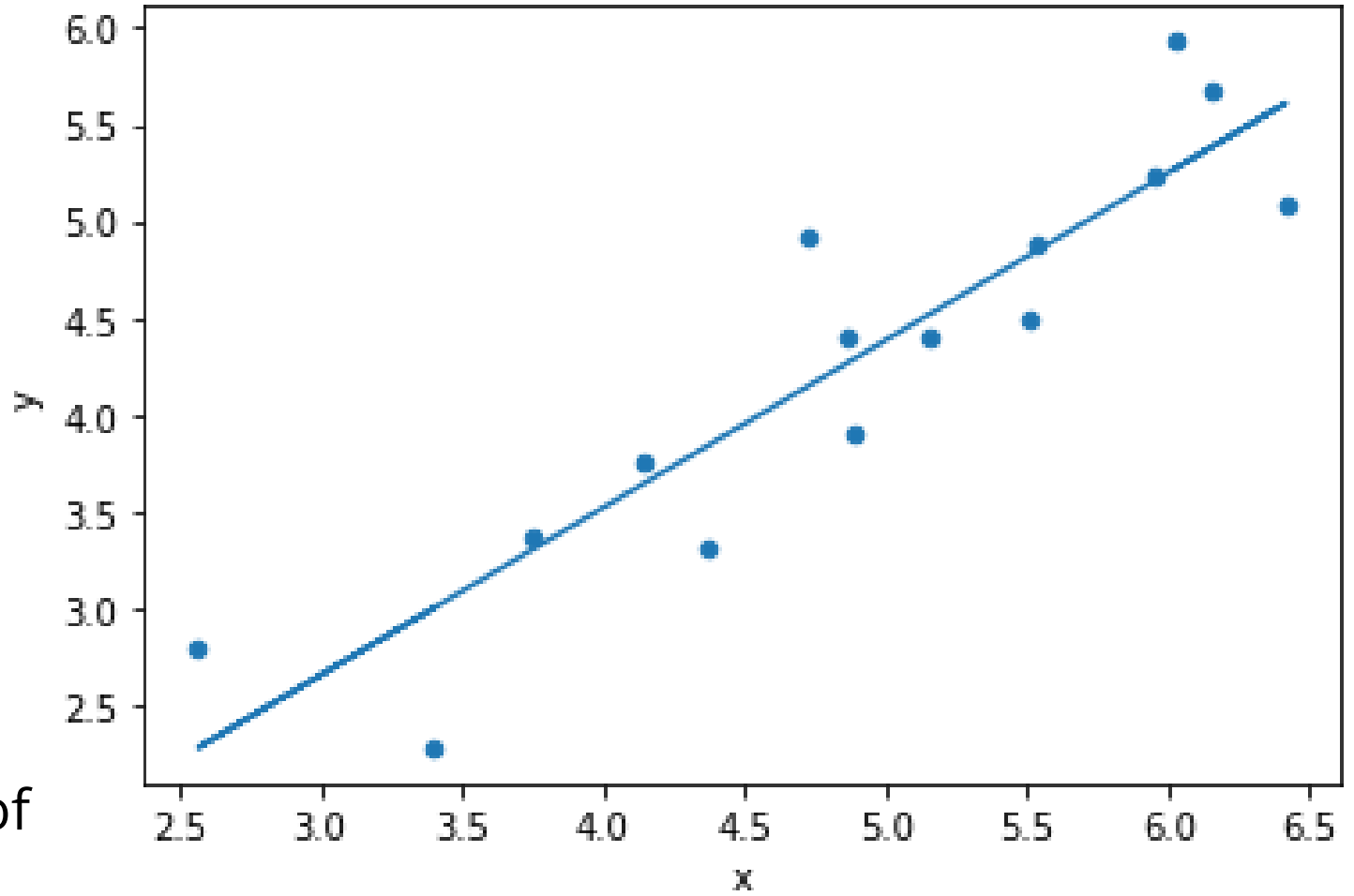


This function creates a “bubble” plot of Studentized residuals versus hat leverage values.

hi residual: outlier on the Y axis

high leverage : at the edge of the X axis

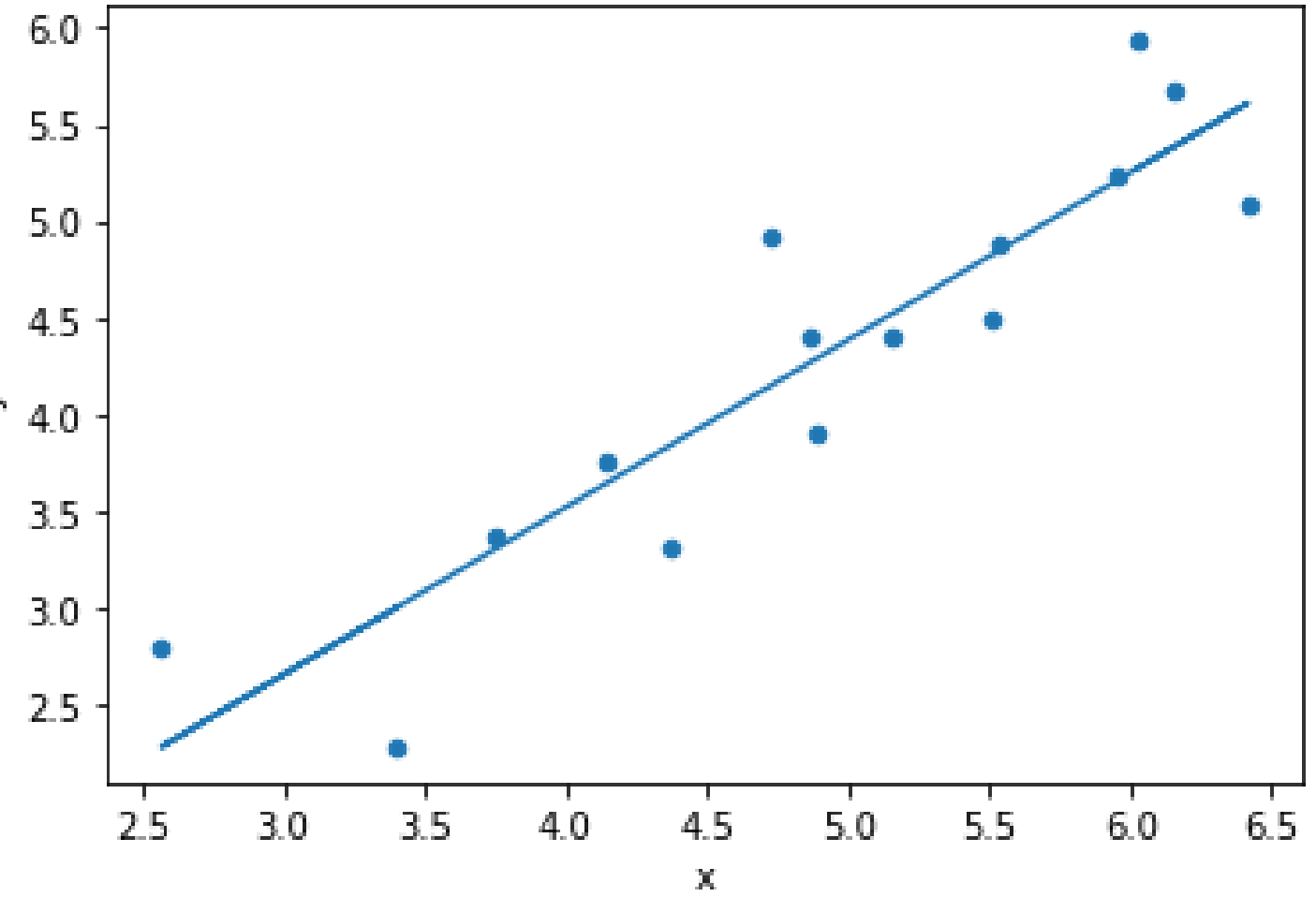
influence analysis



identify data points that have strong influence on the model fit: unusual *x* values or their *y* value is an "outlier". Points that are both are on the top right of the plot and are **high influence points**. Cook's distance is represented by the size of the bubble.

Normal equation

It can be shown that OLS minimization of the sum of the square errors (SSE) is equivalent to calculating the slope and intercept as:



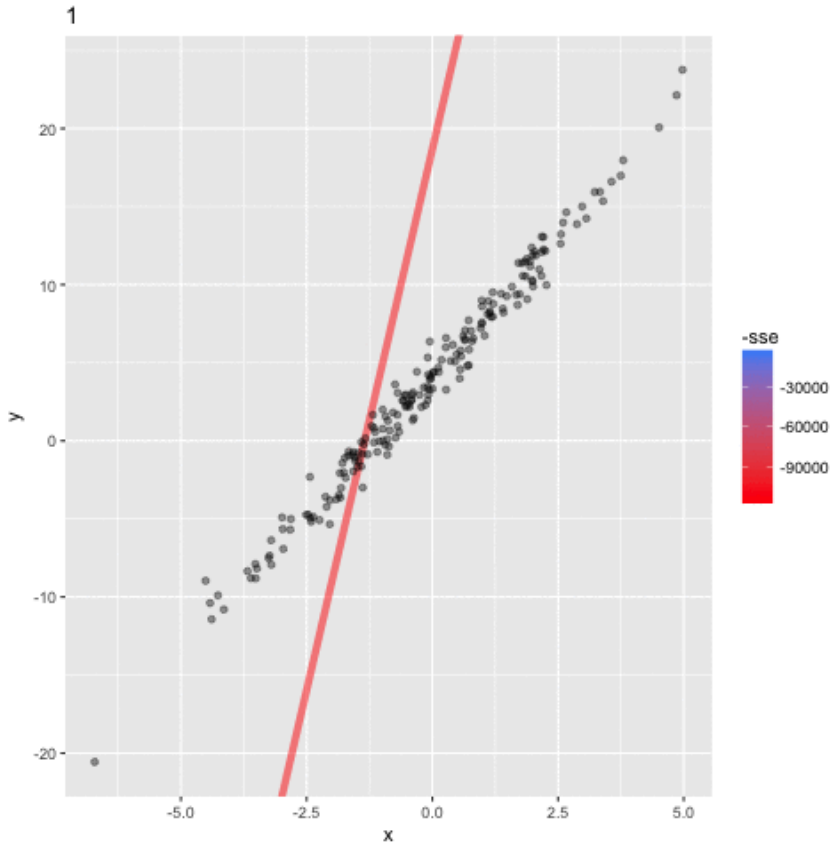
$$s = \frac{\sum_{i=1}^{N}(x_i - \bar{X})(y_i - \bar{Y})}{\sum_{i=1} N(x_i - \bar{X})^2}$$

$$b = \bar{Y} - \bar{X}$$

what is a machine learning?

ML: Any model with parameters learned from the data

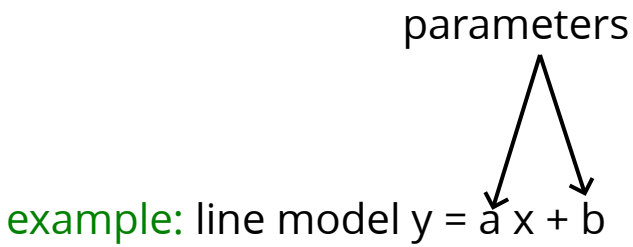
ML models are a parameterized representation of "reality" where the parameters are learned from finite sets (*samples*) of realizations of that reality (*population*)



how do we model?

1 **Choose the model:**

a mathematical formula to represent the behavior in the data

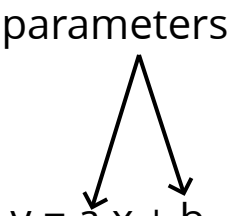


how do we model?

1

Choose the model:

a mathematical formula to represent the behavior in the data



example: line model y = a x + b

Choose the hyperparameters:

parameters chosen **before** the learning process, which govern the model and training process

example: the *degree N* of the polynomial $y = \sum_{i=0}^{N} c_i x^i$

how do we model?

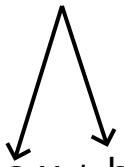
2 Choose an objective function:

in order to find the "best" parameters of the model: we need to "optimize" a function.

We need something to be either **MINIMIZED** or **MAXIMIZED**

parameters

example:



line model: y = a x + b

objective function: sum of residual squared (least square fit method)

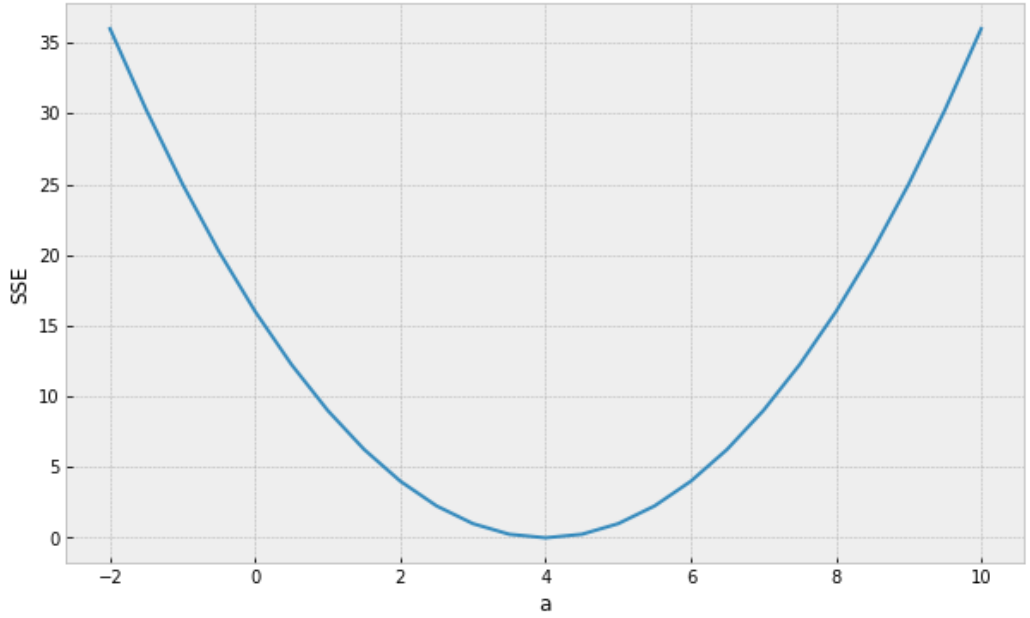
$$SSE = \sum (y_{i,observed} - y_{i,predicted})^2$$
$$SSE = \sum (y_{i,observed} - (ax_i + b))^2$$

we want to **minimize** *SSE* as much as possible

Optimizing the Objective Function

assume a simpler line model $y = ax$

($b = 0$) so we only need to find the "best" parameter a



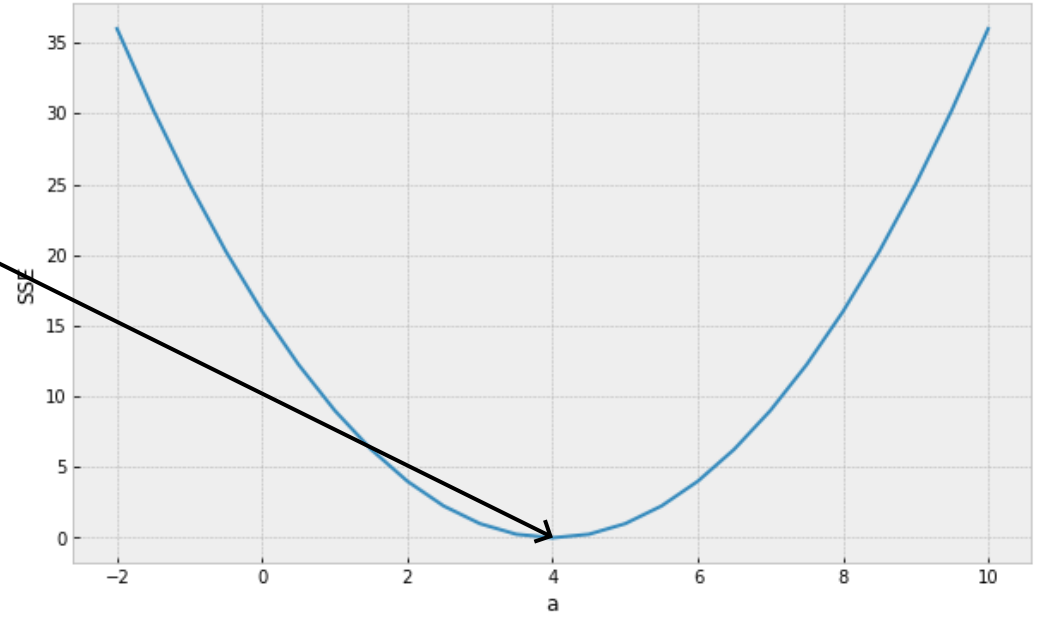
Optimizing the Objective Function

assume a simpler line model $y = ax$

($b = 0$) so we only need to find the "best" parameter a

Minimum (optimal) SSE

➡ a = 4



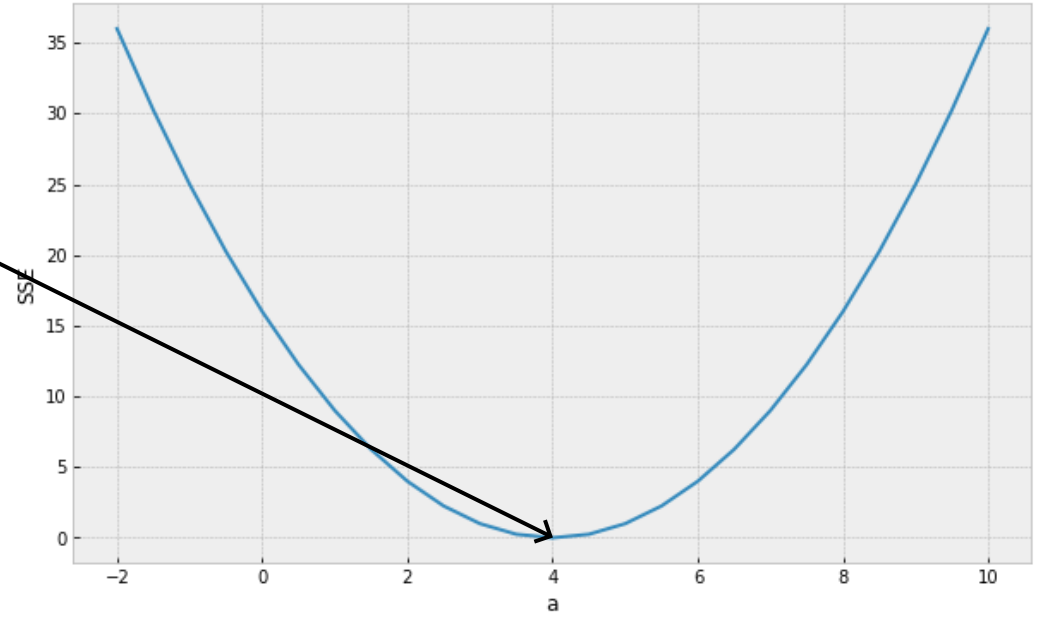
Optimizing the Objective Function

assume a simpler line model $y = ax$

($b = 0$) so we only need to find the "best" parameter a

Minimum (optimal) SSE

➡ a = 4



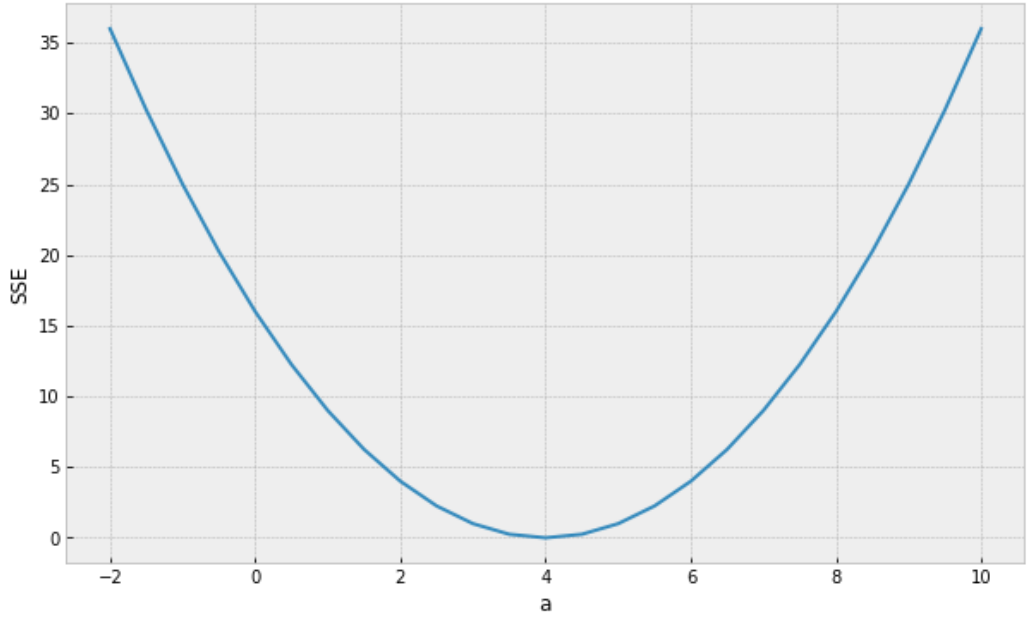
How do we find the minimum if we do not know beforehand how the SSE curve looks like?

31 stochastic gradient descent (SGD)

the algorithm: **Stochastic Gradient Descent (SGD)**

assume a simpler line model $y = ax$

($b = 0$) so we only need to find the "best" parameter a

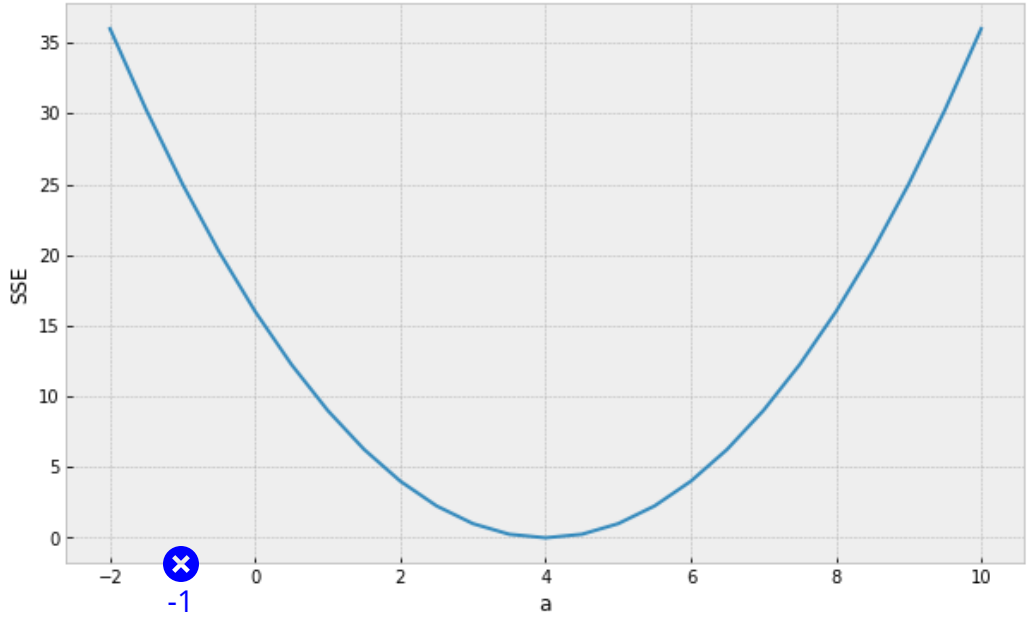


the algorithm: ~~Stochastic~~ Gradient Descent

assume a simpler line model $y = ax$

($b = 0$) so we only need to find the "best" parameter a

1. choose initial value for a

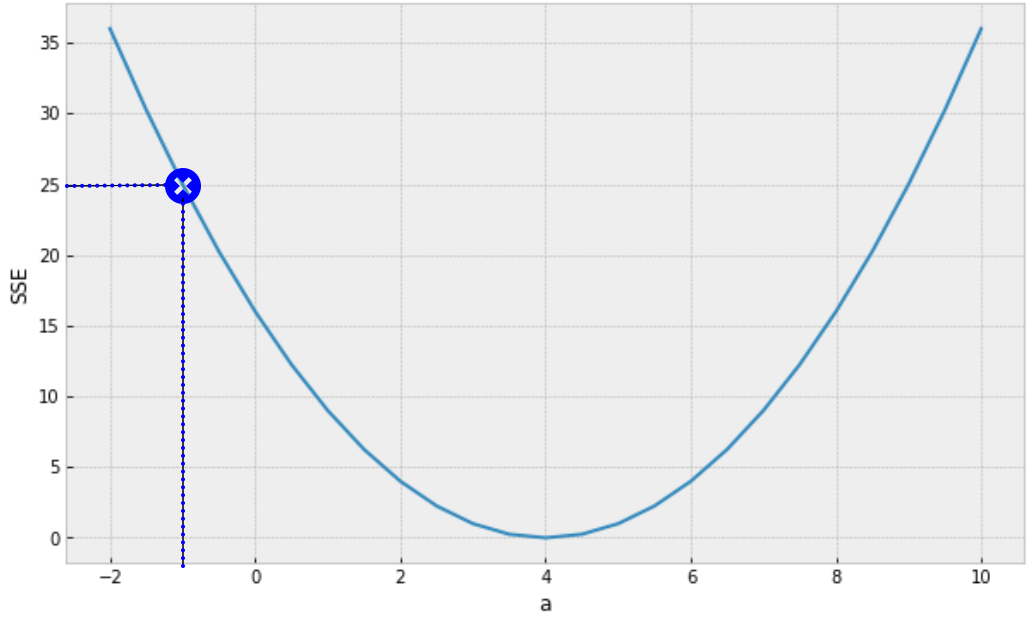


the algorithm: ~~Stochastic~~ Gradient Descent

assume a simpler line model $y = ax$

($b = 0$) so we only need to find the "best" parameter a

1. choose initial value for a
2. calculate the SSE

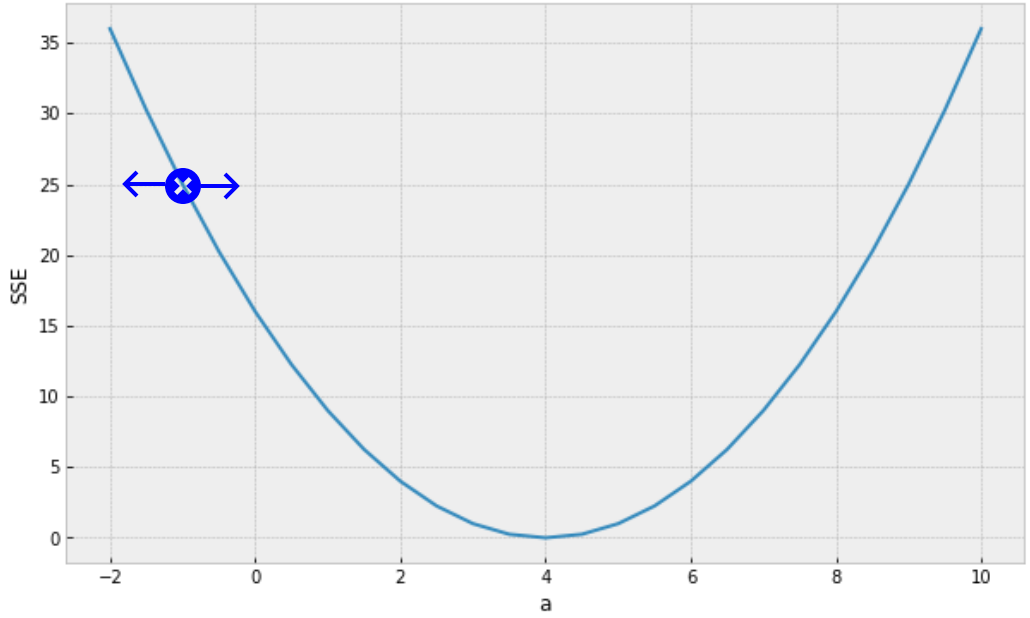


the algorithm: ~~Stochastic~~ Gradient Descent

assume a simpler line model $y = ax$

($b = 0$) so we only need to find the "best" parameter a

1. choose initial value for a
2. calculate the SSE
3. calculate best direction to go to decrease the SSE

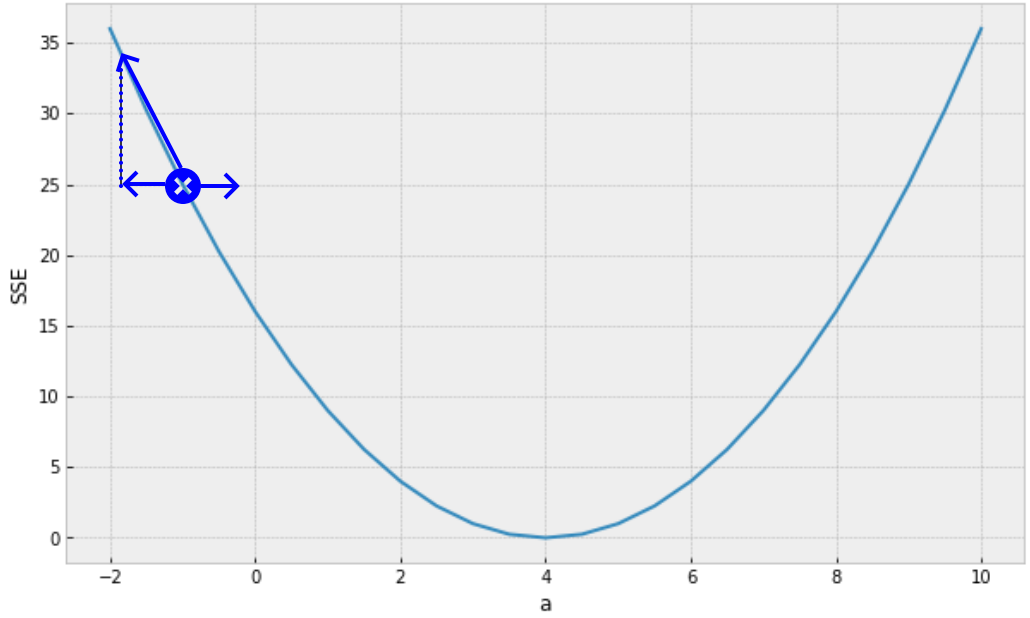


the algorithm: ~~Stochastic~~ Gradient Descent

assume a simpler line model $y = ax$

($b = 0$) so we only need to find the "best" parameter a

1. choose initial value for a
2. calculate the SSE
3. calculate best direction to go to decrease the SSE

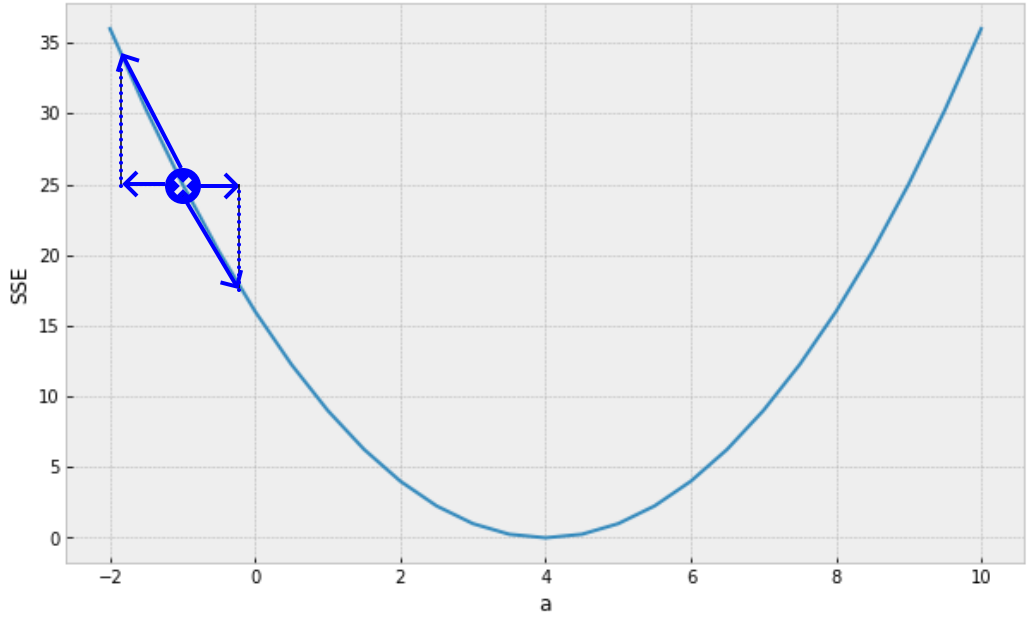


the algorithm: ~~Stochastic~~ Gradient Descent

assume a simpler line model $y = ax$

($b = 0$) so we only need to find the "best" parameter a

1. choose initial value for a
2. calculate the SSE
3. calculate best direction to go to decrease the SSE

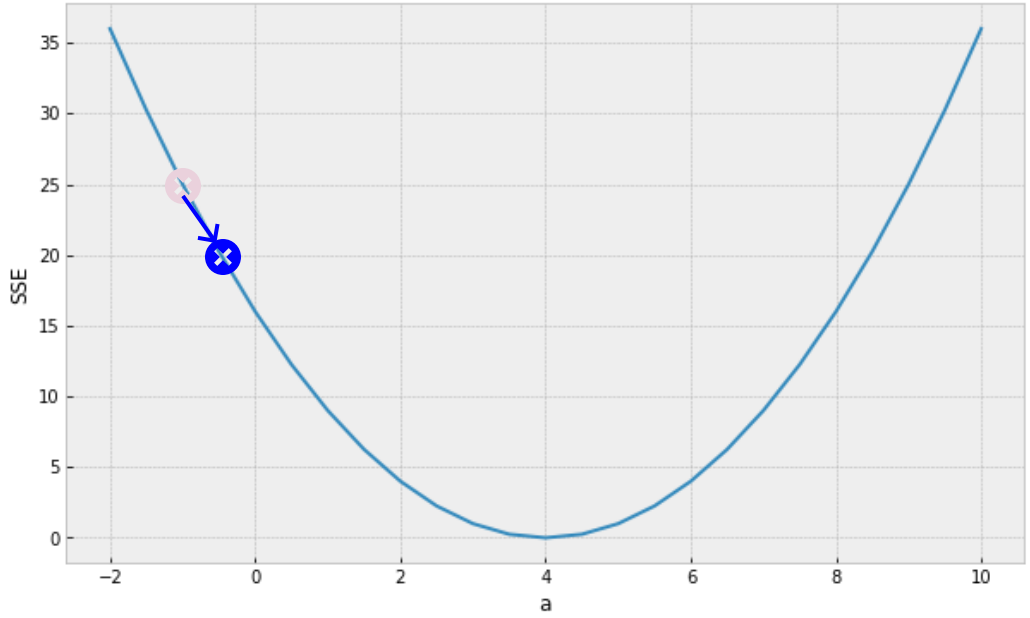


the algorithm: ~~Stochastic~~ Gradient Descent

assume a simpler line model $y = ax$

($b = 0$) so we only need to find the "best" parameter a

1. choose initial value for a
2. calculate the SSE
3. calculate best direction to go to decrease the SSE
4. step in that direction

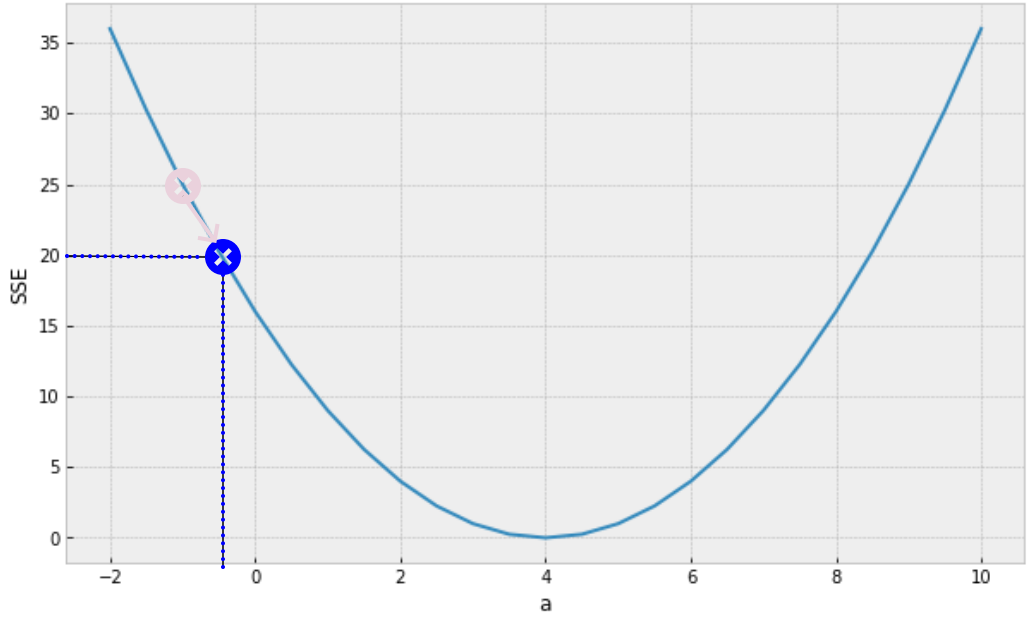


the algorithm: ~~Stochastic~~ Gradient Descent

assume a simpler line model $y = ax$

($b = 0$) so we only need to find the "best" parameter a

1. choose initial value for a
2. calculate the SSE
3. calculate best direction to go to decrease the SSE
4. step in that direction
5. go back to step 2 and repeat

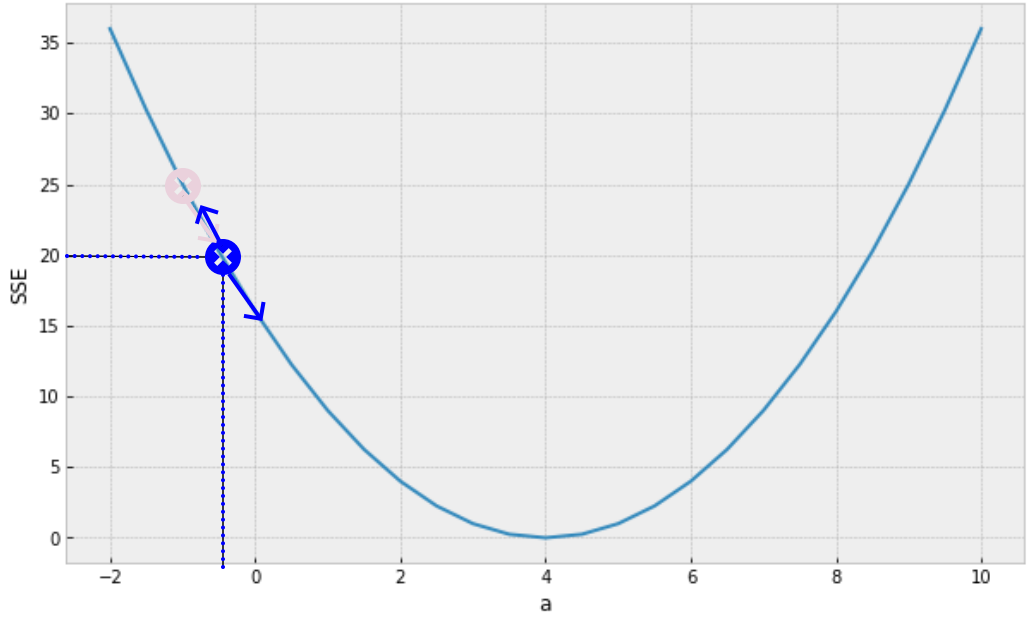


the algorithm: ~~Stochastic~~ Gradient Descent

assume a simpler line model $y = ax$

($b = 0$) so we only need to find the "best" parameter a

1. choose initial value for a
2. calculate the SSE
3. calculate best direction to go to decrease the SSE
4. step in that direction
5. go back to step 2 and repeat

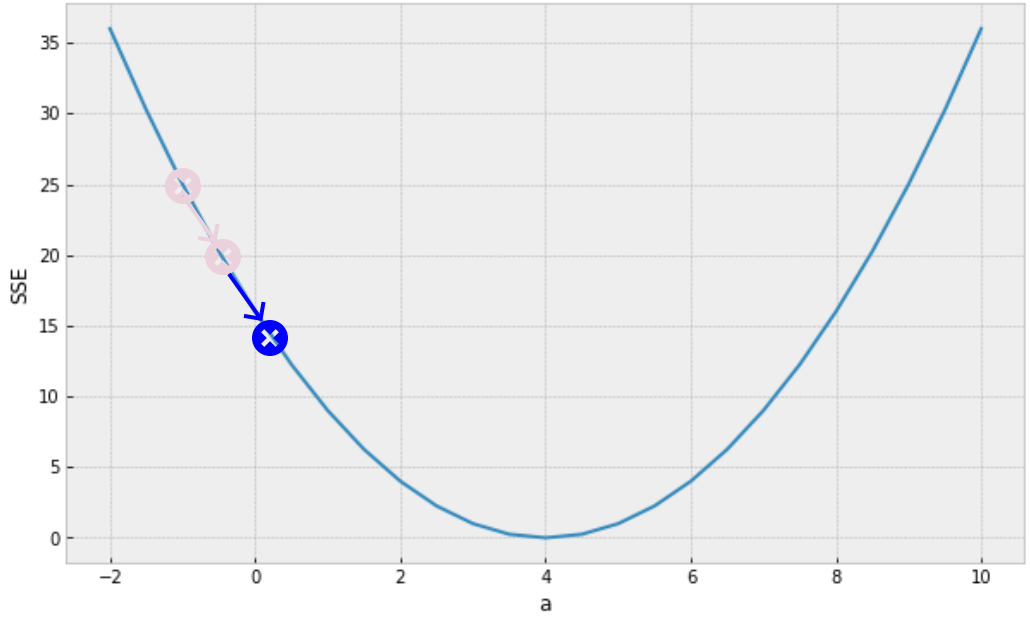


the algorithm: ~~Stochastic~~ Gradient Descent

assume a simpler line model $y = ax$

($b = 0$) so we only need to find the "best" parameter a

1. choose initial value for a
2. calculate the SSE
3. calculate best direction to go to decrease the SSE
4. step in that direction
5. go back to step 2 and repeat

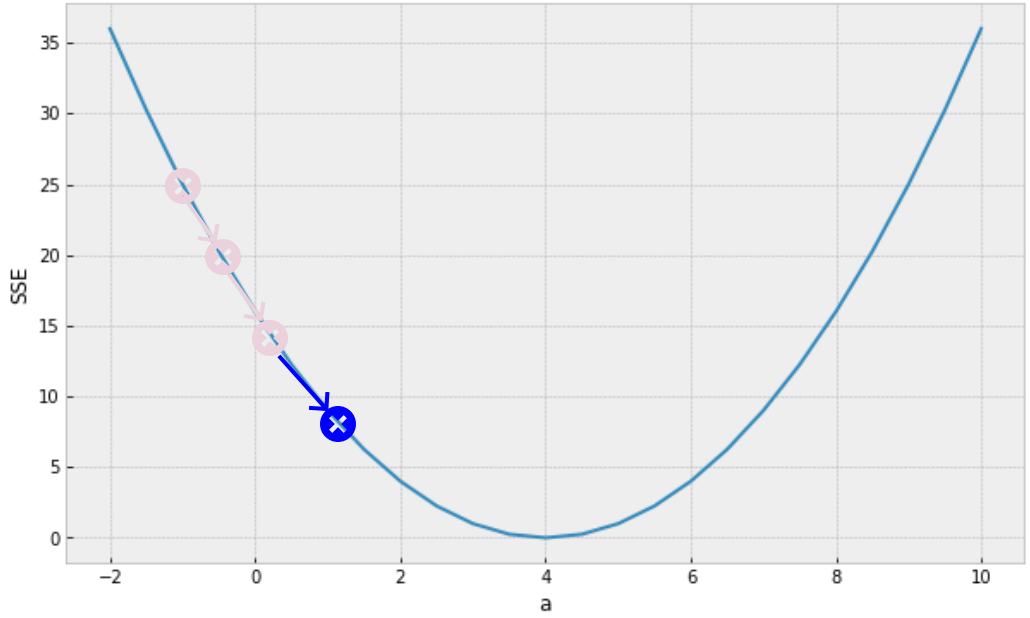


the algorithm: ~~Stochastic~~ Gradient Descent

assume a simpler line model $y = ax$

($b = 0$) so we only need to find the "best" parameter a

1. choose initial value for a
2. calculate the SSE
3. calculate best direction to go to decrease the SSE
4. step in that direction
5. go back to step 2 and repeat

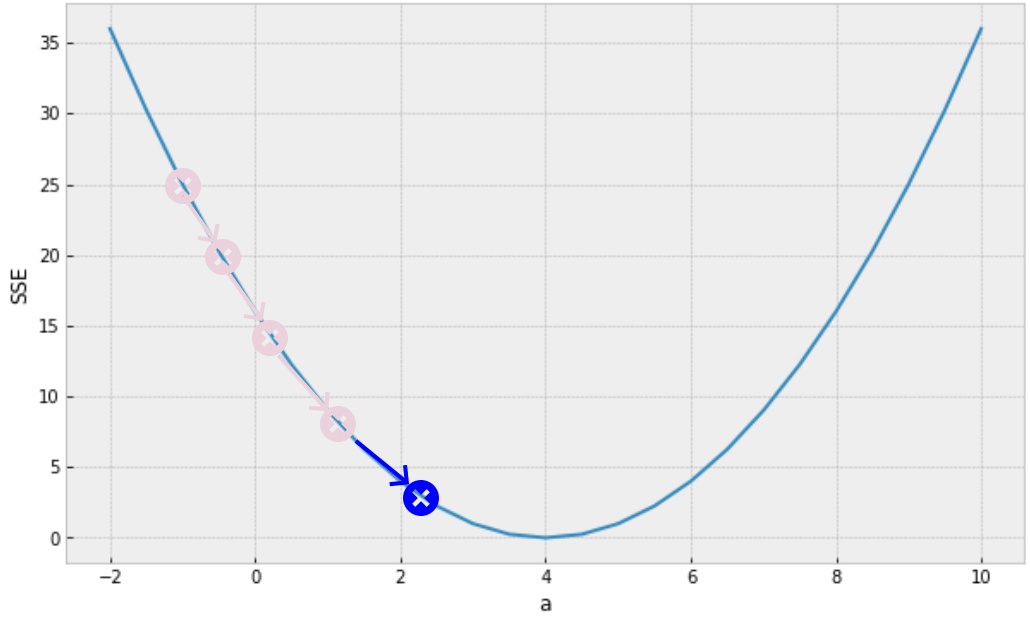


the algorithm: ~~Stochastic~~ Gradient Descent

assume a simpler line model $y = ax$

($b = 0$) so we only need to find the "best" parameter a

1. choose initial value for a
2. calculate the SSE
3. calculate best direction to go to decrease the SSE
4. step in that direction
5. go back to step 2 and repeat

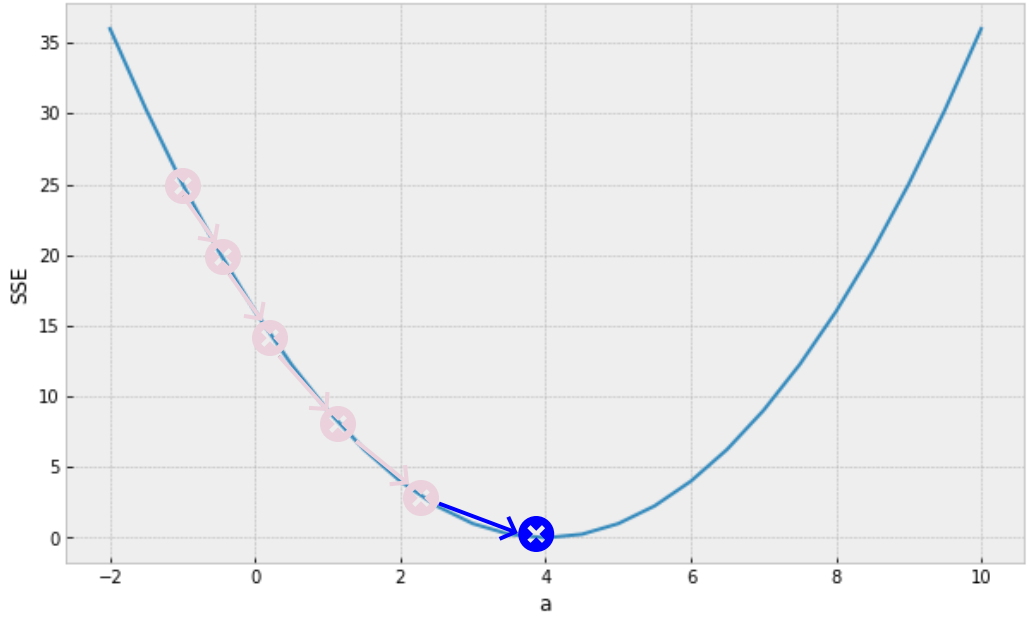


the algorithm: ~~Stochastic~~ Gradient Descent

assume a simpler line model $y = ax$

($b = 0$) so we only need to find the "best" parameter a

1. choose initial value for a
2. calculate the SSE
3. calculate best direction to go to decrease the SSE
4. step in that direction
5. go back to step 2 and repeat

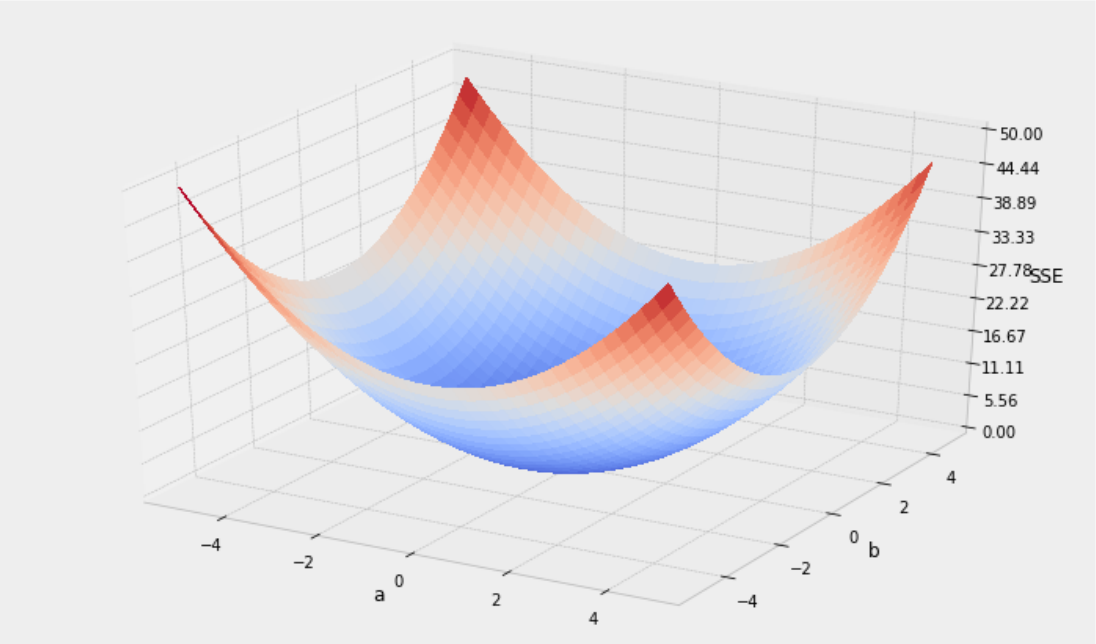


the algorithm: ~~Stochastic~~ Gradient Descent

for a line model $y = ax + b$

we need to find the "best" parameters a and b

1. choose initial value for a & b
2. calculate the SSE
3. calculate best direction to go to decrease the SSE
4. step in that direction
5. go back to step 2 and repeat

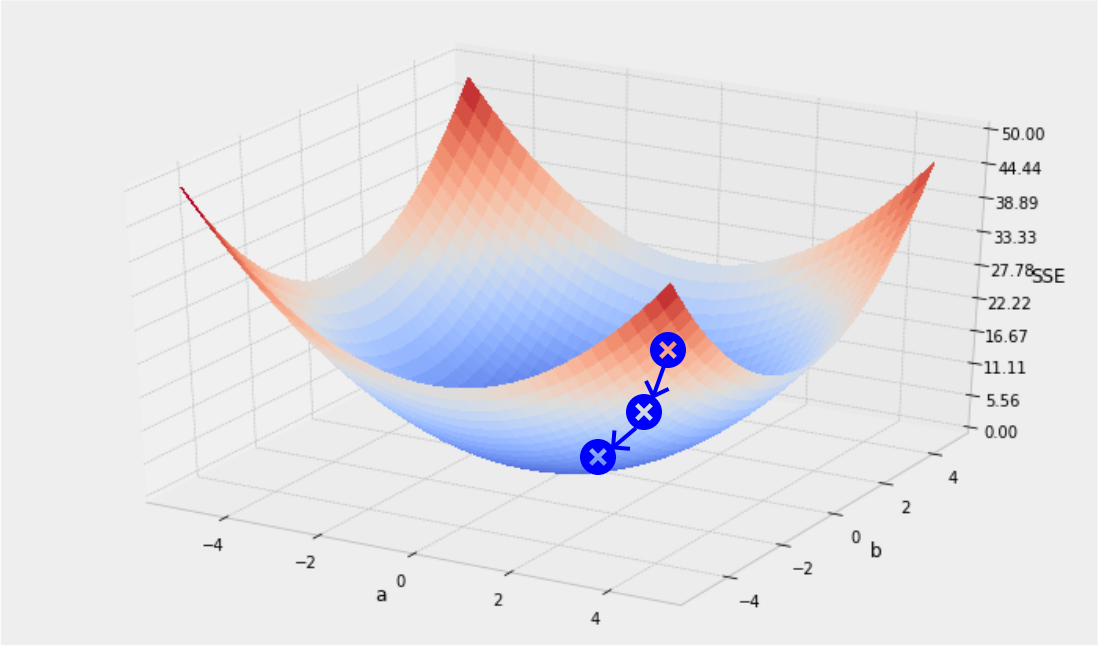


the algorithm: ~~Stochastic~~ Gradient Descent

for a line model $y = ax + b$

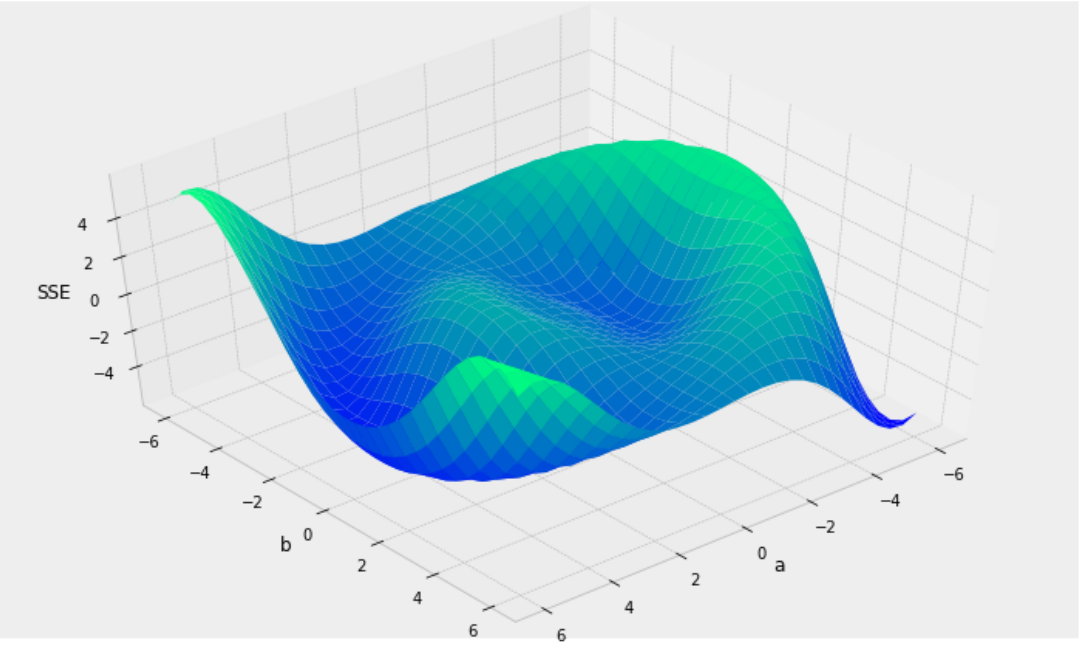
we need to find the "best" parameters a and b

1. choose initial value for a & b
2. calculate the SSE
3. calculate best direction to go to decrease the SSE
4. step in that direction
5. go back to step 2 and repeat



the algorithm: ~~Stochastic~~ Gradient Descent

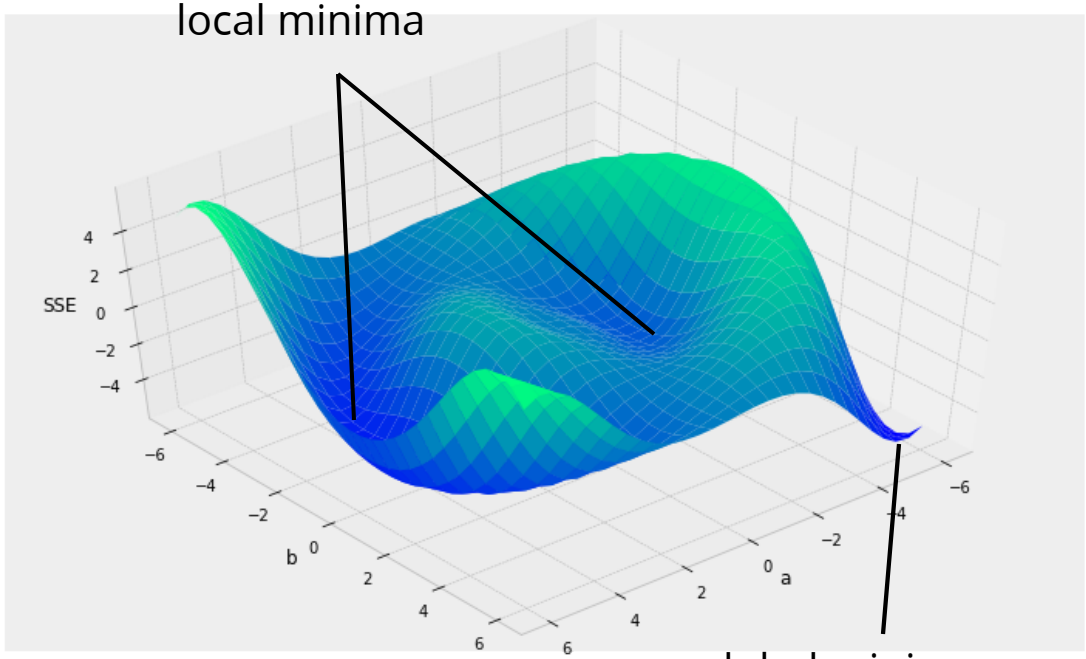
Things to consider:



the algorithm: ~~Stochastic~~ Gradient Descent

Things to consider:

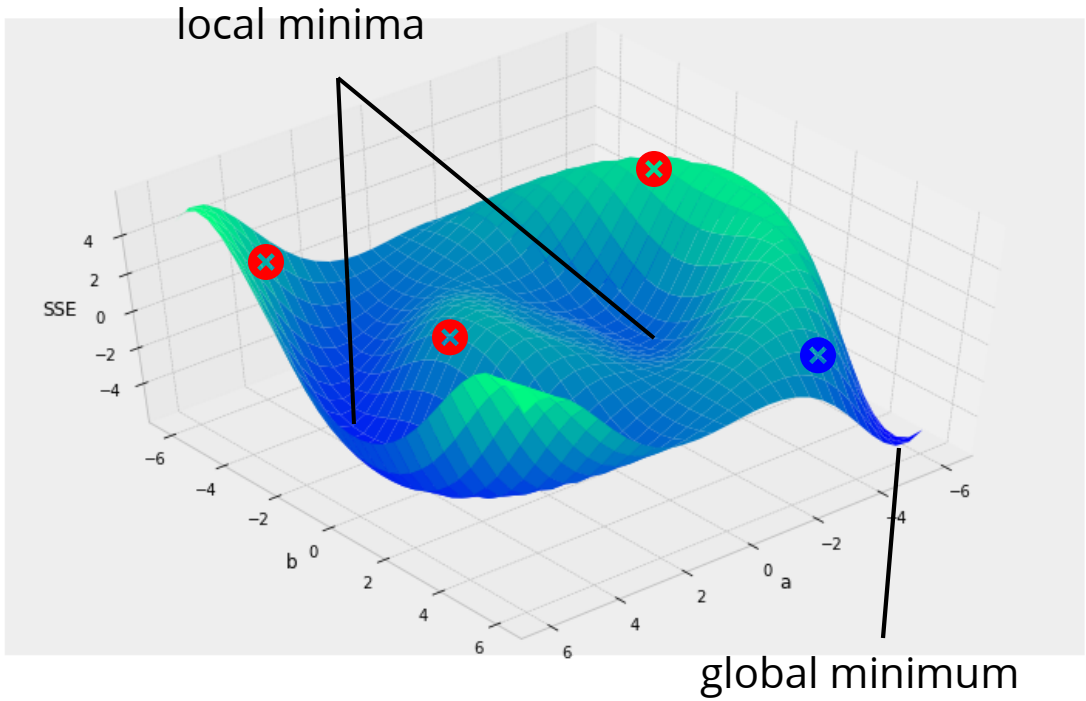
- local vs. global minima



the algorithm: ~~Stochastic~~ Gradient Descent

Things to consider:

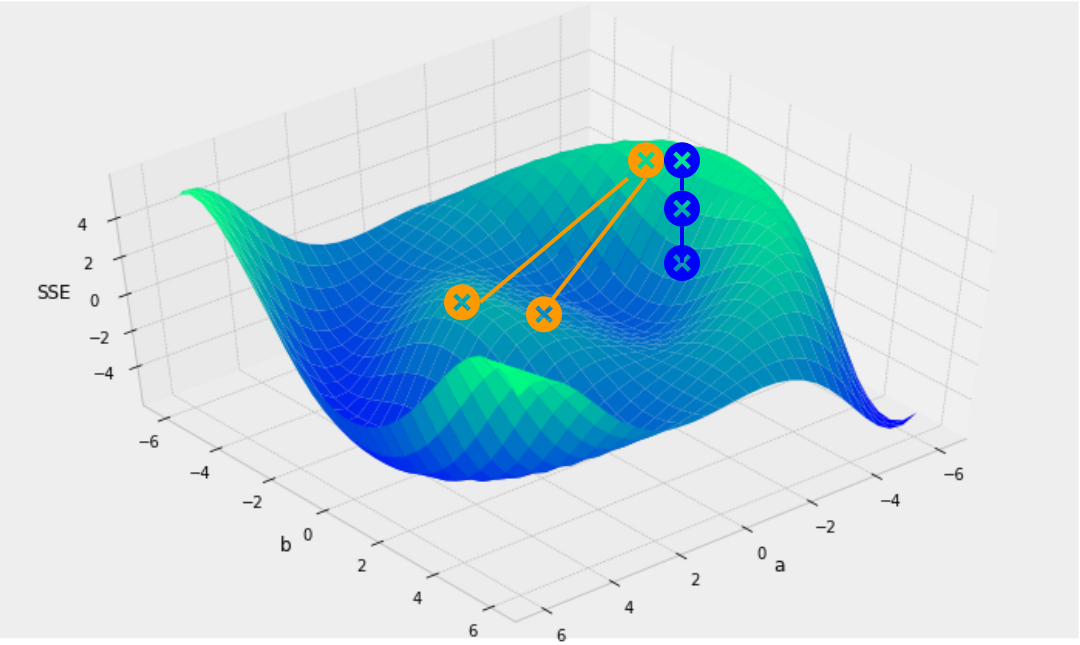
- local vs. global minima
- initialization: choosing starting spot?



the algorithm: ~~Stochastic~~ Gradient Descent

Things to consider:

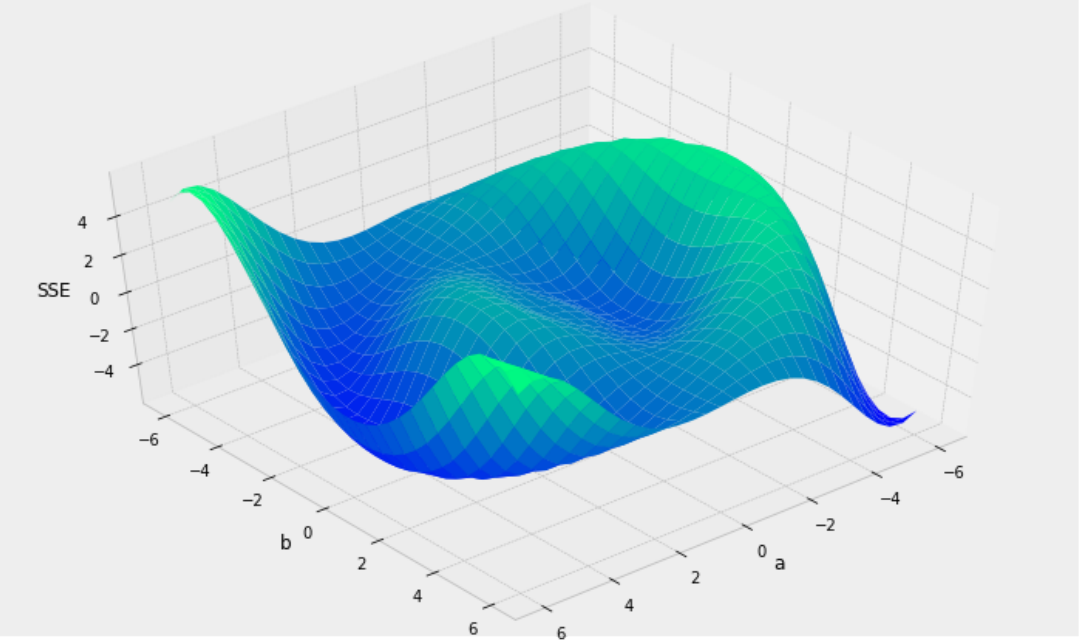
- local vs. global minima
- initialization: choosing starting spot?
- learning rate: how far to step?



the algorithm: ~~Stochastic~~ Gradient Descent

Things to consider:

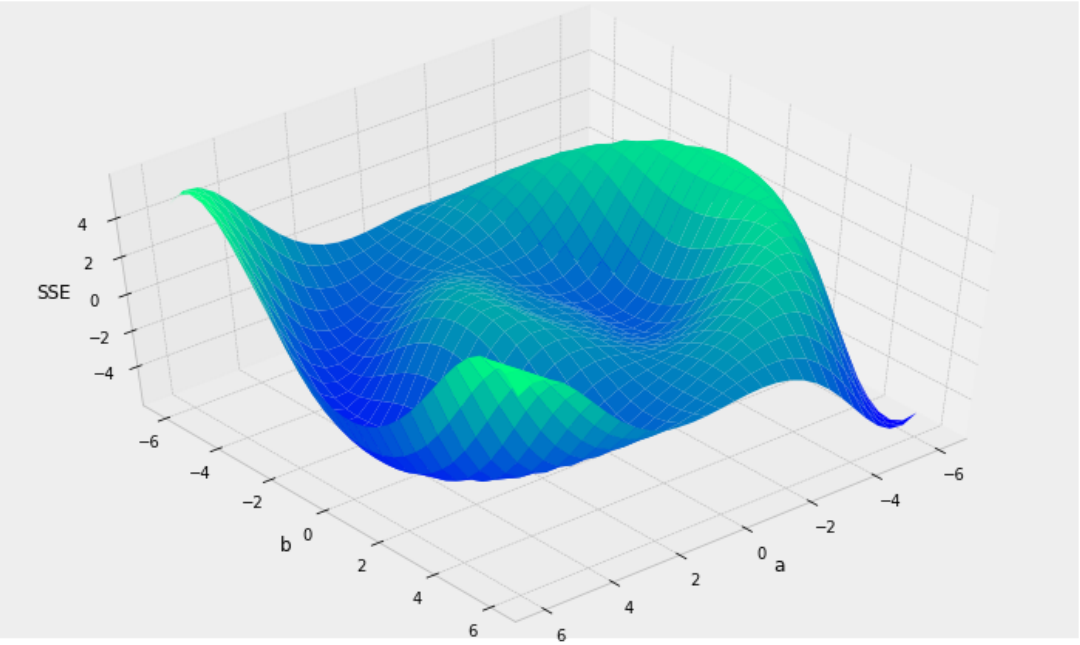
- local vs. global minima
- initialization: choosing starting spot?
- learning rate: how far to step?
- stopping criterion: when to stop?



the algorithm: **Stochastic Gradient Descent**

Things to consider:

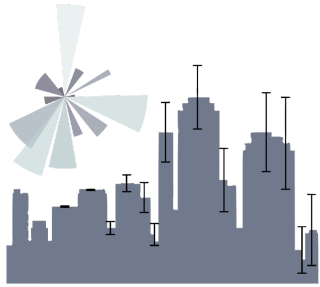
- local vs. global minima
- initialization: choosing starting spot?
- learning rate: how far to step?
- stopping criterion: when to stop?



<u>Stochastic</u> Gradient Descent (SGD): use a different (random) sub-sample of the data at each iteration

4 epistemological rooots of overfitting

Okham's razor

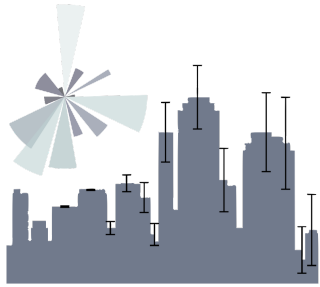


Ockham's razor: ***Pluralitas non est ponenda sine neccesitate***
or "the law of parsimony"

William of Ockham (logician and Franciscan friar) 1300ca
but probably to be attributed to John Duns Scotus

"Complexity needs not to be postulated without a need for it"
"Between 2 theories choose the simpler one"

Okham's razor



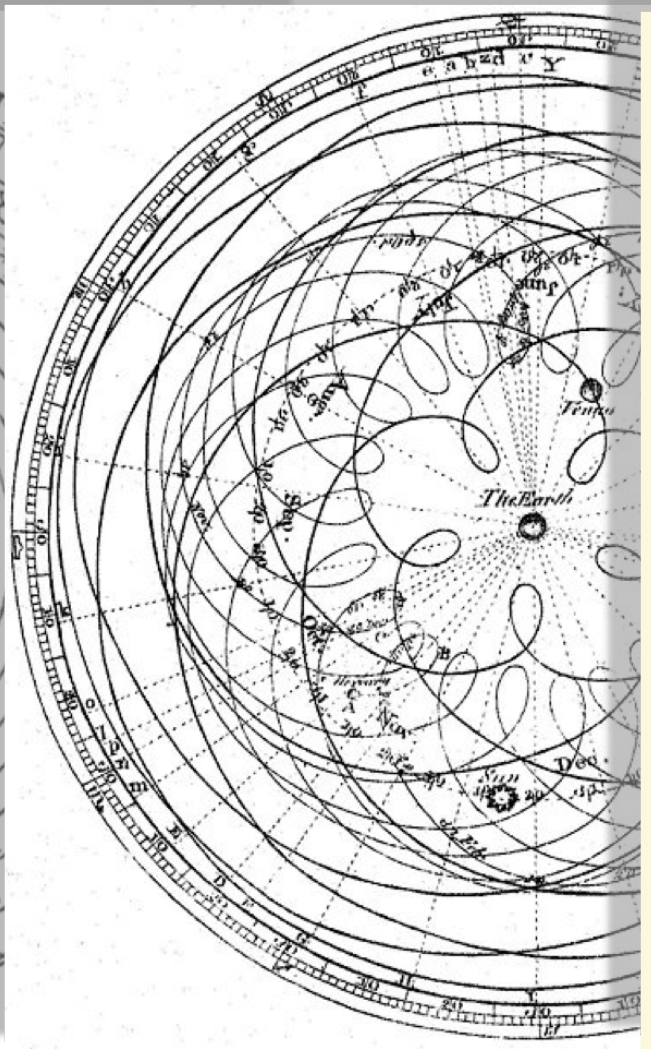
Ockham's razor: ***Pluralitas non est ponenda sine neccesitate***
or "the law of parsimony"

William of Ockham (logician and Franciscan friar) 1300ca
but probably to be attributed to John Duns Scotus

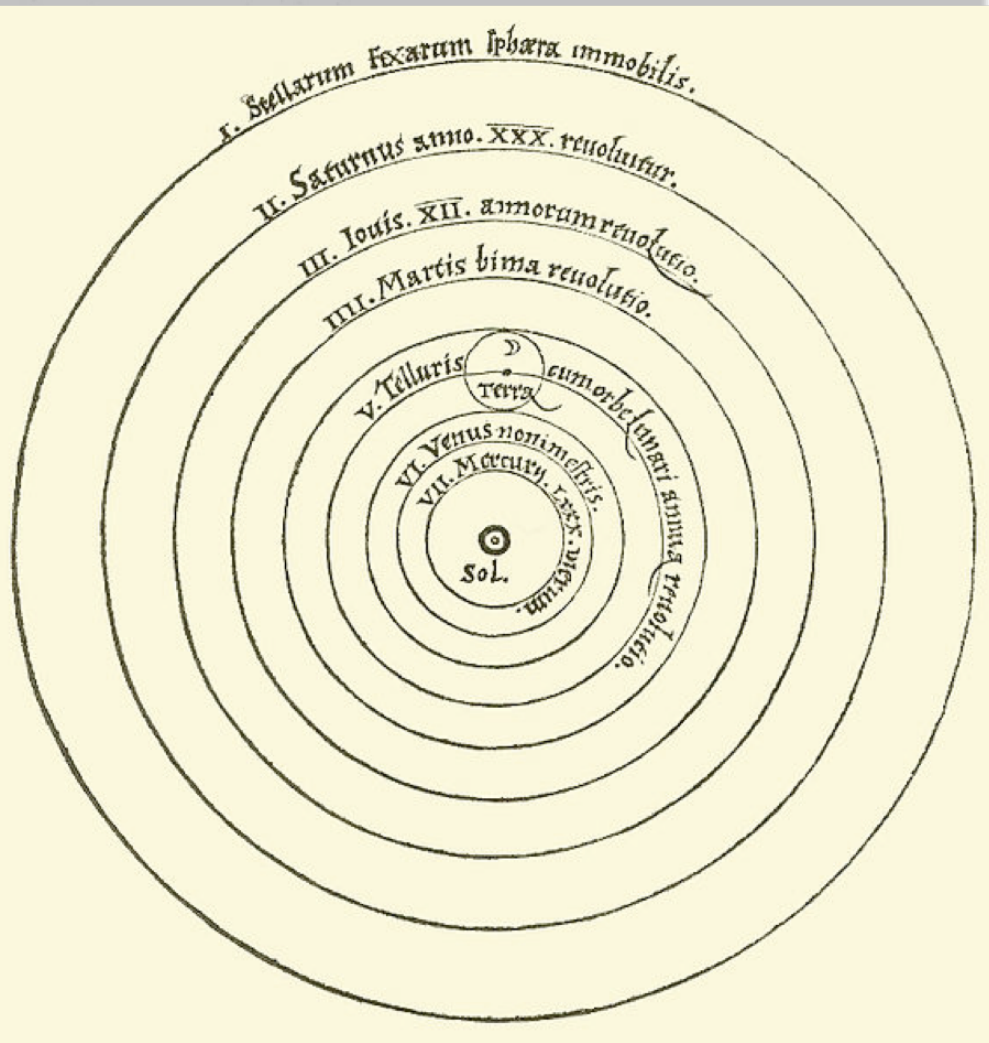
"Complexity needs not to be postulated without a need for it"
"Between 2 theories choose the simpler one"
"Between 2 theories choose the one with fewer parameters"

Okham's razor

Peter Apian, Cosmographia, Antwerp, 1524



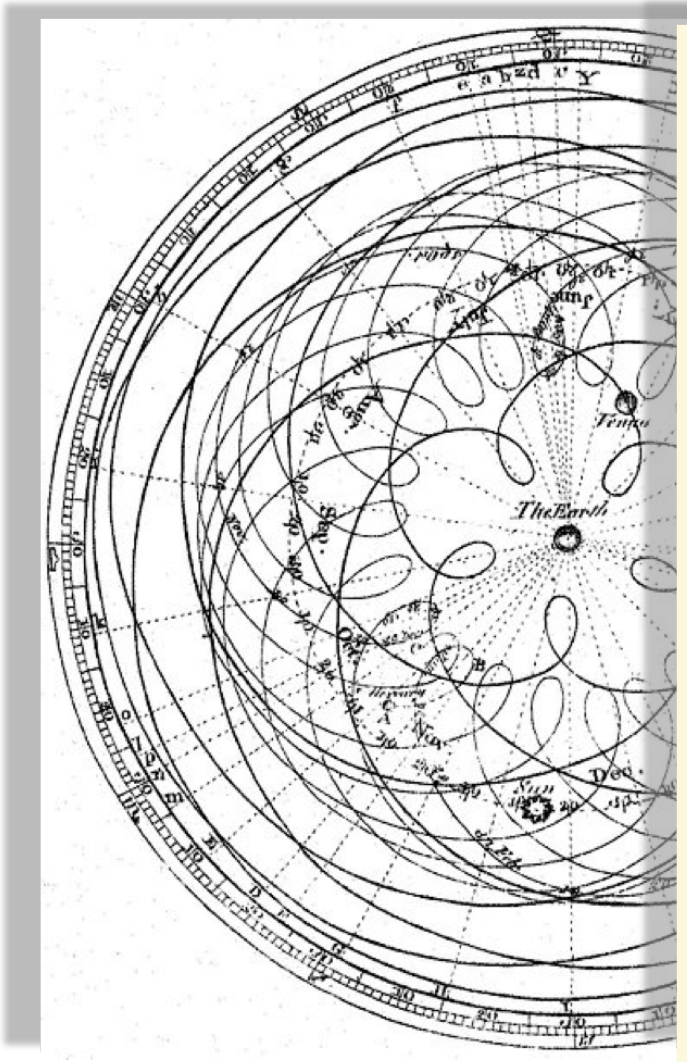
Author Dr Long's copy of Cassini, 1777



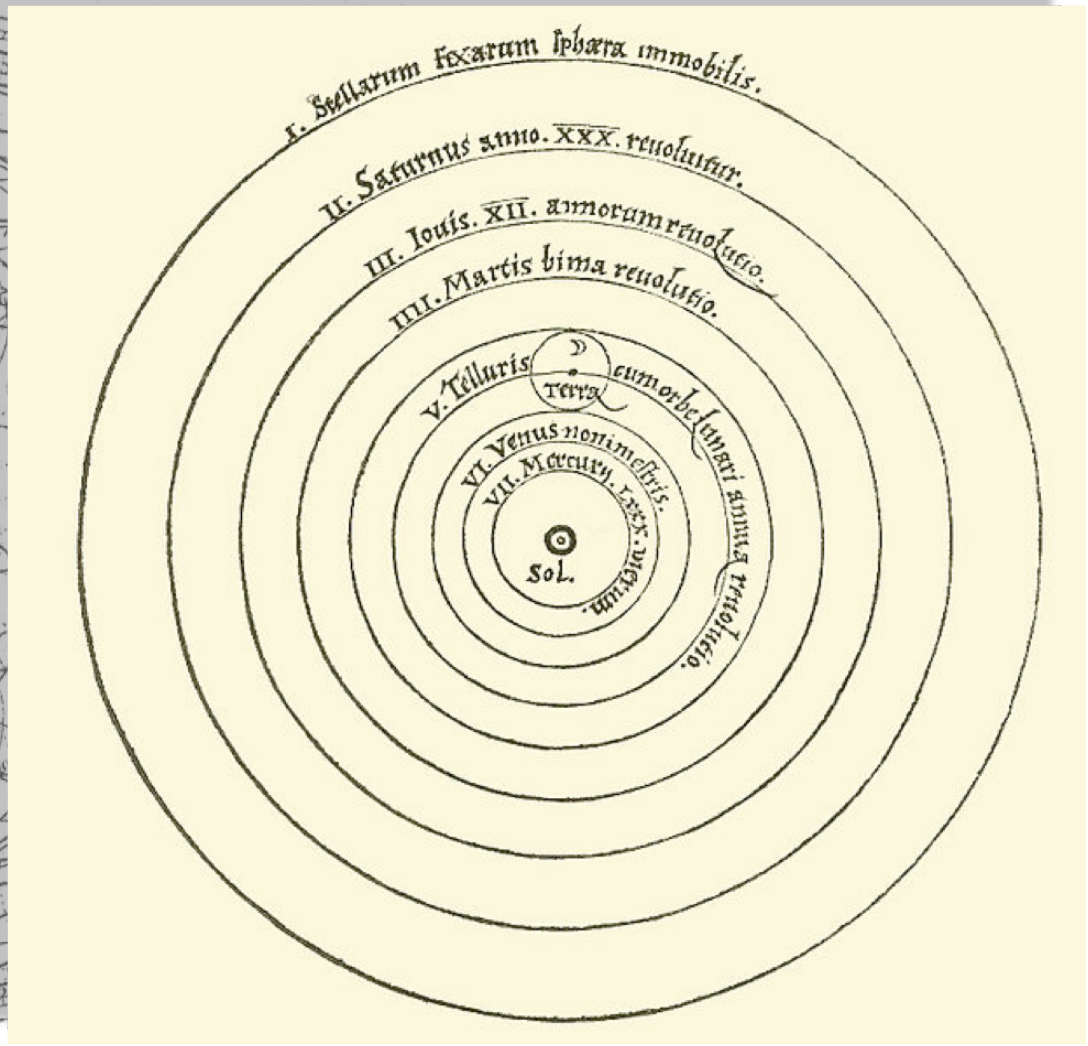
Heliocentric model from Nicolaus Copernicus' "De revolutionibus orbium coelestium".

Okham's razor

Two theories may explain a phenomenon just as well as each other. In that case you should prefer the simpler one

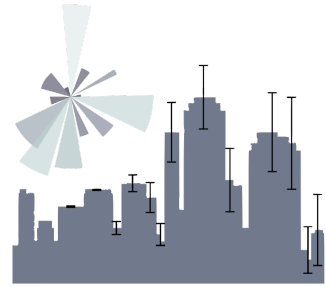


Author Dr Long's copy of Cassini, 1777

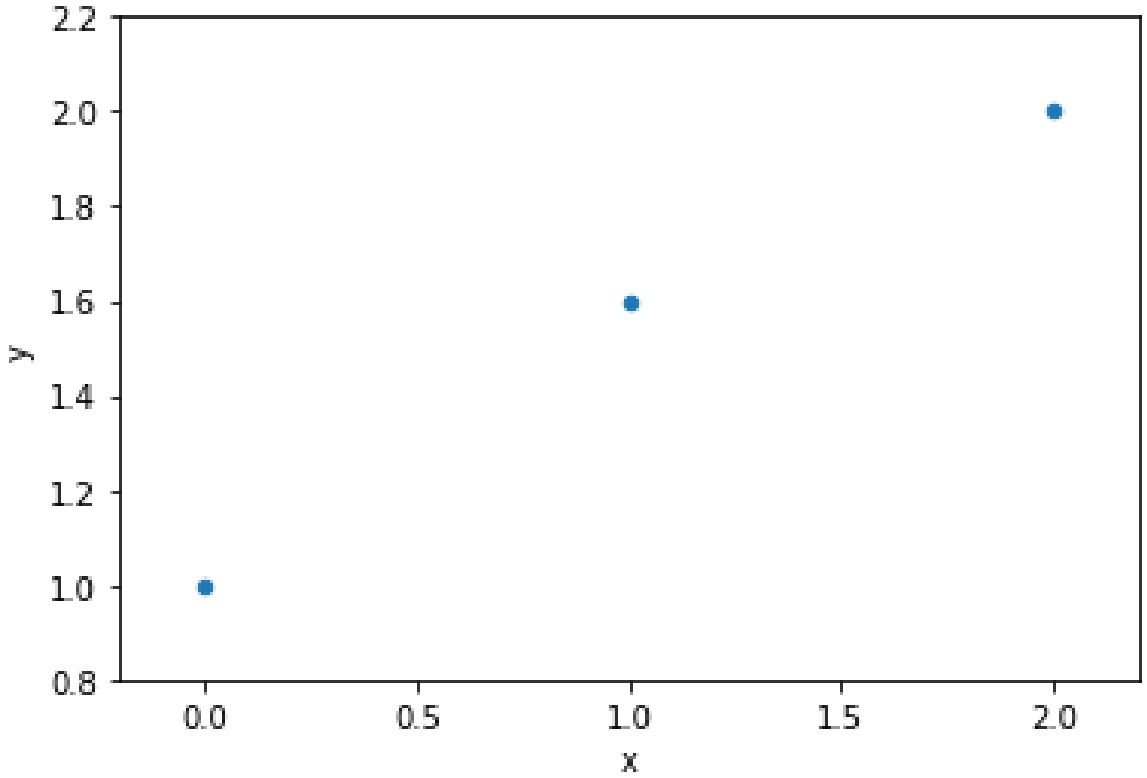


Heliocentric model from Nicolaus Copernicus' "De revolutionibus orbium coelestium".

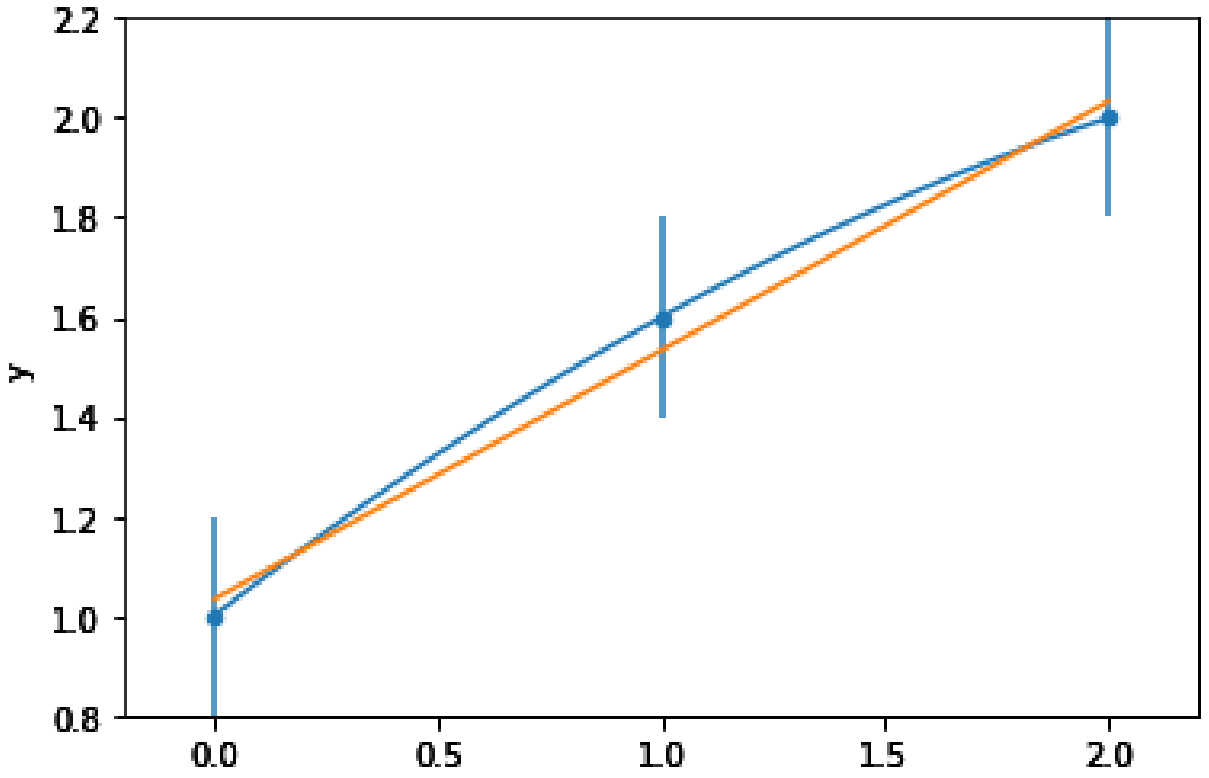
Okham's razor



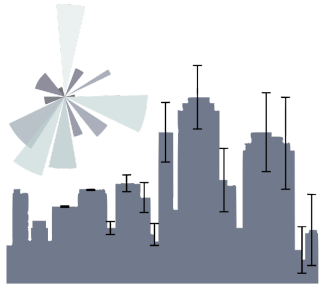
data



model fit to data



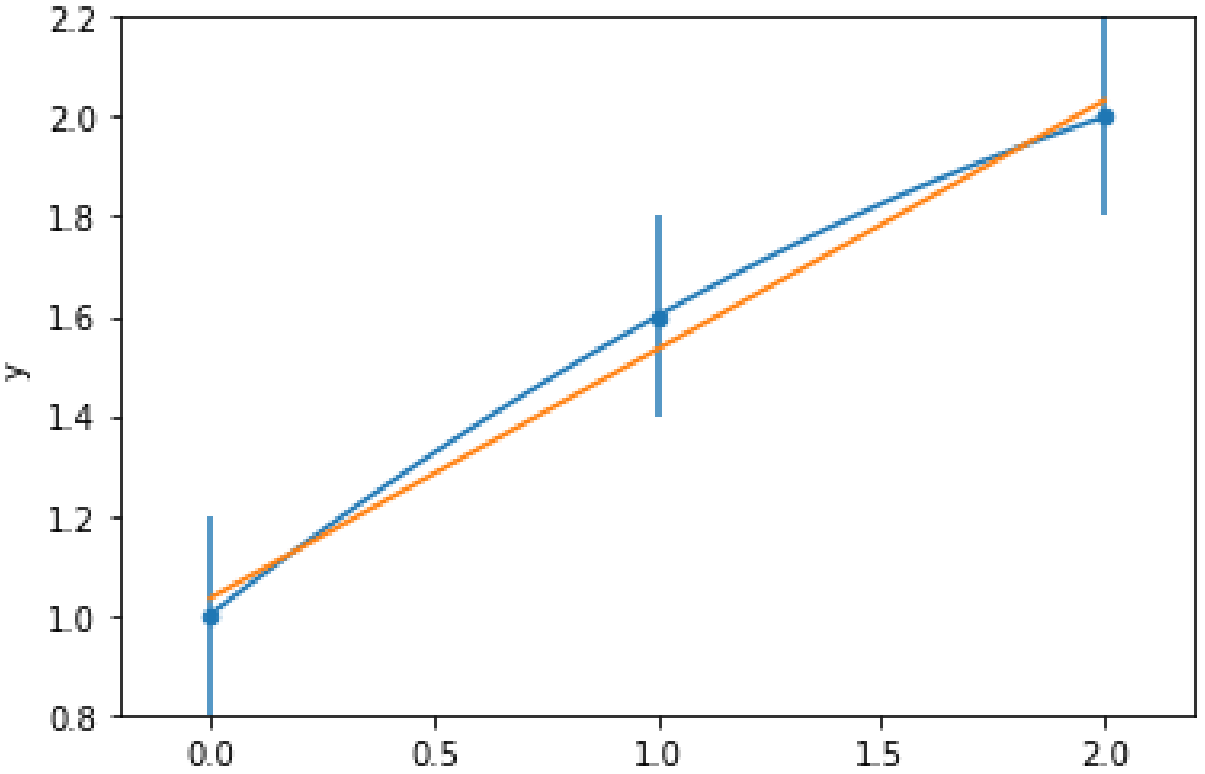
Okham's razor



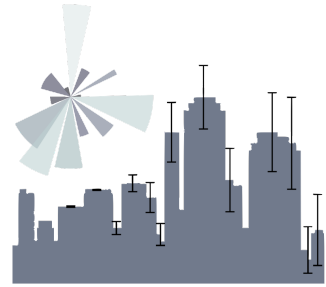
model fit to data

$y = ax + b$

$y = ax^2 + bx + c$



Okham's razor

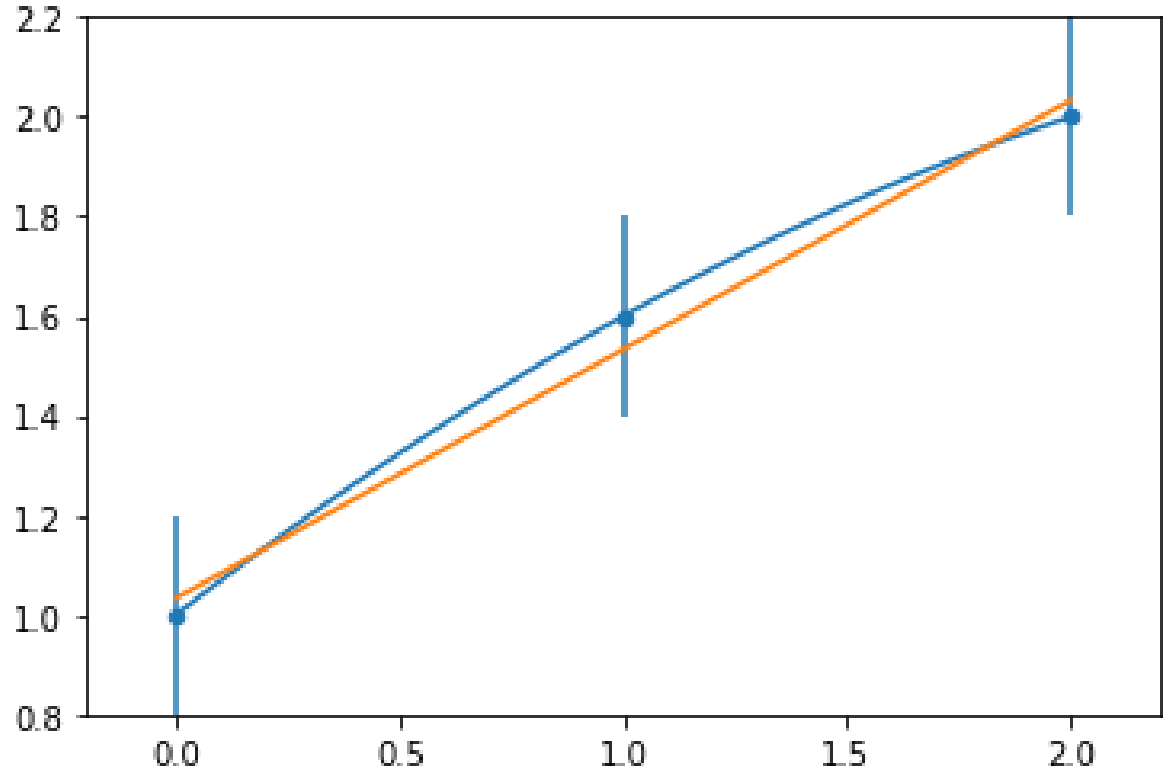


model fit to data

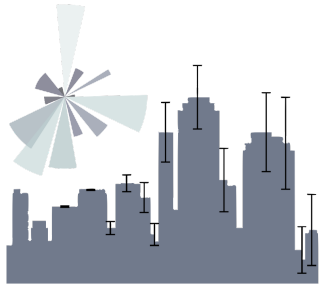
$y = ax + b$

$y = ax^2 + bx + c$

1 variable: x



Okham's razor



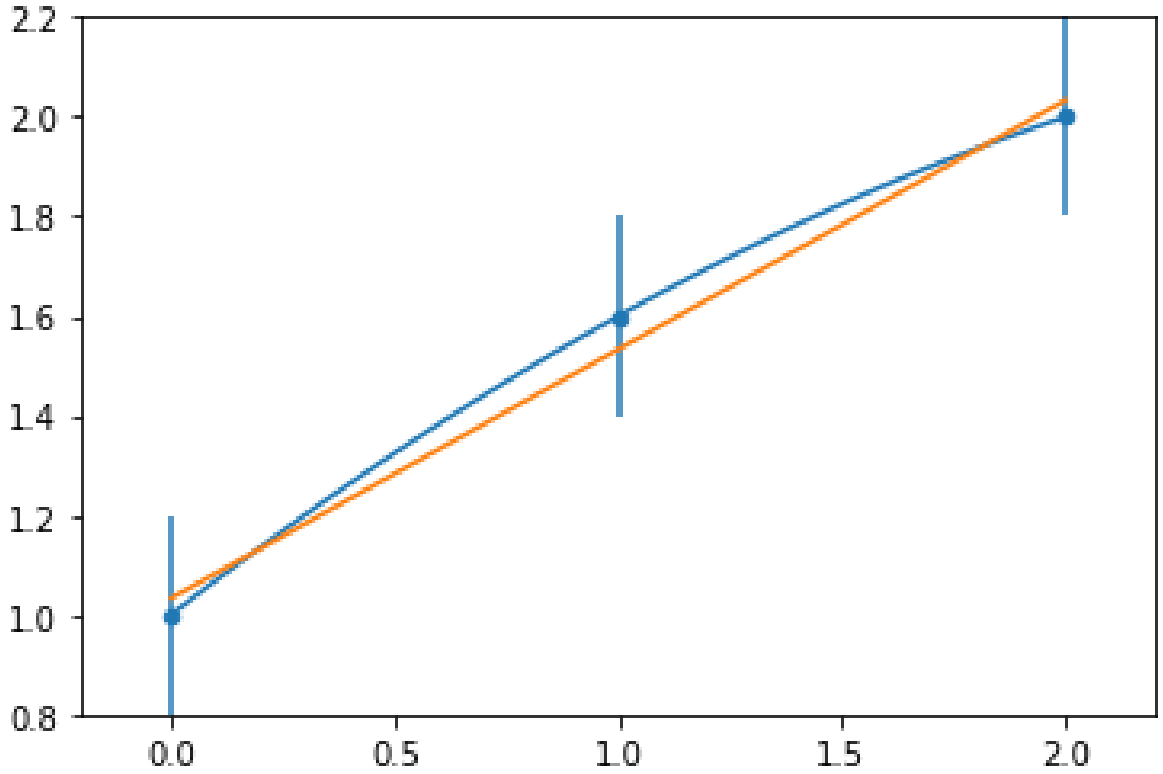
the complexity of a model can be measured by the number of variables and the numbers of parameters

model fit to data

$y = 0x + 0$

$y = 0x^2 + 0x + 0$

parameters



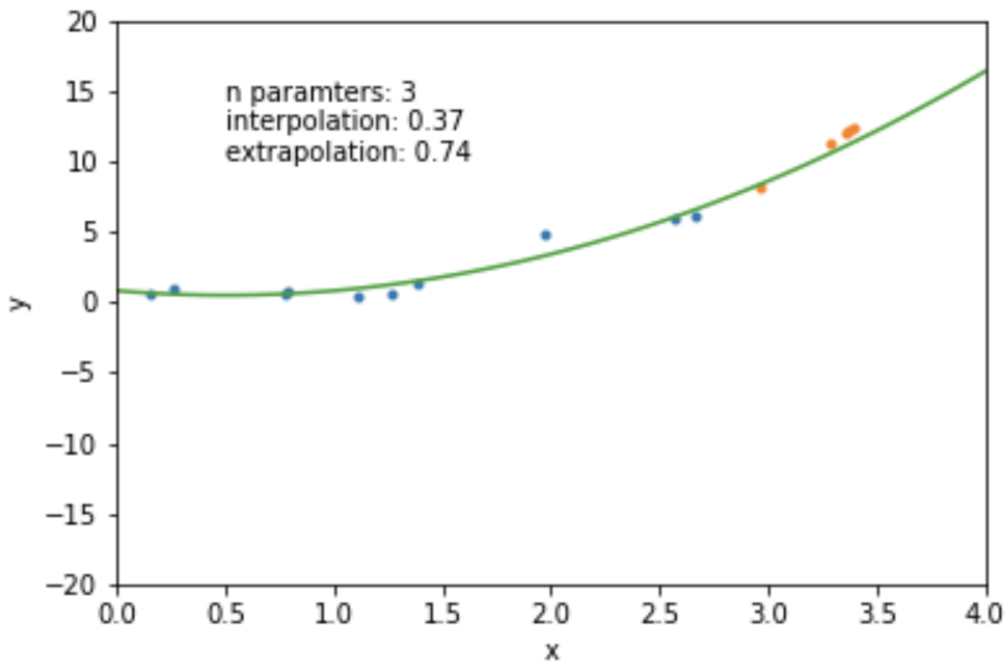
Okham's razor

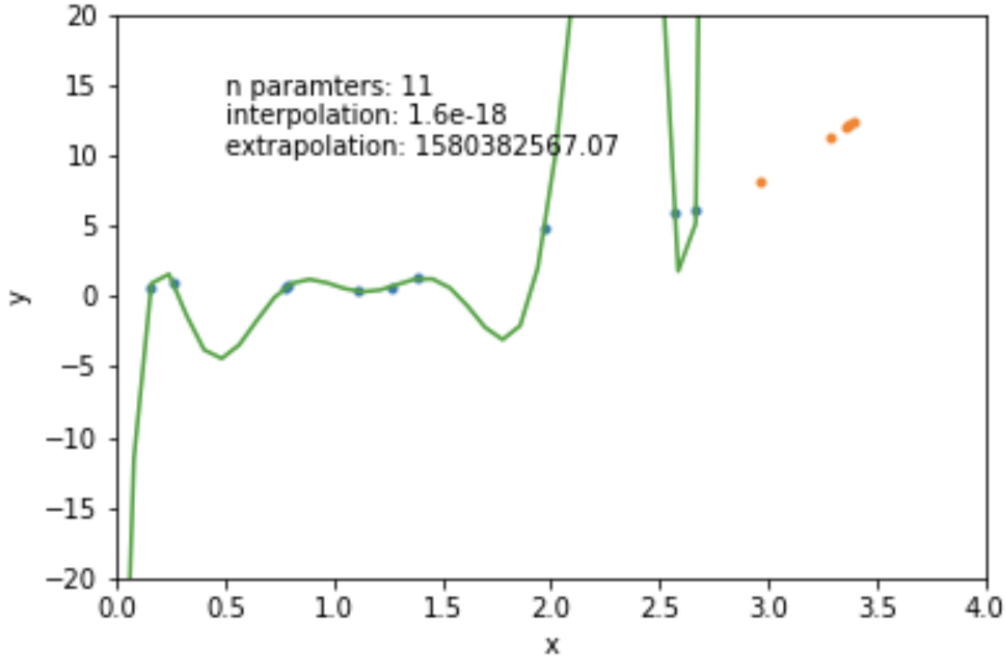
the complexity of a model can be measured by the number of variables and the numbers of parameters

mathematically: given N data points there exist an N-features model that goes exactly through each data point. but is it useul??

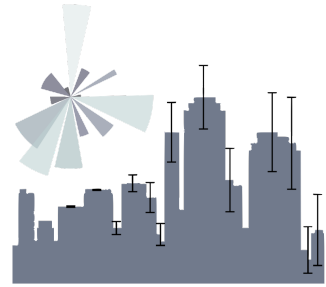
Overfitting: fitting data with a model that is too complex and that does not extend to new data (low predictive power on test data)

https://github.com/fedhere/PUS2020_FBianco/blob/master/classdemo/overfit_animation.ipynb

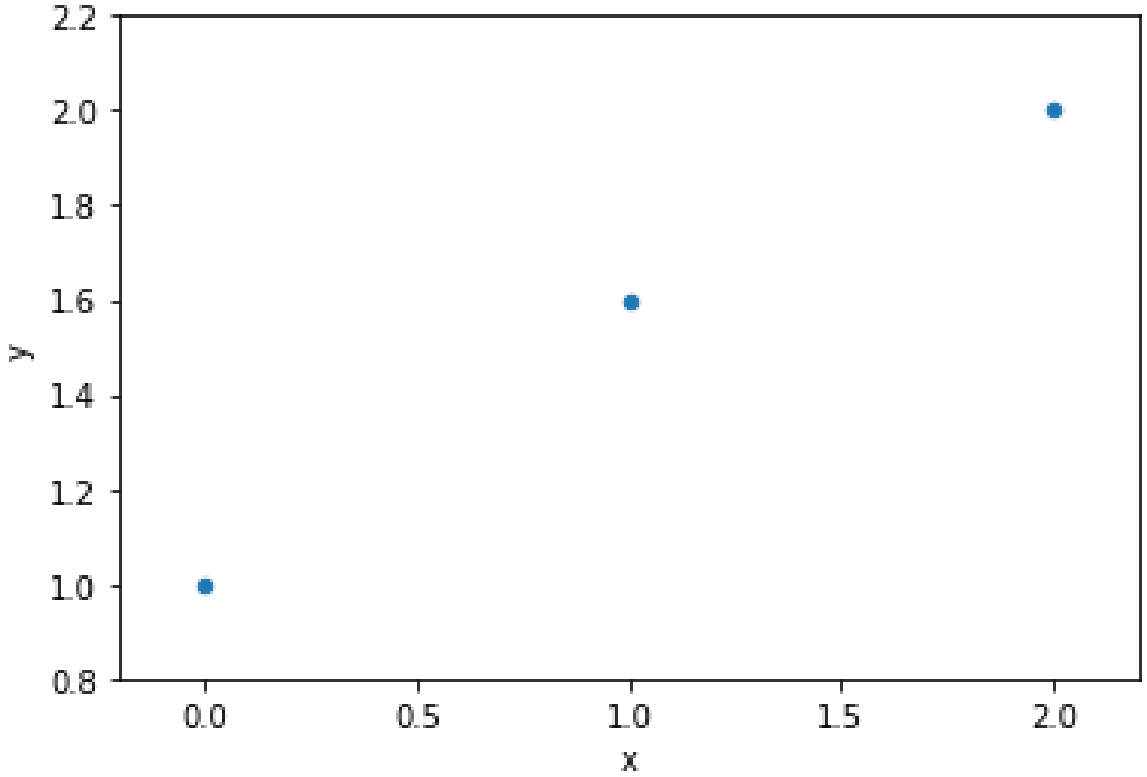




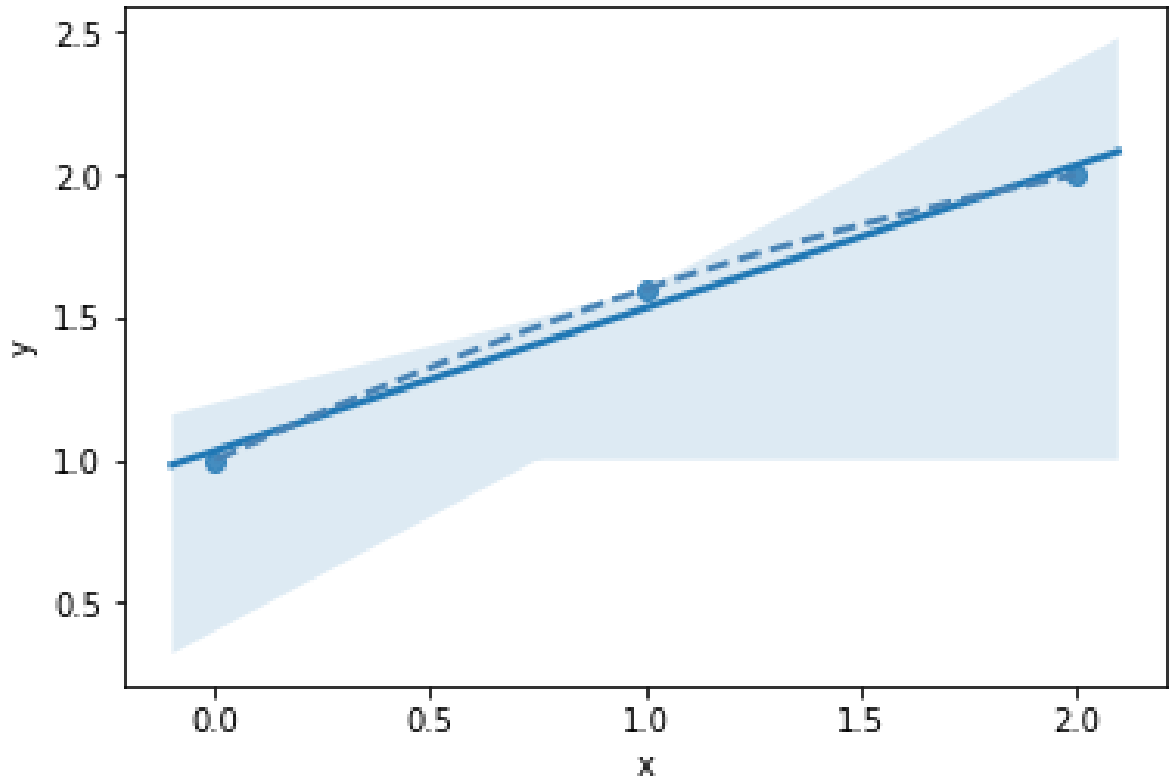
Okham's razor



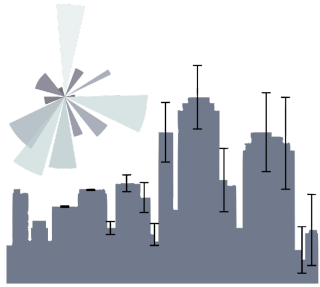
data



model fit to data



Okham's razor



https://nbviewer.jupyter.org/gist/fedhere/ef2da384b9e114267e8e93b7366e4ff6

Dep. Variable:	y	**R-squared:**	0.987
Model:	OLS	**Adj. R-squared:**	0.974
Method:	Least Squares	**F-statistic:**	75.00
Date:	Mon, 07 Sep 2020	**Prob (F-statistic):**	0.0732
Time:	09:49:48	**Log-Likelihood:**	4.9071
No. Observations:	3	**AIC:**	-5.814
Df Residuals:	1	**BIC:**	-7.617
Df Model:	1		

https://nbviewer.jupyter.org/gist/fedhere/ef2da384b9e114267e8e93b7366e4ff6

5 how to avoid overfitting in machine learning

Cross validation

Train and Test split

```
[46] from sklearn.linear_model import LinearRegression
     from sklearn.model_selection import train_test_split
     #  split train and test set
     X_train, X_test, y_train, y_test = train_test_split(
       x, y, test_size=0.33, random_state=302)
```

```
[47] # fit to training data
     lm = LinearRegression().fit(X_train,
                                 y_train, sample_weight=None)
```

```
[50] # in sample score
     print("linear regression in-sample score: {:.0f}% ".format(
         lm.score(X_train,  y_train) * 100))
```

```
linear regression in-sample score: 88%
```

```
# out of sample score
print("linear regression ot-of-sample score: {:.0f}% ".format(
    lm.score(X_test,  y_test) * 100))
```

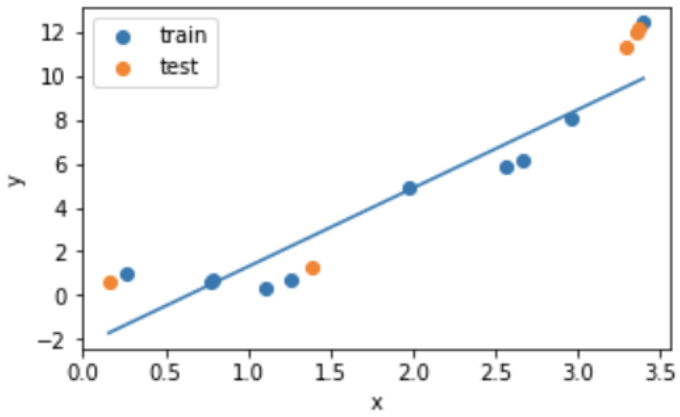
```
linear regression ot-of-sample score: 85%
```

poor model: - low in-sample score

overfitting: - high in-sample score, low out-of-sample score

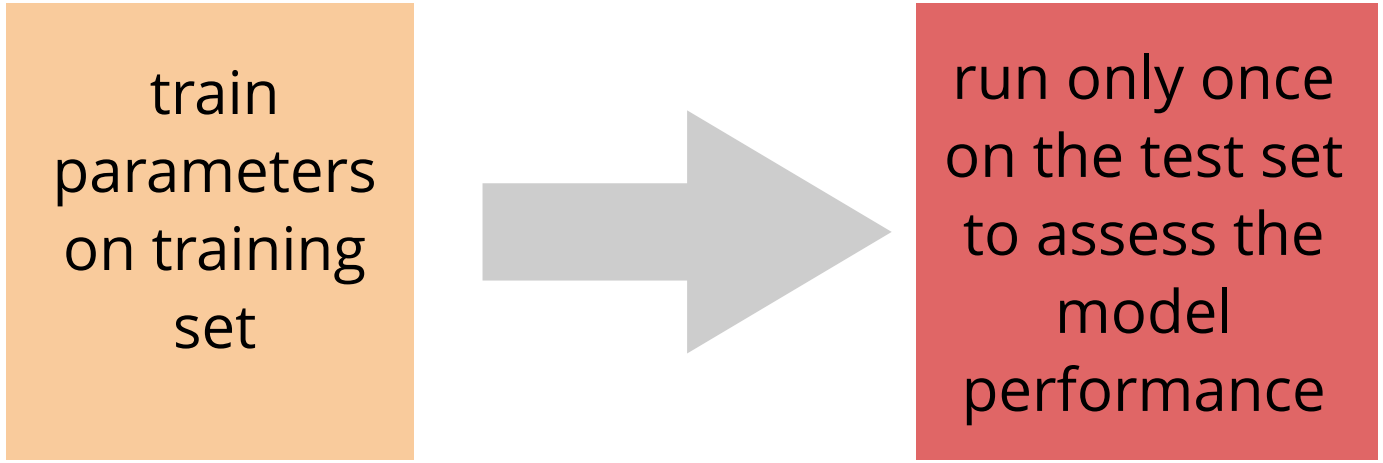
a small drop in score, like the one seen above, is normal

```
plt.figure(figsize=(5,3))
plt.scatter(X_train, y_train, label="train")
plt.scatter(X_test, y_test, label="test")
plt.plot(x, lm.predict(x))
plt.xlabel('x')
plt.ylabel('y')
plt.legend();
```



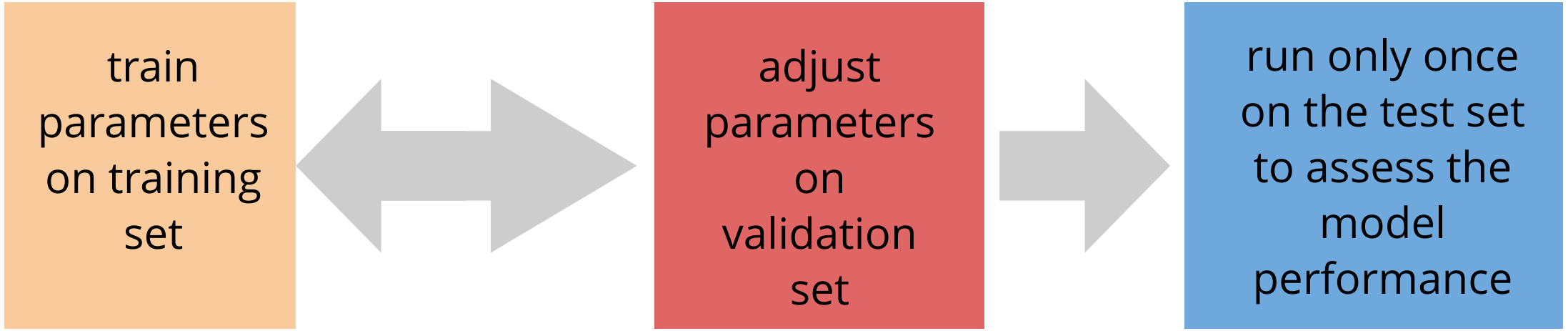
Cross validation

test train validation



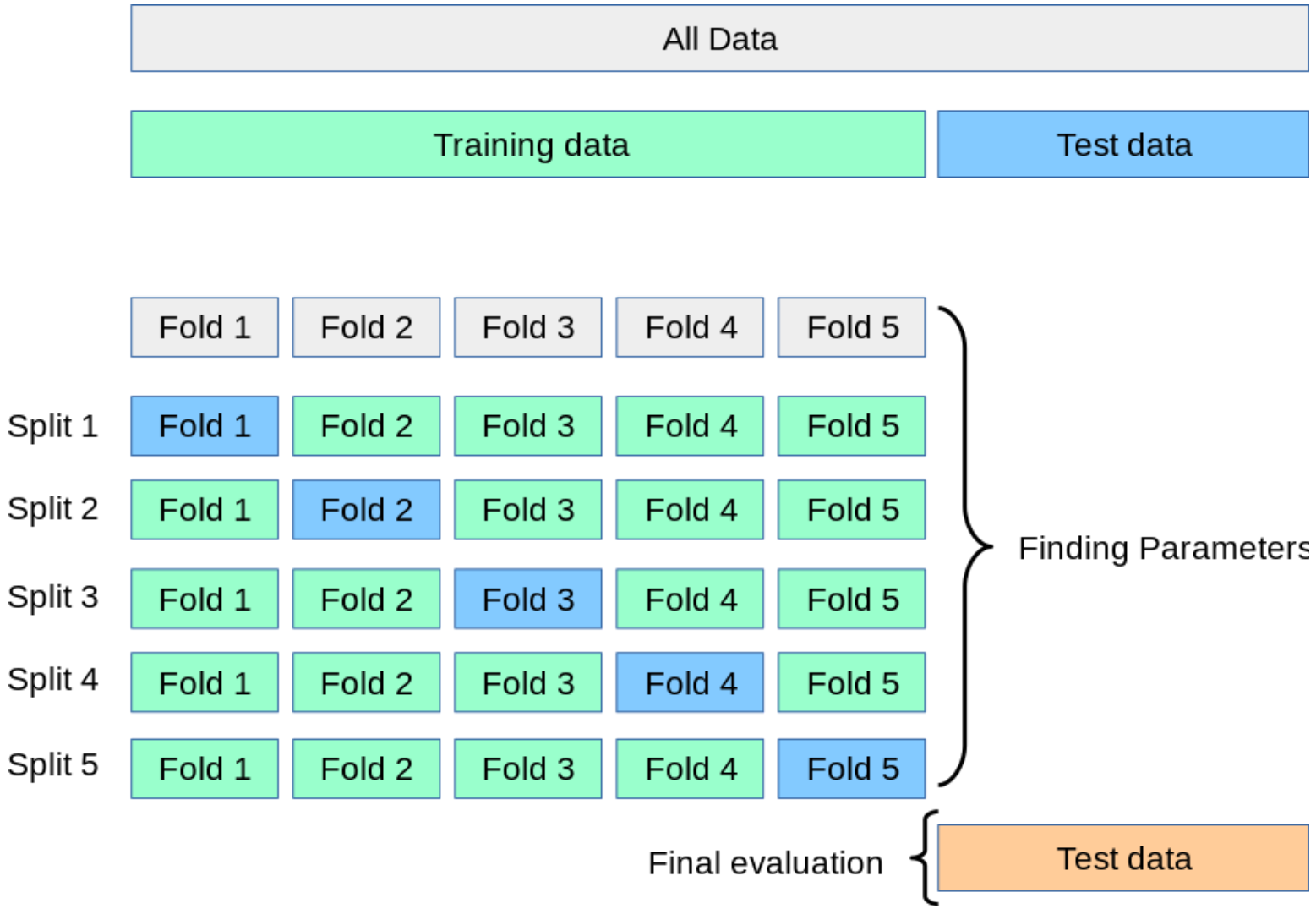
Cross validation

test + train + validation

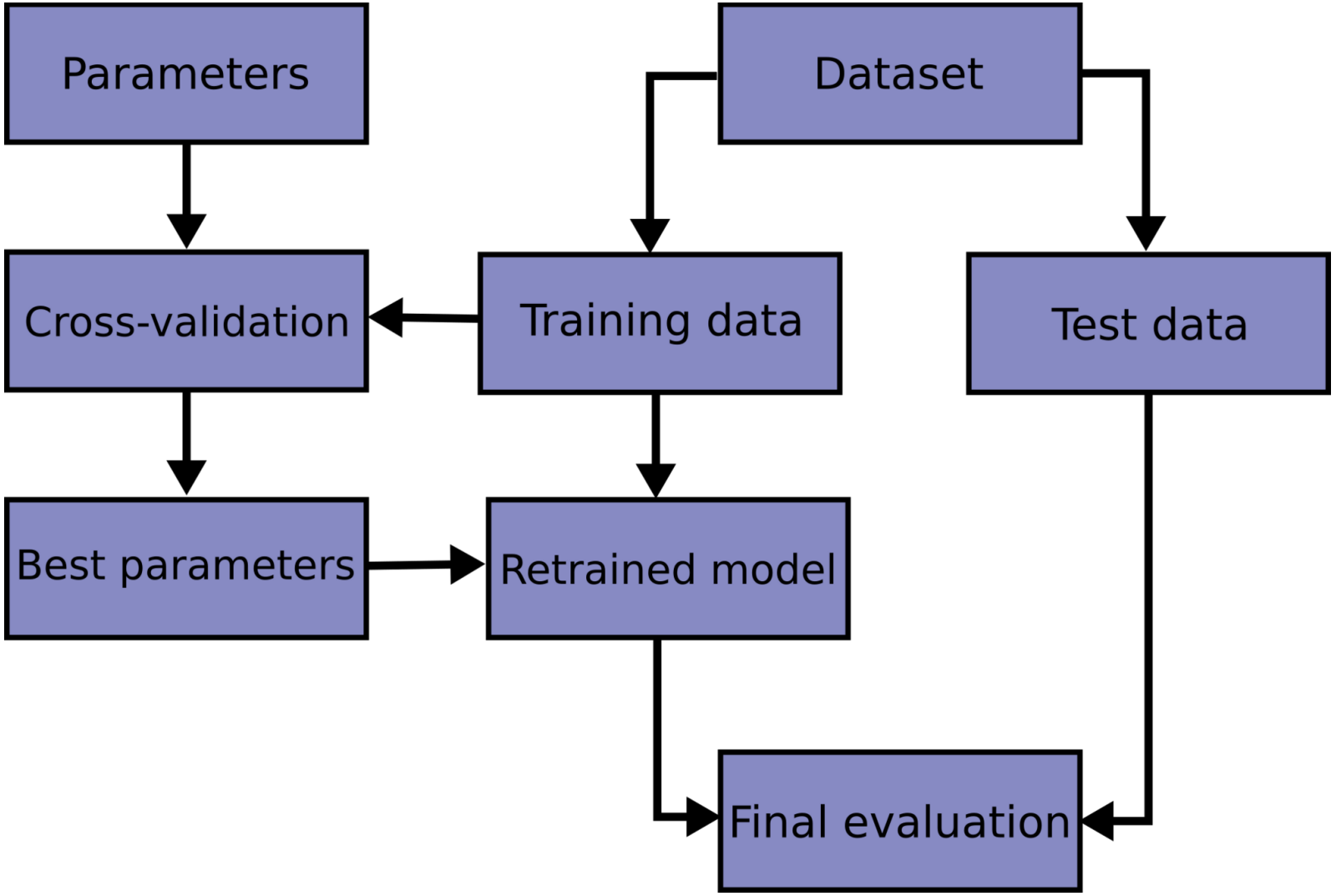


Cross validation

k-fold cross validation



Cross validation



https://scikit-learn.org/stable/modules/cross_validation.html

NEXT WEEK:

10/5 : **MIDTERM in class**

what is covered in the midterm? everything we do until 9/31!

how is the midterm graded?

- data ingestion (30%)
- data exploration (descriptive statistics) (50%)
- NHRT (P-value tests) (20%)
-

RULES: work on your on, ALL CAMERAS MUST BE ON

The grading scheme reflects the complexity of the tasks and the expectation of how well they have been absorbed based on how long we have been working on it

key concepts

line models and polynomial models

parameters and hyperparameters

objective function: what do we minimize to choose parameters?

model diagnostics and choosing models

influence points

cross validation

references

this is a pretty comprehensive and clear medium post with coding examples

https://medium.com/@kylecaron/introduction-to-linear-regression-part-1-implementation-in-python-with-statsmodels-7dbf24461072

this is a whole paper about modeling in practice and in theory which covers extensively how to deal with uncertainties. it is not for the faint of heart. The notes are entertaining

https://arxiv.org/abs/1008.4686

https://www.nature.com/articles/nmeth.3812.pdf

POINTS OF SIGNIFICANCE

Analyzing outliers: influential or nuisance?

Some outliers influence the regression fit more than others.

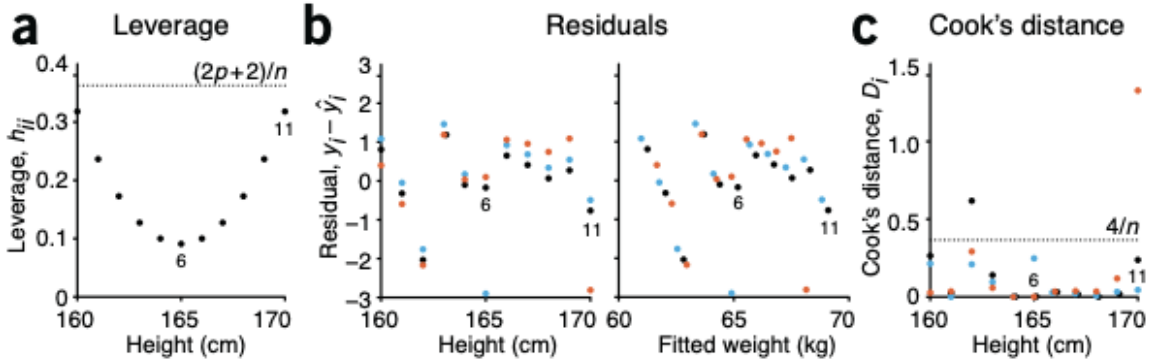


Figure 2 | The leverage, residual and Cook's distance of an observation are used to assess the robustness of the fit. (**a**) The leverage of an observation tells us about its potential to influence the fit and increases as the square