Welcome to Week #13!

K-Nearest Neighbors

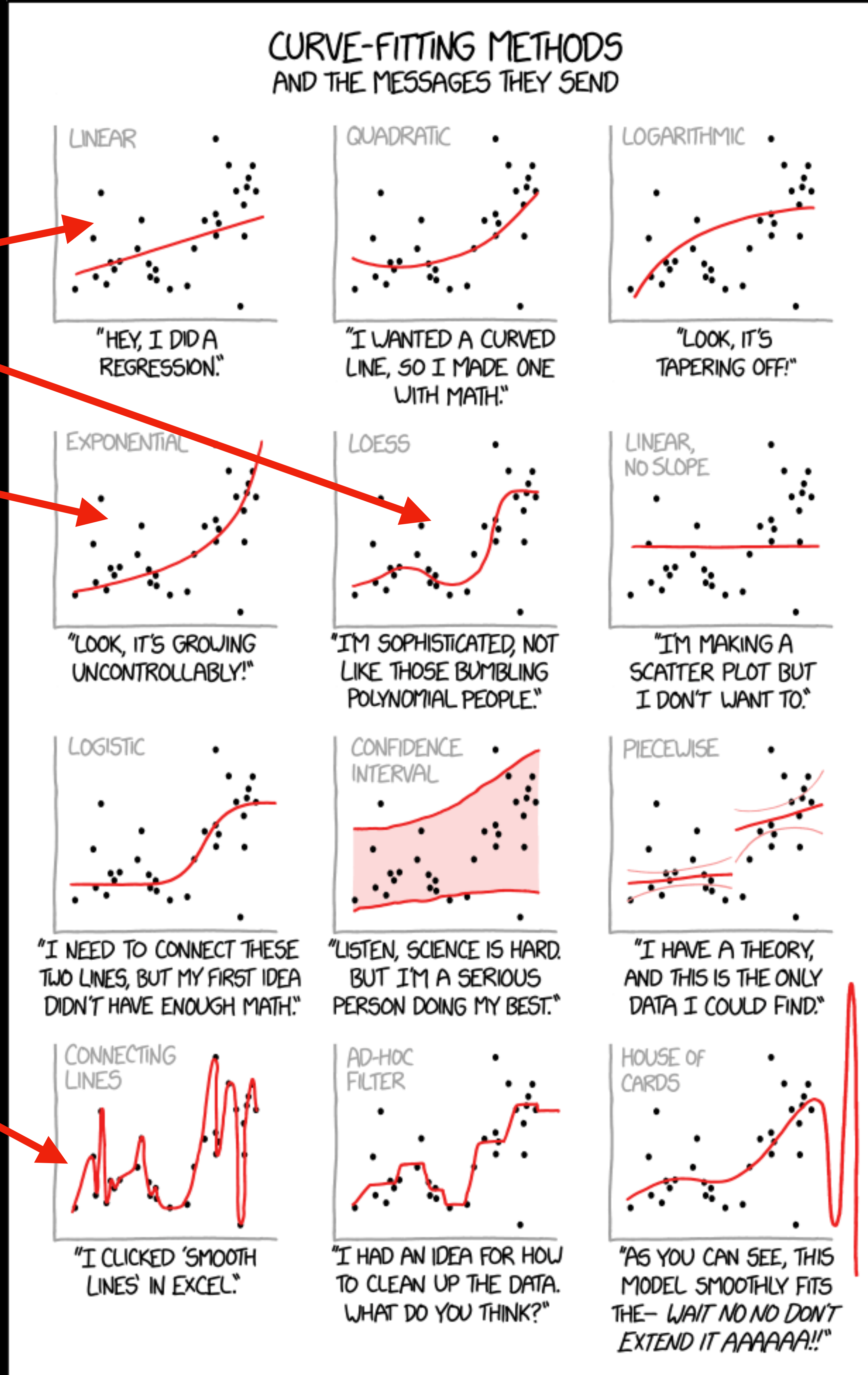
K-Nearest Neighbors

First: an intro to overfitting

Overfit?

Overfit?

Underfit?



Overfit

Is overfitting or underfitting worse?

Bias-Variance Trade-Off (First Glance)

$$E\left(y_0 - \hat{f}(x_0)\right)^2 = \mathrm{Var}(\hat{f}(x_0)) + [\mathrm{Bias}(\hat{f}(x_0))]^2 + \mathrm{Var}(\epsilon)$$

Bias-Variance Trade-Off (First Glance)

$$E\left(y_0 - \hat{f}(x_0)\right)^2 = \mathrm{Var}(\hat{f}(x_0)) + [\mathrm{Bias}(\hat{f}(x_0))]^2 + \mathrm{Var}(\epsilon)$$

mean square error if we kept estimating response variable y by our fitted function of our explanatory variables, f(x) with different sample datasets at point x_0

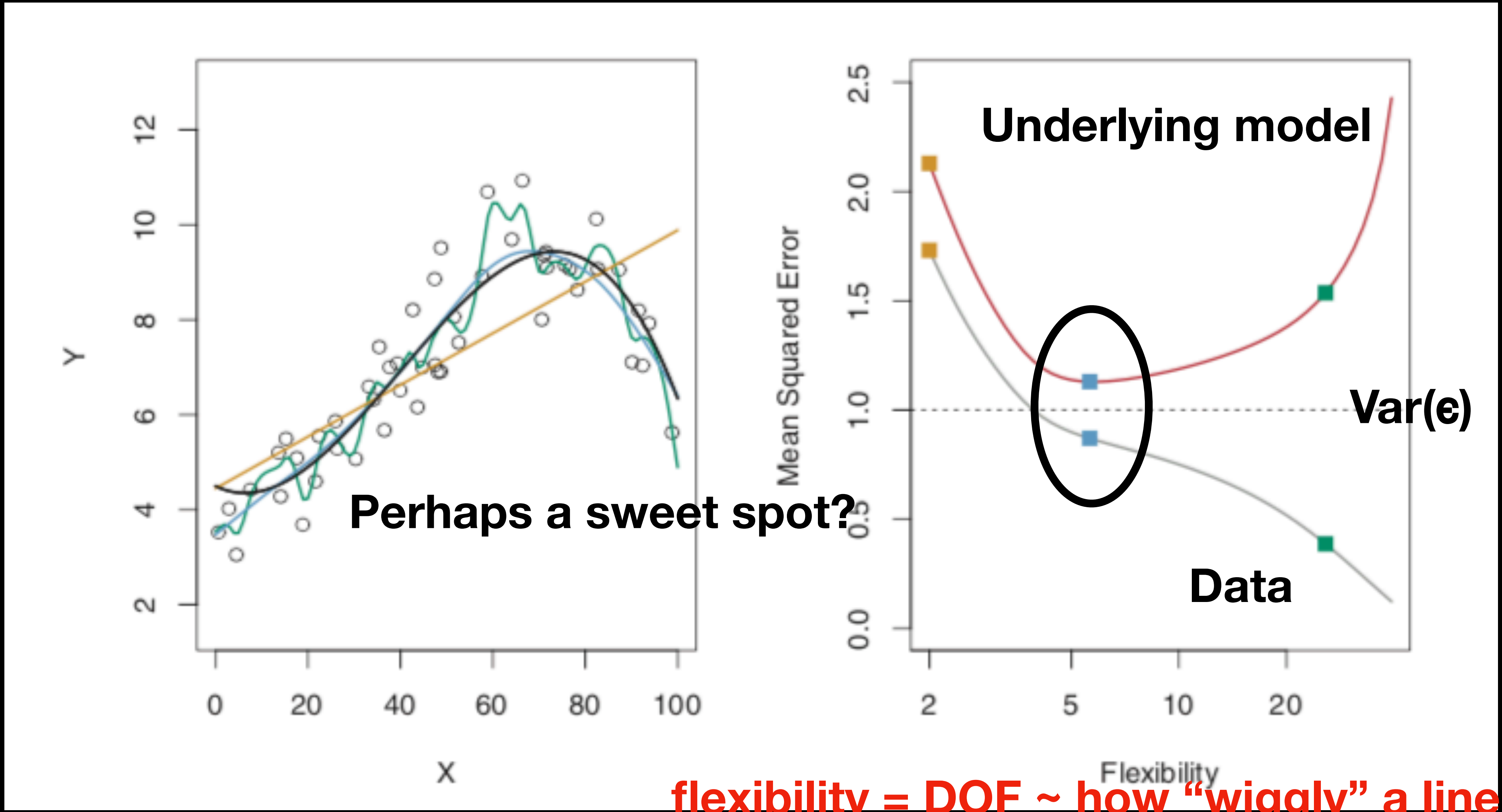
Inherent error in our measurements

Inherent error (bias) in the fact that any model is only an approximation to reality

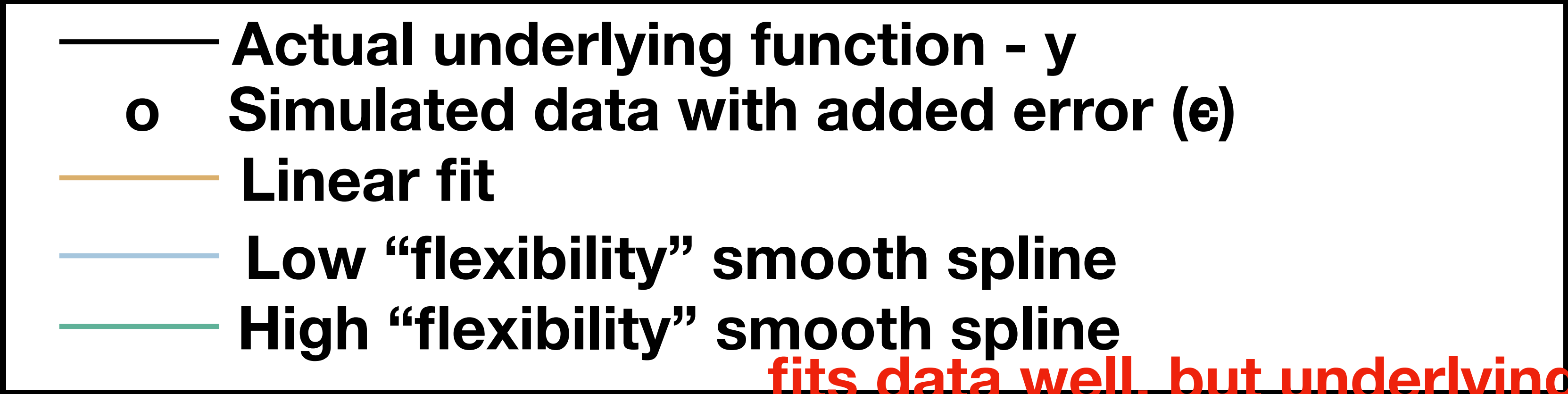
how much our function, f, changes if we use a different random sample (variance)

Bias-Variance Trade-Off (First Glance)

$$E\left(y_0 - \hat{f}(x_0)\right)^2 = \mathrm{Var}(\hat{f}(x_0)) + [\mathrm{Bias}(\hat{f}(x_0))]^2 + \mathrm{Var}(\epsilon).$$



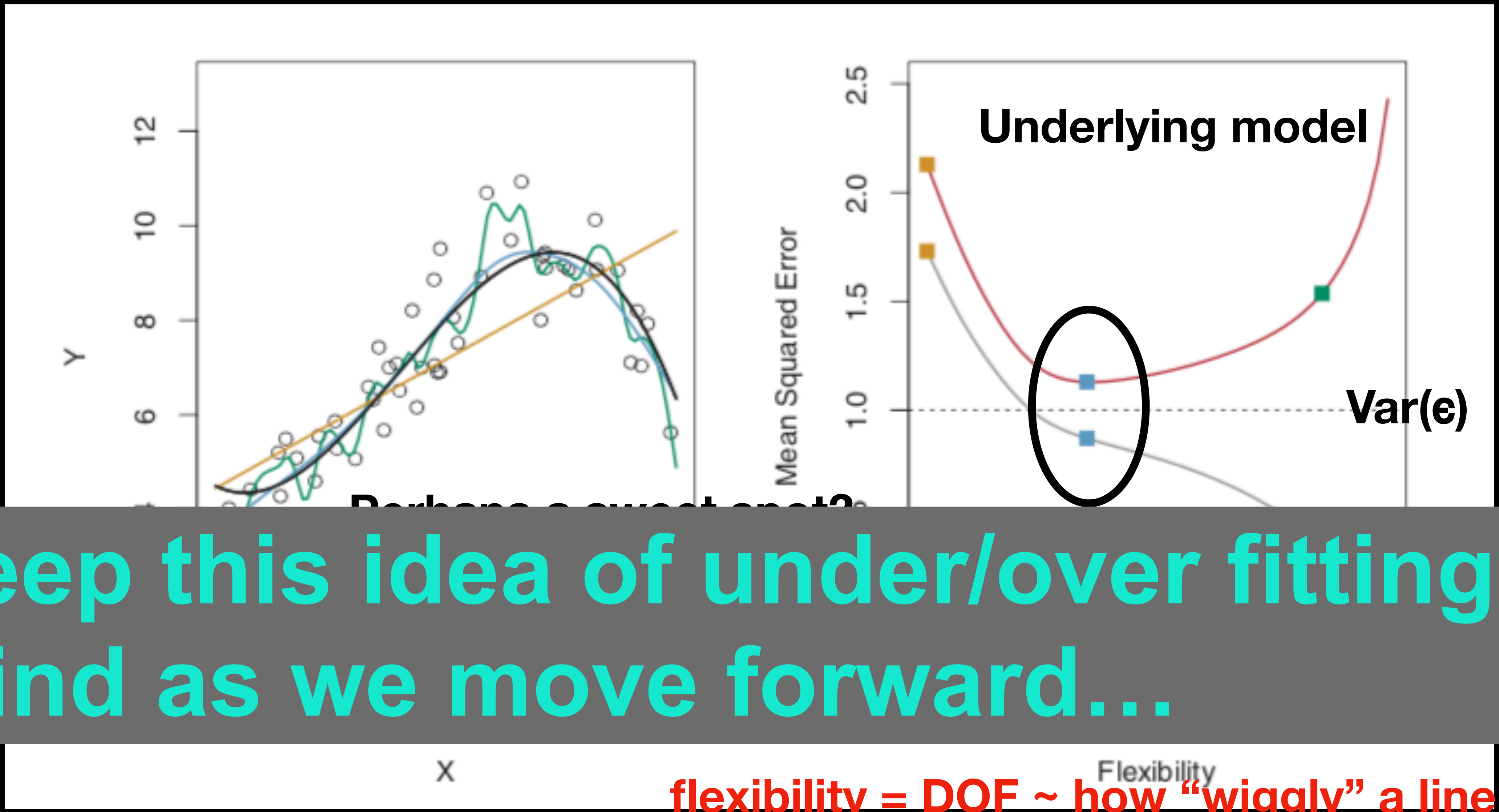
flexibility = DOF ~ how “wiggly” a line is



fits data well, but underlying model badly

Bias-Variance Trade-Off (First Glance)

$$E\left(y_0 - \hat{f}(x_0)\right)^2 = \mathrm{Var}(\hat{f}(x_0)) + [\mathrm{Bias}(\hat{f}(x_0))]^2 + \mathrm{Var}(\epsilon).$$



Keep this idea of under/over fitting in mind as we move forward…

flexibility = DOF ~ how “wiggly” a line is

Actual underlying function - y

o Simulated data with added error (ϵ)

Linear fit

Low “flexibility” smooth spline

High “flexibility” smooth spline

fits data well, but underlying model badly

K-Nearest Neighbors

~~First: an intro to overfitting~~

So far…

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \ldots + \beta_n x_n$$

predicted y

intercept

So far…

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \ldots + \beta_n x_n$$

predicted y

intercept

So far we've been saying:
"I have a basic idea of what the functional form of y looks like (a line or a logit) - computer go find the parameters of that functional form.

So far…

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \ldots + \beta_n x_n$$

predicted y

intercept

So far we've been saying:
"I have a basic idea of what the functional form of y looks like (a line or a logit) - computer go find the parameters of that functional form.

This is nice because we have some hope of gaining intuition from our models.

So far…

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \ldots + \beta_n x_n$$

predicted y

intercept

So far we've been saying:
"I have a basic idea of what the functional form of y looks like (a line or a logit) - computer go find the parameters of that functional form.

This is nice because we have some hope of gaining intuition from our models.

Now we classify…

So far…

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \ldots + \beta_n x_n$$

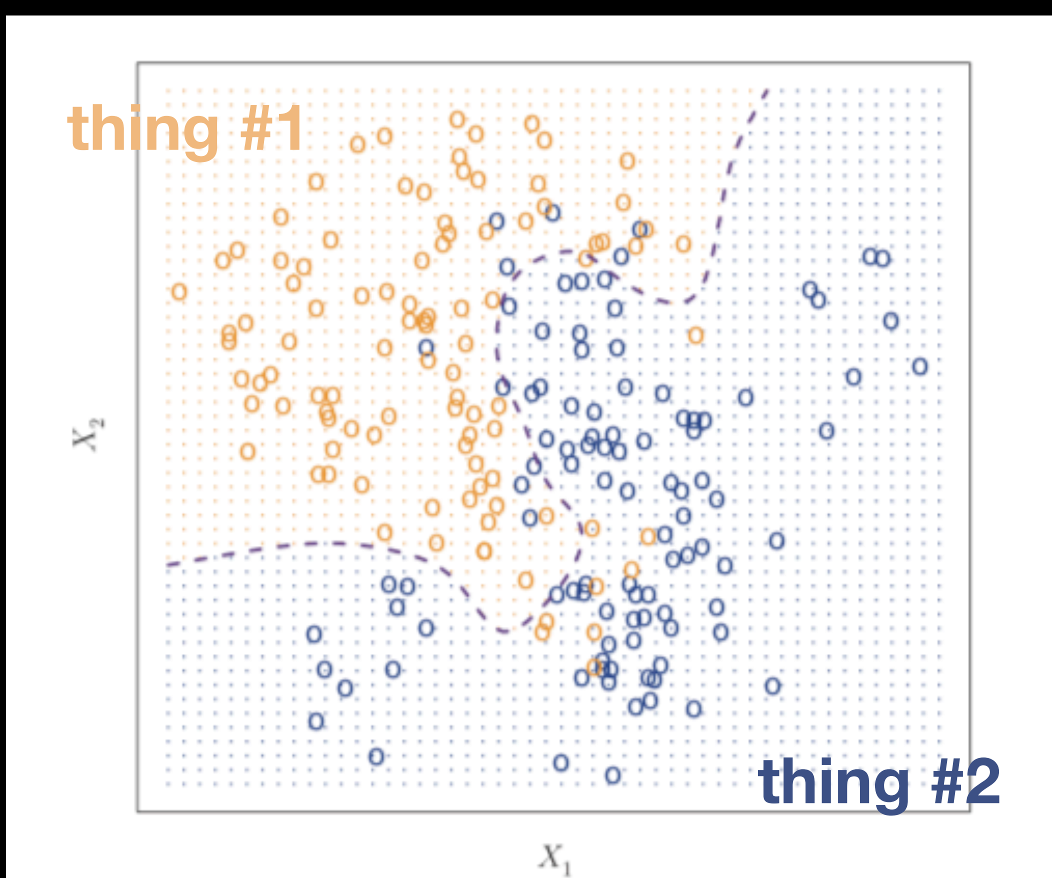
predicted y

intercept

**So far we've been saying:
"I have a basic idea of what the functional form of y looks like (a line or a logit) - computer go find the parameters of that functional form.**

This is nice because we have some hope of gaining intuition from our models.

Now we classify…



"I want to know where thing #1 and thing #2 live in some 2D space - computer, go figure out the boundary between these two things and let me know"

So far…

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \ldots + \beta_n x_n$$

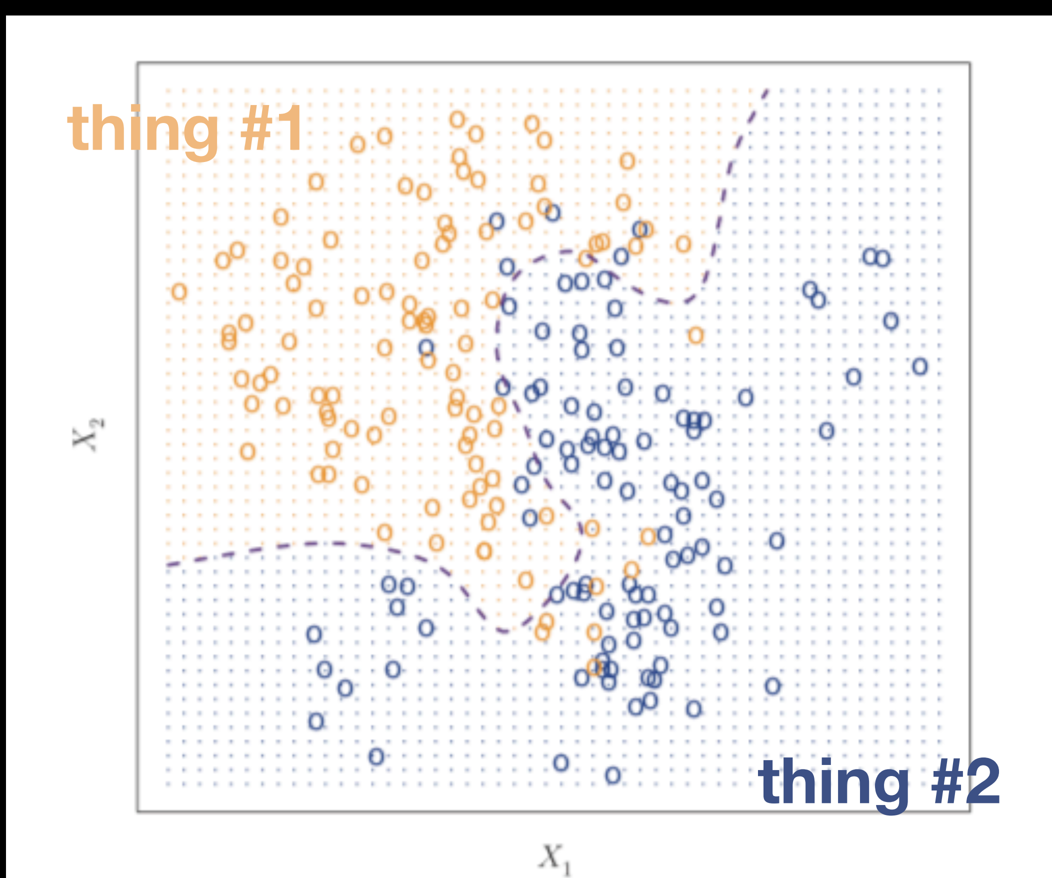
predicted y

intercept

So far we've been saying:
"I have a basic idea of what the functional form of y looks like (a line or a logit) - computer go find the parameters of that functional form.

This is nice because we have some hope of gaining intuition from our models.

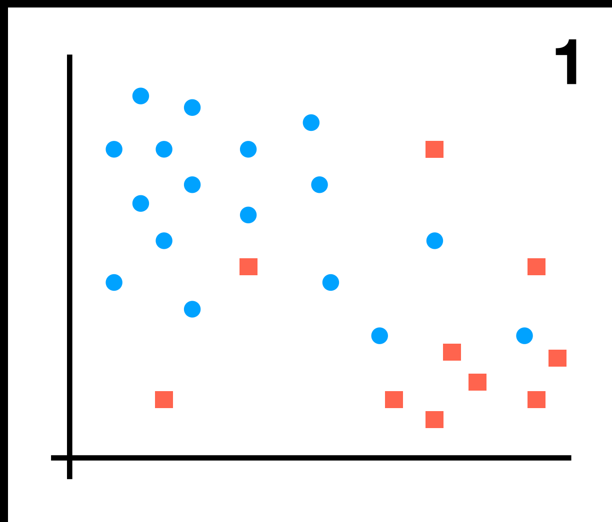
Now we classify…



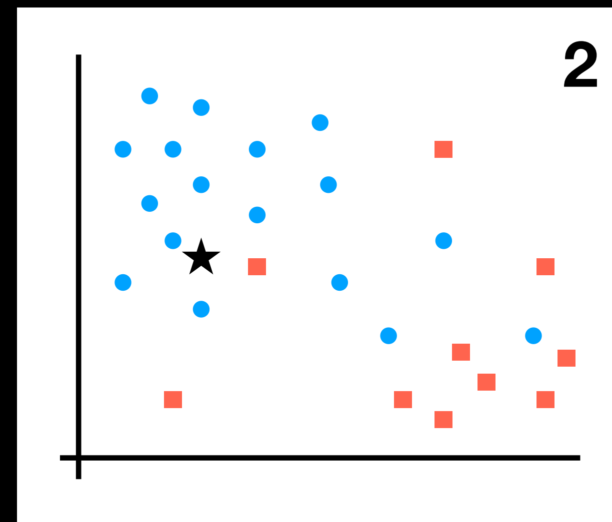
"I want to know where thing #1 and thing #2 live in some 2D space - computer, go figure out the boundary between these two things and let me know"

This is nice because we don't have to assume some model beforehand.

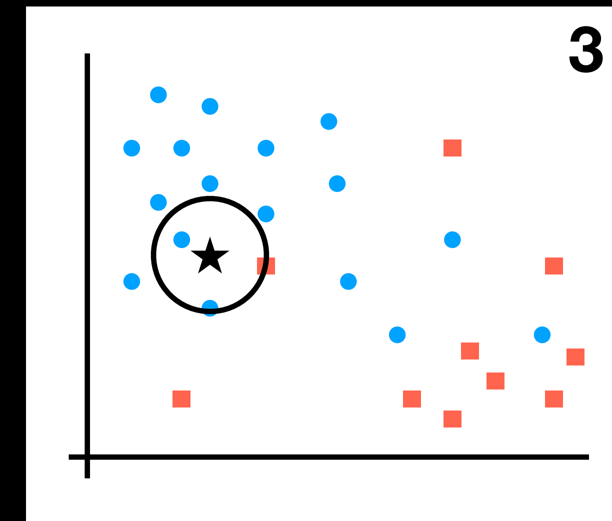
K Nearest Neighbors, in pictures



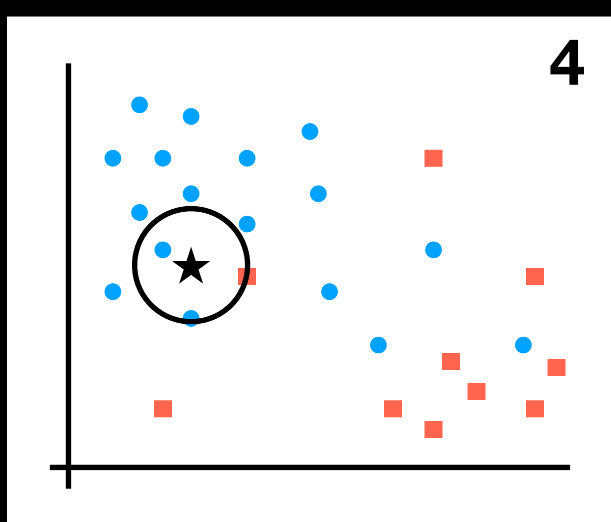
Sample (training) data representing underlying population



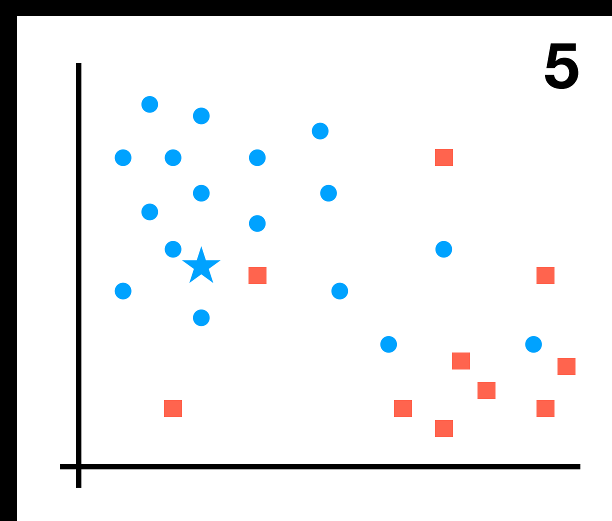
New point of interest



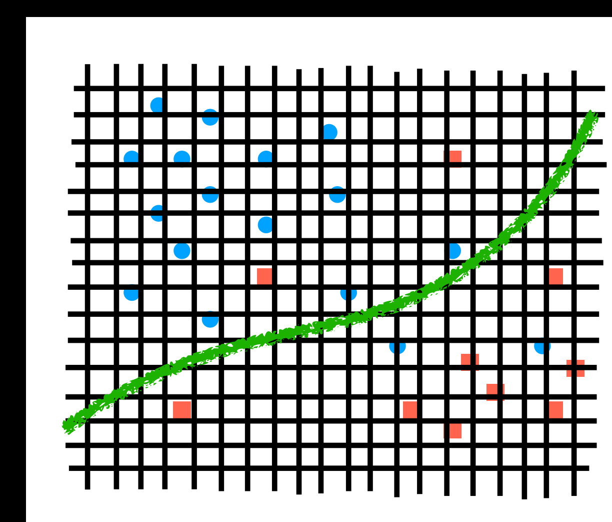
Find k nearest neighbors (here k = 3)



count “types” - here 2/3 points are blue
P(blue) = 2/3
P(red) = 1/3

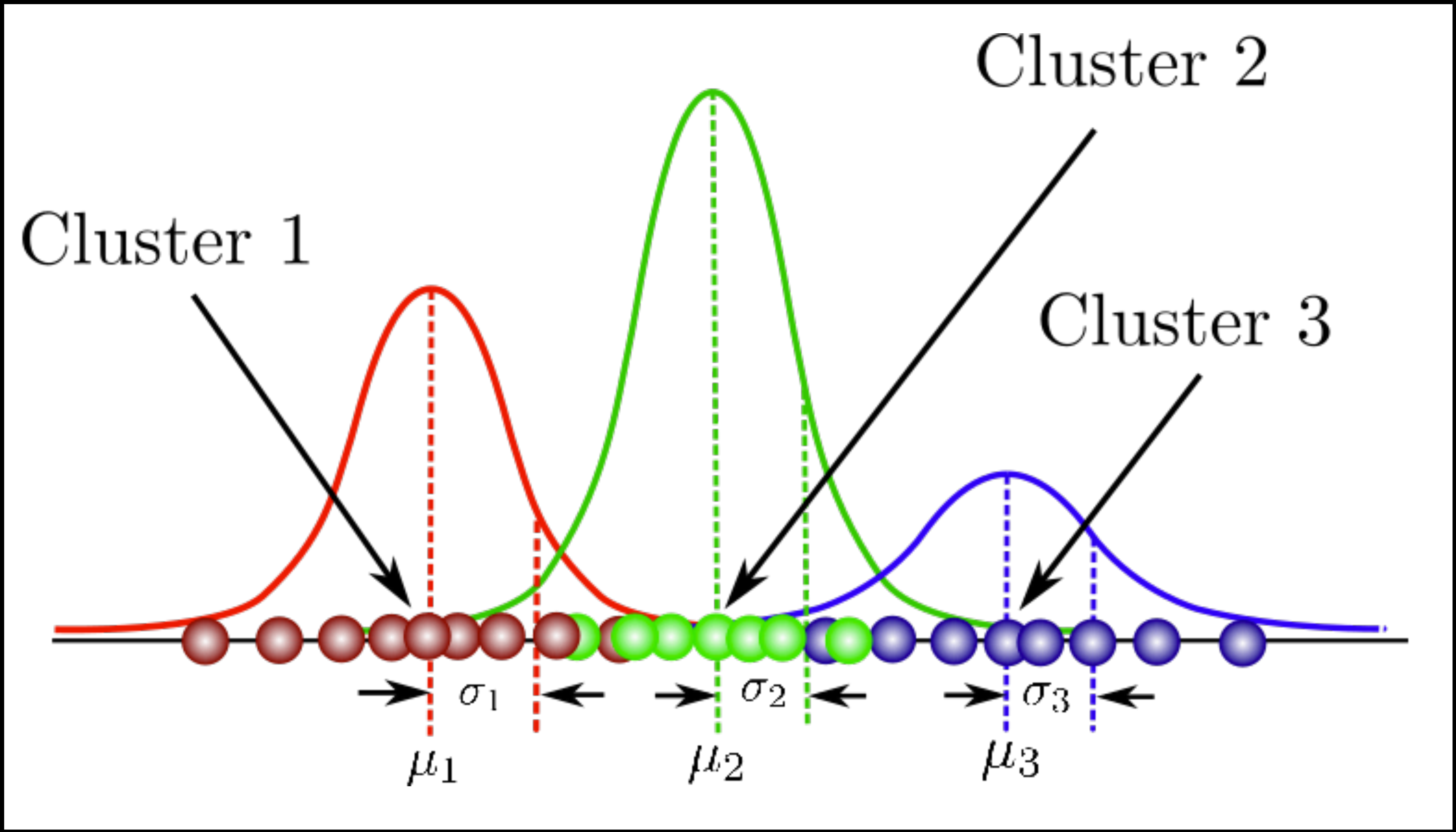


if P > cut off say new point is in that group here: P(blue) > 0.5

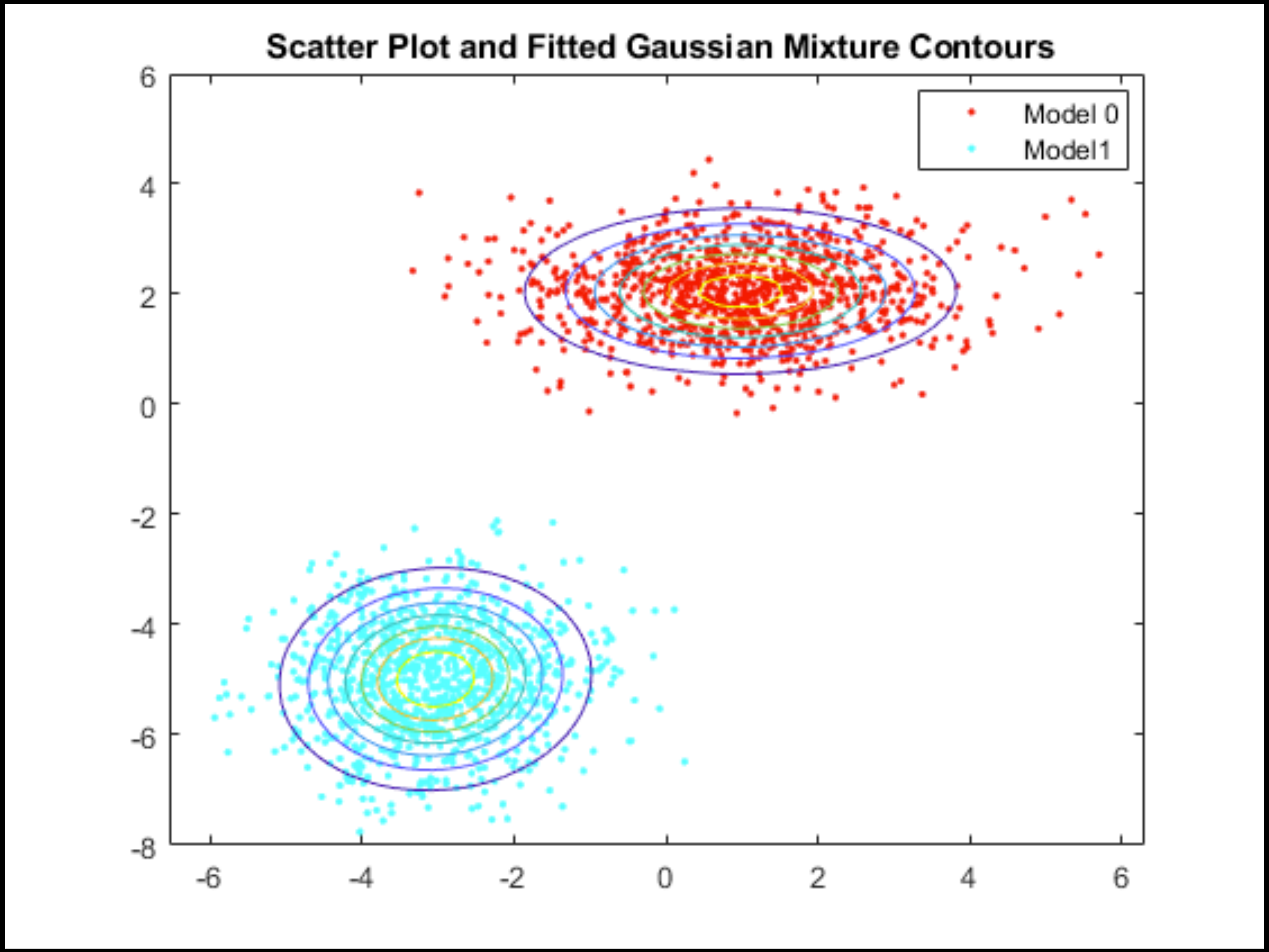


Repeat 2-3 on a grid & draw a separating line

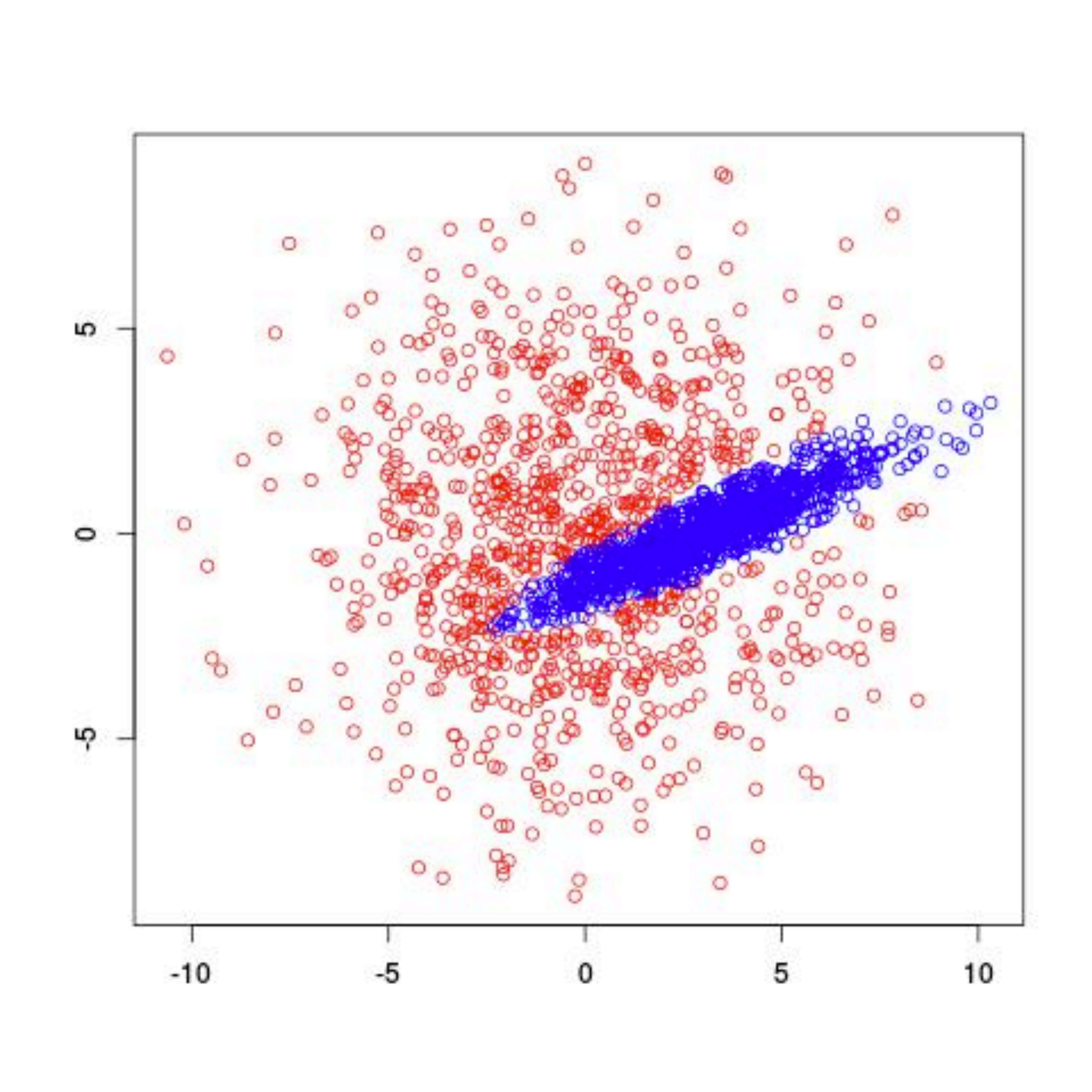
K Nearest Neighbors - with Gaussian Mixture Models.



K Nearest Neighbors - with Gaussian Mixture Models.

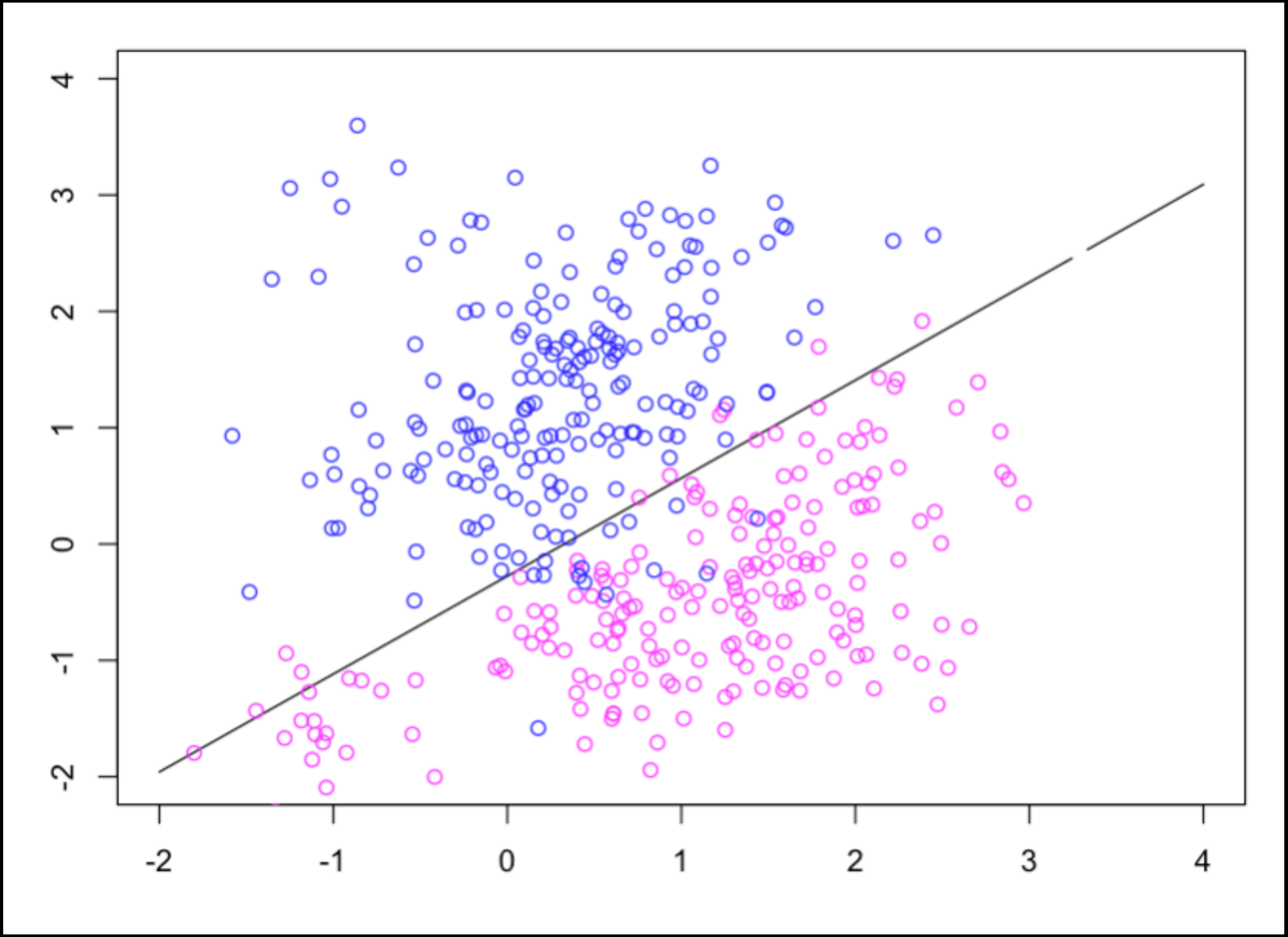


K Nearest Neighbors - with Gaussian Mixture Models.

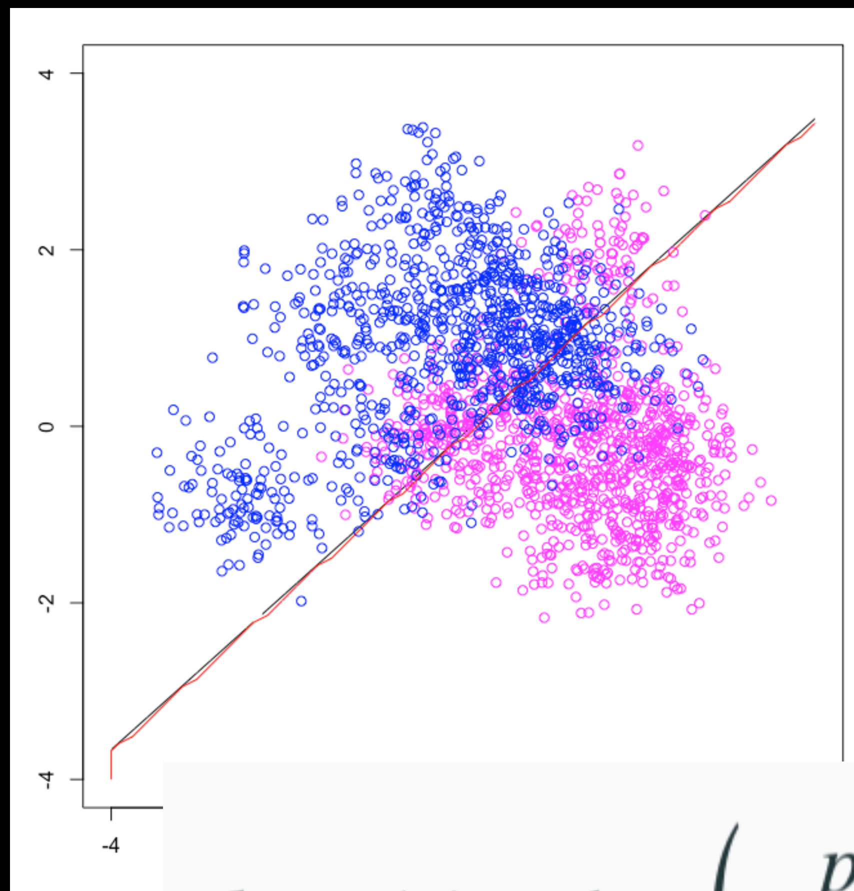


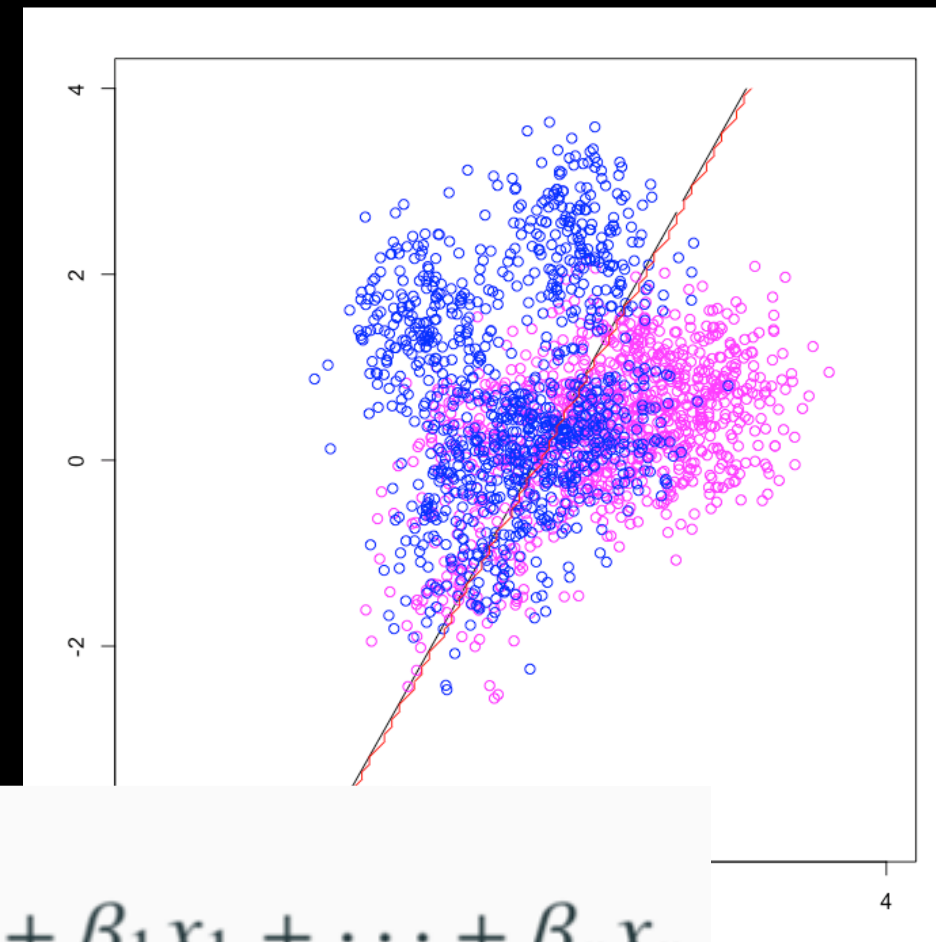
K Nearest Neighbors, in R!

K Nearest Neighbors, in R!
- just kidding logistic regression

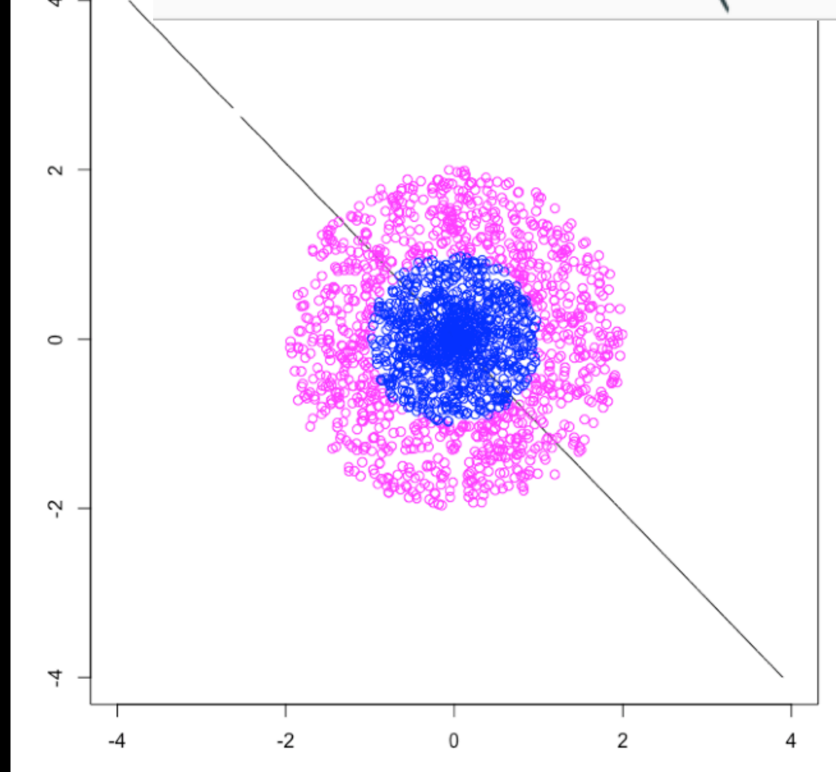


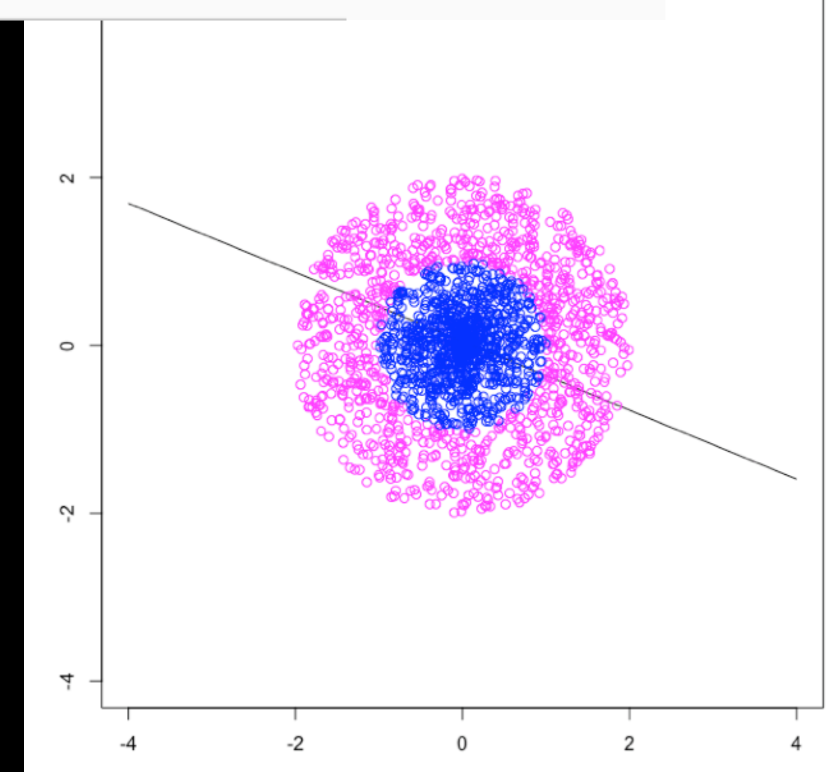
GLM is clearly getting something wrong



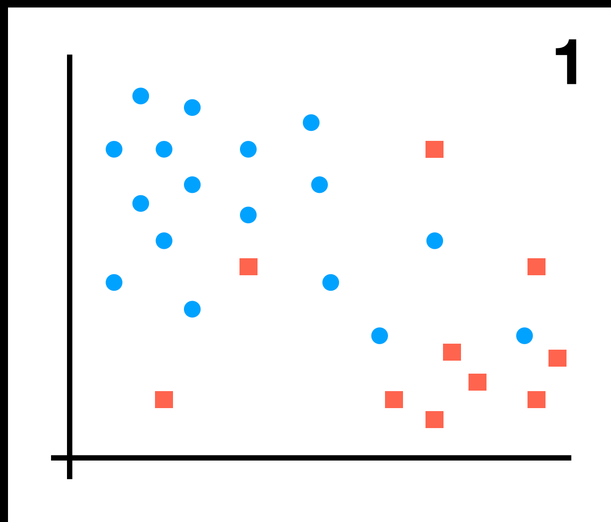


$$logit(p) = \log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n$$

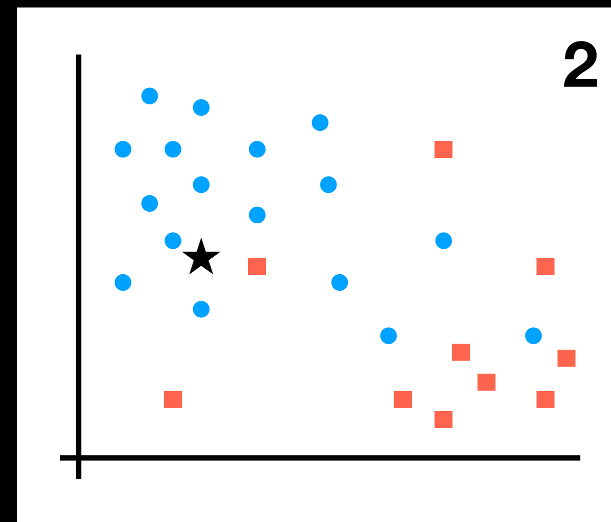




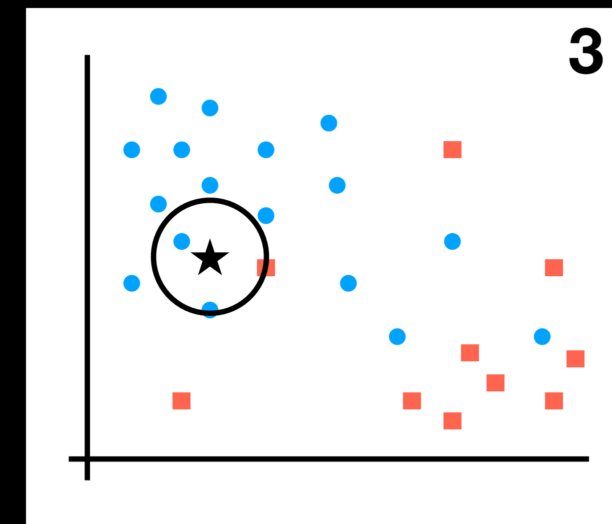
K Nearest Neighbors, in pictures



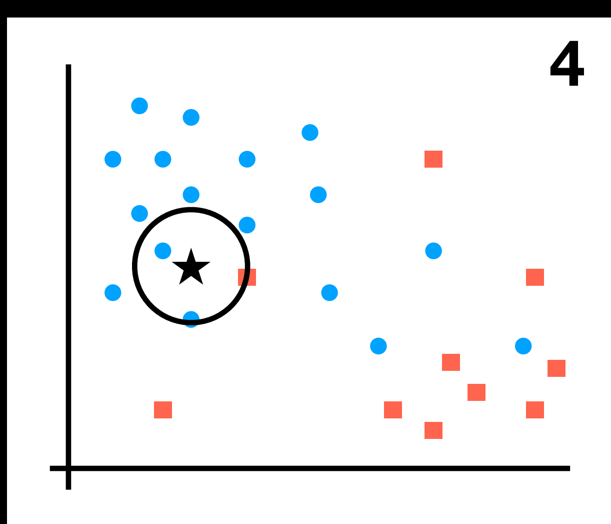
Sample (training) data representing underlying population



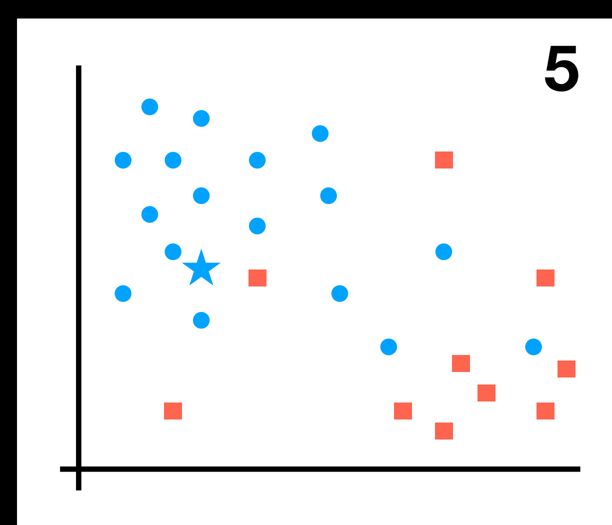
New point of interest



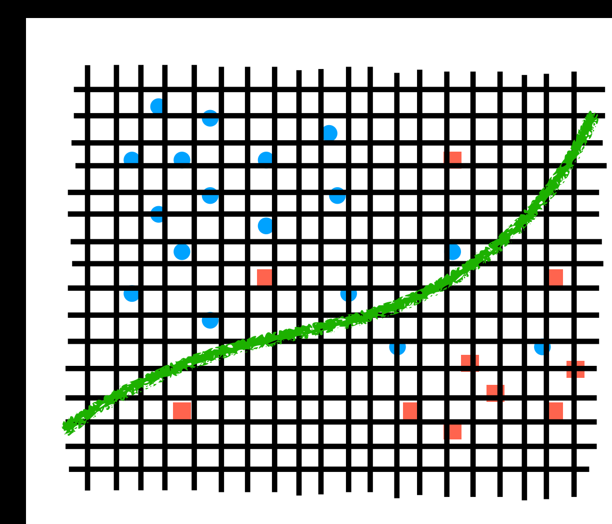
Find k nearest neighbors (here k = 3)



count “types” - here 2/3 points are blue
P(blue) = 2/3
P(red) = 1/3



if P > cut off say new point is in that group here: P(blue) > 0.5



Repeat 2-3 on a grid & draw a separating line

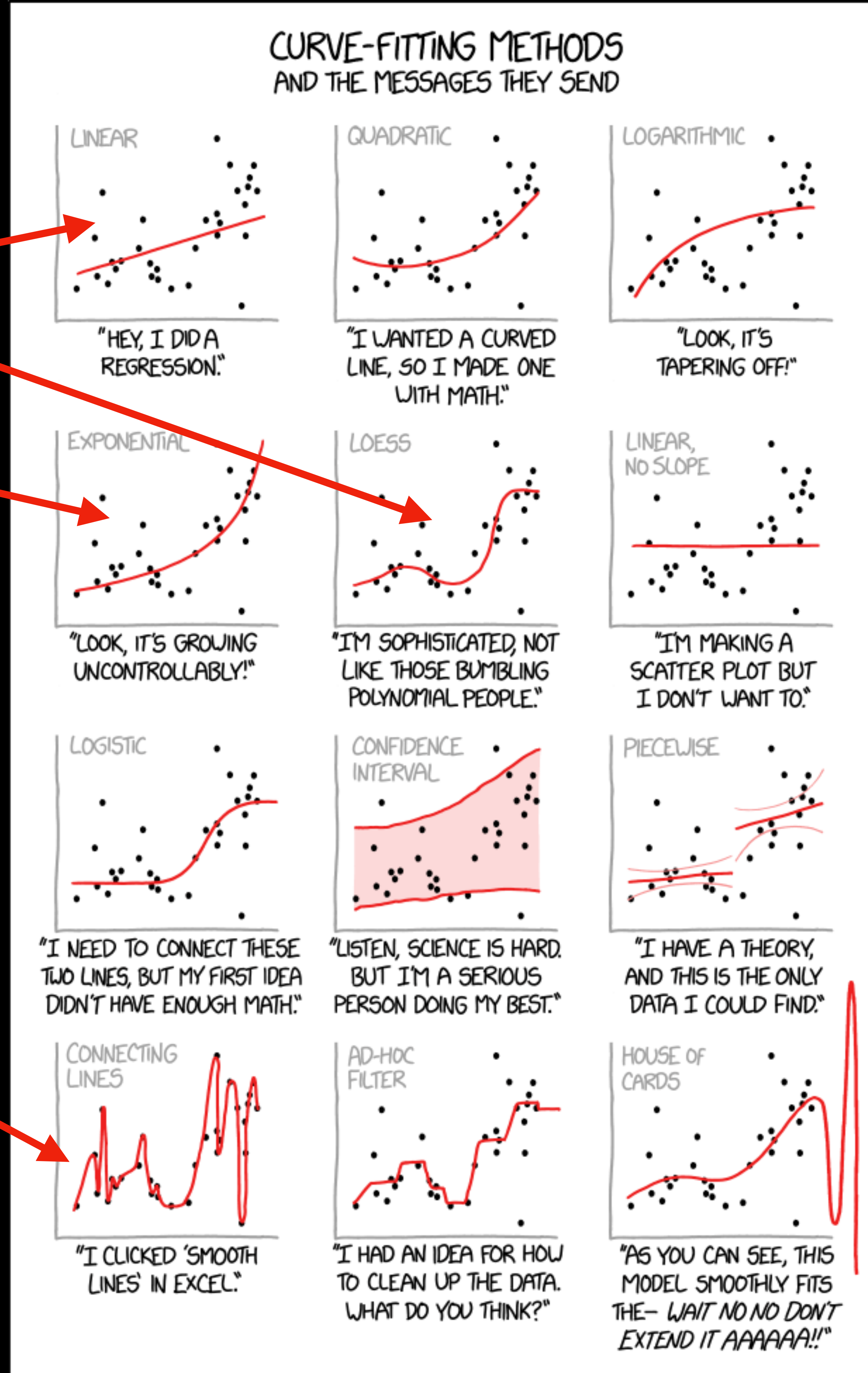
K Nearest Neighbors, in R!
For real this time!

Over/Under fitting - Quantifying how good your model is

Overfit?

Overfit?

Underfit?

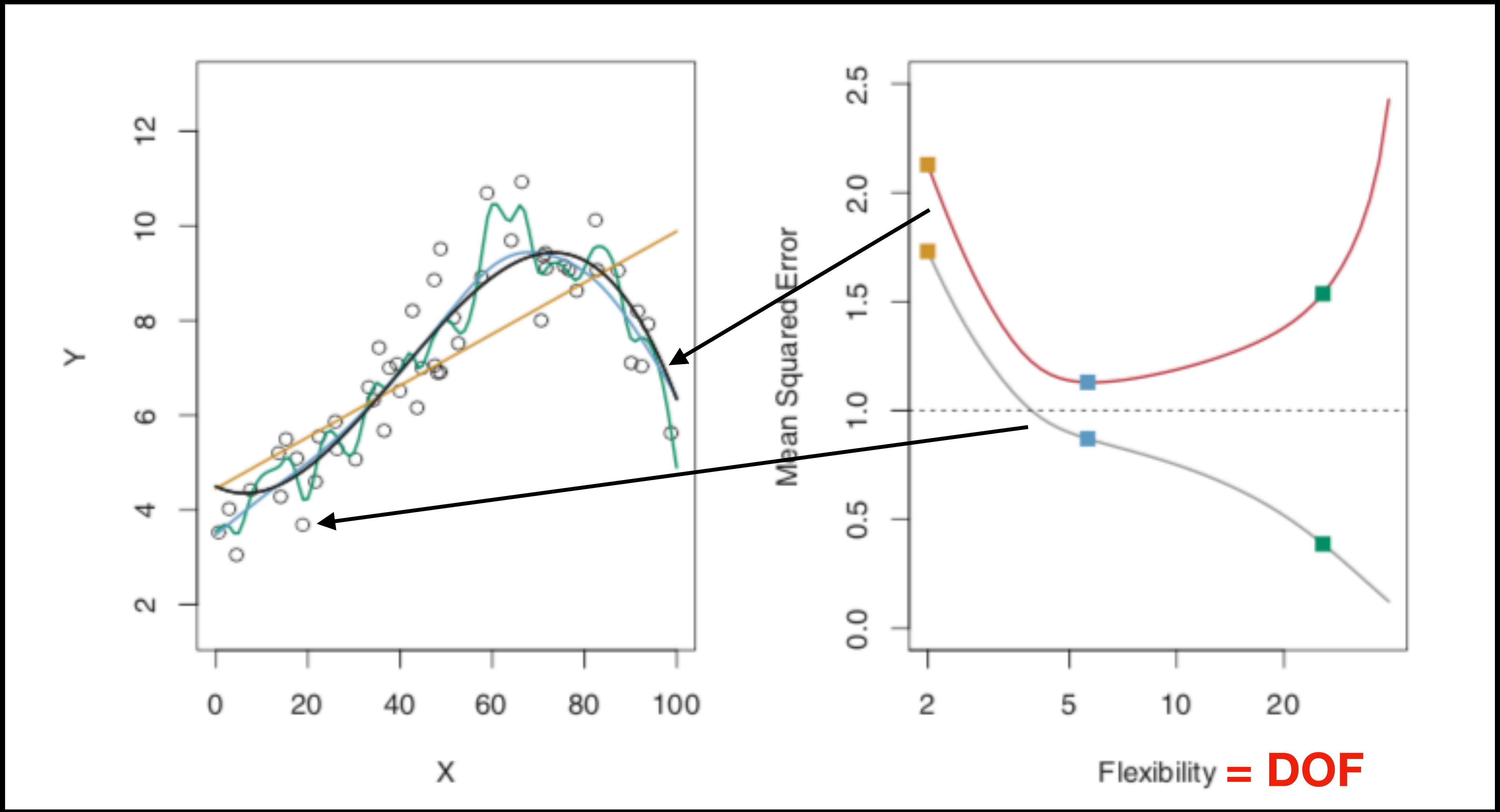


Overfit

Is overfitting or underfitting worse?

Bias-Variance Trade-Off (Second Glance)

$$E\left(y_0 - \hat{f}(x_0)\right)^2 = \mathrm{Var}(\hat{f}(x_0)) + [\mathrm{Bias}(\hat{f}(x_0))]^2 + \mathrm{Var}(\epsilon).$$



Actual underlying function - y

o Simulated data with added error (ϵ)

Linear fit

Low “flexibility” smooth spline

High “flexibility” smooth spline

Bias-Variance Trade-Off (First Glance)

$$E\left(y_0 - \hat{f}(x_0)\right)^2 = \text{Var}(\hat{f}(x_0)) + [\text{Bias}(\hat{f}(x_0))]^2 + \text{Var}(\epsilon)$$

mean square error if we kept estimating response variable y by our fitted function of our explanatory variables, f(x) with different sample datasets at point x_0

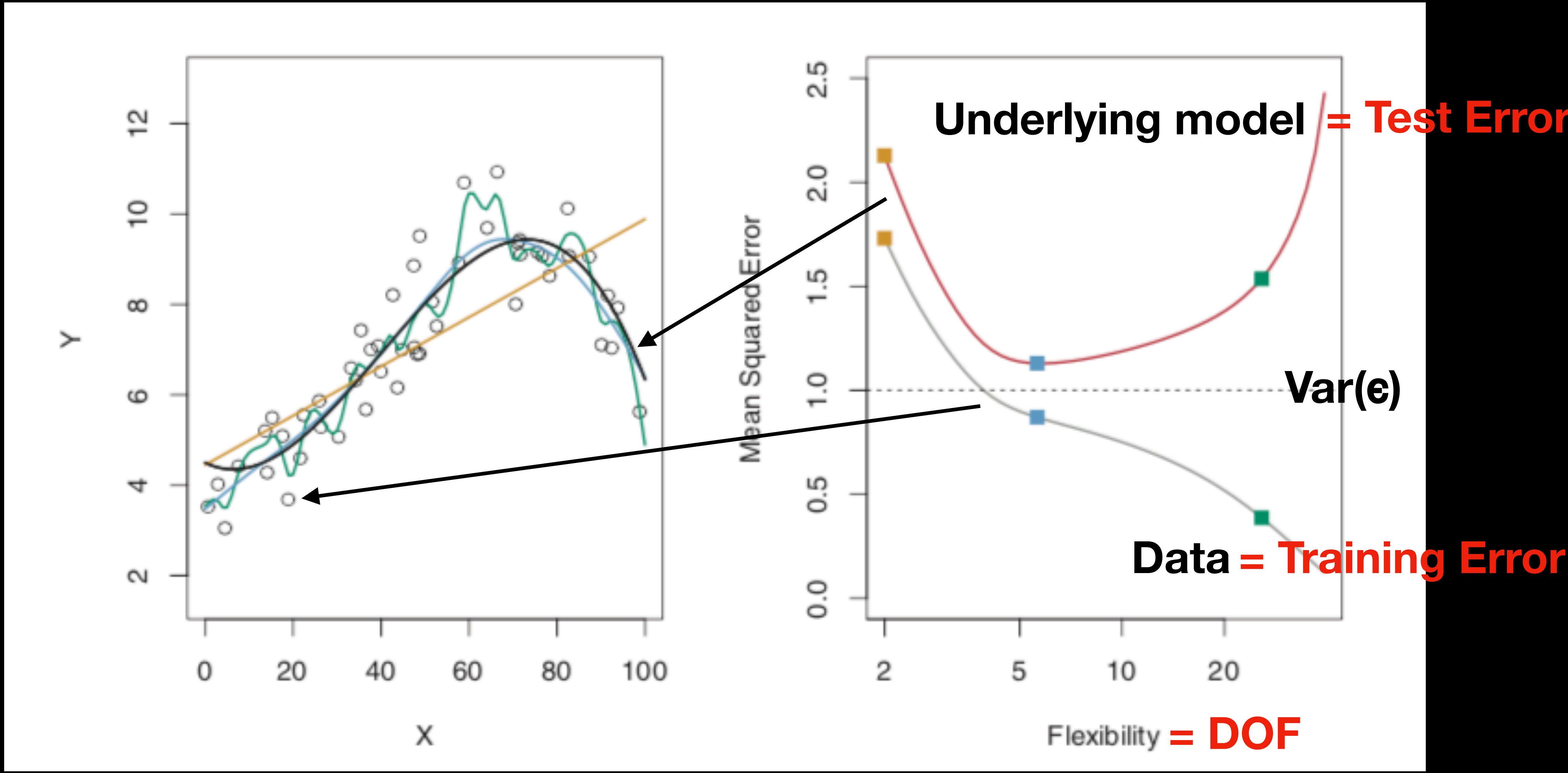
Inherent error in our measurements

Inherent error (bias) in the fact that any model is only an approximation to reality

how much our function, f, changes if we use a different random sample (variance)

Bias-Variance Trade-Off (Second Glance)

$$E\left(y_0 - \hat{f}(x_0)\right)^2 = \mathrm{Var}(\hat{f}(x_0)) + [\mathrm{Bias}(\hat{f}(x_0))]^2 + \mathrm{Var}(\epsilon).$$



Actual underlying function - y

o Simulated data with added error (ϵ)

Linear fit

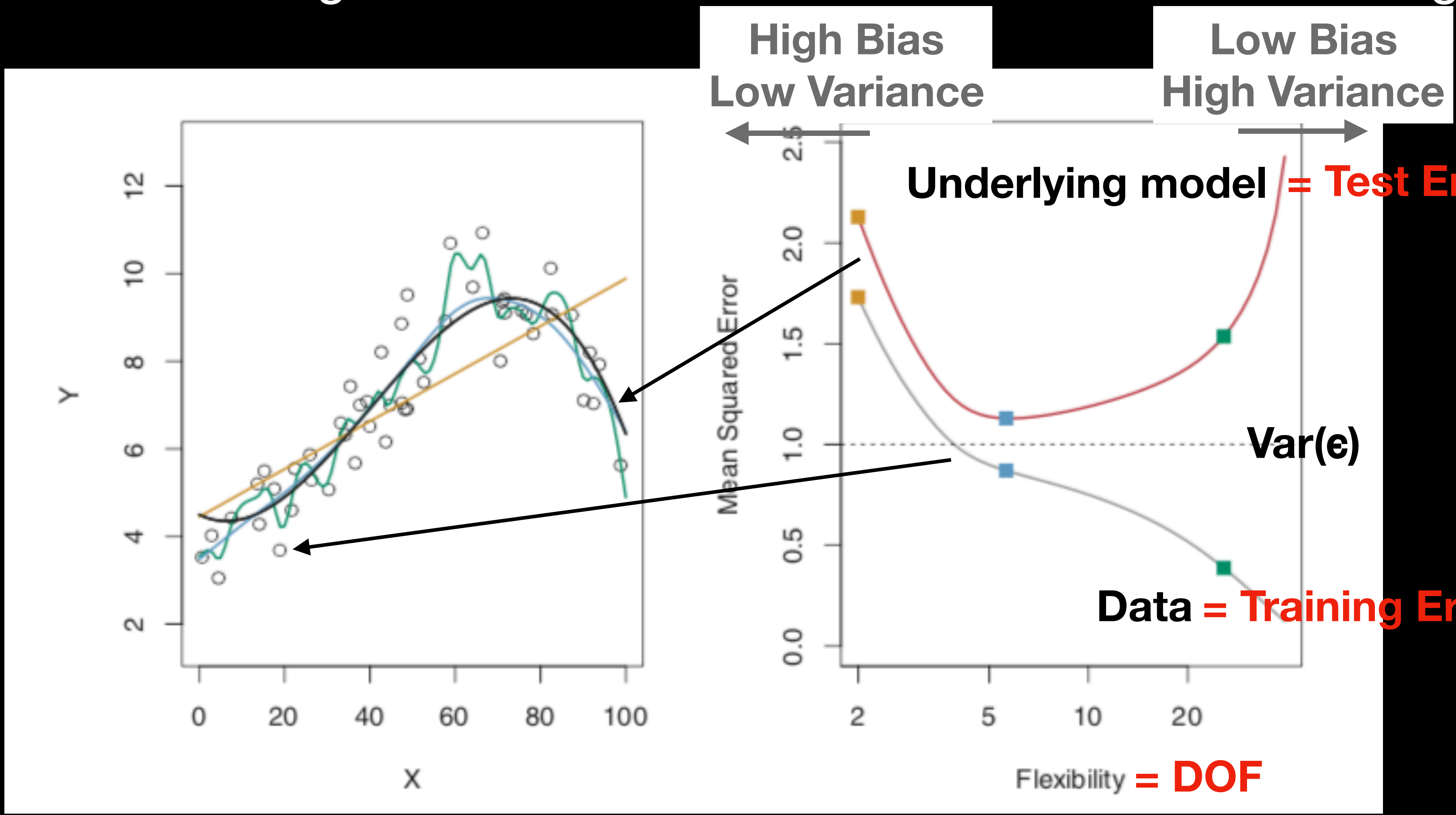
Low “flexibility” smooth spline

High “flexibility” smooth spline

Bias-Variance Trade-Off (Second Glance)

$$E\left(y_0 - \hat{f}(x_0)\right)^2 = \mathrm{Var}(\hat{f}(x_0)) + [\mathrm{Bias}(\hat{f}(x_0))]^2 + \mathrm{Var}(\epsilon).$$

- The test error is the average error that results from using a statistical learning method to predict the response on a new observation, one that was *not* used in training the method. (Hard to calculate w/o an underlying model.)
- In contrast, the training error can be easily calculated by applying the statistical learning method to the observations used in its training.



Test & Training Error in KNN: With Math!

- The test error is the average error that results from using a statistical learning method to predict the response on a new observation, one that was *not* used in training the method. (Hard to calculate w/o an underlying model.)

$$\text{Ave}\left(I(y_0 \neq \hat{y}_0)\right)$$

new observation, requires we know what that would be from an underlying model (or more observations)

What our calculated fit/ model would predict

- In contrast, the training error can be easily calculated by applying the statistical learning method to the observations used in its training.

$$\frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{y}_i)$$

an observation in our current dataset

What our fit/model would predict

Here I = 1 if y_i != $\hat{y}_i$ and I = 0 if y_i = $\hat{y}_i$, so larger I means worse model

K Nearest Neighbors (and MLR) → Cross-Validation & Bootstrap

p-values quantify the fit of individual parameters

quantify how good the *model* is

But first: some definitions!

Using our KNN example in R with an underlying model!