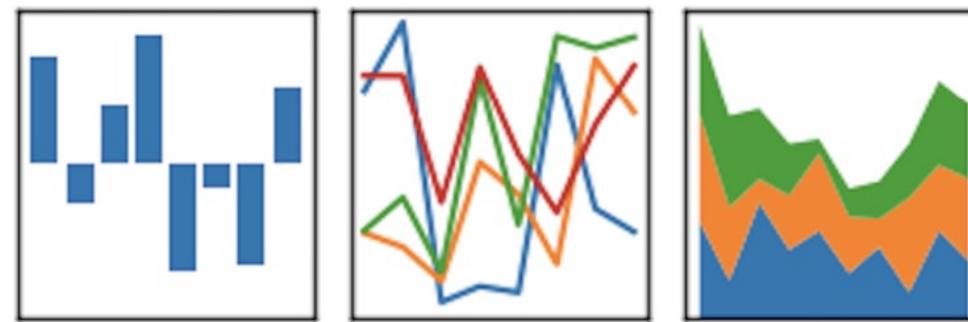


# Today's class

- Pandas

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



# Pandas Data Structures: Data Frames, Series, and Indices

# Pandas Data Structures

There are three fundamental data structures in pandas:

- Data Frame: 2D data tabular data.
- Series: 1D data. I usually think of it as columnar data.
- Index: A sequence of row labels.

Data Frame

	Candidate	Party	%	Year	Result
0	Obama	Democratic	52.9	2008	win
1	McCain	Republican	45.7	2008	loss
2	Obama	Democratic	51.1	2012	win
3	Romney	Republican	47.2	2012	loss
4	Clinton	Democratic	48.2	2016	loss
5	Trump	Republican	46.1	2016	win

Series

0	Obama
1	McCain
2	Obama
3	Romney
4	Clinton
5	Trump

Name: Candidate, dtype: object

Index

# The Relationship Between Data Frames, Series, and Indices

We can think of a Data Frame as a collection of Series that all share the same Index.

- Candidate, Party, %, Year, and Result Series all share an index from 0 to 5.

The diagram illustrates the relationship between five series and a DataFrame. Arrows point from each series name to its corresponding column in the DataFrame table. The 'Candidate Series' arrow points to the 'Candidate' column, the 'Party Series' arrow points to the 'Party' column, the '% Series' arrow points to the '%' column, the 'Year Series' arrow points to the 'Year' column, and the 'Result Series' arrow points to the 'Result' column. The DataFrame table contains six rows, indexed from 0 to 5, with data for each series.

	Candidate	Party	%	Year	Result
0	Obama	Democratic	52.9	2008	win
1	McCain	Republican	45.7	2008	loss
2	Obama	Democratic	51.1	2012	win
3	Romney	Republican	47.2	2012	loss
4	Clinton	Democratic	48.2	2016	loss
5	Trump	Republican	46.1	2016	win

# Indices Are Not Necessarily Row Numbers

Indices (a.k.a. row labels) can also:

- Be non-numeric.
- Have a name, e.g. “State”.

State	Motto	Translation	Language	Date Adopted
Alabama	Audemus jura nostra defendere	We dare defend our rights!	Latin	1923
Alaska	North to the future	—	English	1967
Arizona	Ditat Deus	God enriches	Latin	1863
Arkansas	Regnat populus	The people rule	Latin	1907
California	Eureka (Εὕρηκα)	I have found it	Greek	1849

# Indices

The row labels that constitute an index do not have to be unique.

- Left: The index values are all unique and numeric, acting as a row number.
- Right: The index values are named and non-unique.

Candidate	Party	%	Year	Result
0	Obama	Democratic	52.9	2008 win
1	McCain	Republican	45.7	2008 loss
2	Obama	Democratic	51.1	2012 win
3	Romney	Republican	47.2	2012 loss
4	Clinton	Democratic	48.2	2016 loss
5	Trump	Republican	46.1	2016 win

Candidate	Party	%	Result	
Year				
2008	Obama	Democratic	52.9	win
2008	McCain	Republican	45.7	loss
2012	Obama	Democratic	51.1	win
2012	Romney	Republican	47.2	loss
2016	Clinton	Democratic	48.2	loss
2016	Trump	Republican	46.1	win

# Column Names Are Usually Unique!

Column names in Pandas are almost always unique!

- Example: Really shouldn't have two columns named “Candidate”.

	Candidate	Party	%	Year	Result
0	Obama	Democratic	52.9	2008	win
1	McCain	Republican	45.7	2008	loss
2	Obama	Democratic	51.1	2012	win
3	Romney	Republican	47.2	2012	loss
4	Clinton	Democratic	48.2	2016	loss
5	Trump	Republican	46.1	2016	win

# Indexing with The [] Operator

# Indexing by Column Names Using [] Operator

Given a dataframe, it is common to extract a Series or a collection of Series. This process is also known as “Column Selection” or sometimes “indexing by column”.

- Column name argument to [] yields Series.
- List argument to [] yields a Data Frame.

```
elections[ "Candidate" ].head(6)
```

Year	Candidate
1980	Reagan
1980	Carter
1980	Anderson
1984	Reagan
1984	Mondale
1988	Bush

Name: Candidate, dtype: object

```
elections[ [ "Candidate", "Party" ] ].head(6)
```

Year	Candidate	Party
1980	Reagan	Republican
1980	Carter	Democratic
1980	Anderson	Independent
1984	Reagan	Republican
1984	Mondale	Democratic
1988	Bush	Republican

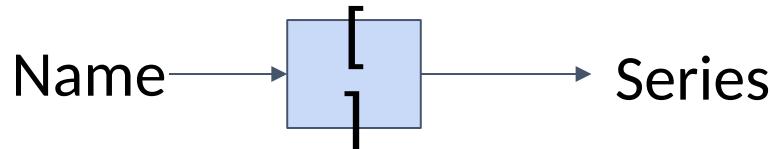
# Indexing by Row Slices Using [] Operator

We can also index by row numbers using the [] operator.

- Numeric slice argument to [] yields rows.
- Example: [0:3] yields rows 0 to 2.

elections[0:3]				
	Candidate	Party	%	Result
Year				
1980	Reagan	Republican	50.7	win
1980	Carter	Democratic	41.0	loss
1980	Anderson	Independent	6.6	loss

# [] Summary

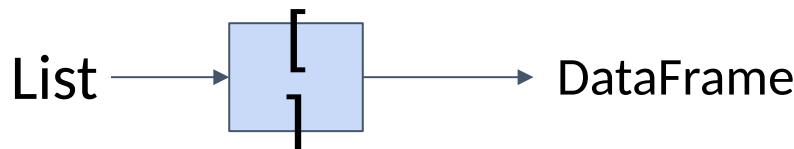


Single Column Selection

```
elections["Candidate"].head(6)
```

Year	Candidate
1980	Reagan
1980	Carter
1980	Anderson
1984	Reagan
1984	Mondale
1988	Bush

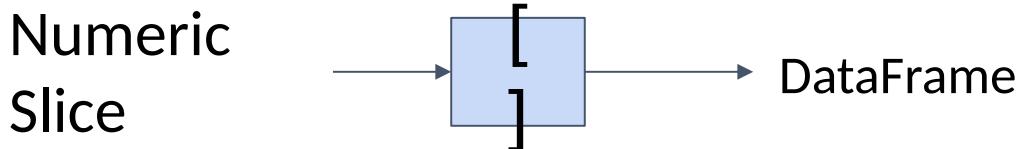
Name: Candidate, dtype: object



Multiple Column Selection

```
elections[["Candidate"]].head(6)
```

Year	Candidate
1980	Reagan
1980	Carter
1980	Anderson
1984	Reagan
1984	Mondale
1988	Bush



(Multiple) Row Selection

```
elections[0:3]
```

Year	Candidate	Party	%	Result
1980	Reagan	Republican	50.7	win
1980	Carter	Democratic	41.0	loss
1980	Anderson	Independent	6.6	loss

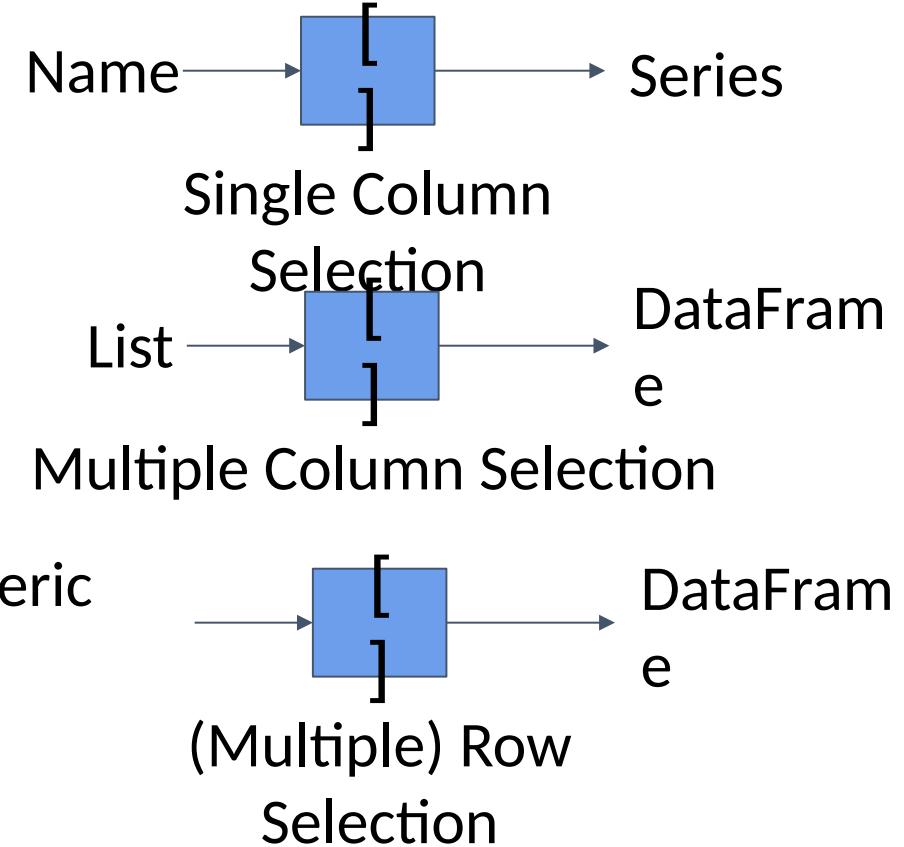
# Question

```
weird = pd.DataFrame({1:["topdog", "botdog"], "1": ["topcat", "botcat"]})  
weird
```

	1	1
0	topdog	topcat
1	botdog	botcat

Try to predict the output of the following:

- weird[1]
- weird["1"]
- weird[1:]



# Boolean Array Selection and Querying

# Boolean Array Input

Yet another input type supported by [] is the boolean array.

Entry number 7

```
elections[[False, False, False, False, False,  
          False, False, True, False, False,  
          True, False, False, False, True,  
          False, False, False, False, False,  
          False, False, True]]
```

	Candidate	Party	%	Year	Result
7	Clinton	Democratic	43.0	1992	win
10	Clinton	Democratic	49.2	1996	win
14	Bush	Republican	47.9	2000	win
22	Trump	Republican	46.1	2016	win



# Boolean Array Input

Yet another input type supported by [] is the boolean array. Useful because boolean arrays can be generated by using logical operators on Series.

Length 23 Series where every entry is “Republican”, “Democrat” or “Independent.”

Length 23 Series where every entry is either “True” or “False”, where “True” occurs for every independent candidate.

	Candidate	Party	%	Year	Result
2	Anderson	Independent	6.6	1980	loss
9	Perot	Independent	18.9	1992	loss
12	Perot	Independent	8.4	1996	loss

# Boolean Array Input

Boolean Series can be combined using the & operator, allowing filtering of results by multiple criteria.

```
elections[(elections['Result'] == 'win')  
          & (elections['%'] < 50)]
```

	Candidate	Party	%	Year	Result
7	Clinton	Democratic	43.0	1992	win
10	Clinton	Democratic	49.2	1996	win
14	Bush	Republican	47.9	2000	win
22	Trump	Republican	46.1	2016	win

# isin

The `isin` function makes it more convenient to find rows that match one of many possible values.

Example: Suppose we want to find “Republican” or “Democratic” candidates. Could use the `|` operator (`|` means or), or we can use `isin`.

- Ugly: `df[(df["Party"] == "Democratic") | (df["Party"] == "Republican")]`
- Better: `df[df["Party"].isin(["Republican", "Democratic"])]`

# The Query Command

The query command provides an alternate way to combine multiple conditions.

```
elections.query("Result == 'win' and Year < 2000")
```

	Candidate	Party	%	Year	Result
0	Reagan	Republican	50.7	1980	win
3	Reagan	Republican	58.8	1984	win
5	Bush	Republican	53.4	1988	win
7	Clinton	Democratic	43.0	1992	win
10	Clinton	Democratic	49.2	1996	win

**Indexing with .loc and .iloc  
Sampling with .sample**

# Loc and iloc

Loc and iloc are alternate ways to index into a DataFrame.

---

- They take a lot of getting used to! Documentation and ideas behind them are quite complex.
- I'll go over common usages (see docs for weirder ones).

Documentation:

- loc: <https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.loc.html>
- iloc: <https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.iloc.html>
- More general docs on indexing and selecting: [Link](#)

# Loc

Loc does two things:

---

- Access values by labels.
- Access values using a boolean array (a la Boolean Array Selection).

# Loc with Lists

The most basic use of loc is to provide a list of row and column labels, which returns a DataFrame.

```
elections.loc[[0, 1, 2, 3, 4], ['Candidate', 'Party', 'Year']]
```

	Candidate	Party	Year
0	Reagan	Republican	1980
1	Carter	Democratic	1980
2	Anderson	Independent	1980
3	Reagan	Republican	1984
4	Mondale	Democratic	1984

# Loc with Lists

The most basic use of loc is to provide a list of row and column labels, which returns a DataFrame.

```
elections_year_index.loc[[1980, 1984], ['Candidate', 'Party']]
```

	Candidate	Party
Year		
1980	Reagan	Republican
1980	Carter	Democratic
1980	Anderson	Independent
1984	Reagan	Republican
1984	Mondale	Democratic

# Loc with Slices

Loc is also commonly used with slices.

- Slicing works with all label types, not just numeric labels.
- Slices with loc are **inclusive**, not **exclusive**.

```
elections.loc[0:4, 'Candidate':'Year']
```

	Candidate	Party	Year
0	Reagan	Republican	1980
1	Carter	Democratic	1980
2	Anderson	Independent	1980
3	Reagan	Republican	1984
4	Mondale	Democratic	1984

# Loc with Slices

Loc is also commonly used with slices.

- Slicing works with all label types, not just numeric labels.
- Slices with loc are **inclusive**, not **exclusive**.

```
elections_year_index.loc[1980:1984, 'Candidate':'Party']
```

	Candidate	Party
Year		
1980	Reagan	Republican
1980	Carter	Democratic
1980	Anderson	Independent
1984	Reagan	Republican
1984	Mondale	Democratic

# Loc with Single Values for Column Label

If we provide only a single label as column argument, we get a Series.

```
elections.loc[0:4, 'Candidate']
```

```
0      Reagan
1      Carter
2    Anderson
3      Reagan
4     Mondale
Name: Candidate, dtype: object
```

# Loc with Single Values for Column Label

As before with the [] operator, if we provide a list of only one label as an argument, we get back a dataframe.

```
elections.loc[0:4, 'Candidate']
```

```
0      Reagan  
1      Carter  
2    Anderson  
3      Reagan  
4    Mondale  
Name: Candidate, dtype: object
```

```
elections.loc[0:4, ['Candidate']]
```

	Candidate
0	Reagan
1	Carter
2	Anderson
3	Reagan
4	Mondale

# Loc with Single Values for Row Label

If we provide only a single row label, we get a Series.

- Such a series represents a ROW not a column!
- The index of this Series is the names of the columns from the data frame.
- Putting the single row label in a list yields a dataframe version.

```
elections.loc[0, 'Candidate':'Year']
```

Candidate	Reagan
Party	Republican
%	50.7
Year	1980
Name:	0, dtype: object

```
elections.loc[[0], 'Candidate':'Year']
```

	Candidate	Party	%	Year
0	Reagan	Republican	50.7	1980

# Loc Supports Boolean Arrays

Loc supports Boolean Arrays exactly as you'd expect.

```
elections.loc[(elections['Result'] == 'win') & (elections['%'] < 50), 'Candidate':'%']
```

	Candidate	Party	%
7	Clinton	Democratic	43.0
10	Clinton	Democratic	49.2
14	Bush	Republican	47.9
22	Trump	Republican	46.1

# iloc: Integer-Based Indexing for Selection by Position

In contrast to loc, iloc doesn't think about labels at all. Instead, it returns the items that appear in the numerical positions specified.

elections.iloc[0:3, 0:3]			
	Candidate	Party	%
0	Reagan	Republican	50.7
1	Carter	Democratic	41.0
2	Anderson	Independent	6.6

mottos.iloc[0:3, 0:3]			
	Motto	Translation	Language
<b>State</b>			
Alabama	Audemus jura nostra defendere	We dare defend our rights!	Latin
Alaska	North to the future	—	English
Arizona	Ditat Deus	God enriches	Latin

Advantages of loc:

- Harder to make mistakes.
- Easier to read code.
- Not vulnerable to changes to the ordering of rows/cols in raw data files.

Nonetheless, iloc can be more convenient. *Use iloc judiciously.*

# Annoying Question Challenge

Which of the following pandas statements returns a DataFrame of the first 3 Candidate names only for candidates that won with more than 50% of the vote.

```
elections.iloc[[0, 3, 5], [0, 3]]
```

```
elections.loc[[0, 3, 5], ["Candidate": "Year"]]
```

```
elections.loc[elections["%"] > 50, ["Candidate", "Year"]].head(3)
```

```
elections.loc[elections["%"] > 50, ["Candidate",  
"Year"]].iloc[0:2, :]
```

	Candidate	Party	%	Year	Result
0	Reagan	Republican	50.7	1980	win
1	Carter	Democratic	41.0	1980	loss
2	Anderson	Independent	6.6	1980	loss
3	Reagan	Republican	58.8	1984	win
4	Mondale	Democratic	37.6	1984	loss
5	Bush	Republican	53.4	1988	win
6	Dukakis	Democratic	45.6	1988	loss



	Candidate	Year
0	Reagan	1980
3	Reagan	1984
5	Bush	1988

# Annoying Question Challenge

Which of the following pandas statements returns a DataFrame of the first 3 Candidate names only for candidates that won with more than 50% of the vote.

`elections.iloc[[0, 3, 5], [0, 3]]`

`elections.loc[[0, 3, 5], ["Candidate": "Year"]]`

`elections.loc[elections["%"] > 50, ["Candidate", "Year"]].head(3)`

`elections.loc[elections["%"] > 50, ["Candidate", "Year"]].head(3).iloc[0:2, :]`

	Candidate	Party	%	Year	Result
0	Reagan	Republican	50.7	1980	win
1	Carter	Democratic	41.0	1980	loss
2	Anderson	Independent	6.6	1980	loss
3	Reagan	Republican	58.8	1984	win
4	Mondale	Democratic	37.6	1984	loss
5	Bush	Republican	53.4	1988	win
6	Dukakis	Democratic	45.6	1988	loss



	Candidate	Year
0	Reagan	1980
3	Reagan	1984
5	Bush	1988

See notebook for why!

# Sample

If you want a DataFrame consisting of a random selection of rows, you can use the sample method.

- By default, *it is without replacement*. Use replace=True for replacement.
- Naturally, can be chained with our selection operators [], loc, iloc.

`elections.sample(10)`

	Candidate	Party	%	Year	Result
15	Kerry	Democratic	48.3	2004	loss
16	Bush	Republican	50.7	2004	win
22	Trump	Republican	46.1	2016	win
9	Perot	Independent	18.9	1992	loss
21	Clinton	Democratic	48.2	2016	loss
11	Dole	Republican	40.7	1996	loss
20	Romney	Republican	47.2	2012	loss
14	Bush	Republican	47.9	2000	win
8	Bush	Republican	37.4	1992	loss
1	Carter	Democratic	41.0	1980	loss

`elections.query("Year < 1992").sample(4, replace=True)`

	Candidate	Party	%	Year	Result
1	Carter	Democratic	41.0	1980	loss
4	Mondale	Democratic	37.6	1984	loss
6	Dukakis	Democratic	45.6	1988	loss
1	Carter	Democratic	41.0	1980	loss

# Handy Properties and Utility Functions for Series and DataFrames

# Numpy Operations

Pandas Series and DataFrames support a large number of operations, including mathematical operations so long as the data is numerical.

---

```
winners = elections.query("Result == 'win'")[%]
winners
```

```
0      50.7
3      58.8
5      53.4
7      43.0
10     49.2
14     47.9
16     50.7
17     52.9
19     51.1
22     46.1
Name: %, dtype: float64
```

```
np.mean(winners)
```

```
50.38
```

```
max(winners)
```

```
58.8
```

# **head, size, shape, and describe**

**head**: Displays only the top few rows.

---

**size**: Gives the total number of data points.

**shape**: Gives the size of the data in rows and columns.

**describe**: Provides a summary of the data.

# **index and columns**

`index`: Returns the index (a.k.a. row labels).

---

`columns`: Returns the labels for the columns.

# The `sort_values` Method

One incredibly useful method for DataFrames is `sort_values`, which creates a copy of a DataFrame sorted by a specific column.

```
elections.sort_values('%', ascending=False)
```

	Candidate	Party	%	Year	Result
3	Reagan	Republican	58.8	1984	win
5	Bush	Republican	53.4	1988	win
17	Obama	Democratic	52.9	2008	win
19	Obama	Democratic	51.1	2012	win
0	Reagan	Republican	50.7	1980	win

# The `sort_values` Method

We can also use `sort_values` on a Series, which returns a copy with the values in order.

```
mottos['Language'].sort_values().head(5)
```

State	Language
Washington	Chinook Jargon
Wyoming	English
New Jersey	English
New Hampshire	English
Nevada	English

Name: Language, dtype: object

# The `value_counts` Method

Series also has the function `value_counts`, which creates a new Series showing the counts of every value.

```
elections['Party'].value_counts()
```

```
Democratic      10
Republican      10
Independent     3
Name: Party, dtype: int64
```

# The unique Method

Another handy method for Series is `unique`, which returns all unique values as an array.

```
mottos['Language'].unique()
```

```
array(['Latin', 'English', 'Greek', 'Hawaiian', 'Italian', 'French',
       'Spanish', 'Chinook Jargon'], dtype=object)
```

# The Things We Just Saw

---

- sort\_values
  - value\_counts
  - unique
-

# Summary

- Operations on String series, e.g. `babynames["Name"].str.startswith()`
  - Creating and dropping columns.
    - Creating temporary columns is often convenient for sorting.
  - Passing an index as an argument to `loc`.
    - Useful as an alternate way to sort a dataframe.
-



# Welcome to YSC2239!

Hengnan (Henry) Hu

Spring 2023

# Today's class

- ❑ Course organization
- ❑ Introduction: What is Data Science?
- ❑ Jupyter notebook for Python
- ❑ Demo: Huckleberry Fin,  
Little Women

# Course staff

- **Instructor:**

- *Hengnan Hu*
  - Email: [henry.hu@nus.edu.sg](mailto:henry.hu@nus.edu.sg)
  - Office Hours: By appointment over Zoom

- **Lectures:**

- Time: Monday, Thurs 7pm - 8:30pm
- Location: Y-LT1
- Remotely (need to check with IT and update over Canvas)

# Course staff (Peer tutors)

•John Jacob Go

Email:

[jacobgo@u.yale-nus.edu.sg](mailto:jacobgo@u.yale-nus.edu.sg)

•Nihal Zuhayar Parash Miaji

Email:

[nihalzuhayar@u.yale-nus.edu.sg](mailto:nihalzuhayar@u.yale-nus.edu.sg)

# Weekly drop-in sessions

- Weekly drop-in sessions will begin on **Week 2**
- Detailed schedule and venue: TBA

# Social distancing policy

- We suggest the following social distancing policy:
- 1. Wear masks in lectures, drop-in sessions and
- 2. Keep appropriate distance between you and other classmates, peer tutors and professors
- 3. Maintain good personal hygiene at all times

# Course contents

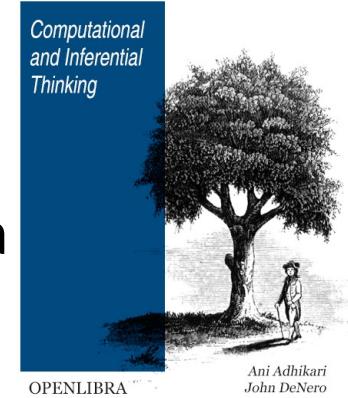
- Introduction of Python for data science: tables, data types, charts, histograms, functions, groups, joins, iteration
- Statistics: chance, sampling, models, distributions, A/B testing, confidence intervals, central limit theorem, correlation, p-values
- Machine learning: linear regression, multiple linear regression, regression diagnostics, feature engineering, logistic regression, classification, clustering, decision tree

# Course resources

- We will follow the lecture slides and demo of the instructor.
- I strongly encourage bringing your laptop to lecture and play around with the demo yourself in real-time for best learning experience.

# Course resources

The Foundations of Data Science



## Text for the first-half:

- Title: Computational and Inferential Thinking: The Foundations of Data Science
- Authors: Ani Adhikari and John DeNero
- Link: <https://inferentialthinking.com/chapters/intro.html>
- We will use Canvas for communication and posting of material

# Assessment Scheme

## **COURSE ASSESSMENT BREAKDOWN**

Attendance and participation 5%

Assignments 20%

Labs 15%

Project 8%

Peer evaluation 2%

Midterm 20%

Final 30%

# Assessment Scheme

Attendance: QR code at the beginning of each lecture + 4 in-class quizzes

Assignments: weekly in the first half of the course. First assignment will be posted on Week 2, **due on Friday 23:59.**

Labs: weekly. First lab will be posted on Week 1, **due on Wednesday 23:59.**

Midterm exam: **March 2nd (Thursday) 7pm – 8:30pm, lecture-time, in-person.** Please block this timeslot. No make-up exams will be given for midterm exam.

Final exam: April 28<sup>th</sup> (Friday) 3pm – 5pm, **please take note this timeslot.** No make-up exams will be given for final exam.

# Project

The course culminates in a final group project in which students are expected to create appropriate data science models using the methods covered in class for data analysis. Each group composes of 2-3 students, except in special circumstances approved by the instructor.

Further details on the projects will be provided as the due dates approach.

# Late work policy

- Checkout the syllabus on penalty for late work policy

# Other policies and resources

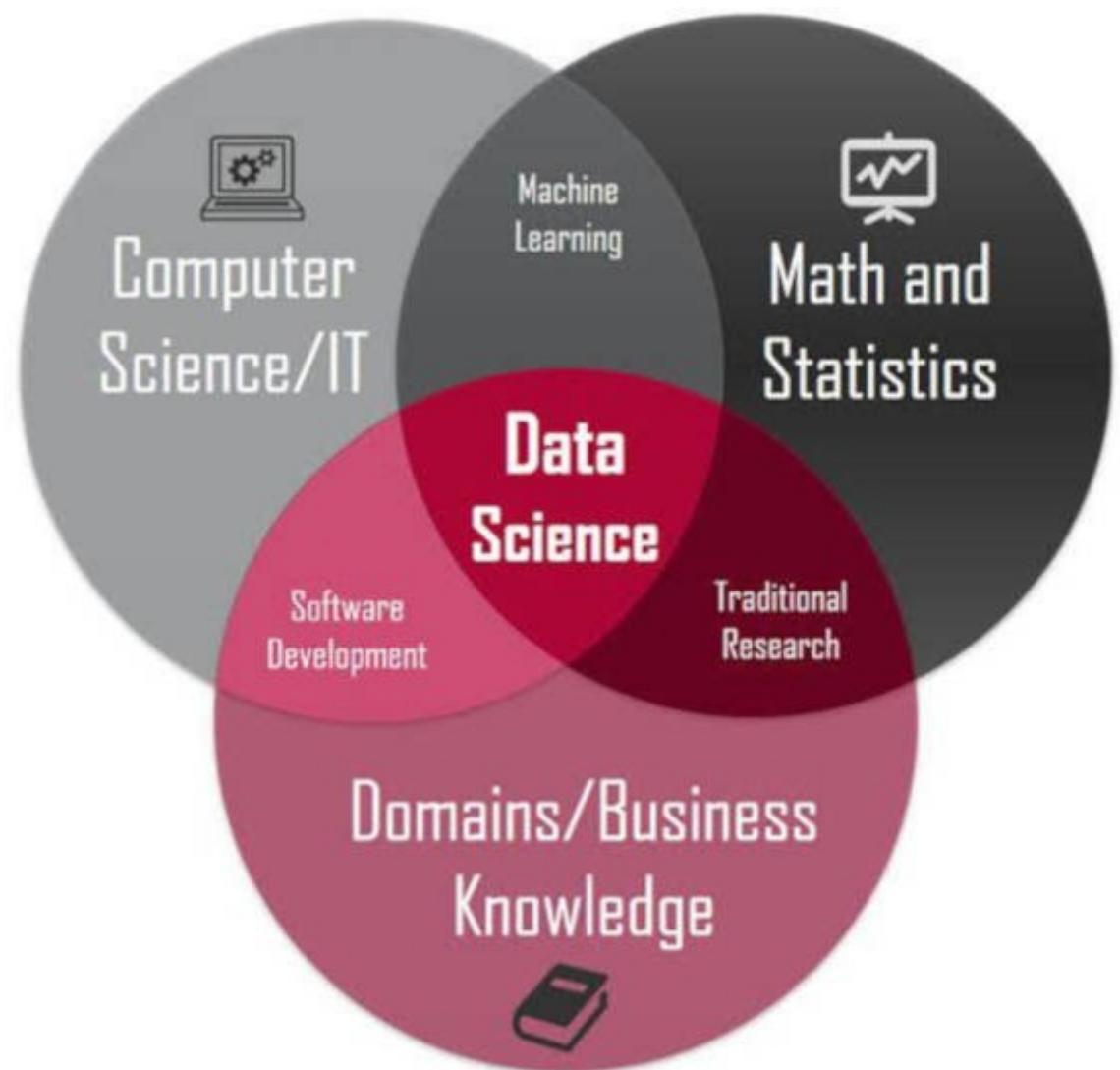
- Checkout the syllabus on penalty for academic integrity policy, intellectual property and privacy, class climate, etc.

# Today's class

- Course organization
- Introduction: What is Data Science?
- Jupyter notebook for Python
- Demo: Huckleberry Fin,  
Little Women

# Data Science

Reading: Book Chapter 1



# Data is the new oil

- We now live in a time of big data
- Ability to leverage these data to make intelligent decisions and to help designing faster and better algorithms are very important
- A wide range of fields that rely on data-driven decision making, e.g. Biology, Economics, self-driving cars etc..

# Applications of data science: computer vision and self-driving cars



Scharfsinn/Shutterstock.com

# Applications of data science: accelerate scientific discovery in protein folding

- Google Deepmind has developed AlphaFold that can accurately predict 3D models of protein structures and is accelerating research in nearly every field of biology:
- <https://www.deepmind.com/research/highlighted-research/alphafold>

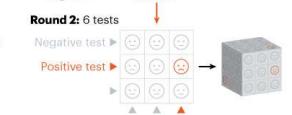
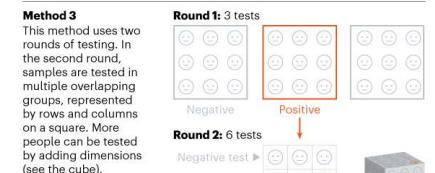
# Applications of data science: combat COVID-19

- Group testing
- <https://www.nature.com/articles/d41586-020-02053-6>



## GROUP TESTING

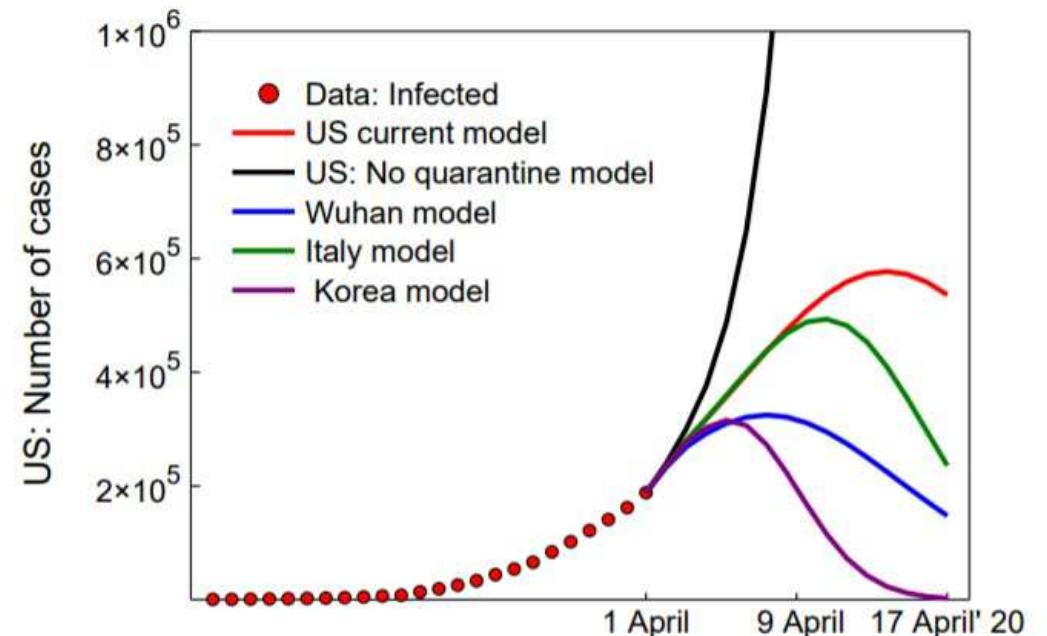
Countries can save time and money by testing many people at once. Researchers are trialling various methods for group testing.



©nature

# Applications of data science: combat COVID-19

- Forecasting models
- <https://projects.fivethirtyeight.com/covid-forecasts/>

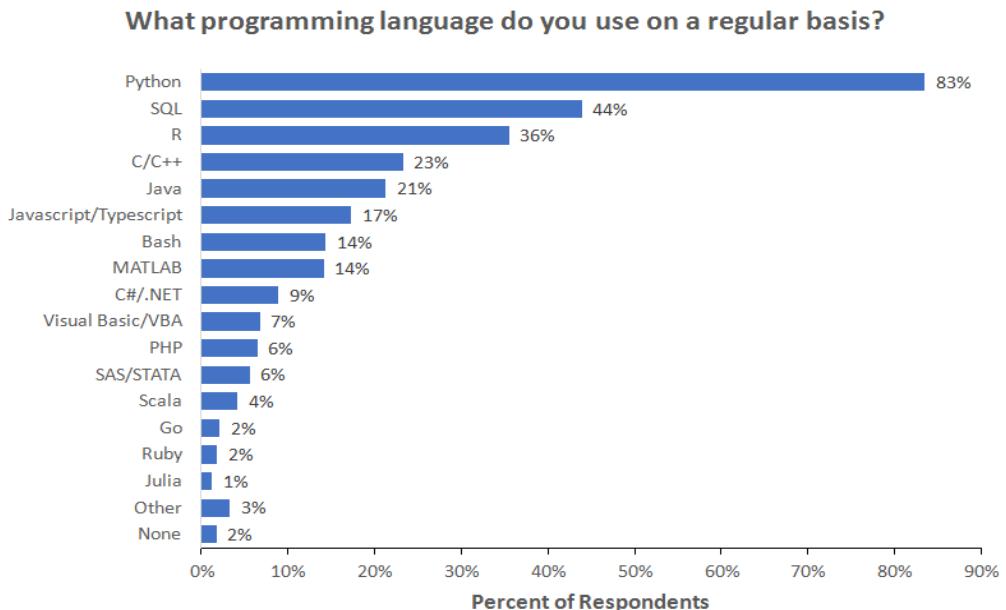


# Today's class

- Course organization
- Introduction: What is Data Science?
- Jupyter notebook for Python
- Demo: Huckleberry Fin,  
Little Women

# Python

- Python is a very popular computing language for data science and software development



Note: Data are from the 2018 Kaggle Machine Learning and Data Science Survey. You can learn more about the study here: <http://www.kaggle.com/kaggle/kaggle-survey-2018>. A total of 18827 respondents answered the question.



# Jupyter notebook

- In this course, we will be using Jupyter notebook with Python **3** for all of the Labs, Assignments and Exams!
- Install the latest version: <https://www.anaconda.com/distribution/>
- Checkout the installation guide on Canvas
- Please let the instructors or peer tutors know if you have any issue on installation

# datascience package in Python

- We shall need the datascience package in Python for the first half of the course
- Checkout the installation guide on Canvas

# Python packages that we will learn in this course

- Datascience
- Pandas
- Scikit-learn
- NumPy
- Matplotlib



IP[y]: IPython  
Interactive Computing



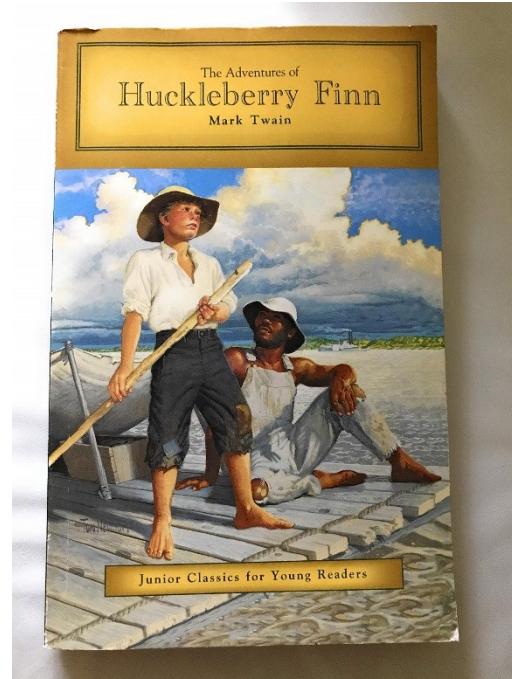
# Today's class

Course organization

Introduction: What is Data Science?

Jupyter notebook for Python

Demo: Huckleberry Fin,  
Little Women



# To do

- Install Jupyter notebook and the datascience package
- Lab 0 (ungraded)

**YaleNUSCollege**

**YSC2239 Lecture 02**

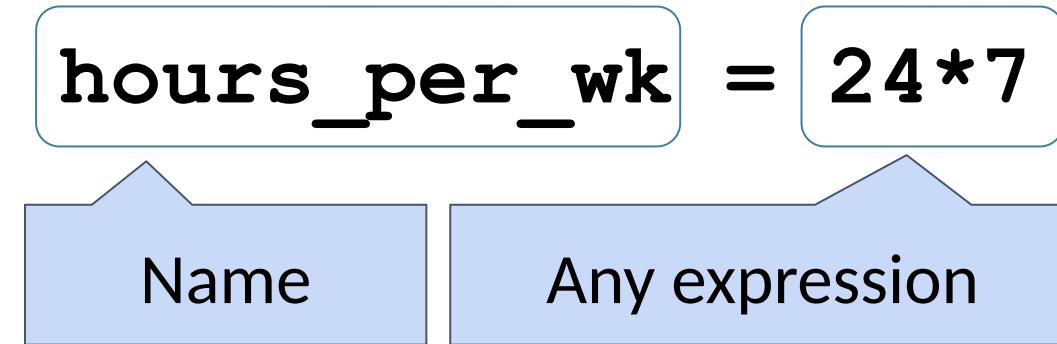
# Today's class

- Python basics
  - Tables
  - Data types
- 
- Reading: Chapter 3, 4, 5

# Names

# Assignment Statements

---



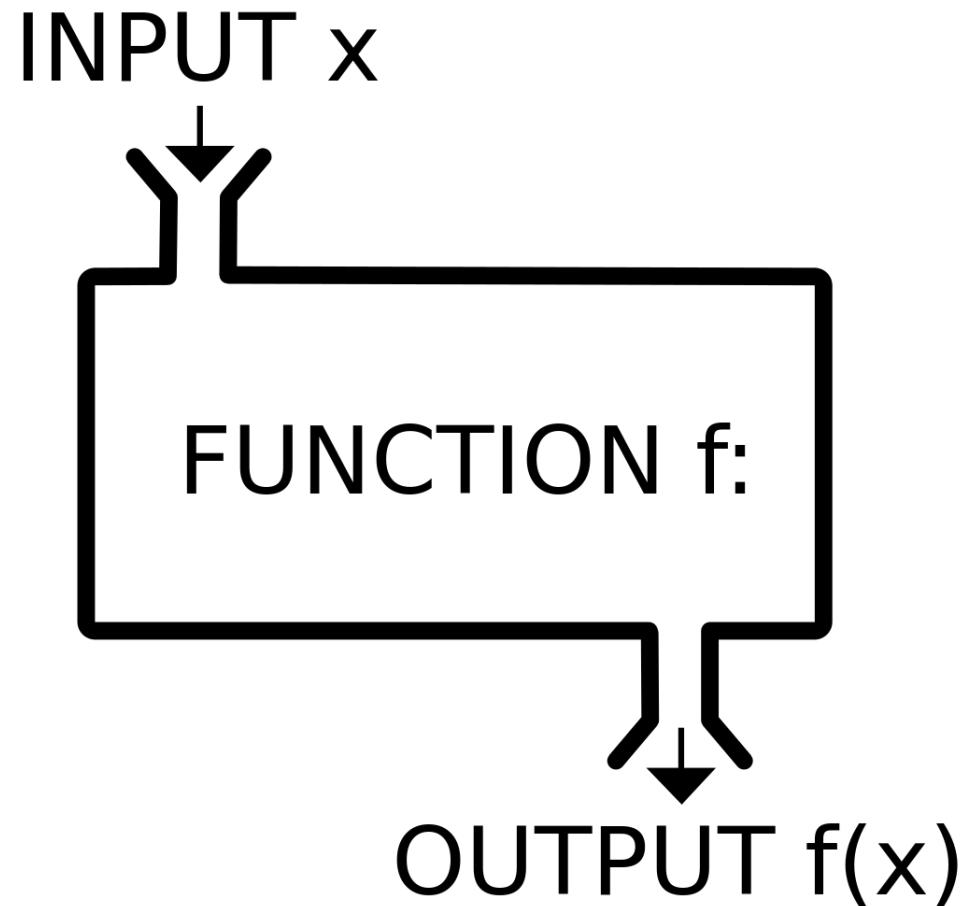
- Statements don't have a value; they perform an action
- An assignment statement changes the meaning of the name to the left of the = symbol
- The name is bound to a value (not an equation)

(Demo)

---

# Functions

# Functions



# Anatomy of a Call Expression

---

What function to call

Argument/parameter/input to  
the function

The diagram shows the call expression `f(27)`. A dashed blue rectangular box encloses the identifier `f`, and another dashed blue rectangular box encloses the argument `27`. The boxes overlap at their right edges.

`f ( 27 )`

"Call `f` on `27`."

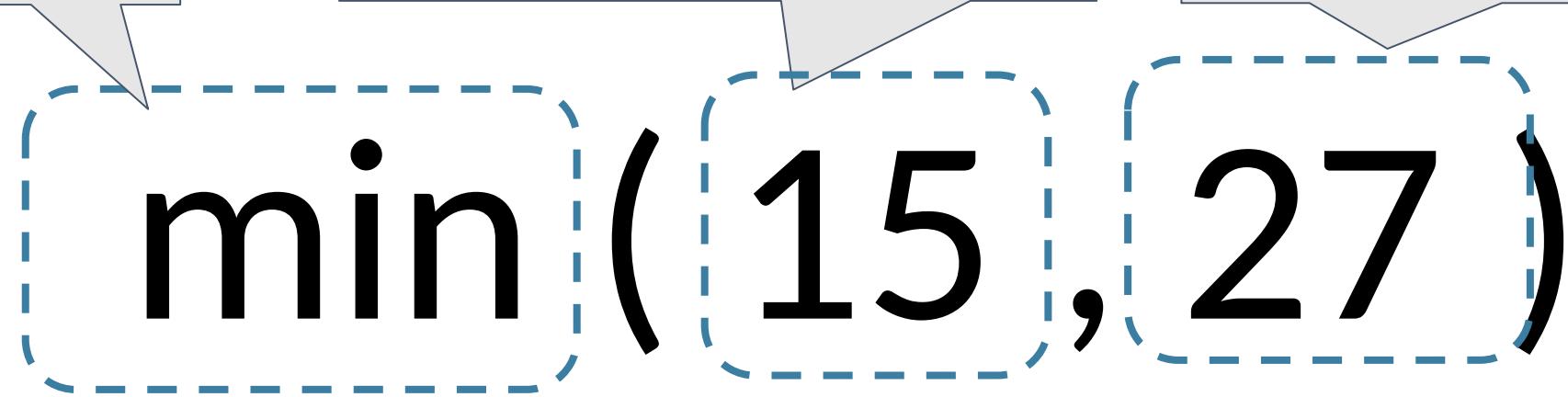
---

# Anatomy of a Call Expression

What  
function  
to call

First  
argument/parameter

Second  
argument/parameter



(Demo)

# Tables

# Table Structure

- A Table is a sequence of labeled columns
- Each row represents one individual
- Data within a column represents one attribute of the individuals

The diagram illustrates a table structure with three columns: Name, Code, and Area (m2). The first row contains the column labels. The second row contains the data for California, and the third row contains the data for Nevada. A blue callout labeled "Label" points to the "Code" column header. A blue callout labeled "Row" points to the second row. A blue callout labeled "Column" points to the "Code" column. A blue callout labeled "(Demo)" is positioned at the bottom right.

Name	Code	Area (m2)
California	CA	163696
Nevada	NV	110567

Row

Column

Label

(Demo)

# Some Table Operations

---

- `t.select(label)` - constructs a new table with just the specified columns
- `t.drop(label)` - constructs a new table in which the specified columns are omitted
- `t.sort(label)` - constructs a new table with rows sorted by the specified column
- `t.where(label, condition)` - constructs a new table with just the rows that match the condition

# Numbers

(Demo)

# Ints and Floats

---

Python has two real number types

- **int**: an integer of any size
- **float**: a number with an optional fractional part

An **int** never has a decimal point; a **float** always does

A **float** might be printed using scientific notation

Three limitations of float values:

- They have limited size (but the limit is huge)
- They have limited precision of 15-16 decimal places
- After arithmetic, the final few decimal places can be wrong

# Strings

(Demo)

# Text and Strings

---

A string value is a snippet of text of any length

- 'a'
- 'word'
- "there can be 2 sentences. Here's the second!"

Strings consisting of numbers can be converted to numbers

- int('12')
- float('1.2')

Any value can be converted to a string

- str(5)
-

# Discussion Question

---

Assume you have run the following statements:

x = 3

y = '4'

z = '5.6'

What's the source of the error in each example?

A. x + y

B. x + int(y + z)

C. str(x) + int(y)

D. y + float(z)

# Types

(Demo)

# Every value has a type

---

We've seen 5 types so far:

- int: 2
- float: 2.2
- str: 'Red fish, blue fish'
- builtin\_function\_or\_method: abs
- Table

The `type` function can tell you the type of a value

- `type(2)`
- `type(2 + 2)`

An expression's “type” is based on its value, not how it looks

- `x = 2`
  - `type(x)`
-

# Conversions

---

Strings that contain numbers can be converted to numbers

- `int('12')`
- `float('1.2')`
- ~~`float('one point two')`~~ # Not a good idea!

Any value can be converted to a string

- `str(5)`

Numbers can be converted to other numeric types

- `float(1)`
- `int(1.2)` # DANGER: loses information!

# To do

- Lab 1

**YaleNUSCollege**

**YSC2239 Lecture 3**

# Today's class

- Arrays
  - Creating new tables
  - Manipulating columns of tables
- 
- Reading: Chapter 5, 6.1, 6.2

# Arrays

# Arrays

---

An array contains a sequence of values

- All elements of an array should have the same type
- Arithmetic is applied to each element individually
- Adding arrays adds elements (**if same length!**)
- A column of a table is an array

(Demo)

---

# Ranges

---

A range is an array of consecutive numbers

- `np.arange(end)` :  
An array of increasing integers from 0 up to `end`
- `np.arange(start, end)` :  
An array of increasing integers from `start` up to `end`
- `np.arange(start, end, step)` :  
A range with `step` between consecutive values

The range always includes `start` but excludes `end`

---

# Building Tables

# Ways to create a table

---

- `Table.read_table(filename)` - reads a table from a spreadsheet
- `Table()` - an empty table
- and... `select`, `where`, `sort` and so on all create new tables

# Example

# W. E. B. Du Bois, 1868-1963

---

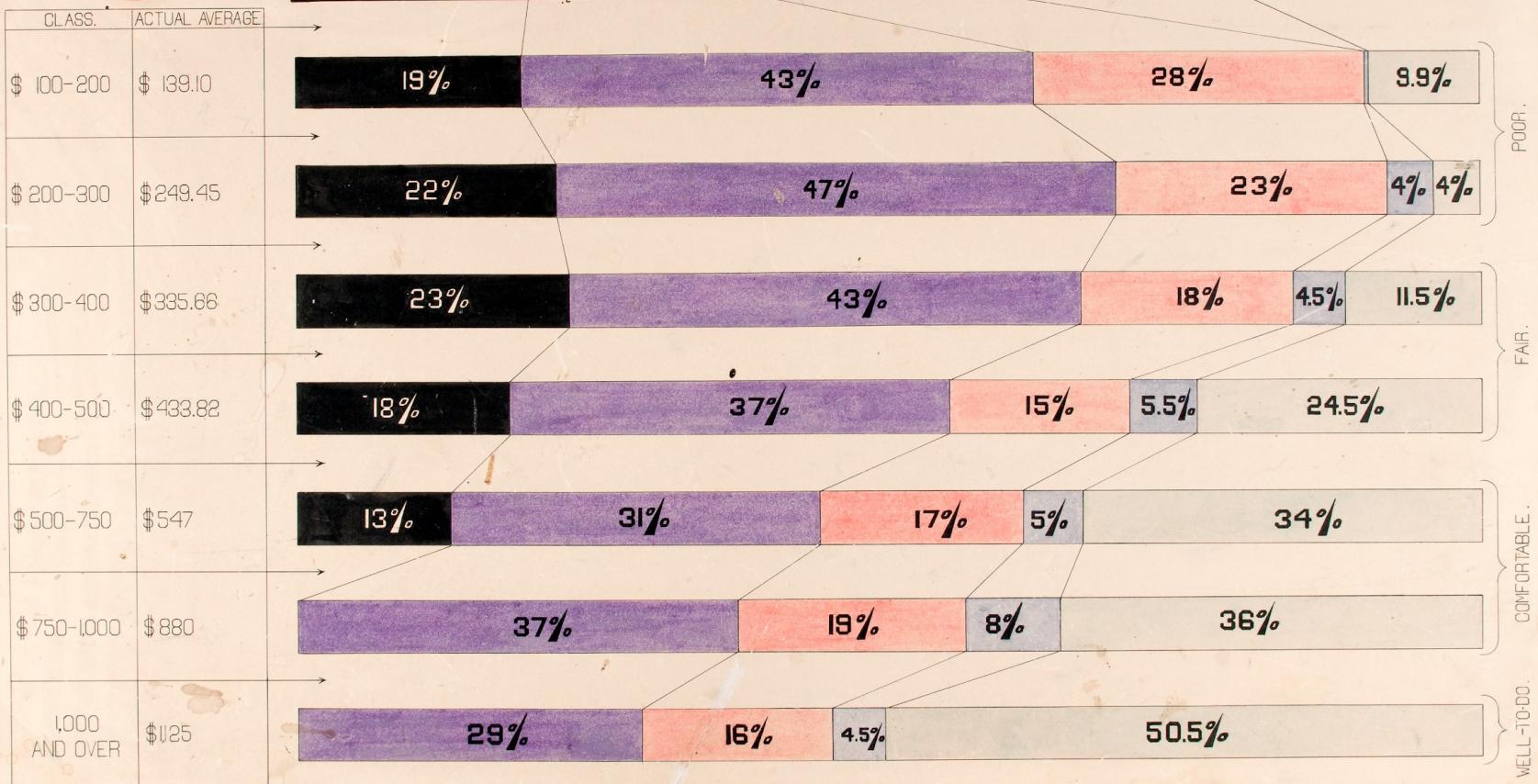


- Scholar, historian, activist, and data scientist
- NAACP founder
- Made a series of visualizations for the 1900 Paris Exposition
  - Goal: change the way people see Black Americans
  - Hundreds of photographs and patents
  - 60+ handmade graphs in 3 months

# INCOME AND EXPENDITURE OF 150 NEGRO FAMILIES IN ATLANTA, GA., U.S.A.



ANNUAL EXPENDITURE FOR				
RENT.	FOOD.	CLOTHES.	DIRECT TAXES.	OTHER EXPENSES AND SAVINGS.
			<p>THE STATE TAX RATE IS: 1880 - \$ 3.50 PER \$1,000 1885 - \$ 3.50 .. .. 1890 - \$ 3.96 .. .. 1895 \$ 4.56 .. .. 1899 \$ 5.36 .. .. STATE AND COUNTY TAXES RAISE THIS TO \$21 PER \$1,000 IN ATLANTA.</p>	<p>THE HIGHER LIFE. RELIGION. ART. EDUCATION. SICKNESS. SAVINGS. AMUSEMENTS. BOOKS AND PAPERS. TRAVEL.</p>



(Demo)

# Discussion Question

---

Use the table functions we learned this week to find the income bracket (“class”) that spent the highest percentage of their income on rent.

# Table Methods

---

- Creating and extending tables:
  - `Table().with_column` and `Table.read_table`
- Finding the size: `num_rows` and `num_columns`
- Referring to columns: labels, relabeling, and indices
  - `labels` and `relabelled`; column indices start at 0
- Accessing data in a column
  - `column` takes a label or index and returns an array
- Using array methods to work with data in columns
  - `item`, `sum`, `min`, `max`, and so on
- Creating new tables containing some of the original columns:
  - `select`, `drop`

---

(Demo)

# Manipulating Rows

---

- `t.sort(column)` sorts the rows in increasing order
- `t.sort(column, descending=True)` sorts the rows in decreasing order
- `t.take(row_numbers)` keeps the numbered rows
  - Each row has an index, starting at 0
- `t.where(column, are.condition)` keeps all rows for which a column's value satisfies a condition
- `t.where(column, value)` keeps all rows for which a column's value equals some particular value
  - Same as `t.where(column, are.equal_to(value))`

# Discussion Questions

---

The table **nba** has columns **PLAYER**, **POSITION**, and **SALARY**.

- a) Create an array containing the names of all point guards (PG) who make more than \$15M/year

```
guards = nba.where('POSITION', 'PG')  
guards.where('SALARY', are.above(15)).column('PLAYER')
```

- b) How to combine two tables into one?

```
nba.where('POSITION', 'PG').append(nba.where('POSITION', 'C')).show(100)
```

(Demo)

---

# Attribute Types

# Types of Attributes

---

All values in a column of a table should be both the same type and be comparable to each other in some way

- **Numerical** — Each value is from a numerical scale
  - Numerical measurements are ordered
  - Differences are meaningful
- **Categorical** — Each value is from a fixed inventory
  - May or may not have an ordering
  - Categories are the same or different

# To do

- Lab 2 already posted on Canvas and due on Wednesday

**YaleNUSCollege**

**YSC2239 Lecture 4**

# Today's class

- Census data
- Charts
- Histograms
- Reading: Chapter 6.3, 6.4, 7

# Census Data

# The Decennial Census

---

- Every ten years, the Census Bureau counts how many people there are in the U.S.
- In between censuses, the Bureau estimates how many people there are each year.
- Article 1, Section 2 of the Constitution:
  - “Representatives and direct Taxes shall be apportioned among the several States ... according to their respective Numbers ...”

# Census Table Description

---

- Values have column-dependent interpretations
  - The SEX column: 1 is *Male*, 2 is *Female*
  - The POPESTIMATE2010 column: *7/1/2010 estimate*
- In this table, some rows are sums of other rows
  - The SEX column: 0 is *Total (of Male + Female)*
  - The AGE column: 999 is *Total of all ages*
- Numeric codes are often used for storage efficiency
- Values in a column have the same type, but are not necessarily comparable (AGE 12 vs AGE 999)

# Analyzing Census Data

---

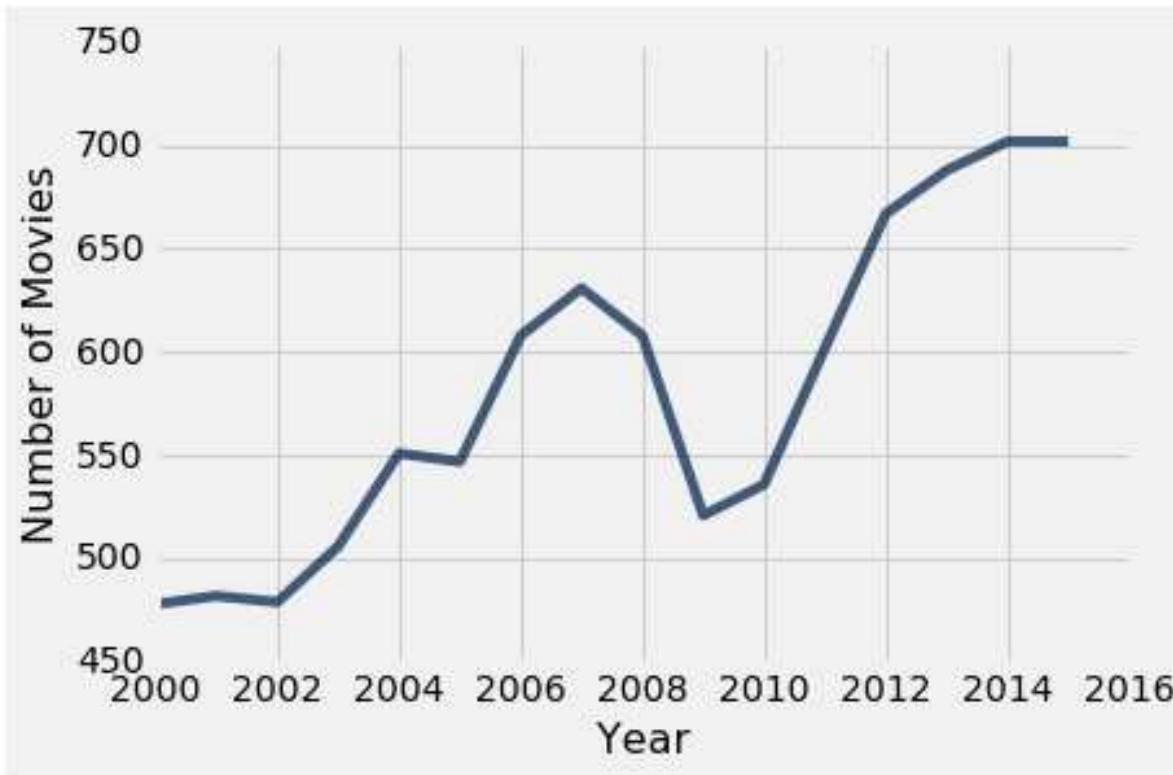
Leads to the discovery of interesting features and trends in the population

(Demo)

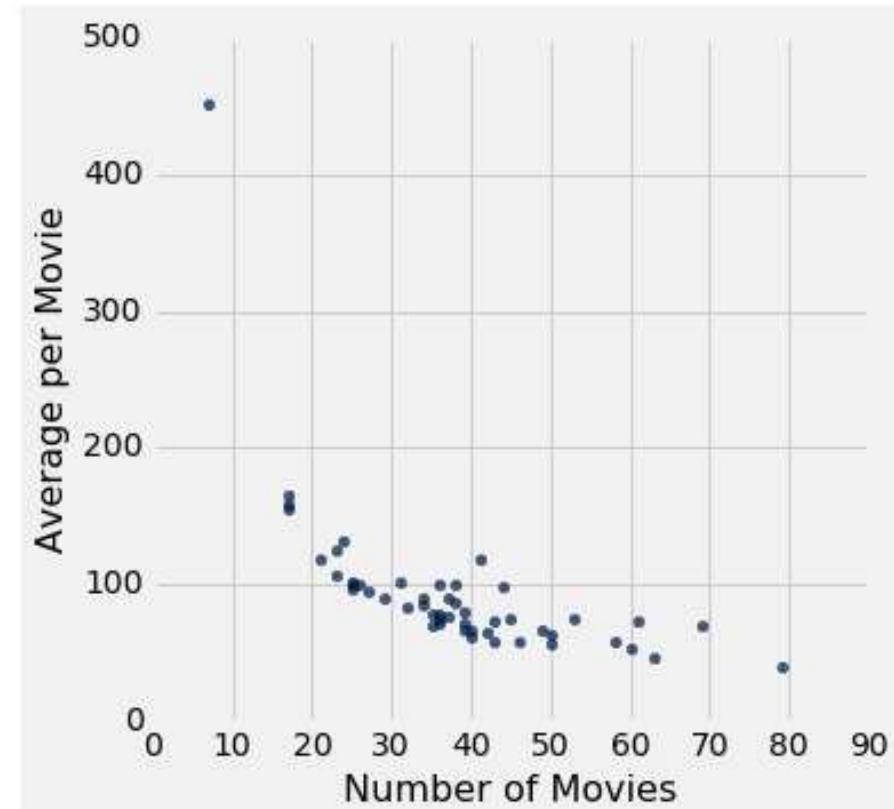
# Numerical Data

# Plotting Two Numerical Variables

Line graph: `plot`



Scatter plot : `scatter`



(Demo)

# When to use a line vs scatter plot?

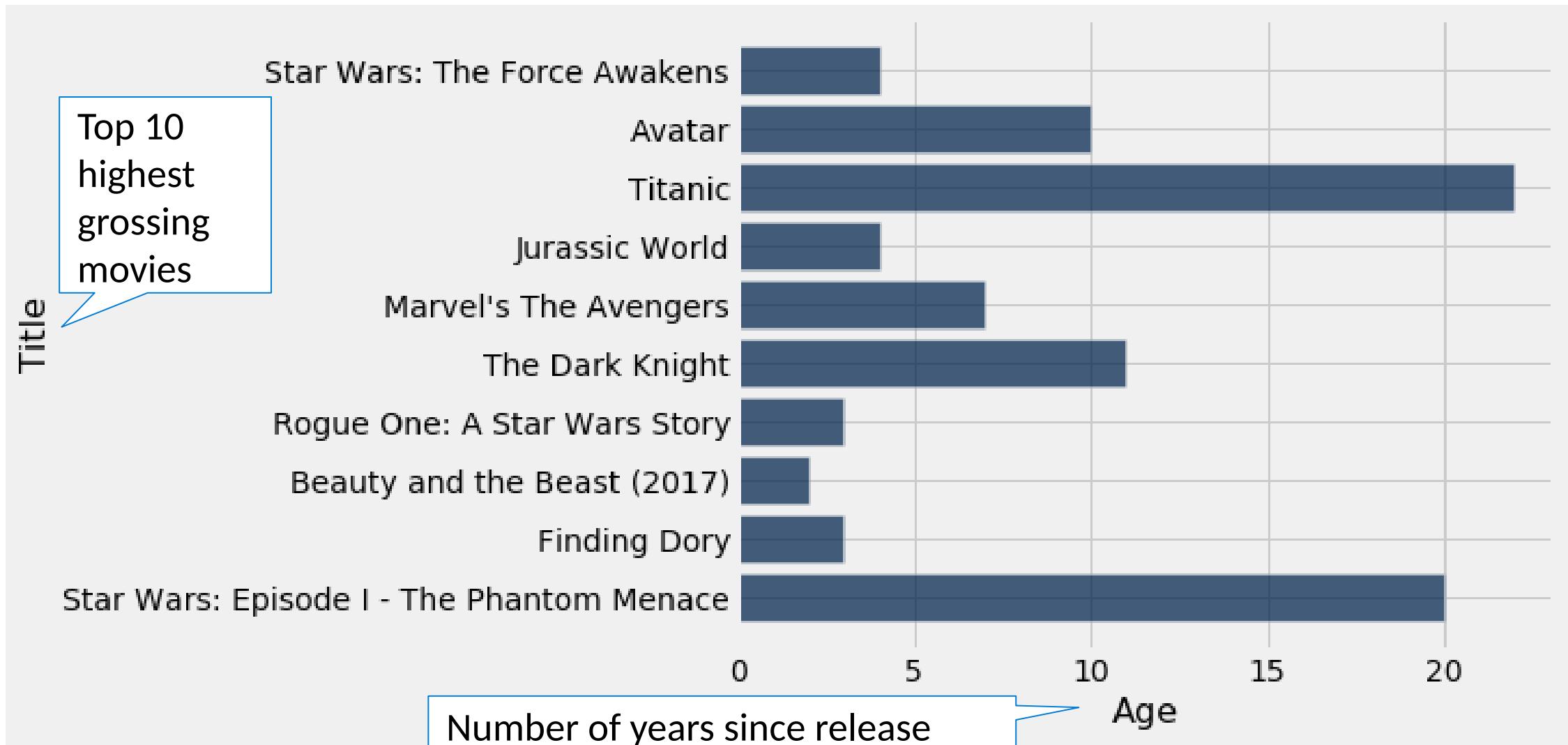
---

- Use line plots for sequential data: if...
  - ...your x-axis has an order
  - ...sequential differences in y values are meaningful
  - ...there's only one y-value for each x-value
  - Usually: x-axis is **time** or **distance**
- Use scatter plots for non-sequential data
  - When you're looking for associations

# Categorical Data

(Demo)

# How Do You Generate This Chart?



# Terminology

---

- **Individuals:** those whose features are recorded
- **Variable:** a feature, an attribute
- A variable has different **values**
- Values can be **numerical or categorical**, and of many sub-types within these
- Each **individual has exactly one value** of the variable
  
- **Distribution:** For each different value of the variable, the frequency of individuals that have that value

# Categorical Distributions

# Visualization

---

- Bar charts are commonly used to visualize categorical distributions
- One axis is categorical, one numerical

(Demo)

---

# Displaying a Categorical Distribution

---

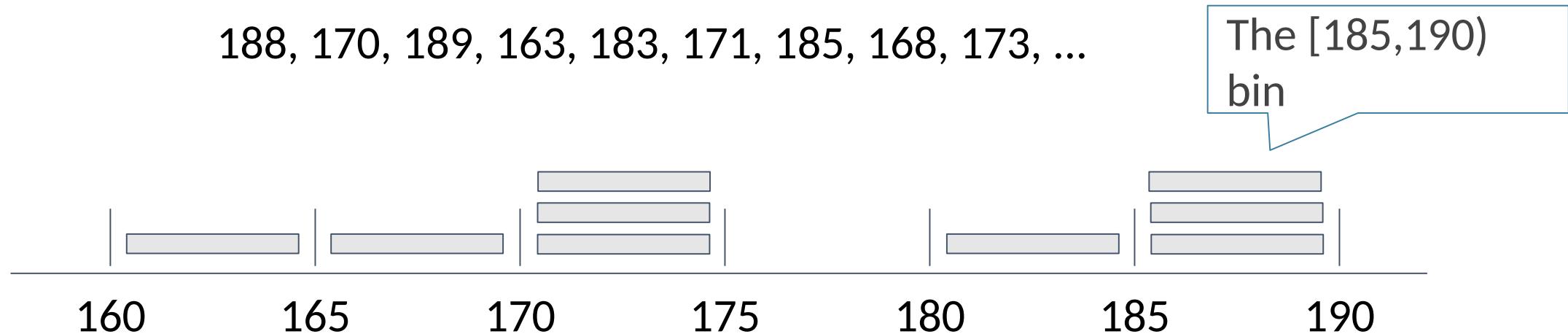
- The distribution of a variable (a column, e.g. Studios) describes the frequencies of its different values
- The **group** method counts the number of rows for each value in the column (e.g. the number of top movies released by each studio)
- Bar charts can display the distribution of a categorical variable (e.g. studios):
  - One bar for each category
  - Length of bar is the count of individuals in that category
  - You can choose the order of the bars

# Binning a Numerical Variable

# Binning Numerical Values

Binning is counting the number of numerical values that lie within ranges, called bins.

- Bins are defined by their lower bounds (inclusive)
- The upper bound is the lower bound of the next bin



# Area Principle

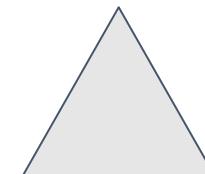
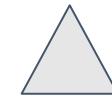
# Area Principle

---

Areas should be proportional to the values they represent.

For example

- If you represent 20% of a population by
- Then 40% can be represented by:
- But not by:



# Drawing Histograms

# Histogram

---

- Chart that displays the distribution of a numerical variable
- Uses bins; there is one bar corresponding to each bin
- Uses the area principle:
  - The *area* of each bar is the percent of individuals in the corresponding bin

(Demo)

---

# Bar Chart or Histogram?

---

To display a distribution:

## Bar Chart

- Distribution of categorical variable
- Bars have arbitrary (but equal) widths and spacings
- height (or length) of bars proportional to the percent of individuals

## Histogram

- Distribution of numerical variable
- Horizontal axis is numerical: to scale, no gaps, bins can be unequal
- Area of bars proportional to the percent of individuals; height measures density

# Summary: charts

---

- **Scatter plot**: relation between numerical variables
  - **Line graph**: sequential data (over time, etc.)
  - **Bar chart**: distribution of categorical data
  - **Histogram**: distribution of numerical data
-

# To do

- Assignment 1 due on Friday
- No Lecture on next Monday (due to CNY)
- No Lab next week (due to CNY)
- Assignment 2 will be uploaded before this weekend and due on Next Friday

**YaleNUSCollege**

**YSC2239 Lecture 10**

# Recap

- A/B testing
- Confidence Intervals
- Significant level (also called alpha level)

## Python command

- Percentile

# Today's class

- Central and Spread
  - Central limit theorem
  - Correlation
- 
- Reading: Chapter 14, 15

# Confidence Intervals For Testing

# Using a CI for Testing

---

*What if we want to do a hypothesis test, but we can't simulate under the null?*

- Null hypothesis: **Population average =  $x$**
- Alternative hypothesis: **Population average  $\neq x$**
- Cutoff for P-value:  $p\%$
- Method:
  - Construct a  $(100-p)\%$  confidence interval for the population average
  - If  $x$  is not in the interval, reject the null
  - If  $x$  is in the interval, can't reject the null

# Center and Spread

# Questions

---

- How can we quantify natural concepts like “center” and “variability”?
  - Why do many of the empirical distributions that we generate come out bell shaped?
  - How is sample size related to the accuracy of an estimate?
-

Average

# The Average (or Mean)

---

Data: 2, 3, 3, 9    **Average =  $(2+3+3+9)/4 = 4.25$**

- Need not be a value in the collection
- Need not be an integer even if the data are integers
- Somewhere between min and max, but not necessarily halfway in between
- Same units as the data
- Smoothing operator: collect all the contributions in one big pot, then split evenly

(Demo)

---

# Comparing Mean and Median

---

- **Mean:** Balance point of the histogram
- **Median:** Half-way point of data; half the area of histogram is on either side of median
- If the distribution is symmetric about a value, then that value is both the average and the median.
- If the histogram is skewed, then the mean is pulled away from the median in the direction of the tail.

# Standard Deviation

# Defining Variability

---

**Plan A:** “biggest value - smallest value”

- Doesn’t tell us much about the shape of the distribution

**Plan B:**

- Measure variability around the mean
- Need to figure out a way to quantify this

(Demo)

---

# How Far from the Average?

---

- Standard deviation (SD) measures roughly how far the data are from their average
- $SD = \text{root mean square of deviations from average}$
- SD has the same units as the data

# Why Use the SD?

---

There are two main reasons.

- **The first reason:**

No matter what the shape of the distribution,  
the bulk of the data are in the range “average  $\pm$  a few SDs”

- **The second reason:**

Coming up in the next lecture.

---

# Standard Units

# Standard Units

---

- How many SDs above average?
- $z = (\text{value} - \text{average})/\text{SD}$ 
  - Negative z: value below average
  - Positive z: value above average
  - $z = 0$ : value equal to average
- When values are in standard units: average = 0, SD = 1
- Chebyshev: At least 96% of the values of z are between -5 and 5 ( i.e.: average - 5 \* SD , average + 5 \* SD)

---

(Demo)

# Discussion Question

---

Find whole numbers  
that are close to:

(a) the average age

(b) the SD of the ages

(Demo)

Age in Years	Age in Standard Units
27	-0.0392546
33	0.992496
28	0.132704
23	-0.727088
25	-0.383171
33	0.992496
23	-0.727088
25	-0.383171
30	0.476621
27	-0.0392546

... (1164 rows omitted)

---

# The SD and the Histogram

---

- Usually, it's not easy to estimate the SD by looking at a histogram.
- But if the histogram has a bell shape, then you can.

# The SD and Bell-Shaped Curves

---

If a histogram is bell-shaped, then

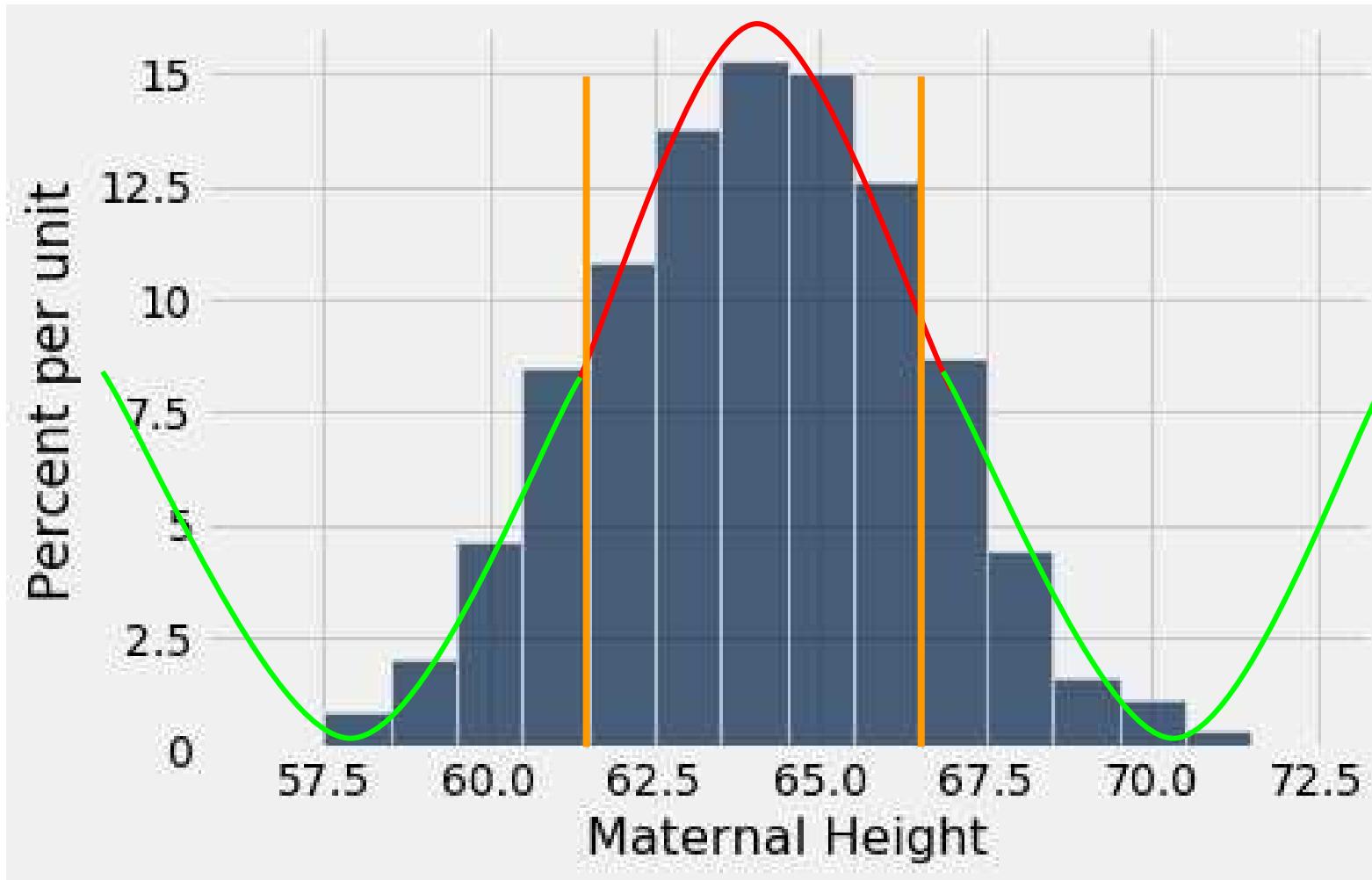
- the average is at the center
- the SD is the distance between the average and the points of inflection on either side

(Demo)

---

# Point of Inflection

---



# The Normal Distribution

# The Standard Normal Curve

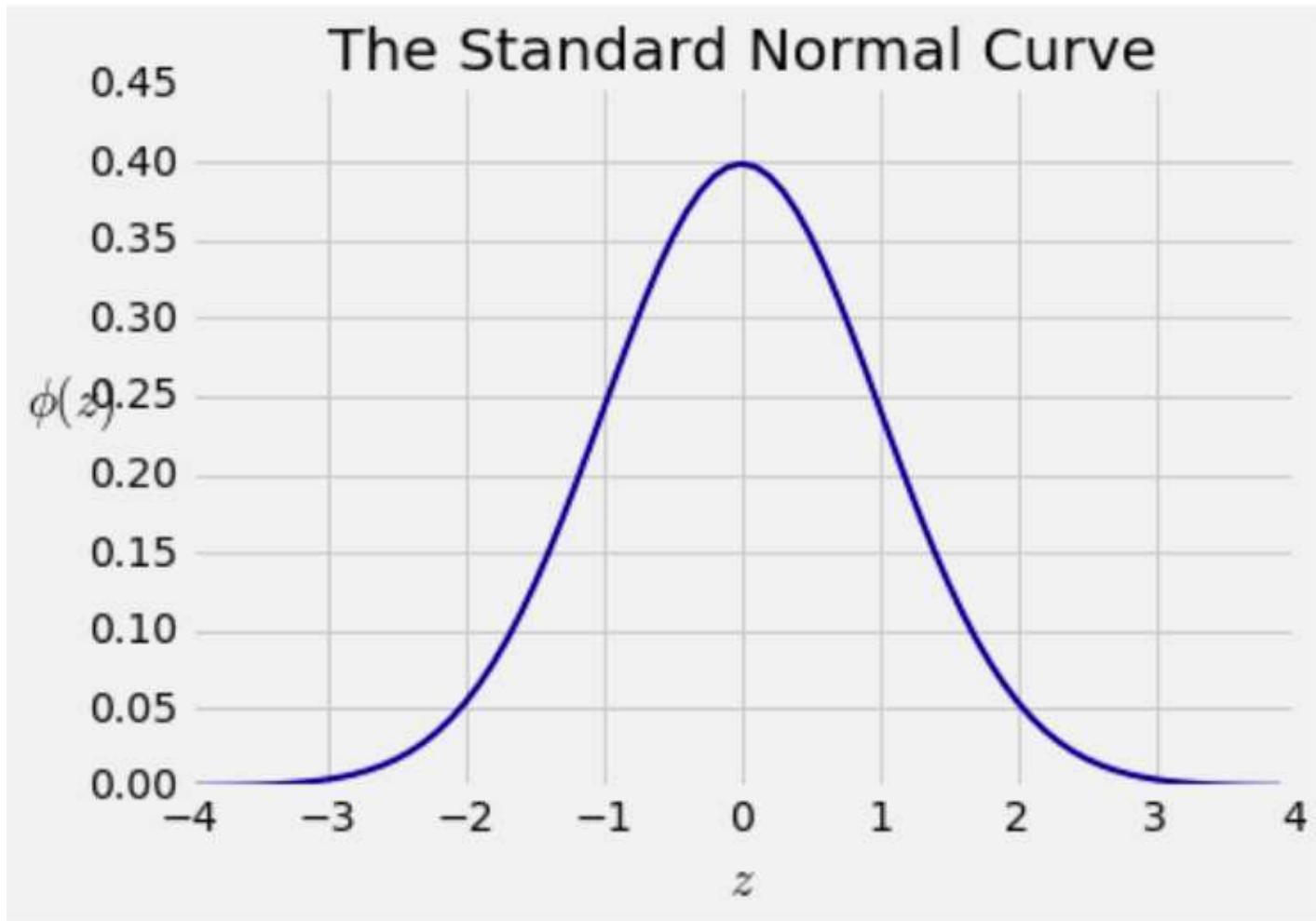
---

A beautiful formula that we won't use at all:

$$\phi(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}z^2}, \quad -\infty < z < \infty$$

# Bell Curve

---



# Normal Proportions

# How Big are Most of the Values?

---

*No matter what the shape of the distribution,*  
the bulk of the data are in the range “average  $\pm$  a few SDs”

*If a histogram is bell-shaped, then*

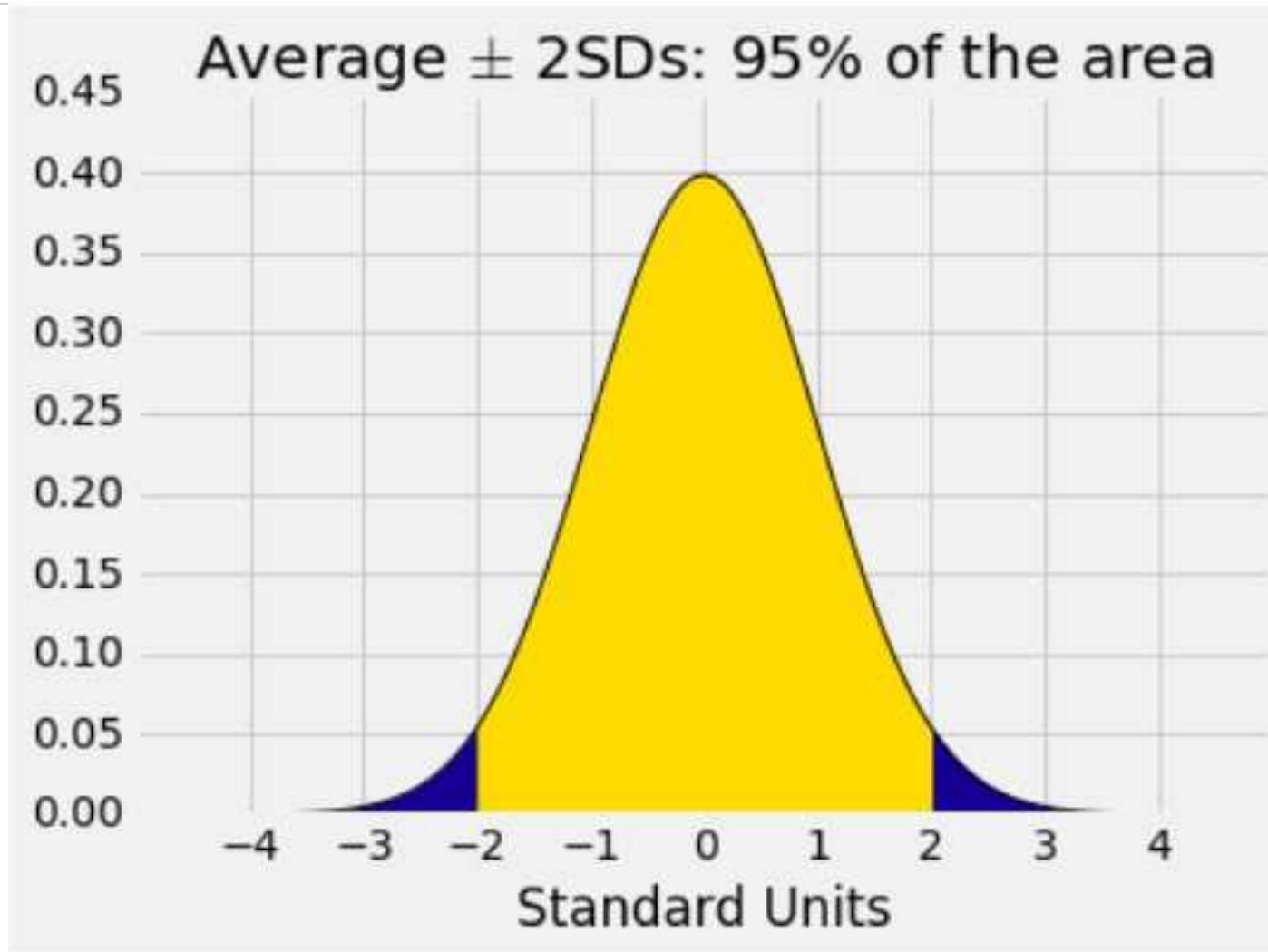
- Almost all of the data are in the range  
“average  $\pm 3$  SDs”

# Bounds and Normal Approximations

---

<b>Percent in Range</b>	<b>All Distributions</b>	<b>Normal Distribution</b>
average $\pm$ 1 SD	at least 0%	about 68%
average $\pm$ 2 SDs	at least 75%	about 95%
average $\pm$ 3 SDs	at least 88.888...%	about 99.73%

# A “Central” Area



# Central Limit Theorem

# Sample Averages

---

- The Central Limit Theorem describes how the normal distribution (a bell-shaped curve) is connected to random sample averages.
- We care about sample averages because they estimate population averages.

# Central Limit Theorem

---

If the sample is

- large, and
- drawn at random with replacement,

Then, *regardless of the distribution of the population,*

**the probability distribution of the sample sum  
(or the sample average) is roughly normal**

(Demo)

---

# Distribution of the Sample Average

# Why is There a Distribution?

---

- You have only one random sample, and it has only one average.
- But **the sample could have come out differently.**
- And then the sample average might have been different.
- So there are many possible sample averages.

# Distribution of the Sample Average

---

- Imagine all possible random samples of the same size as yours. There are lots of them.
- Each of these samples has an average.
- The **distribution of the sample average** is the distribution of the averages of all the possible samples.

(Demo)

---

# Specifying the Distribution

---

Suppose the random sample is large.

- We have seen that the distribution of the sample average is roughly bell shaped.
- Important questions remain:
  - Where is the center of that bell curve?
  - How wide is that bell curve?

# Center of the Distribution

# The Population Average

---

The distribution of the sample average is roughly a bell curve centered at the population average.

# Variability of the Sample Average

# Why Is This Important?

---

- Along with the center, the spread helps identify exactly which normal curve is the distribution of the sample average.
- The variability of the sample average helps us measure how accurate the sample average is as an estimate of the population average.
- If we want a specified level of accuracy, understanding the variability of the sample average helps us work out how large our sample has to be.

(Demo)

---

# Variability of the Sample Average

---

- The distribution of all possible sample averages of a given size is called the *distribution of the sample average*.
- We approximate it by an empirical distribution.
- By the CLT, it's roughly normal:
  - Center = the population average
  - SD = (population SD) / sample size

(Demo)

---

# Discussion Question

---

A city has 500,000 households. The annual incomes of these households have an average of \$65,000 and an SD of \$45,000. The distribution of the incomes [pick one and explain]:

- (a) is roughly normal because the number of households is large.
- (b) is not close to normal.
- (c) may be close to normal, or not; we can't tell from the information given.

# Correlation Coefficient

# Definition of $r$

---

**Correlation Coefficient ( $r$ ) =**

average of

product of

x in  
standard  
units

and

y in  
standard  
units

---

Measures how clustered the scatter is around a straight line

# The Correlation Coefficient $r$

---

- Measures **linear** association
- Based on standard units
- $-1 \leq r \leq 1$ 
  - $r = 1$ : scatter is perfect straight line sloping up
  - $r = -1$ : scatter is perfect straight line sloping down
- $r = 0$ : No linear association; *uncorrelated*

(Demo)

---

# Watch Out For ...

---

- Nonlinearity
- Outliers
- Correlation does not imply causations (  
<https://www.tylervigen.com/spurious-correlations>)

# To-do

- Lab 5
- Assignment 5

**YaleNUSCollege**

**YSC2239 Lecture 11**

# Recap

- Center and Spread
- Central Limit Theorem (CLT)

Python command: `np.std, np.average`

# Today's class

- Linear regression
  - Method of least squares
  - Residuals
- 
- Reading: Chapter 15

# Linear Regression

# Linear Regression

A statement about x and y pairs

- Measured in *standard units*
- Describing the deviation of x from 0 (the average of x's)
- And the deviation of y from 0 (the average of y's)

*On average*, y deviates from 0 less than x deviates from 0

Regression  
Line

$$y_{(\text{su})} = r \times x_{(\text{su})}$$

Correlation

Not true for all points — a statement about averages

# Slope & Intercept

# Regression Line Equation

In original units, the regression line has this equation:

$$\frac{\text{estimate of } y - \text{average of } y}{\text{SD of } y} = r \times \frac{\text{the given } x - \text{average of } x}{\text{SD of } x}$$

estimated  $y$  in standard units

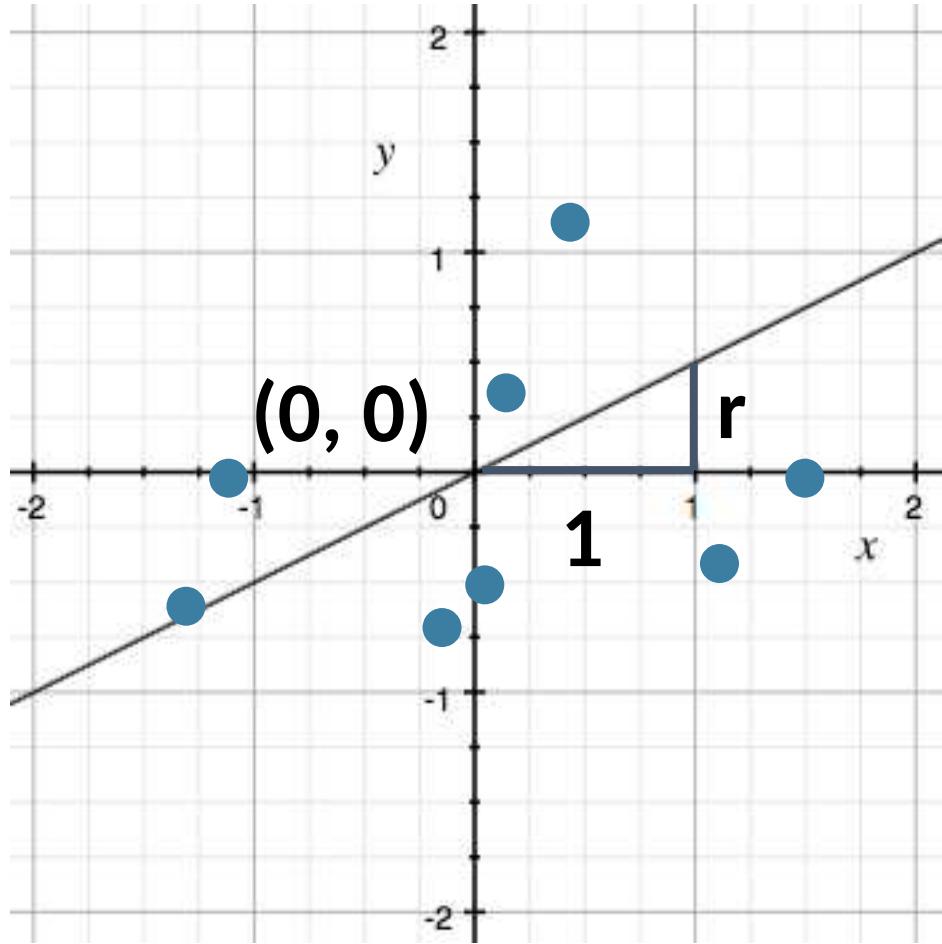
$x$  in standard units

Lines can be expressed by *slope & intercept*

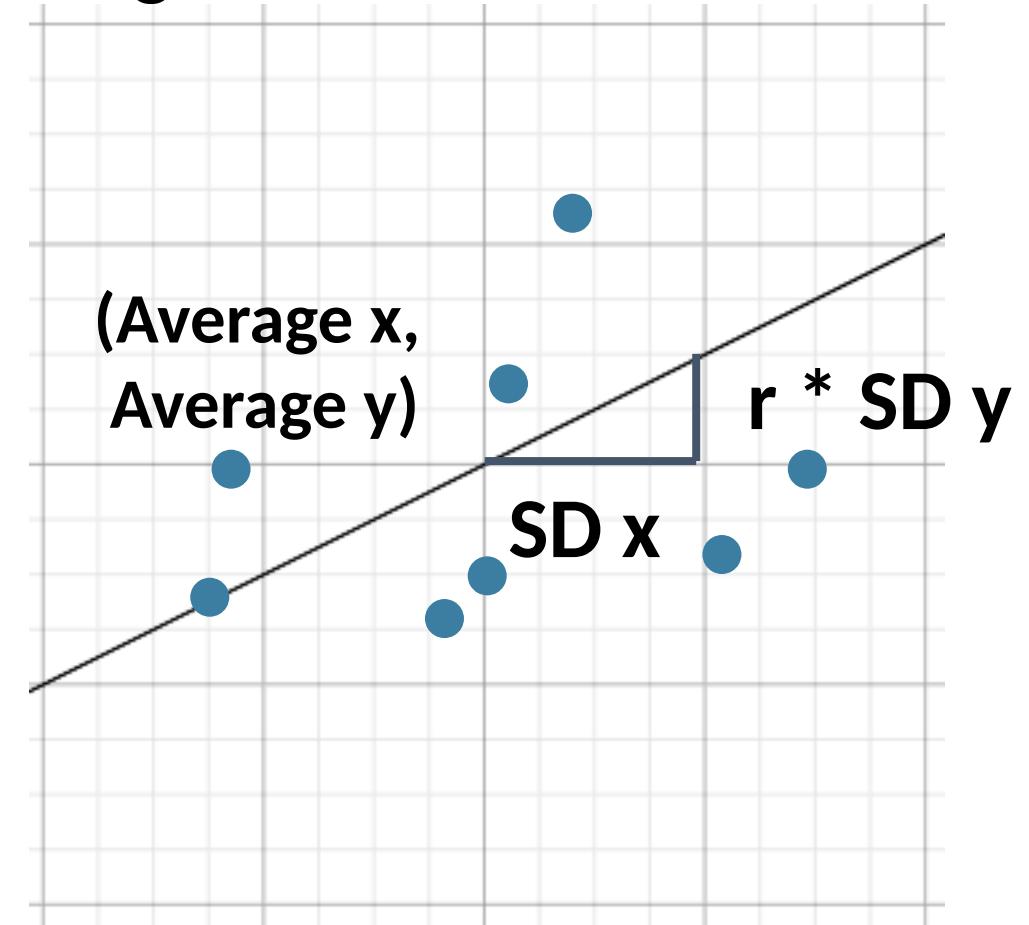
$$y = \text{slope} \times x + \text{intercept}$$

# Regression Line

Standard Units



Original Units



# Slope and Intercept

---

estimate of  $y = \text{slope} * x + \text{intercept}$

$$\text{slope of the regression line} = r \cdot \frac{\text{SD of } y}{\text{SD of } x}$$

$$\text{intercept of the regression line} = \text{average of } y - \text{slope} \cdot \text{average of } x$$

(Demo)

---

# Least Squares

# Error in Estimation

---

- **error = actual value – estimate**
- Typically, some errors are positive and some negative
- To measure the rough size of the errors
  - **square** the **errors** to eliminate cancellation
  - take the **mean** of the squared errors
  - take the square **root** to fix the units
  - **root mean square error (rmse)**

(Demo)

---

# Least Squares Line

---

- Minimizes the root mean squared error (rmse) among all lines
- Equivalently, minimizes the mean squared error (mse) among all lines
- Names:
  - “Best fit” line
  - Least squares line
  - Regression line

(Demo)

# Numerical Optimization

---

- Numerical minimization is approximate but effective
- Lots of machine learning uses numerical minimization
- If the function `mse (a, b)` returns the mse of estimation using the line “estimate =  $ax + b$ ”,
  - then `minimize (mse)` returns array `[a0, b0]`
  - $a_0$  is the slope and  $b_0$  the intercept of the line that *minimizes* the mse among lines with arbitrary slope `a` and arbitrary intercept `b` (that is, among all lines)

---

(Demo)

# Regression Diagnostics

# Residuals

---

- Error in regression estimate
- One residual corresponding to each point  $(x, y)$
- **residual**
  - = observed  $y$  - regression estimate of  $y$
  - = observed  $y$  - height of regression line at  $x$
  - = vertical distance between the point and the best line

(Demo)

---

# Residual Plot

---

A scatter diagram of residuals

- Should look like an unassociated blob for linear relations
- But will show patterns for non-linear relations
- Used to check whether linear regression is appropriate
- Look for curves, trends, changes in spread, outliers, or any other patterns

# Properties of residuals

---

- Residuals from a linear regression **always** have
  - **Zero** mean
    - (so **rmse = SD of residuals**)
  - **Zero** correlation with x
  - **Zero** correlation with the fitted values
- These are all true **no matter what the data look like**
  - Just like deviations from mean are zero on average

(Demo)

---

# A Measure of Clustering

# Correlation, Revisited

---

- “The correlation coefficient measures how clustered the points are about a straight line.”
- We can now quantify this statement.

(Demo)

---

# SD of Fitted Values

---

- SD of fitted values

$$\frac{\text{-----}}{\text{SD of } y} = |r|$$

- SD of fitted values =  $|r| * (\text{SD of } y)$

# Variance of Fitted Values

---

- Variance = Square of the SD  
= Mean Square of the Deviations
- Variance has weird units, but good math properties
- Variance of fitted values  
$$\frac{\text{-----}}{\text{Variance of } y} = r^2$$

# A Variance Decomposition

---

By definition,

$$y = \text{fitted values} + \text{residuals}$$

Tempting (but wrong) to think that:

$$\text{SD}(y) = \text{SD}(\text{fitted values}) + \text{SD}(\text{residuals})$$

But it is true that:

$$\text{Var}(y) = \text{Var}(\text{fitted values}) + \text{Var}(\text{residuals})$$

(a result of the Pythagorean theorem!)

---

# A Variance Decomposition

---

$$\text{Var}(y) = \text{Var}(\text{fitted values}) + \text{Var}(\text{residuals})$$

- Variance of fitted values

$$\text{-----} = r^2$$

Variance of  $y$

- Variance of residuals

$$\text{-----} = 1 - r^2$$

Variance of  $y$

---

# Residual Average and SD

---

- The average of residuals is always 0

- Variance of residuals

$$\text{-----} = 1 - r^2$$

Variance of  $y$

- SD of residuals  $= \sqrt{1 - r^2}$  SD of  $y$

(Demo)

# **End of midterm coverage**

---

# To-do

- Assignment 5

**YaleNUSCollege**

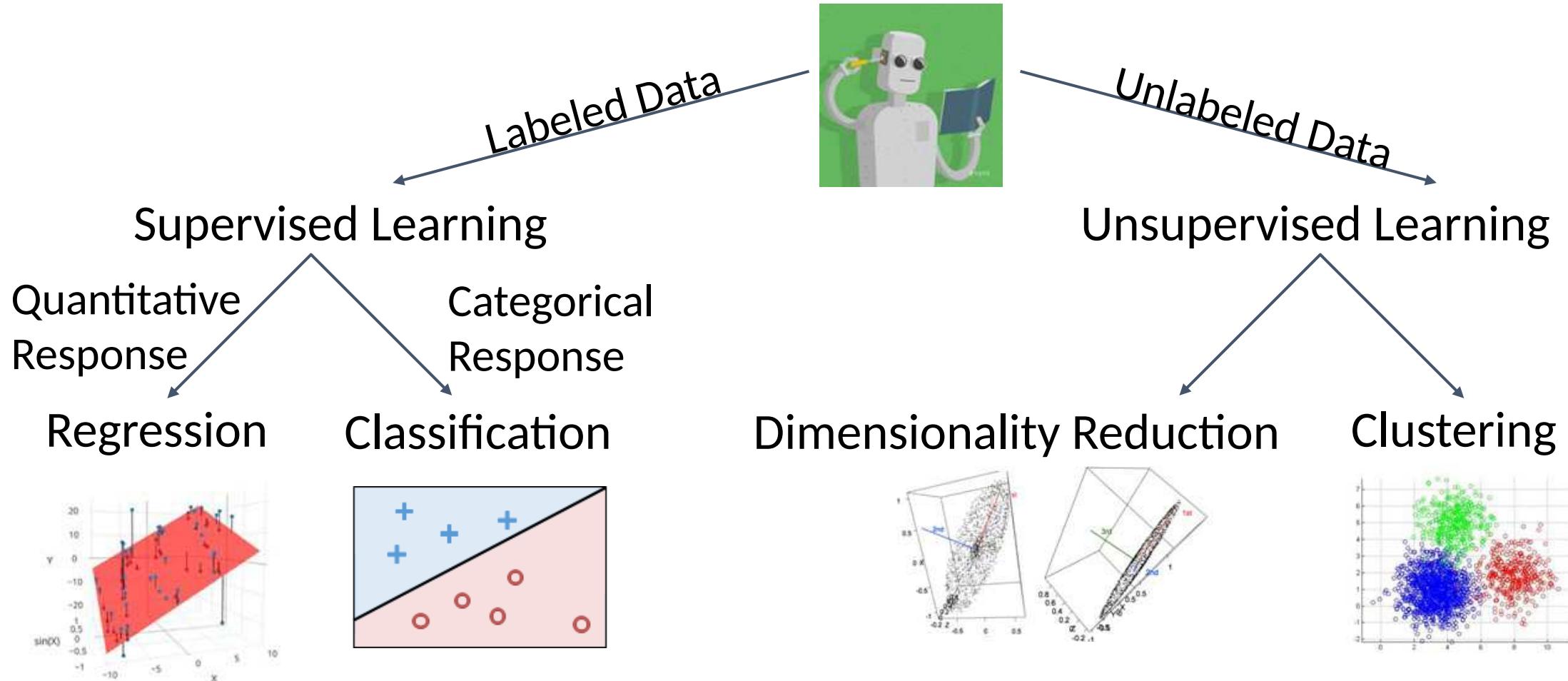
**YSC2239 Lecture 12**

# Today's class

- Classification
- Reading: Chapter 16, 17

# Prediction

# Review: Taxonomy of Machine Learning



# Guessing the Value of an Attribute

---

- Based on incomplete information
- One way of making predictions:
  - To predict an outcome for an individual,
  - find others who are like that individual
  - and whose outcomes you know.
  - Use those outcomes as the basis of your prediction.
- Two Types of Prediction
  - Classification = Categorical; Regression = Numeric

# Prediction Example: Spam or Not?

---

You made a Wells Fargo payment - wellsfargo.com You recently submitted a payment The ...

BUSINESS TRUST - -- I have a legal business proposal for you worth \$23,000,000. If you kn...

Hi - Today???!!!! What a wonderful day! Congrats again! I am definitely not doing s...

Michael Kors Handbags Up To 84% Plus Free Shipping! - Shop Handbags Online & In Store...

# Machine Learning Algorithm

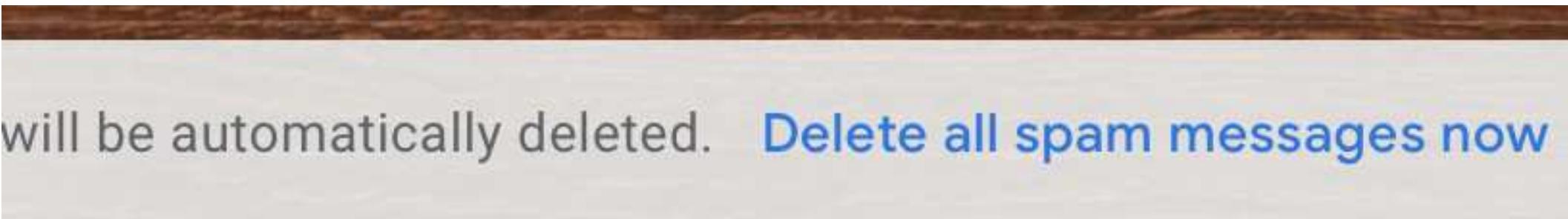
---

- A mathematical model
- calculated based on sample data ("training data")
- that makes predictions or decisions without being explicitly programmed to perform the task

# Classification

# Classification Examples

---



will be automatically deleted. [Delete all spam messages now](#)



I have a legal business proposal for you worth \$23,000,000....

# Classification Examples

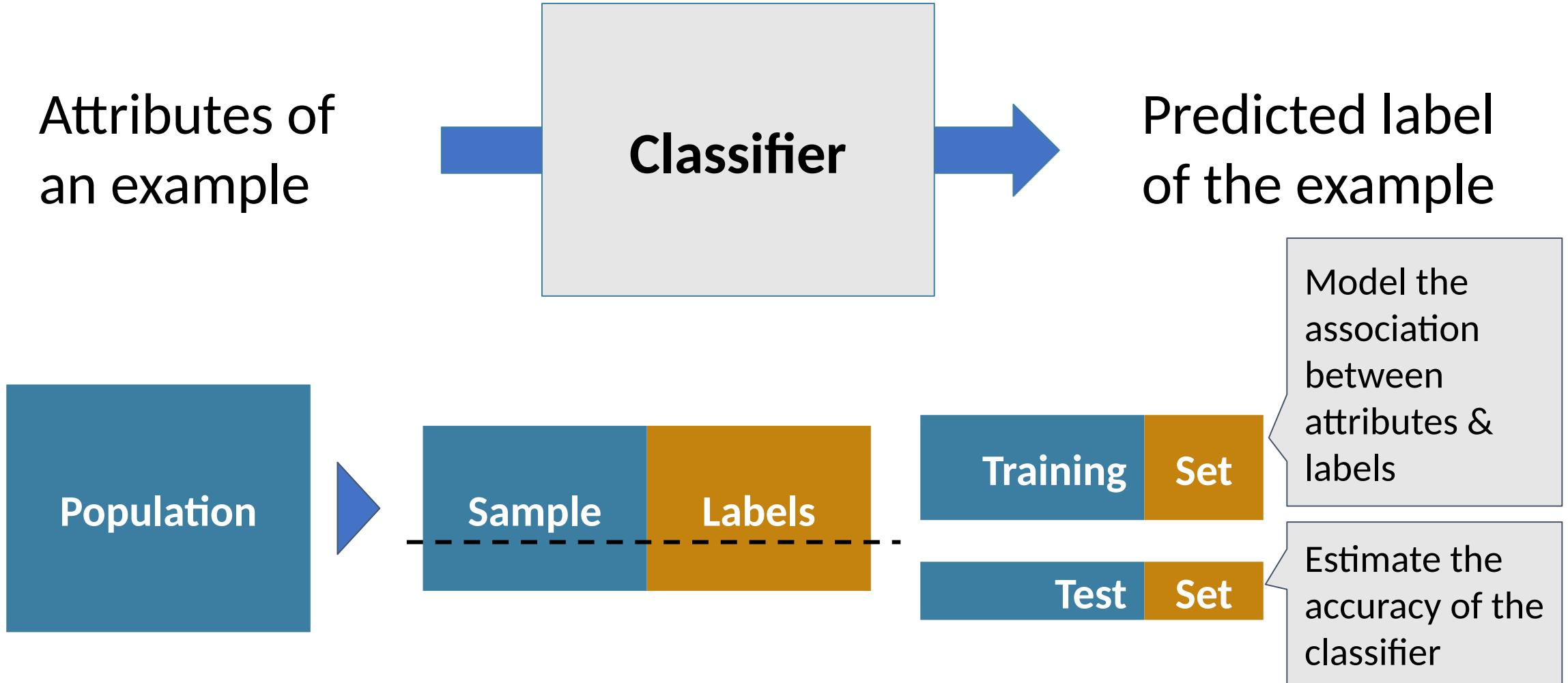
---

Top picks for you

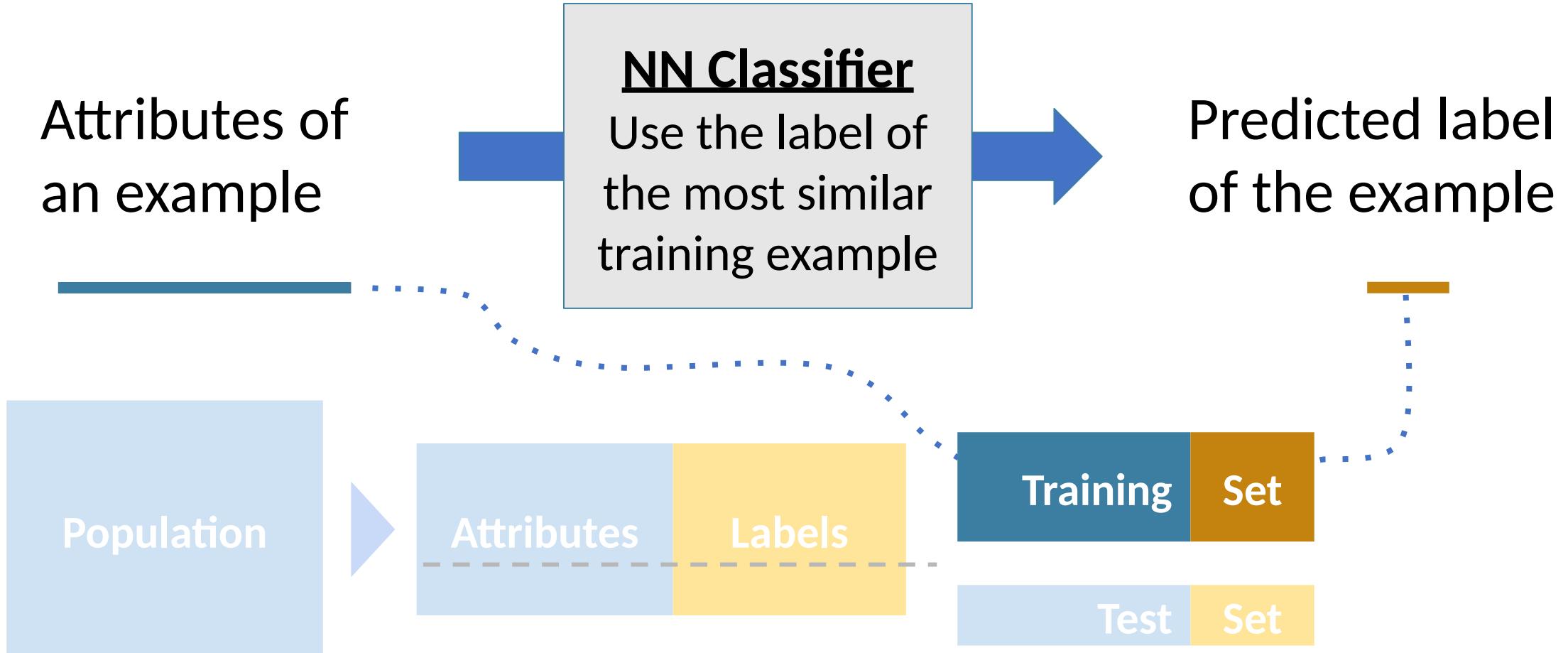


# Classifiers

# Training a Classifier

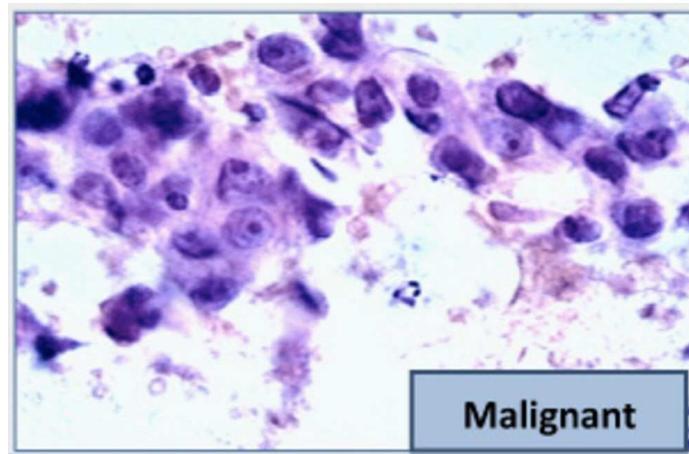
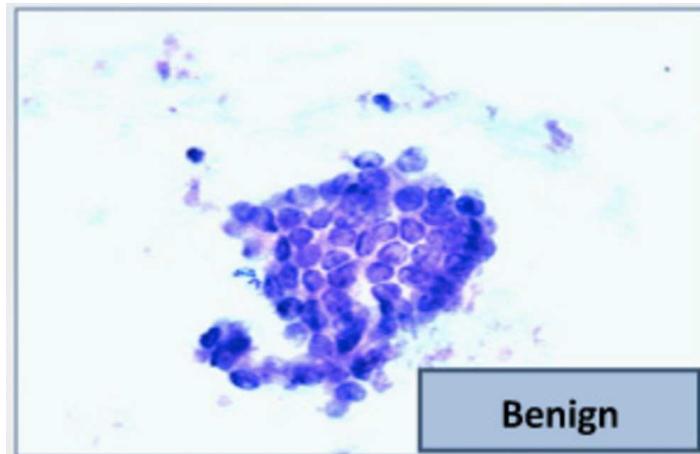


# Nearest Neighbor Classifier



# The Google Science Fair

- Brittany Wenger, a 17-year-old high school student in 2012
- Won by building a breast cancer classifier with 99% accuracy



(Demo)

# Distance

# Rows of Tables

---

Each row contains all the data for one individual

- `t.row(i)` evaluates to `i`th row of table `t`
- `t.row(i).item(j)` is the value of column `j` in row `i`
- If all values are numbers, then `np.array(t.row(i))` evaluates to an array of all the numbers in the row.
- To consider each row individually, use  
`for row in t.rows:`  
    `... row.item(j) ...`

# Distance Between Two Points

---

- Two attributes  $x$  and  $y$ :

$$D = \sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2}.$$

- Three attributes  $x$ ,  $y$ , and  $z$ :

$$D = \sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2 + (z_0 - z_1)^2}$$

- and so on ...

(Demo)

---

# Nearest Neighbors

# Finding the $k$ Nearest Neighbors

---

To find the  $k$  nearest neighbors of an example:

- Find the distance between the example and each example in the training set
- Augment the training data table with a column containing all the distances
- Sort the augmented table in increasing order of the distances
- Take the top  $k$  rows of the sorted table

(Demo)

---

# The Classifier

---

To classify a point:

- Find its  $k$  nearest neighbors
- Take a majority vote of the  $k$  nearest neighbors to see which of the two classes appears more often
- Assign the point the class that wins the majority vote

(Demo)

---

# Evaluation

# Accuracy of a Classifier

---

The accuracy of a classifier on a labeled data set is the proportion of examples that are labeled correctly

Need to compare classifier predictions to true labels

If the labeled data set is sampled at random from a population, then we can infer accuracy on that population



**YaleNUSCollege**

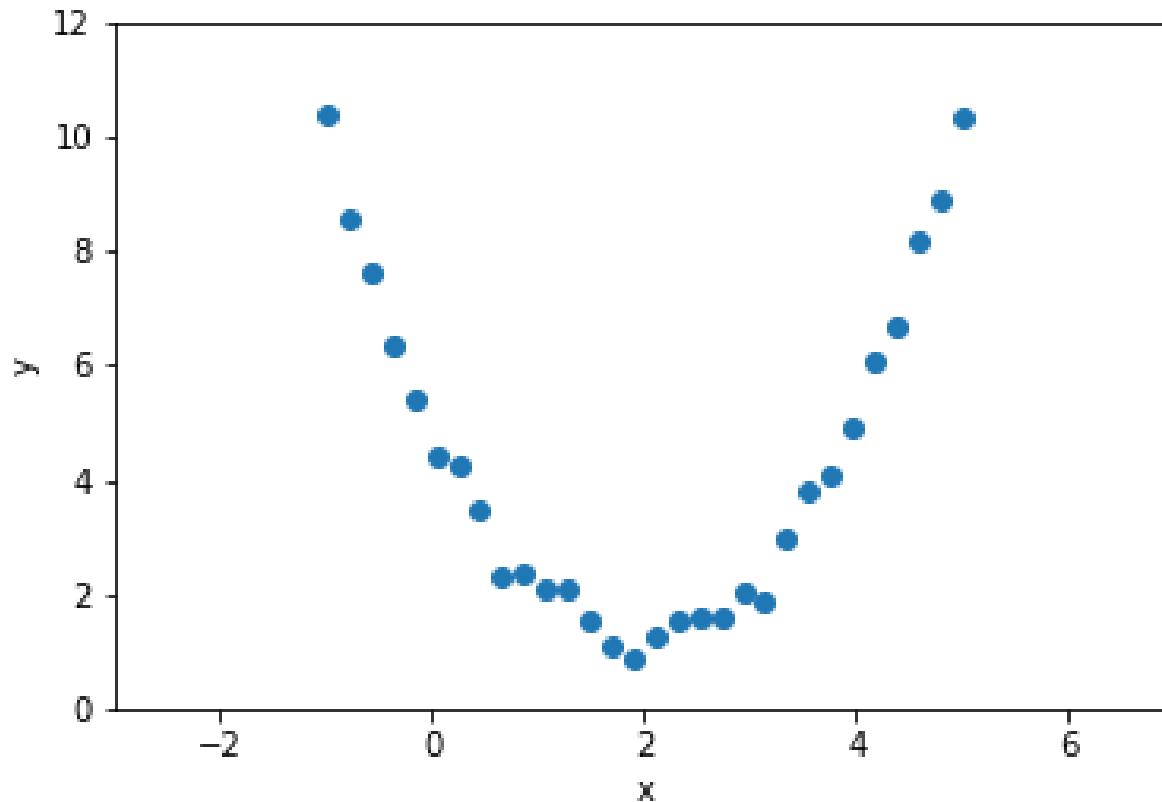
**YSC2239 Lecture 16**

# Today's class

- Introduction to scikit-learn
- Feature engineering

# Motivating Feature Engineering

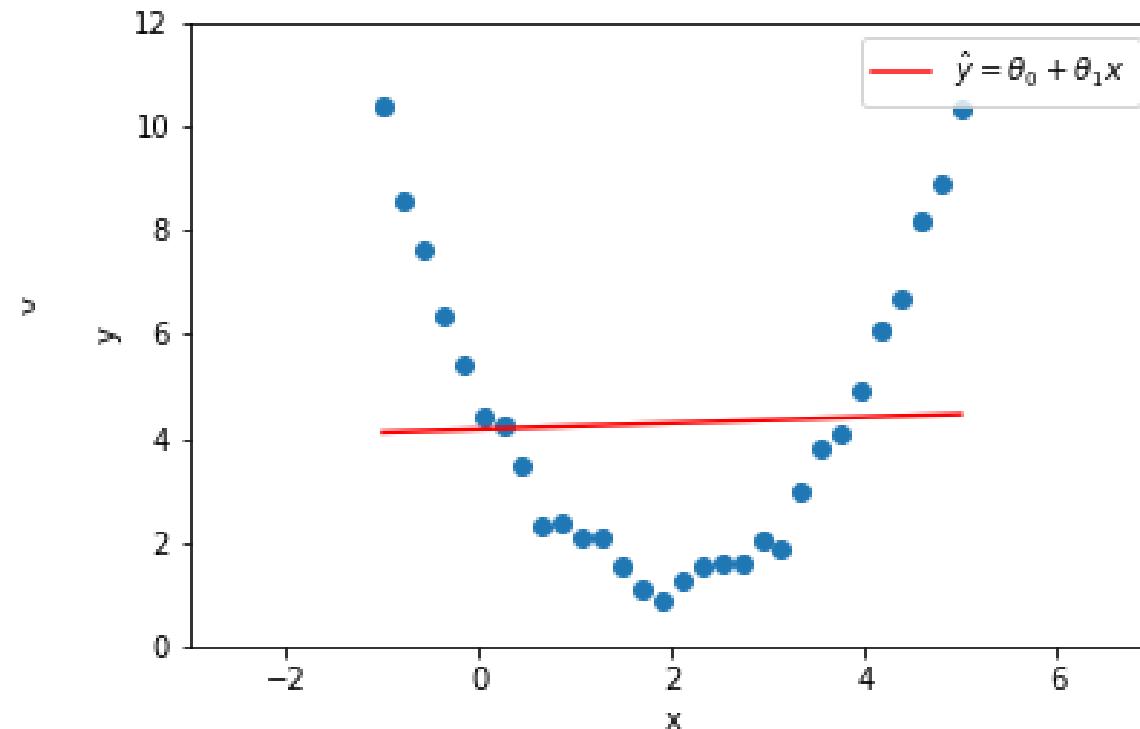
# Can we fit a model to this data?



# Can we fit a model to this data?

Simple Linear Regression?

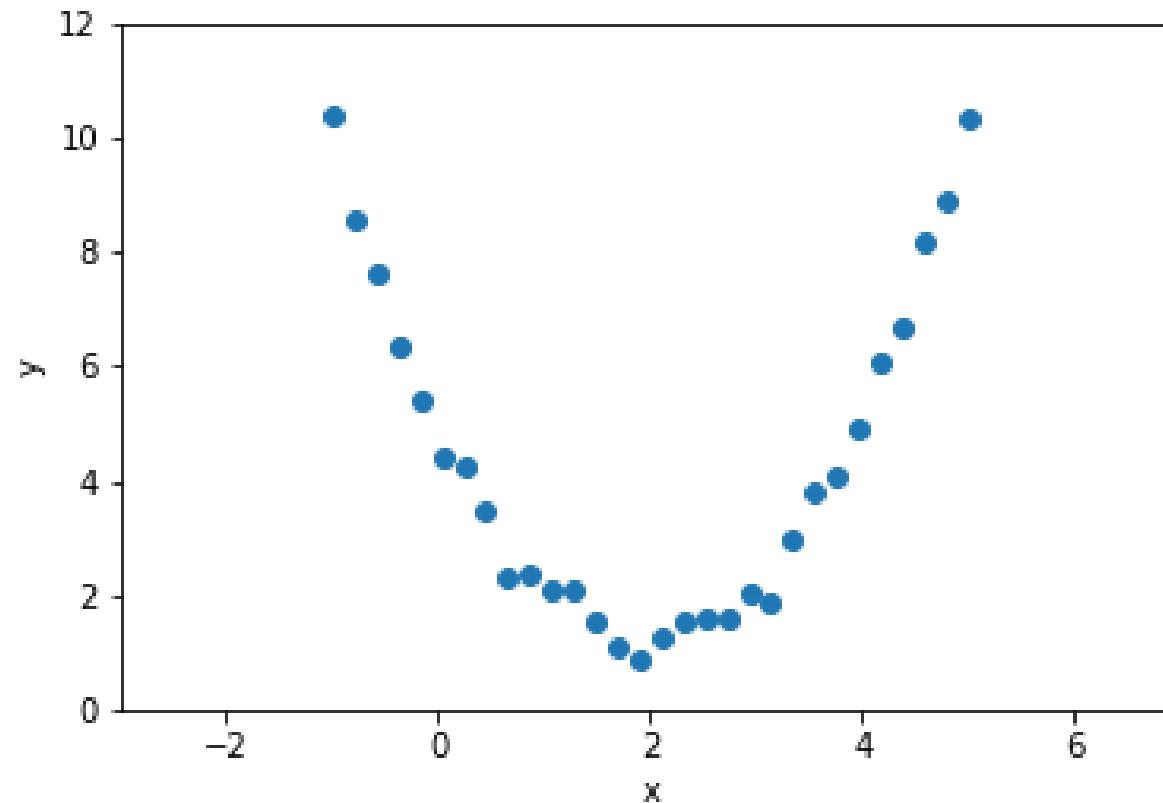
No, because the data is fundamentally nonlinear



# Can we fit a model to this data?

Multiple Linear Regression?

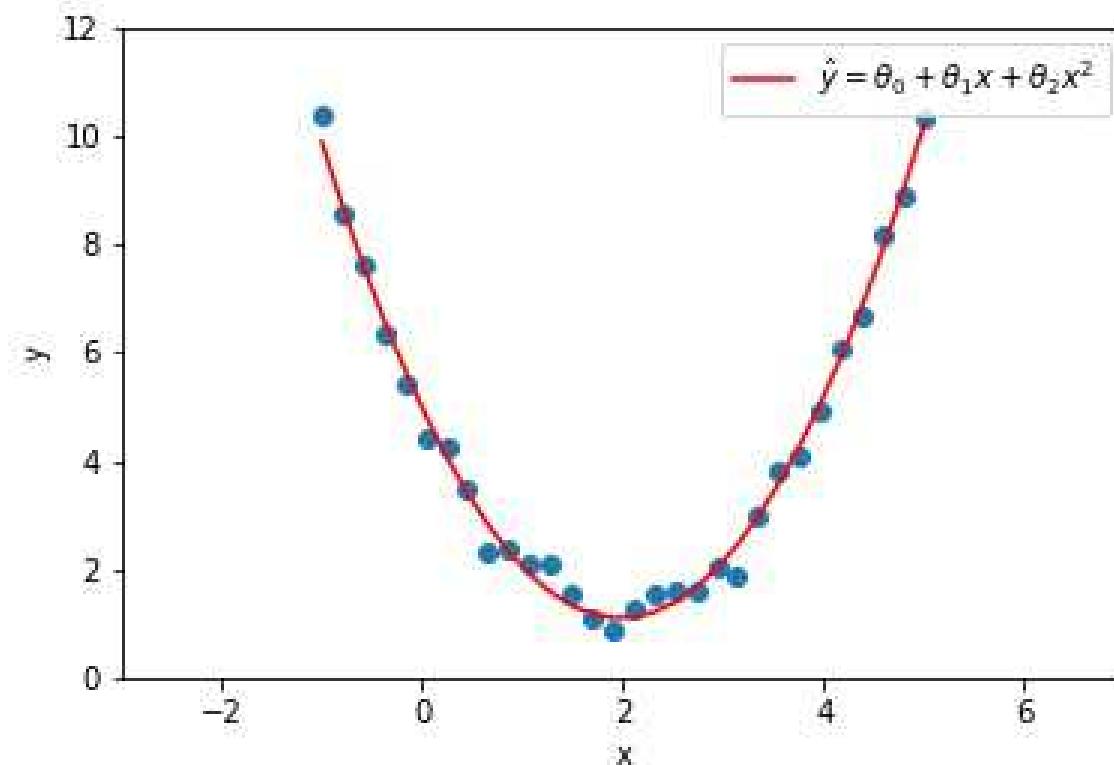
No, because there are no other features to add



# Can we fit a model to this data?

**Idea:** Create an extra feature to use in the model. What feature should we add?

Since the data looks like a parabola, let's add a quadratic feature



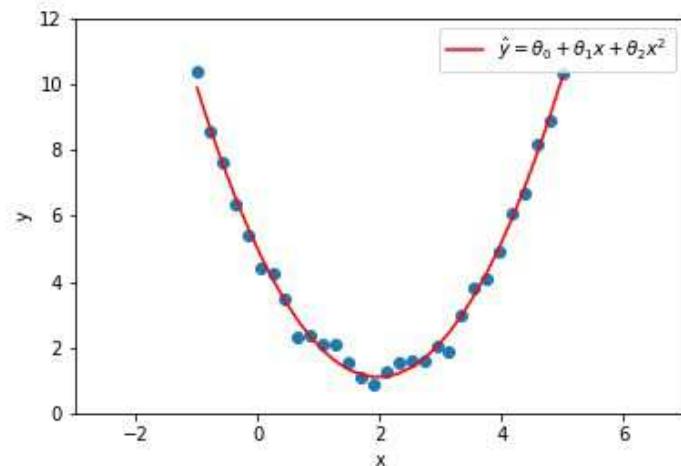
# What is a feature?

A feature is an input to our model

- So far, we have just used the raw data as features

We can also create new features to use as inputs to our model

- The process of creating new features is called **feature engineering**



model

$$\theta_0 + \theta_1x + \theta_2x^2$$

Features:  $x, x^2$

Parameters:  $\theta_0, \theta_1, \theta_2$

# Can we really just create our own features?

**Yes!** (with some restrictions)

We can create any feature we want as long we can write the model in the form

$$\hat{y} = x^T \theta$$

This is a **linear combination of the features**. The features cannot depend on the parameters of the model!

**Exam**

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 = \begin{bmatrix} 1 \\ x_1 \\ x_1^2 \end{bmatrix} \cdot \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix} = x^T \theta$$

# Lecture Roadmap

We can choose/create **x** any way we like as long as our model follows the form

$$\hat{y} = x^T \theta$$

The rest of the lecture will discuss different techniques we can use to create **x**:

- How can we create features from **quantitative data**?
- How can we create features from **categorical data**?
- How can we create features from **text data**?

# Intro to Scikit-Learn (Demo)

# Feature Engineering: Quantitative Data

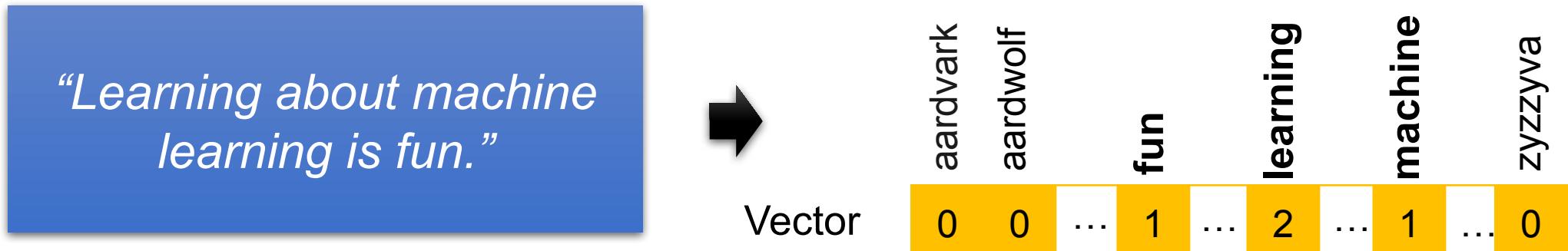
# Feature Engineering: Categorical Data

# Feature Engineering: Imputation

# Feature Engineering: Text Data

# Bag-of-words Encoding

- Generalization of one-hot-encoding for a string of text:

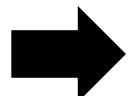


- Encode text as a long vector of word counts (Issues?)
  - Typically high dimensional (millions of columns) and very sparse
  - Word order information is lost... (is this an issue?)
  - What happens when you see a word not in the dictionary?
- A **bag** is another term for a multiset: *an unordered collection which may contain multiple instances of each element.*
- Stop words:** words that do not contain significant information
  - Examples: the, in, at, or, on, a, an, and ...
  - Typically removed

# N-Gram Encoding

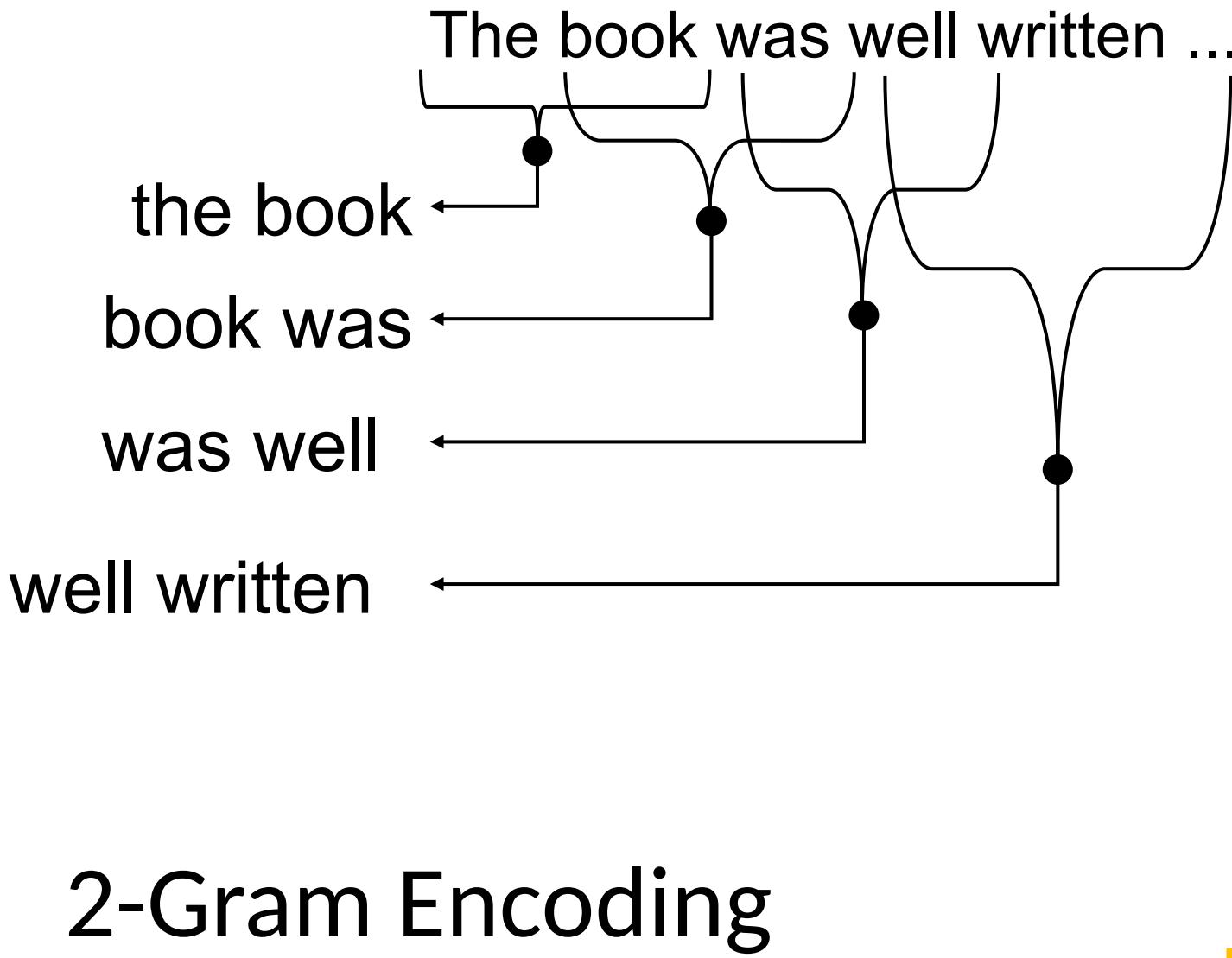
- Sometimes word order matters:

*The book was **not** well  
written but I did enjoy it.*

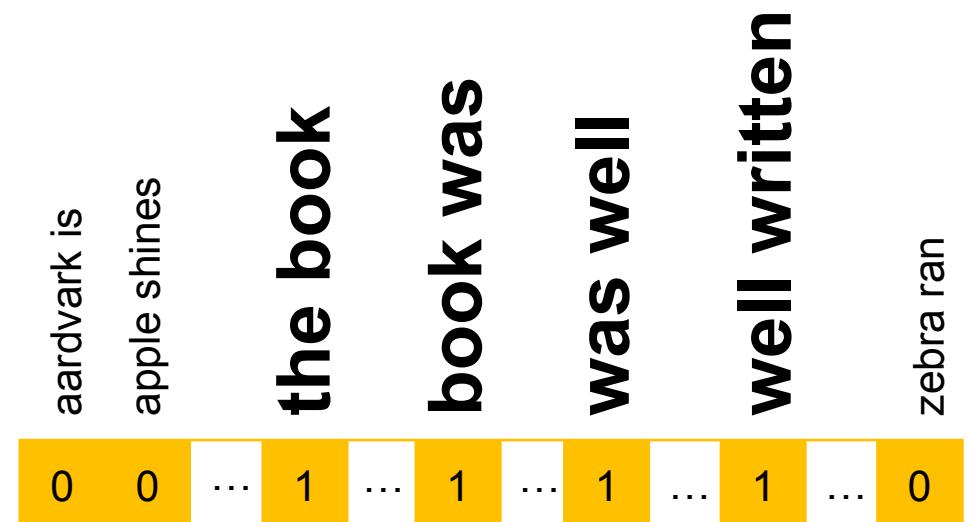


*The book was well written  
but I did **not** enjoy it.*

- How do we capture word order in a “vector” model?
  - N-Gram: “*Bag-of- sequences-of-words*”



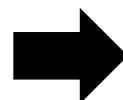
Vector



# N-Gram Encoding

- Sometimes word order matters:

*The book was **not** well  
written but I did enjoy it.*



*The book was well written  
but I did **not** enjoy it.*

- How do we capture word order in a “vector” model?
  - N-Gram: “*Bag-of- sequences-of-words*”
- Issue:
  - Can be very sparse (many combinations occur only once)

# Mathematical Implications of Feature Engineering

# One-Hot Encoding and Linear Dependence

bias	...	co
1	...	Apple
1	...	Samsung
1	...	Apple



bias	...	co_appl	co_sam
1	...	1	0
1	...	0	1
1	...	1	0

Notice that  $\text{co\_appl} + \text{co\_sam} = \text{bias}$ ! This means the columns are linearly dependent

- **Solution:** Drop one of the one-hot encoded columns per variable

# Too Many Features

If you add too many features, the normal equations will have infinite solutions

The normal equations can be thought of as a system of equations with  $N$  equations and  $P$  unknown quantities to solve for

- $N$ : # of data points
- $P$ : # of parameters

If  $P > N$ , you have more unknowns than equations so there can be no unique solution

Additionally, too many features can cause overfitting, which will be covered in future lectures

# Summary

# Feature Engineering Summary

- Feature engineering is the process of creating new useful features from your data to build more sophisticated models
- Feature engineering allows you to utilize non-numerical data
  - One-hot encoding is a widely used technique
- Need to be careful in choosing how many and which features to create
  - Linearly dependent features
  - Too many features
- Feature engineering is as much an art as it is a skill
  - Neural networks try to automatically do feature engineering

**YaleNUSCollege**

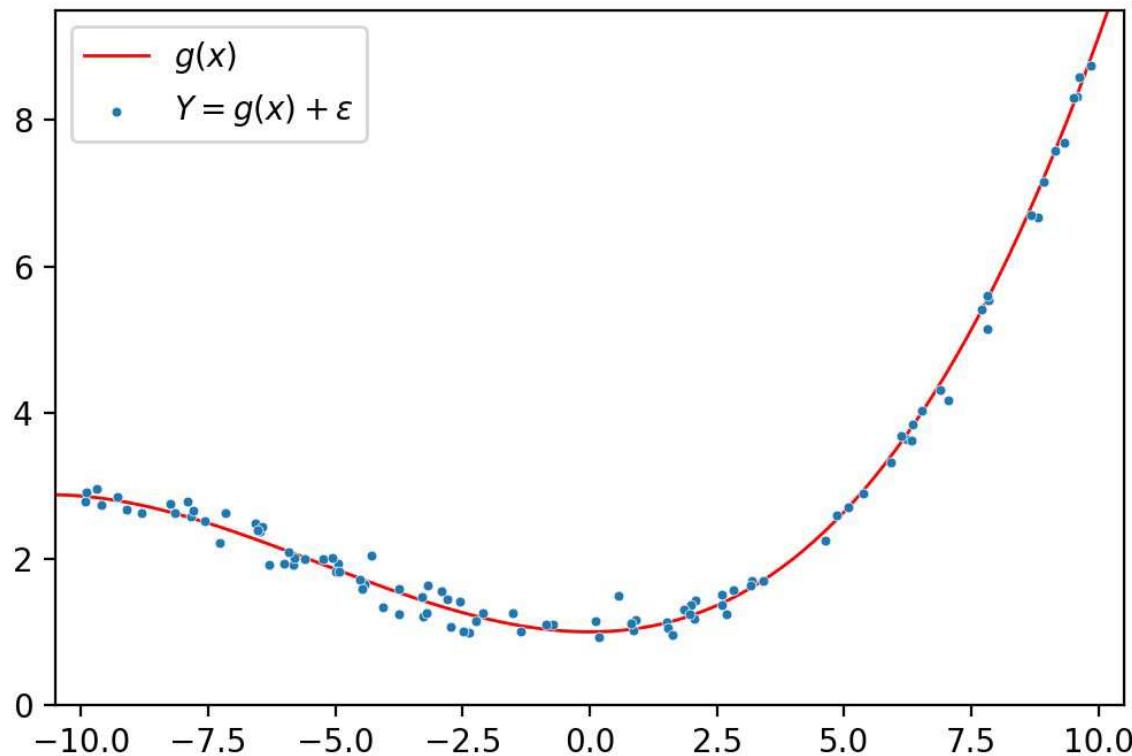
**YSC2239 Lecture 17**

# Today's class

- Variance-bias tradeoff
- Overfitting
- Cross-validation

# Data Generation Process

# Data Generation Process

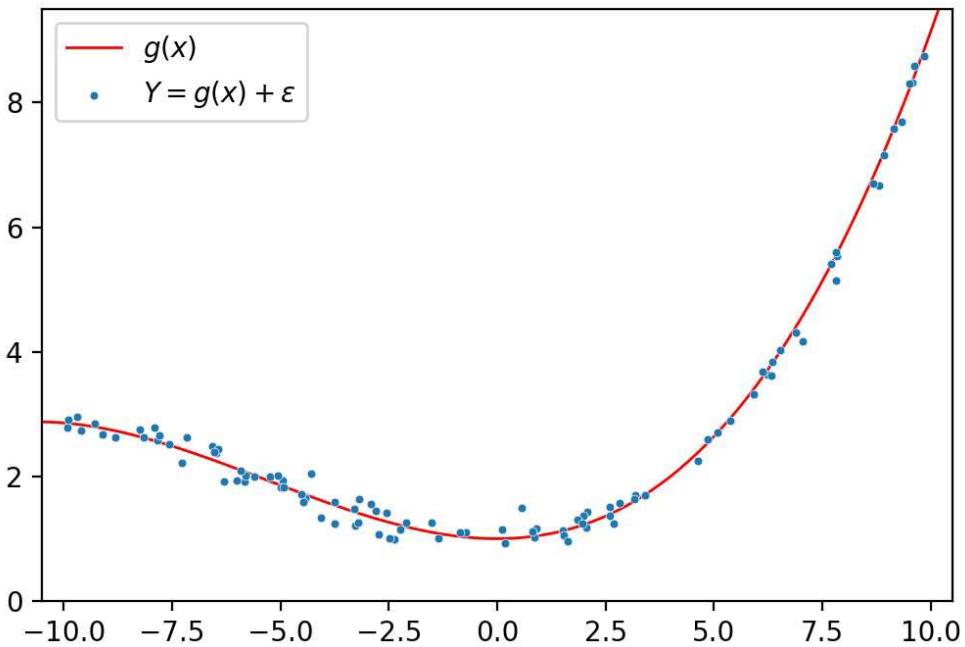


- **Assume** true relation  $g$
- For example:  $g(x) = \theta_0 + \theta_1 x$
- For each individual:
  - fixed value of  $x$ , so also  $g(x)$
  - random error  $\epsilon$
  - Observation is:  $Y = g(x) + \epsilon$

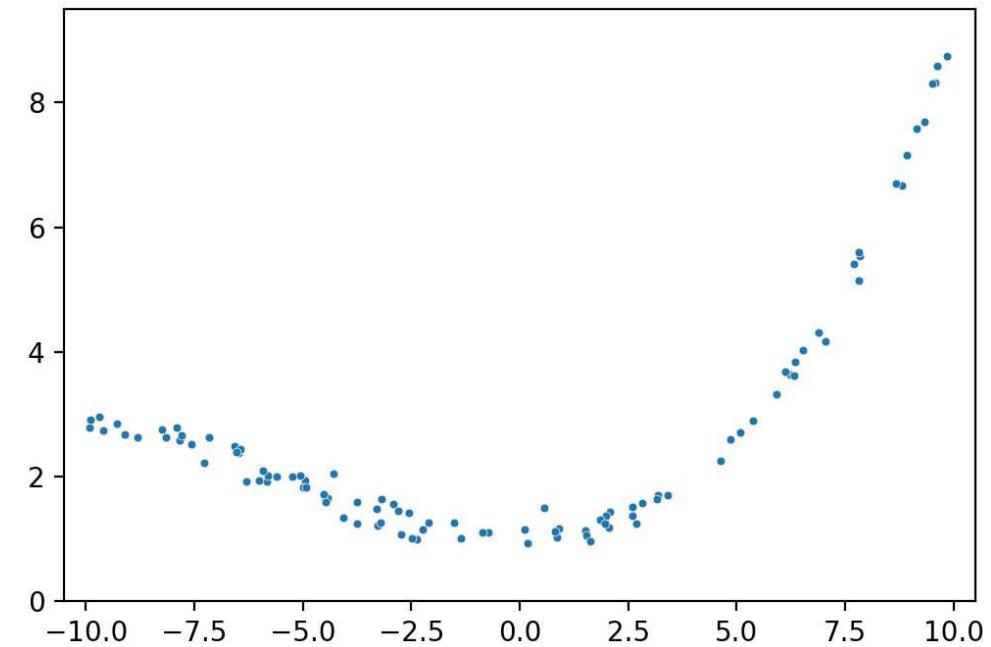
Errors  $\epsilon$  have expectation 0, and are “independent and identically distributed” across individuals

# The Data

- At each  $x$ , truth is  $g(x)$
- noise is  $\epsilon$
- Observation is  $Y = g(x) + \epsilon$

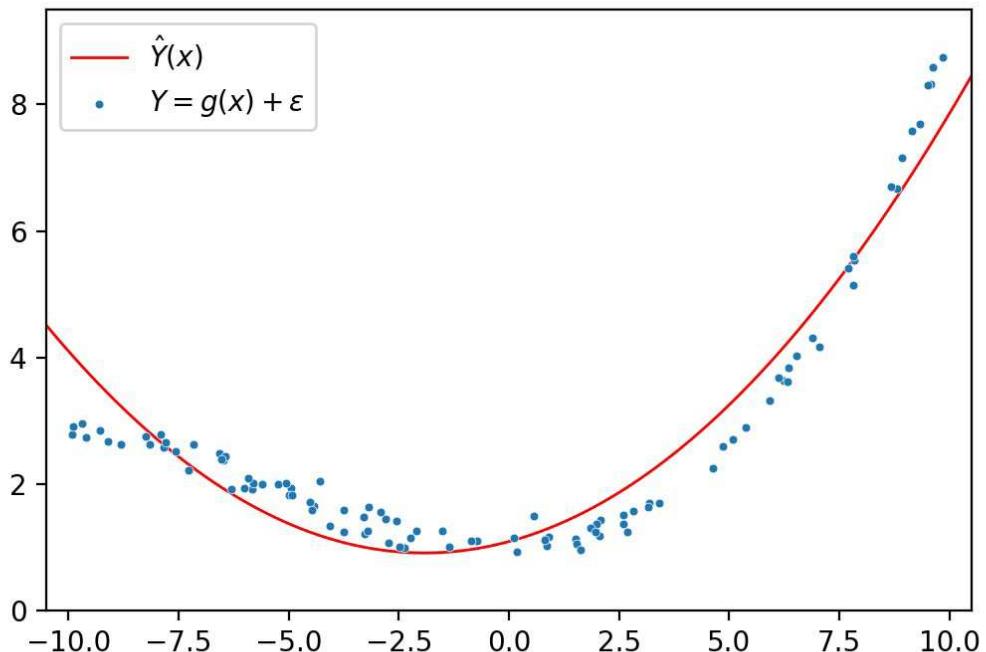


We only see  $Y$



# Our Predictions

- We **choose** a model and fit it to our data
  - Choosing a model is codifying our assumption of the form of  $g(x)$
- The red line is our fitted function—the best possible function given  $g(x)$



At every  $x$ , our prediction for  $Y$  is

- the height of the red line at  $x$
- Denote this  $\hat{Y}(x)$

# Bias and Variance in Modeling

# A Constant Model

Let's say you want to estimate how often a coin lands on heads when flipped.

- The result of a coin flip follows a Bernoulli( $p$ ) distribution, and you want to estimate  $p$ .
- You do not collect any data, but instead you are given a choice between two models.
- Suppose you are also told that  $p = .5$ .

Which of the following is the better model?

**Model A:** Select a random number between 0 and 1. This is your estimate of  $p$ . This is equivalent to running `np.random.random()` in Python.

**Model B:** Select .75 as your estimate of  $p$ .

# A Constant Model

How do we define “better”?

We can calculate the expected MSE of each model. This is called the “model risk,” a term which we will formalize later on. The lower the risk, the better.

**Model A:** *On average*, we will select .5 as our estimate, so we expect 0 error. But, as we are only selecting one number, there is a chance we select a number really far away from .5.

**Model B:** With this model, we will never be exactly correct. But, we know there is zero chance of a really terrible prediction.

# The Bias-Variance Tradeoff

When building models, we generally face a tradeoff between **bias** and **variance**.

- Lower bias means that our model will predict closer to the truth, *on average*.
- Lower variance means that our model will not change too much given the sample.

We want low bias *and* low variance, but oftentimes, when one decreases, the other increases.

**Model A** has zero bias, but lots of variance. **Model B** has zero variance, but lots of bias.

So which is better? The answer will be revealed later in the lecture.

# Three Sources of Error in Our Predictions

**Irreducible error:** Recall the data generating process:  $Y = g(x) + \epsilon$

$$\text{Var}(\epsilon) = \sigma^2 \quad \text{Var}(Y) = \text{Var}(g(x) + \epsilon) = \text{Var}(\epsilon) = \sigma^2$$

There will be chance error in our predictions due to the natural randomness of the world.

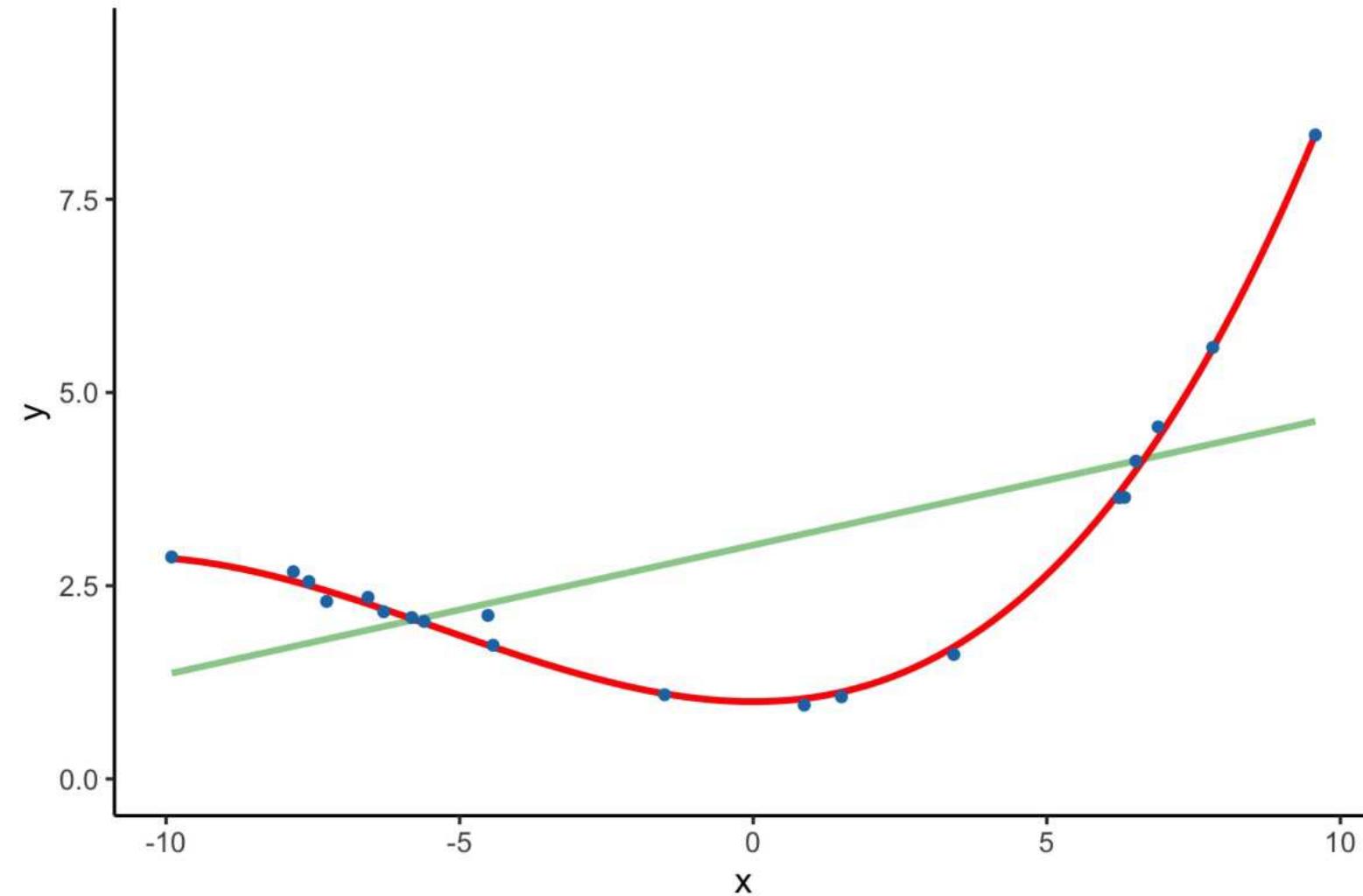
**Model variance:** Our fitted model is based on a random sample.

The sample could have come out differently, then the fitted model would have been different.

**Model bias:** This is the difference between the expected predictions, and the true  $g(x)$ .

Our model may be too limited to find the correct  $g(x)$ , for example if we pick a quadratic model to fit to cubic data.

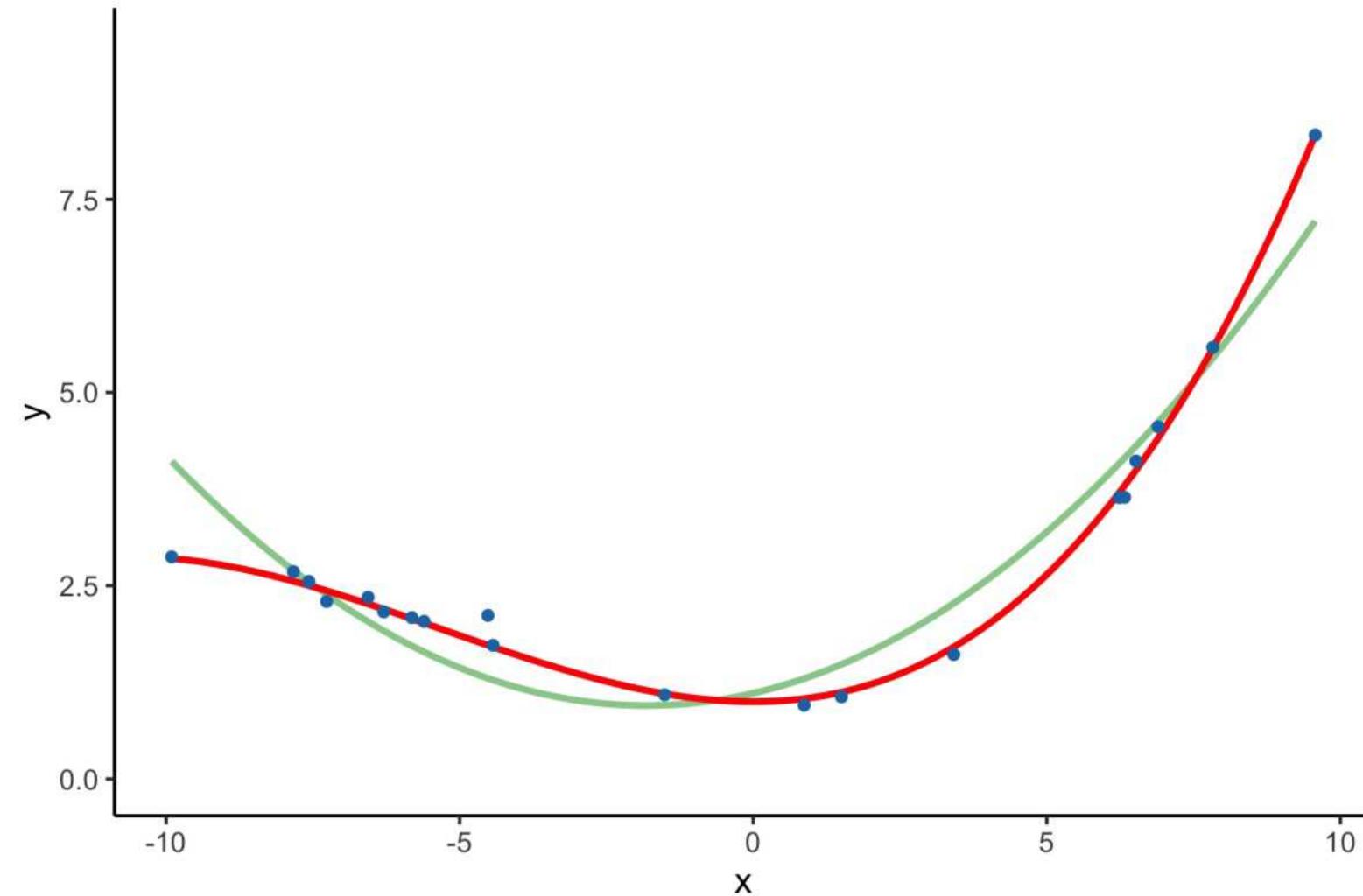
# Simulation



Let's simulate the sampling and modeling process for a strictly linear model

$$g(x) = \theta_0 + \theta_1 x$$

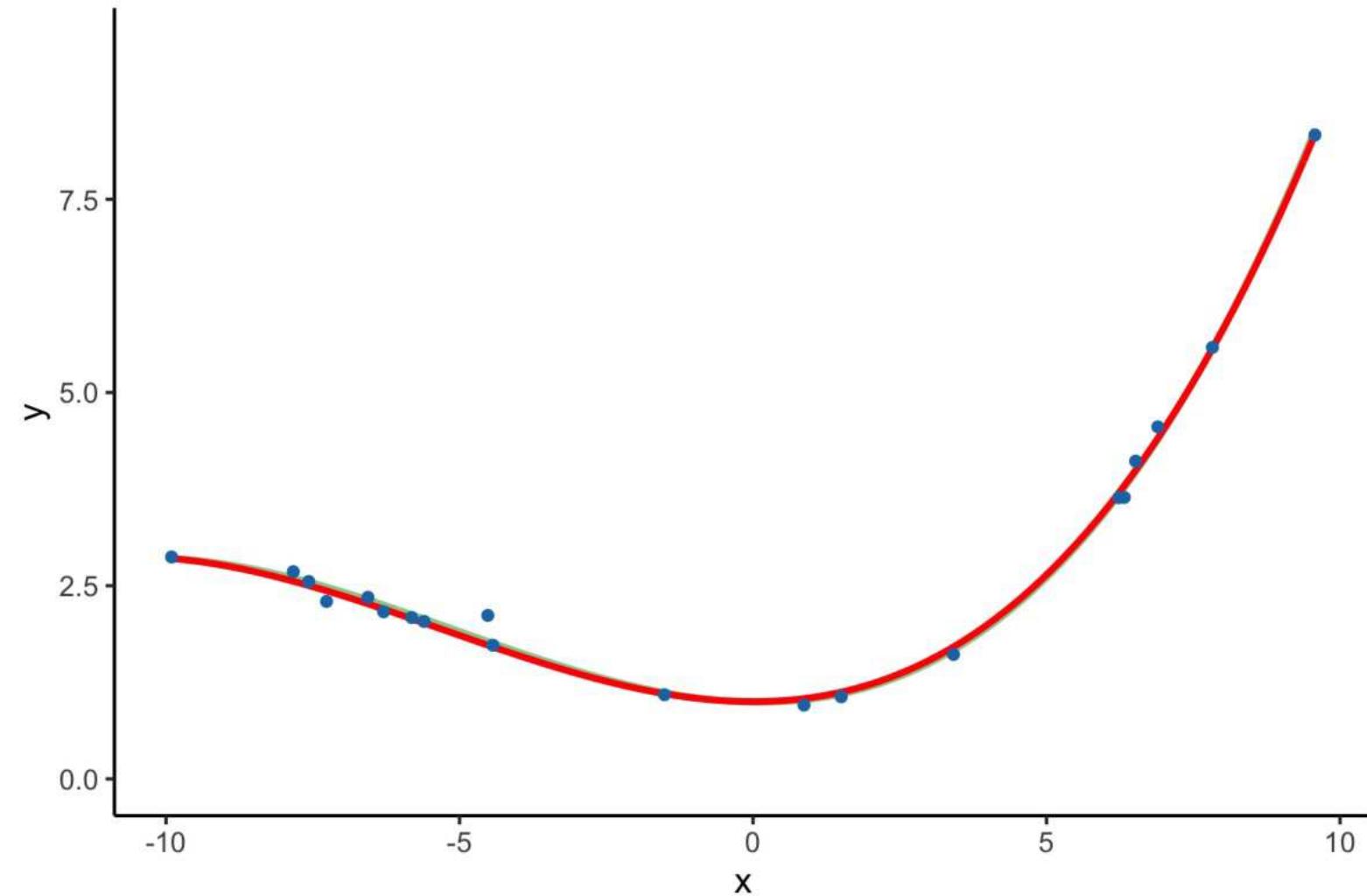
# Simulation



Let's simulate the sampling and modeling process for a quadratic model.

$$g(x) = \theta_0 + \theta_1x + \theta_2x^2$$

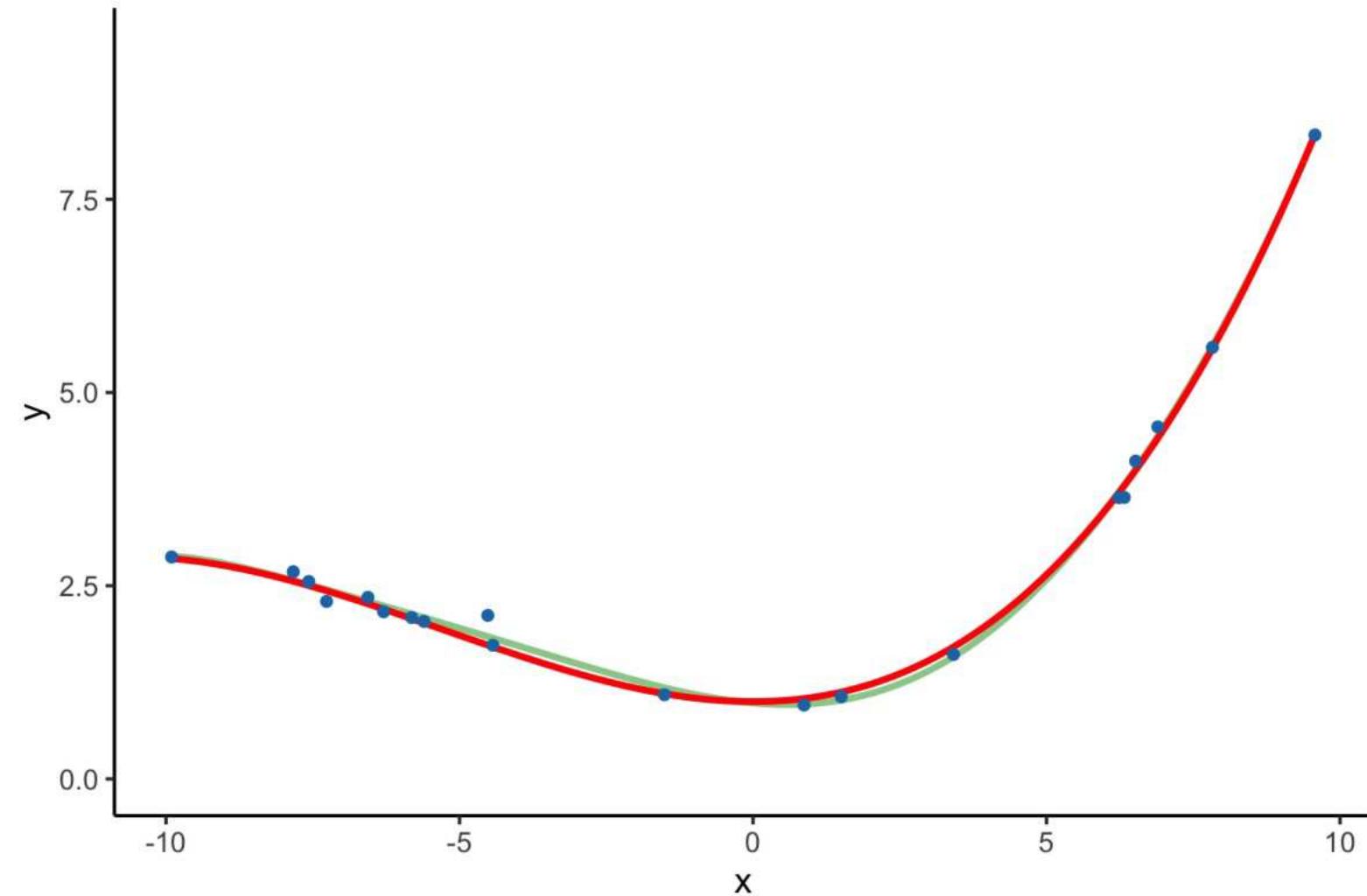
# Simulation



Let's simulate the sampling and modeling process for a cubic model.

$$g(x) = \theta_0 + \theta_1x + \theta_2x^2 + \theta_3x^3$$

# Simulation



Let's simulate the sampling and modeling process for a septic model.

$$g(x) = \theta_0 + \sum_{i=1}^7 \theta_i x^i$$

# Decomposition of Risk

# Model Risk

For a new individual at  $(x, Y)$ :

- Expected mean squared error of prediction:

$$\text{model risk} = \mathbb{E}((Y - \hat{Y}(x))^2)$$

The expectation is taken over all possible samples that we could have collected.

- Remember, each new sample would generate a different  $\hat{Y}(x)$
- Also, for some fixed  $x$ ,  $Y$  can be different due to the random error  $\varepsilon$

# Decomposition of Error and Risk

The model risk can be decomposed into three pieces:

$$\begin{aligned}\mathbb{E}((Y - \hat{Y}(x))^2) &= \mathbb{E}(\epsilon^2) \\ &\quad + (g(x) - \mathbb{E}(\hat{Y}(x)))^2 \\ &\quad + \mathbb{E}((\mathbb{E}(\hat{Y}(x)) - \hat{Y}(x))^2)\end{aligned}$$

$$\text{model risk} = \sigma^2 + (\text{model bias})^2 + \text{model variance}$$

# Bias-Variance Decomposition

model risk = observation variance + (model bias)<sup>2</sup> + model variance

$$\mathbb{E}((Y - \hat{Y}(x))^2) = \sigma^2 + (\mathbb{E}(\hat{Y}(x)) - g(x))^2 + \mathbb{E}((\hat{Y}(x) - \mathbb{E}(\hat{Y}(x)))^2)$$

Remember our assumption about the true relationship  $g(x)$ . When we fit our model, we find some function  $\hat{Y}_H$  that estimates  $g(x)$ .  $\hat{Y}_H$  is random and is just another name for  $\hat{Y}$ .

# Observation Variance

$$\mathbb{V}ar(Y) = \mathbb{V}ar(g(x) + \epsilon) = \mathbb{V}ar(\epsilon) = \sigma^2$$

Some reasons:

- Measurement error
- Missing information acting like noise

Some remedies:

- Could try to get more precise measurements.
- Often this is beyond the control of the data scientist.

# Model Variance

$$\text{model variance} = \text{Var}(\hat{Y}(x)) = \mathbb{E}((\hat{Y}(x) - \mathbb{E}(\hat{Y}(x)))^2)$$

Main reason:

- Overfitting: small differences in random samples lead to large differences in the fitted model

Some remedies:

- Reduce model complexity
- Don't fit the noise

# Model Bias

$$\text{model bias} = \mathbb{E}(\hat{Y}(x)) - g(x)$$

Some reasons:

- Underfitting
- Lack of domain knowledge

Remedies:

- Increase model complexity (but don't overfit)
- Consult domain experts to see which models make sense

# A Constant Model

So which model is better? Model A or Model B?

**Model A:** Select a random number between 0 and 1. This is your estimate of  $p$ .  
This is equivalent to running `np.random.random()` in Python.

**Model B:** Select .75 as your estimate of  $p$ .

We can calculate the model risks directly. Note that the observation variance is 0.

## **Model A:**

$$\text{Model Bias} = .5 - .5 = 0$$

$$\text{Model Variance} = (1 - 0)^2 / 12 = 1/12$$

$$\text{Risk} = 0^2 + 1/12 = \mathbf{1/12}$$

## **Model B:**

$$\text{Model Bias} = .75 - .5 = .25$$

$$\text{Model Variance} = 0$$

$$\text{Risk} = .25^2 + 0 = \mathbf{1/16}$$

# Overfitting

# Introduction to Overfitting

In the previous lectures, our goal has been to **minimize** a loss function (MSE)

- We do this by collecting more features, or through **feature engineering**

However, we only ever evaluated our model on the **data on which it was trained**

- The whole point in building a model is to *learn something about the world*
- Why do we care about finding a  $\hat{y}$  if we already know  $y$ ?

We care about how well our model performs on **new** data, for which we want to **predict**

# Complexity

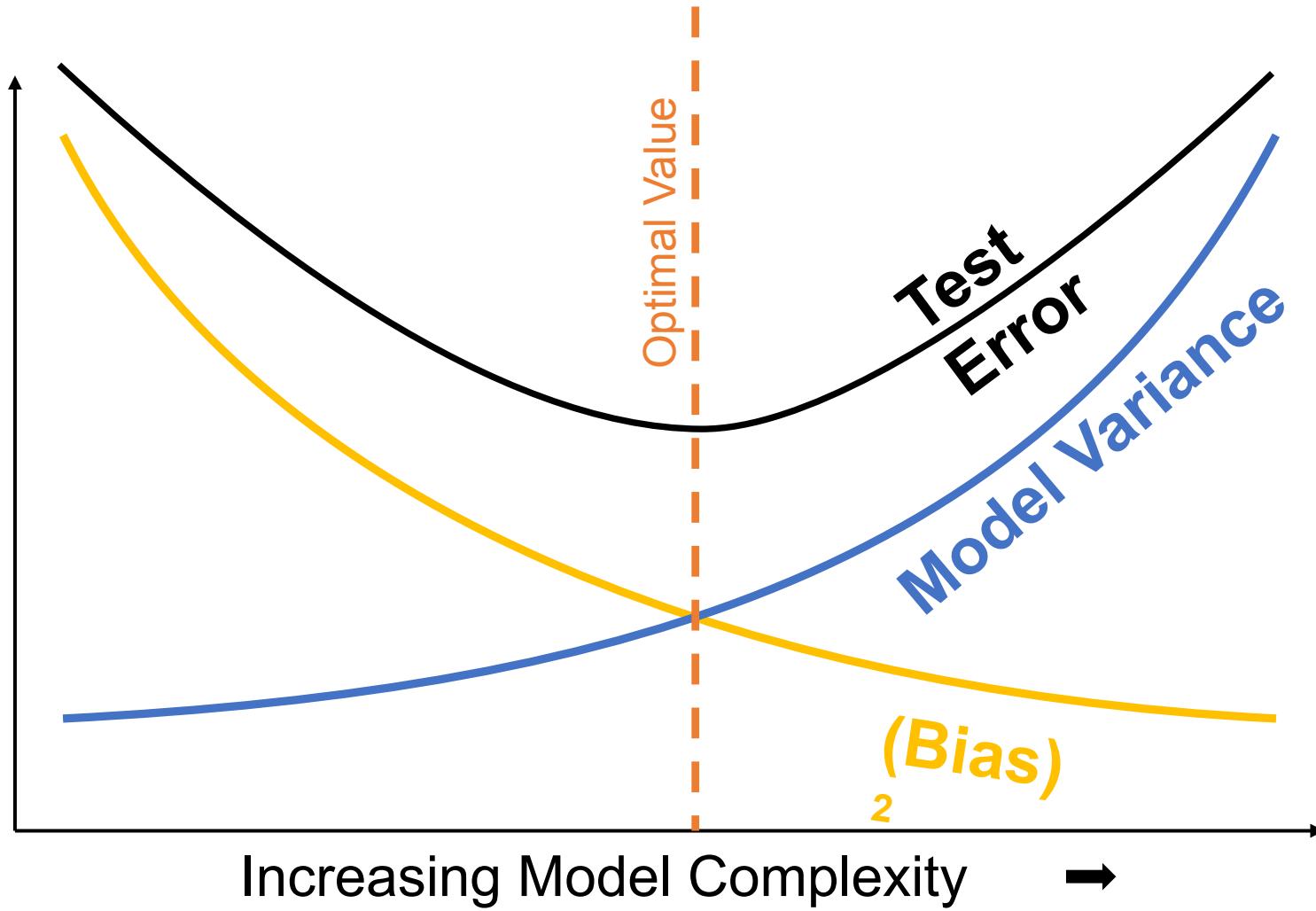
# Modeling Goals

- Try to minimize all three of observation variance, model bias, and model variance.

*But*

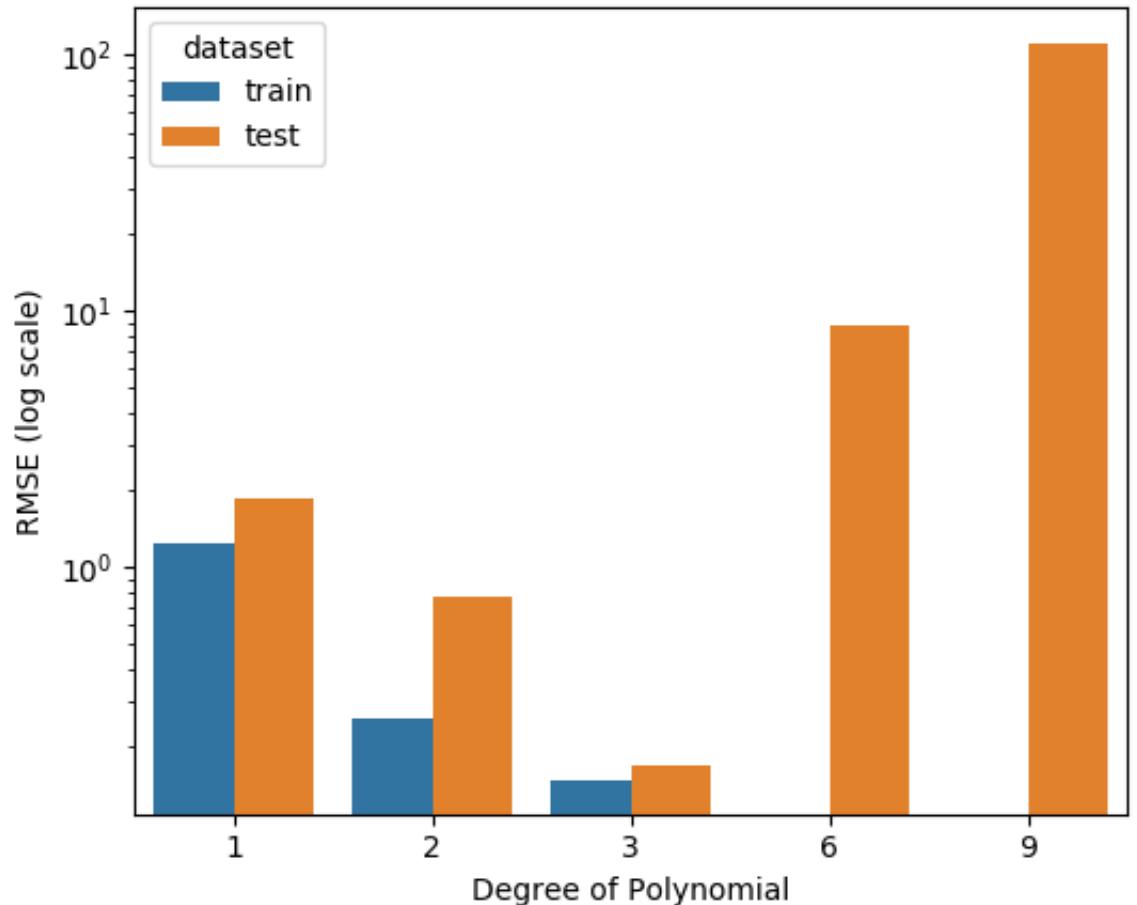
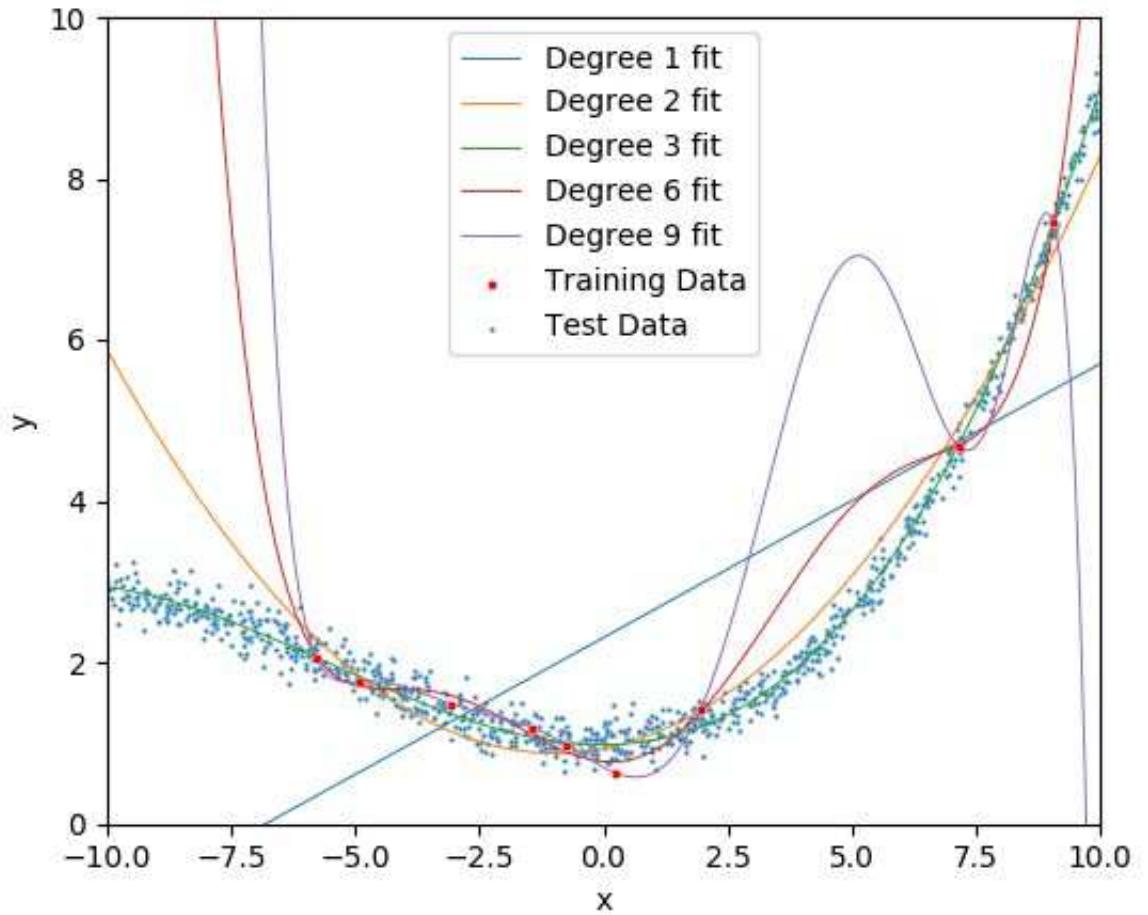
- Observation variance is often out of our control
- Reducing complexity to reduce model variance can increase bias
- Increasing model complexity to reduce bias can increase model variance
- Domain knowledge matters: the right model structure!

# Bias Variance Plot

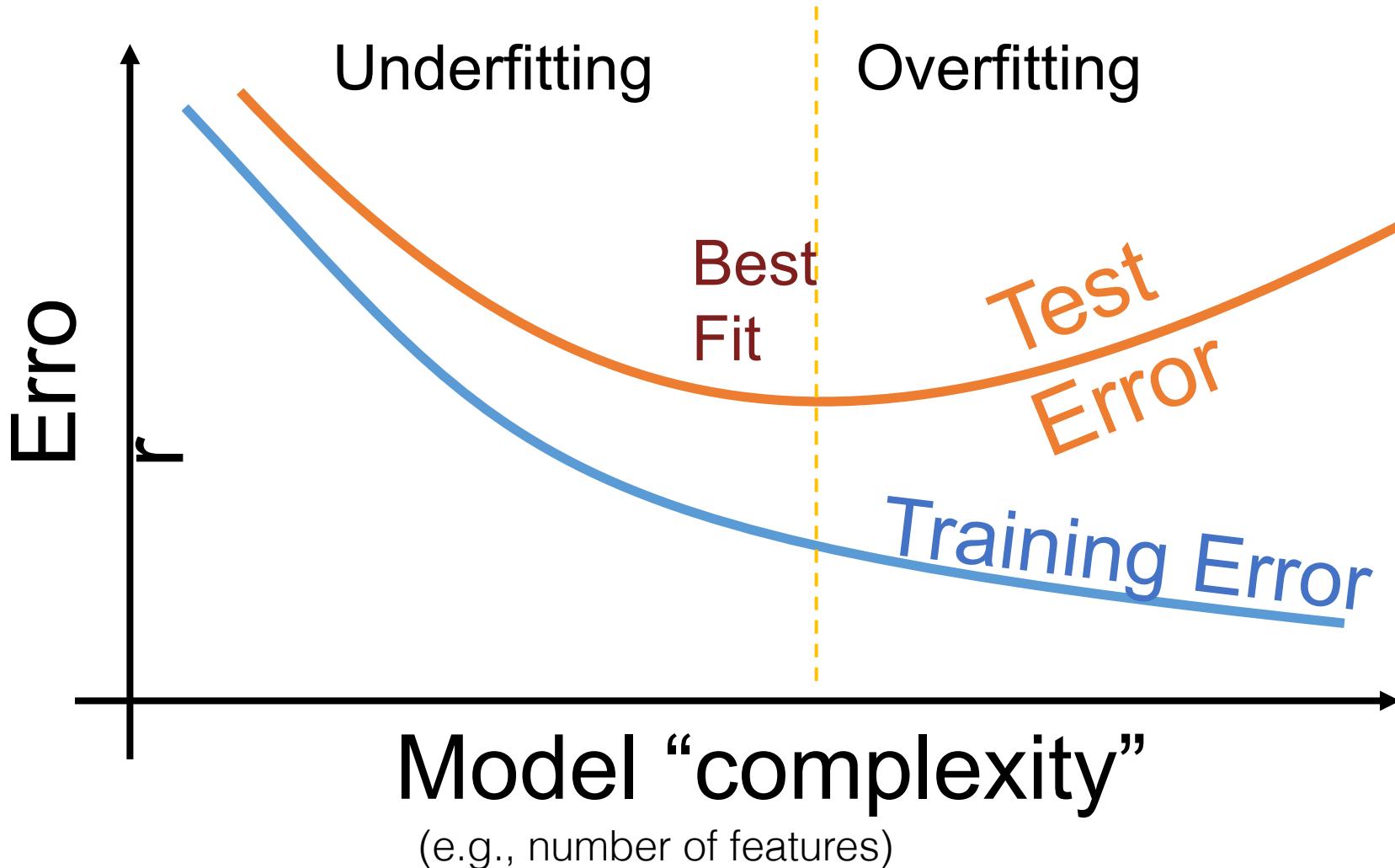


# Cross-Validation

# Training Error vs Test Error



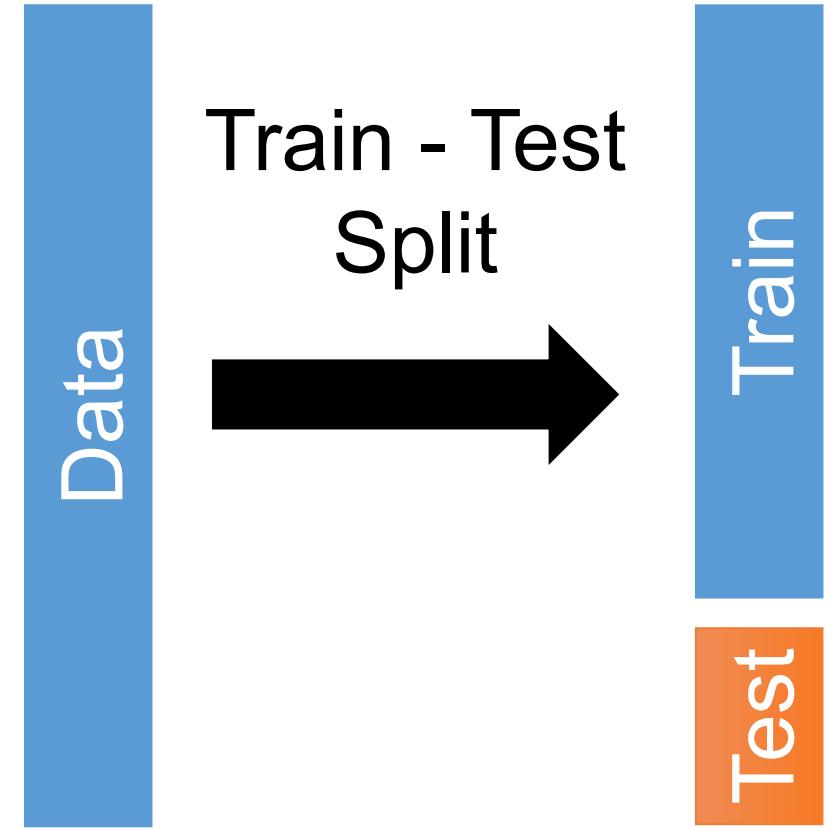
# Training Error vs Test Error



**Training error**  
typically  
*underestimates*  
**test error.**

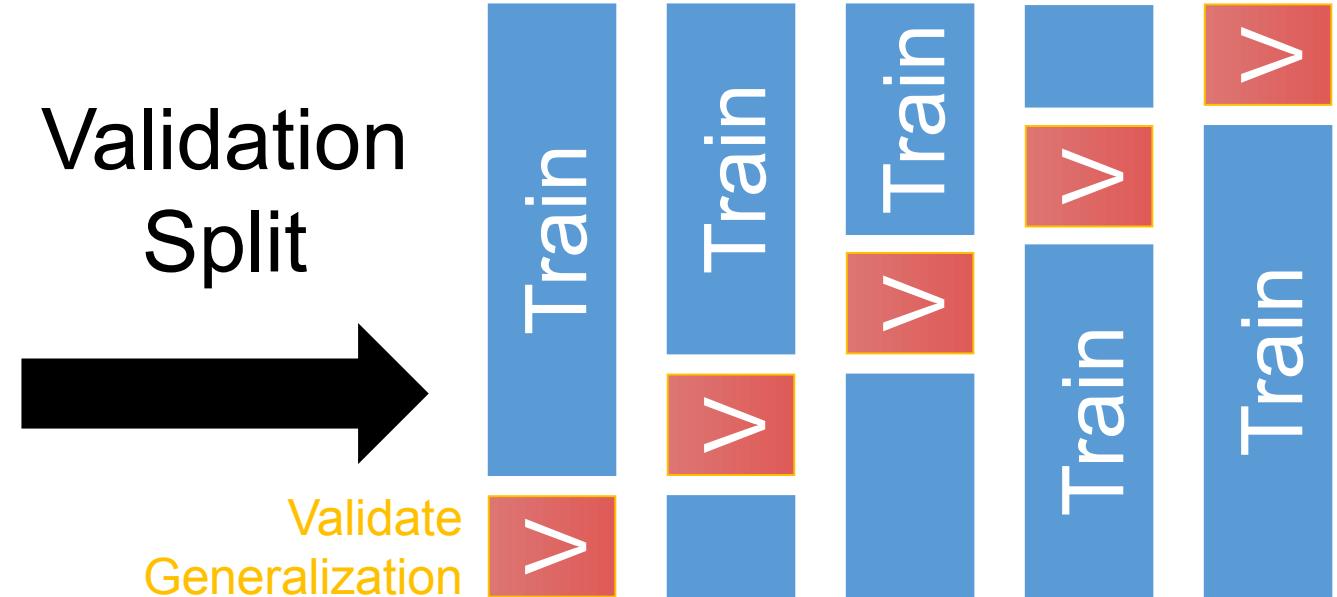
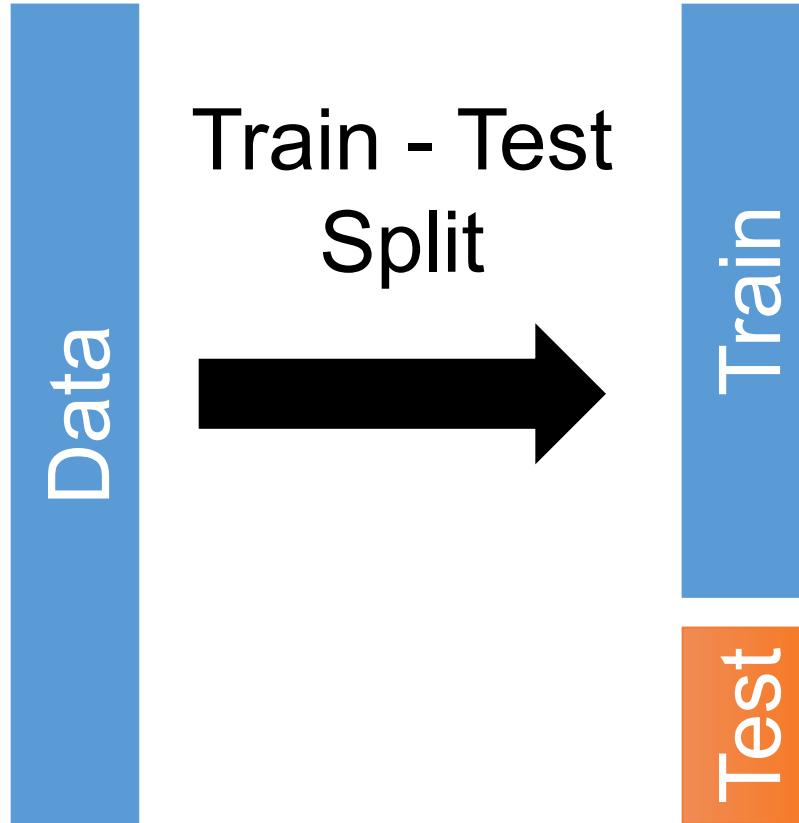
# Generalization: *The Train-Test Split*

- **Training Data:** used to fit model
- **Test Data:** check generalization error
- How to split?
  - Depends on application (usually randomly)
- What size? (90%-10%)
  - Larger training set – more complex models
  - Larger test set – better estimate of generalization error
  - Typically between 70%-30% and 90%-10%



You can only use the test dataset once after deciding on the model.

# Generalization: Validation Split



5-Fold  
Cross Validation

*Cross validation **simulates multiple train test-splits** within the training data.*

# Recipe for Successful Generalization

1. Split your data into **training** and **test** sets (90%, 10%)
2. Use **only the training data** when designing, training, and tuning the model
  - Use **cross validation** to test *generalization* during this phase
  - **Do not look at the test data!**
1. Commit to your final model and train once more using **only the training data**.
2. Test the final model using the **test data**.
3. Train on **all available data** and ship it!

# Demo

# Next week

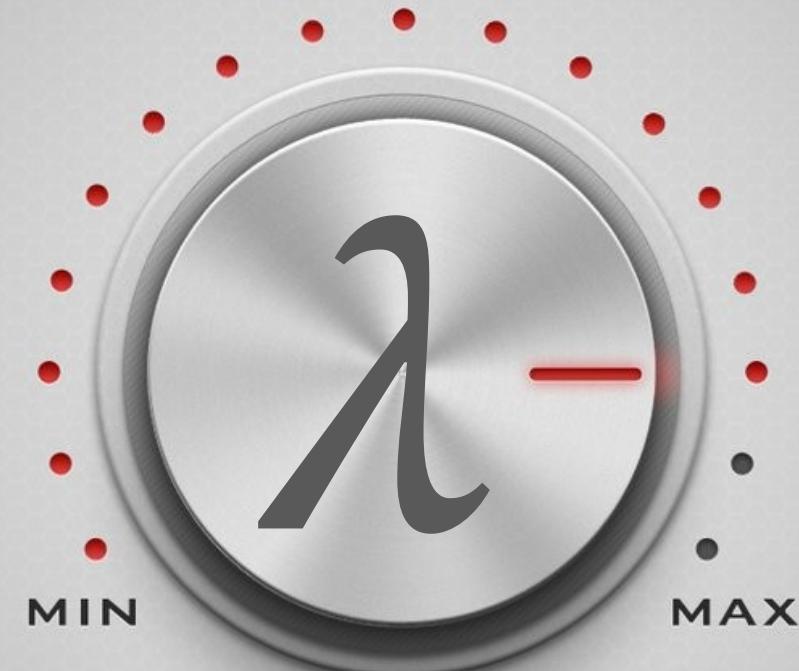
- Another strategy to possibly overcome overfitting:
  - Regularization

**YaleNUSCollege**

**YSC2239 Lecture 19**

# Regularization

Controlling the  
*Model Complexity*

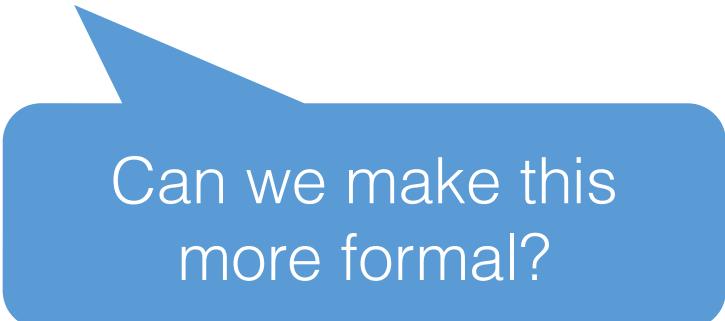


# Basic Idea

$$\hat{\theta} = \arg \min_{\theta} \frac{1}{n} \sum_{i=1}^n \text{Loss}(y_i, f_{\theta}(x_i))$$

**Such  
that:**

$f_{\theta}$  does not “overfit”



Can we make this  
more formal?

# Basic Idea

$$\hat{\theta} = \arg \min_{\theta} \frac{1}{n} \sum_{i=1}^n \text{Loss}(y_i, f_{\theta}(x_i))$$

Such  
that:

$$\text{Complexity}(f_{\theta}) \leq \beta$$

How do we define  
this?

Regularization  
Hyperparameter

# Idealized Notion of Complexity

$$\text{Complexity}(f_\theta \leq \beta)$$

- Focus on complexity of **linear models**:
  - Number and kinds of features
- Ideal definition:

$$\text{Complexity}(f_\theta) = \sum_{j=1}^d \mathbb{I}[\theta_j \neq 0]$$

Number of  
non-zero  
parameters

- Why?

# Ideal “Regularization”

Find the best value of  $\theta$  which uses fewer than  $\beta$  features.

$$\hat{\theta} = \arg \min_{\theta} \frac{1}{n} \sum_{i=1}^n \text{Loss}(y_i, f_{\theta}(x_i))$$

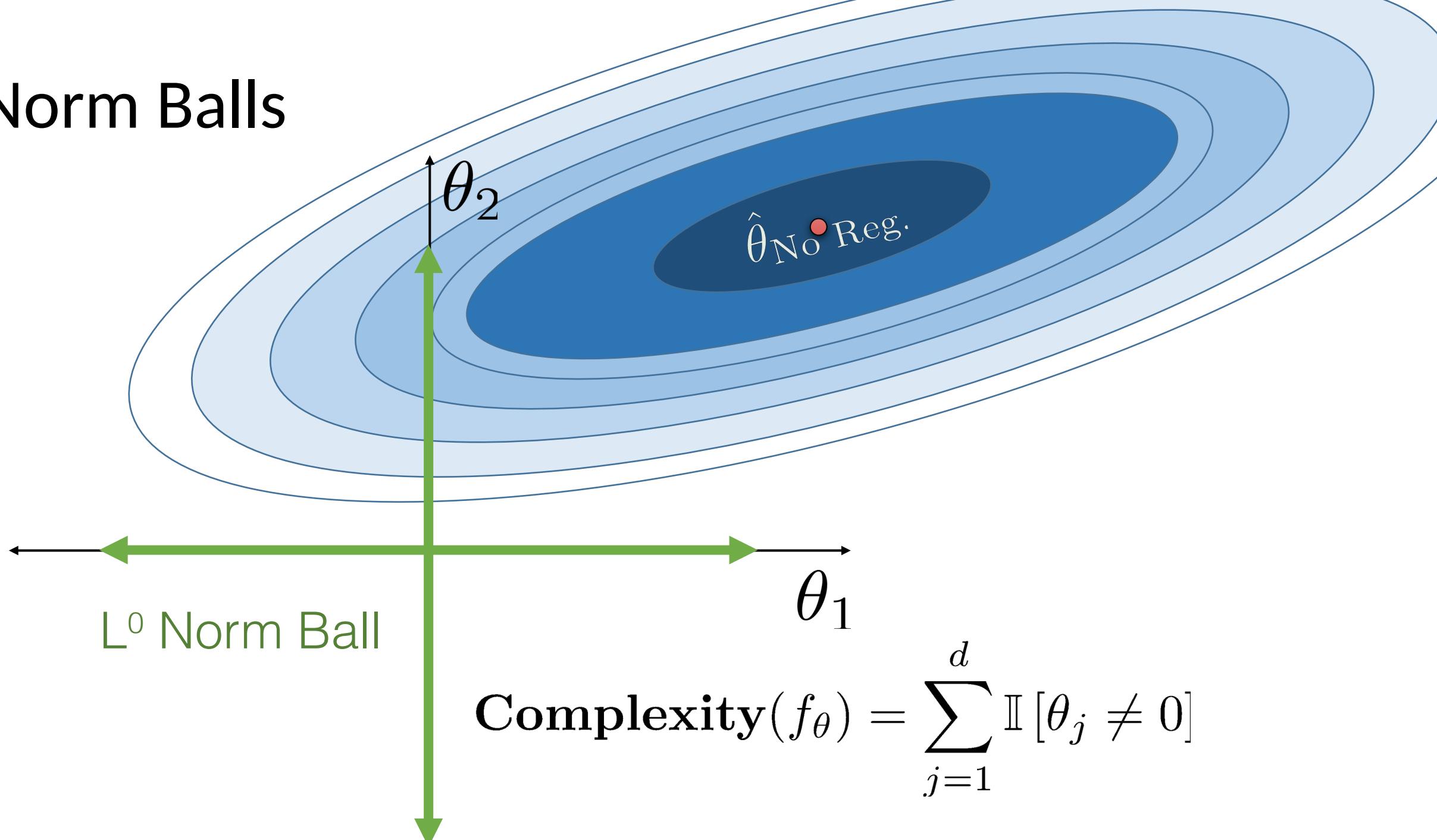
Such  
that:

Need an approximation!

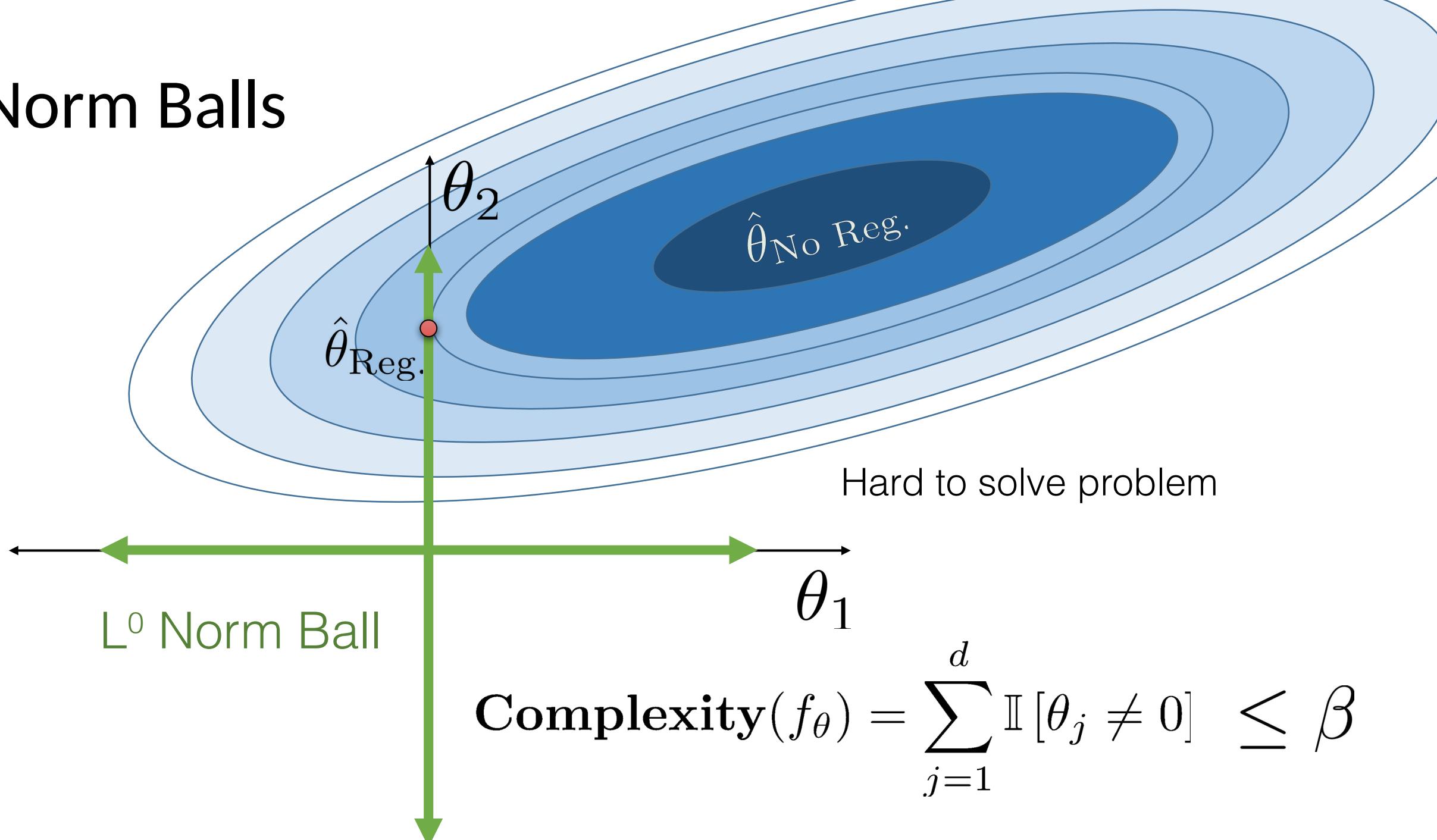
$$\text{Complexity}(f_{\theta}) = \sum_{j=1}^d \mathbb{I}[\theta_j \neq 0] \leq \beta$$

Combinatorial search problem – NP-hard to solve in general.

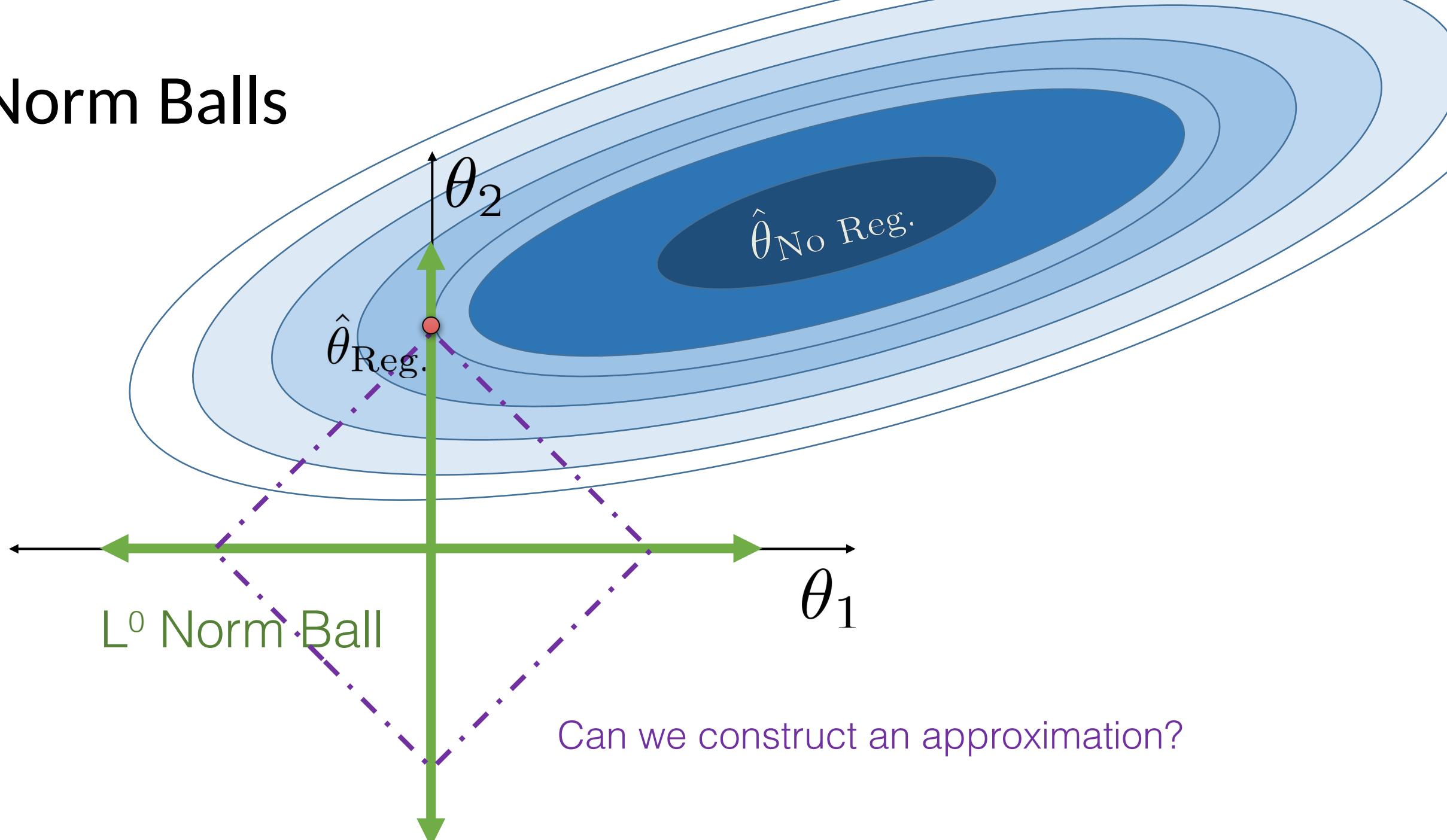
# Norm Balls



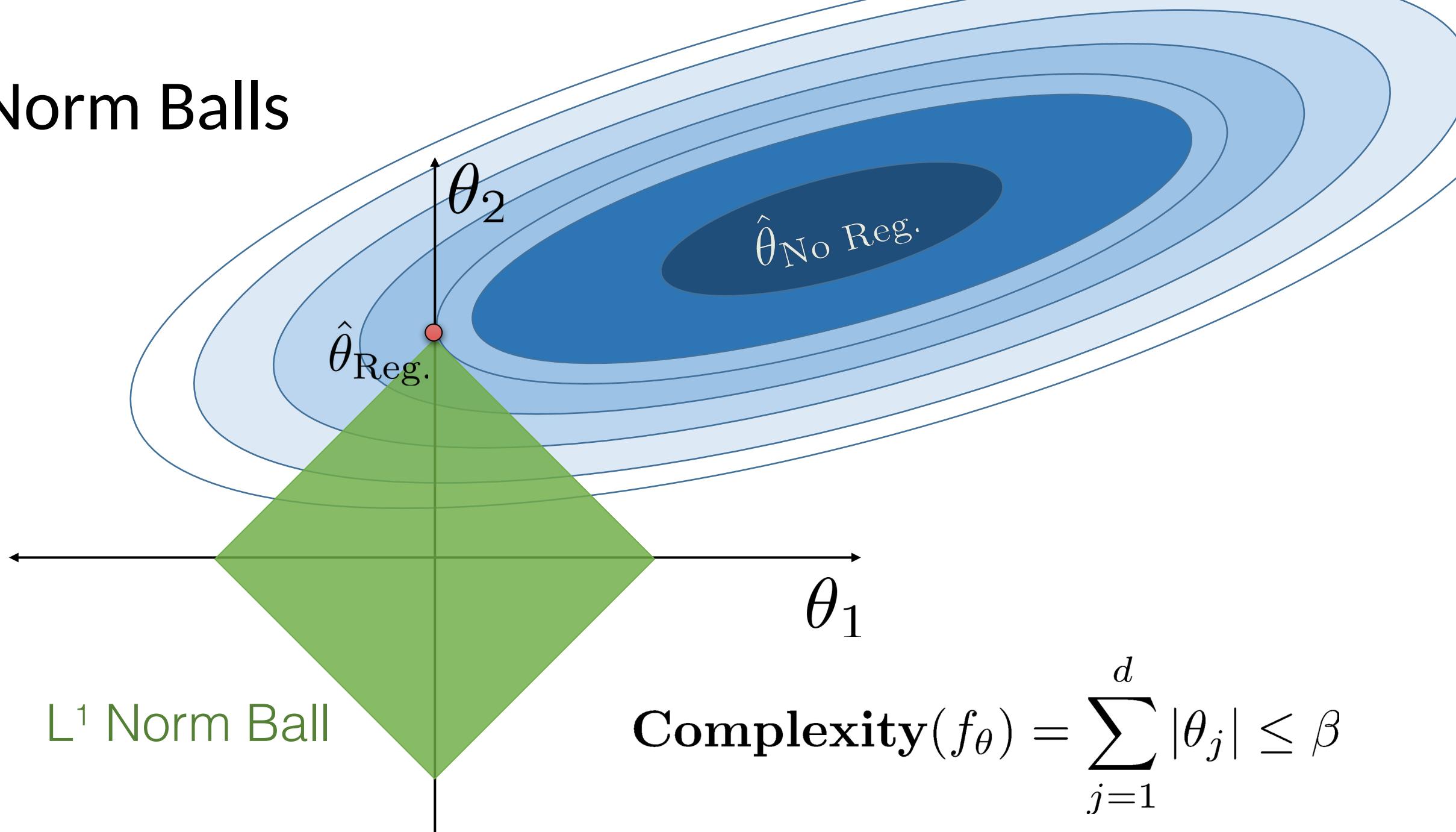
# Norm Balls



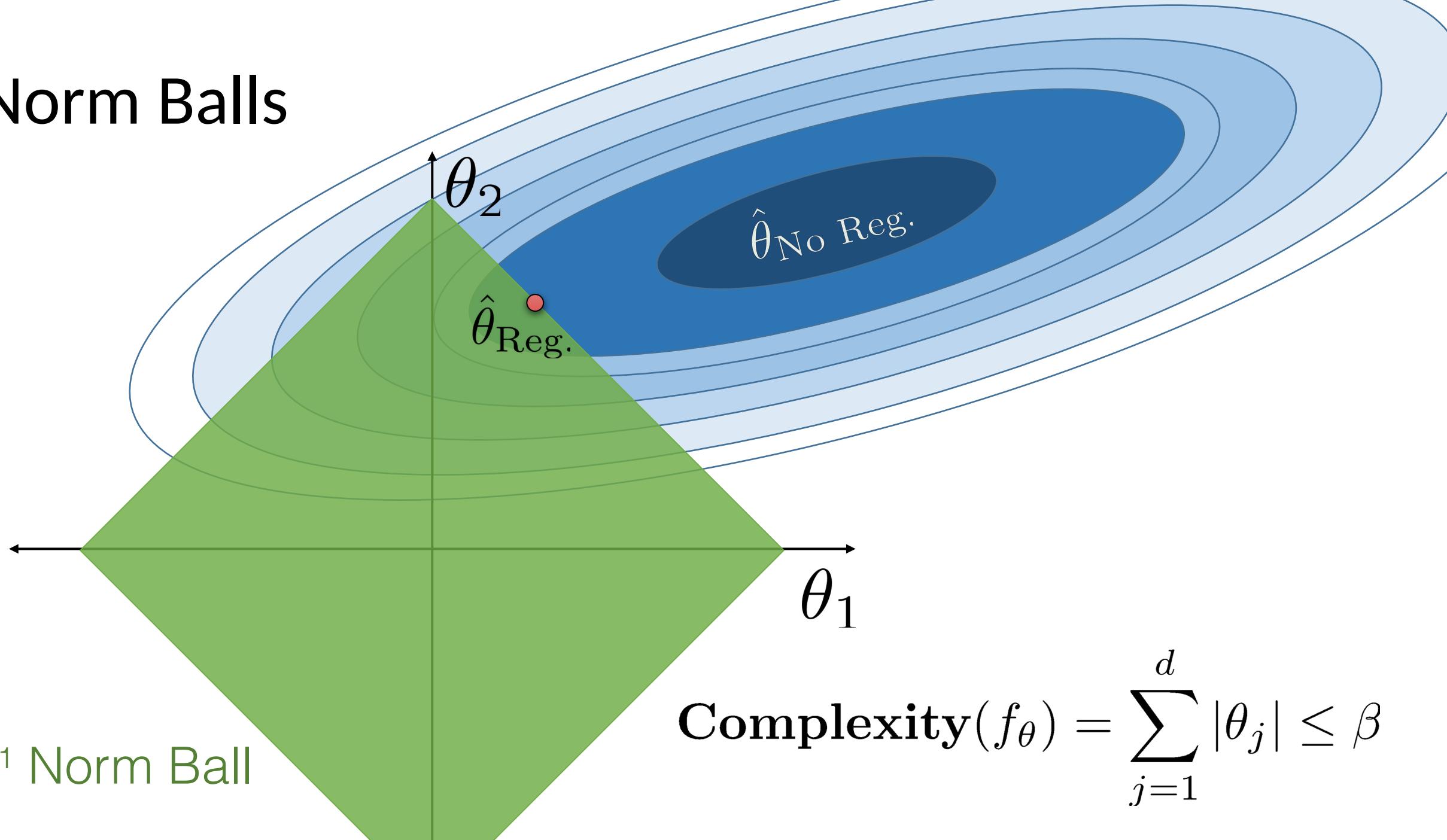
# Norm Balls



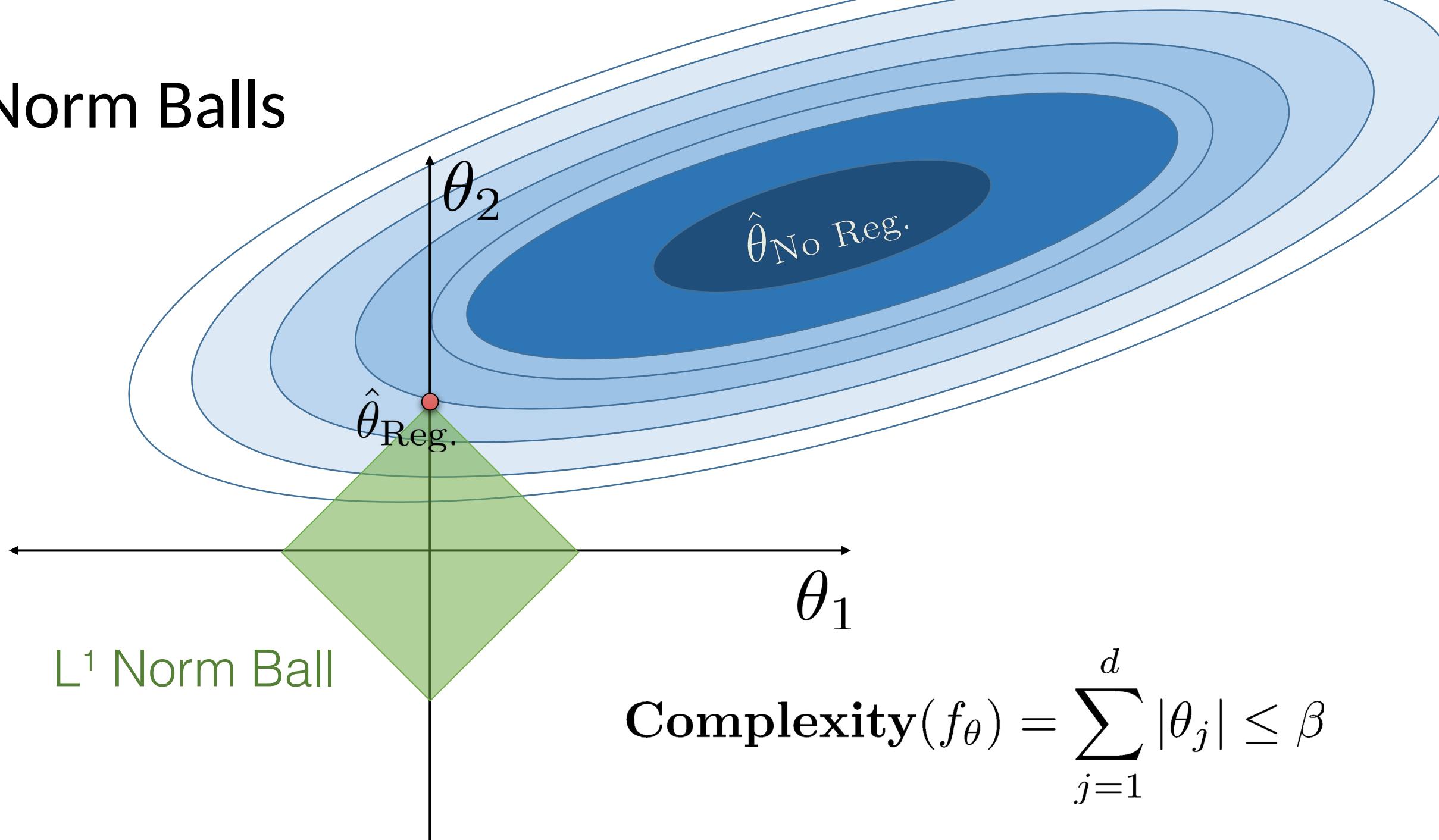
# Norm Balls



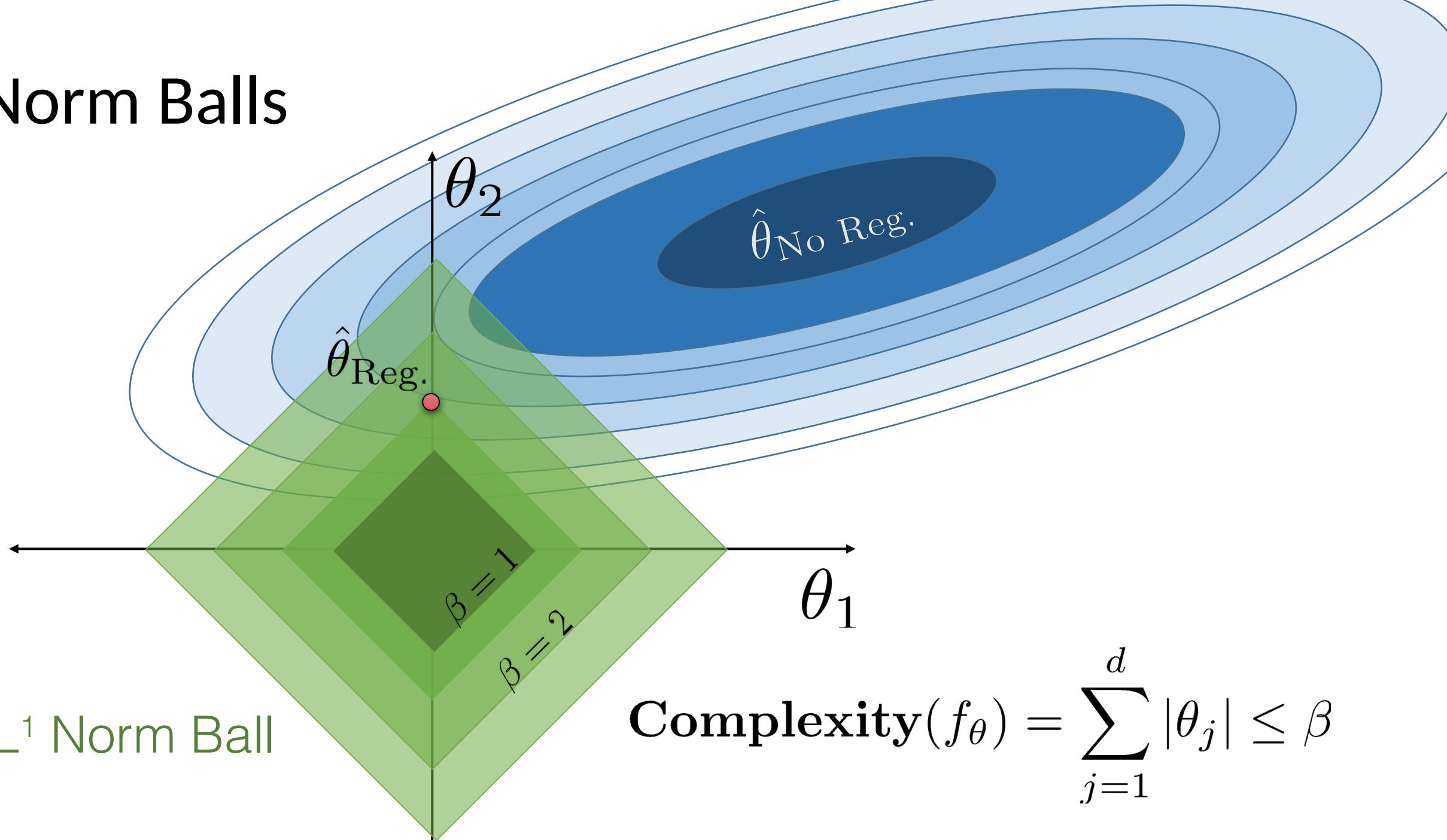
# Norm Balls



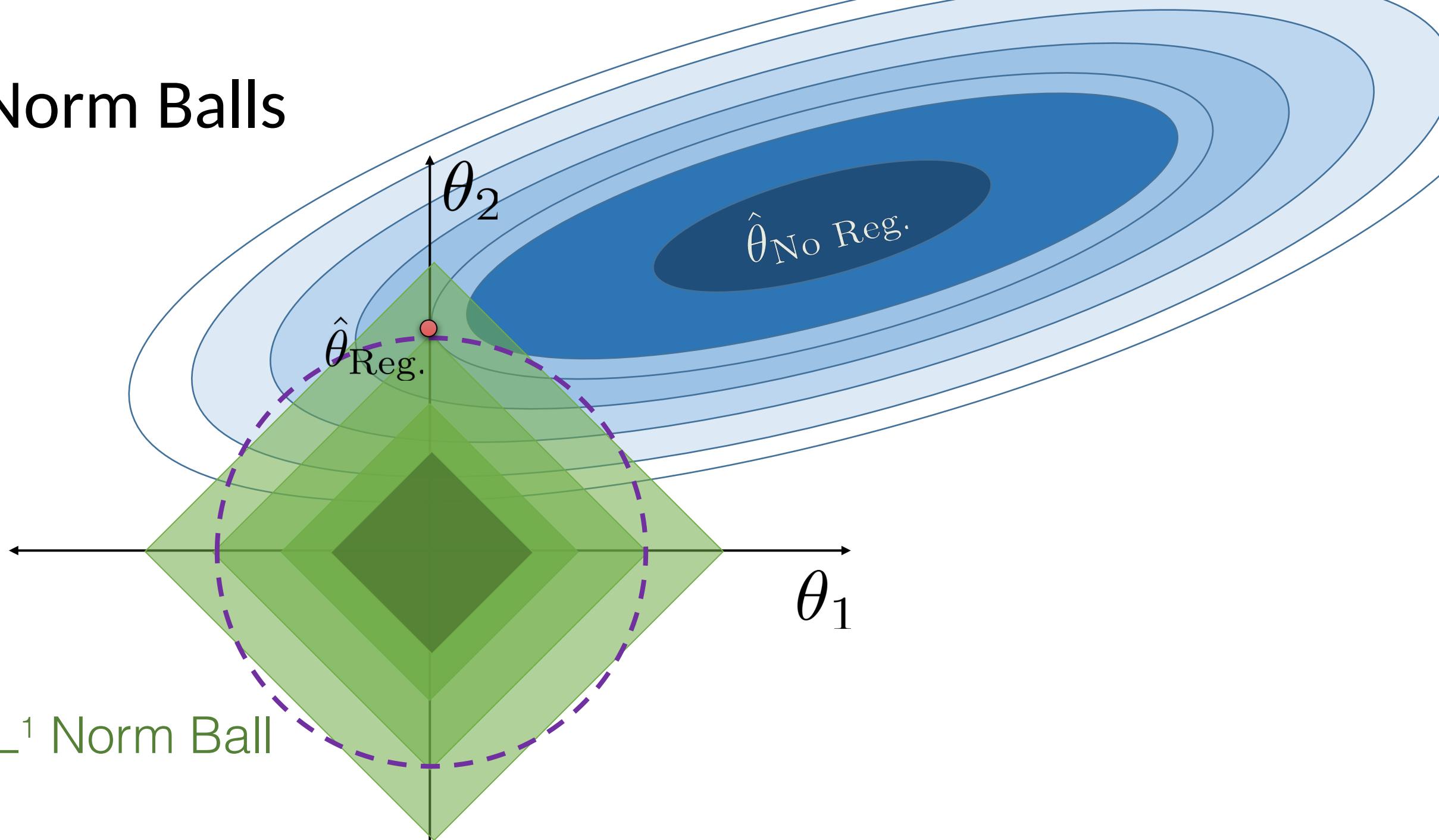
# Norm Balls



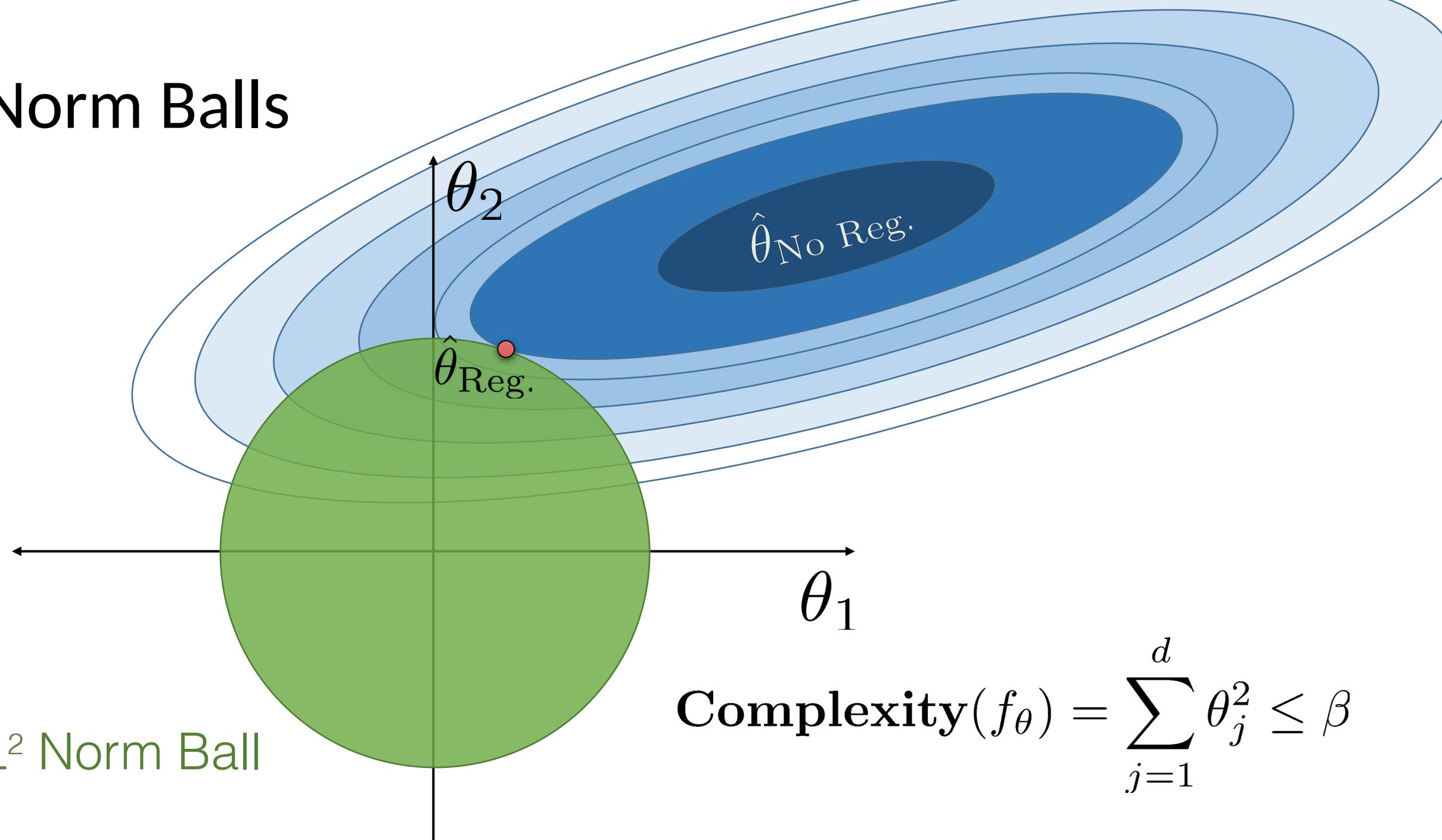
# Norm Balls



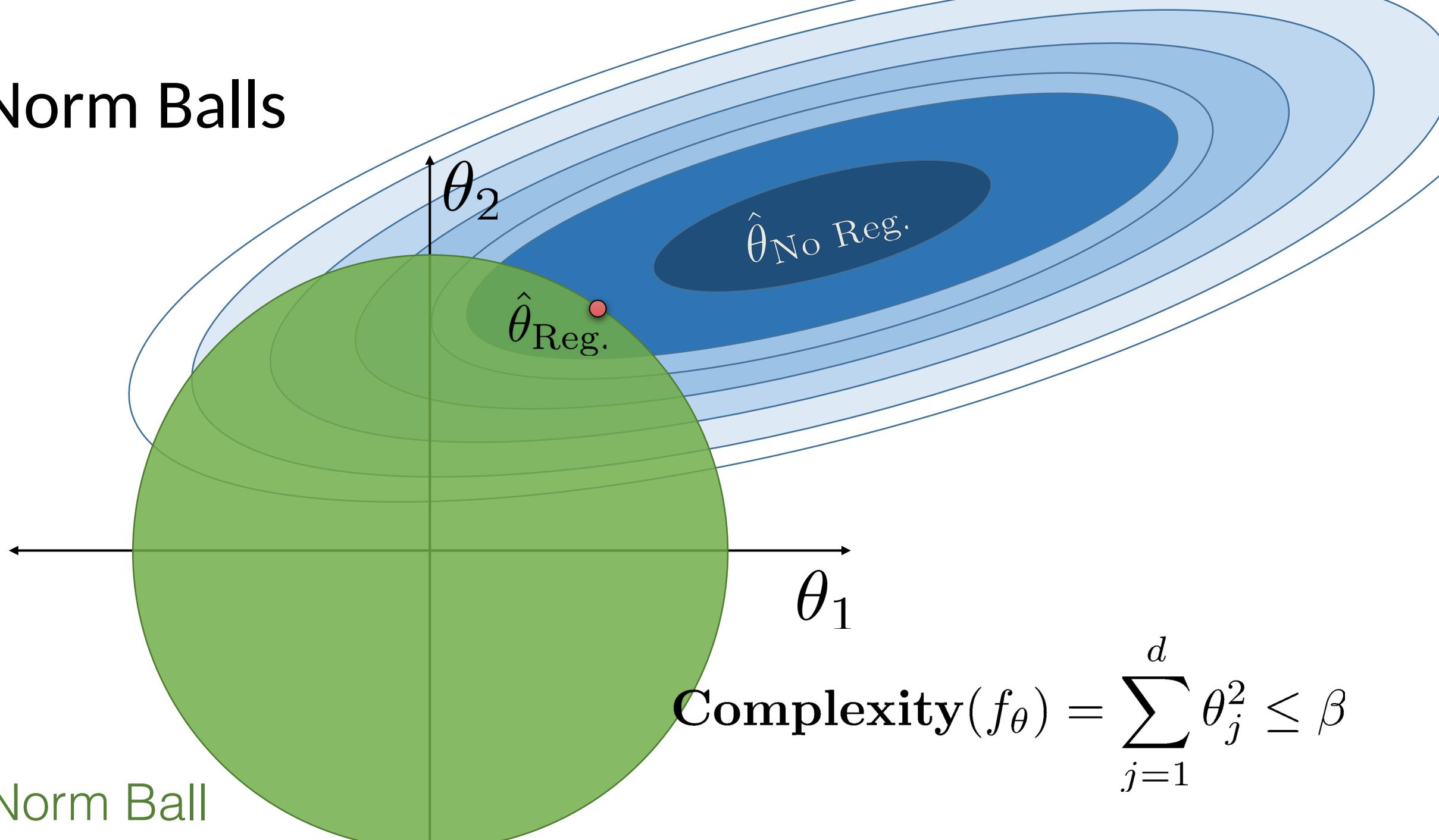
# Norm Balls



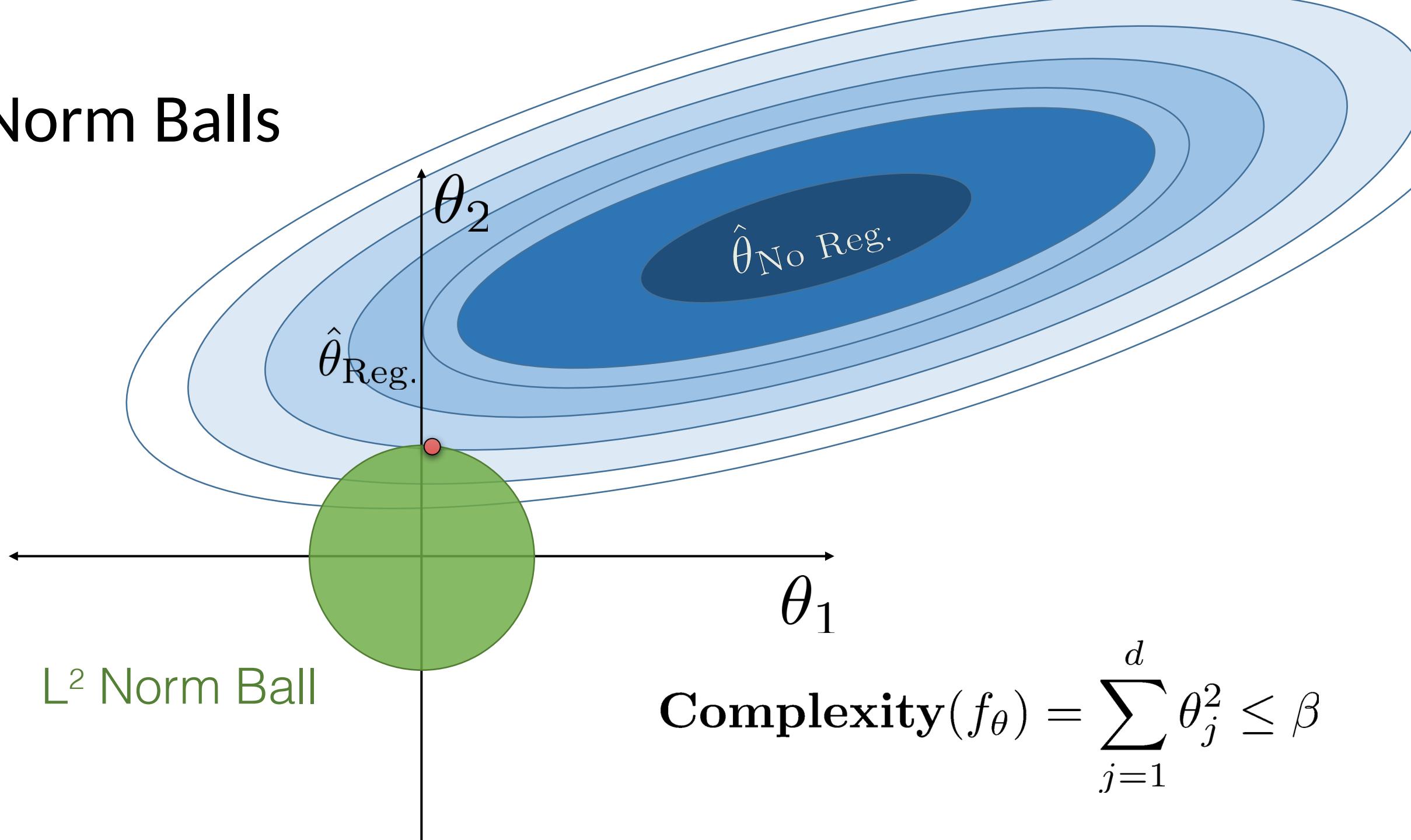
# Norm Balls



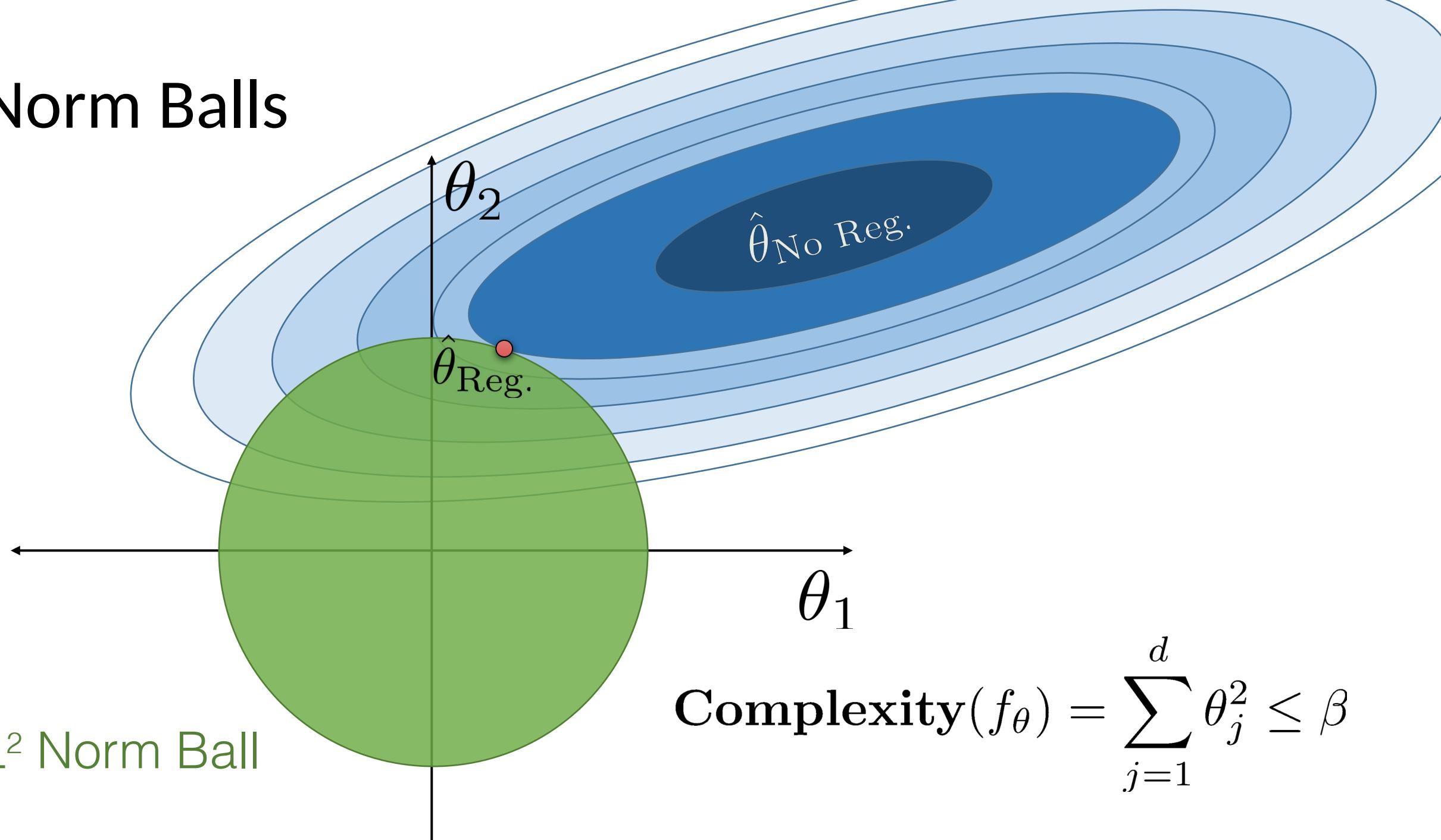
# Norm Balls



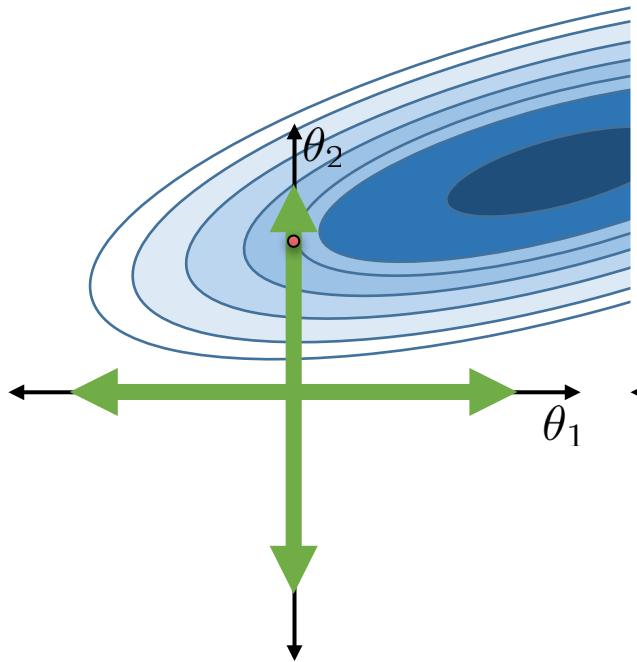
# Norm Balls



# Norm Balls



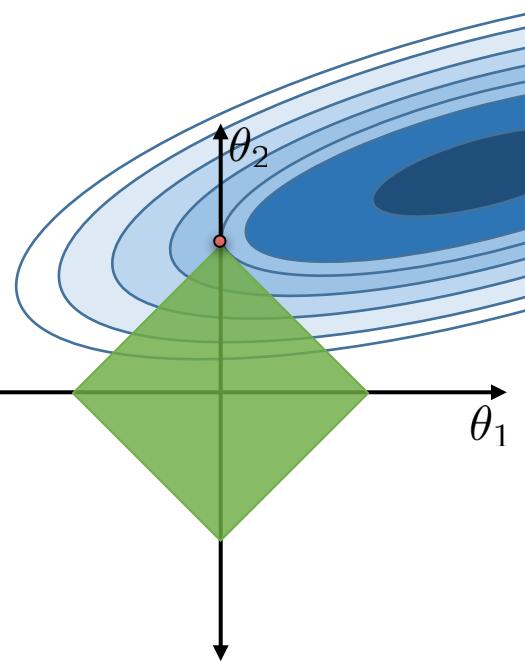
## $L^0$ Norm Ball



Ideal for  
**Feature  
Selection**

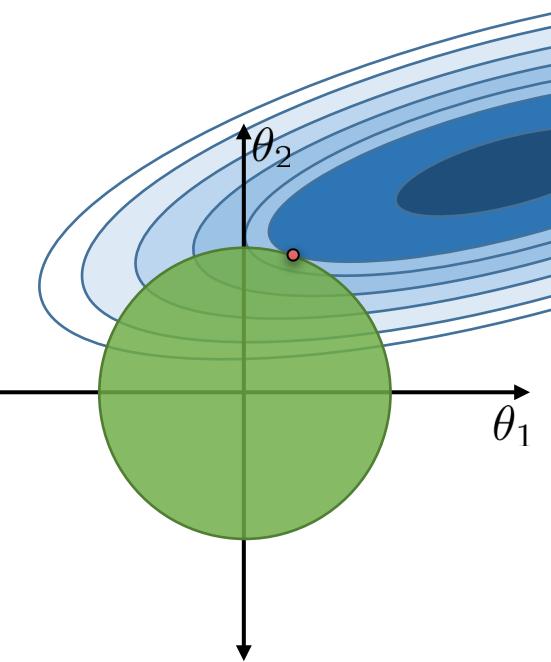
but combinatorically  
difficult to optimize

## $L^1$ Norm Ball



Encourages  
sparse solutions

## $L^2$ Norm Ball



Spreads weight  
over features, but does  
not  
encourage sparsity

# Ridge and LASSO Regression

# Ridge Regression

“Ridge Regression” is a term for the following specific combination of model, loss, and regularization:

- Model:  $\hat{\mathbb{Y}} = \mathbb{X}\hat{\theta}$
- Loss: Squared loss
- Regularization: L2 regularization

The **objective function** we minimize for Ridge Regression is average squared loss, plus an added penalty:

$$\hat{\theta}_{\text{ridge}} = \arg \min_{\theta} \frac{1}{n} \|\mathbb{Y} - \mathbb{X}\theta\|_2^2 + \lambda \sum_{j=1}^d \theta_i^2$$

# LASSO Regression

“LASSO Regression” is a term for the following specific combination of model, loss, and regularization:

- Model:  $\hat{\mathbb{Y}} = \hat{\mathbf{X}}\hat{\theta}$
- Loss: Squared loss
- Regularization: L1 regularization

The **objective function** we minimize for LASSO Regression is average squared loss, plus an added penalty:

$$\hat{\theta}_{\text{LASSO}} = \arg \min_{\theta} \frac{1}{n} \|\mathbb{Y} - \mathbf{X}\theta\|_2^2 + \lambda \sum_{j=1}^d |\theta_j|$$

# Summary of Regression Methods

Name	Model	Loss	Reg.	Objective
OLS	$\hat{Y} = \mathbb{X}\hat{\theta}$	Squared loss	None	$\frac{1}{n} \ \mathbb{Y} - \mathbb{X}\theta\ _2^2$
Ridge Regression	$\hat{Y} = \mathbb{X}\hat{\theta}$	Squared loss	L2	$\frac{1}{n} \ \mathbb{Y} - \mathbb{X}\theta\ _2^2 + \lambda \sum_{j=1}^d \theta_i^2$
LASSO	$\hat{Y} = \mathbb{X}\hat{\theta}$	Squared loss	L1	$\frac{1}{n} \ \mathbb{Y} - \mathbb{X}\theta\ _2^2 + \lambda \sum_{j=1}^d  \theta_i $

# Hyperparameters vs. Parameters

**Parameters** are facts about the world that we want to *estimate*

- Commonly denoted by  $p, \theta, \theta_i$

**Statistics** are the *estimators* of the parameters, based on our data

- Commonly denoted by  $\hat{p}, \hat{\theta}, \hat{\theta}_i$

**Hyperparameters** are design *choices* we make in our modeling process that

affect our model, but do not directly come from the data

- examples: regularization hyperparameter, degree of polynomial  
 $\lambda, \alpha, C$
- Commonly denoted by

# Demo

**YaleNUSCollege**

**YSC2239 Lecture 19-20**

# Today's class

- Logistic regression – Part 1

# Regression vs. Classification

# Linear Regression

In a **linear regression** model, our goal is to predict a **quantitative** variable (i.e., some real number) from a set of features.

- Our output, or **response**,  $y$ , could be any real number.
- We determined optimal model parameters by minimizing some average loss, and added a regularization penalty.

$$\hat{y} = f_{\theta}(x) = x^T \theta$$

$$x^T \theta = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_p x_p$$

Remember,

# Classification

When performing classification, we are instead interested in predicting some **categorical** variable.

win or lose

spam or ham

disease or  
no disease

# Classification

- **Binary** classification: two classes.
  - Examples: spam / not spam.
  - Our **responses** are either 0 or 1.
  - Our focus today.
- **Multiclass** classification: many classes.
  - Examples: Image labeling (cat, dog, car), next word in a sentence, etc.

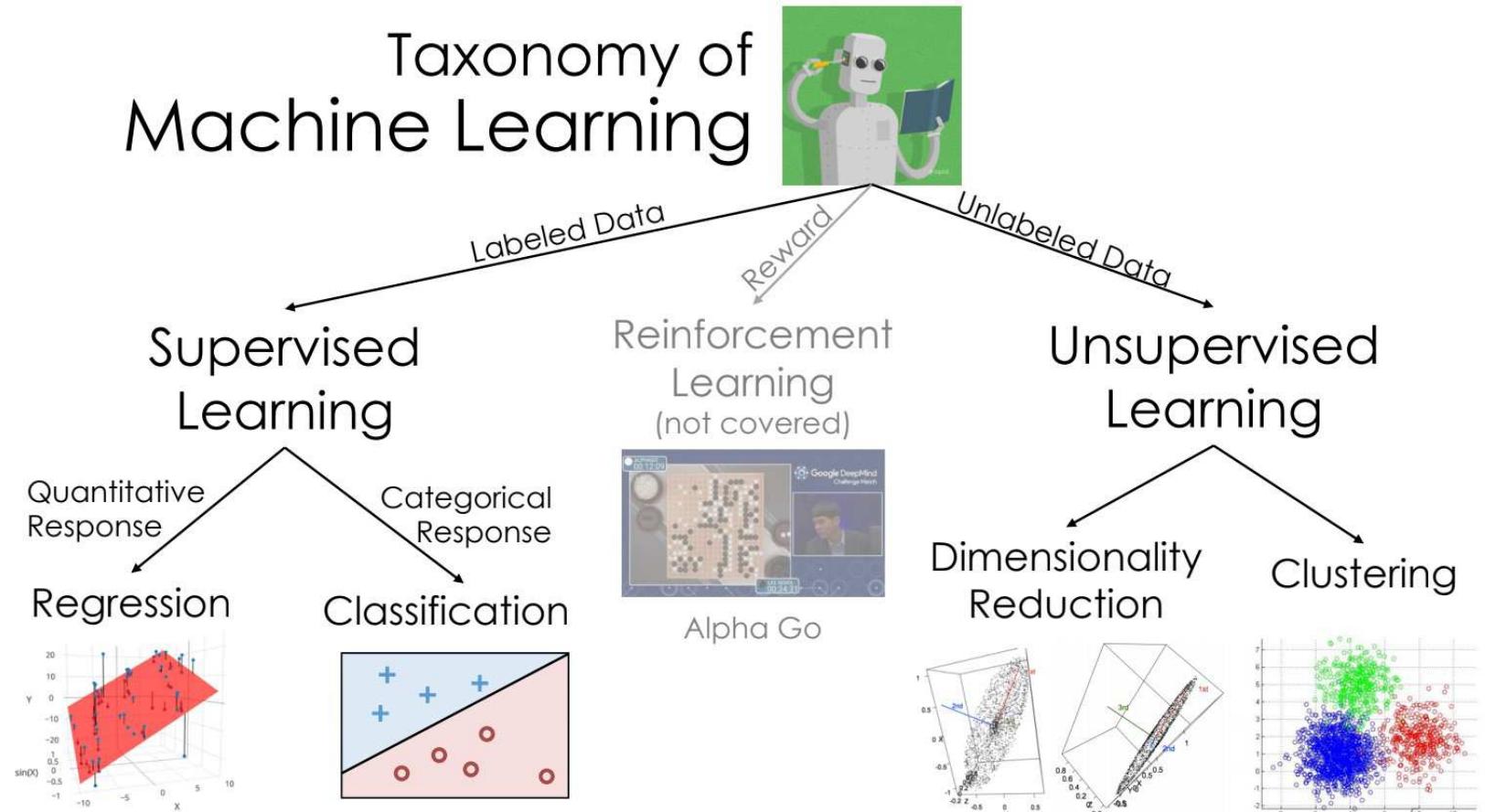
This is not the first time you are seeing classification!

- k-Nearest Neighbors was a classification technique we have learned earlier.

# Machine learning taxonomy

Regression and Classification are both forms of **supervised learning**.

**Logistic regression**, the topic of this lecture, is mostly used for **classification**, even though it has “regression” in the name.



from Joseph Gonzalez

# Deriving the logistic regression model

In this section, we will mostly work out of the lecture notebook.

# Example dataset

In this lecture, we will primarily use data from the 2017-18 NBA season.

**Goal:** Predict whether or not a team will win, given their FG\_PCT\_DIFF.

- This is the difference in field goal percentage between the two teams.
- Positive FG\_PCT\_DIFF: team made more shots than the opposing team.

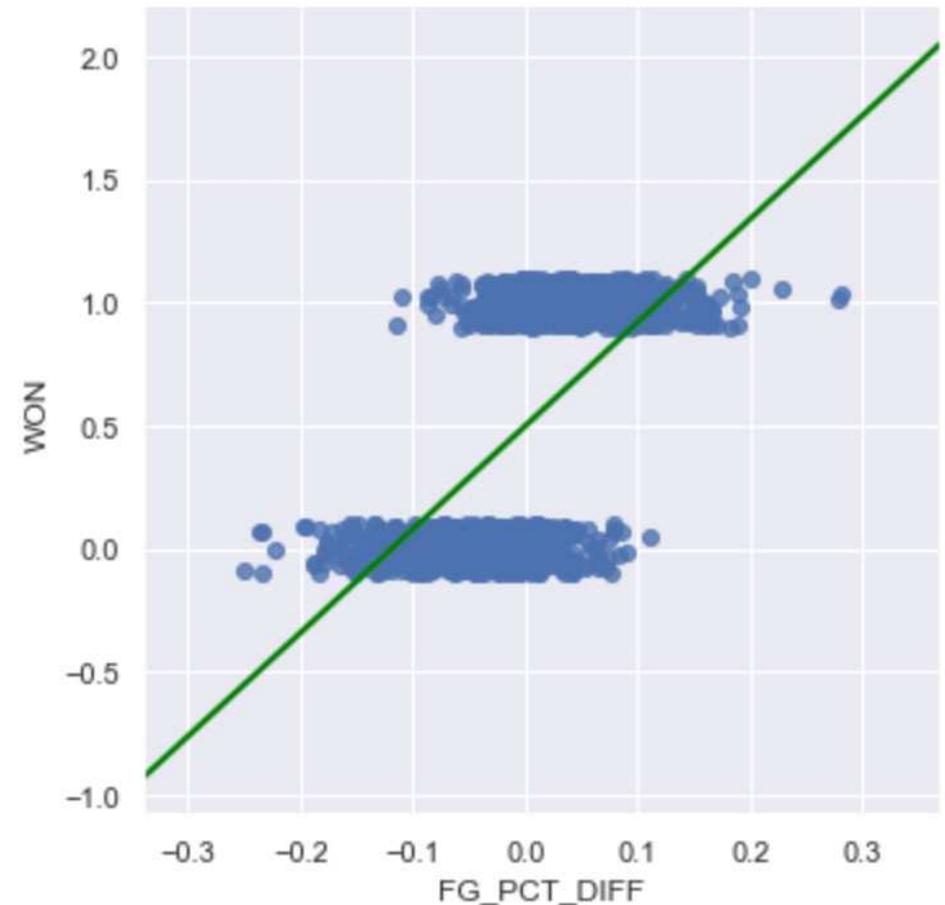
TEAM_NAME	MATCHUP	WON	FG_PCT_DIFF
Boston Celtics	BOS @ CLE	0	-0.049
Golden State Warriors	GSW vs. HOU	0	0.053
Charlotte Hornets	CHA @ DET	0	-0.030
Indiana Pacers	IND vs. BKN	1	0.041
Orlando Magic	ORL vs. MIA	1	0.042

1s represent wins,  
0s represent losses.

# Why not use Ordinary Least Squares?

We already have a model that can predict any quantitative response.  
Why not use it here?

- The output can be outside of the range [0, 1]. What does a predicted WON value of -2 mean?
- Very sensitive to outliers.
- Many other statistical reasons.
  - Not the point of our class.

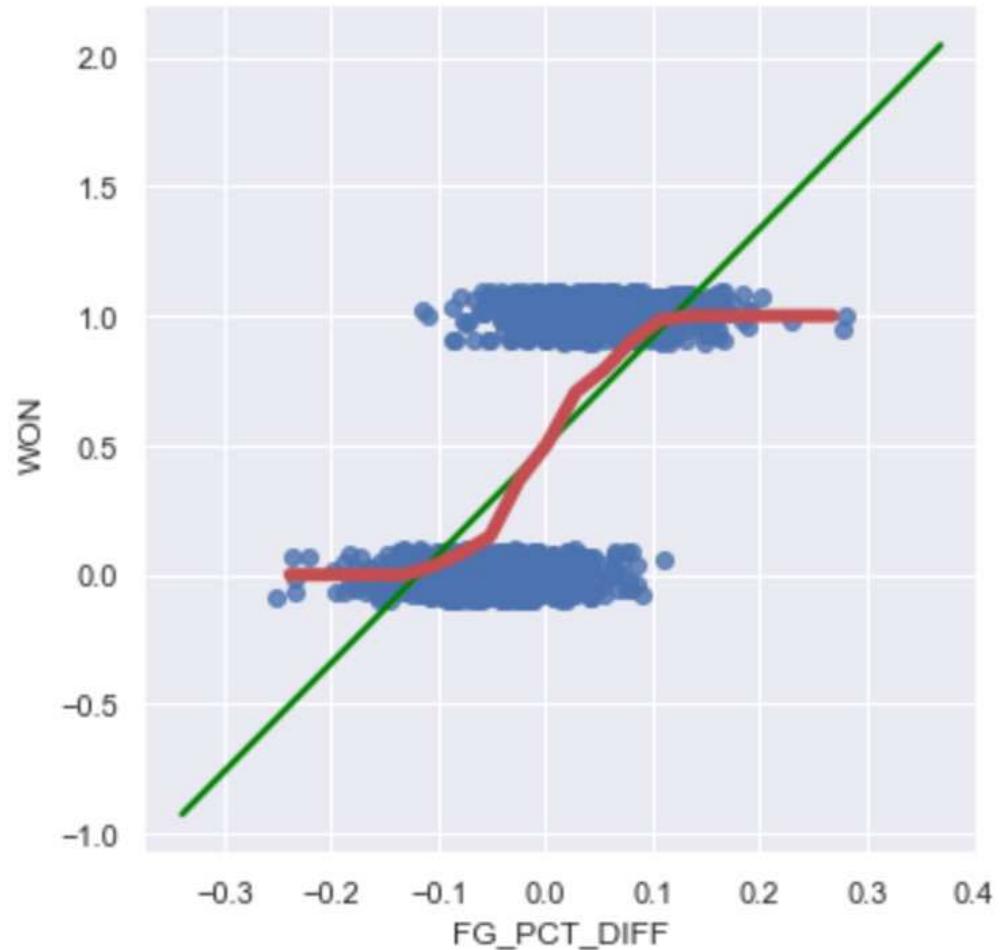


# Graph of averages

When defining the simple linear regression model, we binned the x-axis, and took the average y-value for each bin, and tried to model that.

Doing so here yields a curve that resembles an s.

- Since our true y is either 0 or 1, this curve models the **probability that WON = 1**, given FG\_PCT\_DIFF.
  - WON = 1 means “belong to class 1”.
- **Our goal is to model this red curve as best as possible.**



# Log-odds of probability is roughly linear

In the demo, we noticed that the **log-odds of the probability of belonging to class 1 was linear. This is the assumption that logistic regression is based on**

$$\text{odds}(p) = \frac{p}{1-p} \quad \text{log-odds}(p) = \log\left(\frac{p}{1-p}\right)$$

For now, let's let  $t$  denote our linear function (since log-odds is linear). Solving for  $p$ :

$$t = \log\left(\frac{p}{1-p}\right)$$

$$e^t = \frac{p}{1-p}$$

$$e^t - pe^t = p$$

$$p = \frac{e^t}{1 + e^t} = \frac{1}{1 + e^{-t}}$$

With logistic regression, we are always referring to log base e ("ln").

This is called the **logistic function**,  $\sigma(t)$ .

# Arriving at the logistic regression model

We know how to model linear functions quite well.

- We can substitute  $t = \mathbf{x}^T \boldsymbol{\theta}$ , since  $t$  was just a placeholder.

$p$  represents the probability of belonging to class 1.

$$p = \frac{1}{1 + e^{-t}} = \sigma(t)$$

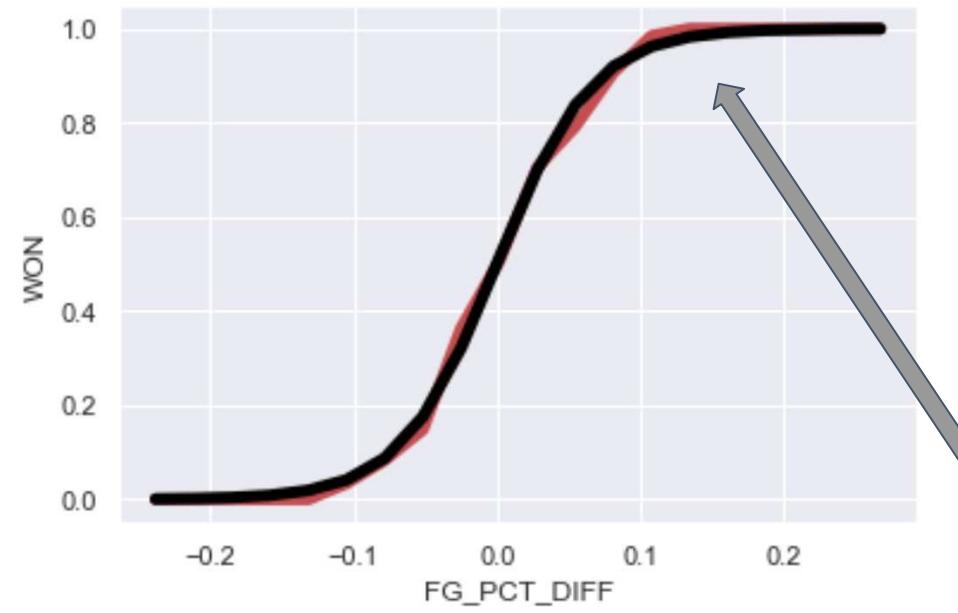
- We are modeling  $P(Y = 1|\mathbf{x})$ .

Putting this all together:

$$P(Y = 1|\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{x}^T \boldsymbol{\theta}}} = \sigma(\mathbf{x}^T \boldsymbol{\theta})$$

Looks just like the linear regression model, with a  $\sigma()$  wrapped around it. We call logistic regression a **generalized linear model**, since it is a non-linear transformation of a linear model.

# Arriving at the logistic regression model



## In red:

Empirical graph of averages

## In black:

$$\hat{y} = \sigma(30 \cdot \text{FG PCT DIFF})$$

# Logistic regression

# Linear vs. logistic regression

In a **linear regression** model, we predict a **quantitative** variable (i.e., some real number) as a linear function of features.

- Our output, or **response**,  $y$ , could be any real number.

$$\hat{y} = f_{\theta}(x) = x^T \theta$$

In a **logistic regression** model, our goal is to predict a binary **categorical** variable (class 0 or class 1) as a linear function of features, passed through the logistic function.

- Our **response** is the probability that our observation belongs to class 1.
- Haven't yet done classification!

$$\hat{y} = f_{\theta}(x) = P(Y = 1|x) = \sigma(x^T \theta)$$

# Example calculation

Suppose I want to predict the probability that LeBron's shot goes in, given **shot distance** (first feature) and **# of seconds left on the shot clock** (second feature).

I fit a logistic regression model using my training data, and somehow compute

$$\hat{\theta}^T = [0.1 \quad -0.5]$$



Under the logistic model, compute the probability his shot goes in, given that

- He shoots it from 15 feet.
- There is 1 second left on the shot clock.

# Example calculation (solution)

$$x^T = [15 \quad 1] \quad \hat{\theta}^T = [0.1 \quad -0.5]$$

$$\begin{aligned} P(Y = 1|x) &= \sigma(x^T \hat{\theta}) \\ &= \sigma(\hat{\theta}_1 \cdot \text{SHOT DISTANCE} + \hat{\theta}_2 \cdot \text{SECONDS LEFT}) \\ &= \sigma(0.1 \cdot 15 + (-0.5) \cdot 1) \\ &= \sigma(1) \\ &= \frac{1}{1 + e^{-1}} \\ &\approx 0.7311 \end{aligned}$$

An explicit expression representing our model.

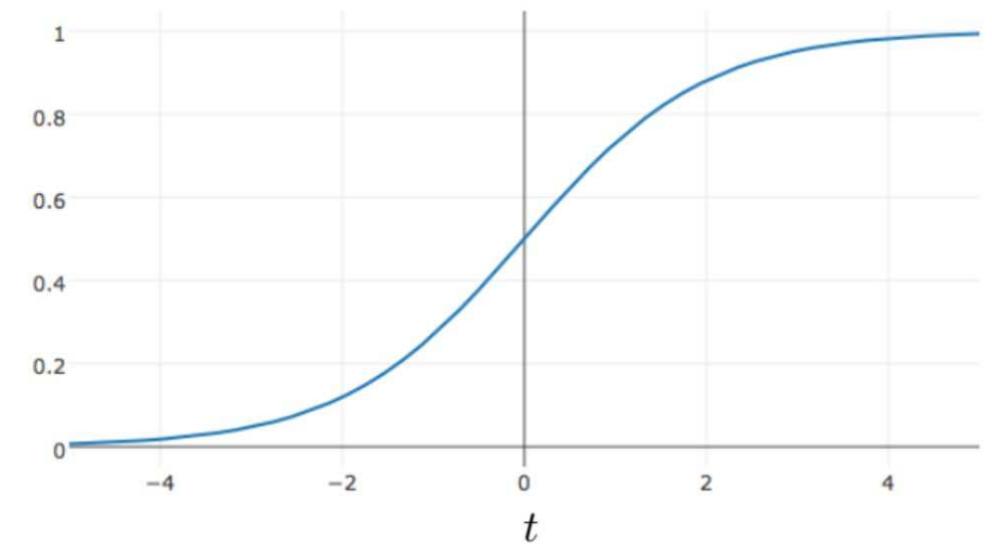


# Properties of the logistic function

The logistic function is a type of **sigmoid**, a class of functions that share certain properties.

$$\sigma(t) = \frac{1}{1 + e^{-t}} \quad -\infty < t < \infty$$

- Its output is bounded between 0 and 1, no matter how large  $t$  is.
  - Fixes an issue with using linear regression to predict probabilities.
- We can interpret it as mapping real numbers to probabilities.



# Properties of the logistic function

**Definition**

$$\sigma(t) = \frac{1}{1 + e^{-t}} = \frac{e^t}{1 + e^t}$$

**Range**

$$0 < \sigma(t) < 1$$

**Inverse**

$$t = \sigma^{-1}(p) = \log\left(\frac{p}{1-p}\right)$$

**Reflection and Symmetry**

$$1 - \sigma(t) = \frac{e^{-t}}{1 + e^{-t}} = \sigma(-t)$$

**Derivative**

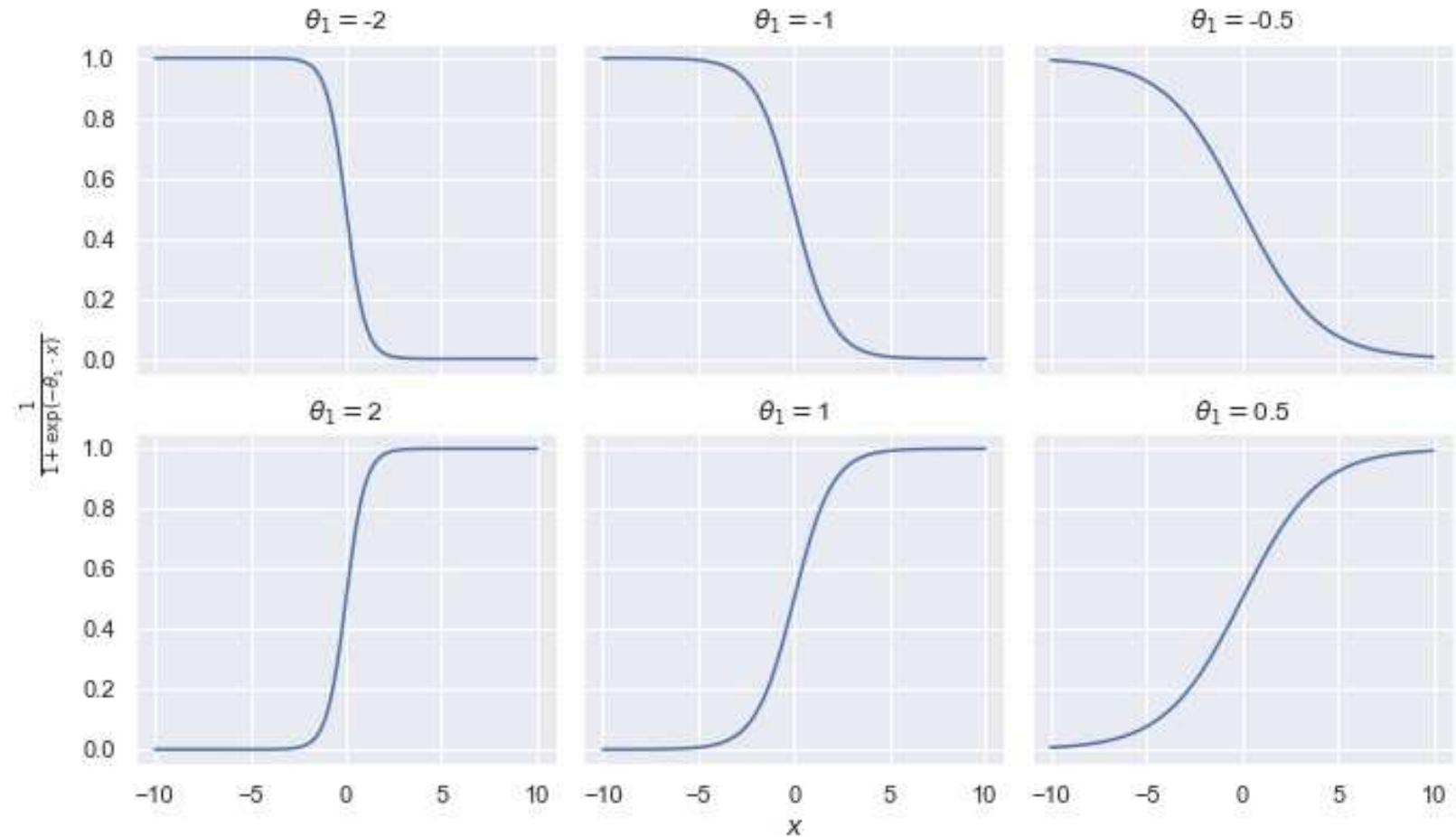
$$\frac{d}{dt} \sigma(t) = \sigma(t)(1 - \sigma(t)) = \sigma(t)\sigma(-t)$$

# Shape of the logistic function

Consider the plot of  $\sigma(\theta_1 x)$ , for several different values of  $\theta_1$ .

- If  $\theta_1$  is positive, the curve increases to the right.
- The further  $\theta_1$  is from 0, the steeper the curve.

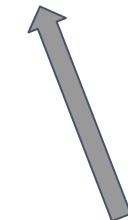
In the notebook, we explore more sophisticated logistic curves.



# Parameter interpretation

Recall, we arrived at the model by assuming that the log-odds of the probability of belonging to class 1 was linear.

$$P(Y = 1|x) = \sigma(x^T \theta) \quad \leftarrow \quad \log\left(\frac{P(Y = 1|x)}{P(Y = 0|x)}\right) = x^T \theta \quad \leftarrow \quad \frac{P(Y = 1|x)}{P(Y = 0|x)} = e^{x^T \theta}$$



This is the same as  $\frac{p}{1-p}$ , because  
 $P(Y = 1|x) + P(Y = 0|x) = 1$

(Remember, we are dealing with binary classification – we are predicting 1 or 0.)

# Parameter interpretation

Let's suppose our linear component has just a single feature, along with an intercept term.

$$\frac{P(Y = 1|x)}{P(Y = 0|x)} = e^{\theta_0 + \theta_1 x}$$

What happens if you increase  $x$  by one unit?

- Odds is multiplied by  $e^{\theta_1}$ .
- If  $\theta_1 > 0$ , the odds increase.
- If  $\theta_1 < 0$ , the odds decrease.

What happens if  $x^T \theta = \theta_0 + \theta_1 x = 0$  ?

- This means class 1 and class 0 are equally likely.

$$e^0 = 1 \implies \frac{P(Y = 1|x)}{P(Y = 0|x)} = 1 \implies P(Y = 1|x) = P(Y = 0|x)$$

The odds ratio can be interpreted as the “number of successes for each failure.”

# Today's class

- Logistic regression – Part 2

# Logistic regression with squared loss

# Logistic regression with squared loss

To find  $\hat{\theta}$  so that we can make predictions, we need to choose a loss function.

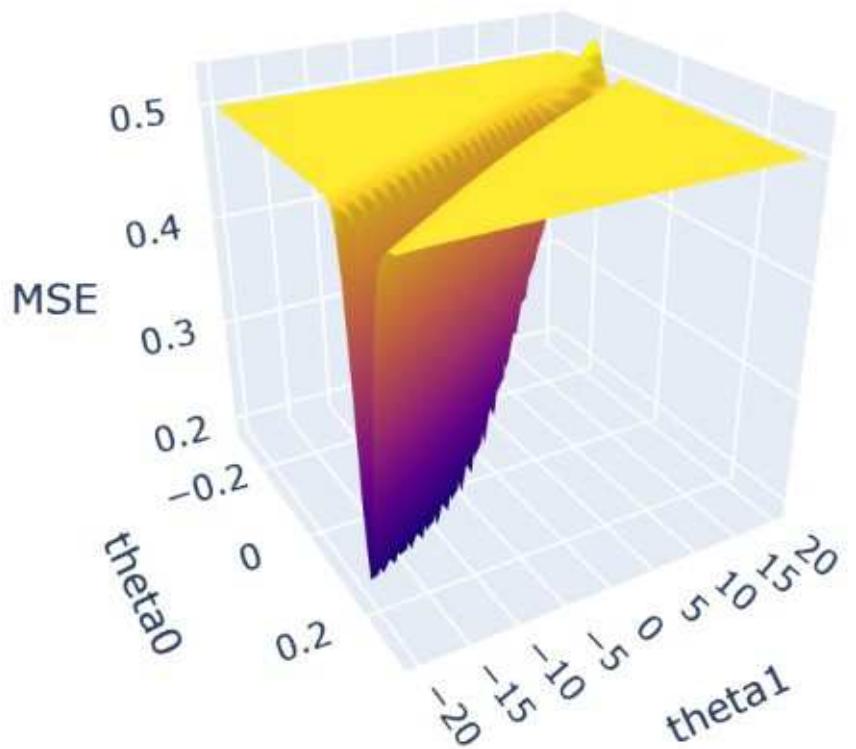
- We can start with our old friend, squared loss.
- Doing so yields the following MSE:

$$R(\theta) = \frac{1}{n} \sum_{i=1}^n (y_i - \sigma(\mathbb{X}_i^T \theta))^2$$

Sometimes, this works fine (and it is actually still used in some applications). Other times...

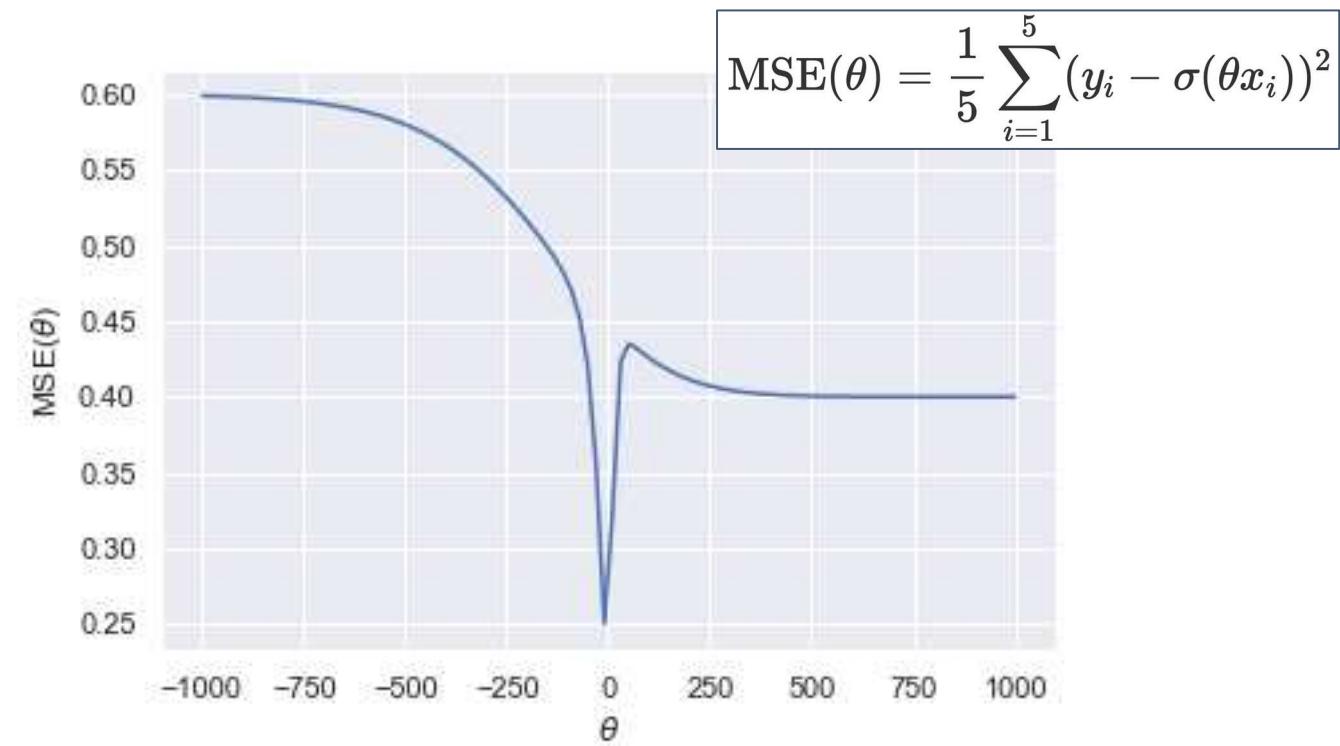
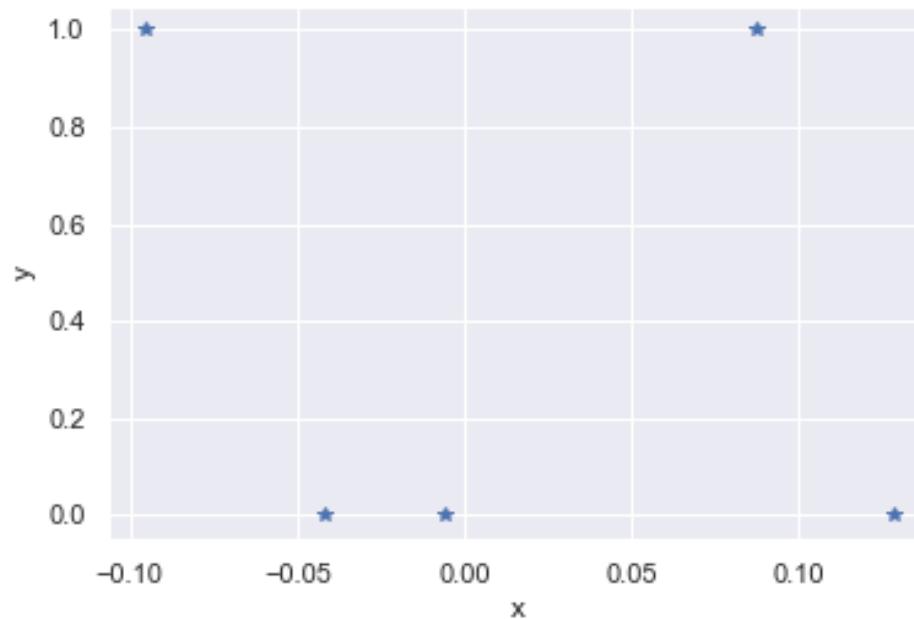
# Pitfalls of squared loss with logistic regression

The loss surface of MSE for a logistic regression model with a single slope plus an intercept often looks something like this.



# Pitfalls of squared loss with logistic regression

On the [left](#), we have a toy dataset (i.e. we've plotted the [original data](#),  $y$  vs.  $x$ ). On the [right](#), we have a plot of the [mean squared error](#) of this dataset when fitting a single-feature logistic regression model, for different values  $\theta$  (i.e. the [loss surface](#)).



# Pitfalls of squared loss with logistic regression

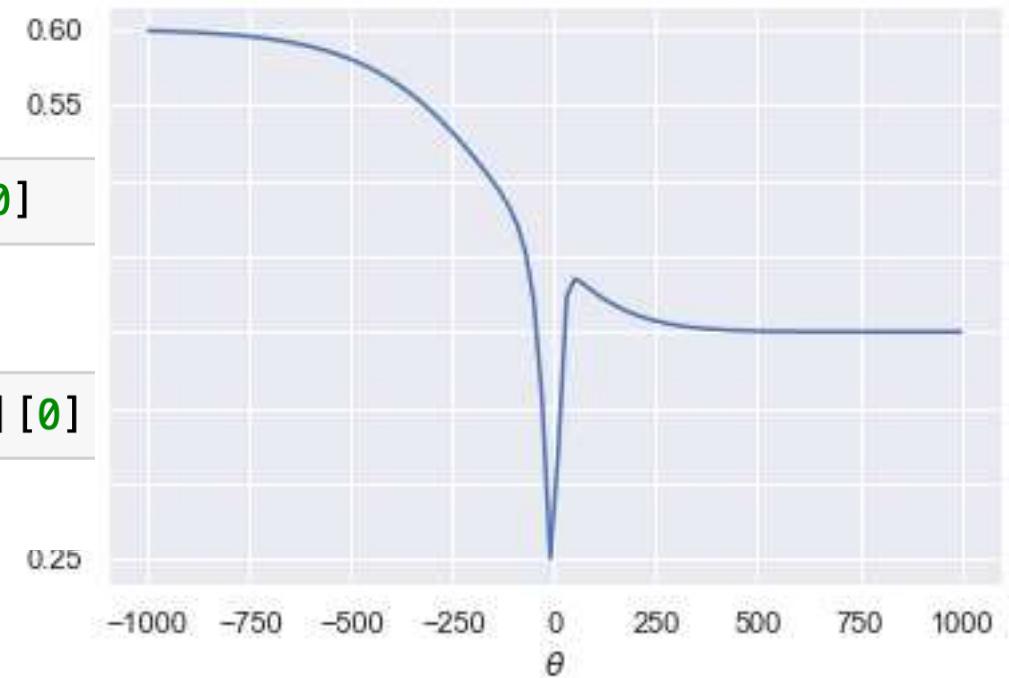
For this particular loss surface,  
different initial guesses for  $\theta$  yield different “optimal values”, as per `scipy.optimize.minimize`:

```
1 minimize(mse_loss_single_arg_toy, x0 = 0) ["x"] [0]
```

-4.801981341432673

```
1 minimize(mse_loss_single_arg_toy, x0 = 500) ["x"] [0]
```

500.0



This loss surface is not convex.  
We'd like it to be convex.

# Pitfalls of squared loss with logistic regression

Another issue: since  $y_i$  is either 0 or 1, and  $\hat{y}_i$  is between 0 and 1,  $(y_i - \hat{y}_i)^2$  is also bounded between 0 and 1.

- Even if our probability is nowhere close, the loss isn't that large in magnitude.
  - If we say the probability is  $10^{-6}$ , but it happens anyway, error should be large.
- We want to penalize wrong answers significantly.

# Summary of issues with squared loss and logistic regression

While it can work, squared loss is not the best choice of loss function for logistic regression.

- Average squared loss is not nice (non-convex).
  - Numerical methods will struggle to find a solution.
- Wrong predictions aren't penalized significantly enough.
  - Squared loss (and hence, average squared loss) are bounded between 0 and 1.

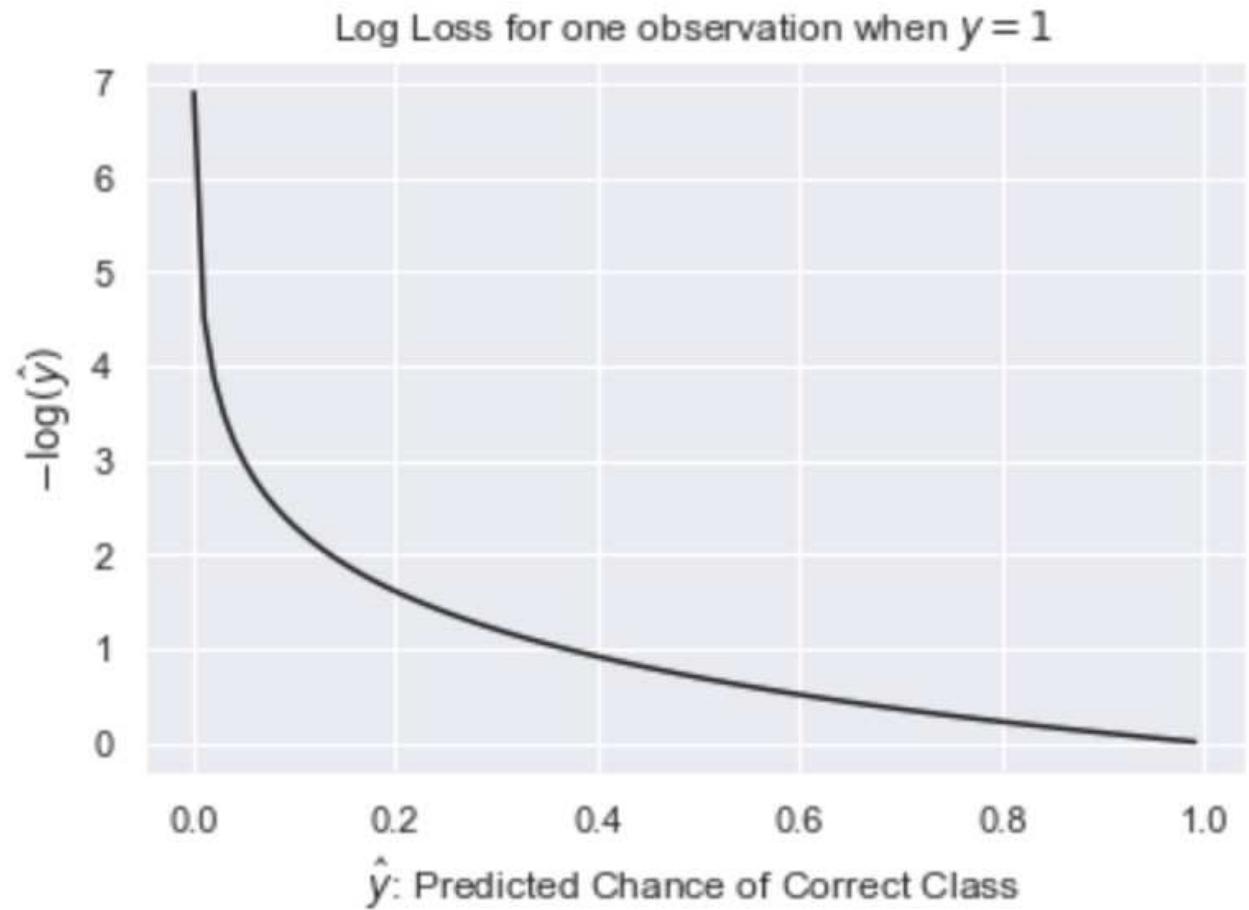
Fortunately, there's a solution.

# Cross-entropy loss

# Log loss

Consider this new loss, called the (negative) **log loss**, for a single observation when the true  $y$  is equal to 1.

We can see that as our prediction gets further and further from 1, the loss approaches infinity (unlike squared loss, which maxed out at 1).

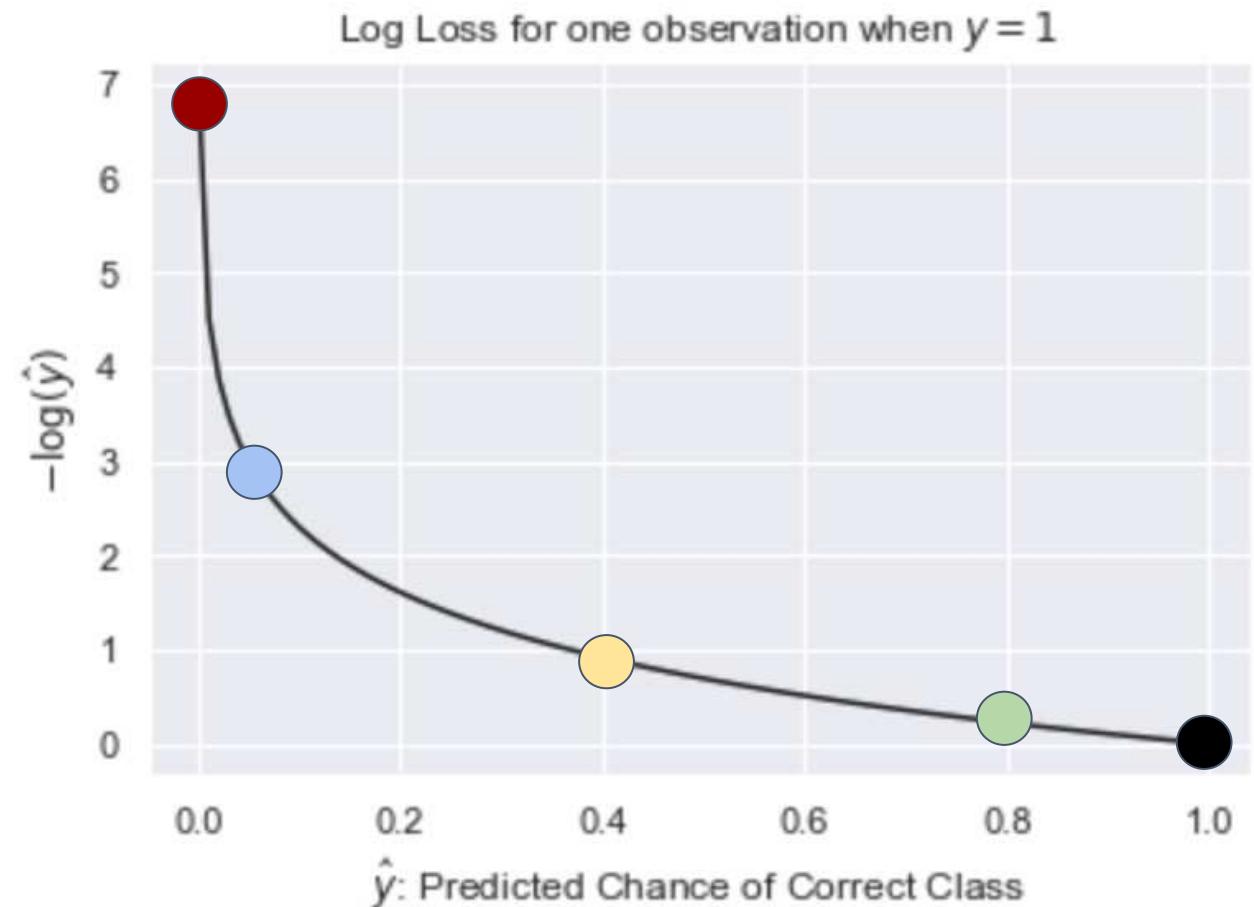


# Log loss

Let's look at some losses in particular:

$\hat{y}$	$-\log(\hat{y})$
1	0
0.8	0.25
0.4	1
0.05	3
0	infinity!

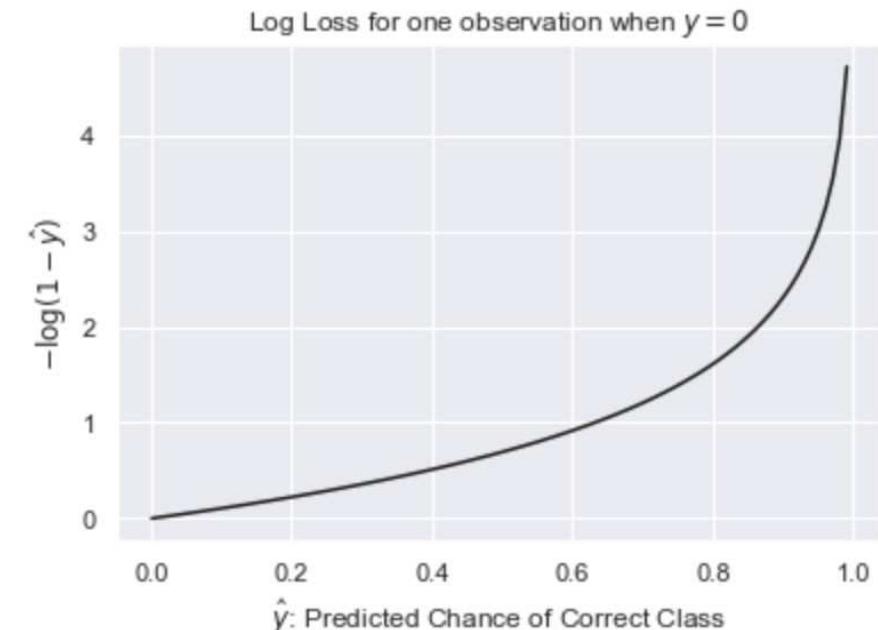
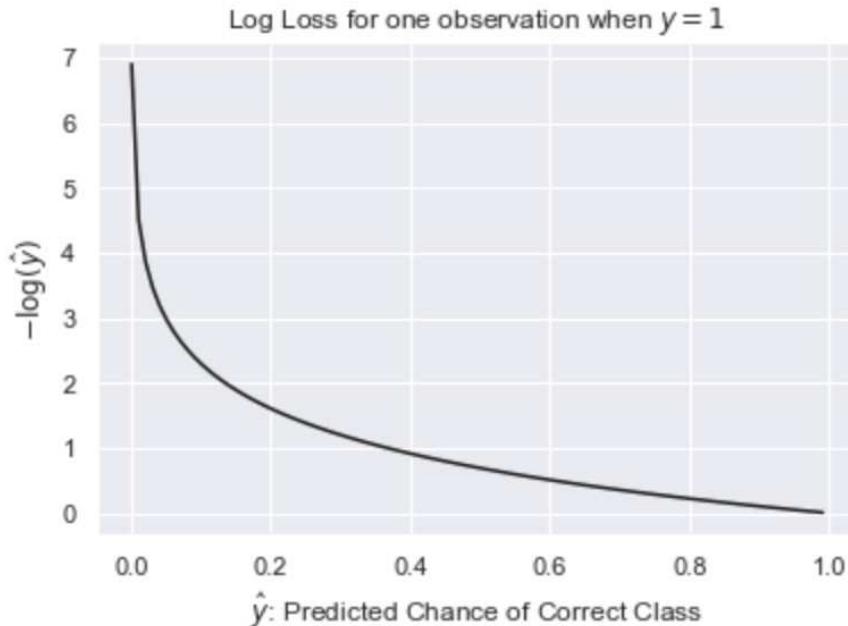
**Note:** The logistic function never outputs 0 or 1 exactly, so there's never actually 0 loss or infinite loss.



# Log loss

So far, we've only looked at log loss when the correct class was 1.

**What if our correct class is 0?**



If the correct class is 0, we want to have low loss for values of  $\hat{y}$  close to 0, and high loss for values of  $\hat{y}$  close to 1. [This is achieved by just “flipping” the plot on the left!](#)

# Cross-entropy loss

We can combine the two cases from the previous slide into a single loss function:

$$\text{loss} = \begin{cases} -\log(1 - \hat{y}) & y = 0 \\ -\log(\hat{y}) & y = 1 \end{cases}$$

This is often written unconditionally as:

$$\text{loss} = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y})$$

*Note: Since  $y = 0$  or  $1$ , one of these two terms is always equal to  $0$ , which reduces this equation to the piecewise one above.*

We call this loss function **cross-entropy** loss (or “log loss”).

# Mean cross-entropy loss

The empirical risk for the logistic regression model when using cross-entropy loss is then

$$R(\theta) = -\frac{1}{n} \sum_{i=1}^n (y_i \log(\sigma(\mathbb{X}_i^T \theta)) + (1 - y_i) \log(1 - \sigma(\mathbb{X}_i^T \theta)))$$

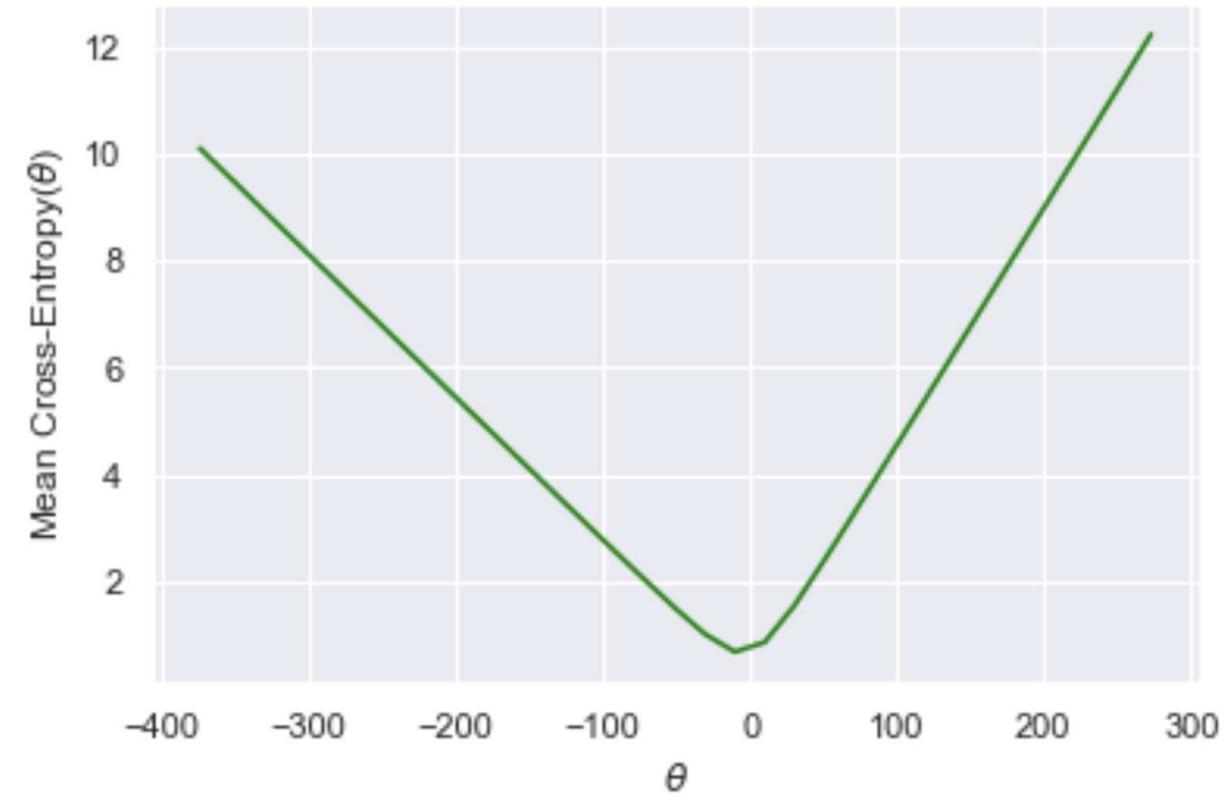
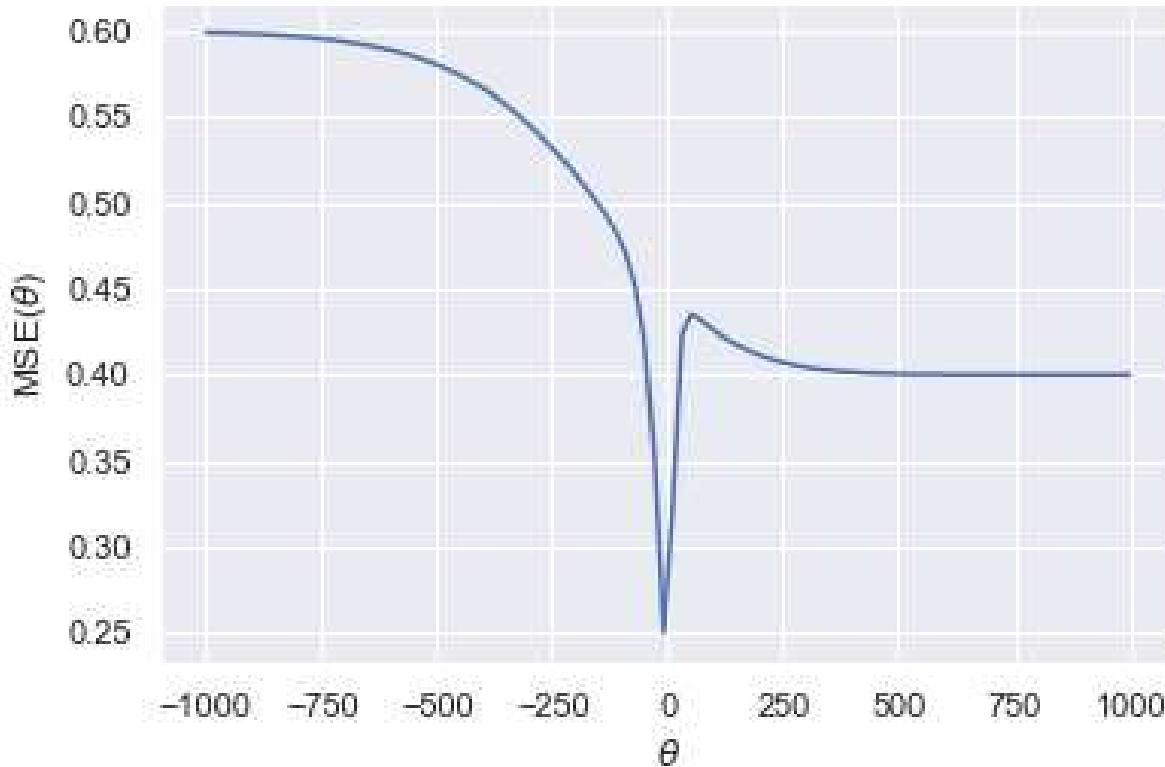
Benefits over mean squared error for logistic regression:

- Loss surface is guaranteed to be nice (convex).
- More strongly penalizes bad predictions.
- Has roots in probability and information theory

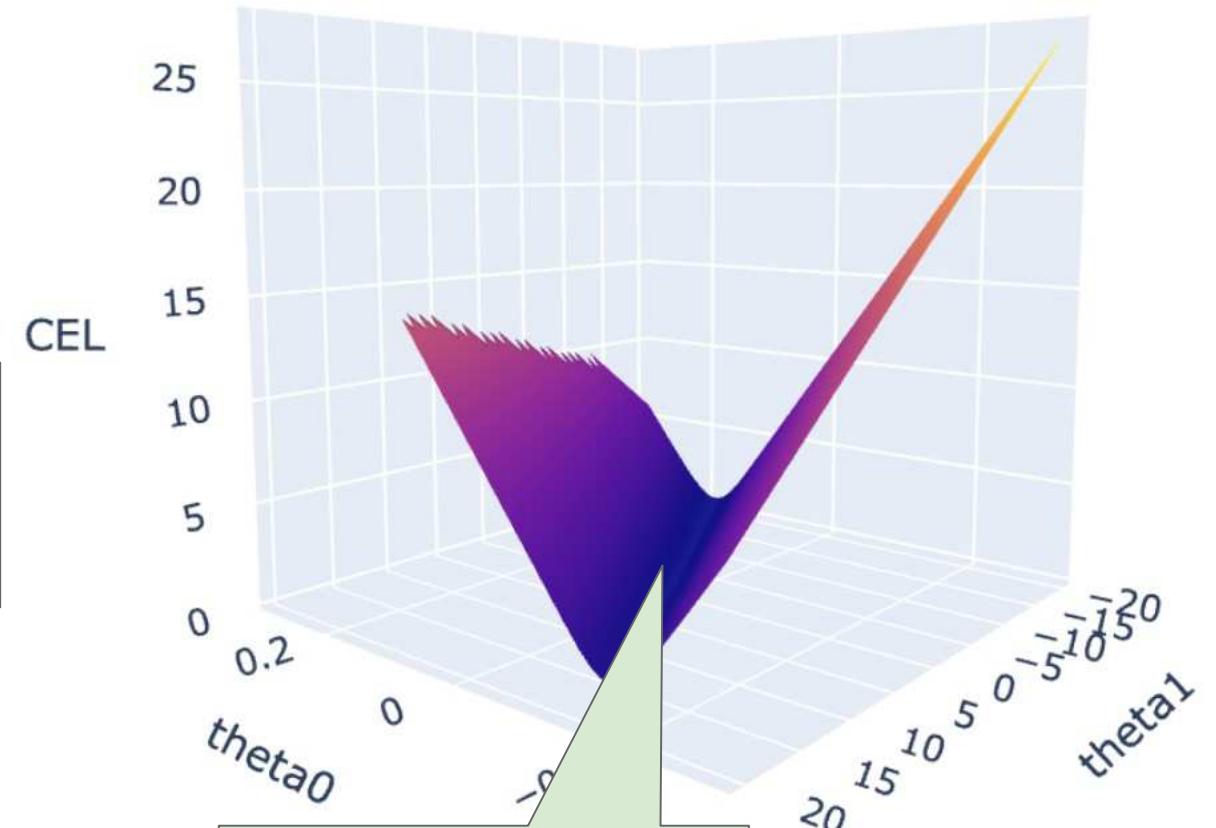
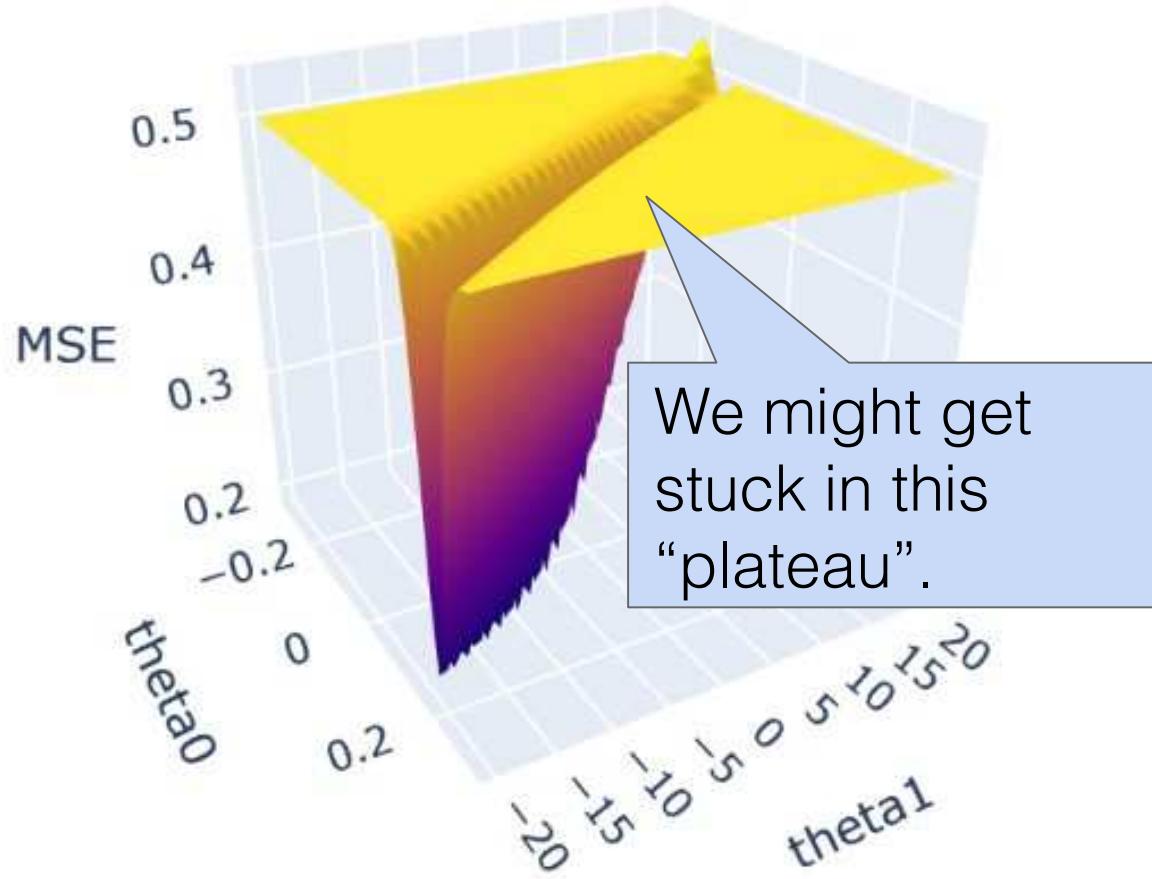
# Comparing loss surfaces

On the [left](#), we have a plot of the MSE loss surface on our toy dataset from before.

On the [right](#), we have a plot of the mean cross-entropy loss surface on the same dataset.



# Comparing loss surfaces



We will always  
end up in this  
“valley”.

# Summary

# Logistic regression

- In a **logistic regression** model, our goal is to predict a binary **categorical** variable (class 0 or class 1) as a linear function of features, passed through the logistic function.

- Our **response** is the probability that our observation belongs to class 1.

$$\hat{y} = f_{\theta}(x) = P(Y = 1|x) = \sigma(x^T \theta)$$

- We arrived at this model by assuming that the **log-odds of the probability of belonging to class 1 is linear**.
- To find  $\hat{\theta}$ , we can choose squared loss or cross-entropy loss.
  - Squared loss works, but is generally not a good idea.
  - Cross-entropy loss is much better (convex, better suited for modeling probabilities).

**YaleNUSCollege**

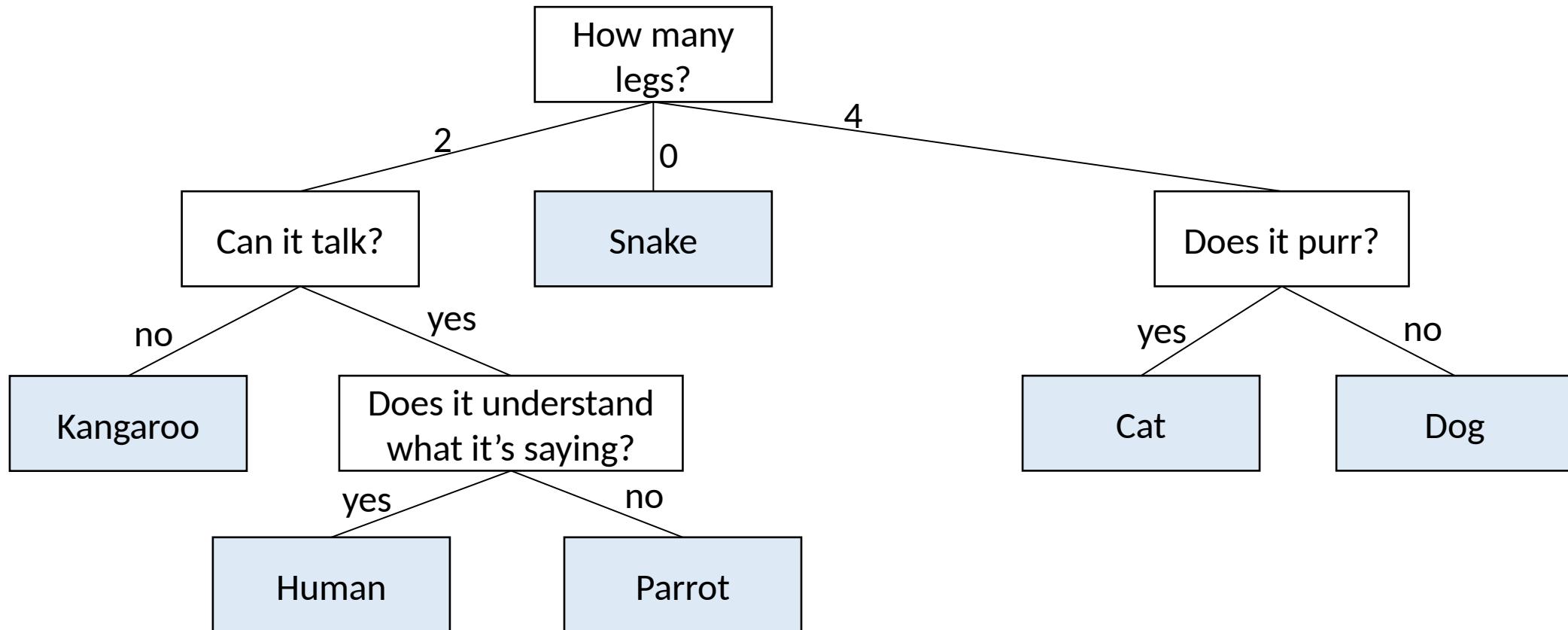
**YSC2239 Lecture 21**

# Today's class

- Decision Trees

# Decision Trees

A Decision Tree is a very simple way to classify data. It is simply a tree of questions that must be answered in sequence to yield a predicted classification.

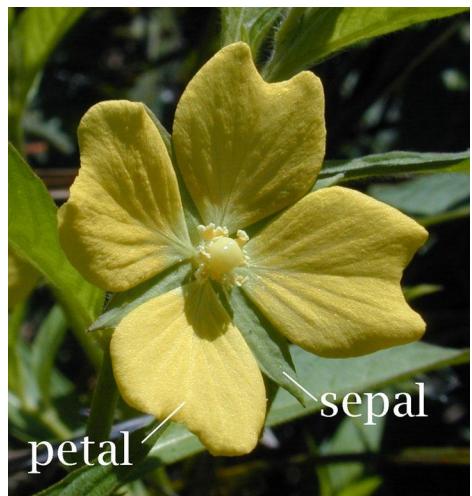


# Example: Flower Classification

The [Iris flower data set](#) is a commonly used example:

- Created by statistician/biologist Ronald Fisher for his paper “The use of multiple measurements in taxonomic problems”.
- Data set consists of 150 flower measurements from 3 different species.
- For each, we have “petal length”, “petal width”, “sepal length”, “sepal width”.

Goal is to predict species from other data.



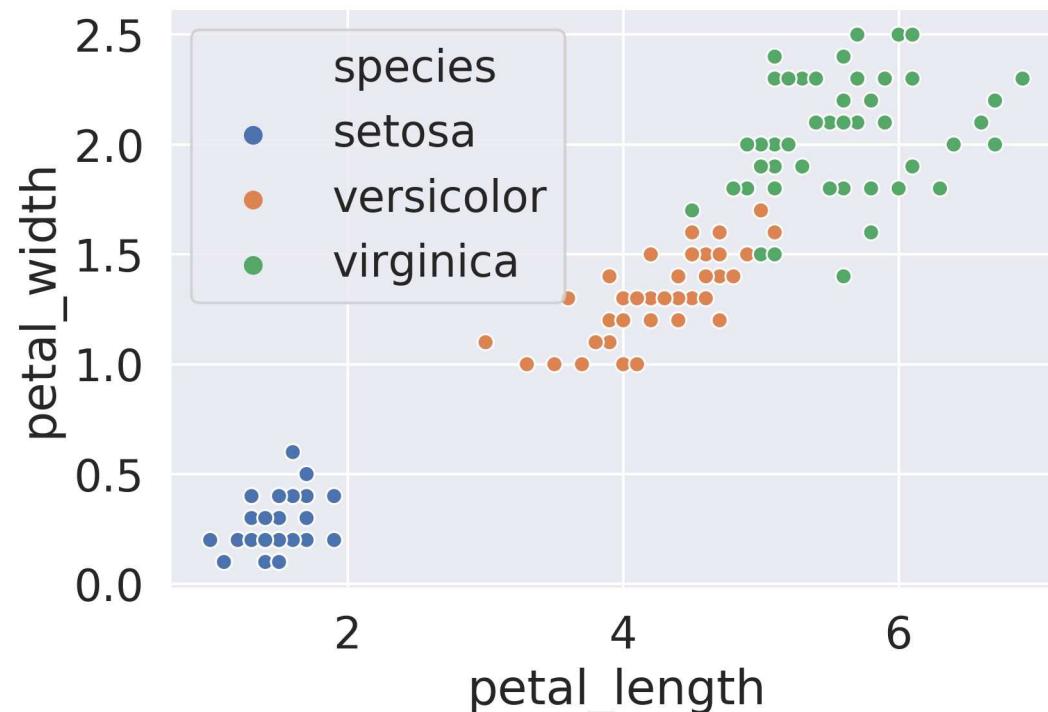
sepal_length	sepal_width	petal_length	petal_width	species
5.5	2.5	4.0	1.3	versicolor
6.4	2.9	4.3	1.3	versicolor
4.8	3.4	1.6	0.2	setosa
5.3	3.7	1.5	0.2	setosa
6.7	2.5	5.8	1.8	virginica

# Decision Tree Basics

## Example: Using Petal Data Only

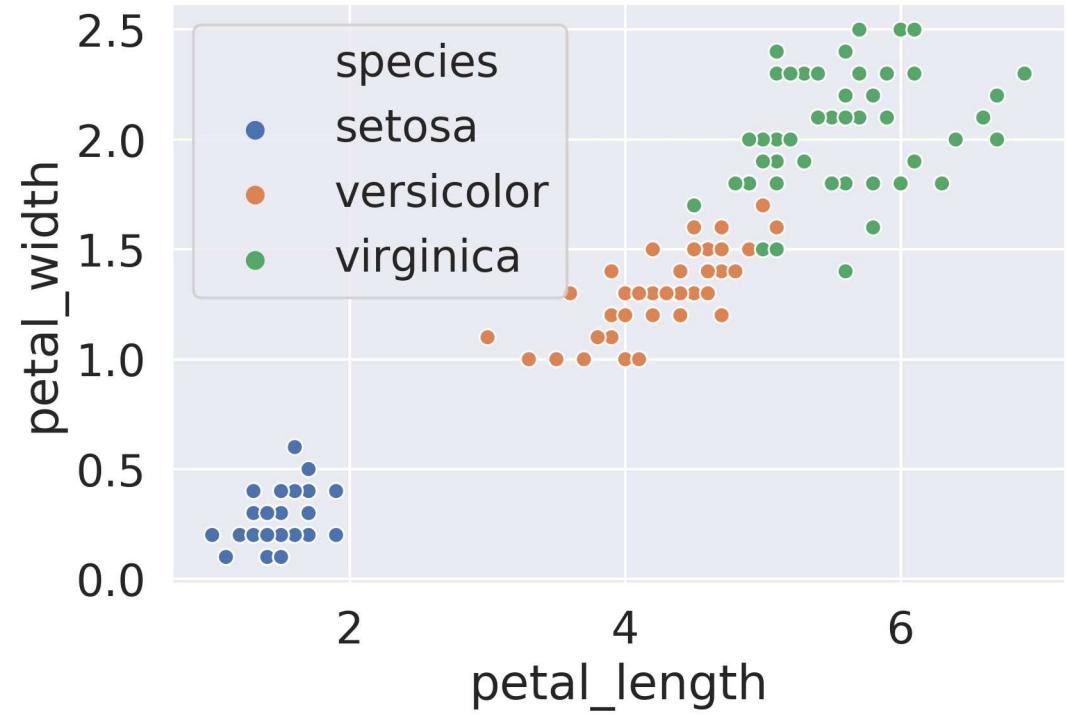
The plot below shows the width and length of the petals of each flower, with the species annotated in the form of color.

We can build a decision tree manually just by looking at this picture.

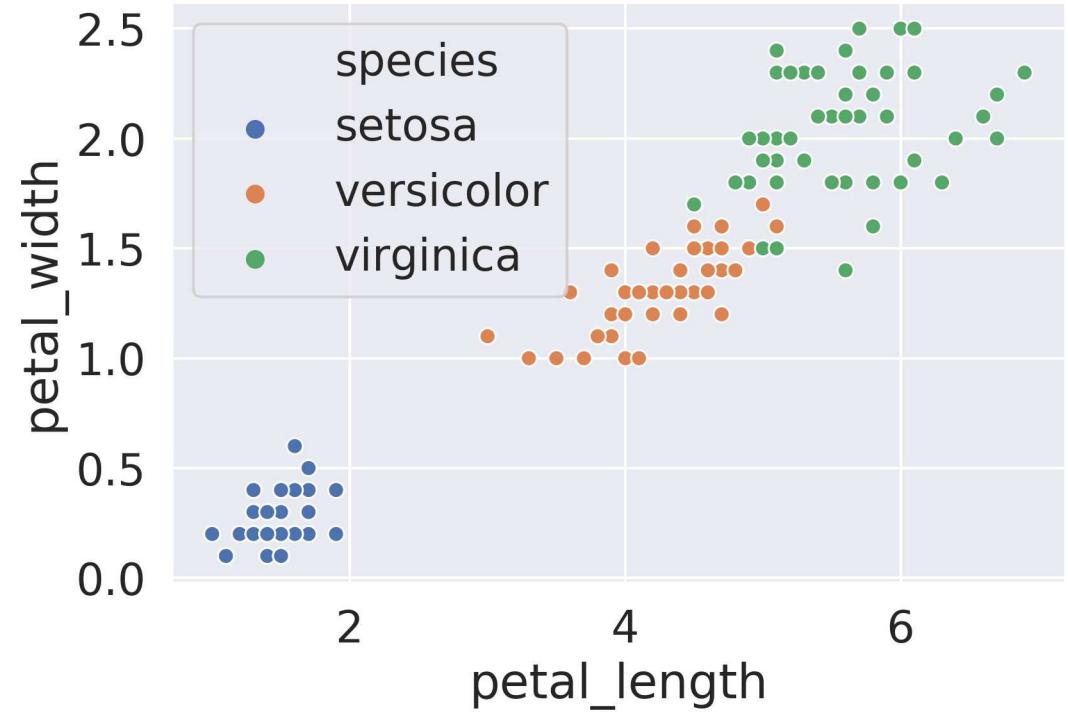
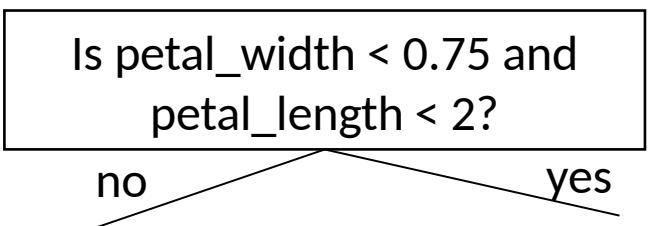


## Example: Using Petal Data Only

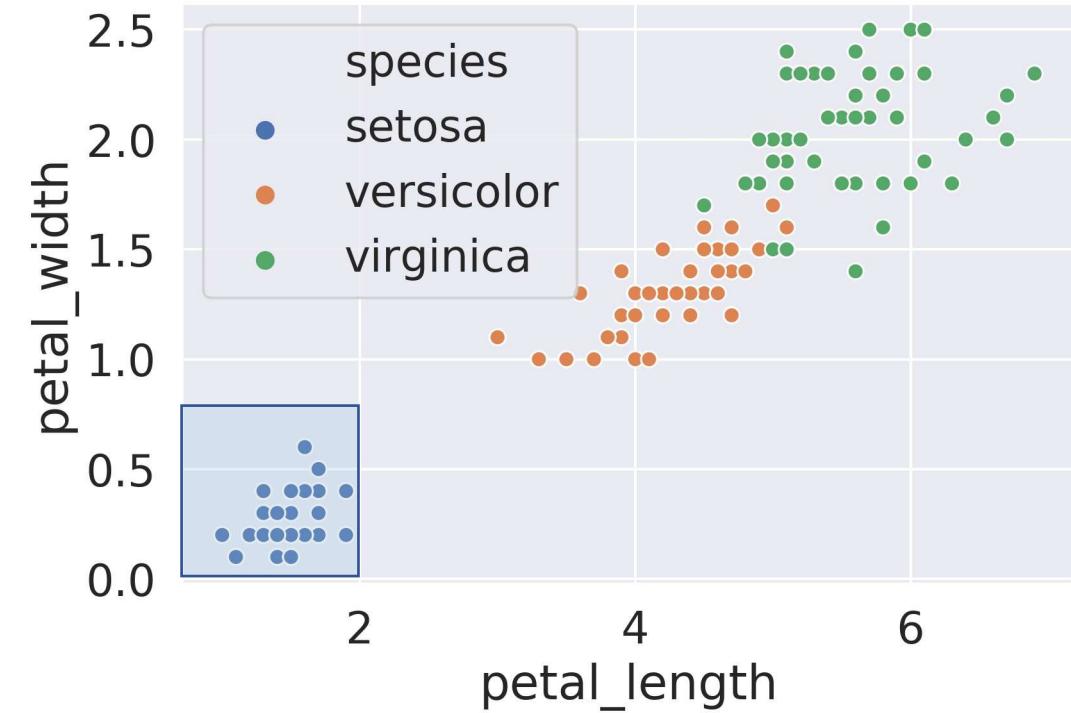
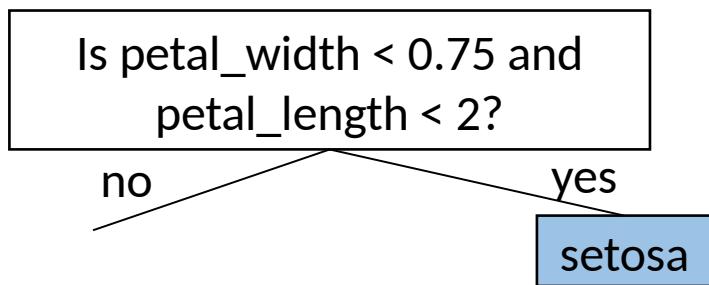
---



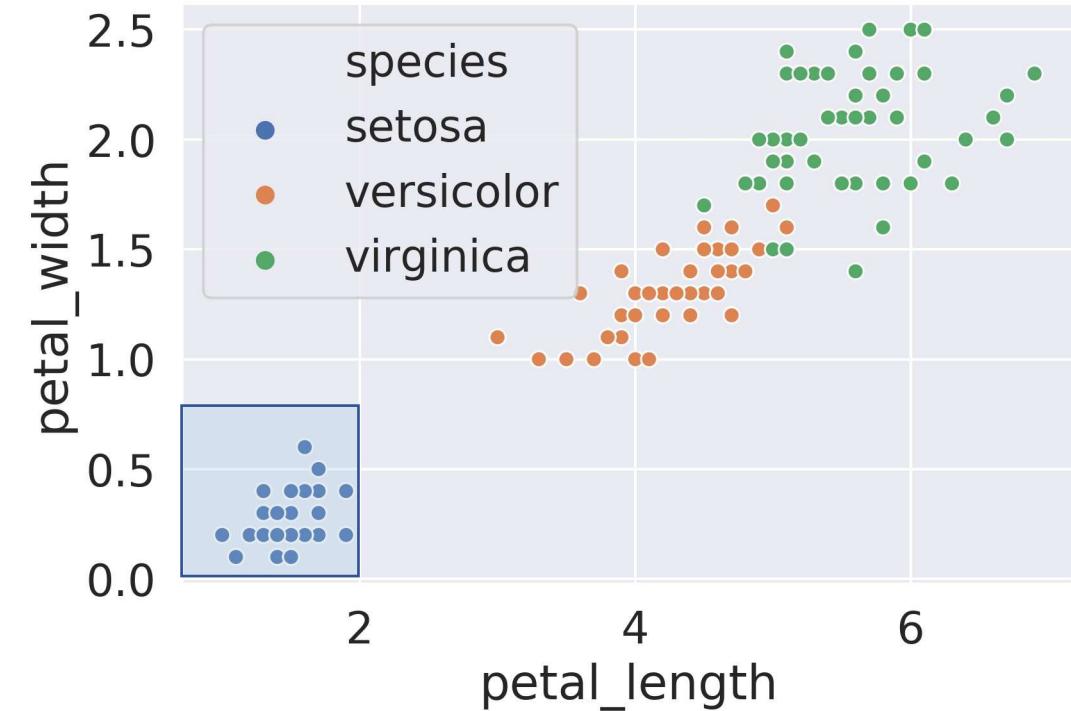
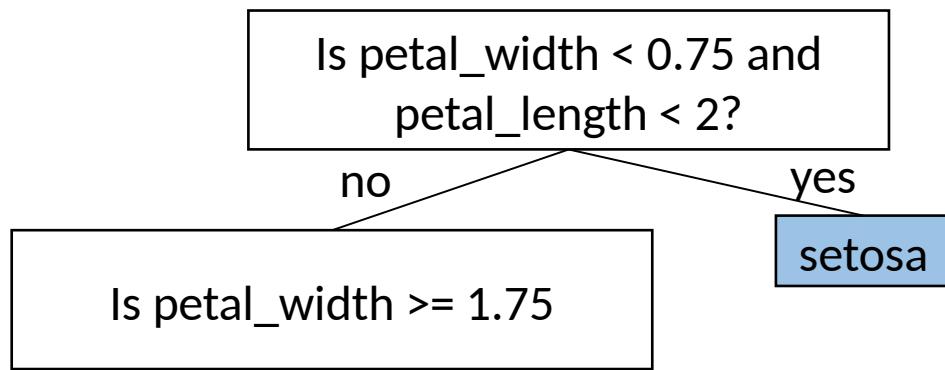
# Example: Using Petal Data Only



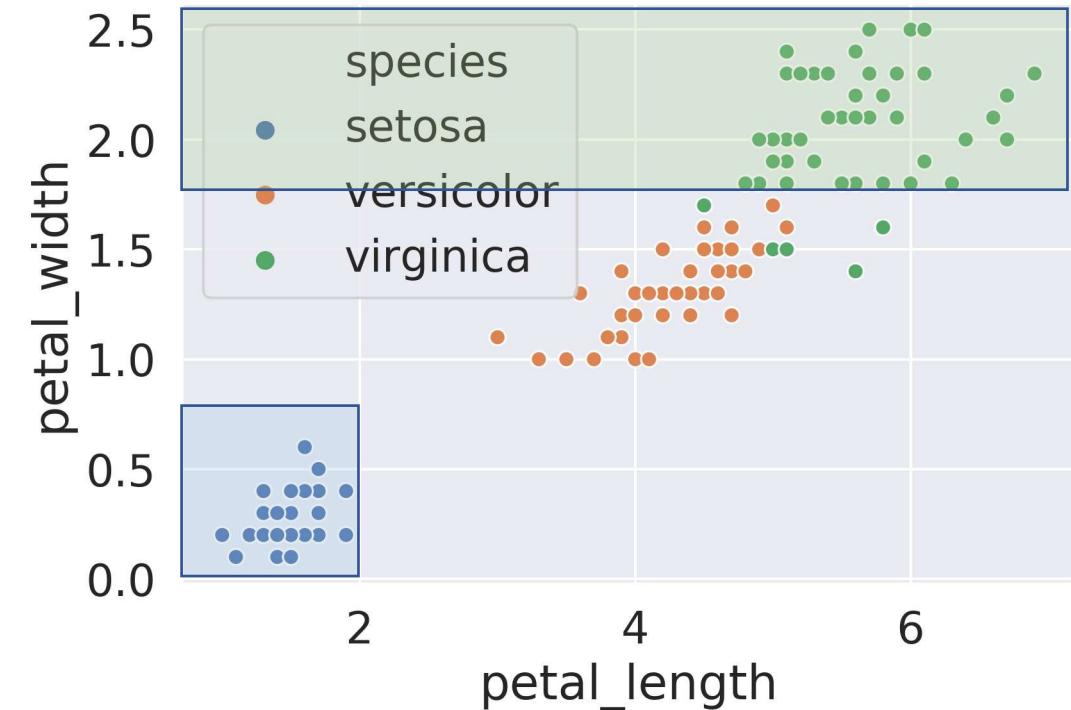
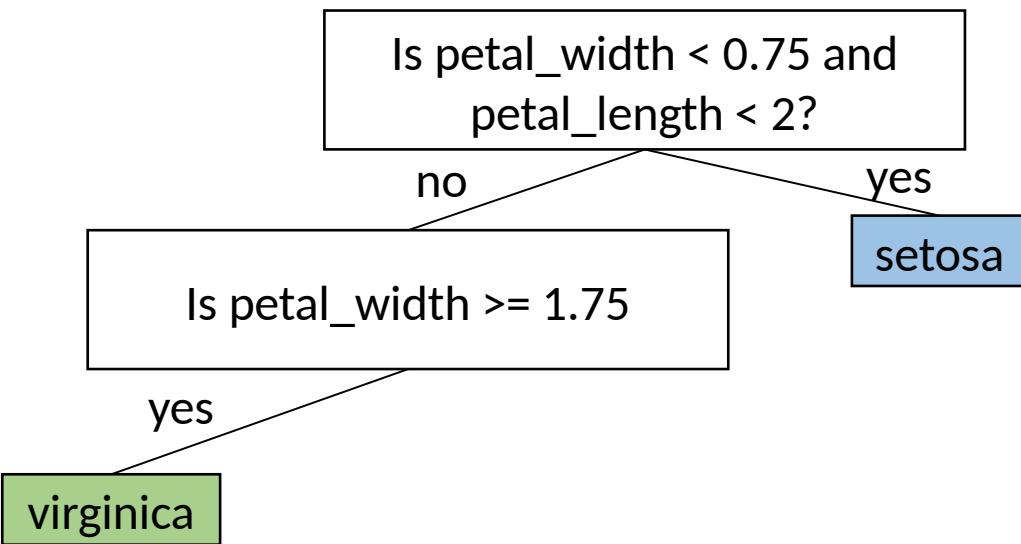
# Example: Using Petal Data Only



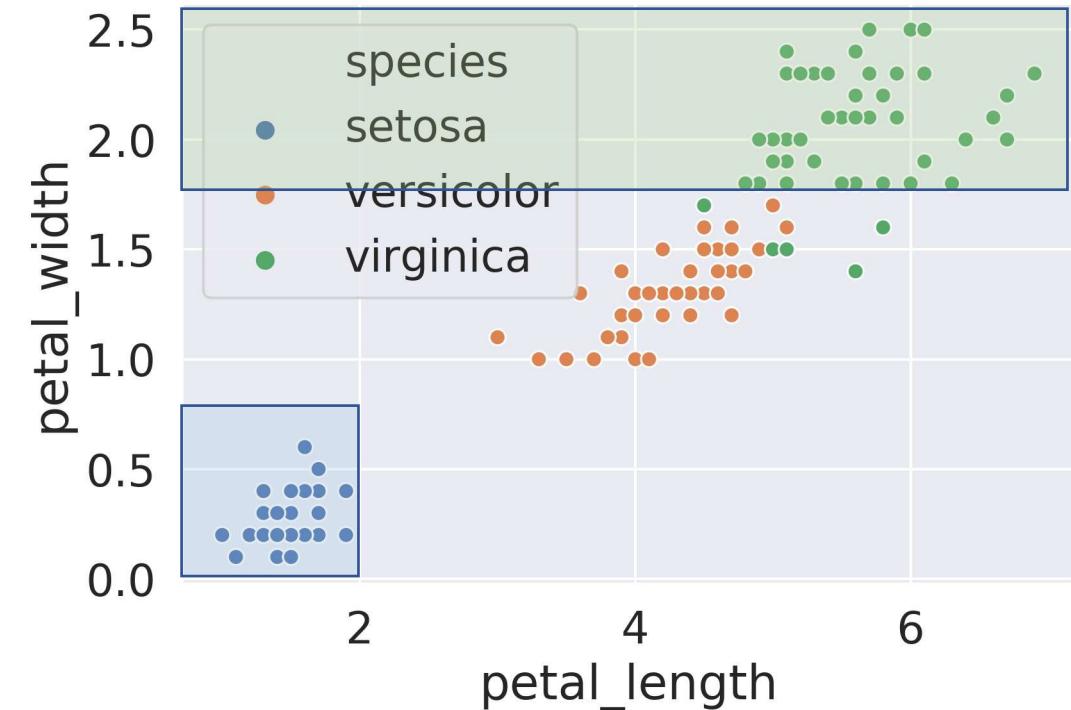
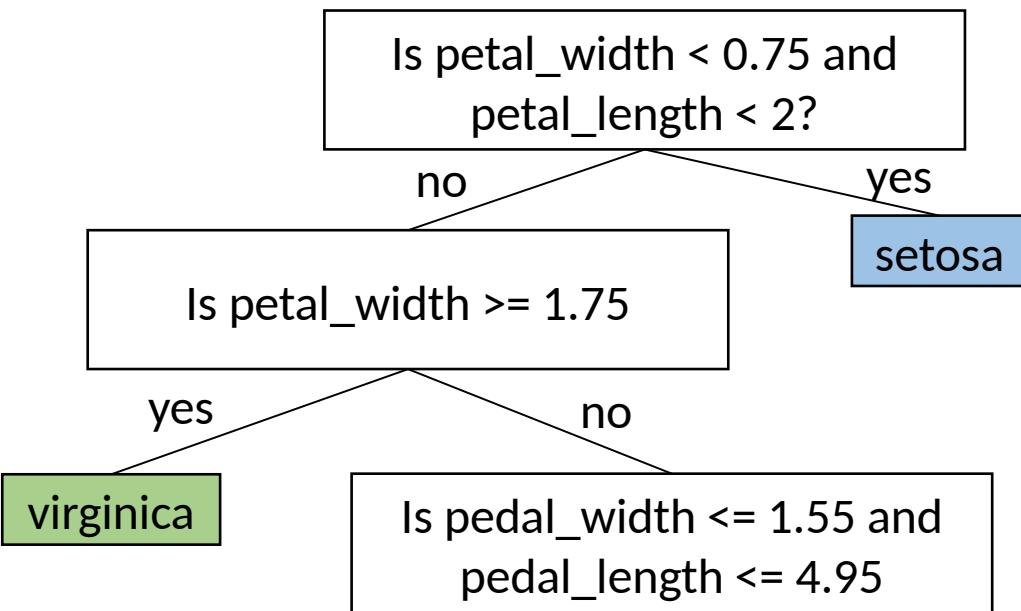
# Example: Using Petal Data Only



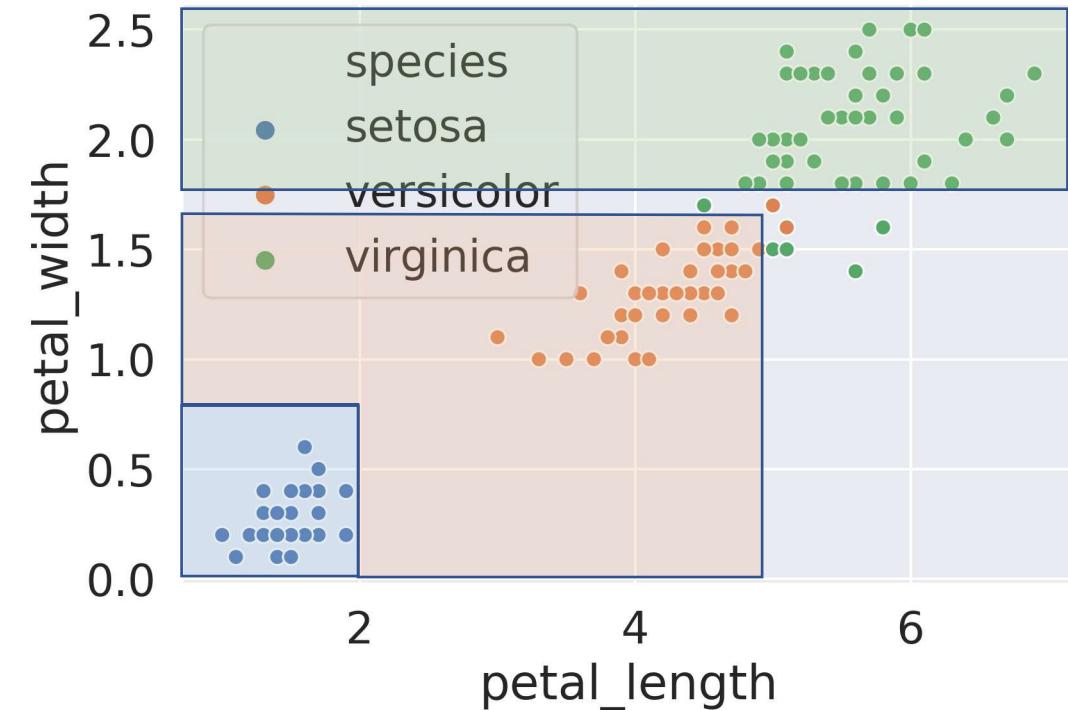
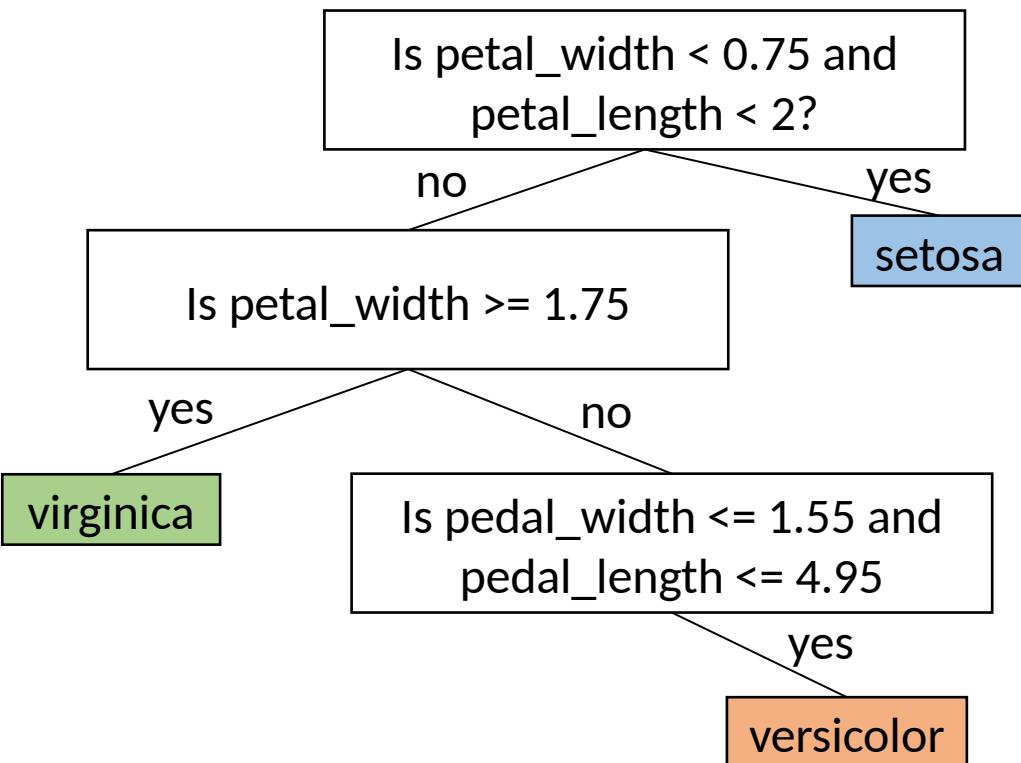
# Example: Using Petal Data Only



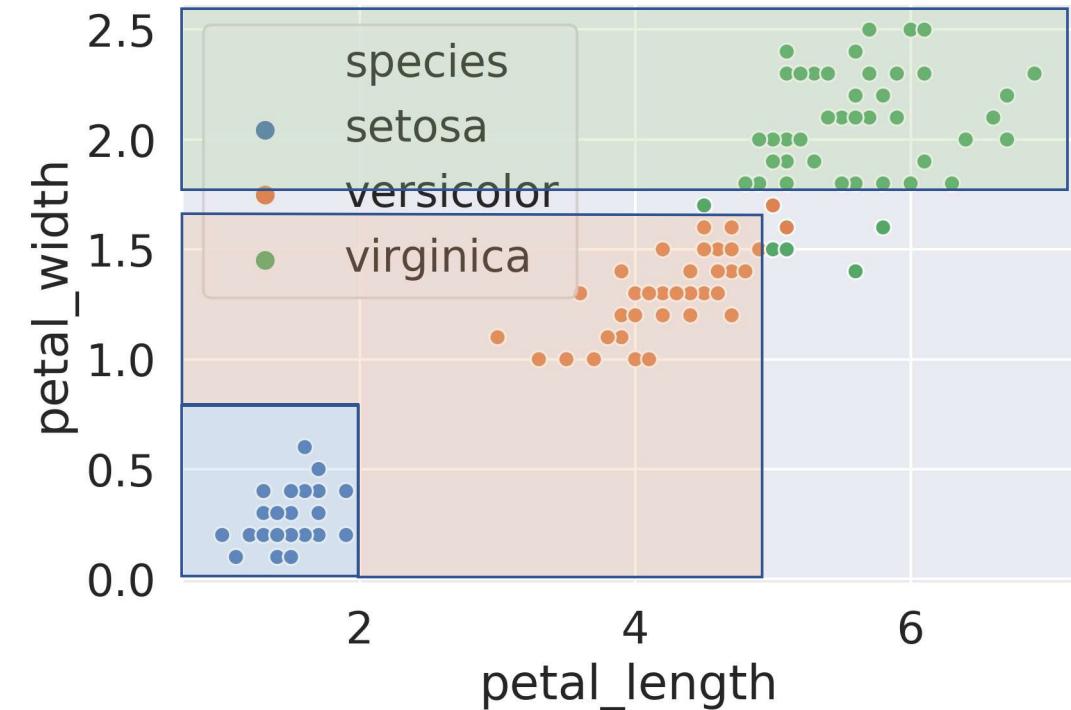
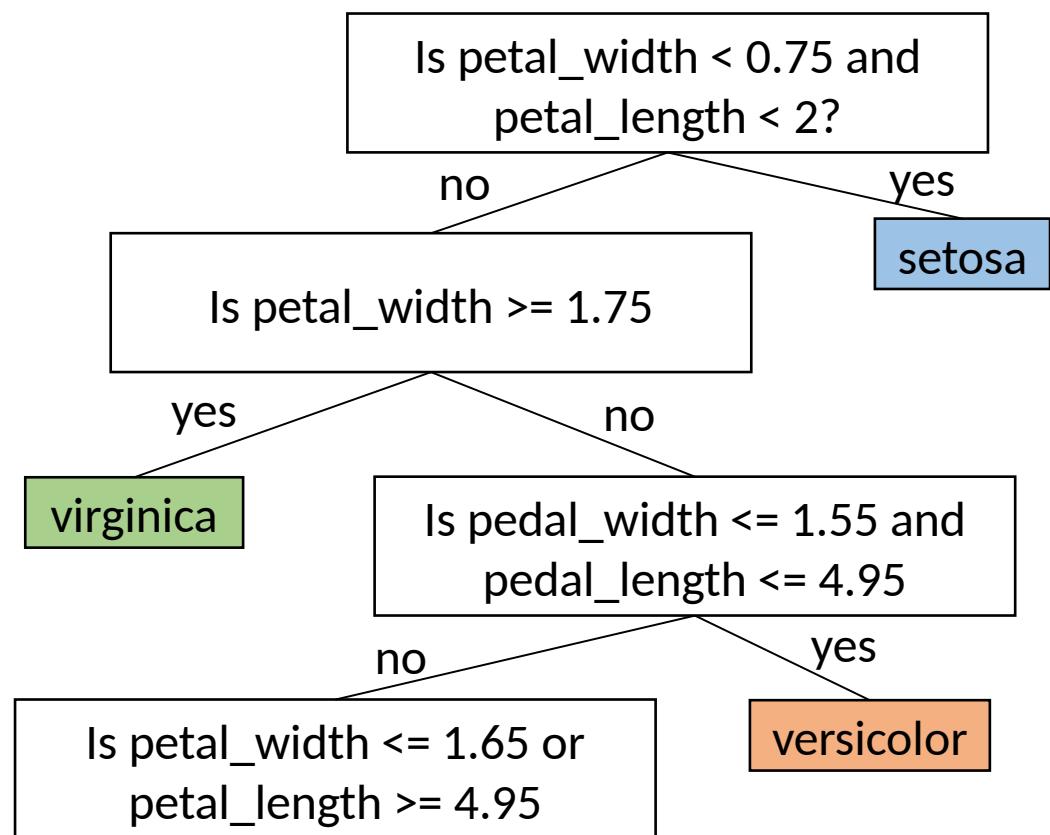
# Example: Using Petal Data Only



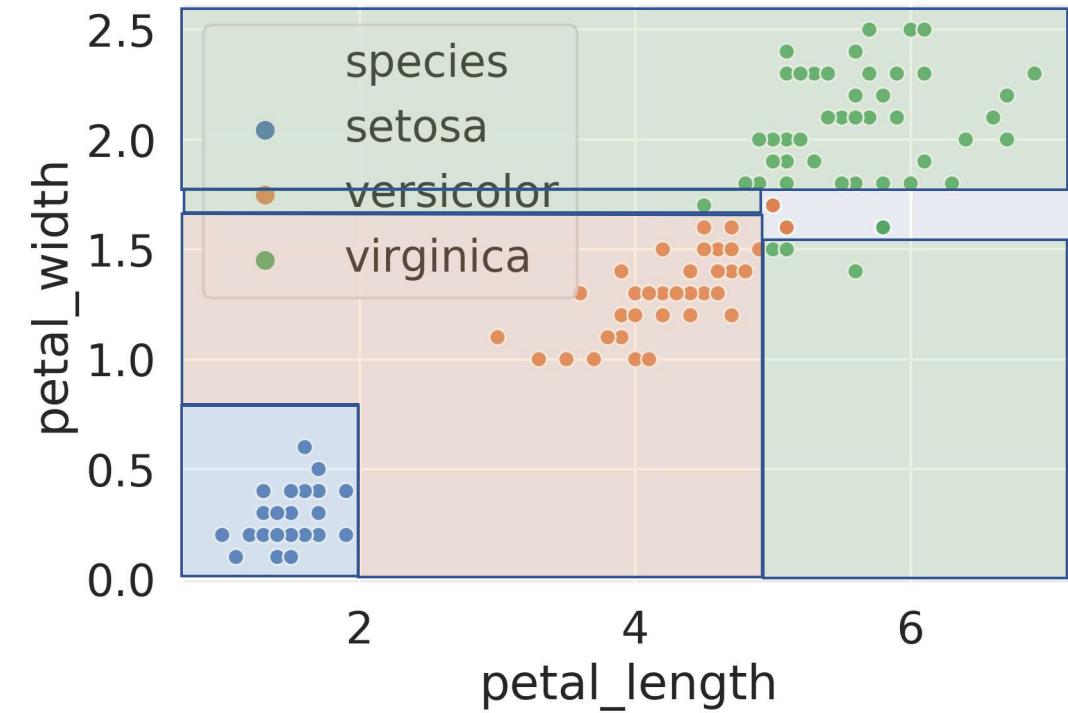
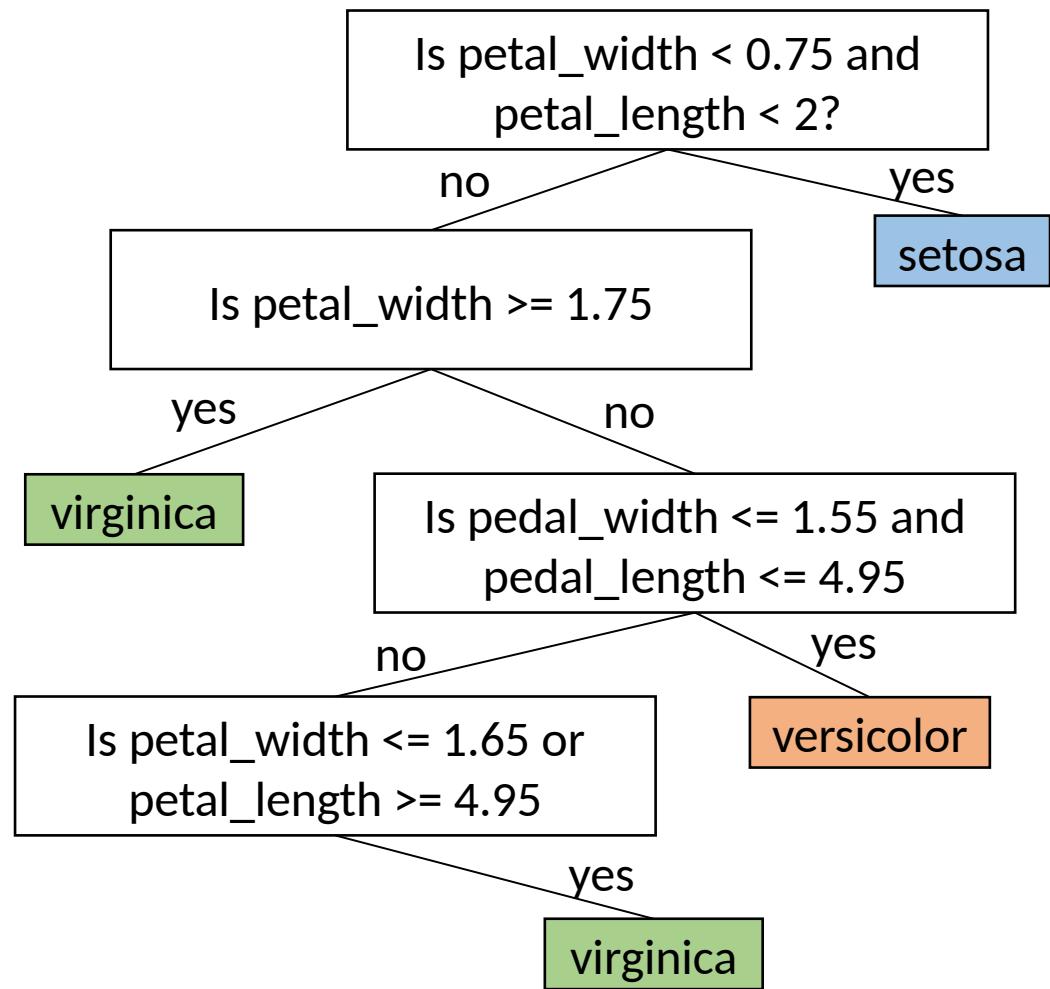
# Example: Using Petal Data Only



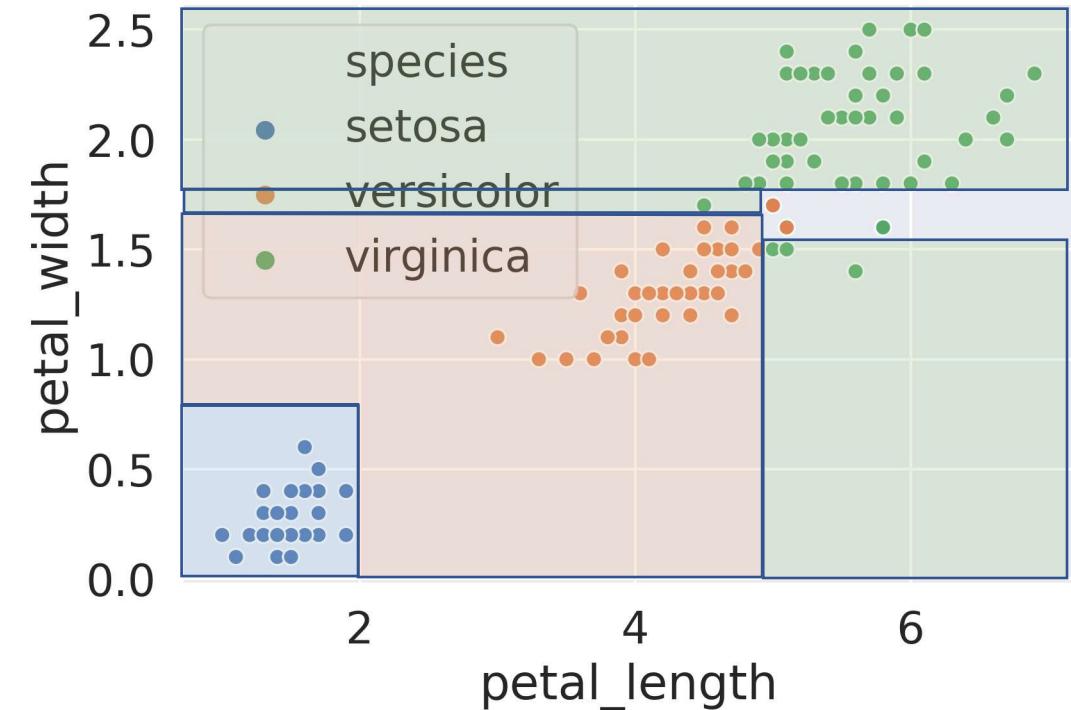
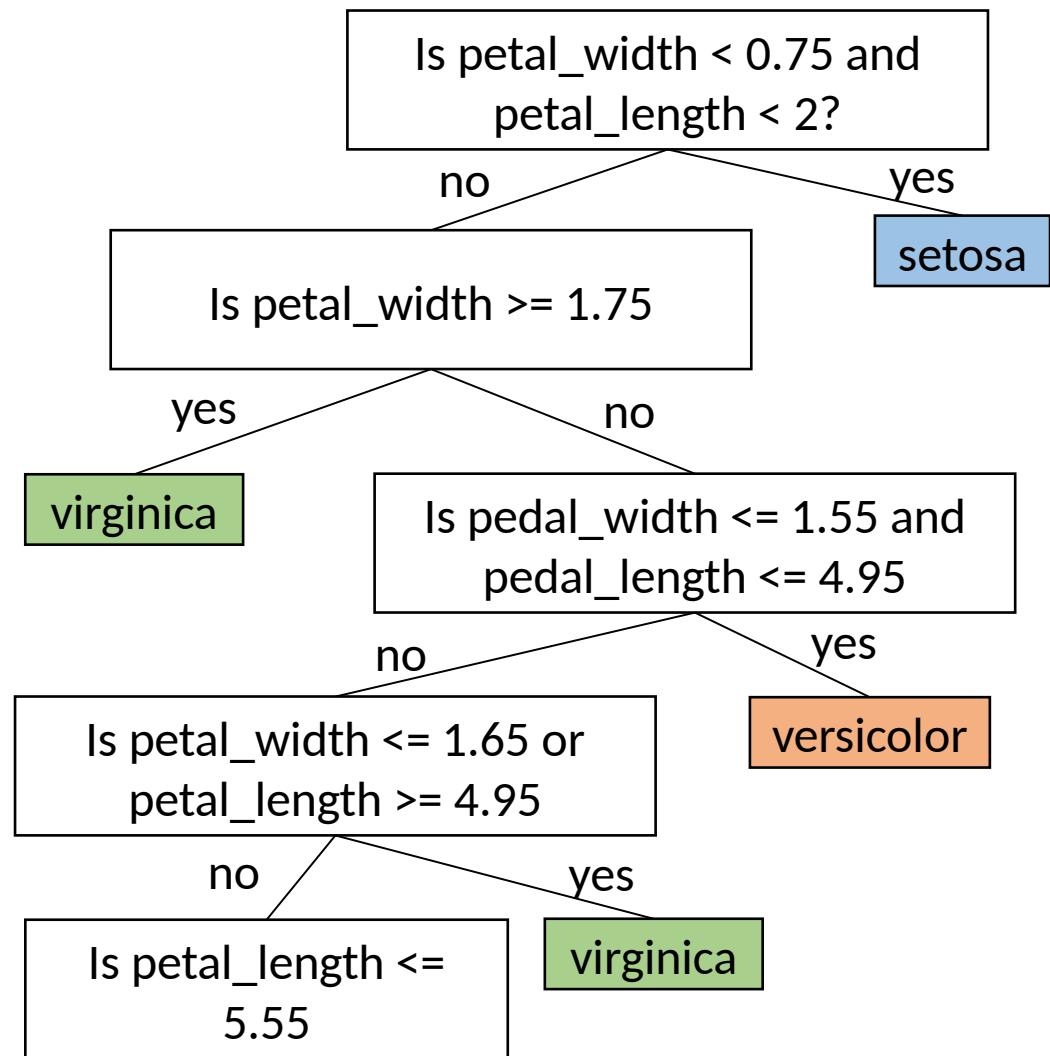
# Example: Using Petal Data Only



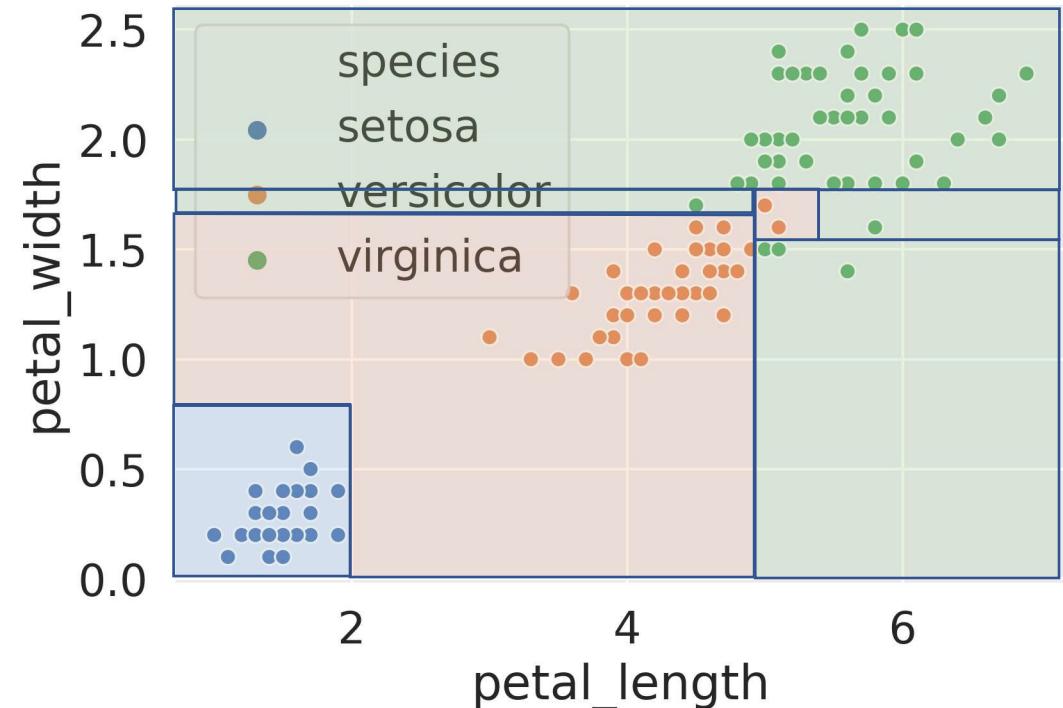
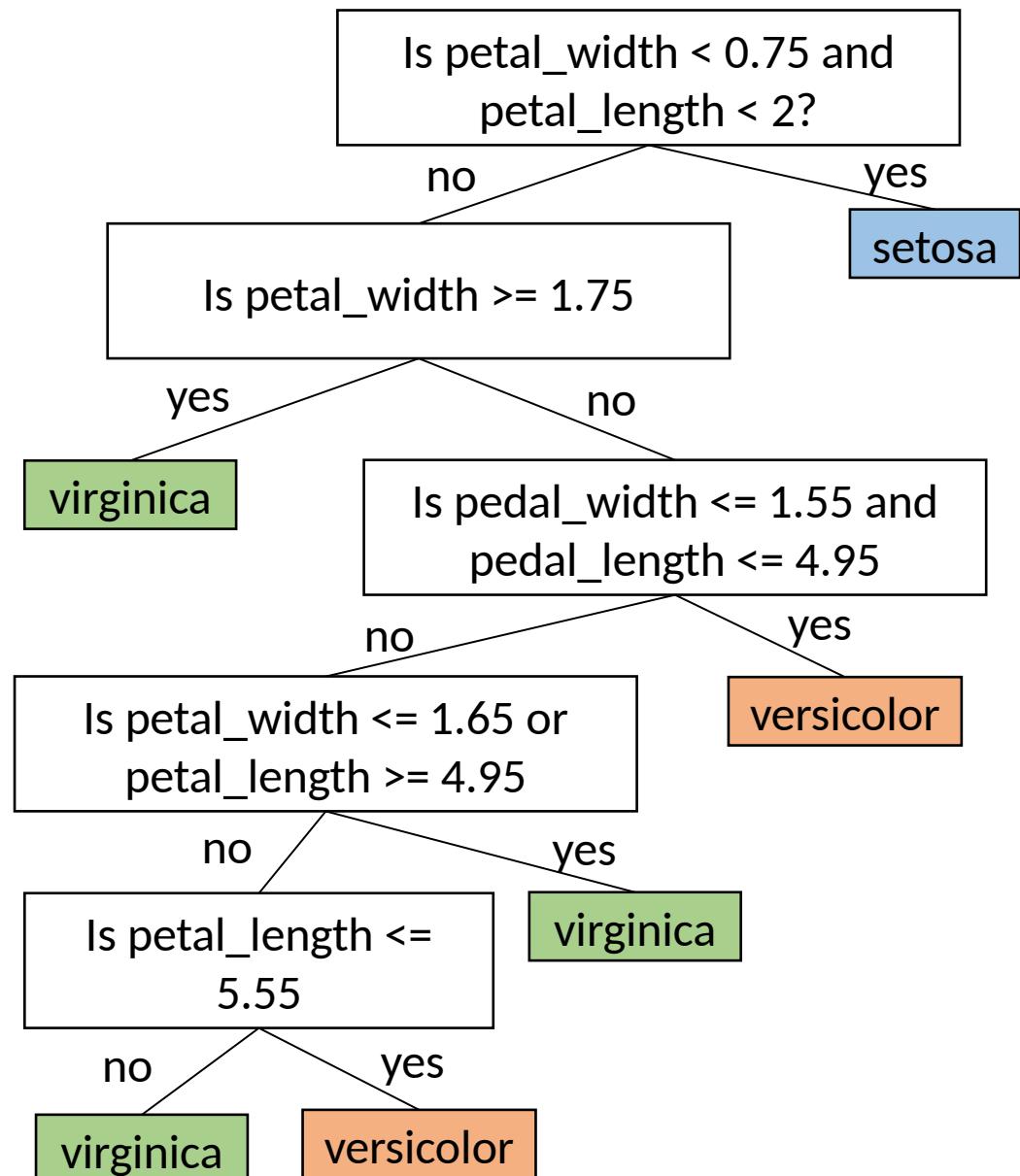
# Example: Using Petal Data Only



# Example: Using Petal Data Only



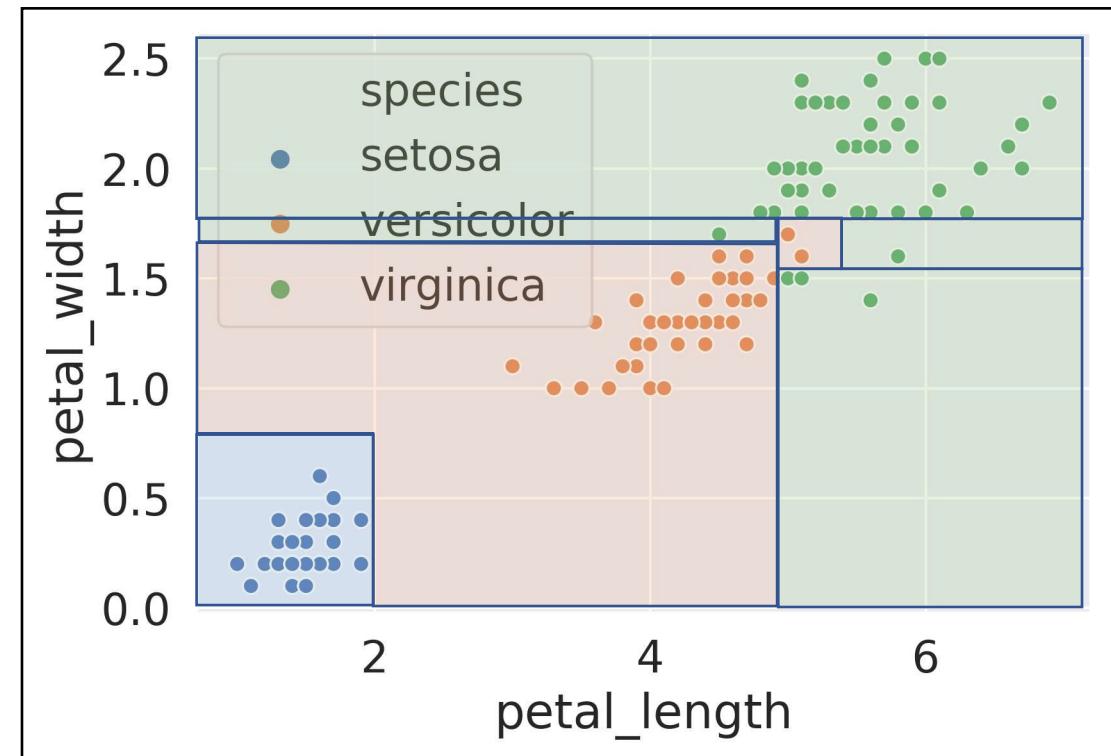
# Example: Using Petal Data Only



## Example: Using Petal Data Only

How accurate is our decision tree model on the training data?

Is this good or bad?



## Example: Using Petal Data Only

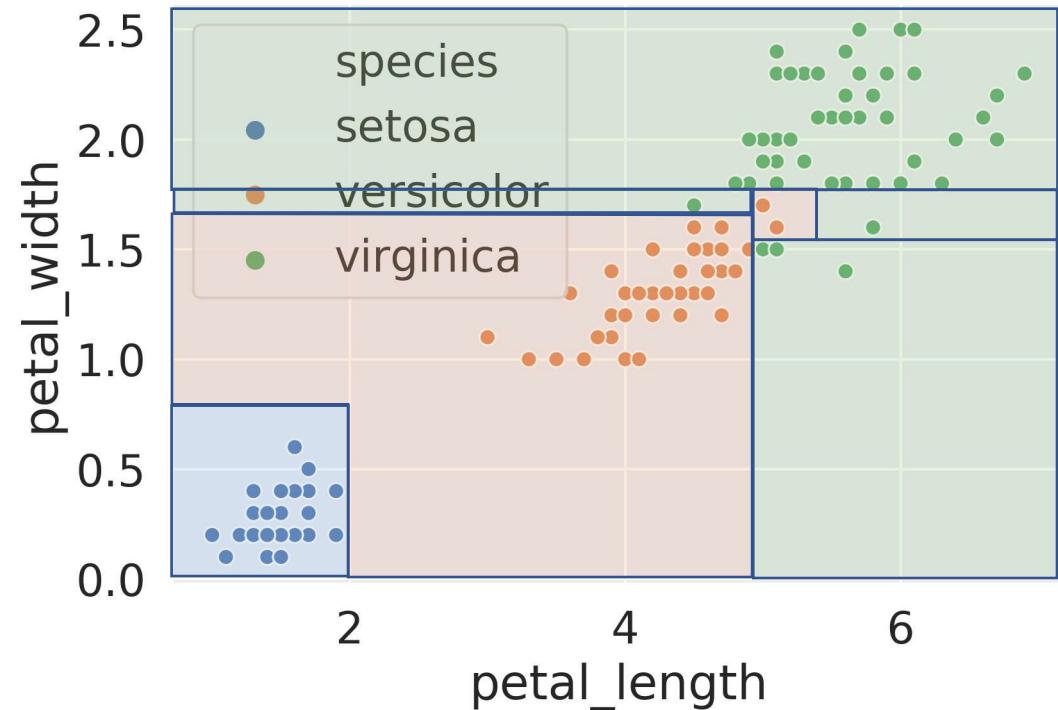
How accurate is our decision tree model on the training data?

- It seems like it gets every point correct

Is this good or bad?

- I'd argue bad
- Seems likely to result in overfitting!

First, let's see how we can build decision trees for classification using `scikit-learn`



# Decision Trees in scikit-learn

# Decision Tree Models With scikit-learn

---

The code to build a decision tree model in scikit-learn is very similar to what we saw for building linear and logistic regression models:

```
from sklearn import tree
decision_tree_model = tree.DecisionTreeClassifier(criterion='entropy')
decision_tree_model = decision_tree_model.fit(iris_data[["petal_length", "petal_width"]], iris_data["species"])
```

# Decision Tree Models With scikit-learn

The code to build a decision tree model in scikit-learn is very similar to what we saw for building linear and logistic regression models:

```
from sklearn import tree
decision_tree_model = tree.DecisionTreeClassifier(criterion='entropy')
decision_tree_model = decision_tree_model.fit(iris_data[["petal_length", "petal_width"]], iris_data["species"])
```

```
four_random_rows = iris_data.sample(4)
four_random_rows
```

	sepal_length	sepal_width	petal_length	petal_width	species
18	5.7	3.8	1.7	0.3	setosa
140	6.7	3.1	5.6	2.4	virginica
104	6.5	3.0	5.8	2.2	virginica
11	4.8	3.4	1.6	0.2	setosa

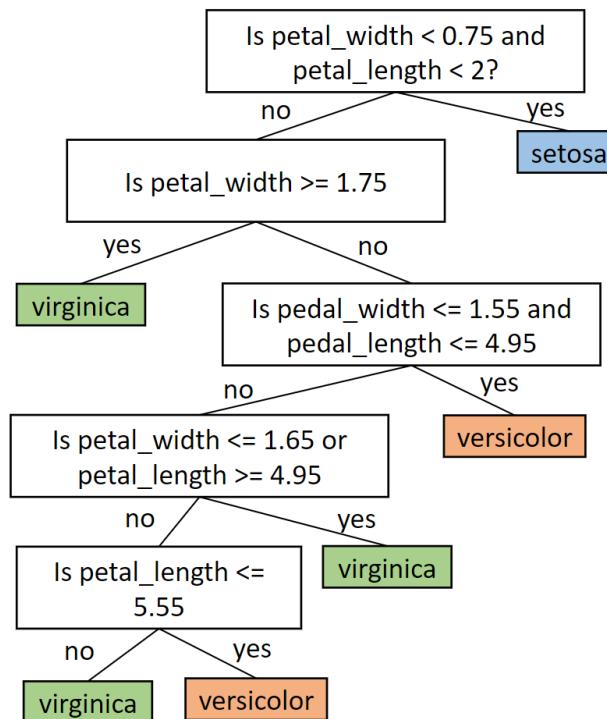
```
decision_tree_model.predict(four_random_rows[["petal_length", "petal_width"]])
```

```
array(['setosa', 'virginica', 'virginica', 'setosa'], dtype=object)
```

# Visualizing Decision Tree Models

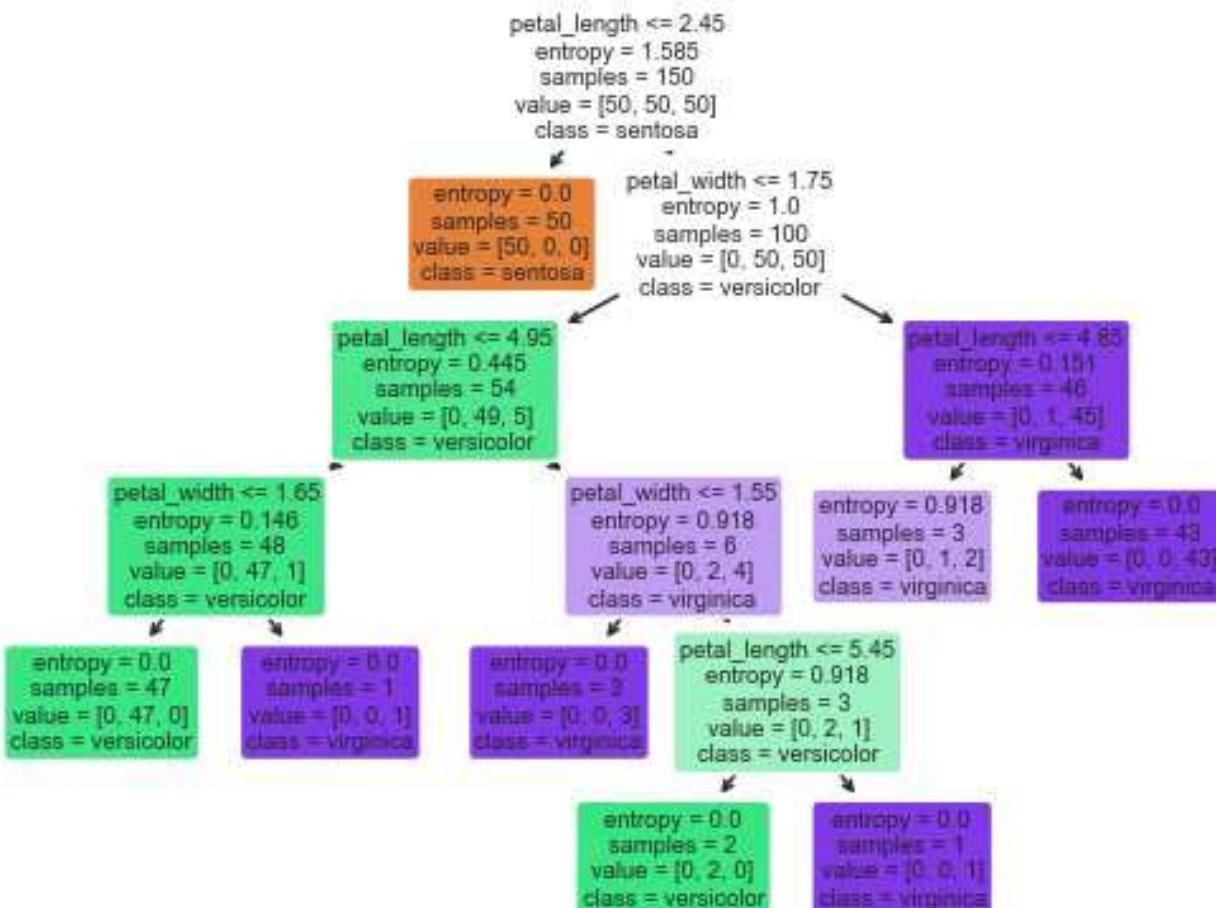
```
from sklearn import tree
decision_tree_model = tree.DecisionTreeClassifier(criterion='entropy')
decision_tree_model = decision_tree_model.fit(iris_data[["petal_length", "petal_width"]], iris_data["species"])
```

Suppose we want to visualize the decision tree, similar to what we saw earlier:



# Visualizing Decision Tree Models - default

```
from sklearn import tree
decision_tree_model = tree.DecisionTreeClassifier(criterion='entropy')
decision_tree_model = decision_tree_model.fit(iris_data[["petal_length", "petal_width"]], iris_data["species"])
```

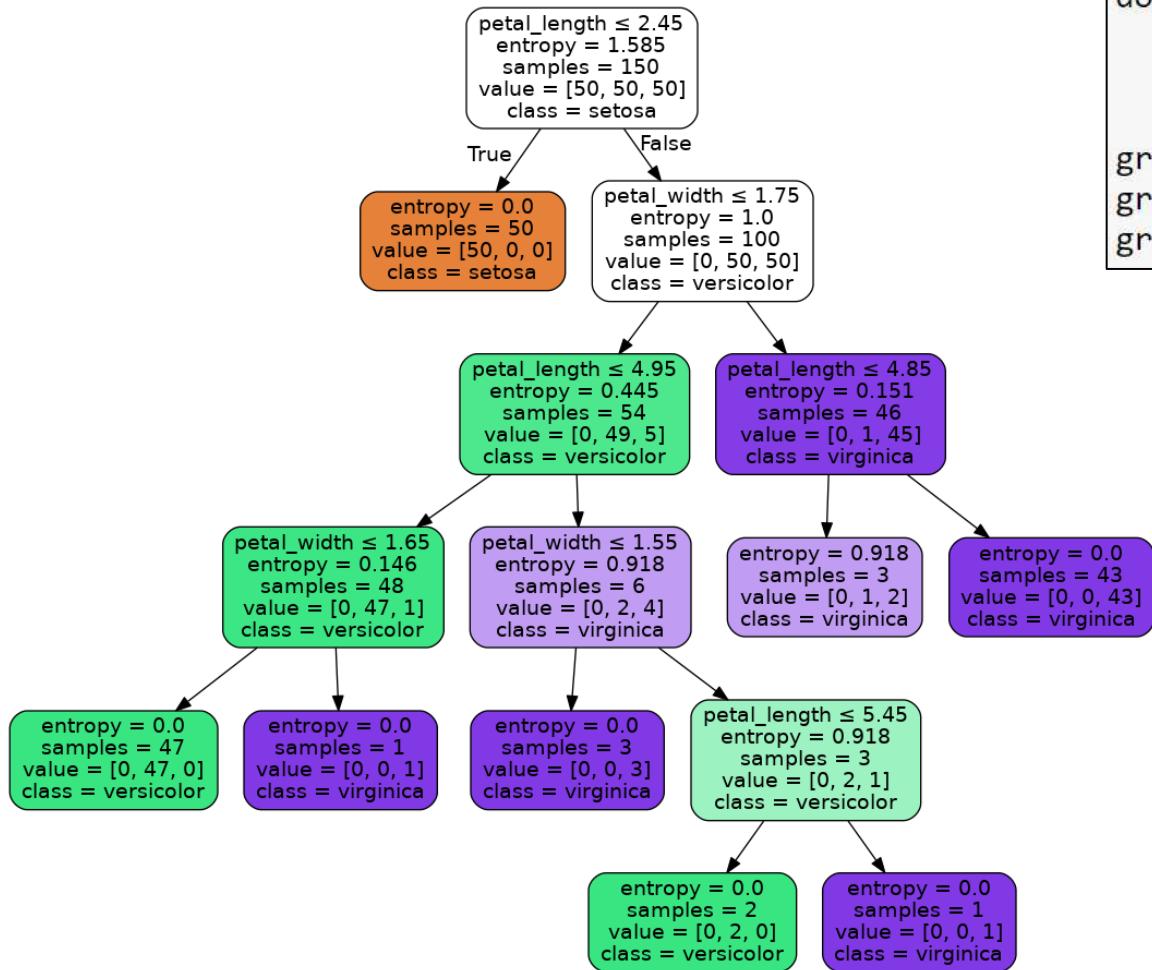


There is a built in DecisionTree visualizer

- Unfortunately, it isn't very good

# Visualizing Decision Tree Models - use GraphViz

Can use GraphViz to get a much nicer picture.



```
import graphviz
dot_data = tree.export_graphviz(decision_tree_model, out_file=None,
                                feature_names=["petal_length", "petal_width"],
                                class_names=["setosa", "versicolor", "virginica"],
                                filled=True, rounded=True)
graph = graphviz.Source(dot_data)
graph.render(format="png", filename="iris_tree")
graph
```

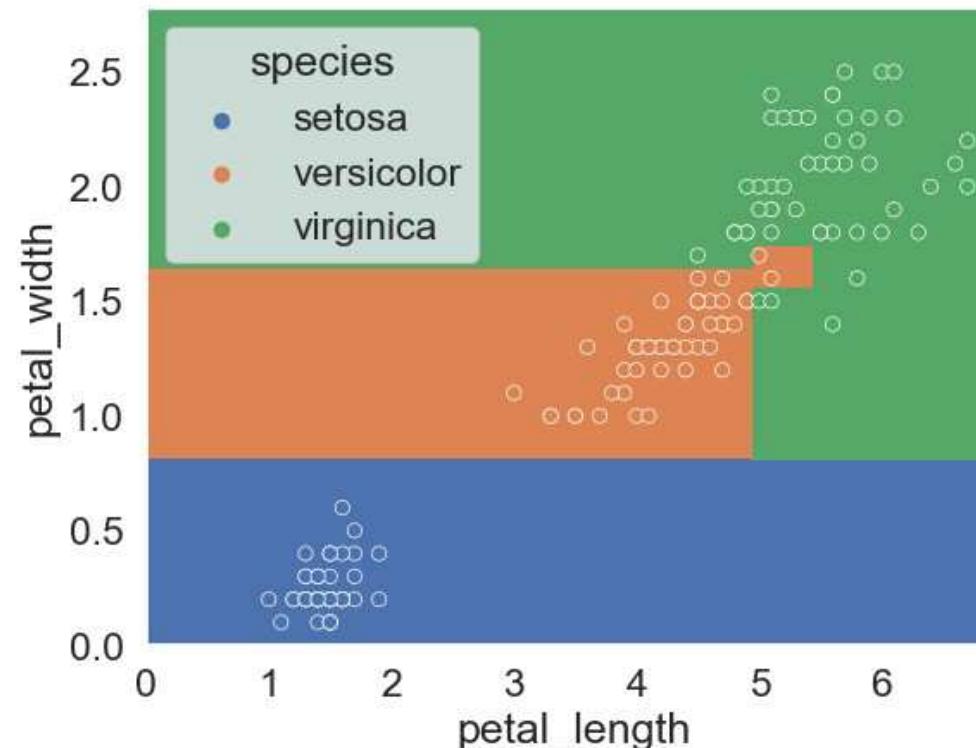
In each box, we see:

- The rule.
- The entropy at that node (more later).
- The number of samples that remain after applying all of the above rules.
- The number of samples that remain.
- The most likely class.

## Visualizing Decision Boundary for Decision Tree

Plotting the decision boundaries for decision tree models, we get the results below.

- Decision tree has nonlinear boundary, and appears to get 100% accuracy.
  - Let's calculate the exact accuracy rather than just relying on our eyes to look at a somewhat complex visualization.



# Measuring the Performance of Our Model

---

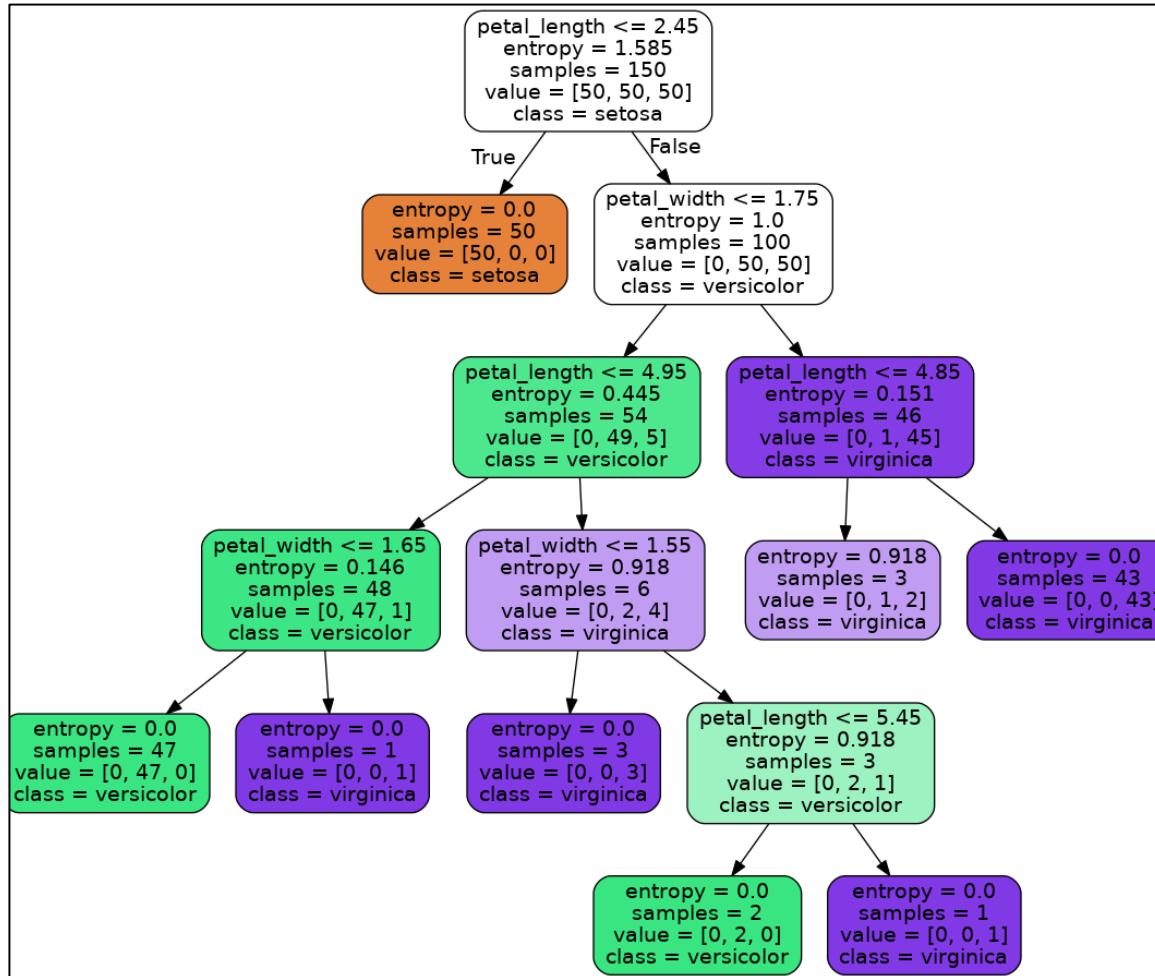
Running the code below, we see that we only get 99.3% accuracy

[https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy\\_score.html](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html)

```
from sklearn.metrics import accuracy_score
predictions = decision_tree_model.predict(iris_data[["petal_length", "petal_width"]])
accuracy_score(predictions, iris_data["species"])
```

To understand why, let's look back at our decision tree model.

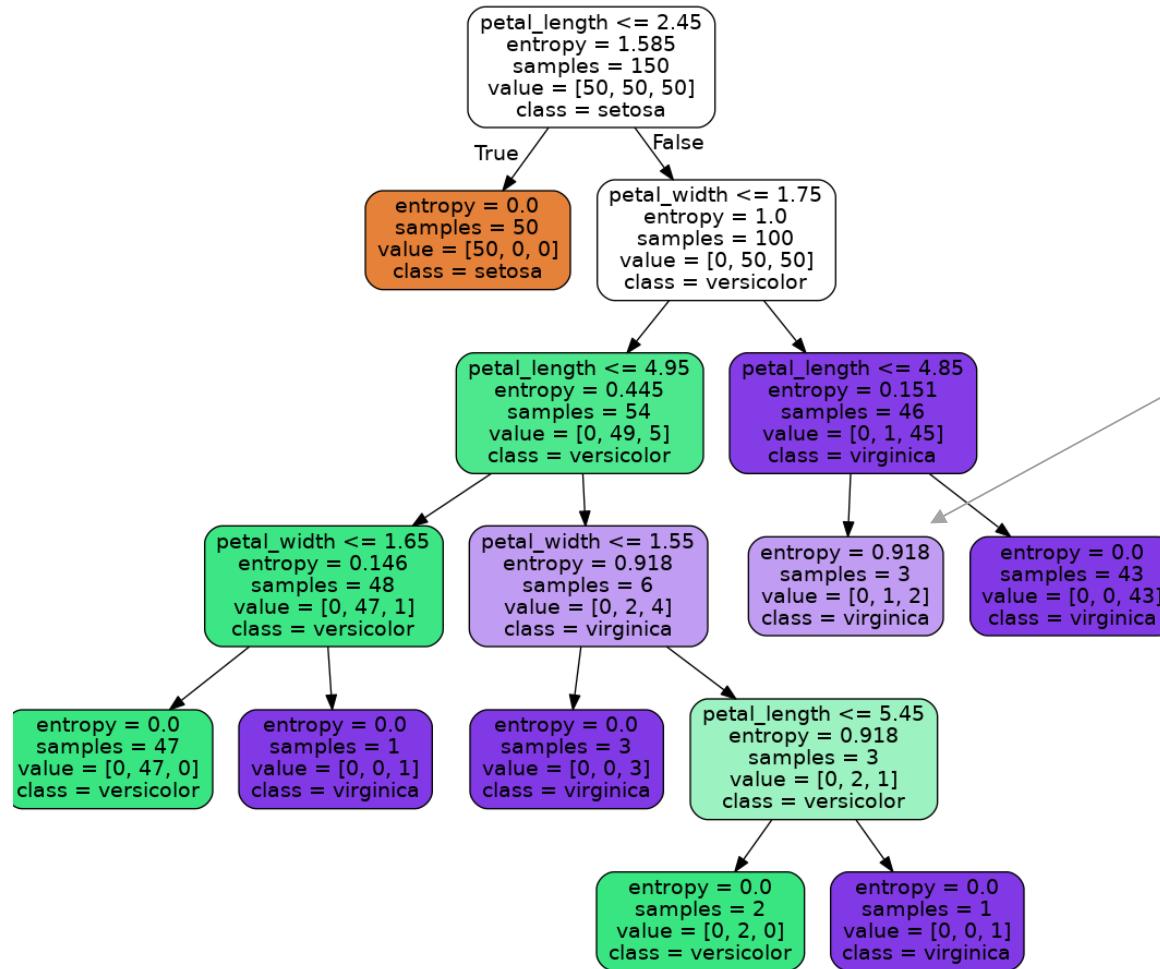
# Understanding Our Decision Tree



There is one terminal decision point where there is more than one possible right answer.

Can you find it?

# Understanding Our Decision Tree



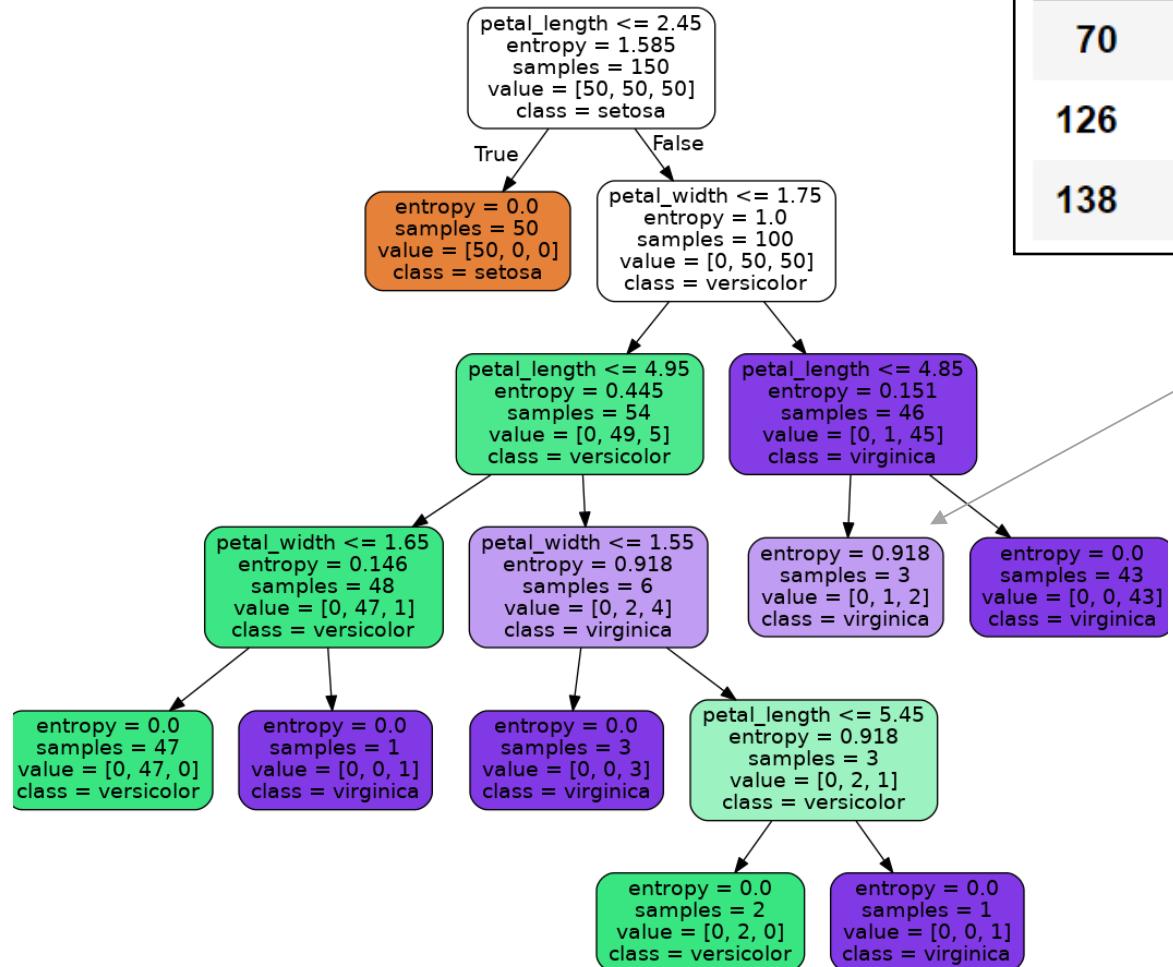
There is one terminal decision point where there is more than one possible right answer.

The model was unable to come up with a decision rule to resolve these last 3 samples.

Let's see why using the query method of the dataframe class.

# Understanding Our Decision Tree

```
iris_data.query("petal_length > 2.45 and petal_width > 1.75 and petal_length < 4.85")
```



sepal_length	sepal_width	petal_length	petal_width	species
70	5.9	3.2	4.8	versicolor
126	6.2	2.8	4.8	virginica
138	6.0	3.0	4.8	virginica

There is one terminal decision point where there is more than one possible right answer.

- In the original data set, there was a  $\text{versicolor}$  iris with the same petal measurements as two  $\text{virginicas}$ .

# Overfitting

## Overfitting and Decision Trees

---

	sepal_length	sepal_width	petal_length	petal_width	species
70	5.9	3.2	4.8	1.8	versicolor
126	6.2	2.8	4.8	1.8	virginica
138	6.0	3.0	4.8	1.8	virginica

sk-learn decision trees will always have perfect accuracy on the training data, EXCEPT when there are samples from different categories with the exact same features.

- Example: If the versicolor above had a petal\_length of 4.8001, we'd have 100% training accuracy.

This tendency for perfect accuracy should give us concern about overfitting.

- Model is extremely sensitive / has high variance.

## Sepal Decision Tree

In order to understand how to avoid overfitting, let's first discuss how decision trees are created from data.

Traditional decision tree generation algorithm:

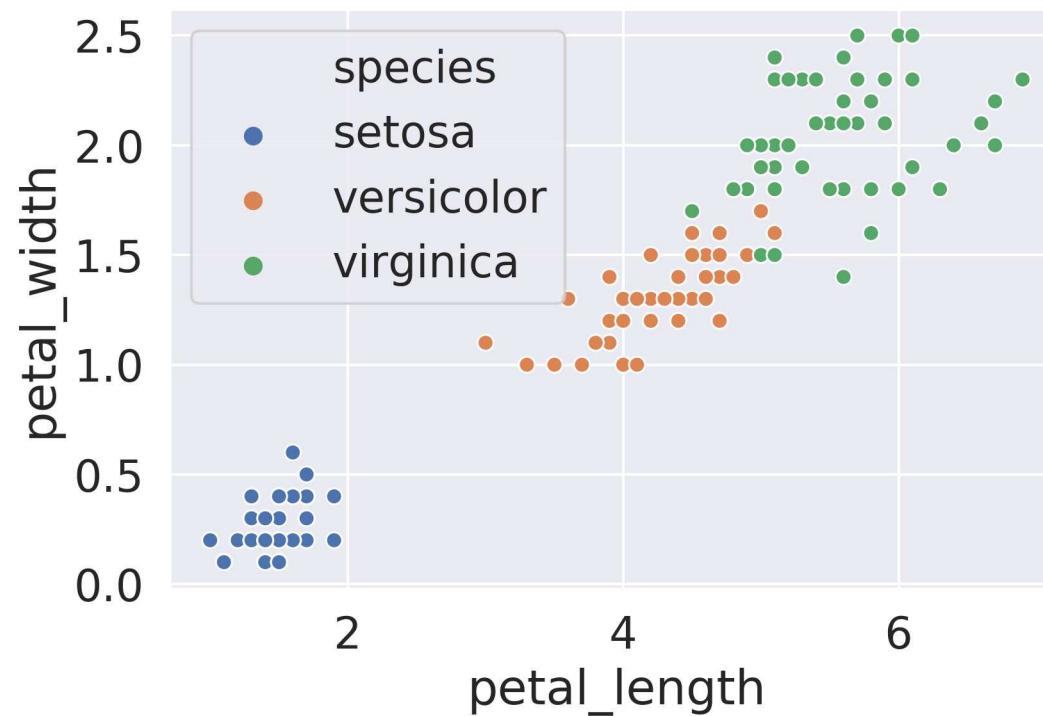
- All of the data starts in the root node.
- **Repeat** until every node is either **pure** or **unsplittable**:
  - **Pick the best feature**  $x$  and **best split value**  $\beta$ , e.g.  $x = \text{petal\_length}$ ,  $\beta = 2$ .
  - **Split data into two nodes**, one where  $x < \beta$ , and one where  $x \geq \beta$ .

Notes: A node that has only one samples from one class is called a “**pure**” node. A node that has overlapping data points from different classes and thus that cannot be split is called “**unsplittable**”.

## Defining a Best Feature

Question: Which feature and split value is best?

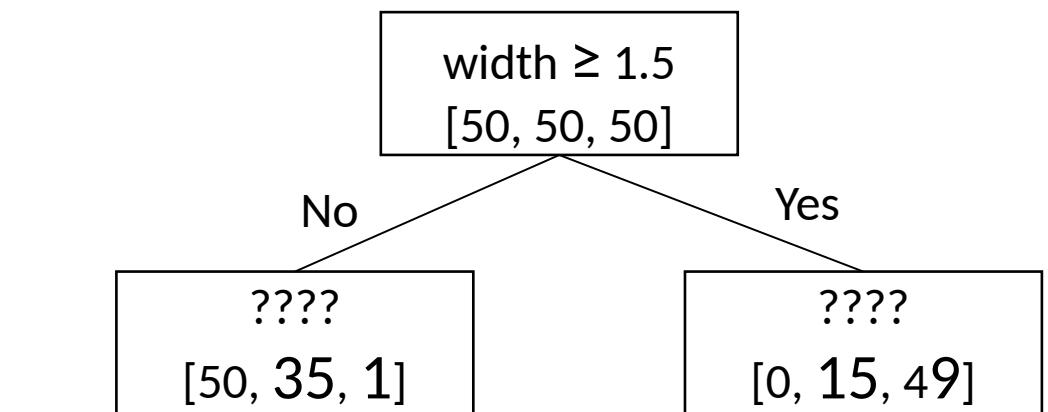
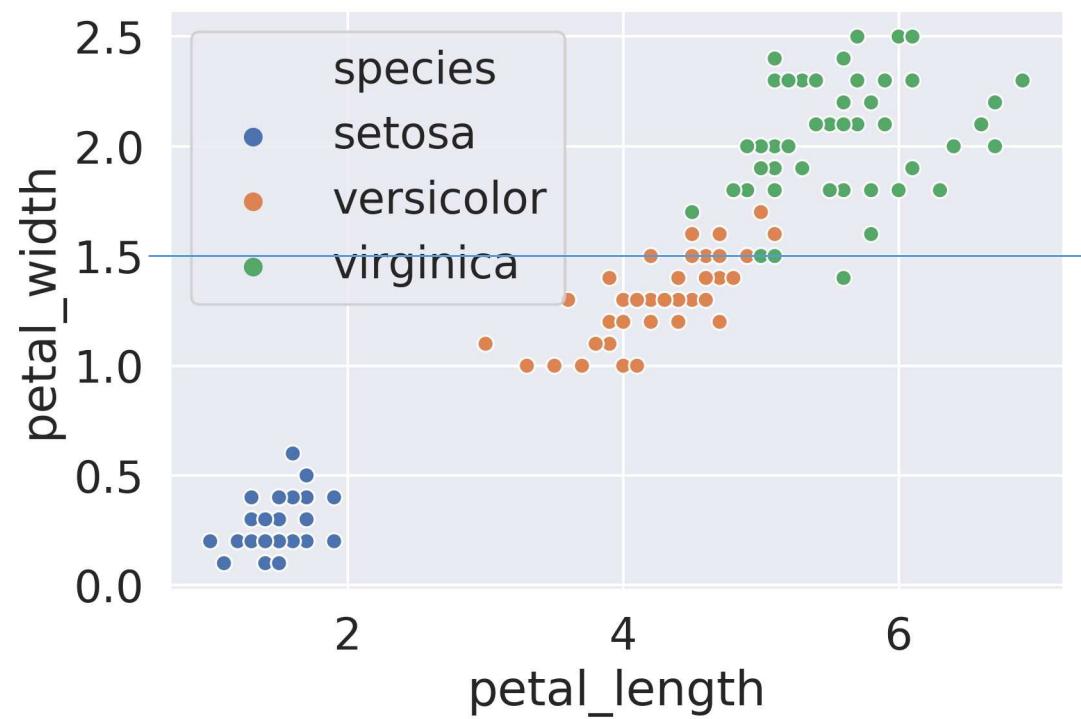
- Equivalently: Which horizontal or vertical line do we want to draw?



## Defining a Best Feature

Question: Which feature and split value is best?

- Equivalently: Which horizontal or vertical line do we want to draw?

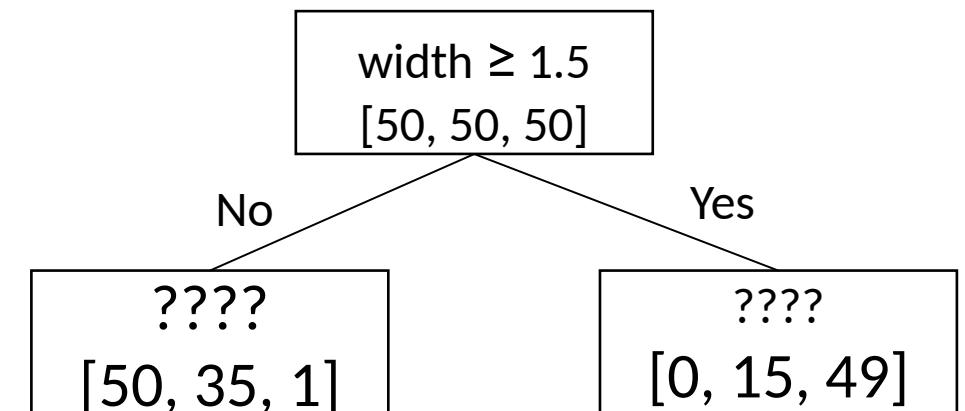
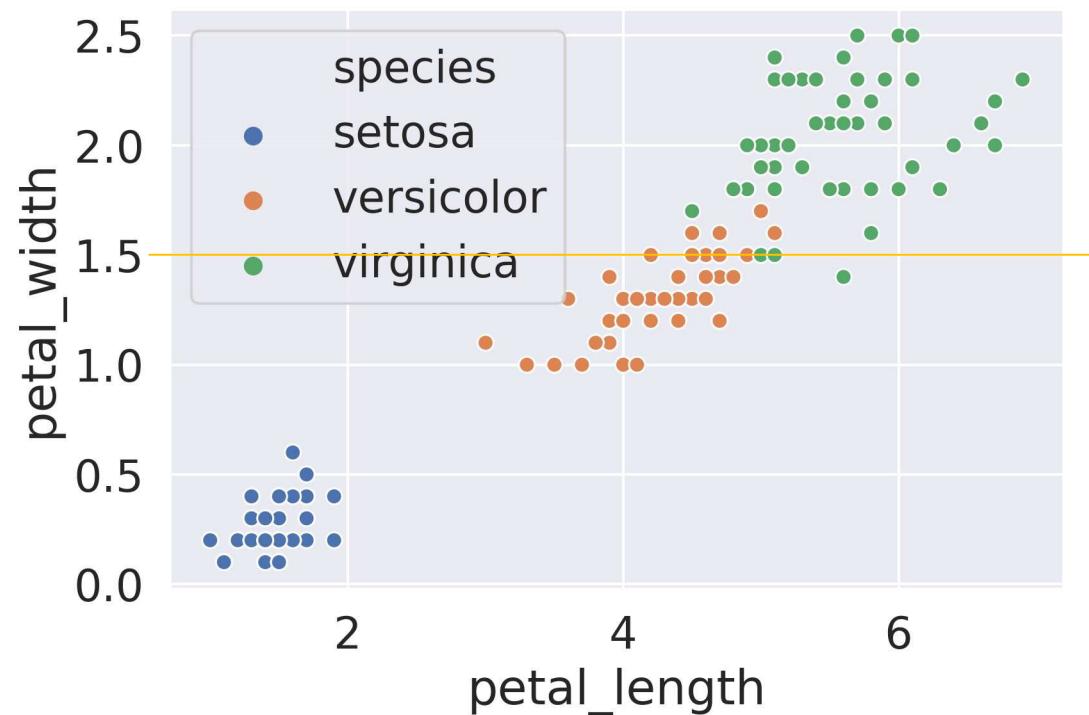


Is this  
good?

## Defining a Best Feature

Question: Which feature and split value is best?

- Equivalently: Which horizontal or vertical line do we want to draw?

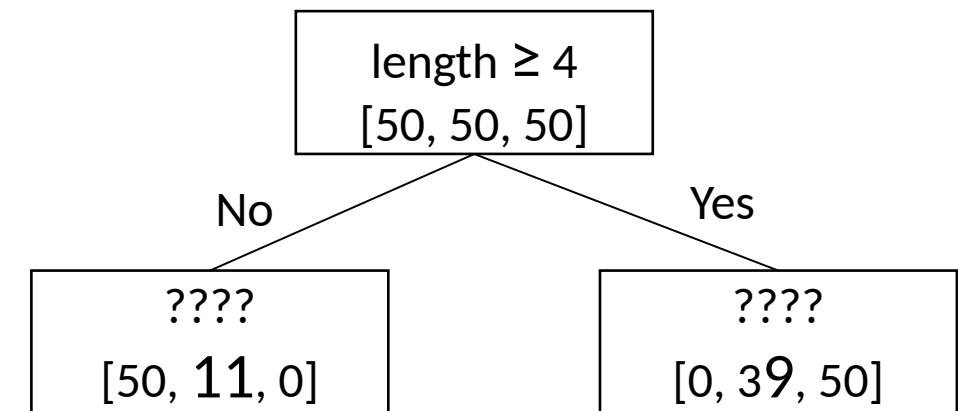
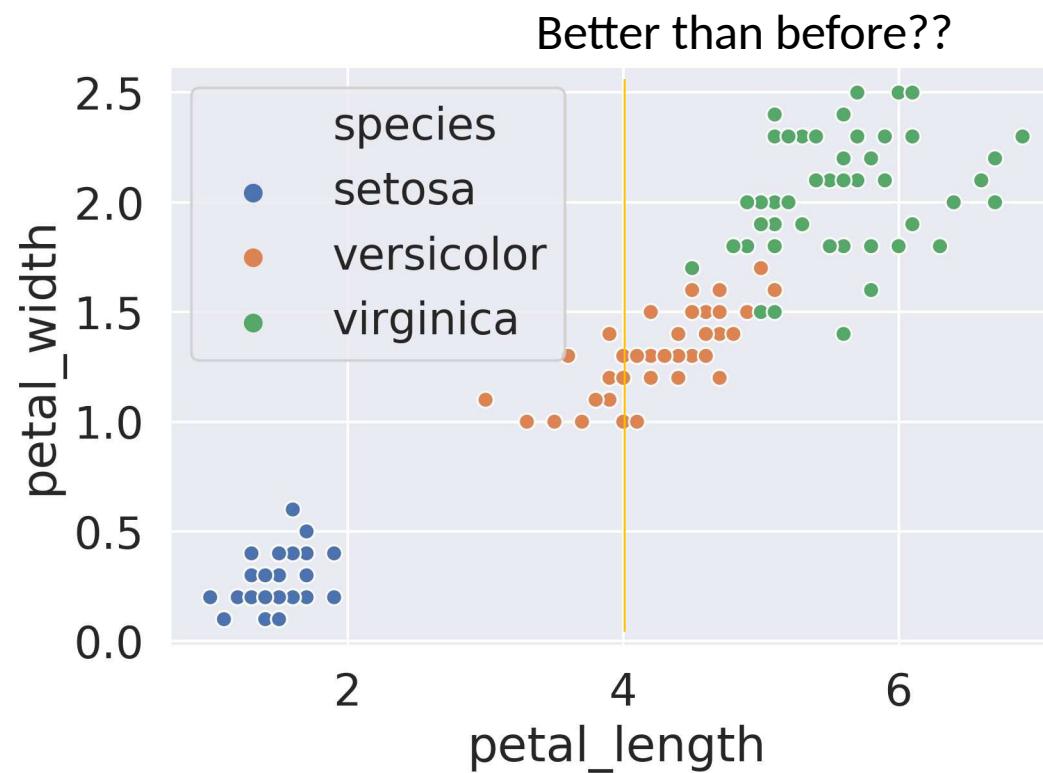


Is this good? It does help,  
but we could do better!

## Defining a Best Feature

Question: Which feature and split value is best?

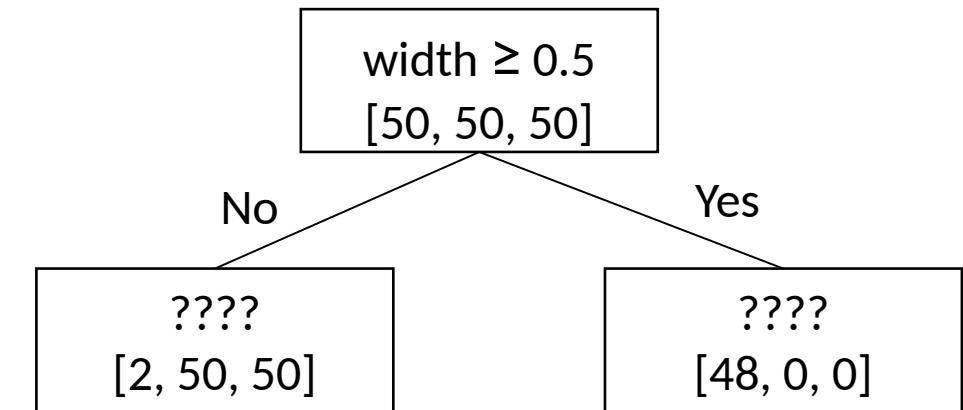
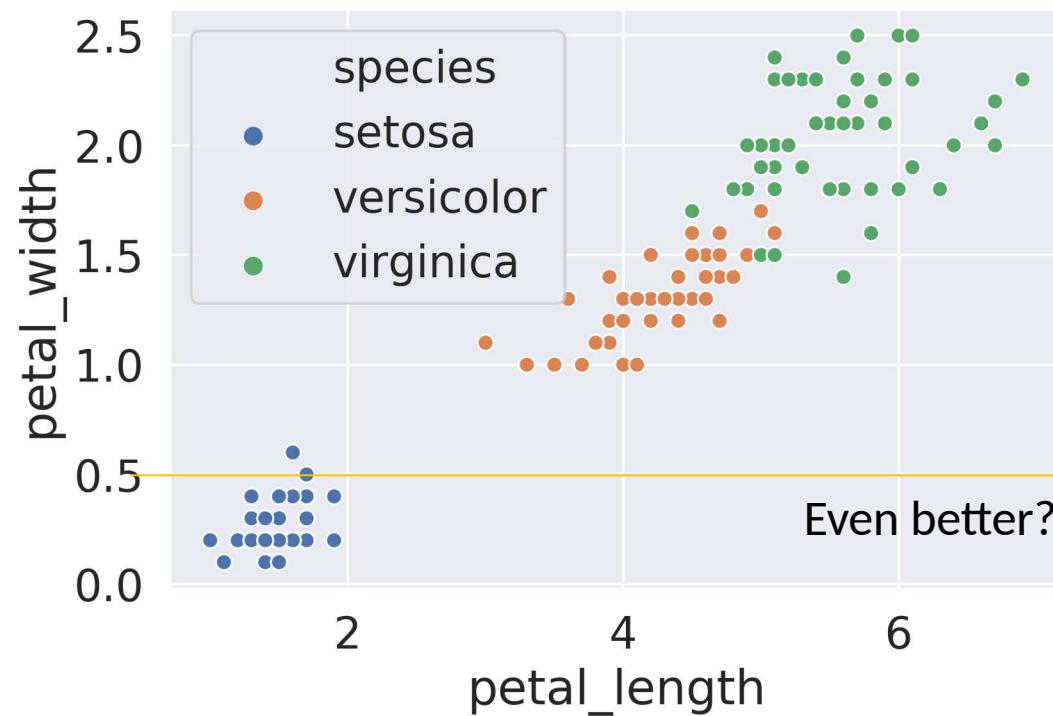
- Equivalently: Which horizontal or vertical line do we want to draw?



## Defining a Best Feature

Question: Which feature and split value is best?

- Equivalently: Which horizontal or vertical line do we want to draw?

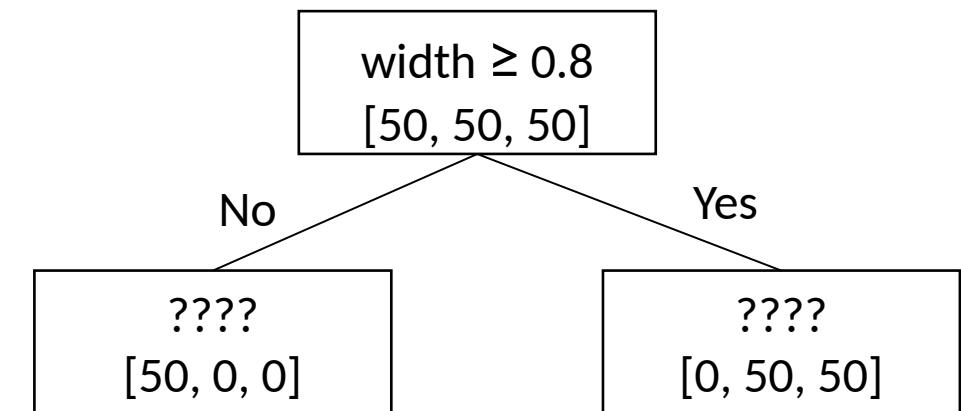
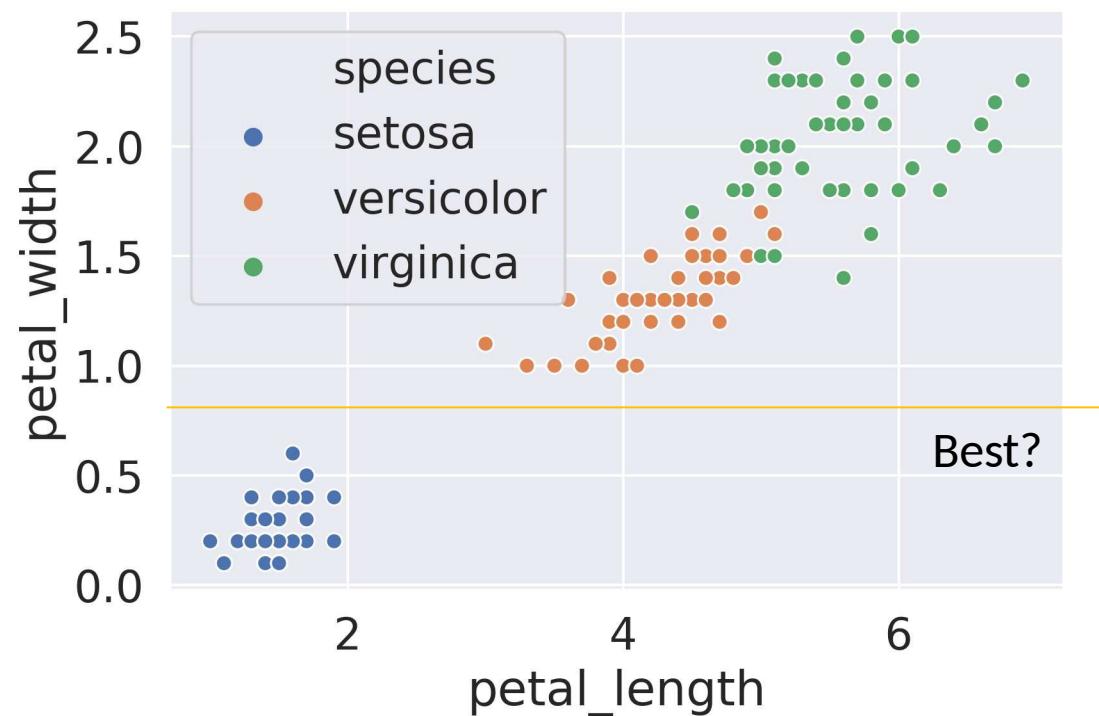


## Defining a Best Feature

Question: Which feature and split value is best?

- Equivalently: Which horizontal or vertical line do we want to draw?

We need some sort of rigorous definition for a good split.



## Node Entropy

Let  $p_C$  be the proportion of data points in a node with label C.

For example, for the node at the top of the decision tree,  $p_0 = 34/110 = 0.31$ ,  $p_1 = 36/110 = 0.33$ , and  $p_2 = 40/110 = 0.36$ .

Define the entropy S of a node (in bits) as:

$$S = - \sum_c p_c \log_2 p_c$$

For example, S for the top node is:

$$-0.31 \log_2 0.31 - 0.33 \log_2 0.33 - 0.36 \log_2 0.36 = 0.52 + 0.53 + 0.53 = 1.58 \text{ bits}$$

## Test Your Understanding

---

What is the entropy of the node with [31, 4, 1] in each class?

- $p_0 = 31/36 = 0.86$ ,  $p_1 = 4/36 = 0.11$ , and  $p_2 = 1/36 = 0.028$
- $S = -0.86 \log_2 0.86$ 
  - $0.11 \log_2 0.11$
  - $0.028 \log_2 0.028 = 0.68 \text{ bits}$

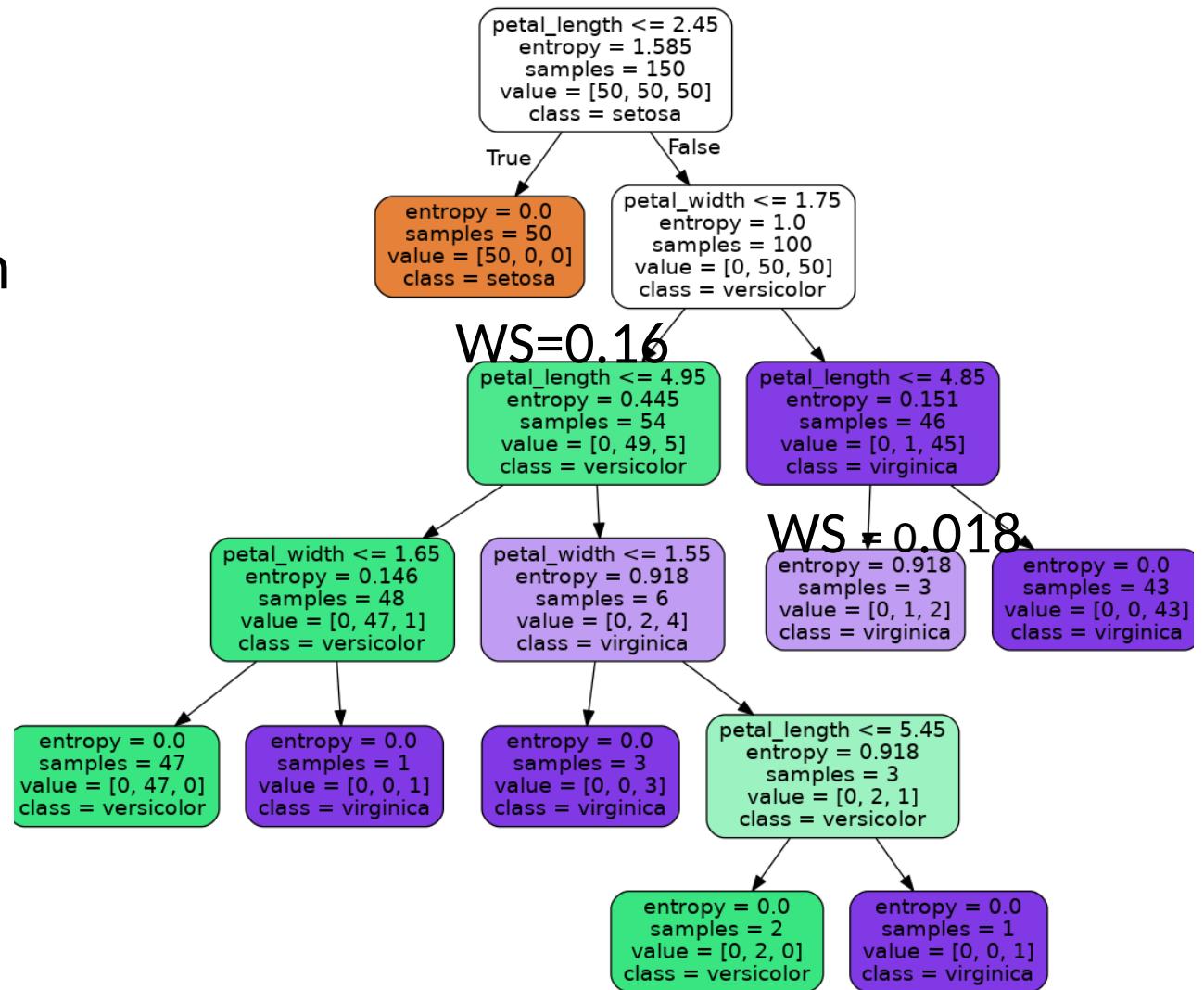
Can think of entropy as how unpredictable a node is. Low entropy means more predictable.  
High entropy means less predictable.

## Weighted Entropy

Define the weighted entropy of a node as its entropy scaled by the fraction of the samples in that node.

Consider our decision tree to the right.

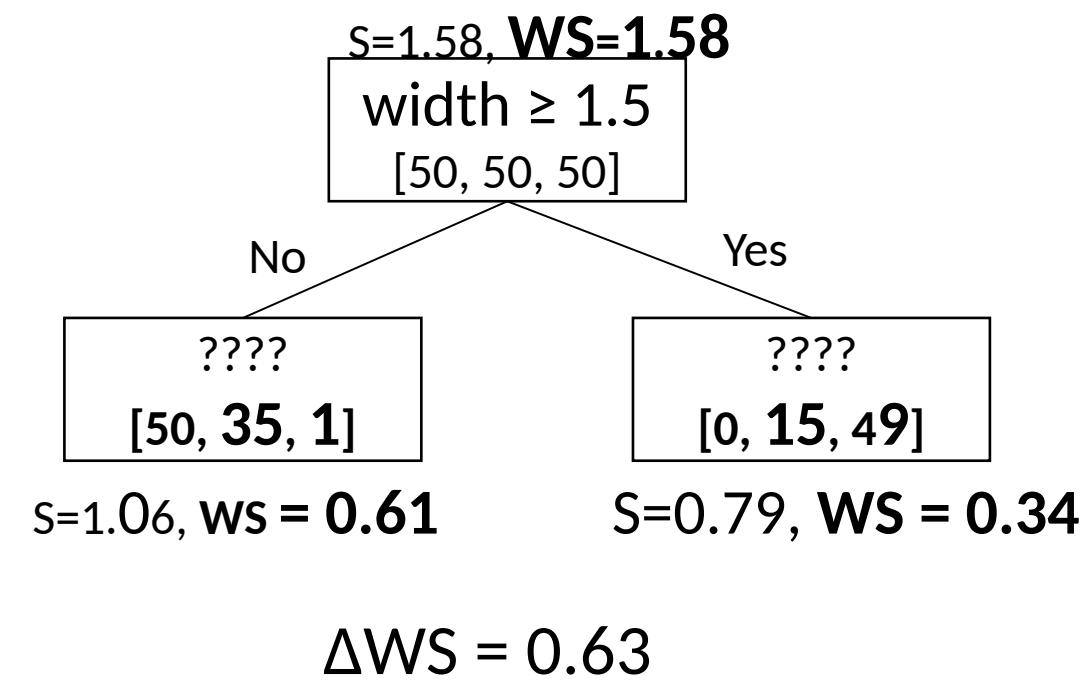
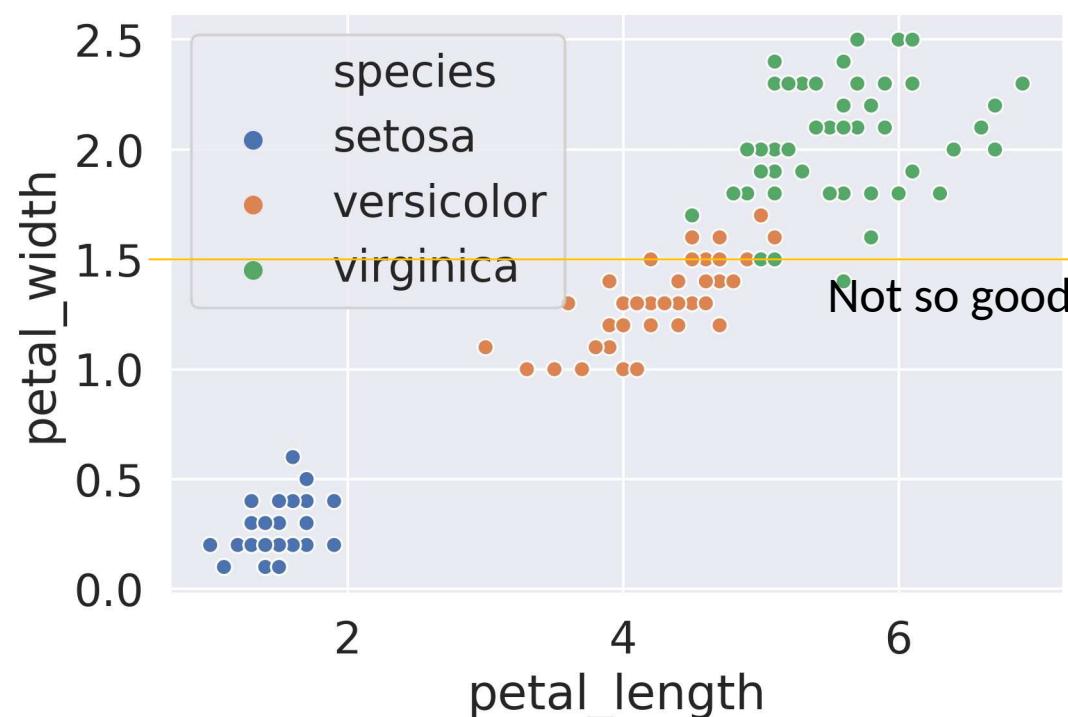
- Weighted entropy of the node with 54 samples is  $WS = 54/150 * 0.445 = 0.16$ .
- Weighted entropy of the terminal node with 3 samples is  $3/150 * 0.918 = 0.018$



## Defining a Best Feature

Split choice #1: width  $\geq 1.5$ . Compute entropy of child nodes:

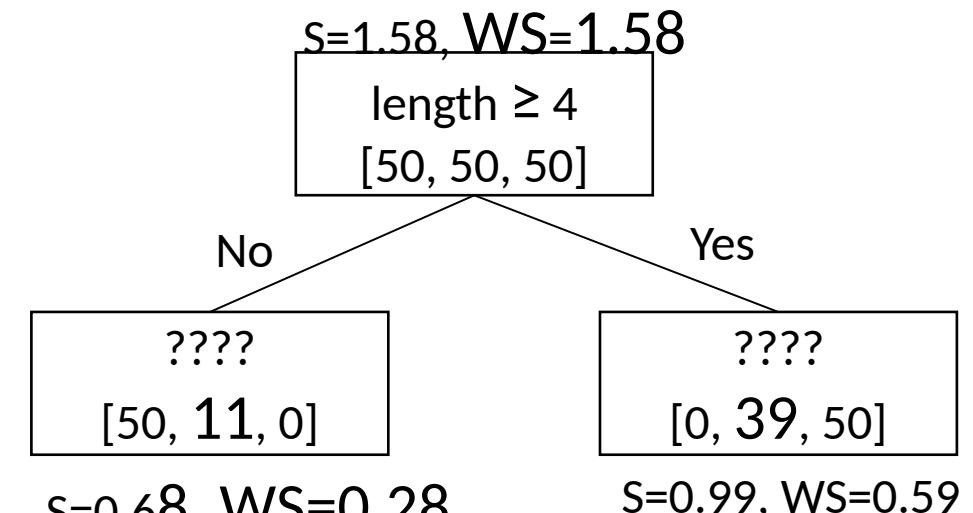
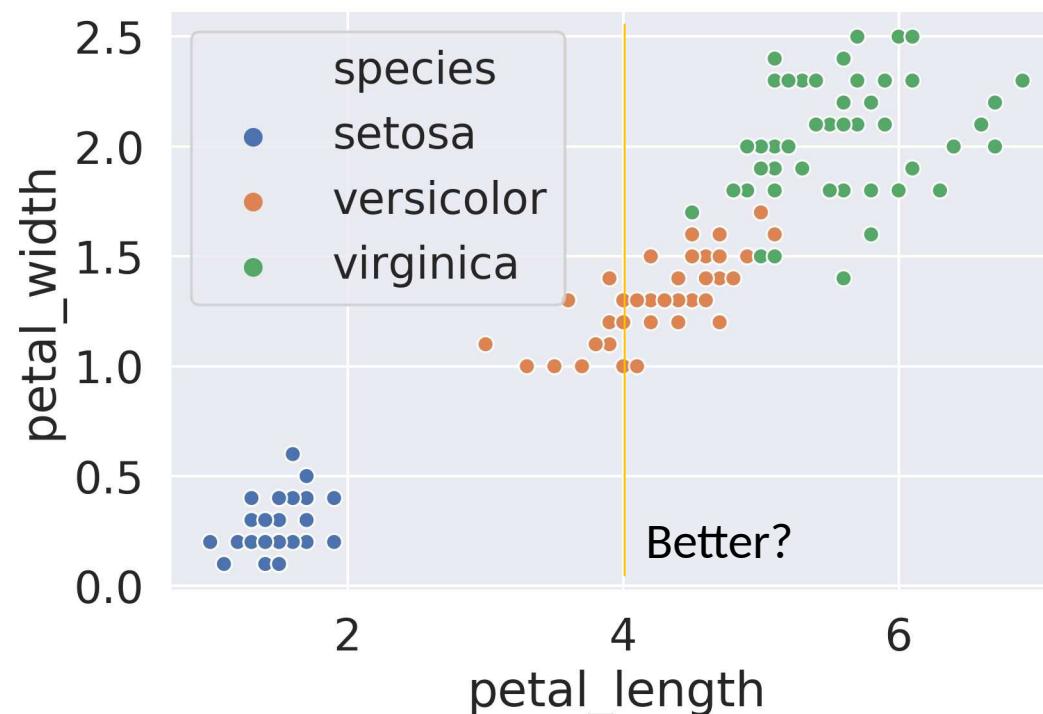
- $\text{entropy}([50, 50, 50]) = 1$ , weighted entropy =  $150/150 * 1 = 1.58$
- $\text{entropy}([50, 35, 1]) = 1.06$ , weighted entropy =  $86/150 * 1.06 = 0.61$
- $\text{entropy}([15, 49]) = 0.79$ , weighted entropy =  $64/150 * 0.79 = 0.34$
- $\Delta WS = 1.58 - 0.61 - 0.34 = 0.63$ .



## Defining a Best Feature

Split choice #2:  $\text{length} \geq 4$ . Compute entropy of child nodes:

- $\text{entropy}([50, 11]) = 0.68$ , weighted entropy =  $61/150 * 0.68 = 0.28$
- $\text{entropy}([39, 50]) = 0.99$ , weighted entropy =  $89/150 * 0.99 = 0.59$
- $\Delta\text{WS} = 1.58 - 0.28 - 0.59 = 0.71$ . Better than split choice #1 (had  $\Delta\text{WS} = 0.63$ ).

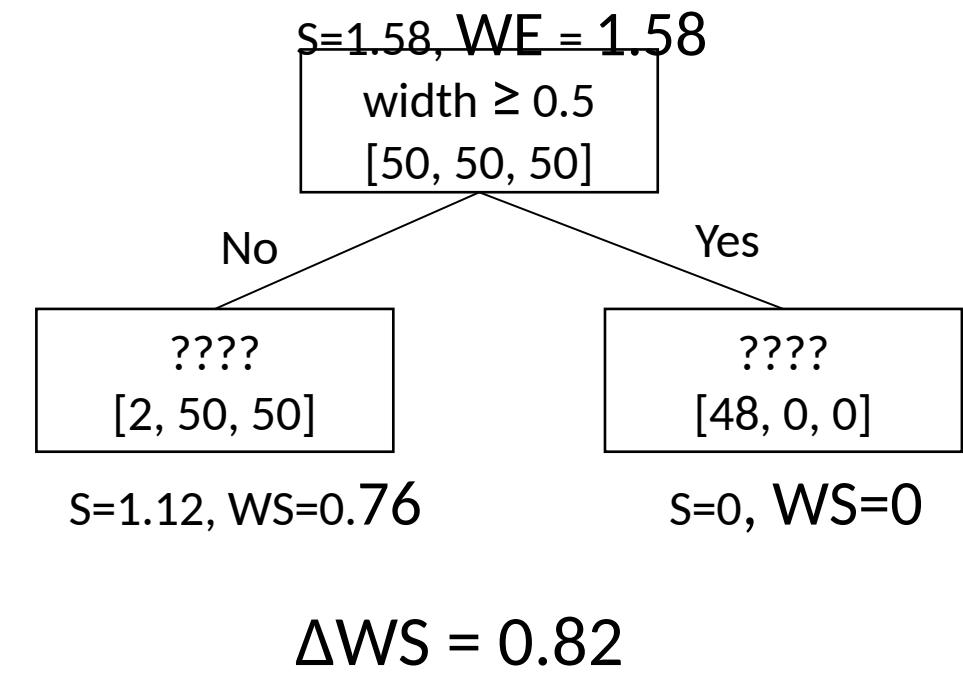
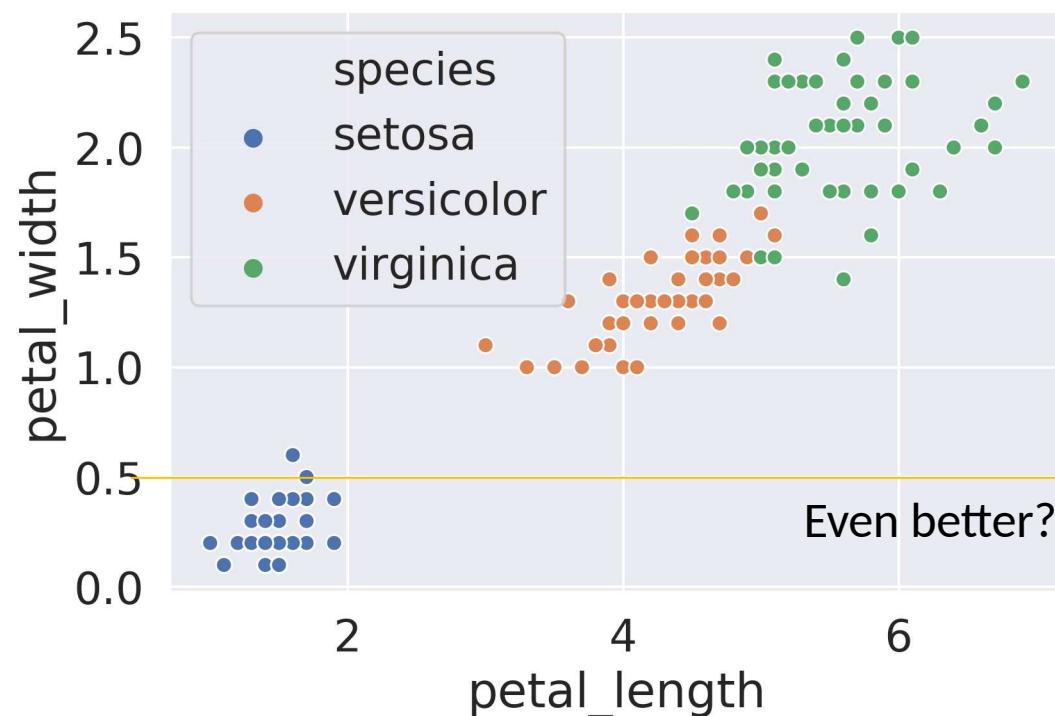


$$\Delta\text{WS} = 0.71$$

## Defining a Best Feature

Split choice #3: width  $\geq 0.5$ . Compute entropy of child nodes:

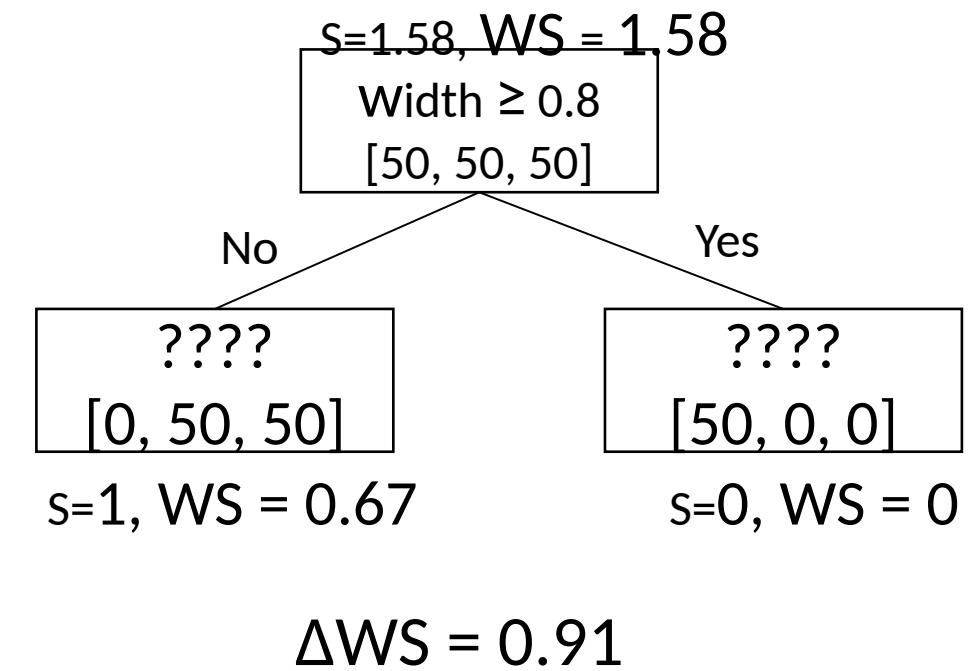
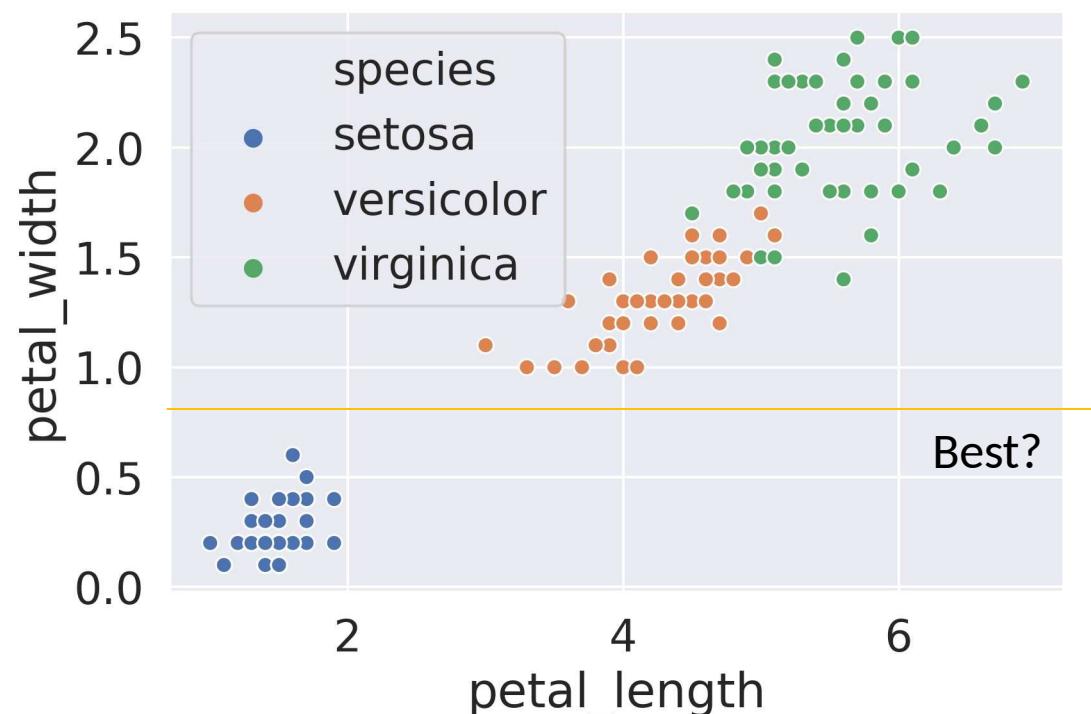
- $\text{entropy}([2, 50, 50]) = 1.12$ , weighted entropy =  $0.68 * 1.12 = 0.76$
- $\text{entropy}([48]) = 0$ , weighted entropy = 0
- $\Delta WS = 1.58 - 0.76 = 0.82$ . Better than split choice #2 (had  $\Delta WS = 0.71$ ).



## Defining a Best Feature

Split choice #4: width  $\geq 0.8$ . Compute entropy of child nodes:

- $\text{entropy}([50, 50]) = 1$ , weighted entropy =  $100/150 * 1 = 0.67$
- $\text{entropy}([50]) = 0$ , weighted entropy =  $50/150 * 0 = 0$
- $\Delta WS = 1.58 - 0.67 = 0.91$ . Better than split choice #3 (had  $\Delta WS = 0.82$ ).



## Decision Tree Generation Algorithm

Traditional decision tree generation algorithm:

- All of the data starts in the root node.
- Repeat until every node is either **pure** or **unsplittable**:
  - **Pick the best feature**  $x$  and **split value**  $\beta$  such that the  $\Delta WS$  is **maximized**, e.g.  $x = \text{petal\_width}$ ,  $\beta = 0.8$  has  $\Delta WS = 0.91$ .
  - **Split data into two nodes**, one where  $x < \beta$ , and one where  $x \geq \beta$ .

47

Notes: A node that has only one samples from one class is called a “**pure**” node. A node that has overlapping data points from different classes and thus that cannot be split is called “**unsplittable**”.

47

Let's now turn our attention to avoiding overfitting.

# Overfitting and Our Algorithm

A “fully grown” decision tree built with our algorithm runs the risk of overfitting.

One idea to avoid overfitting: Don’t allow fully grown trees.

- Approach 1: Set rules to prevent full growth.
  - Don’t split nodes with < 1% of the samples
  - Don’t allow nodes to be more than 7 levels deep in the tree
- Approach 2: Allow full growth then prune branches afterwards.
  - Branch have many rules that only affect few points

Won’t discuss these in any great detail.

- There’s a completely different idea called a “random forest” that is more popular and more beautiful. Won’t discuss in our course.

**YaleNUSCollege**

**YSC2239 Lecture 5**

# Recap

- Understanding Data
- Useful **table** functions: **select**, **sort**, **where**, **drop**, **show**, **take**,  
**with\_column**, **plot**, **group**, **bar/barh**

<https://github.com/data-8/datascience/blob/8d3fb1e0791b9e072c741b6d9c8efc98d554159e/datascience/tables.py>

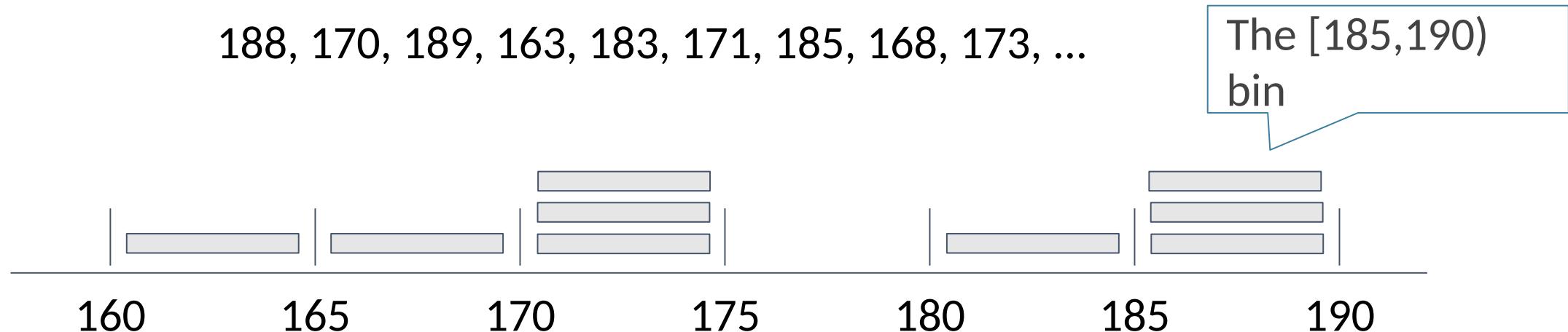
# Today's class

- Binning and Histogram
- Functions
- Apply
- Group
- Join
- Reading: Chapter 8

# Binning Numerical Values

Binning is counting the number of numerical values that lie within ranges, called bins.

- Bins are defined by their lower bounds (inclusive)
- The upper bound is the lower bound of the next bin



# Histogram

---

- Chart that displays the distribution of a numerical variable
- Uses bins; there is one bar corresponding to each bin
- Uses the area principle:
  - The *area* of each bar is the percent of individuals in the corresponding bin

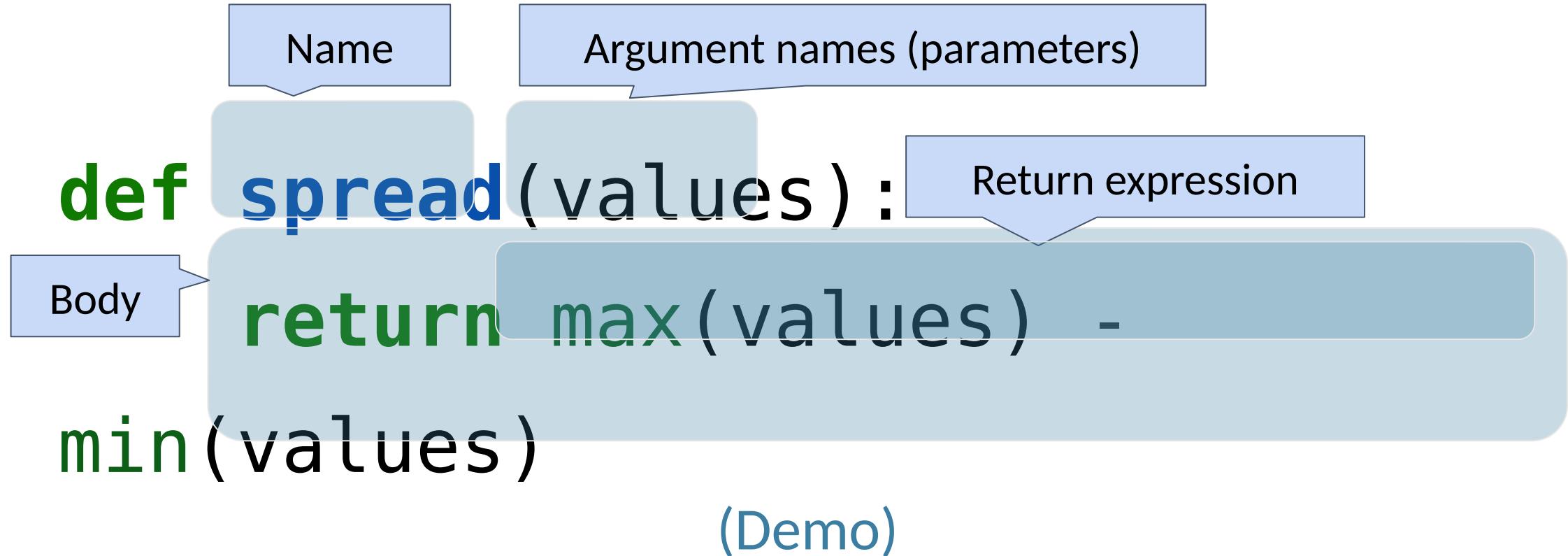
(Demo)

---

# Defining Functions

# Def Statements

User-defined functions give names to blocks of code



# Apply with Multiple Columns

# Apply

---

The `apply` method creates an array by calling a function on every element in one or more input columns

- First argument: Function to apply
- Other arguments: The input column(s)

```
table_name.apply(one_arg_function, 'column_label')
```

```
table_name.apply(two_arg_function,  
                 'column_label_for_first_arg',  
                 'column_label_for_second_arg')
```

`apply` called with only a function applies it to each row

# Grouping by One Attribute

# Grouping by One Column

---

The `group` method aggregates all rows with the same value for a column into a single row in the resulting table.

- First argument: Which column to group by
- Second argument: (Optional) How to combine values
  - `len` — number of grouped values (default)
  - `list` — list of all grouped values
  - `sum` — total of all grouped values

(Demo)

---

# Lists

# Lists are Generic Sequences

---

A list is a sequence of values (just like an array),  
but the values can have different types

```
[2+3, 'four', Table().with_column('K', [3, 4])]
```

---

- Lists can be used to create table rows.
- If you create a table column from a list, it will be converted to an array automatically
- Built into python (you don't need numpy.)

(Demo)

---

# Cross-Classification

# Grouping By Multiple Columns

---

The `group` method can also aggregate all rows that share the combination of values in multiple columns

- First argument: A list of which columns to group by
- Second argument: (Optional) How to combine values

(Demo)

# Pivot Tables

# Pivot

---

- Cross-classifies according to two categorical variables
- Produces a grid of counts or aggregated values
- Two required arguments:
  - First: variable that forms column labels of grid
  - Second: variable that forms row labels of grid
- Two optional arguments (include **both** or **neither**)
  - `values='column_label_to_aggregate'`
  - `collect='function_to_aggregate_with'`

(Demo)

---

# Group or Pivot?

---

- Distribution of one categorical variable → `.group()`
- Cross-classification of two or more categorical variables:
  - One row per combination → `.group()`
  - One variable vertically, one horizontally → `.pivot()`

# Challenge Question

1. For each city, what's the height of the tallest building for each material?
2. For each city, what's the height difference between the tallest steel building and the tallest concrete building?

sky

	name	material	city	height	age
	Metropolitan Tower	concrete	New York City	218.24	35
	Paul Hastings Tower	steel	Los Angeles	213.06	49
	Barclay Tower	concrete	New York City	205.06	13
	Westin Peachtree Plaza	concrete	Atlanta	220.37	44

(Demo)

# Joins

# Joining Two Tables

```
drinks.join('Cafe', discounts, 'Location')
```

Match rows in  
this table ...

... using values  
in this column ...

... with rows in  
that table ...

... using values  
in that column.

Columns from  
both tables

drinks

Drink	Cafe	Price
Milk Tea	Asha	5.5
Espresso	Strada	1.75
Latte	Strada	3.25
Espresso	FSM	2

discounts

Coupon	Location
10%	Asha
25%	Strada
5%	Asha

The joined column is  
sorted automatically

Cafe	Drink	Price	Coupon
Asha	Milk Tea	5.5	10%
Asha	Milk Tea	5.5	5%
Strada	Espresso	1.75	25%
Strada	Latte	3.25	25%

# Important Table Methods

---

`t.select(column, ...)` or `t.drop(column, ...)`

`t.take([row, ...])` or `t.exclude([row, ...])`

`t.sort(column, descending=False, distinct=False)`

`t.where(column, are.condition(...))`

`t.apply(function, column, ...)`

`t.group(column)` or `t.group(column, function)`

`t.group([column, ...])` or `t.group([column, ...], function)`

`t.pivot(cols, rows)` or `t.pivot(cols, rows, vals, function)`

`t.join(column, other_table, other_table_column)`

# Reminders

- Assignment 2

**YaleNUSCollege**

**YSC2239 Lecture 6**

# Today's class

- Iteration
- Chance/Probability
- Reading: Chapter 8, 9

# Important Table Methods

---

`t.select(column, ...)` or `t.drop(column, ...)`

`t.take([row_num, ...])` or `t.exclude([row_num, ...])`

`t.sort(column, descending=False, distinct=False)`

`t.where(column, are.condition(...))`

`t.apply(function_name, column, ...)`

`t.group(column)` or `t.group(column, function_name)`

`t.group([column, ...])` or `t.group([column, ...], function_name)`

`t.pivot(cols, rows)` or `t.pivot(cols, rows, vals, function_name)`

`t.join(column, other_table, other_table_column)`

---

# Comparison and Booleans

# Comparison Operators

The result of a comparison expression is a **bool** value

`x = 2`

`y = 3`

Assignment  
statements

`x > 1`

`x > y`

`y >= 3`

`x == y`

`x != 2`

`2 < x < 5`

Comparison  
expressions

# Aggregating Comparisons

---

Summing an array or list of bool values will count the True values only.

`1 + 0 + 1 == 2`

`True + False + True == 2`

`sum([1, 0, 1]) == 2`

`sum([True, False, True]) == 2`

(Demo)

---

# Control Statements

# Control Statements

---

These statements *control* the sequence of computations that are performed in a program

- The keywords **if** and **for** begin control statements
- The purpose of **if** is to define functions that choose different behavior based on their arguments

(Demo)

---

# Random Selection

# Random Selection

---

## `np.random.choice`

- Selects uniformly at random
- with replacement
- from an array,
- a specified number of times

## `np.random.choice(some_array, sample_size)`

(Demo)

---

# Appending Arrays

# A Longer Array

---

- **`np.append(array_1, value)`**
  - new array with value appended to array\_1
  - value has to be of the same type as elements of array\_1
- **`np.append(array_1, array_2)`**
  - new array with array\_2 appended to array\_1
  - array\_2 elements must have the same type as array\_1 elements

(Demo)

---

# Iteration

# for Statements

---

- **for** is a keyword that begins a control statement
- The purpose of **for** is to perform a computation for every element in a list or array

(Demo)

---

Optional: Advanced where

# A Closer Look at `where`

---

`t.where(array_of_bool_values)`

returns a table  
with only the rows of `t` for which  
the corresponding `bool` is `True`.

(Demo)

---

# Probability

# Basics

---

- Lowest value: 0
  - Chance of event that is impossible
- Highest value: 1 (or 100%)
  - Chance of event that is certain
- If an event has chance 70%, then the chance that it doesn't happen is
  - $100\% - 70\% = 30\%$
  - $1 - 0.7 = 0.3$

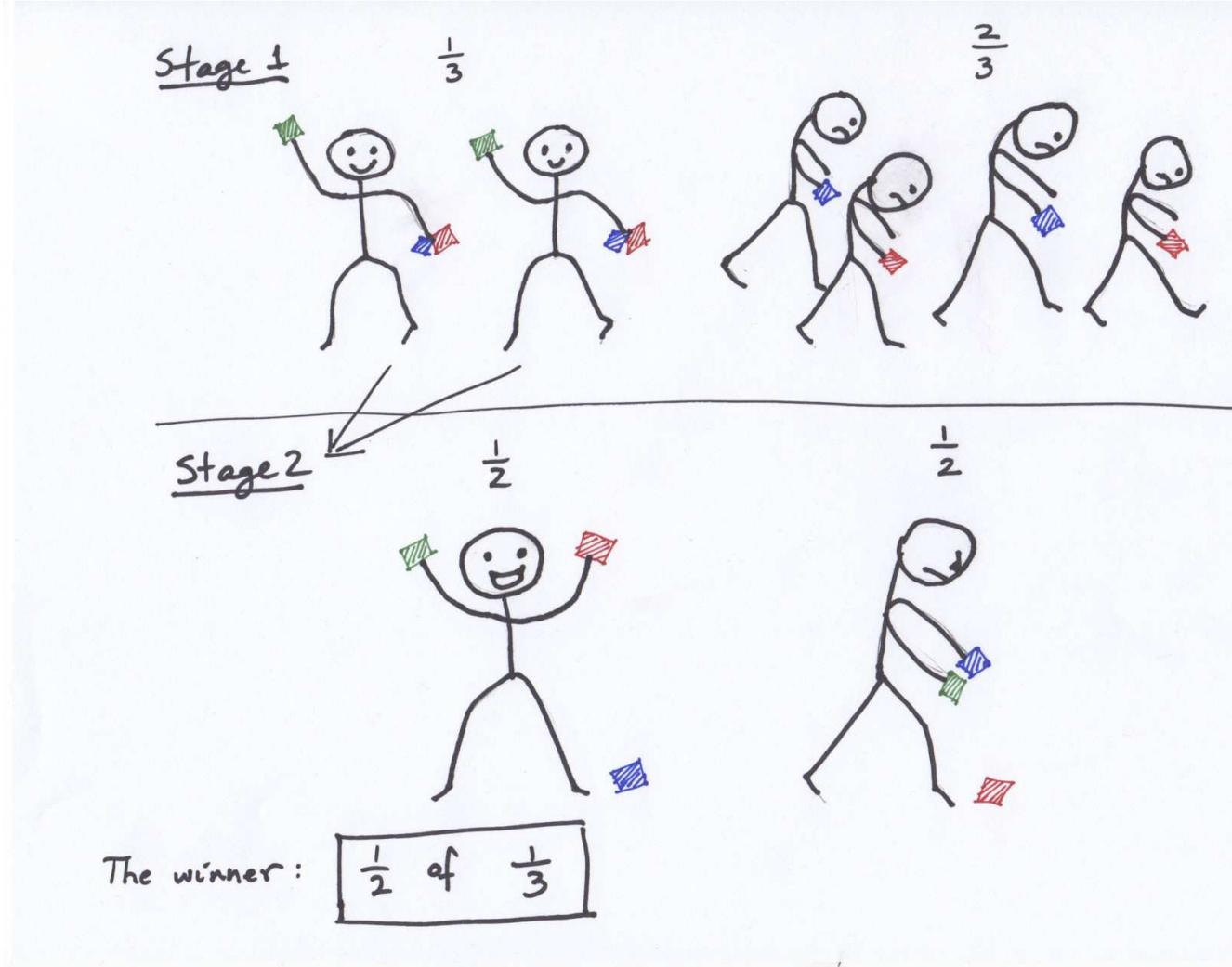
# Equally Likely Outcomes

---

Assuming all outcomes are equally likely, the chance of an event A is:

$$P(A) = \frac{\text{number of outcomes that make A happen}}{\text{total number of outcomes}}$$

# Fraction of a Fraction



# Multiplication Rule

---

Chance that two events  $A$  and  $B$  both happen

=  $P(A \text{ happens}) \times P(B \text{ happens given that } A \text{ has happened})$

- The answer is *less than or equal* to each of the two chances being multiplied
- The more conditions you have to satisfy, the less likely you are to satisfy them all

# Addition Rule

---

If event A can happen in *exactly one* of two ways, then

$$P(A) = P(\text{first way}) + P(\text{second way})$$

- The answer is *greater than or equal* to the chance of each individual way
-

# Example: At Least One Head

---

- In 3 tosses:
  - Any outcome *except* TTT
  - $P(\text{TTT}) = (\frac{1}{2}) \times (\frac{1}{2}) \times (\frac{1}{2}) = \frac{1}{8}$
  - $P(\text{at least one head}) = 1 - P(\text{TTT}) = \frac{7}{8} = 87.5\%$
- In 10 tosses:
  - $1 - (\frac{1}{2})^{10}$
  - 99.9%

**YaleNUSCollege**

**YSC2239 Lecture 7**

# Recap on previous class

- Chance / Probability: Multiplication Rule and Addition Rule
- Python:
  - Join tables
  - Booleans: True or False, comparison operators
  - Control statements (if else/ for)
  - Append array: np.append
  - Random choice: np.random.choice

# Today's class

- Sampling
  - Assessing Models
  - Comparing Distributions
- 
- Reading: Chapter 10, 11

# Sampling

# Probability Samples

---

- Deterministic sample:
  - Sampling scheme doesn't involve chance
- Probability sample:
  - Before the sample is drawn, you have to know the selection probability of every group of people in the population
  - Not all individuals have to have equal chance of being selected

# Sample of Convenience

---

- Example: sample consists of whoever walks by
- Just because you think you're sampling “at random”, doesn't mean you are.
- If you can't figure out ahead of time
  - what's the population
  - what's the chance of selection, for each group in the population

then you don't have a random sample

---

(Demo)

# Distributions

# Probability Distribution

---

- Random quantity with various possible values
- “Probability distribution”:
  - All the possible values of the quantity
  - The probability of each of those values
- If you can do the math, you can work out the probability distribution can without ever simulating the random quantity

# Empirical Distribution

---

- “Empirical”: based on observations
- Observations can be from repetitions of an experiment
- “Empirical Distribution”
  - All observed values
  - The proportion of times each value appears

(Demo)

---

# Large Random Samples

# Law of Averages

---

If a chance experiment is repeated many times, independently and under the same conditions, then the proportion of times that an event occurs gets closer to the theoretical probability of the event

As you increase the number of rolls of a die, the proportion of times you see the face with five spots gets closer to  $1/6$

(Demo)

---

# Empirical Distribution of a Sample

---

If the sample size is large,

then the empirical distribution of a uniform random sample

resembles the distribution of the population,

with high probability

# A Statistic

# Terminology

---

- **Parameter**
  - A number associated with the population
- **Statistic**
  - A number calculated from the sample

A statistic can be used as an **estimate** of a parameter, or to **test hypotheses** about how the data were generated

(Demo)

---

# Inference

---

- **Statistical Inference:**  
Making conclusions based on data in random samples

- **Example:**

Use the data to guess the value of an unknown number

fixed

depends on the random sample

Create an **estimate** of the unknown quantity

# Probability Distribution of a Statistic

---

- Values of a statistic vary because random samples vary
- “Sampling distribution” or “probability distribution” of the statistic:
  - All possible values of the statistic,
  - and all the corresponding probabilities
- Can be hard to calculate
  - Either have to do the math
  - Or have to generate all possible samples and calculate the statistic based on each sample

# Empirical Distribution of a Statistic

---

- Empirical distribution of the statistic:
  - Based on simulated values of the statistic
  - Consists of all the observed values of the statistic,
  - and the proportion of times each value appeared
- Good approximation to the probability distribution of the statistic
  - if the number of repetitions in the simulation is large

# Assessing Models

# Models

---

- A model is a set of assumptions about the data

# Models

---

- A model is a set of assumptions about the data
- In data science, many models involve assumptions about processes that involve randomness
  - “Chance models”

# Jury Selection

# Jury Selection in Alameda County

---

RACIAL AND ETHNIC DISPARITIES

IN

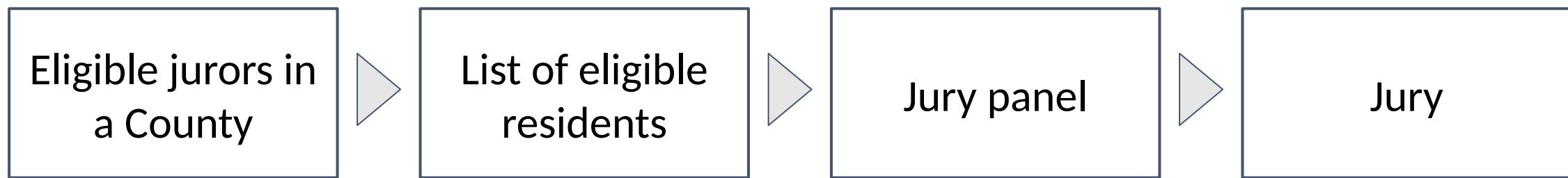
ALAMEDA COUNTY JURY POOLS

A Report by the ACLU of Northern California

October 2010

# Jury Panels

---



Section 197 of California's Code of Civil Procedure says, "All persons selected for jury service shall be selected at random, from a source or sources inclusive of a representative cross section of the population of the area served by the court."

(Demo)

---

# Two Viewpoints

# Model and Alternative

---

- Model:
  - The people on the jury panels were selected at random from the eligible population
- Alternative viewpoint:
  - No, they weren't

# A New Statistic

# Distance Between Distributions

---

- People on the panels are of multiple ethnicities
  - Distribution of ethnicities is categorical
- 
- To see whether the distribution of ethnicities of the panels is close to that of the eligible jurors, we have to measure the distance between two categorical distributions

(Demo)

# Total Variation Distance

---

Every distance has a computational recipe

**Total Variation Distance (TVD):**

- For each category, compute the difference in proportions between two distributions
- Take the absolute value of each difference
- Sum, and then divide the sum by 2

(Demo)

---

# Summary

# Summary of the Method

---

To assess whether a sample was drawn randomly from a known categorical distribution:

- Use TVD as the statistic because it measures the distance between categorical distributions
- Sample at random from the population and compute the TVD from the random sample; repeat numerous times
- Compare:
  - Empirical distribution of simulated TVDs
  - Actual TVD from the sample in the study

# Reminders

- Assignment 3

**YaleNUSCollege**

**YSC2239 Lecture 8**

# Recap

- Population, Sample, Parameter, Statistic
- Probability Distribution, Empirical Distribution
- Case study: jury selection (recommended)
- `sample_proportions(sample_size, probabilities)`

# Today's class

- Decisions and Uncertainty
  - p-values
  - A/B testing
- 
- Reading: Chapter 11, 12

# Decisions and Uncertainty

# Incomplete Information

---

- We are trying to choose between two views of the world, based on data in a sample.
- It is not always clear whether the data are consistent with one view or the other.
- Random samples can turn out quite extreme. It is unlikely, but possible.

# Terminology

# Testing Hypotheses

---

- A test chooses between two views of how data were generated
- E.g. Alameda jury panel in the previous lecture: truly random panel or is it a biased panel?
- The views are called **hypotheses**
- The test picks the hypothesis that is better supported by the observed data

# Null and Alternative

---

The method only works if we can simulate data under one of the hypotheses.

- **Null hypothesis**
  - A well defined chance model about how the data were generated
  - We can simulate data under the assumptions of this model – “under the null hypothesis”
- **Alternative hypothesis**
  - A different view about the origin of the data

# Test Statistic

---

- The statistic that we choose to simulate, to decide between the two hypotheses
- E.g. total variation distance between the proportion of a randomly generated jury panel and the population proportion in Alameda

Questions before choosing the statistic:

- What values of the statistic will make us lean towards the null hypothesis?
- What values will make us lean towards the alternative?
  - Preferably, the answer should be just “high”. Try to avoid “both high and low”.

# Prediction Under the Null Hypothesis

---

- Simulate the test statistic under the null hypothesis; draw the histogram of the simulated values
- This displays the **empirical distribution of the statistic under the null hypothesis**
- It is a prediction about the statistic, made by the null hypothesis
  - It shows all the likely values of the statistic
  - Also how likely they are (**if the null hypothesis is true**)
- The probabilities are approximate, because we can't generate all the possible random samples

# Conclusion of the Test

---

Resolve choice between null and alternative hypotheses

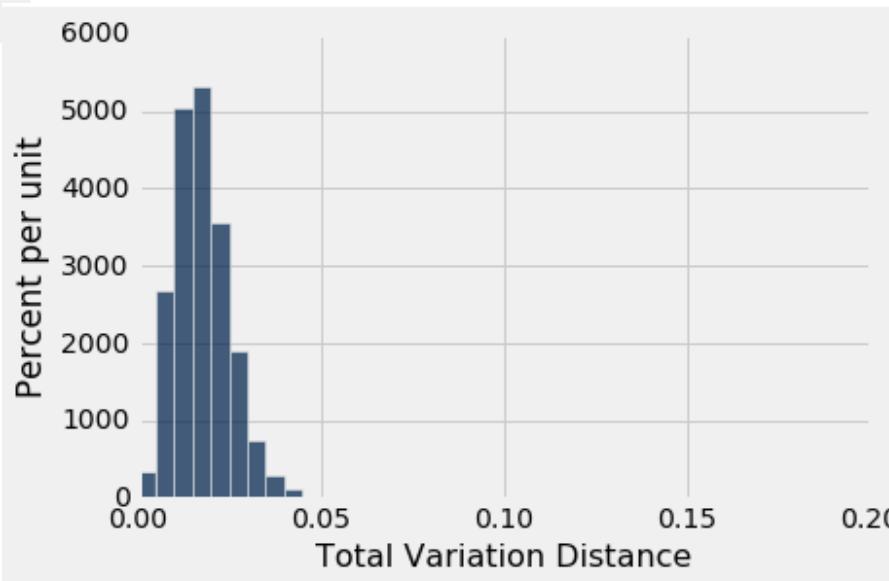
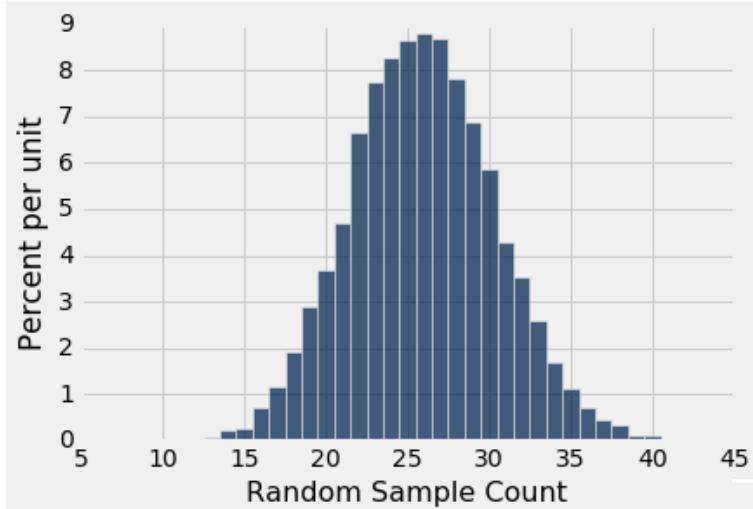
- Compare the **observed test statistic** and its empirical distribution under the null hypothesis
- If the observed value is **not consistent** with the distribution, then the test favors the alternative – “rejects the null hypothesis”

Whether a value is consistent with a distribution:

- A visualization may be sufficient
- If not, there are conventions about “consistency”

# Statistical Significance

# Tail Areas



# Conventions About Inconsistency

---

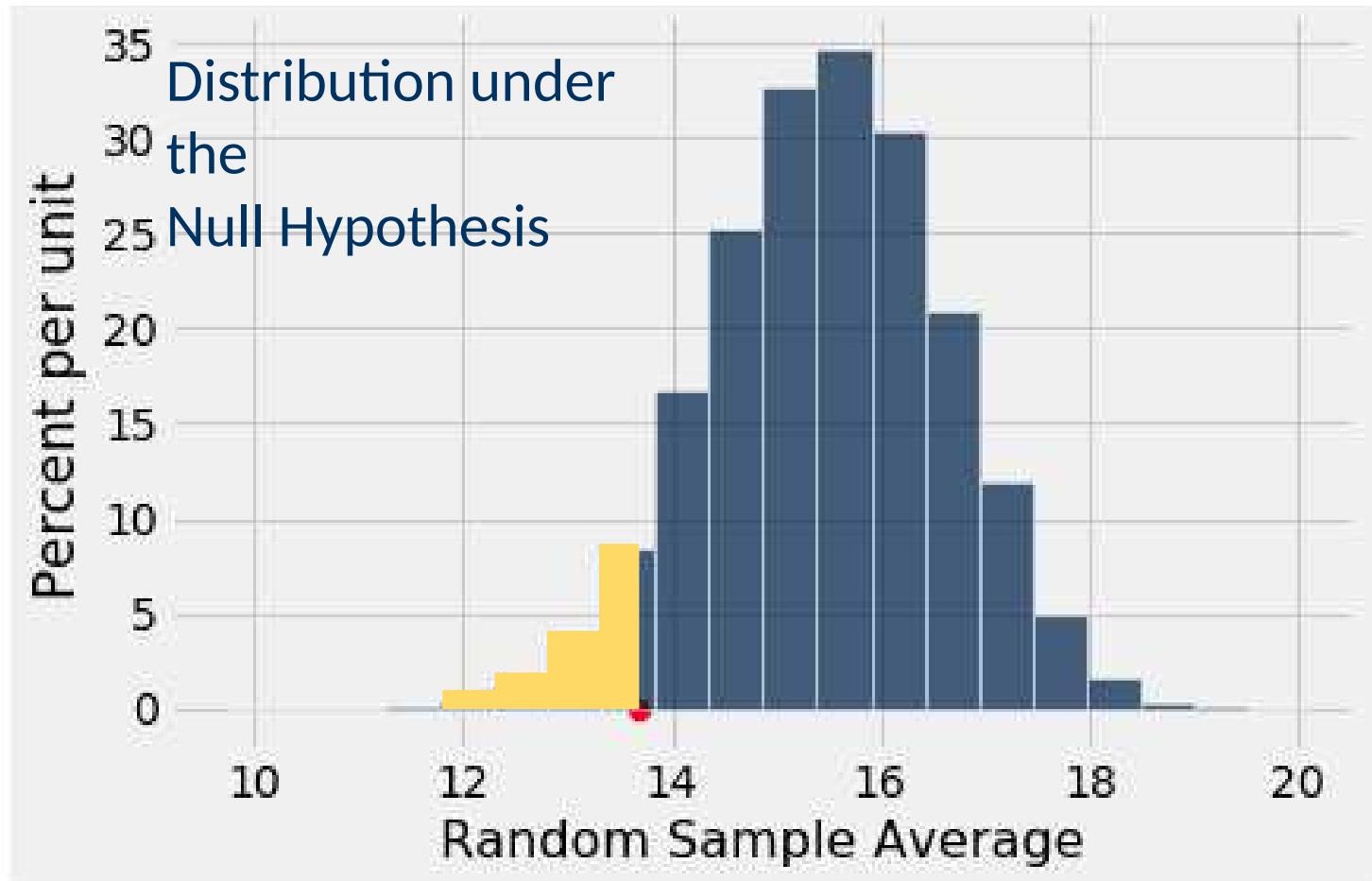
- “**Inconsistent**”: The test statistic is in the tail of the empirical distribution under the null hypothesis
- “**In the tail,**” first convention:
  - The area in the tail is less than 5%
  - The result is “statistically significant”
- “**In the tail,**” second convention:
  - The area in the tail is less than 1%
  - The result is “highly statistically significant”

(Demo)

---

# The p-Value as an Area

- Empirical distribution of the test statistic **under the null hypothesis.**
- Red dot denotes the observed statistic.
- Yellow area denotes the tail probability (p-value).



# Definition of the *p*-value

---

Formal name: **observed significance level**

The *p*-value is the chance (probability),

- under the null hypothesis,
- that the test statistic
- is equal to the value that was observed in the data
- or is even further in the direction of the alternative.
- Last two bullets mean: “test statistic is at least as extreme as the observed value.”

# Origin of the Conventions

# Sir Ronald Fisher, 1890-1962

---



*"We have the duty of formulating, of summarizing, and of communicating our conclusions, in intelligible form, in recognition of the right of other free minds to utilize them in making their own decisions."*

*Ronald Fisher*

# Sir Ronald Fisher, 1925

---

“It is convenient to take this point [5%] as a limit in judging whether a deviation is to be considered significant or not.”

-- *Statistical Methods for Research Workers*

# Sir Ronald Fisher, 1926

---

“If one in twenty does not seem high enough odds, we may, if we prefer it, draw the line at one in fifty (the 2 percent point), or one in a hundred (the 1 percent point). Personally, the author prefers to set a low standard of significance at the 5 percent point ...”

# To-do

- Lab 4
- Assignment 4

**YaleNUSCollege**

**YSC2239 Lecture 9**

# Recap

- Steps for statistical tests of hypotheses
  - Null hypothesis and Alternative hypothesis
  - The test statistic and observed value of the test statistic
  - Distribution of test statistic by simulation under null hypothesis
  - Conclusion: reject or not reject (using p-value)
- p-value: the probability of the observed value or even more extreme results if null hypothesis is true. (in short: p-value if the probability of null hypothesis being true.)
- Significant level (also called alpha level)

# Today's class

- A/B Testing
- Confidence Intervals
- Reading: Chapter 12 and 13

# A/B Testing

# Comparing Two Samples

---

- Compare values of sampled individuals in Group A with values of sampled individuals in Group B.
- Question: Do the two sets of values come from the same underlying distribution?
- Answering this question by performing a statistical test is called **A/B testing**.

---

(Demo)

# The Groups and the Question

---

- Random sample of mothers of newborns. Compare:
  - (A) Birth weights of babies of mothers who smoked during pregnancy
  - (B) Birth weights of babies of mothers who didn't smoke
- Question: Could the difference be due to chance alone?

# Hypotheses

---

- Null:
  - In the population, the distributions of the birth weights of the babies in the two groups are the same. (They are different in the sample just due to chance.)
- Alternative:
  - In the population, the babies of the mothers who smoked weighed less, on average, than the babies of the non-smokers.

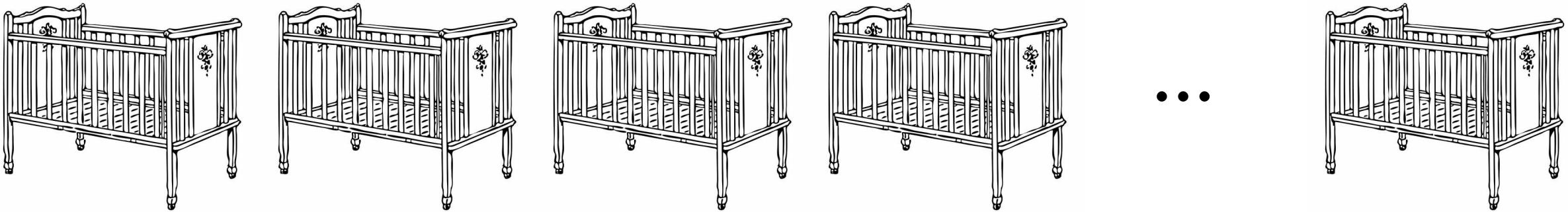
# Test Statistic

---

- Group A: smokers
- Group B: non-smokers
- Statistic: Difference between average weights  
Group A average - Group B average
- Smaller values of this statistic favor the alternative

# Simulating Under the Null

---



Non-smoker

120 oz

Non-smoker

113 oz

Smoker

128 oz

Non-smoker

136 oz

Smoker

108 oz

...

# Simulating Under the Null

---



...

Smoker

120 oz

Non-smoker

113 oz

Non-smoker

128 oz

Smoker

136 oz

Non-smoker

108 oz

# Shuffling Rows

# Random Permutation

---

- `tbl.sample(n)`
  - Table of n rows picked randomly with replacement
- `tbl.sample()`
  - Table with same number of rows as original `tbl`, picked randomly with replacement
- `tbl.sample(n, with_replacement = False)`
  - Table of n rows picked randomly without replacement
- `tbl.sample(with_replacement = False)`
  - All rows of `tbl`, in random order
  - This is what we'll use for A/B testing

(Demo)

---

# Simulating Under the Null

---

- If the null is true, all rearrangements of labels are equally likely
- Plan:
  - Shuffle all group labels
  - Assign each shuffled label to a birth weight
  - Find the difference between the averages of the two shuffled groups
  - Repeat

(Demo)

---

# A/B Tests are Hypothesis Tests

---

- Determine the 2 models (Null Hypothesis and Alternative Hypothesis)
    - Ex. Null hypothesis: In the population, the distributions of the birth weights of the babies in the two groups are the same.
  - Determine a test statistic that gives evidence for the alternative model
    - Test statistic is often (but not always) the difference or absolute difference between group means
  - Simulate the test statistic under the null hypothesis many times and store those values in an array
    - Simulated by shuffling the labels column of the table
  - Compare the **observed test statistic** and its empirical distribution under the null hypothesis
  - Draw a conclusion comparing the p-value to the p-value cutoff
-

# Percentiles

# Computing Percentiles

---

Sort the numerical set in increasing order. The 80th percentile is first value on the sorted list that is at least as large as 80% of the elements in the set

Percentile

For `s = [1, 7, 3, 9, 5]`, `percentile(80, s)` is 7

The 80th percentile is ordered element 4:  $(80/100) * 5$

For a percentile that does not exactly correspond to an element, take the next greater element instead

# The percentile Function

---

- The  $p$ th percentile is the value in a set that is at least as large as  $p\%$  of the elements in the set
- Function in the `datascience` module:  
**`percentile(p, values)`**
- **`p`** is between 0 and 100
- Returns the  $p$ th percentile of the array

# Discussion Question

---

Which are True, when `s = [1, 7, 3, 9, 5]`?

`percentile(10, s) == 0`

`percentile(39, s) == percentile(40, s)`

`percentile(40, s) == percentile(41, s)`

`percentile(50, s) == 5`

(Demo)

---

# Estimation

# Inference: Estimation

---

- How big is an unknown parameter?
- If you have a census (that is, the whole population):
  - Just calculate the parameter and you're done
- If you don't have a census:
  - Take a random sample from the population
  - Use a statistic as an **estimate** of the parameter

(Demo)

---

# Variability of the Estimate

---

- One sample → One estimate
- But the random sample could have come out differently
- And so the estimate could have been different
- Main question:
  - **How different could the estimate have been?**
- The variability of the estimate tells us something about how accurate the estimate is:  
$$\text{estimate} = \text{parameter} + \text{error}$$

# Where to Get Another Sample?

---

- One sample → One estimate
- To get many values of the estimate, we needed many random samples
- Can't go back and sample again from the population:
  - No time, no money
- Stuck?

# The Bootstrap

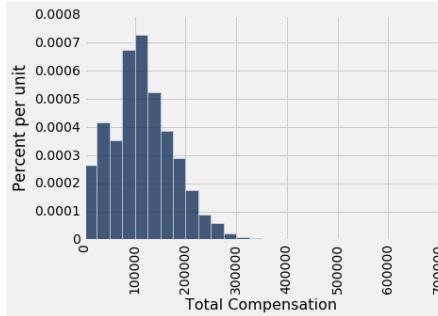
# The Bootstrap

---

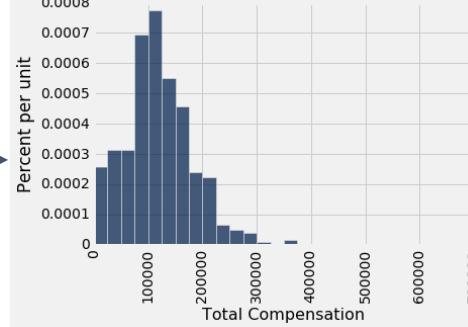
- A technique for simulating repeated random sampling
- All that we have is the original sample
  - ... which is large and random
  - Therefore, it probably resembles the population
- So we sample at random from the original sample!

# Why the Bootstrap Works

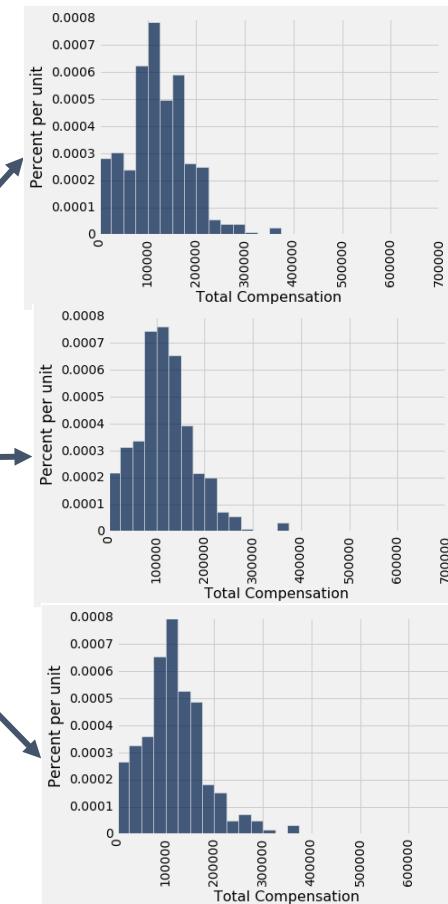
population



sample



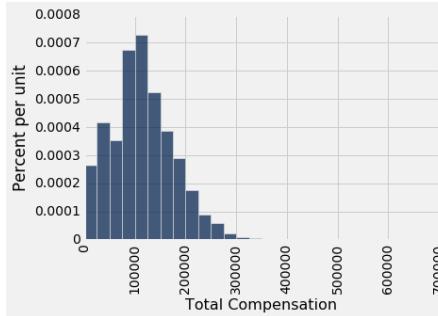
resamples



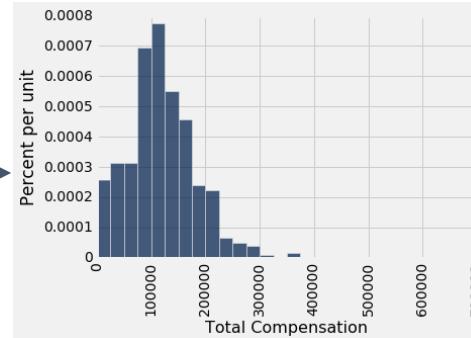
All of these look pretty similar, most likely.

# Why We Need the Bootstrap

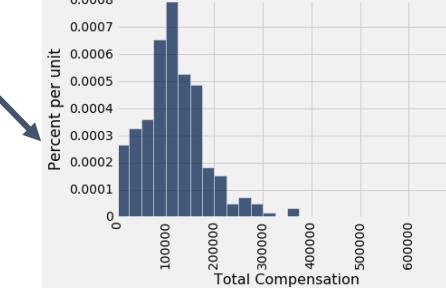
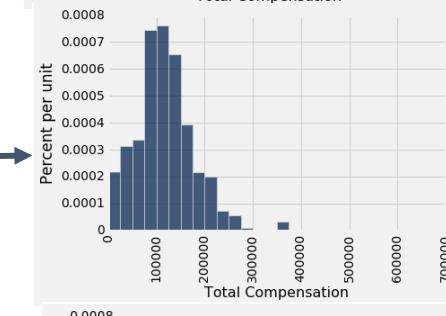
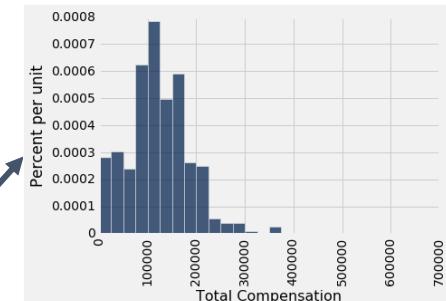
population



sample



resamples



What we wish  
we could get

What we  
really get

# Key to Resampling

---

- From the original sample,
  - draw at random
  - with replacement
  - as many values as the original sample contained
- The size of the new sample has to be the same as the original one, so that the two estimates are comparable

(Demo)

---

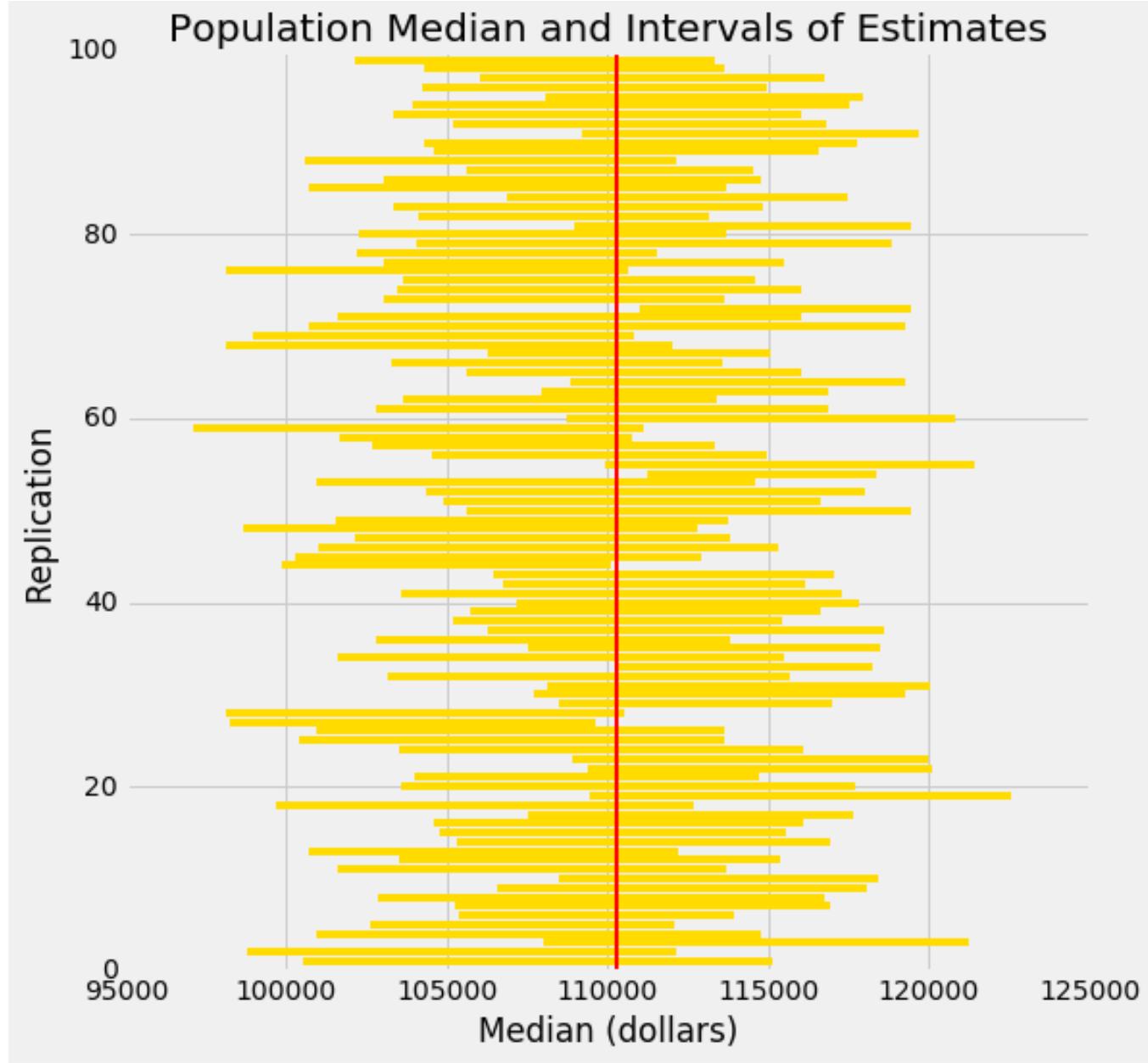
# 95% Confidence Interval

---

- Interval of **estimates of a parameter**
- Based on random sampling
- 95% is called the confidence level
  - Could be any percent between 0 and 100
  - Higher level means wider intervals
- The **confidence is in the process** that generated the interval:
  - It generates a “good” interval about 95% of the time.

(Demo)

---



Each line here is a confidence interval from a fresh sample from the population

Use Methods Appropriately

# Can You Use a CI Like This?

---

By our calculation, an approximate 95% confidence interval for the average age of the mothers in the population is (26.9, 27.6) years.

## True or False:

- About 95% of the mothers in the population were between 26.9 years and 27.6 years old.

**Answer: False.** We're estimating that their **average age** is in this interval.

---

# Is This What a CI Means?

---

An approximate 95% confidence interval for the average age of the mothers in the population is (26.9, 27.6) years.

## True or False:

- There is a 0.95 probability that the average age of mothers in the population is in the range 26.9 to 27.6 years.

**Answer: False.** The average age of the mothers in the population is unknown but it's a constant. It's not random. No chances involved.

---

# When Not to Use The Bootstrap

---

- If you're trying to estimate very high or very low percentiles, or min and max
- If you're trying to estimate any parameter that's greatly affected by rare elements of the population
- If the probability distribution of your statistic is not roughly bell shaped (the shape of the empirical distribution will be a clue)
- If the original sample is very small

(Demo)

---

# To-do

- Assignment 4