Computational Methods for Linguists

Ling 471

Olga Zamaraeva (Instructor)
Yuanhe Tian (TA)
05/11/21

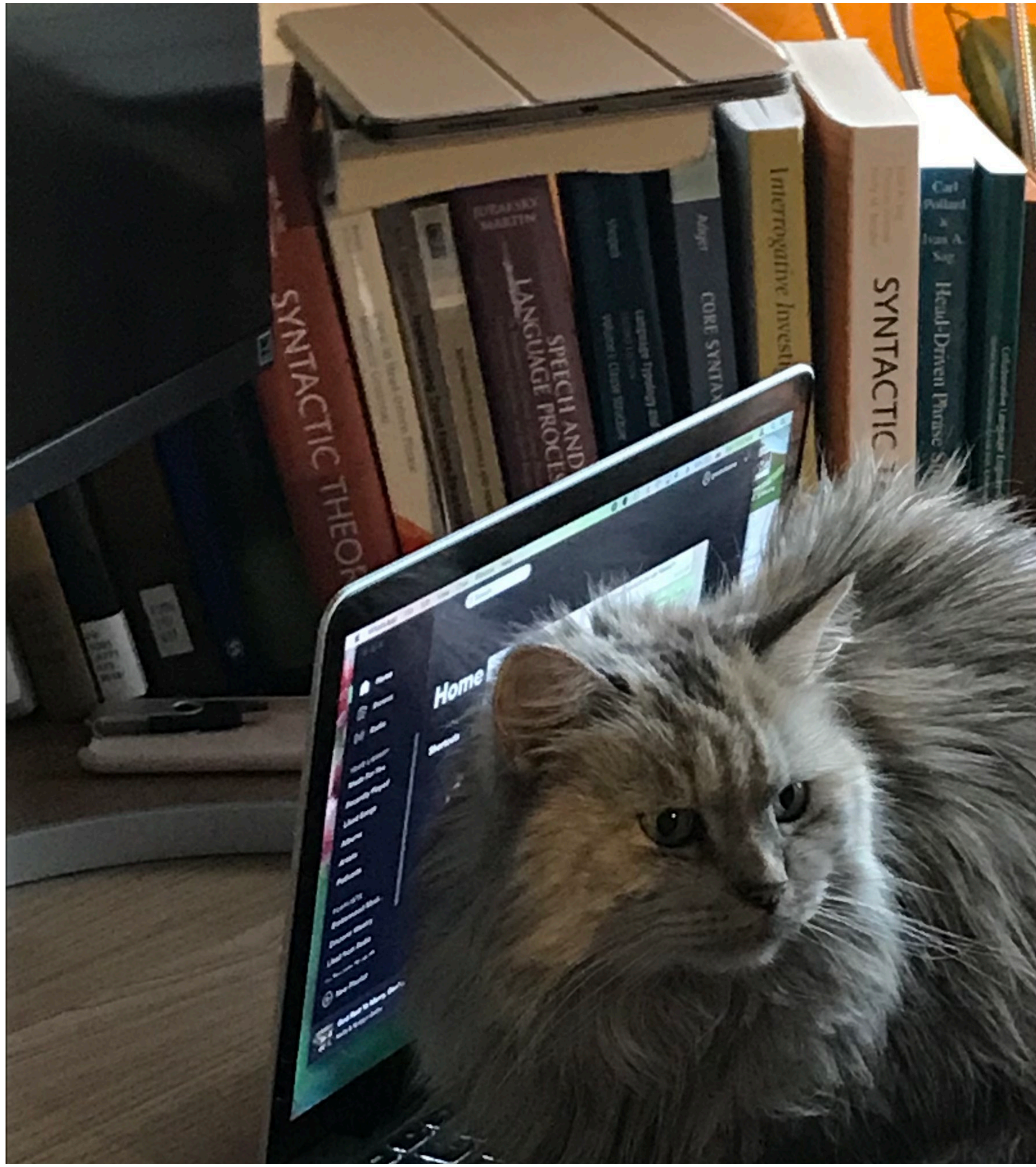
Reminders
and announcements

- Start thinking about presentations
  - More on resources today
- Blog 4 due today
  - Responses by Tuesday…

Presentation

June 1—3, 15% of grade

- Each student will do a short presentation:
  - Must present a project (such as a research paper) that involves statistical analysis of language data
  - Must relate to/reflect on social aspects
  - Otherwise, can discuss systems, programming, ML…
  - Suggest your presentation **topic by May 25 on Canvas**.
- The presentation will be peer reviewed for clarity and effectiveness of communication and visualization
  - During class! We will watch and give feedback.
- Submit your presentation slides (in June) after addressing feedback (but no need to present again!)
- Your original presentation can be prerecorded or not



Presentations
resources

- Some places you can access papers/projects to present on (also see Canvas discussion board for Presentation Topics):
  - https://paperswithcode.com/datasets
  - CL papers:
    - https://www.aclweb.org/anthology/
  - Linguistic (and other) papers:
    - See Blog Week 5
    - Look also for similar papers
      - e.g. in **Google Scholar**

Plan for today

- Tying up lose ends
  - dataframes multiplication exercise recap (questions?)
  - linear regression demo
  - why was there a column of 1s (slide 29 from last time) ?
- Overfitting and regularization
- Classification
  - Logistic regression
  - Naive Bayes
  - Out-of-vocabulary items and Smoothing
- No activity today :)

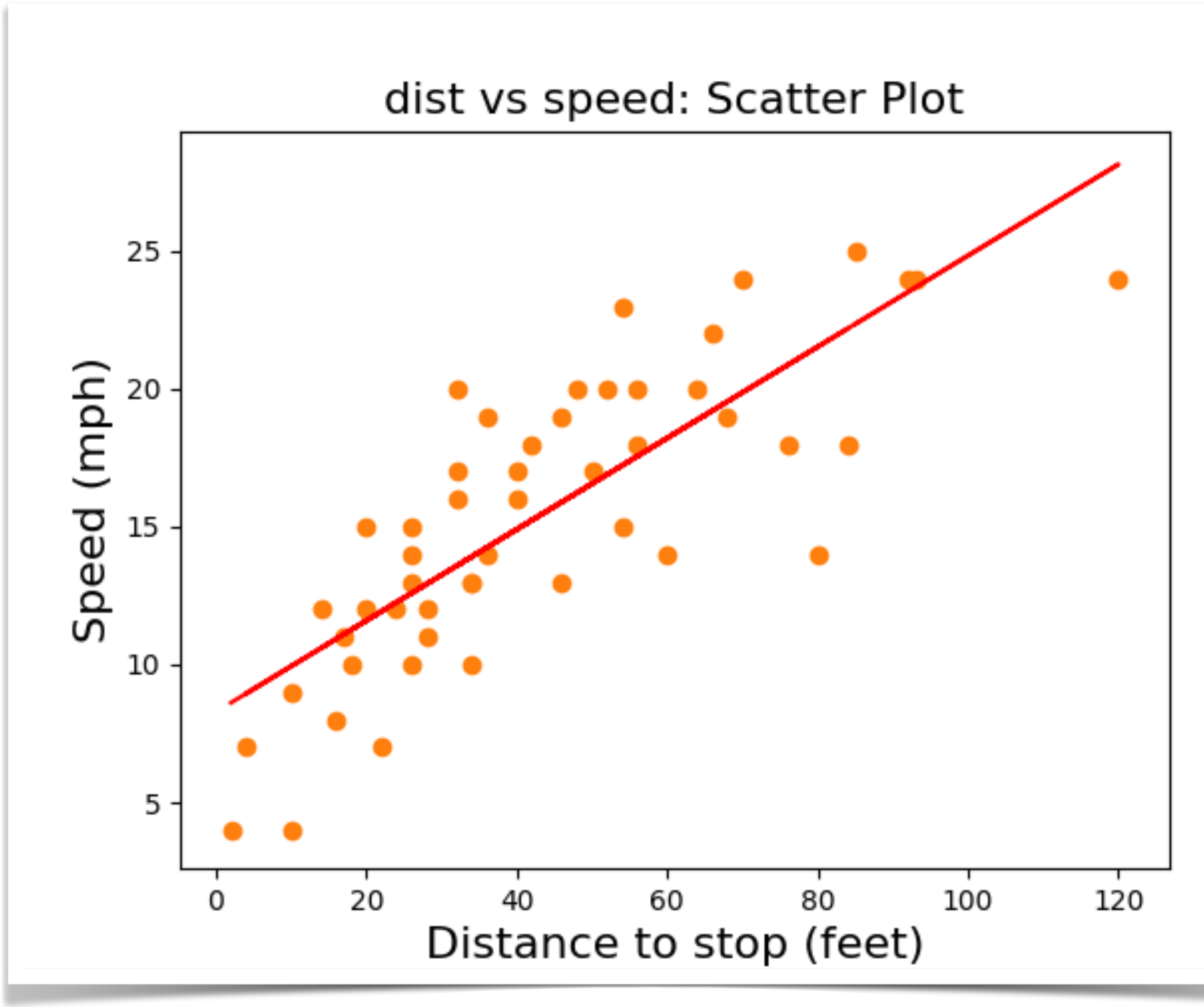
Look at the pie sales exercise in VS Code

Linear regression

“Least squares”

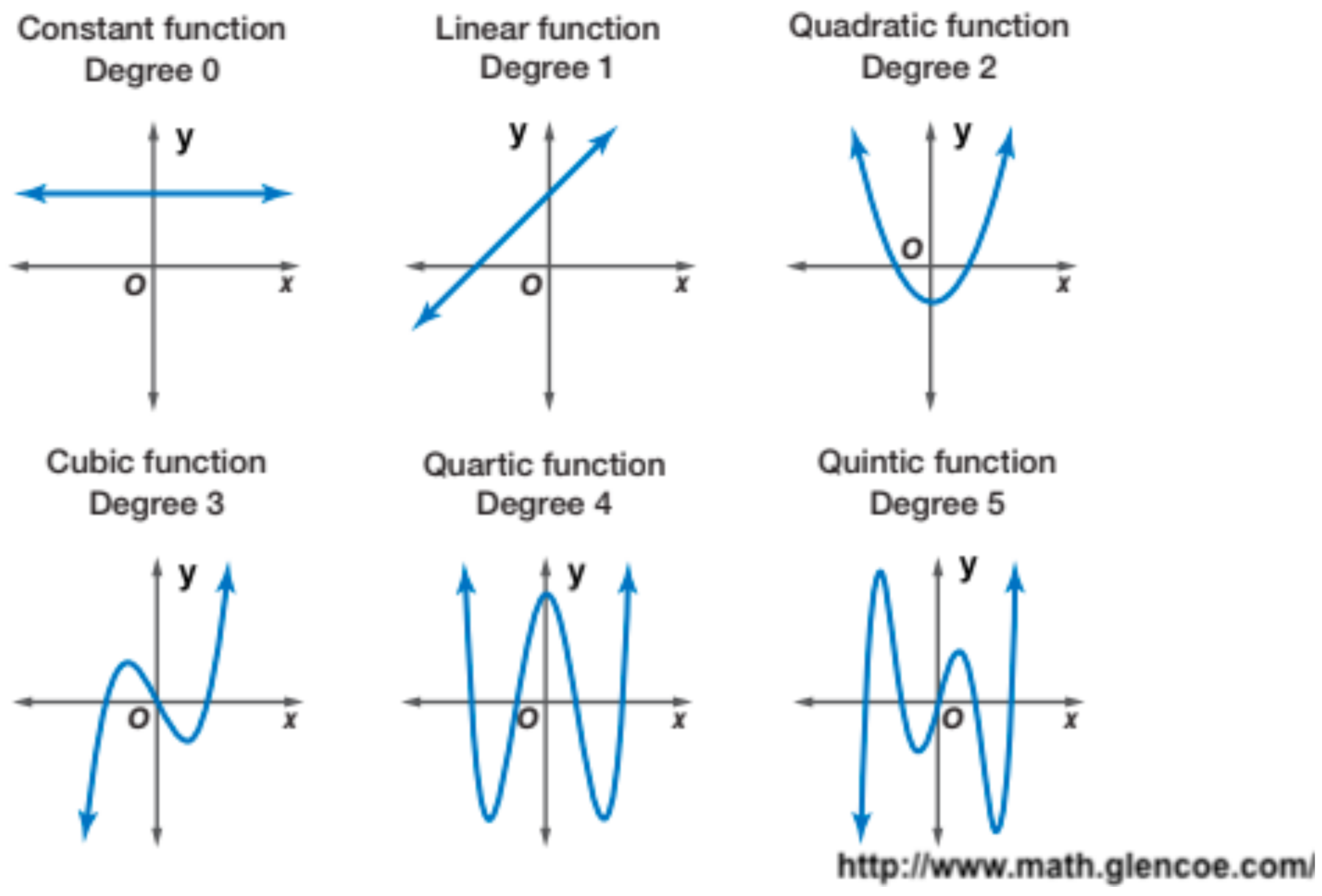
$$X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix} \quad Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \quad A = \begin{bmatrix} b \\ m \end{bmatrix} \quad E = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{bmatrix}$$

- **Y = AX + E**
  - All things here are **matrices**
  - Y, A, E are just **vectors** (matrices of **width 1**)
    - **vectors are matrices**, too!
    - X needs to have the same **width** as the **length** of A
      - ...to conform to matrix multiplication definition
      - **hence the column of 1s**
- Want: **solve for A to minimize** $\sum_{i=1}^{n} e_i^2$
- **NB:** The linear regression fit curve **need not** be straight
  - It can be any polynomial



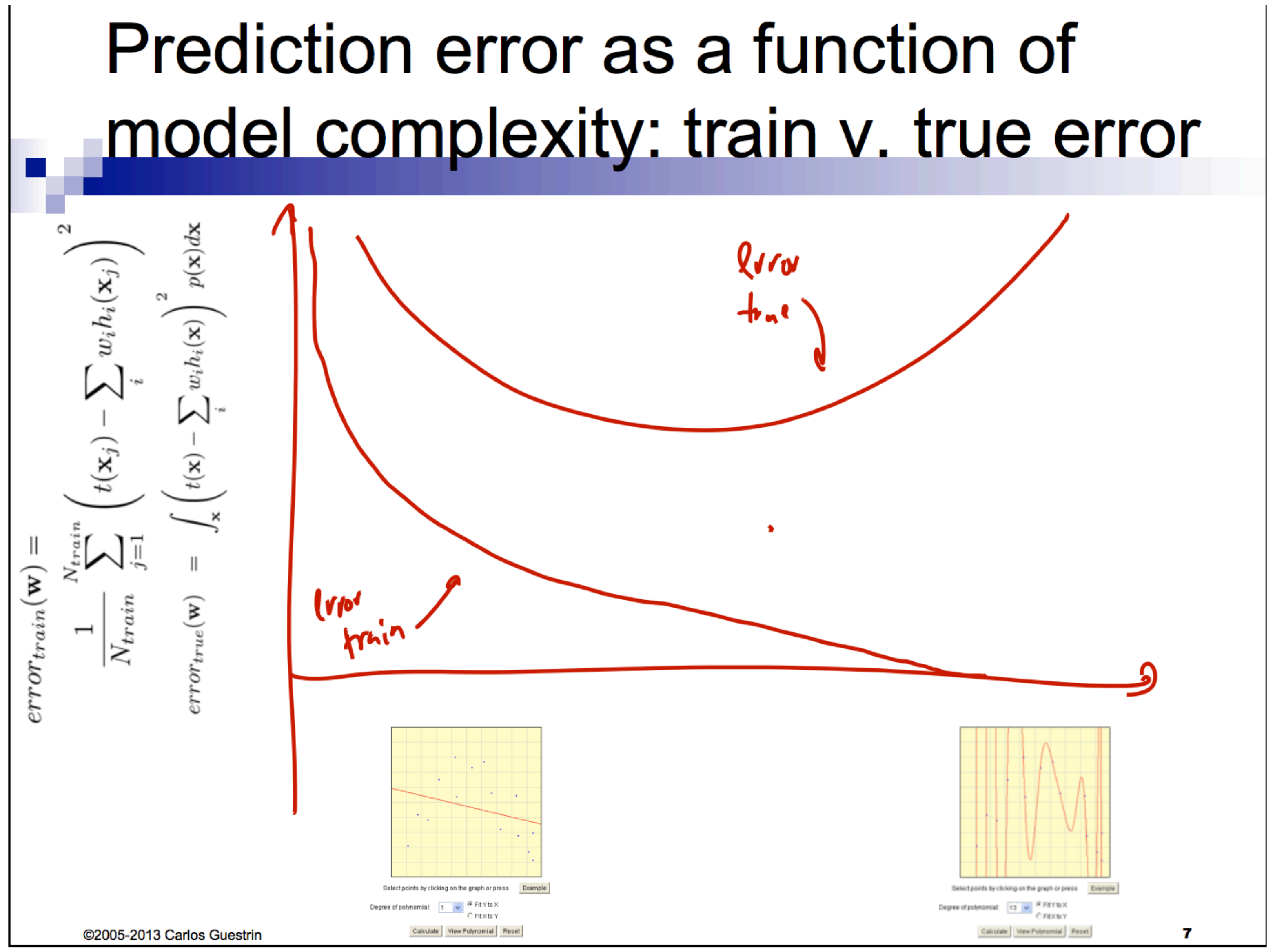
Degree of polynomial

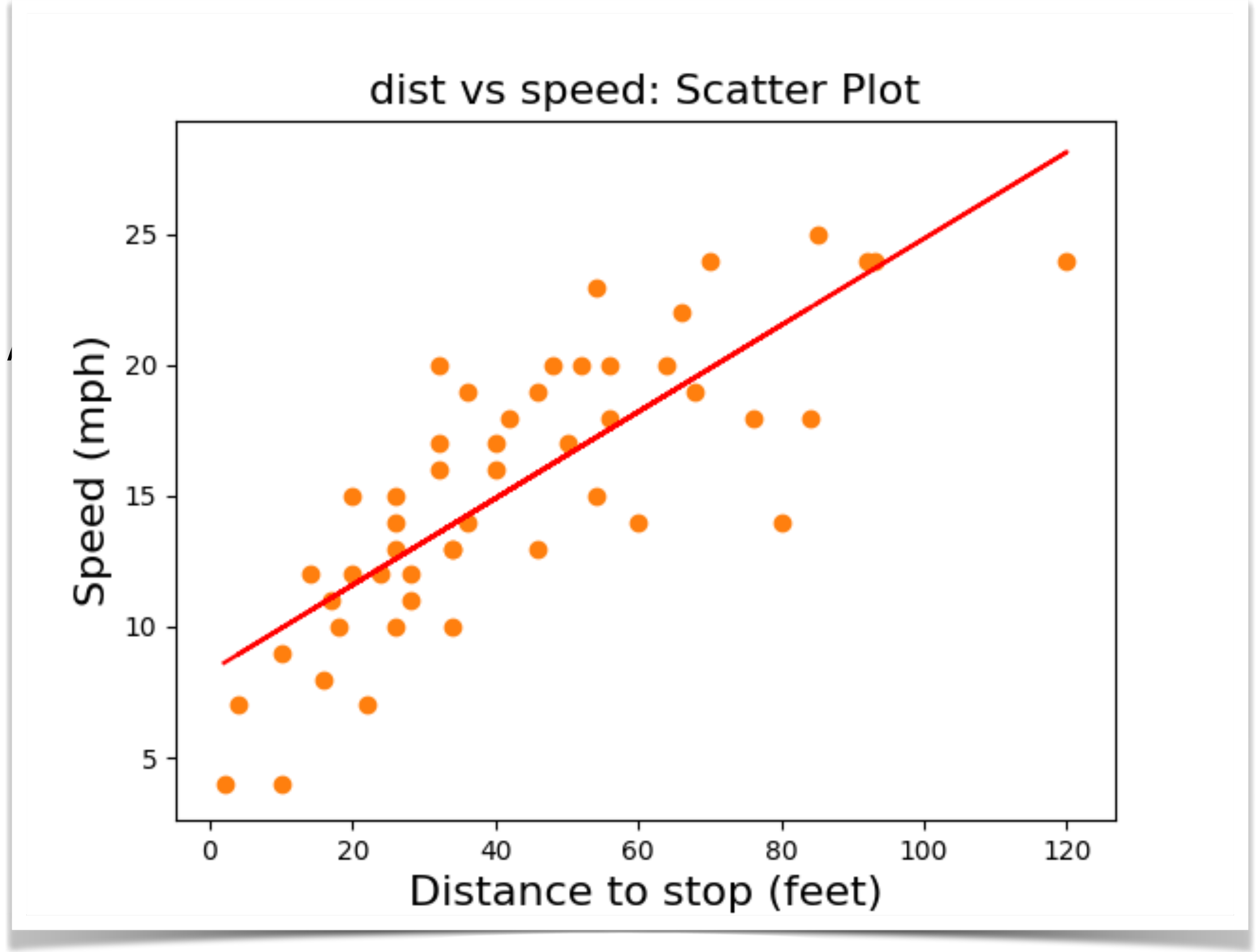
- Polynomial: a linear equation:
  - y = ax^1 + bx^2 + cx^4 + dx^5…
  - a,b,c,d… — coefficients
    - coefficients can be 0!
- The higher the max degree:
  - The more inflection points (the crazier) the curve
- The higher the coefficients:
  - The more “weight” on the higher degree terms
  - 0 * x^123 means x^123 is absent!
- Linear regression algorithm must choose the coefficients
  - including deciding which should just be 0!



https://bookdown.org/tpinto_home/Beyond-Linearity/polynomial-regression.html

Overfitting
and model complexity

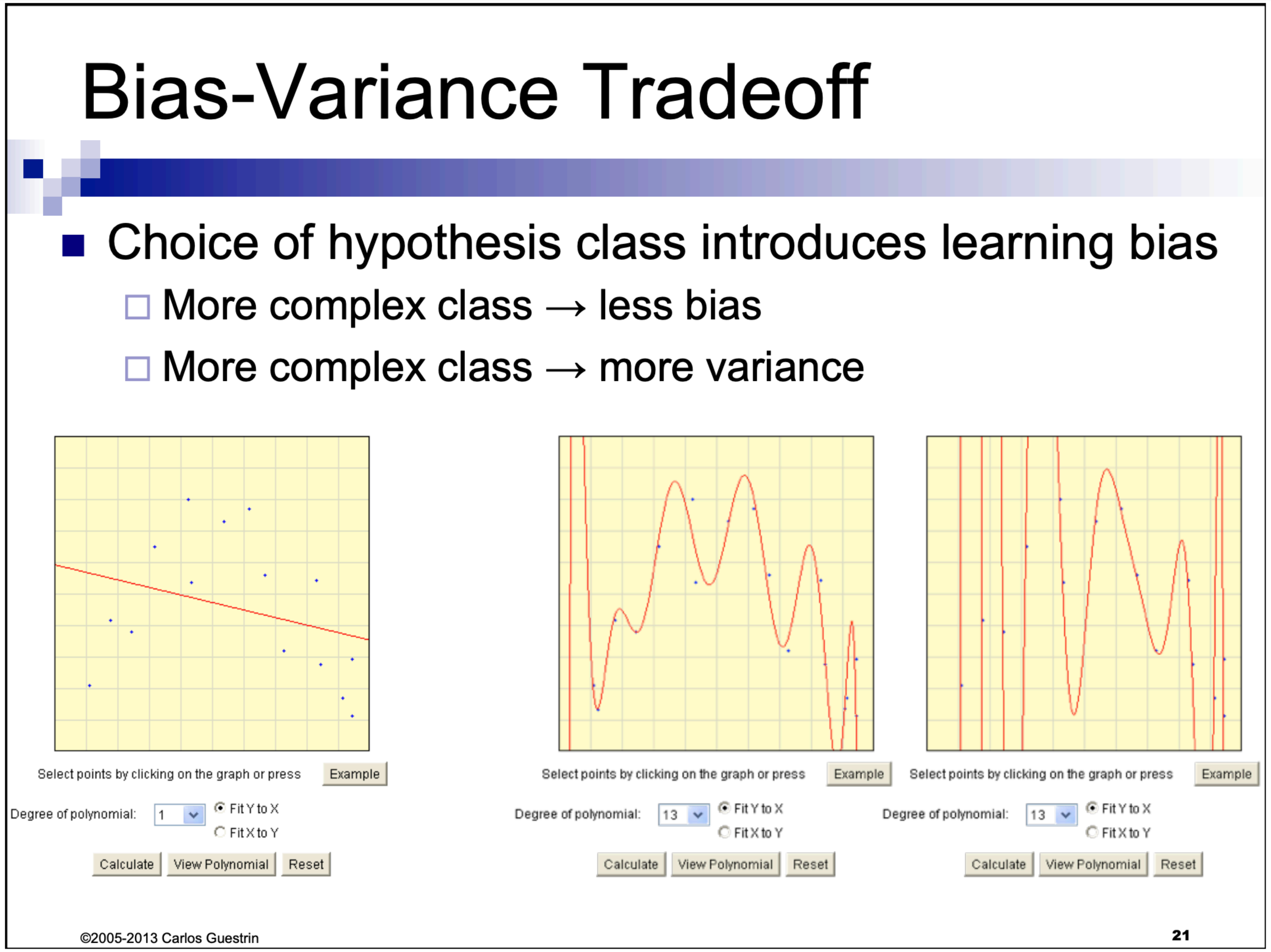




- What kind of function/**curve fits** the observations **best**?
- **Option 1**: a curve which **minimizes training error**
- ...actually, such a curve will go through every point!
  - **Overfitting!** No chance we will get an **unseen** point right (the error will be too large)
- **Option 2:** a curve which **allows** for **some small** error in training
  - ...but results in **smaller test error** in practice
  - such a curve is **smoother** (maybe even straight!)
  - => it is a lower-degree polynomial

Degree of polynomial

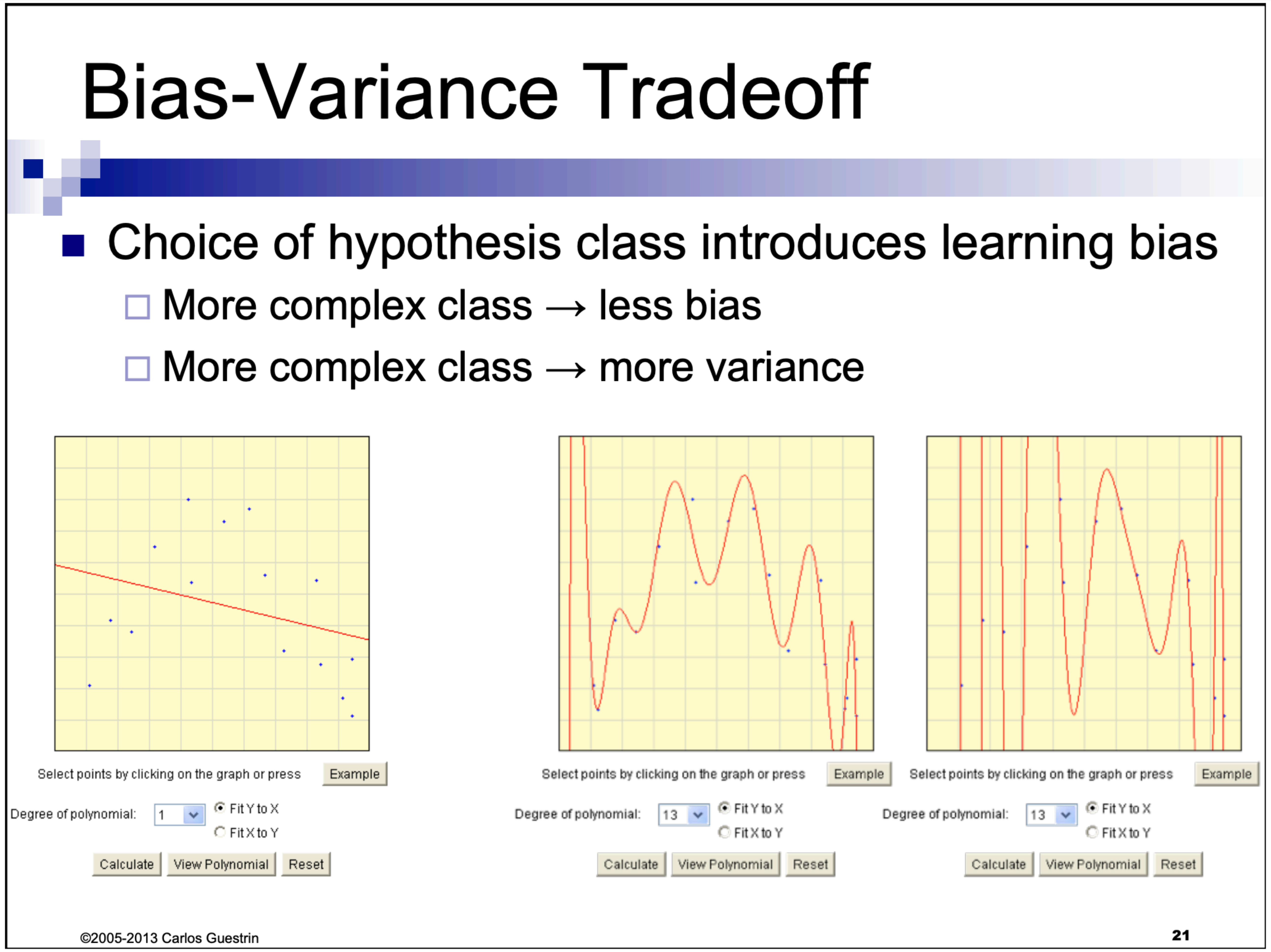
- Polynomial: a linear equation:
  - y = ax^1 + bx^2 + cx^4 + dx^5...
  - a,b,c,d... — coefficients
    - coefficients can be 0!
- The higher the max degree:
  - The more inflection points (the crazier) the curve
- The higher the coefficients:
  - The more “weight” on the higher degree terms
  - 0 * x^123 means x^123 is absent!
- Linear regression algorithm must choose the coefficients
  - including deciding which should just be 0!



Bias-Variance Tradeoff

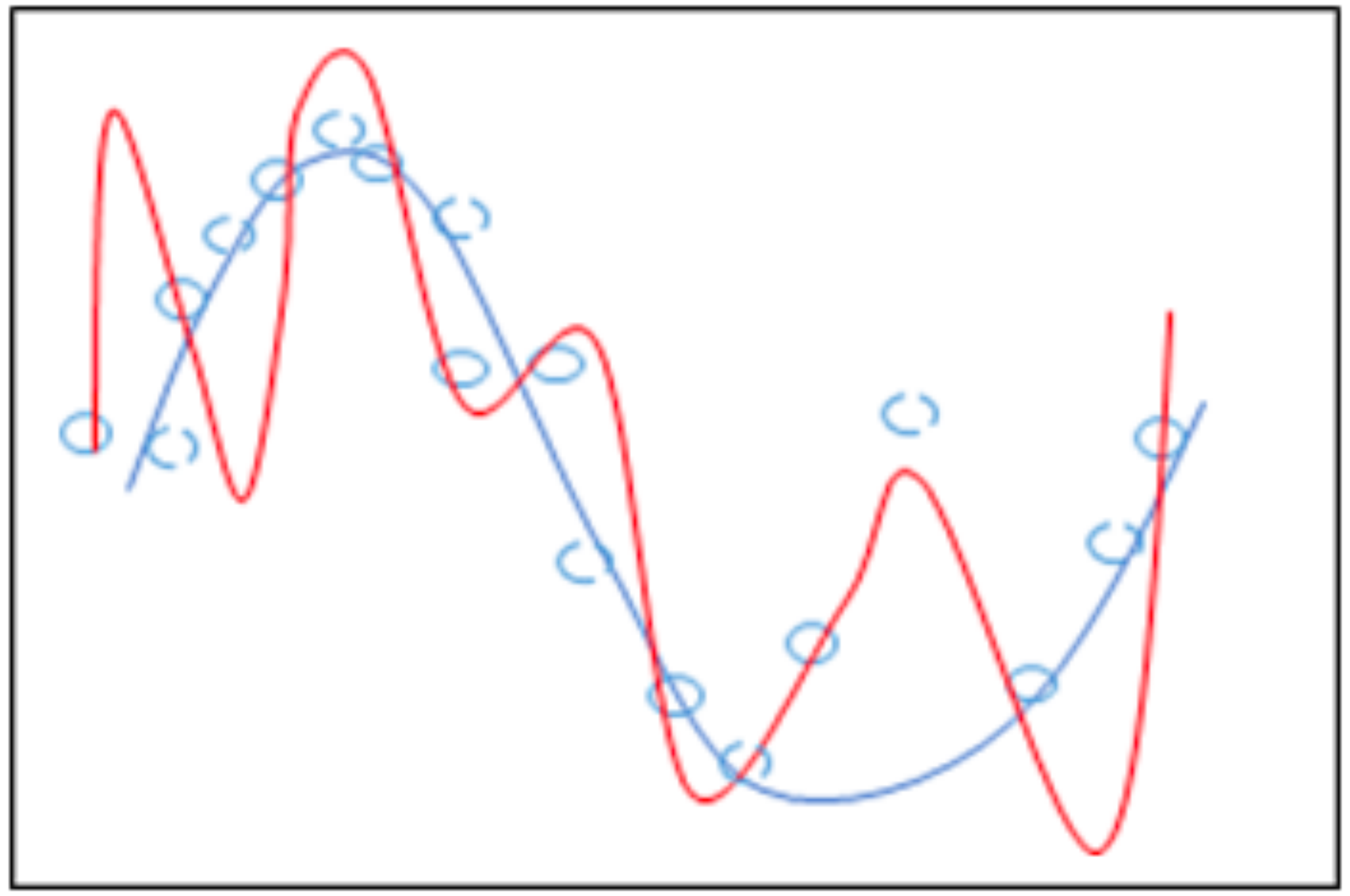
underfitting and overfitting

- A simple line is hardly good!
- A crazy polynomial also…
- What would be good?
    - It depends on the shape of data
    - Here, looks like y = x^2 :)
    - Again, you learn the function automatically by **minimizing SSE**
    - To avoid overfitting, you **penalize model complexity**



Regularization
reducing overfitting

- Overfit functions = highly complex
- **Penalize** complexity:
  - **prefer smaller coefficients:**
    - y = x + 2x^2 + 0.5x^3…
    - y = 482999000x + 78383946x^2 + 9193838x^3…
    - end up with **fewer terms**, as many coefficients will be **driven to 0**!
- **Some kind** of regularization is part of **most** ML pipelines
  - Stay tuned for **smoothing** wrt Assignment 4



https://medium.com/coinmonks/regularization-of-linear-models-with-sklearn-f88633a93a2

Linear regression demo

Classification

Classification

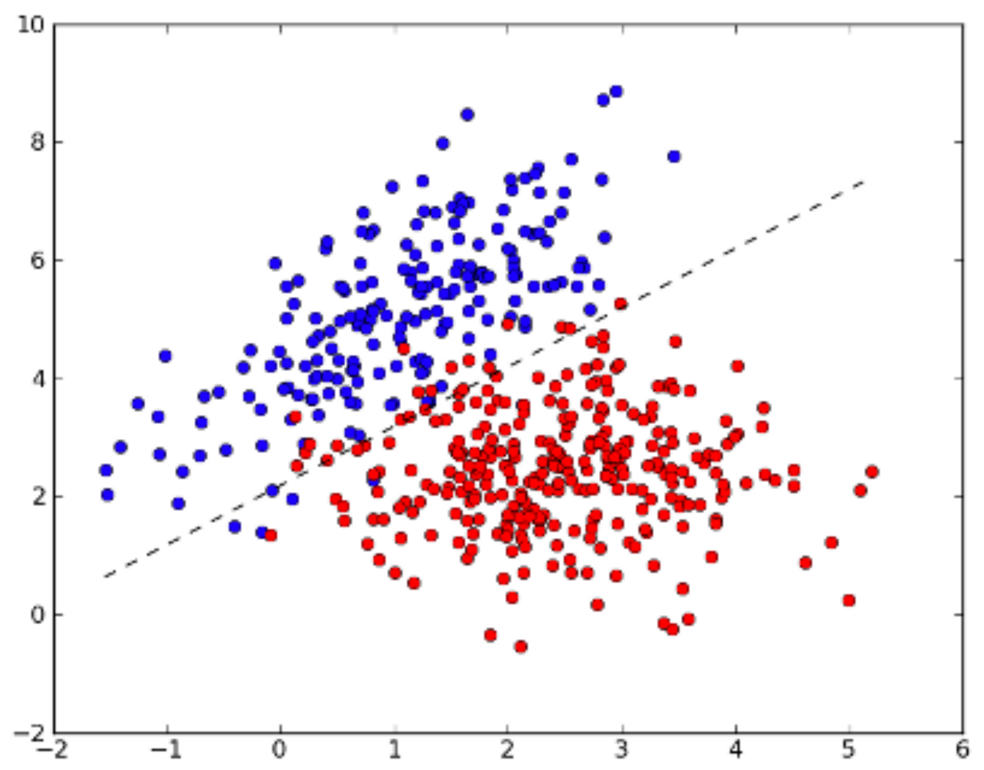
predicting discrete classes

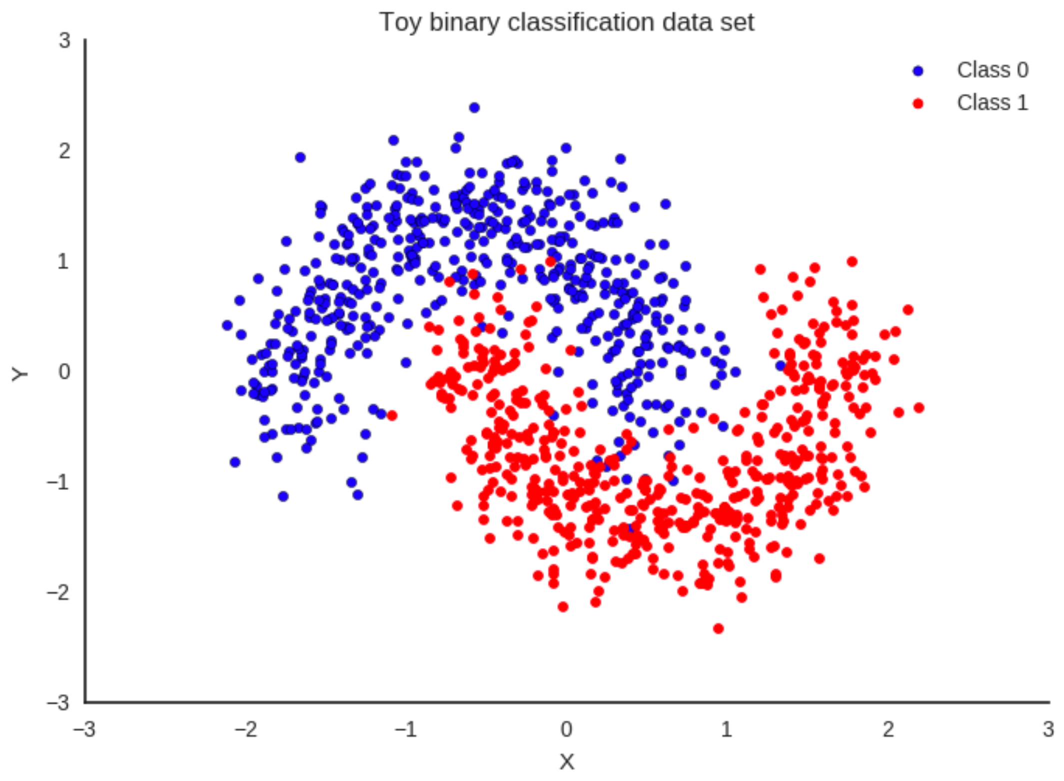
- Is the review positive or negative?
- Is a picture that of a cat or of a dog?
- Handwriting recognition (map to digit, letter)
- ...and many many many other tasks

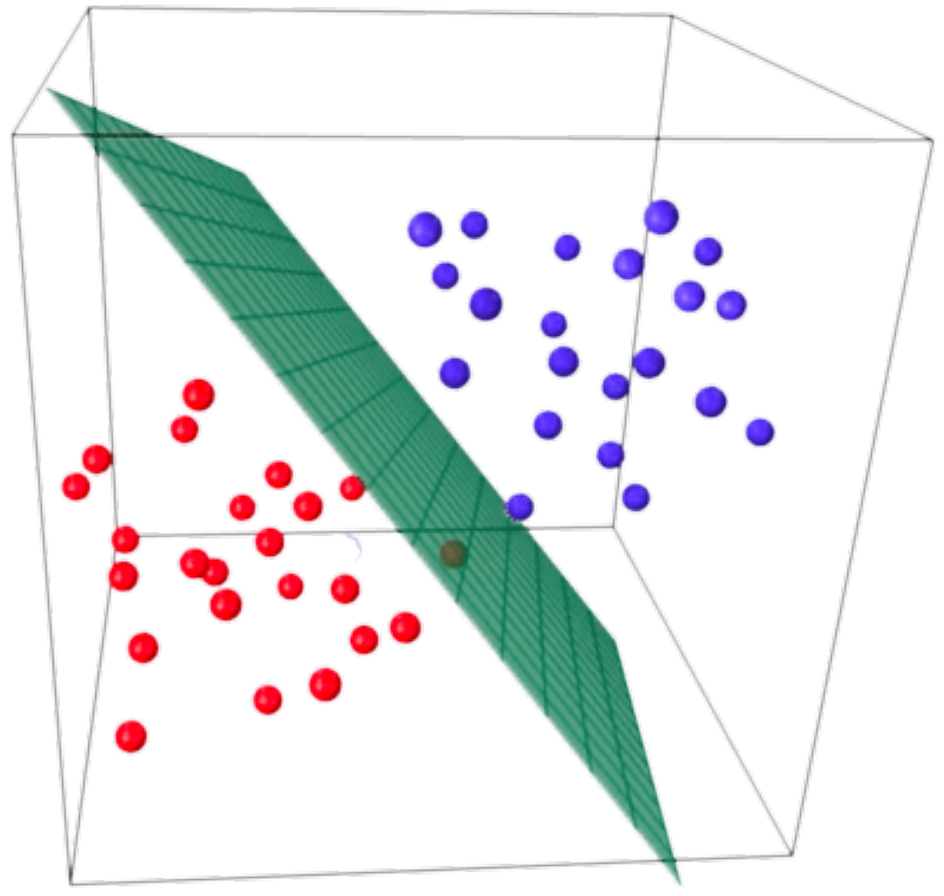
https://medium.com/anubhav-shrimal/dogs-vs-cats-image-classification-using-resnet-d2ed7e6db2bb

Linear Classification
predicting discrete classes

- ...using linear equations
  - Find a line which **separates** the data best
    - (similar multiplication of matrices will be involved!)
  - The linear function can be a higher degree polynomial
    - the "line" need not be straight
    - Get some datapoints wrong but minimize the overall error
    - Same idea as linear regression





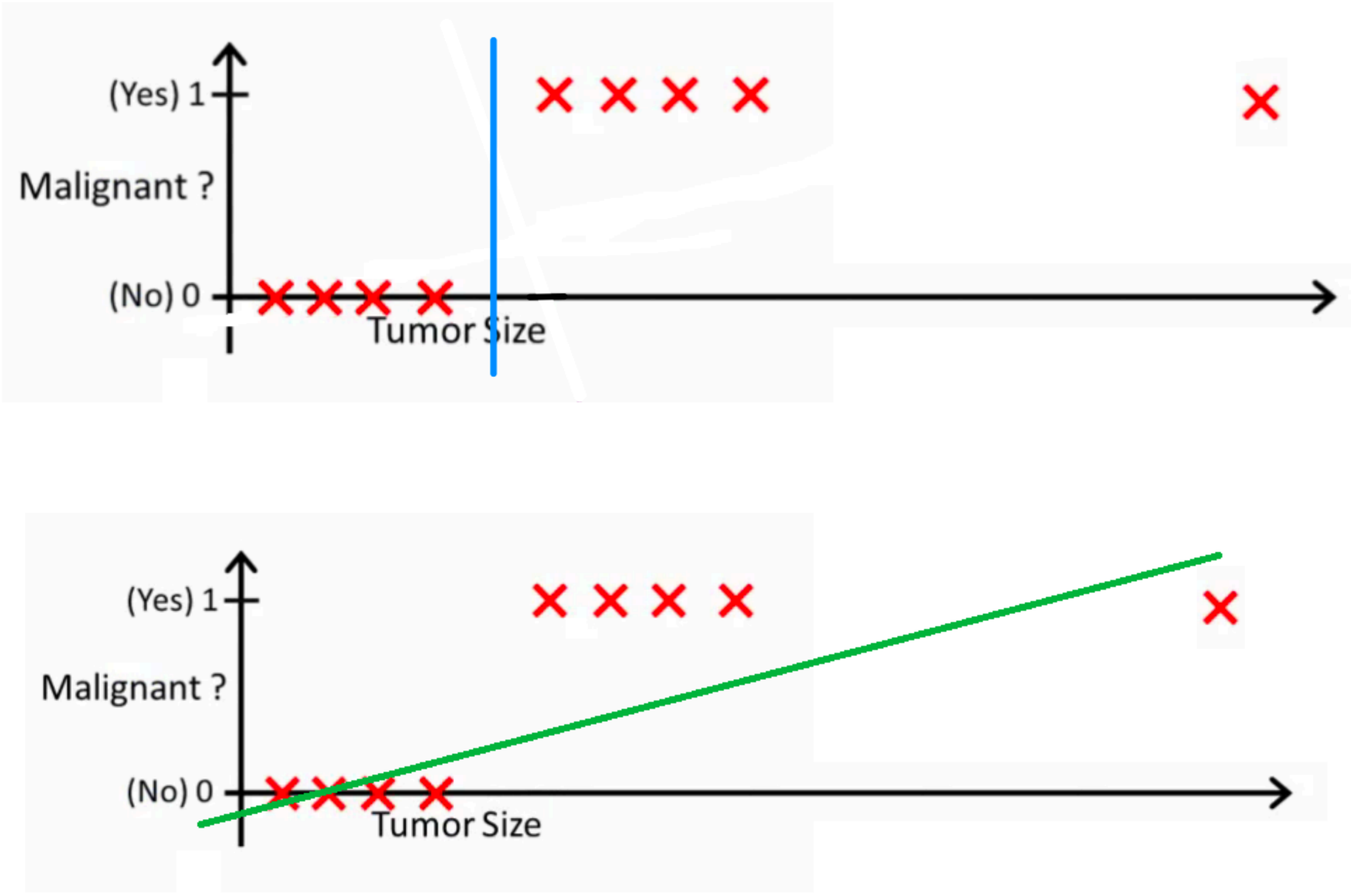


https://www.kaggle.com/kashnitsky/topic-4-linear-models-part-2-classification

Linear Classification

using linear regression

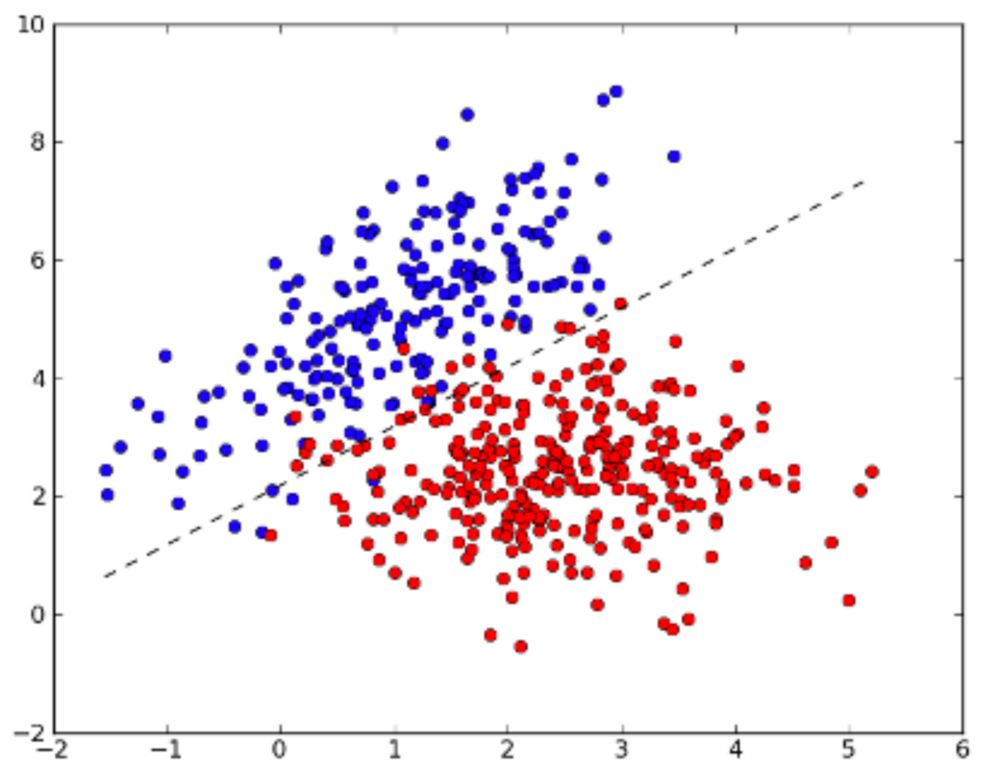
- Can we use **linear regression** for classification?
  - e.g. use data coordinates to classify, above or below the **decision boundary**
- In principle, yes
  - but it won't be very robust because data is not continuous
    - the **variance** will be too high
    - the model will be **too sensitive** to new datapoints
    - we **don't care** about the **distance** from point to line!

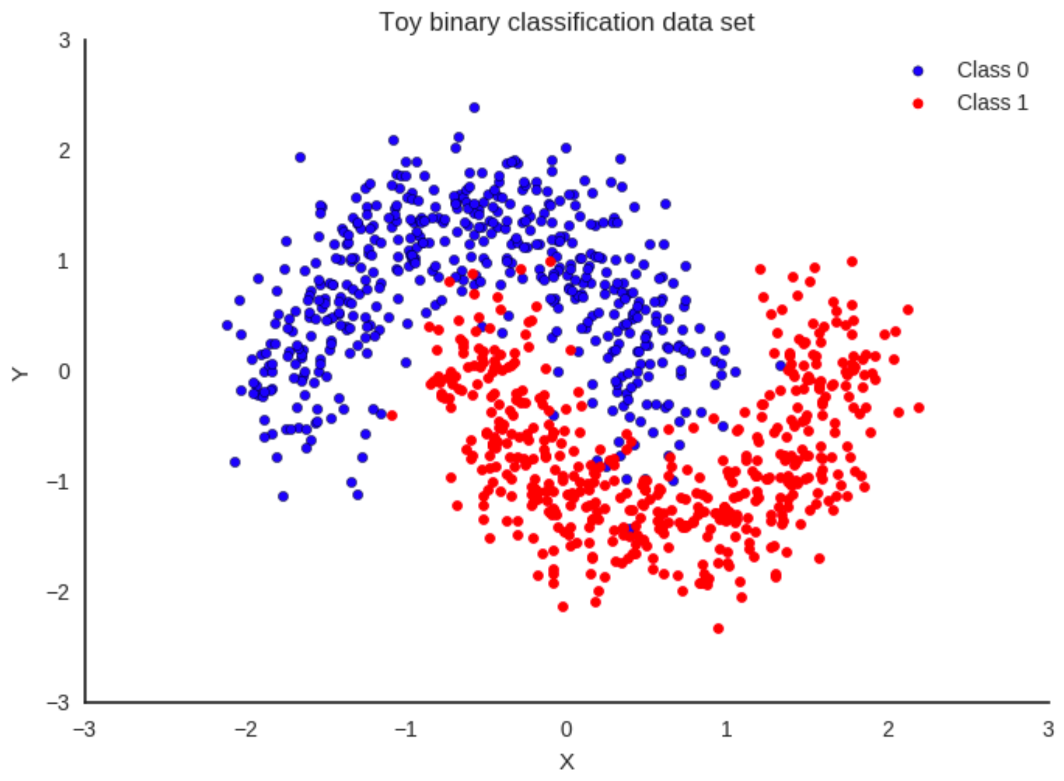


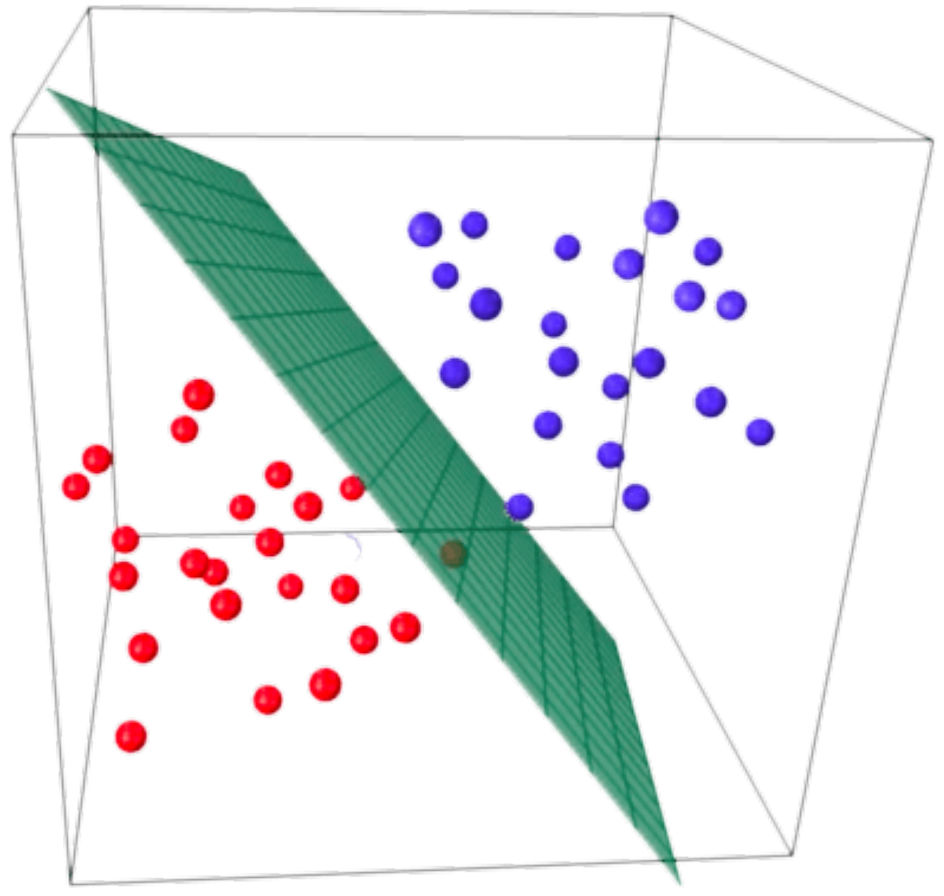
https://stats.stackexchange.com/questions/22381/why-not-approach-classification-through-regression

Linear Classification
predicting discrete classes

- How to dispence with high variance?
- Want a **simple** model:
  - Don’t care about specific distances etc.
  - Consider the **probability** of a point being on either side of the separator
  - Compute the probability of a point being above a certain line/curve/plane
  - If it is **high**, predict class A. **Otherwise** predict B.
    - define “high”, e.g. **0.5**



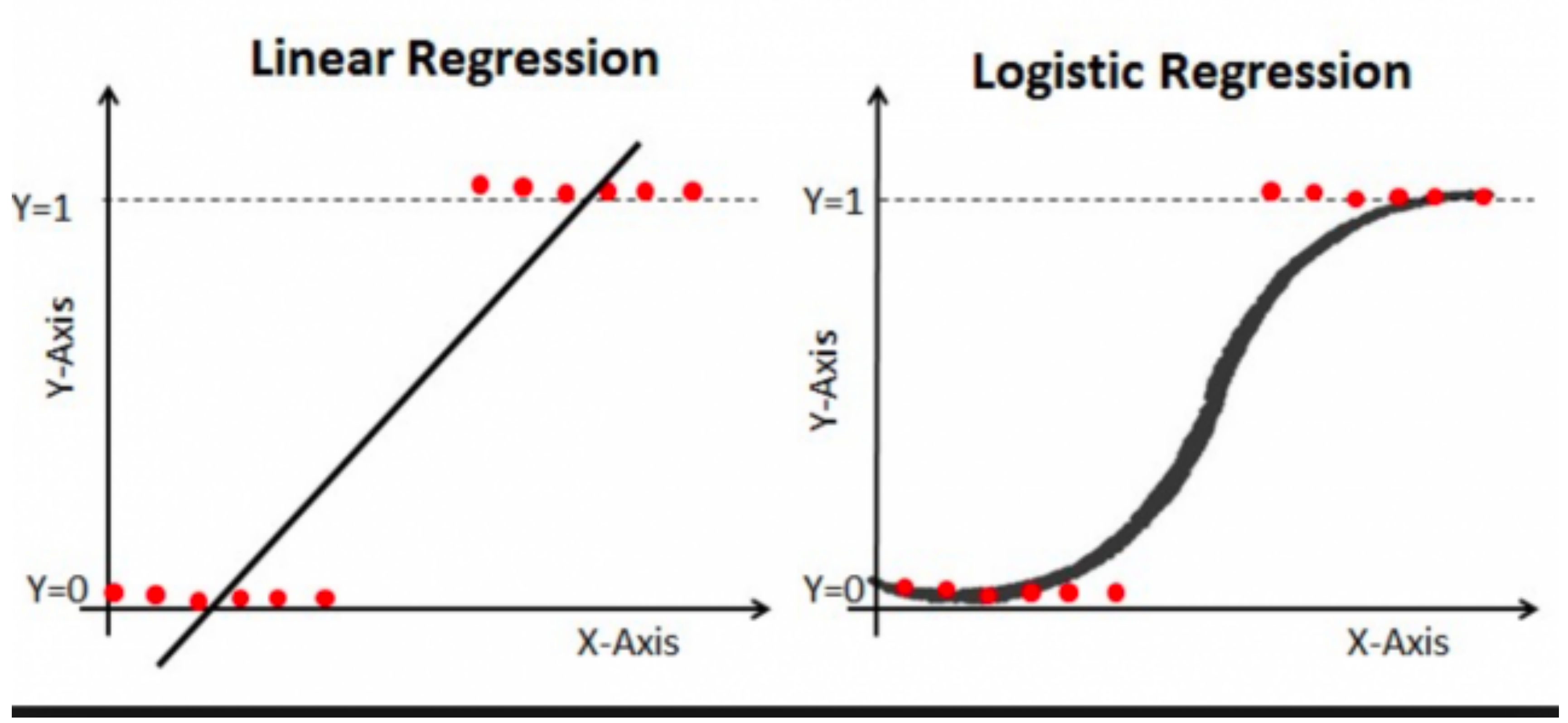




https://www.kaggle.com/kashnitsky/topic-4-linear-models-part-2-classification

Linear Classification

predicting discrete classes



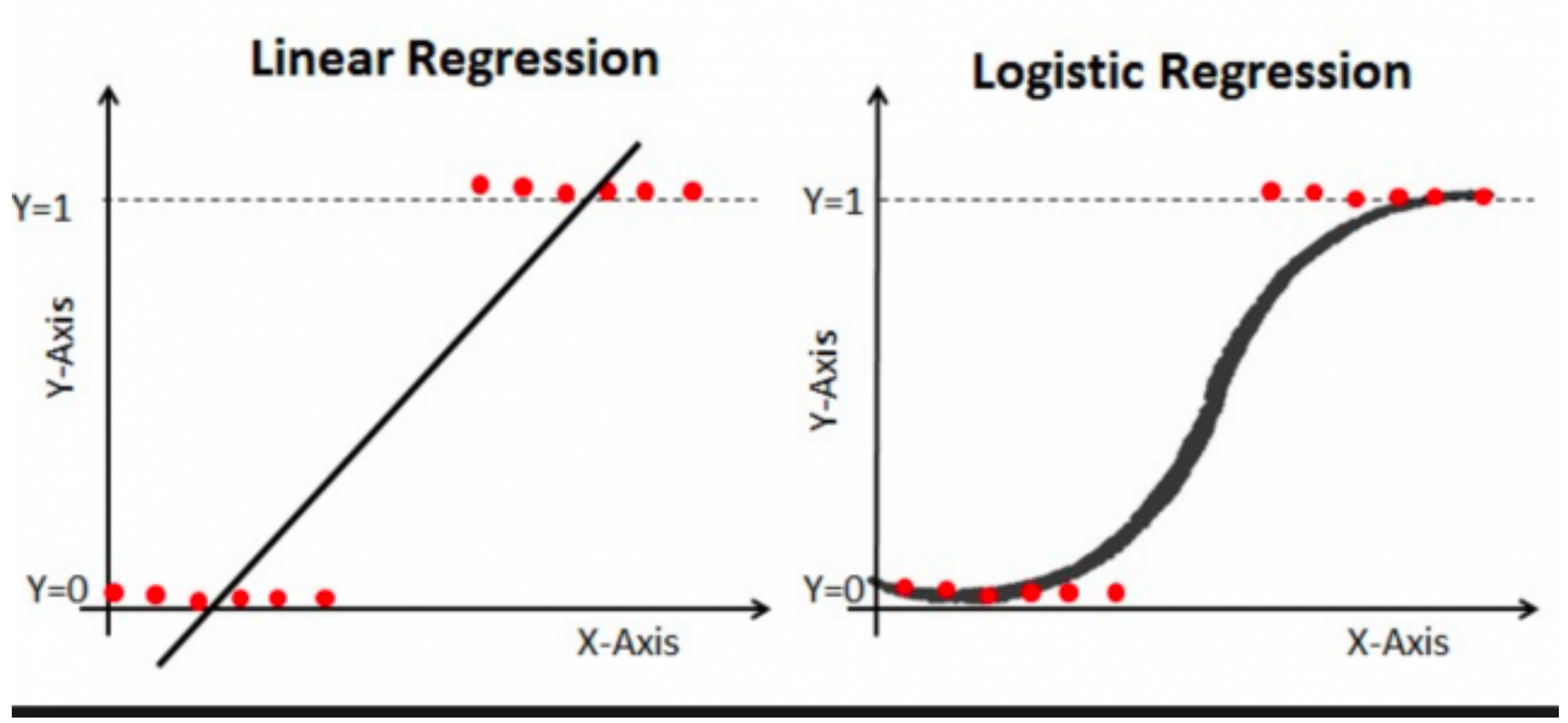
https://medium.com/@ODSC/logistic-regression-with-python-ede39f8573c7

- Can't use linear regression though!
  - We want a function that, given **x**, returns **P(y)**!
  - Probabilities range **from 0 to 1**
  - The output of a linear equation ranges **from -∞ to + ∞**
  - Solution:
    - **Map** a linear function to a function which ranges from 0 to 1
    - e.g. one of the family of **logistic** functions

Logistic regression

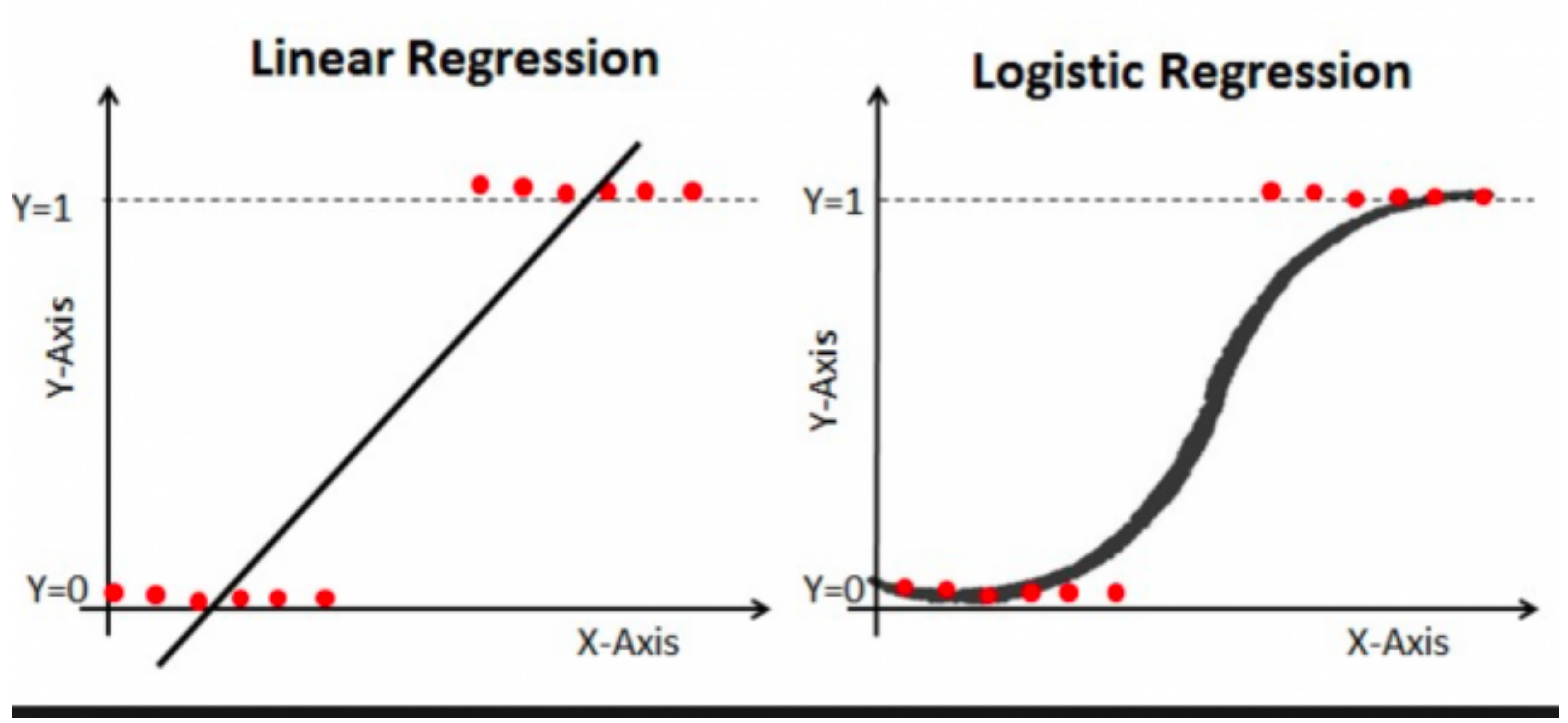
predicting discrete classes

- **Map** a linear function to a function which ranges from 0 to 1
  - e.g. one of the family of **logistic** functions
  - The function then outputs numbers between 0 and 1
  - ...which you can use as probabilities
  - ...to make predictions!



https://medium.com/@ODSC/logistic-regression-with-python-ede39f8573c7

Logistic regression
predicting discrete classes

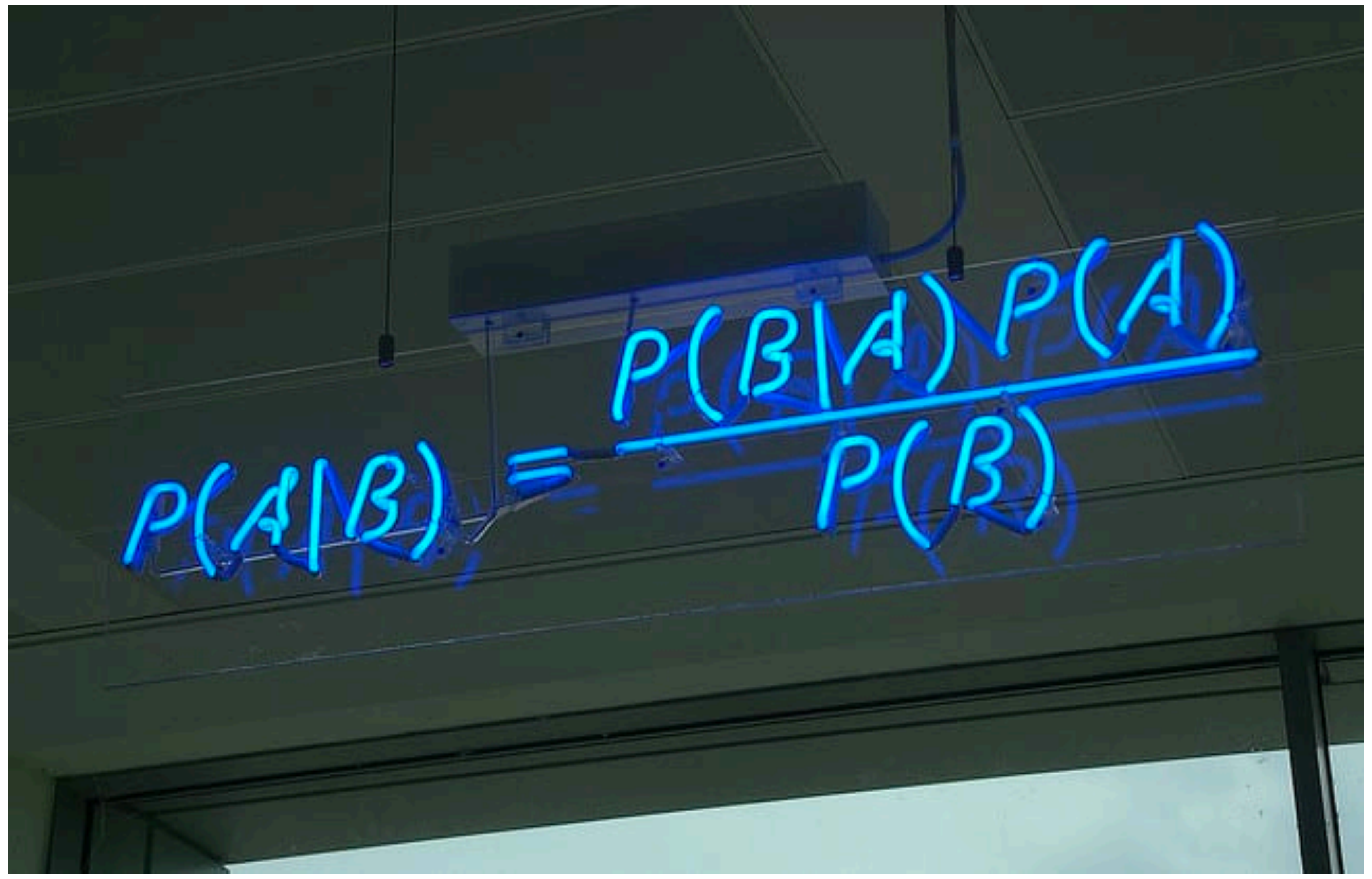


https://medium.com/@ODSC/logistic-regression-with-python-ede39f8573c7

- Is a classic classification method
  - …which is not really used much on its own these days (at least not in research)
  - But, **logistic and similar functions** are still a **core** component of any system
    - because **the mapping of the output to probabilities** is a **core** classification aspect

Naive Bayes

Naive Bayes
a classification algorithm

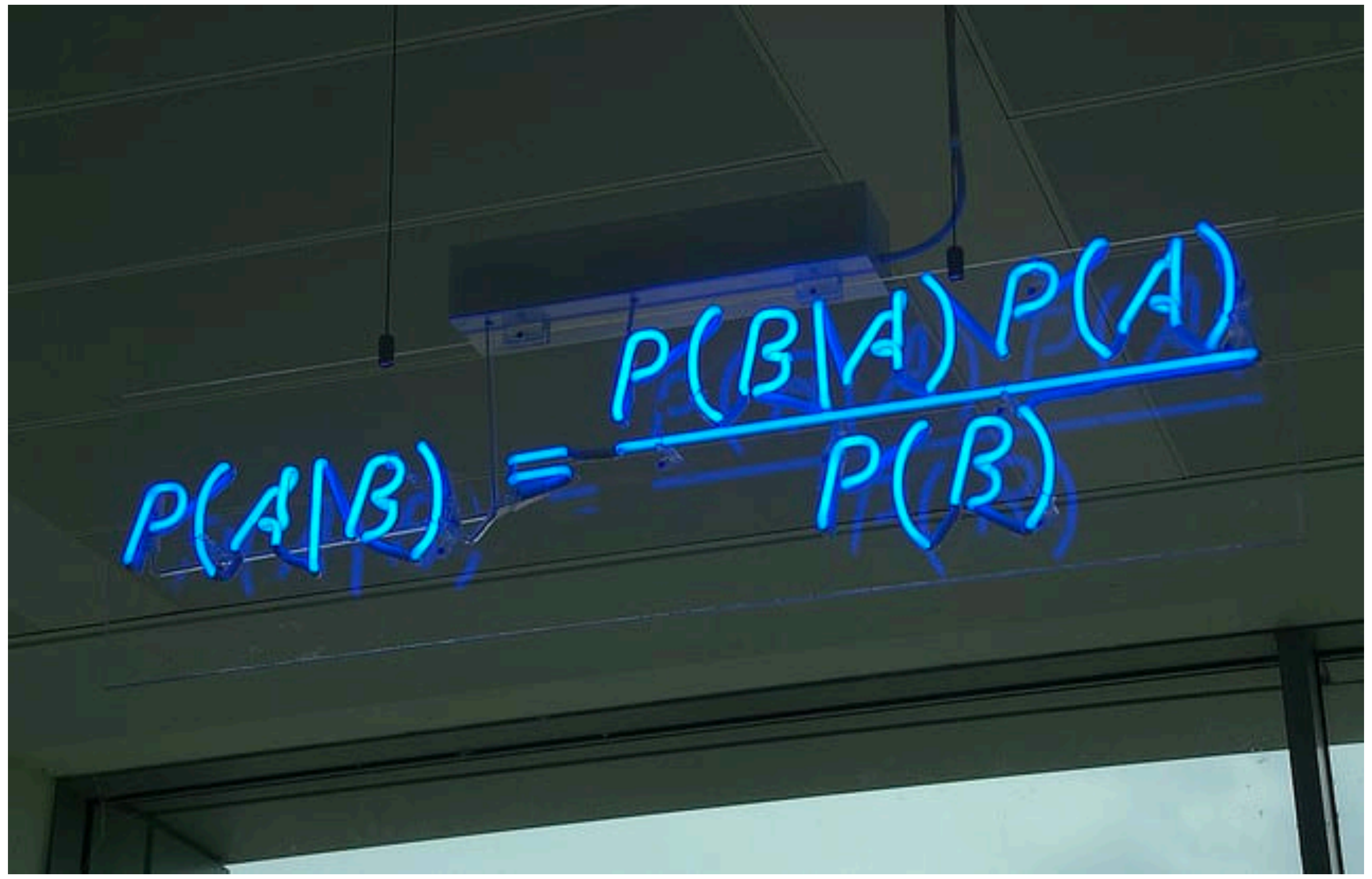


https://machinelearningmastery.com/naive-bayes-classifier-scratch-python/

- Like logistic regression, a classic algorithm which is no longer considered state-of-the-art
  - still very often useful in practice
- Relies on the Bayes Theorem
- And on what we know about probabilities of sequences

Naive Bayes

a classification algorithm



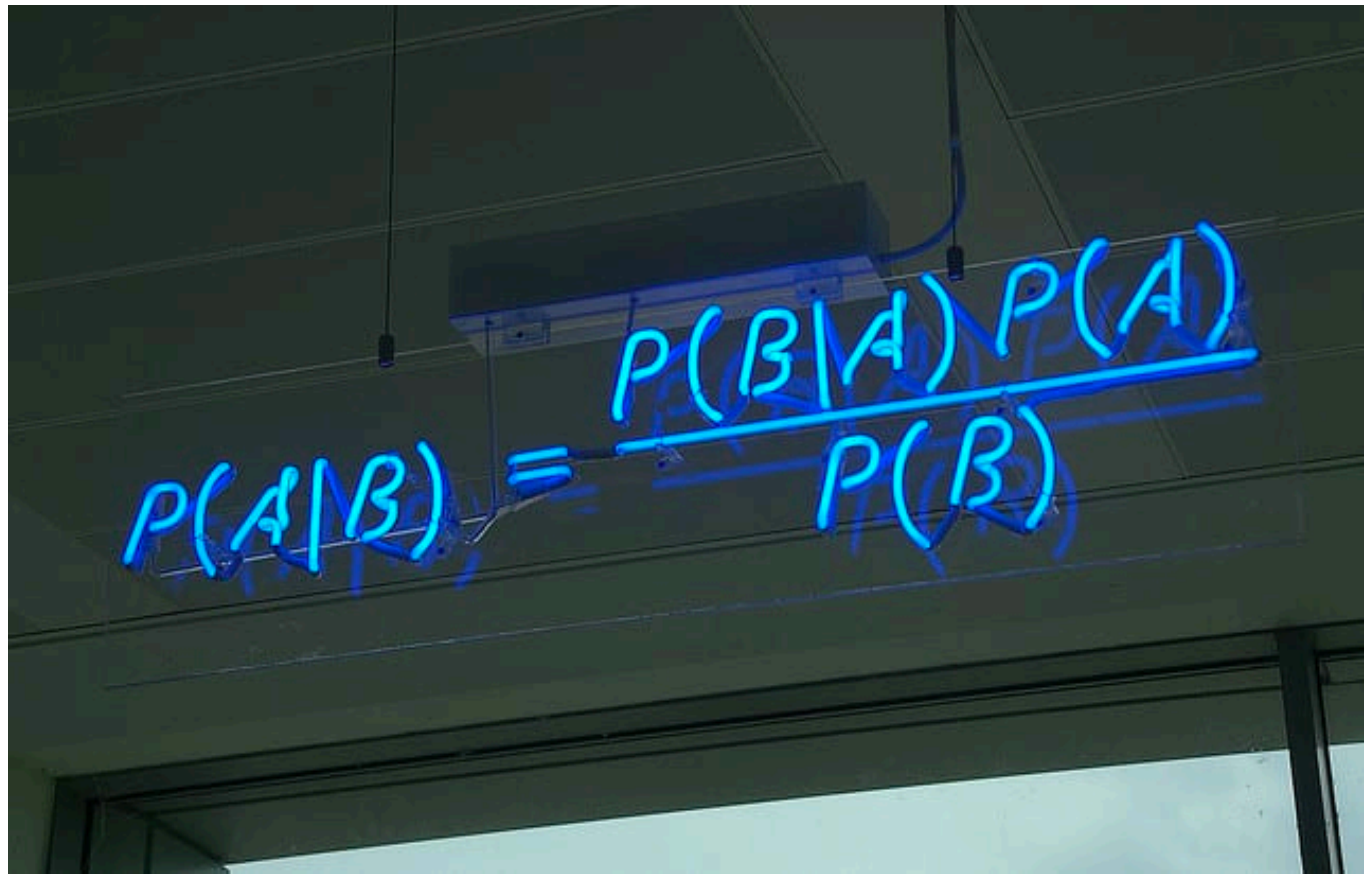
https://machinelearningmastery.com/naive-bayes-classifier-scratch-python/

- P(class|data) = P(data|class)*P(class)/P(data)
- P(POS|text) = P(text|POS)*P(POS)/P(text)
  - What's P(text)?!
    - e.g.: text = "This is a great film!"
    - P(text) = P(This)*P(is)*P(a)*P(great)*P(film)*P(!)
    - or:
    - P(text) = P(This)*P(great)*P(film)*P(!)
    - OK, what's P("great")?!

Naive Bayes

a classification algorithm

- P(text) = P(This)*P(is)*P(a)*P(great)*P(film)*P(!)
  - OK, what's P("great")?!
  - P("great") = count("great")/count(all words)
  - (not that trivial in practice but that's what it is conceptually)

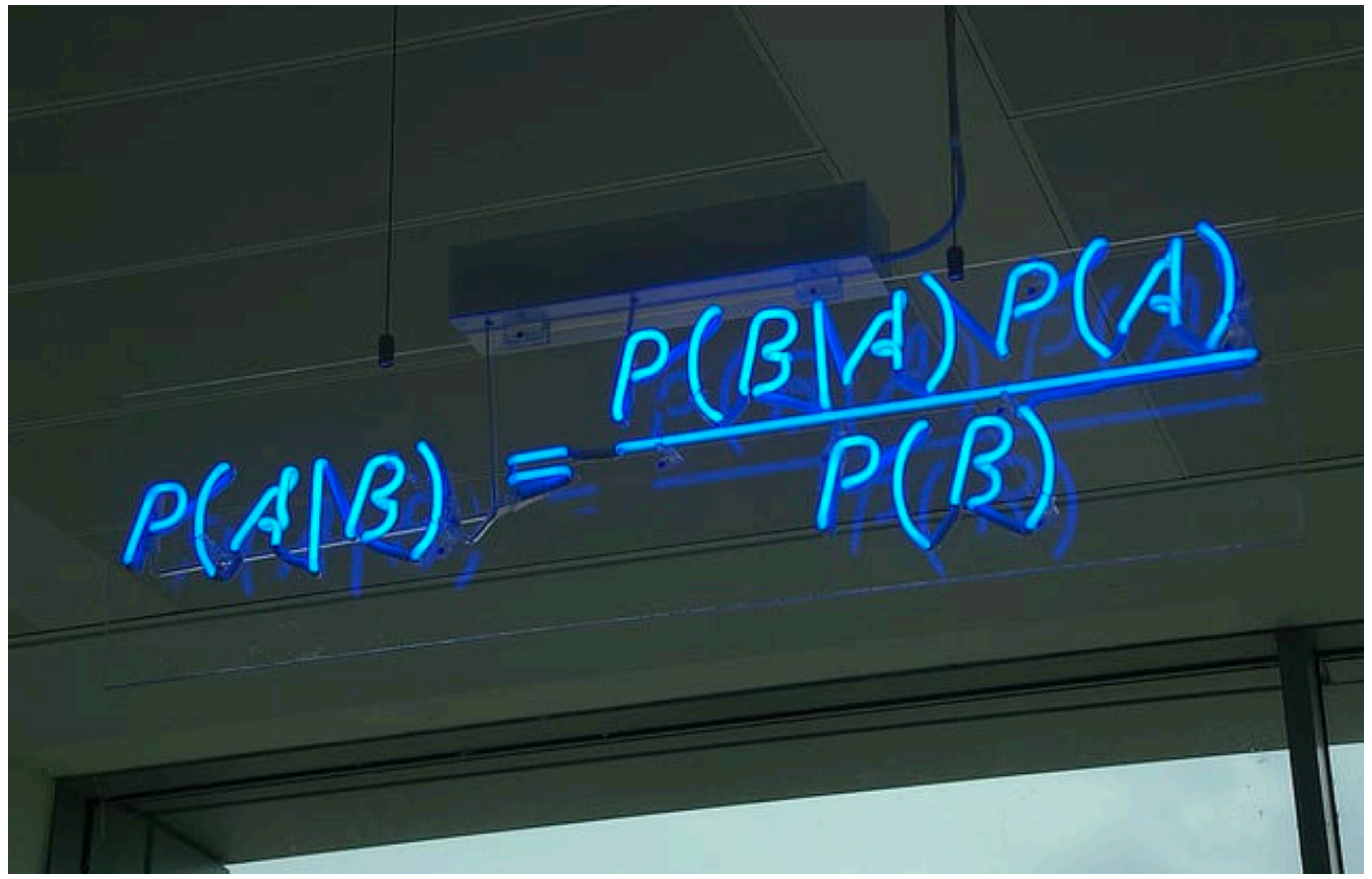


https://machinelearningmastery.com/naive-bayes-classifier-scratch-python/

- Naive Bayes relies on word counts to estimate probailities of word sequences
- ...and trains on labeled data
- ...to predict labels for unseen/unlabeled data

Naive Bayes
a classification algorithm

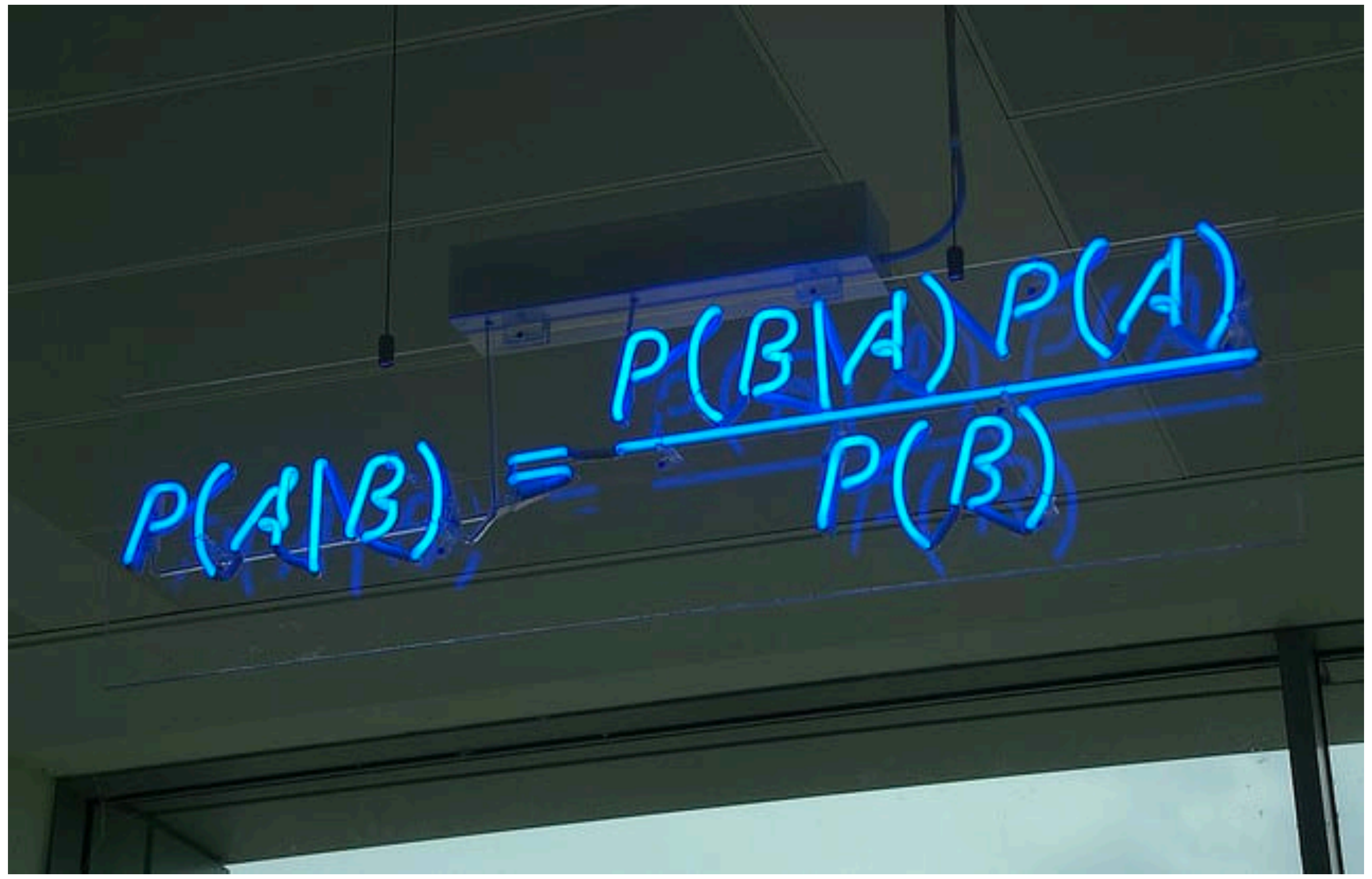
- Naive Bayes relies on word counts to estimate probailities of word sequences
  - ...and trains on labeled data
  - ...to predict labels for unseen/unlabeled data



https://machinelearningmastery.com/naive-bayes-classifier-scratch-python/

- What's "nontrivial" about it
  - Some words are noise
  - Do you care about the probability of "the"?
    - it is going to be the same in all texts, and very high
  - Well, that's easy: can clean that out
    - "stopwords", just remove them from text

Naive Bayes
a classification algorithm



https://machinelearningmastery.com/naive-bayes-classifier-scratch-python/

- Naive Bayes relies on word counts to estimate probailities of word sequences
  - ...and trains on labeled data
  - ...to predict labels for unseen/unlabeled data

- What's "nontrivial" about it?
  - What if you have never seen a word before?
  - It's count will be 0
  - It's probability will be 0
  - You multiply your terms by 0...
    - ...and P(entire text) = 0!
      - Not good!

Lecture survey:
in the chat