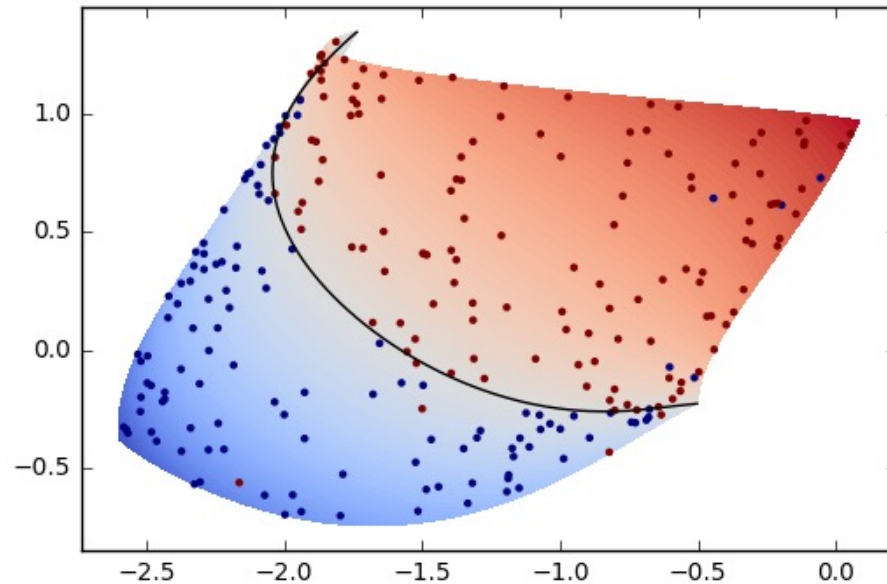


# CS178 Lecture 2: Data



Gavin Kerrigan

Fall 2022

Some materials courtesy Padhraic Smyth, Alex Ihler.

# Today's Lecture

---

Types of Machine Learning Problems

Representing and Exploring Data

Mathematical Notation

# Announcements/Reminders

---

- HW1 available now
  - Expect to have to read the documentation for numpy/matplotlib/etc.
  - See tutorial linked on Canvas calendar
  - Due Friday next week (4/14)
- Discussion sections tomorrow
  - Covering Jupyter notebooks and basics of some Python packages
  - After today's lecture + tomorrow's discussion, you should be able to do HW problem 1
- Use Ed board for all class related questions
  - We'll post occasional announcements here as well
  - Make use of TA office hours as well

# Today's Lecture

---

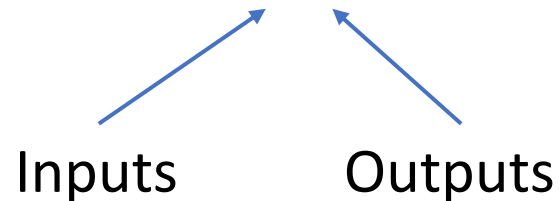
Types of Machine Learning Problems

Representing and Exploring Data

Mathematical Notation

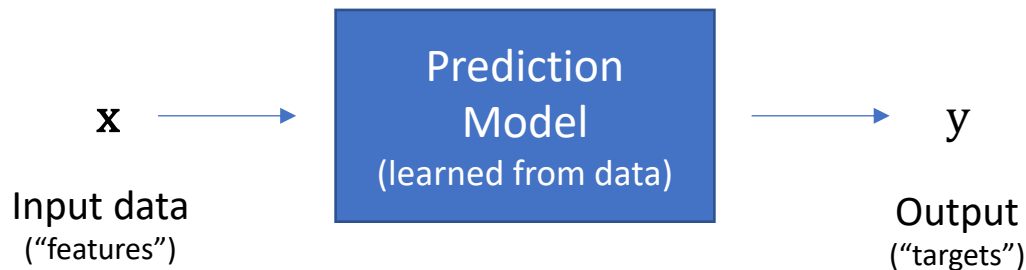
# Types of Machine Learning Problems

- Machine learning is all about understanding **data**
- What types of data can we have?
- **Supervised Learning**: data comes in  $(x, y)$  pairs



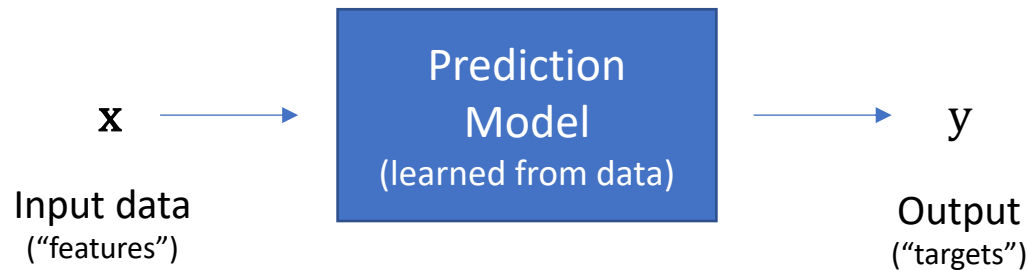
# Types of Machine Learning Problems

- Machine learning is all about understanding **data**
- What types of data can we have?
- **Supervised Learning**: data comes in  $(x, y)$  pairs



# Types of Machine Learning Problems

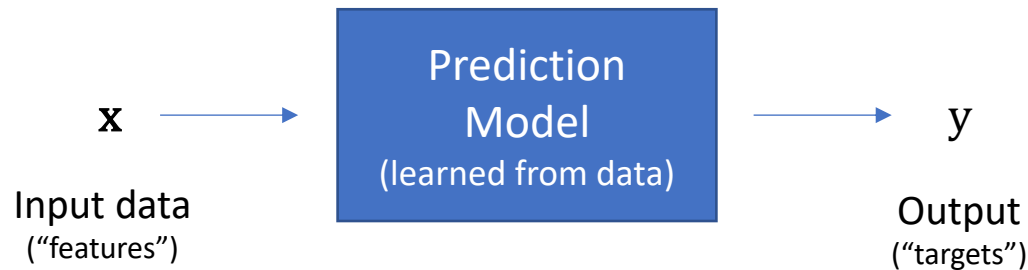
- **Supervised Learning**: data comes in  $(x, y)$  pairs



- **Regression**: the outputs  $y$  are scalars
  - e.g.  $y$  is any real-valued number,  $y > 0$ ,  $y \in [a, b]$

# Types of Machine Learning Problems

- **Supervised Learning**: data comes in  $(x, y)$  pairs

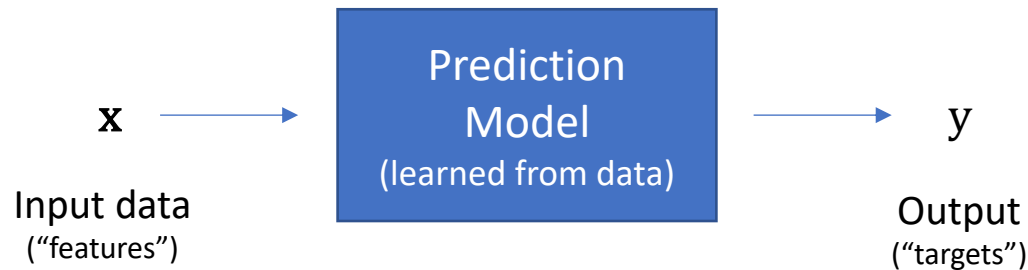


- **Regression**: the outputs  $y$  are scalars
- Example:
  - $x$  could represent features of a house:
    - square footage, number of bedrooms, location, etc.
  - $y$  could represent the price of the house, in dollars



# Types of Machine Learning Problems

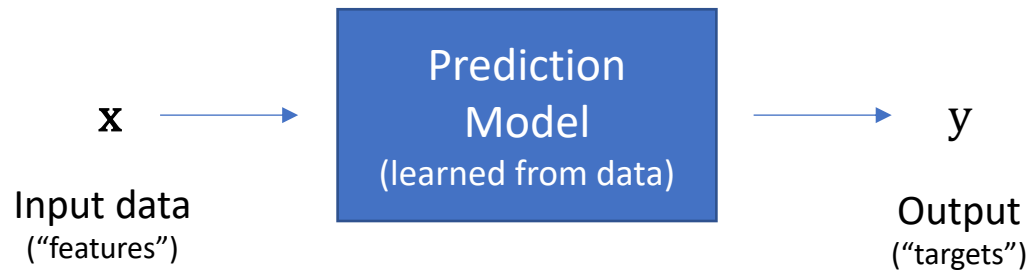
- **Supervised Learning**: data comes in  $(x, y)$  pairs



- **Regression**: the outputs  $y$  are scalars
- Example:
  - $x$  could represent weather history:
    - Yesterday's temperature, amount of rain, humidity, etc.
  - $y$  could represent tomorrow's temperature

# Types of Machine Learning Problems

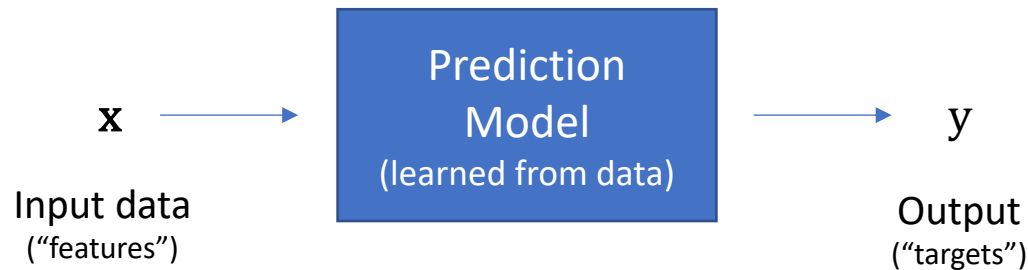
- **Supervised Learning**: data comes in  $(x, y)$  pairs



- **Classification**: the outputs  $y$  are discrete ("labels")
  - i.e.  $y$  can take any value in a finite set of  $K$  labels
  - Binary classification:  $K = 2$ , so that  $y = 0$  or  $y = 1$
  - Multi-class classification:  $K > 2$ , so that  $y = 1, y = 2, \dots, y = K$

# Types of Machine Learning Problems

- **Supervised Learning**: data comes in  $(x, y)$  pairs

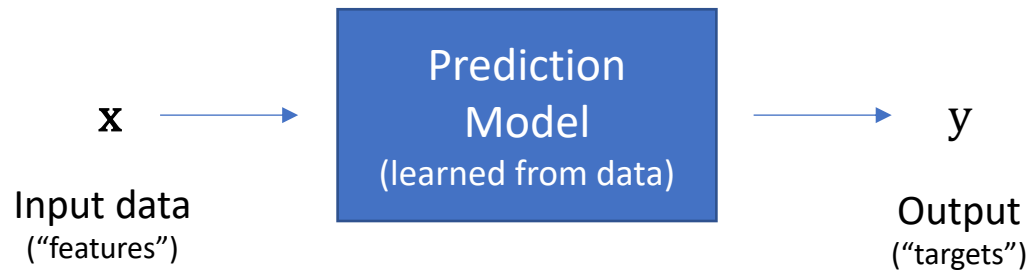


- **Classification**: the outputs  $y$  are discrete ("labels")
  - i.e.  $y$  can take any value in a finite set of  $K$  labels
- Example:

Text of email  $\longrightarrow$  Spam or not ( $K = 2$ )

# Types of Machine Learning Problems

- **Supervised Learning**: data comes in  $(x, y)$  pairs

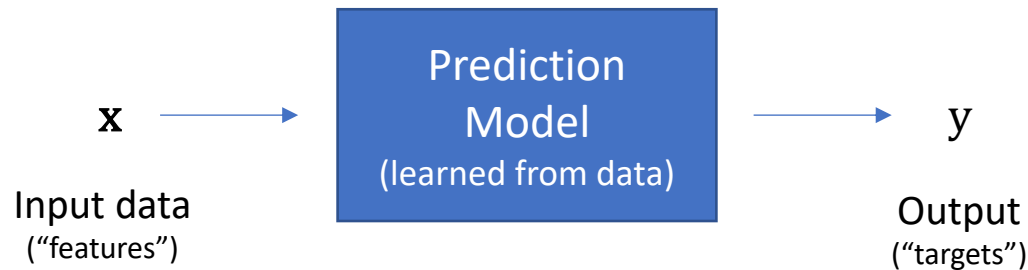


- **Classification**: the outputs  $y$  are discrete ("labels")
  - i.e.  $y$  can take any value in a finite set of  $K$  labels
- Example:

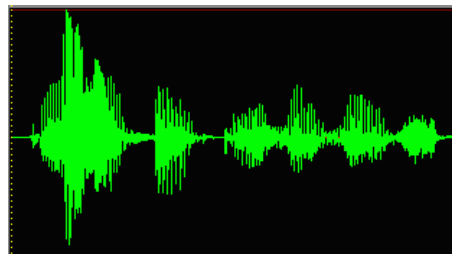
Medical records  $\longrightarrow$  Diagnosis ( $K = 2$ )

# Types of Machine Learning Problems

- **Supervised Learning**: data comes in  $(x, y)$  pairs



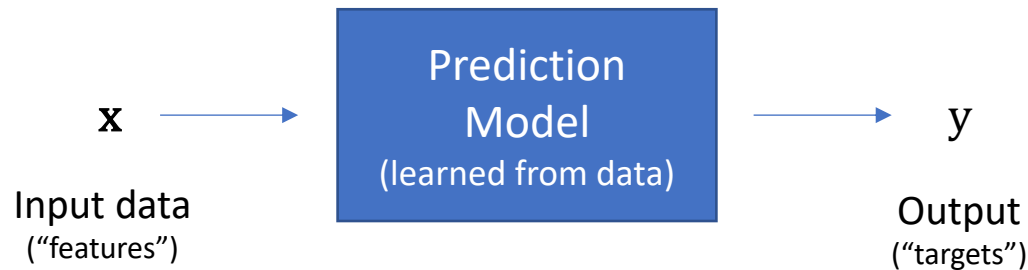
- **Classification**: the outputs  $y$  are discrete ("labels")
  - i.e.  $y$  can take any value in a finite set of  $K$  labels
- Example:



English word ( $K = 100,000$ )

# Types of Machine Learning Problems

- **Supervised Learning**: data comes in  $(x, y)$  pairs



- **Classification**: the outputs  $y$  are discrete ("labels")
  - i.e.  $y$  can take any value in a finite set of  $K$  labels

- Example:



(Wikimedia)

→ Breed of dog ( $K > 300$ )

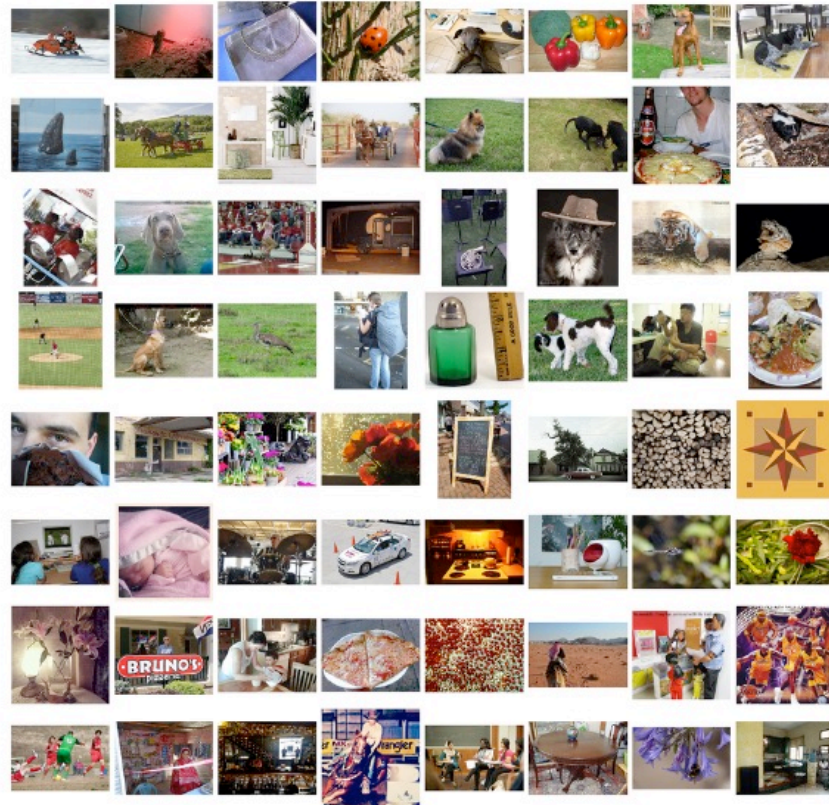
# Image Classification

## ImageNet

A testbed for evaluating  
image classification algorithms

1000 different classes

14 million images



From Russakovsky et al, ImageNet Large Scale Visual Recognition Challenge, 2015

# Error Rates over Time

---

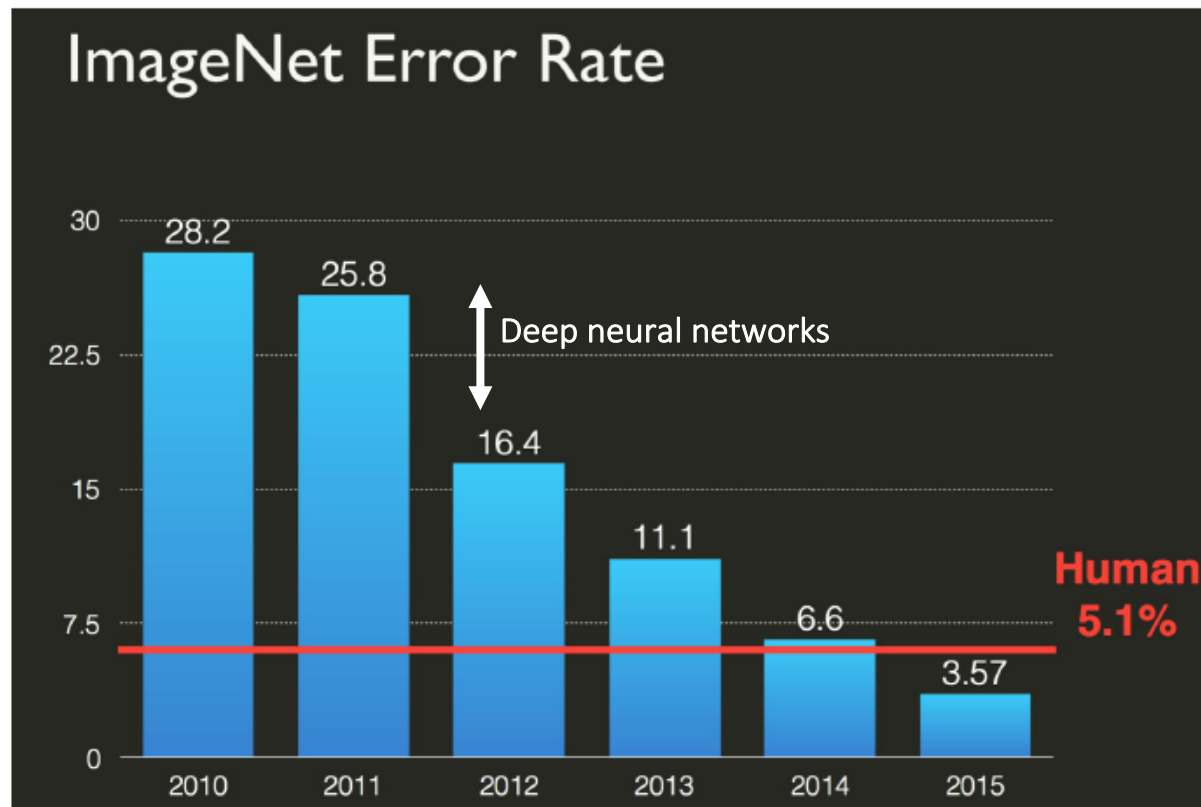
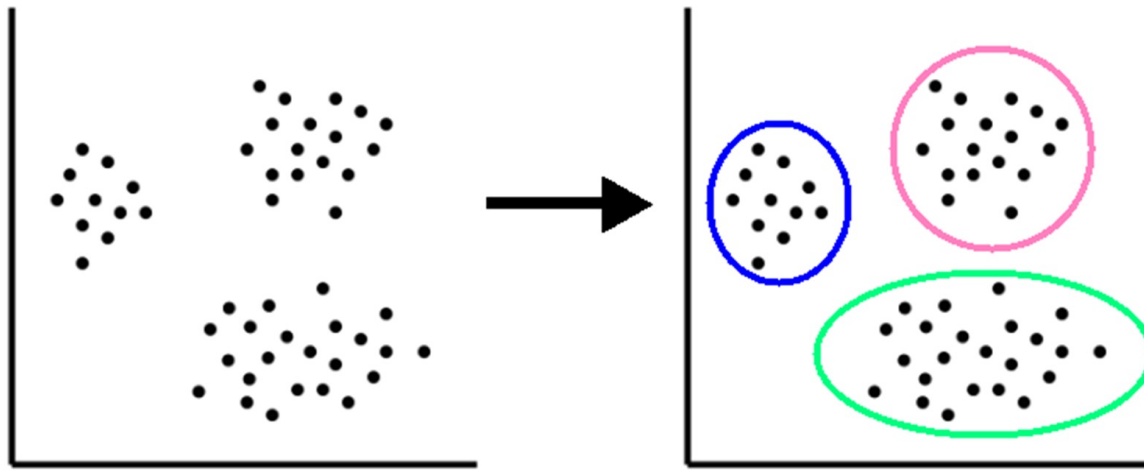


Figure from Kevin Murphy, Google, 2016



# Types of Machine Learning Problems

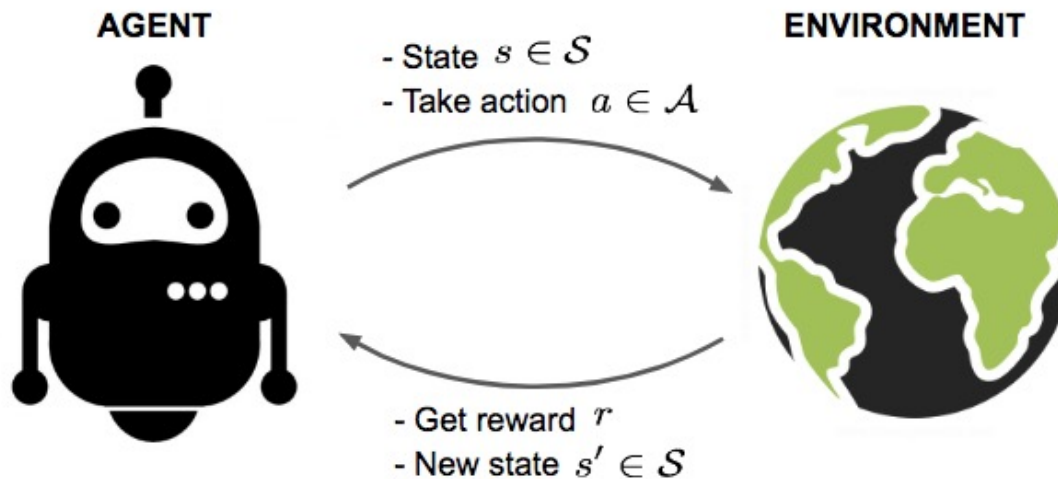
- **Unsupervised Learning**: data has no targets, only features  $x$ 
  - Targets can be expensive and time consuming to obtain
  - E.g. medical diagnosis task requires hiring a doctor
  - Typical task here is **clustering**



# Types of Machine Learning Problems

- **Reinforcement Learning:**

- An agent interacts with an environment over time
- Agent tries to maximize their received rewards



(Lilian Weng)

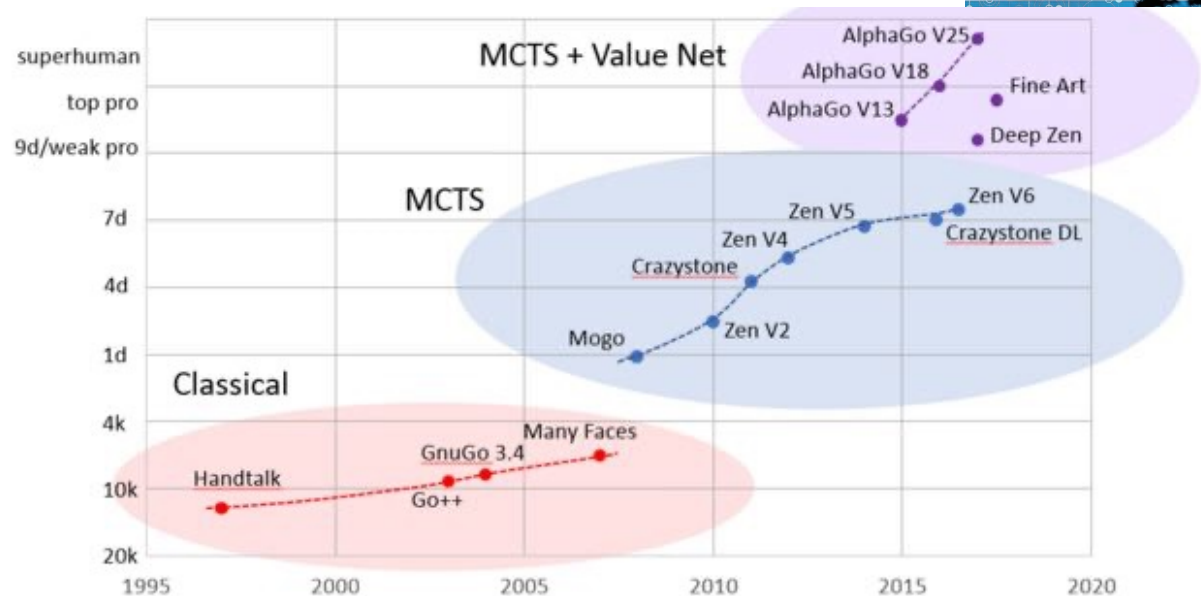
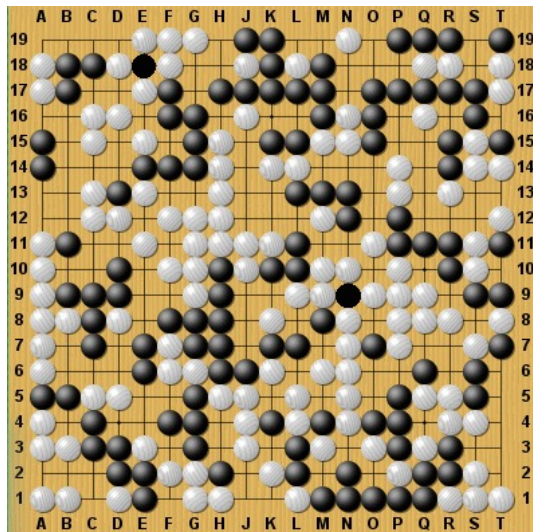
# Types of Machine Learning Problems

- **Reinforcement Learning:**

- An agent interacts with an environment over time
- Agent tries to maximize their received rewards



Positions on a Go board



[https://www.reddit.com/r/baduk/comments/6ttyyz/better\\_graph\\_of\\_go\\_ai\\_strength\\_over\\_time/](https://www.reddit.com/r/baduk/comments/6ttyyz/better_graph_of_go_ai_strength_over_time/)

# Summary of Types of Machine Learning

---

- Supervised Learning
  - $(x, y)$  pairs are provided during training
    - Classification ( $y$  is categorical)
    - Regression ( $y$  is real-valued)
- Unsupervised Learning
  - Only  $x$ 's provided, no  $y$ 's
  - Examples: clustering
- Reinforcement Learning
  - Algorithm interacts with an environment over time
- Other variants
  - Semi-supervised, online learning, ....

Questions?

# Today's Lecture

---

Types of Machine Learning Problems

Representing and Exploring Data

Mathematical Notation

# Computational Programming Environments

---

## Python



- Numpy, Matplotlib, SciPy...
- Tensorflow, PyTorch for Deep Learning (not here)

## Matlab (commercial)

## R

- Used mainly in statistics

## C/C++

- For fast performance (can be called by Python)

+ other, more specialized languages for visualization and modeling...

# Python Libraries for Computation

---

- NumPy and SciPy
  - NumPy: Basic (fast) array operations, linear algebra
  - SciPy: broader scope of numerical algorithms (e.g., optimization)
- Pandas
  - Useful for data exploration, particularly for tabular data
  - Built on top of NumPy, more high-level data manipulation
  - Often uses DataFrames
- Plotting/visualization
  - Matplotlib, pyplot, seaborn, + others
  - Flexible plotting environments for interactive data analysis
  - Homeworks will use pyplot



# Python Libraries for Machine Learning

---

- Scikit-learn (aka sklearn)
  - The standard machine learning library for Python
  - Will be the basis for most of your homeworks
  - Easy to use, flexible. Does not include deep learning
- Deep learning libraries
  - Tensorflow (Google): broad framework for deep learning
  - Keras (Google): high-level API on top of Tensorflow
  - PyTorch (Meta): More recent alternative to Tensorflow
  - All broadly used in research and industry
- All are open-source

# Data Exploration and Visualization

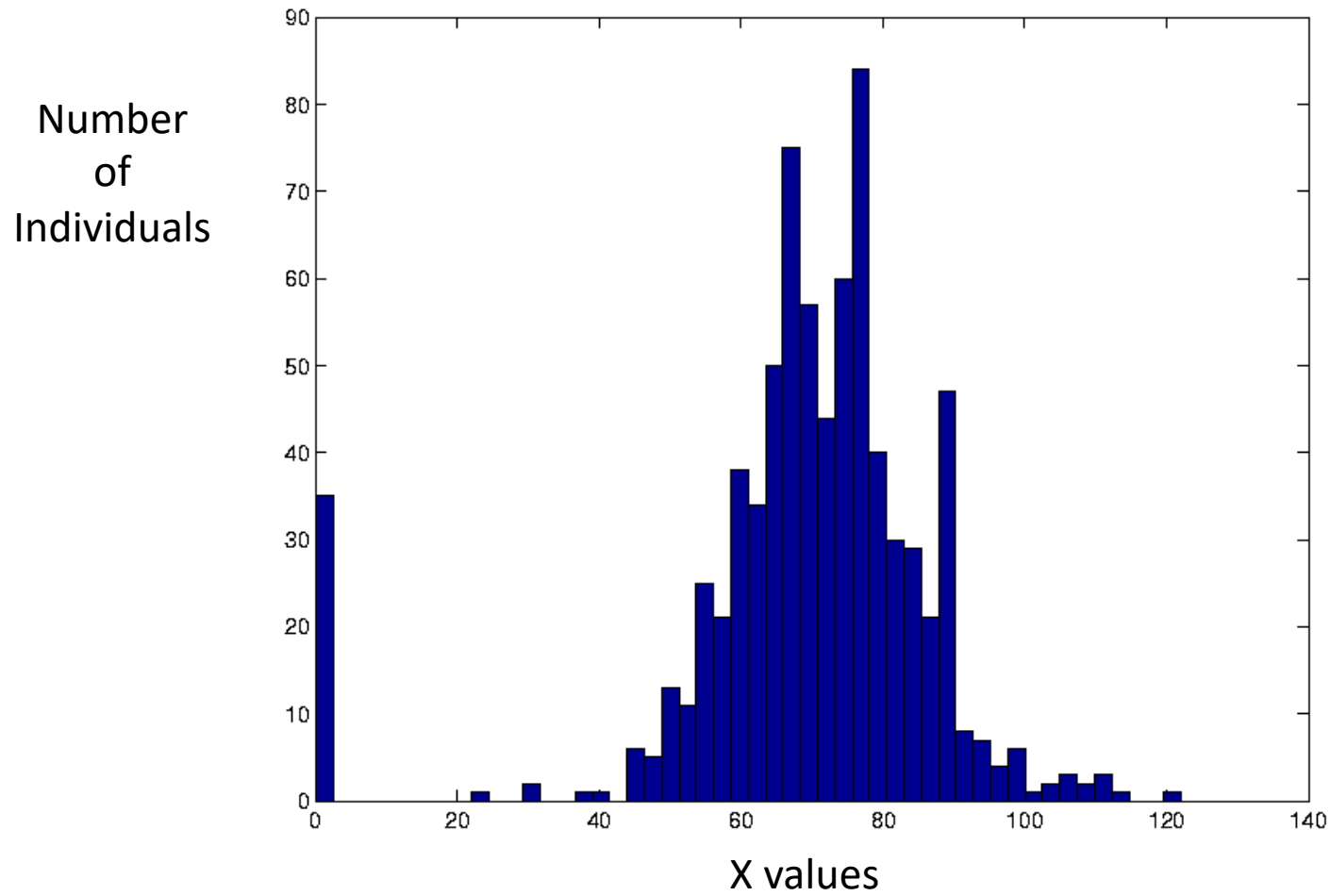
---

- In supervised learning we are interested in inputs  $x$  and outputs  $y$
- Its often useful to try to understand basic properties of the  $x$  and  $y$  data in a machine learning problem
  - e.g., by visualization, by computing summary statistics
- This is often referred to as “exploratory data analysis”
  - Can help us to better understand how  $x$  and  $y$  are related
  - Can potentially help detect problems (e.g., missing values)
- ...but note that for very large high-dimensional datasets, exploratory data analysis may not always be practical

# Data Visualization is Useful

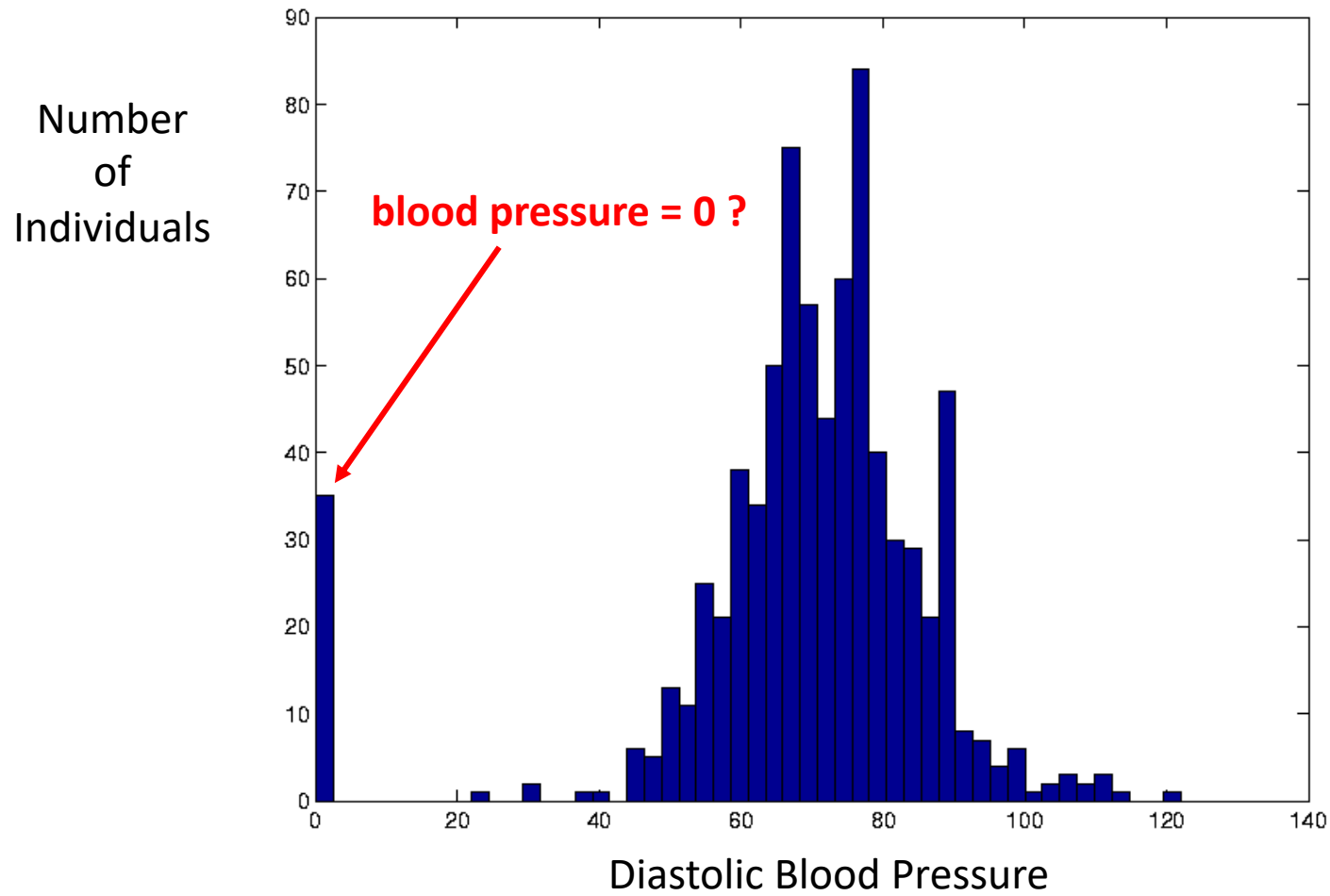
---

Widely used diabetes dataset used in machine learning

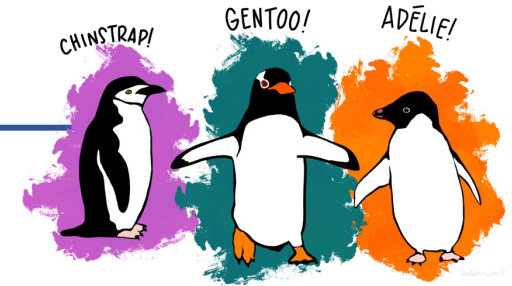


# Data Visualization is Useful

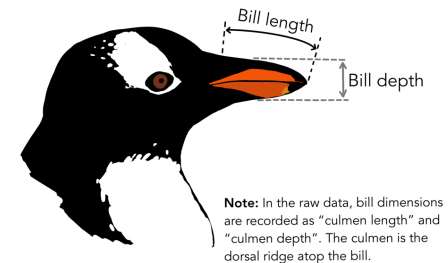
Widely used diabetes dataset used in machine learning



# The Penguin Dataset



- A small dataset useful for exploration
- Data Summary
  - 344 penguins
  - Each described by 4 real-valued features ("attributes")
    - Bill-length, bill-depth, flipper-length, body-mass
    - Other attributes per penguin: gender, year, island
  - Each penguin is in 1 of 3 classes: adelic, chinstrap, gentoo
- Natural to put data in a 344 x 4 table
  - 344 rows (penguins) by 4 columns (features)
- ..with additional 344 x 1 column of class labels



For more information see: [github.com/allisonhorst/palmerpenguins/blob/main/README.md](https://github.com/allisonhorst/palmerpenguins/blob/main/README.md)

# Penguin Data

Class Label

4 features (columns)

344 rows (penguins)

	species	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g
0	Adelie	39.1	18.7	181.0	3750.0
1	Adelie	39.5	17.4	186.0	3800.0
2	Adelie	40.3	18.0	195.0	3250.0
3	Adelie	NaN	NaN	NaN	NaN
4	Adelie	36.7	19.3	193.0	3450.0
..	...	...	...	...	...
339	Gentoo	NaN	NaN	NaN	NaN
340	Gentoo	46.8	14.3	215.0	4850.0
341	Gentoo	50.4	15.7	222.0	5750.0
342	Gentoo	45.2	14.8	212.0	5200.0
343	Gentoo	49.9	16.1	213.0	5400.0

"NaN" = "not a number"  
(standard notation for missing data)

# ... in Python

Python code snippet to generate the table on the last slide

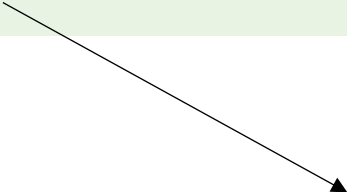
```
# set up seaborn (the plotting library we will use)
import seaborn as sns
sns.set_theme() # Apply the default theme for seaborn visualization

penguins = sns.load_dataset("penguins") # load the penguins dataset

for col in penguins.columns: # find column names for the dataset
    print(col)

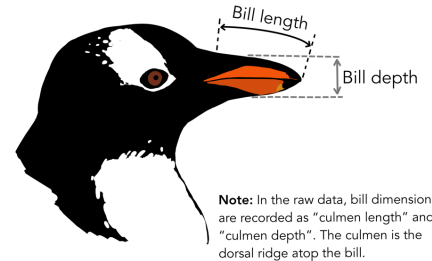
print(penguins.iloc[:, [0, 2, 3, 4, 5]]) # look at the data for selected columns
```

species  
island  
bill\_length\_mm  
bill\_depth\_mm  
flipper\_length\_mm  
body\_mass\_g  
sex



	species	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g
0	Adelie	39.1	18.7	181.0	3750.0
1	Adelie	39.5	17.4	186.0	3800.0
2	Adelie	40.3	18.0	195.0	3250.0
3	Adelie	NaN	NaN	NaN	NaN
4	Adelie	36.7	19.3	193.0	3450.0
...	...	...	...	...	...
339	Gentoo	NaN	NaN	NaN	NaN
340	Gentoo	46.8	14.3	215.0	4850.0
341	Gentoo	50.4	15.7	222.0	5750.0
342	Gentoo	45.2	14.8	212.0	5200.0
343	Gentoo	49.9	16.1	213.0	5400.0

# Summary Statistics for Penguins



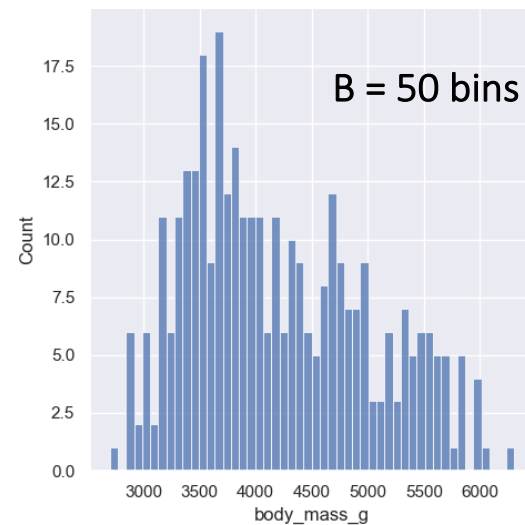
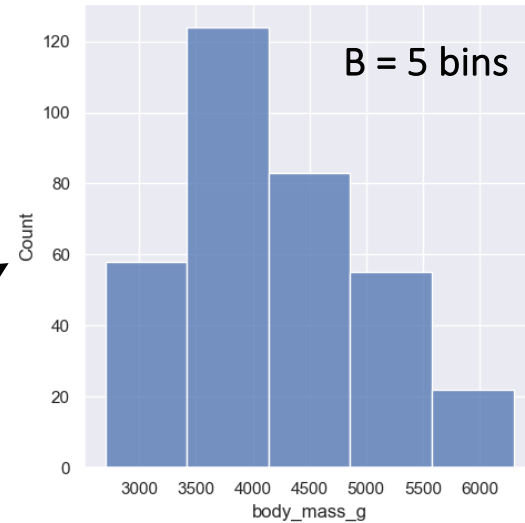
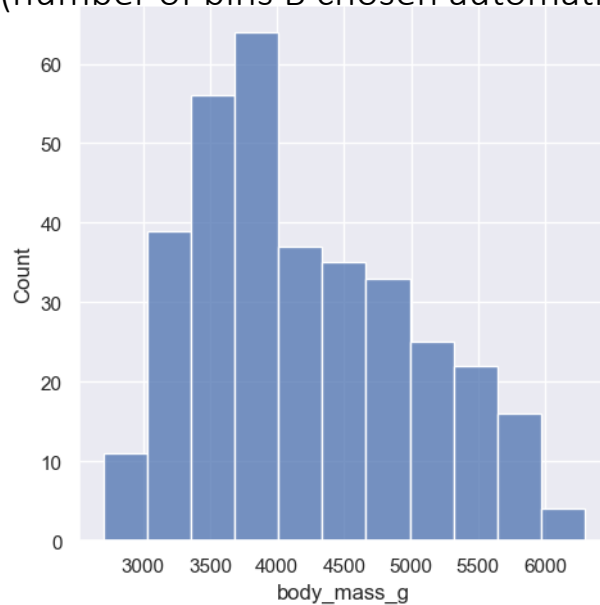
	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g
<b>count</b>	342.000000	342.000000	342.000000	342.000000
<b>mean</b>	43.921930	17.151170	200.915205	4201.754386
<b>std</b>	5.459584	1.974793	14.061714	801.954536
<b>min</b>	32.100000	13.100000	172.000000	2700.000000
<b>25%</b>	39.225000	15.600000	190.000000	3550.000000
<b>50%</b>	44.450000	17.300000	197.000000	4050.000000
<b>75%</b>	48.500000	18.700000	213.000000	4750.000000
<b>max</b>	59.600000	21.500000	231.000000	6300.000000

This table can be generated conveniently in Python using `penguins.describe()` (using the Pandas library, with the data in a dataframe)

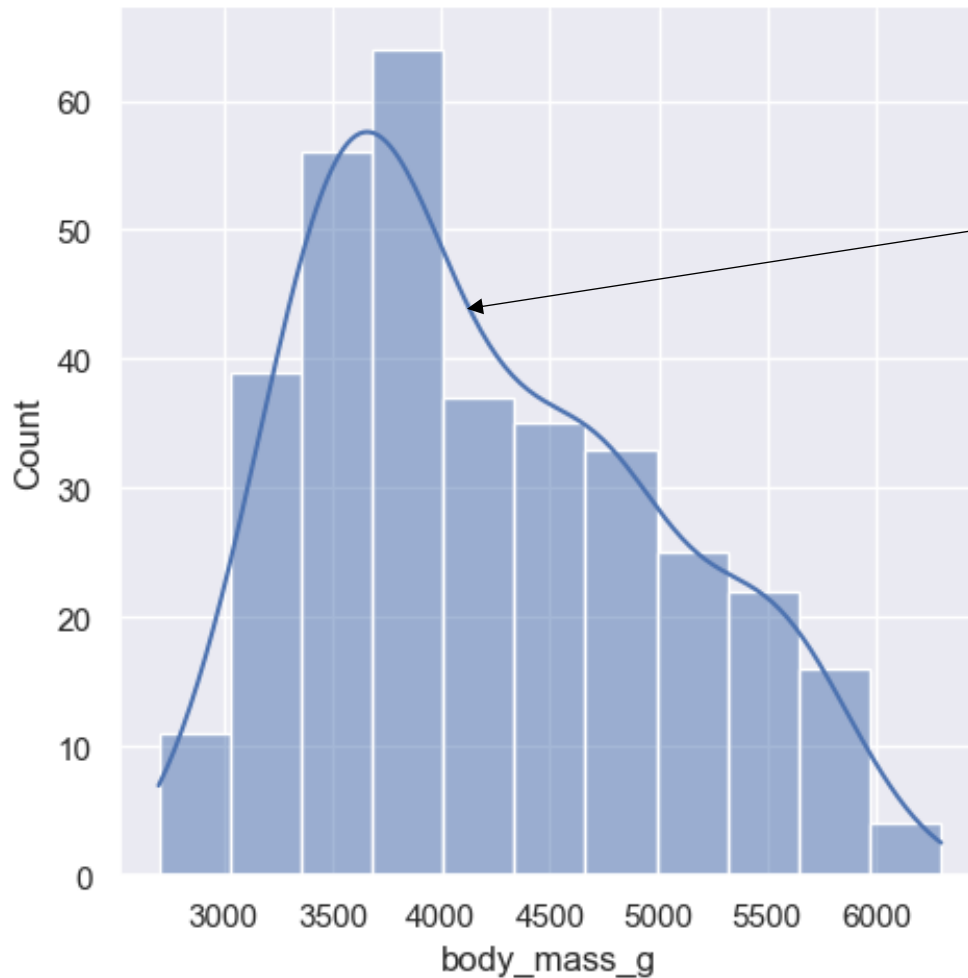


# Distributional Plots for 1 Variable: Histograms

Histogram for body\_mass\_g variable  
(number of bins  $B$  chosen automatically)



## ...adding density smoothing

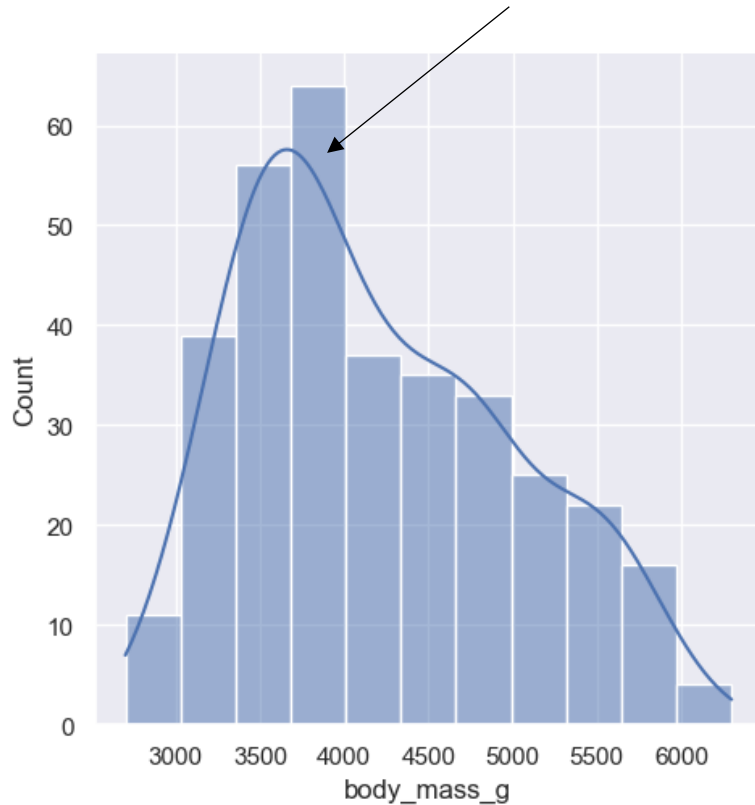


This blue curve is an estimate of the **probability density function** (pdf) of the variable, i.e.,  $P(x)$

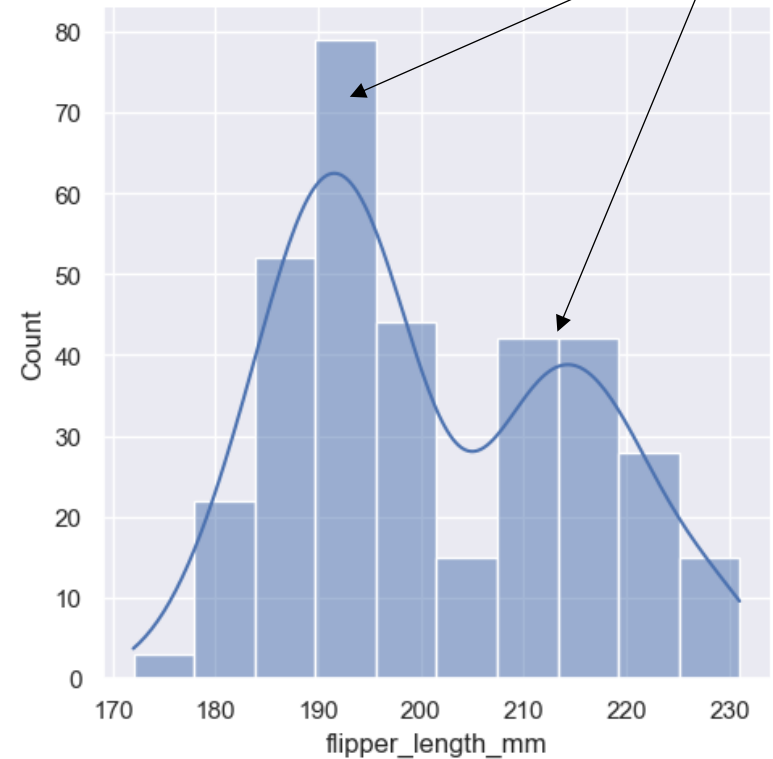
It is computed using a method called **kernel density estimation** using a smoothing parameter

# Unimodal versus Multimodal

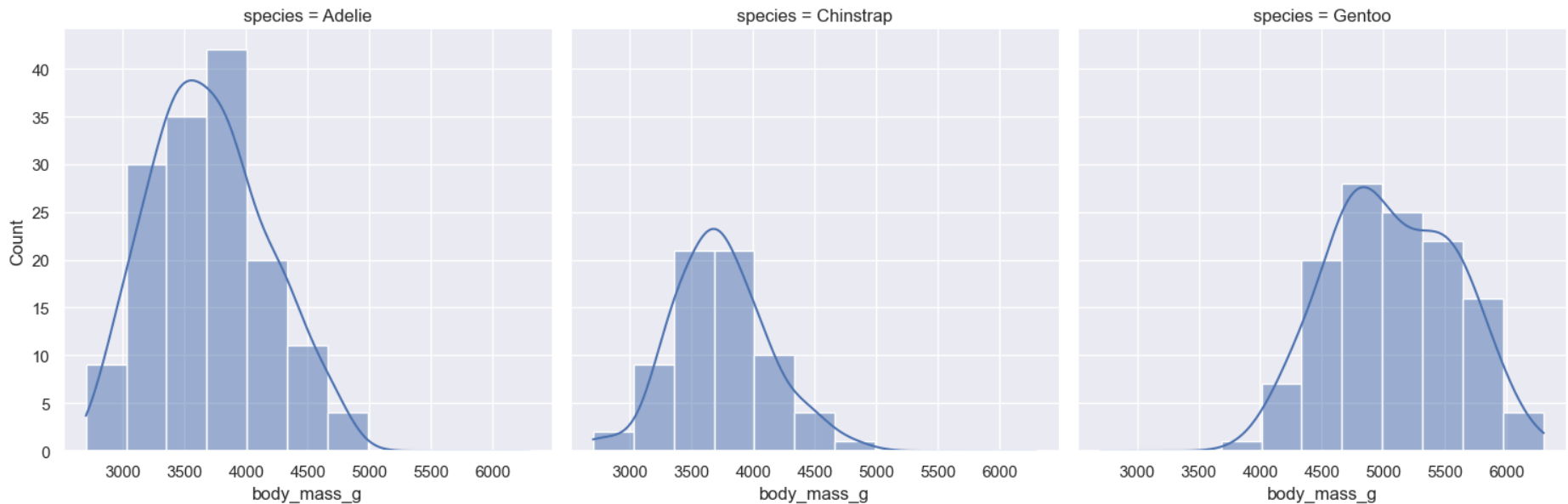
Evidence that data is unimodal



Evidence that data may be multimodal



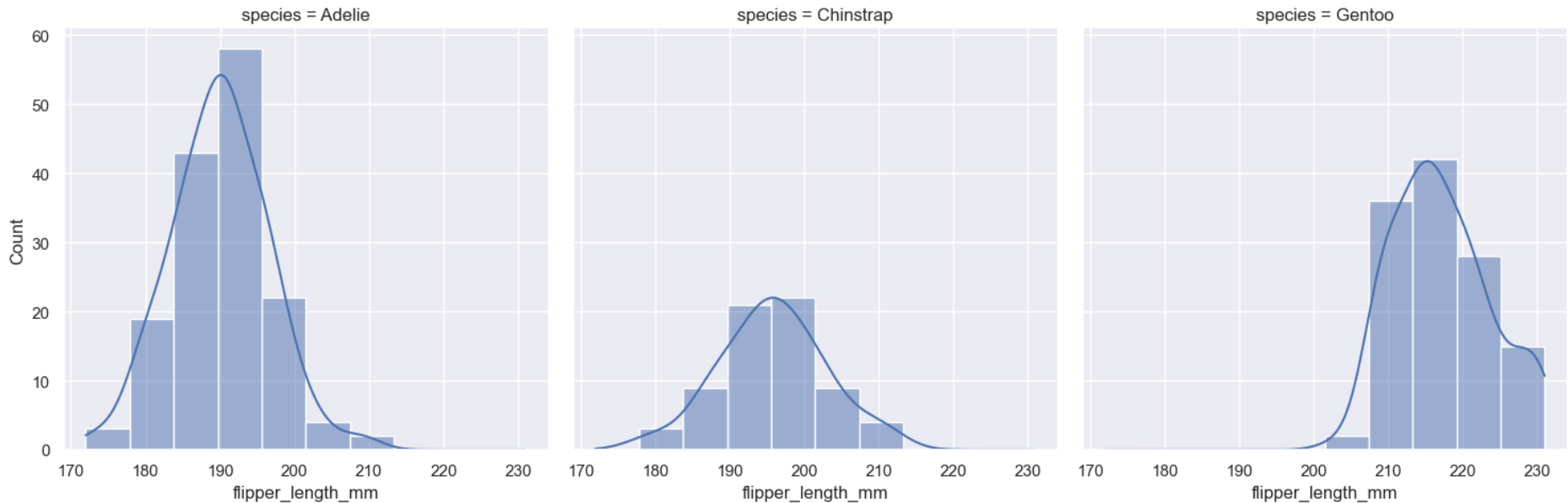
# Histogram/Density by Class (Species)



## Body Mass by Class

(breakdown by class helps to explain multimodality)

# Histogram/Density by Class (Species)



## Flipper Length by Class

(breakdown by class helps to explain multimodality)

# Python code snippets for Plots

---

```
import seaborn as sns
# Apply the default theme for seaborn visualization
sns.set_theme()

# load the penguins dataset
penguins = sns.load_dataset("penguins")

# generate different histograms
sns.displot(data=penguins, x="body_mass_g")
sns.displot(data=penguins, x="body_mass_g", bins=5)
sns.displot(data=penguins, x="body_mass_g", bins=50)

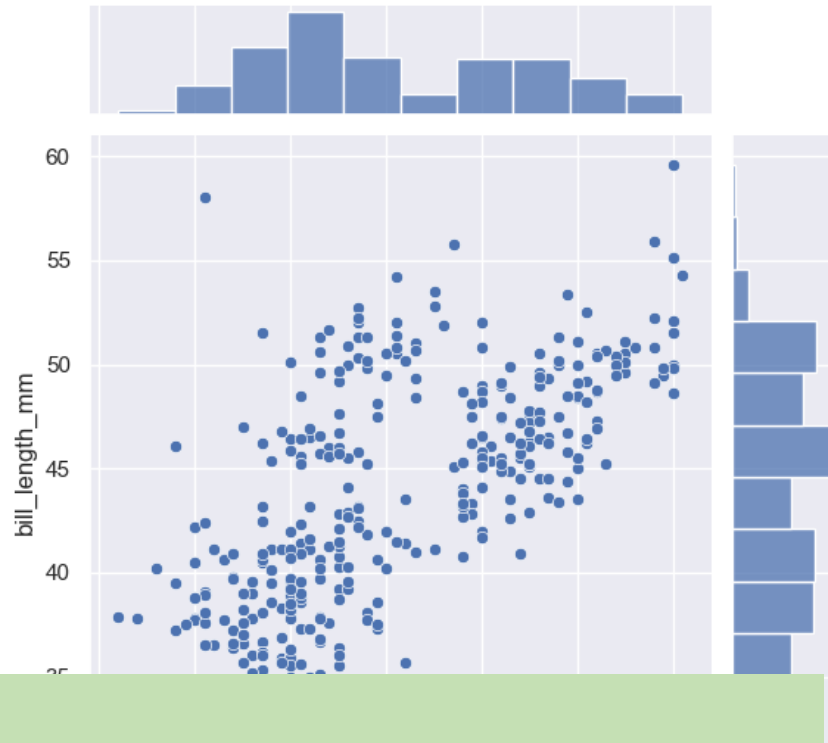
# generate histograms with density estimates overlaid, for 2 other variables
sns.displot(data=penguins, x="bill_length_mm", kde=True)
sns.displot(data=penguins, x="flipper_length_mm", kde=True)

# now look at histograms/densities broken out by class label ("species")
sns.displot(data=penguins, x="body_mass_g", col="species", kde=True)
sns.displot(data=penguins, x="flipper_length_mm", col="species", kde=True)
```

# Scatter Plots

Plots values of one feature against another

This version also shows histograms of individual variables on the side



Python code

```
import seaborn as sns
sns.set_theme() # apply the default theme
```

```
# load the penguins dataset
penguins = sns.load_dataset("penguins")
```

```
# generate a scatter plot
sns.jointplot(data=penguins, x="flipper_length_mm", y="bill_length_mm")
```

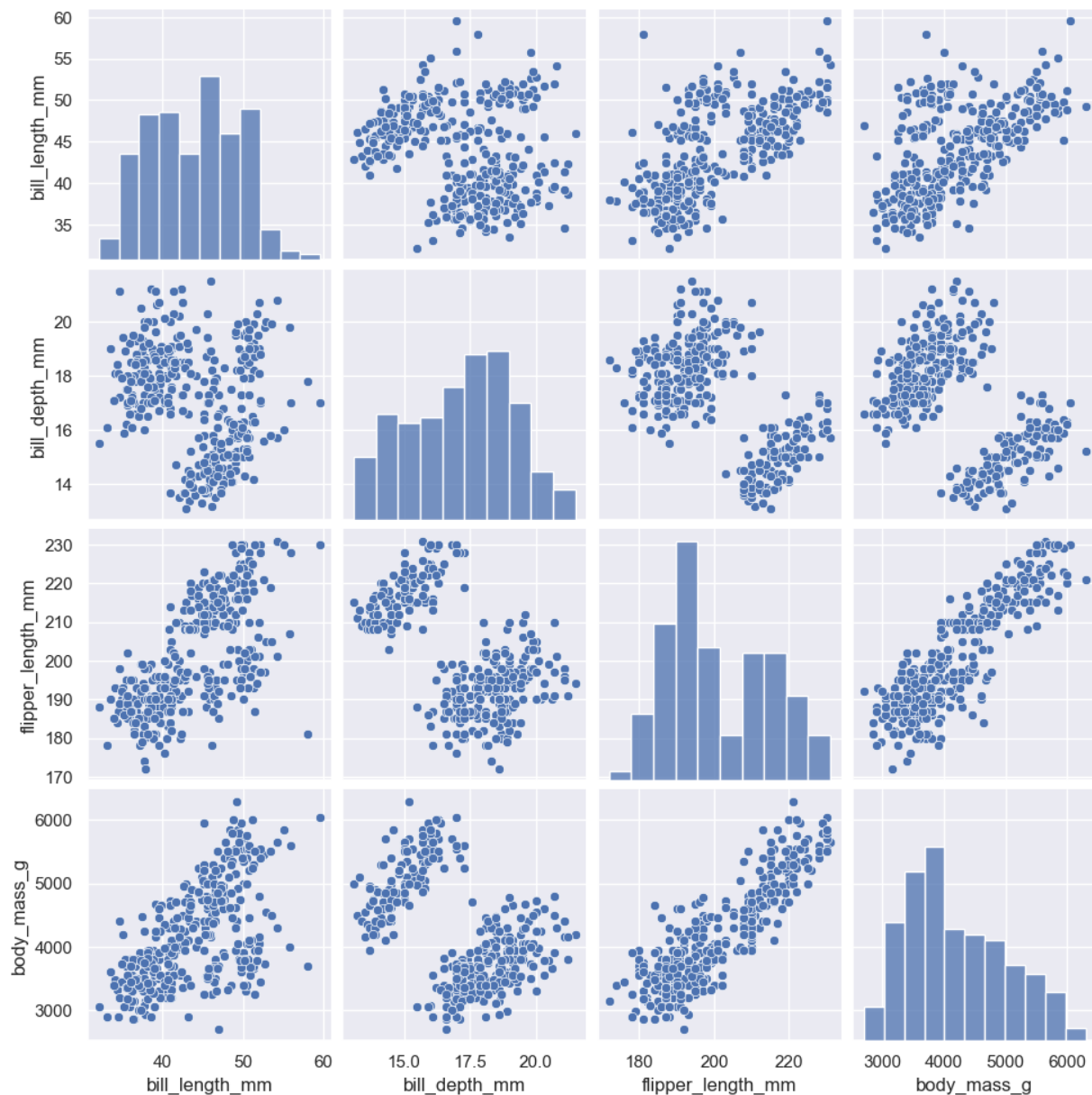
Note that seaborn is the plotting module used in these slides, while the homework is using matplotlib and pyplot

## Multiple Scatter Plots (also known as pairplots)

Here shown for the 4  
Penguin features

Not surprisingly, many  
of the features are  
positively correlated

In Python:  
`sns.pairplot(data=penguins)`



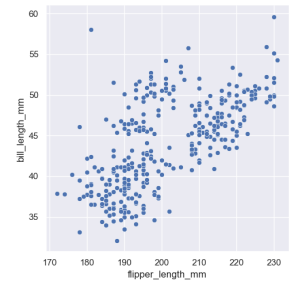
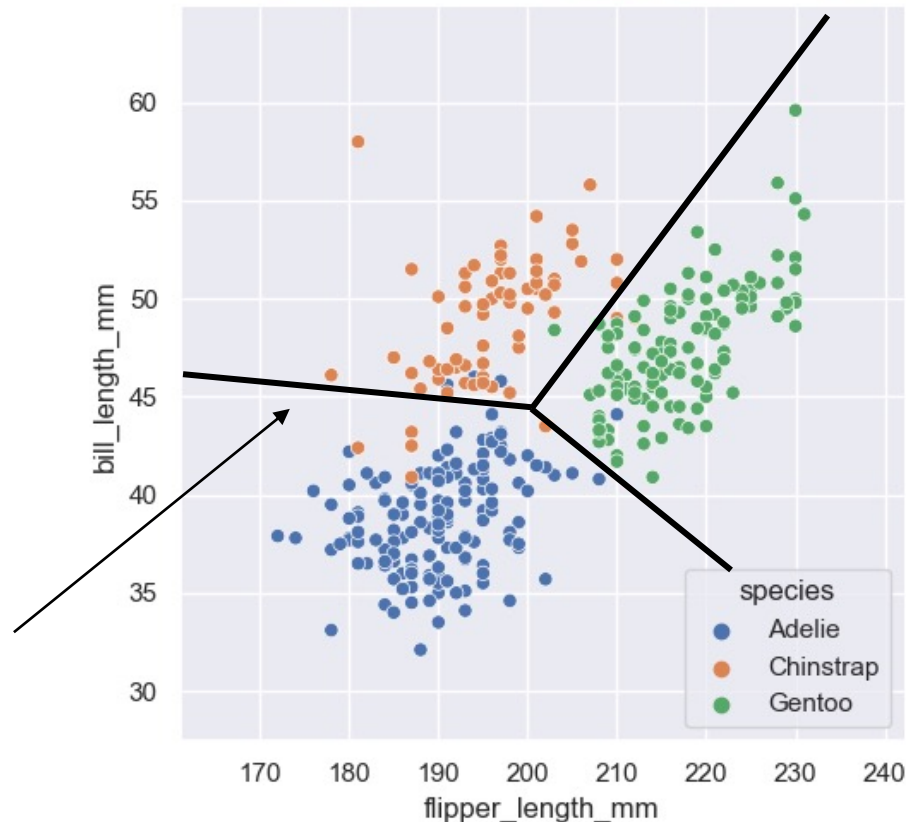


# Scatter Plots by Class

Each datapoint is now colored by its class label

We can see that the classes are well-separated in this 2d feature space

We can imagine creating 2-dim “decision boundaries” to classify the datapoints



In Python:

```
sns.jointplot(data=penguins, x="flipper_length_mm", y="bill_length_mm", hue="species")
```

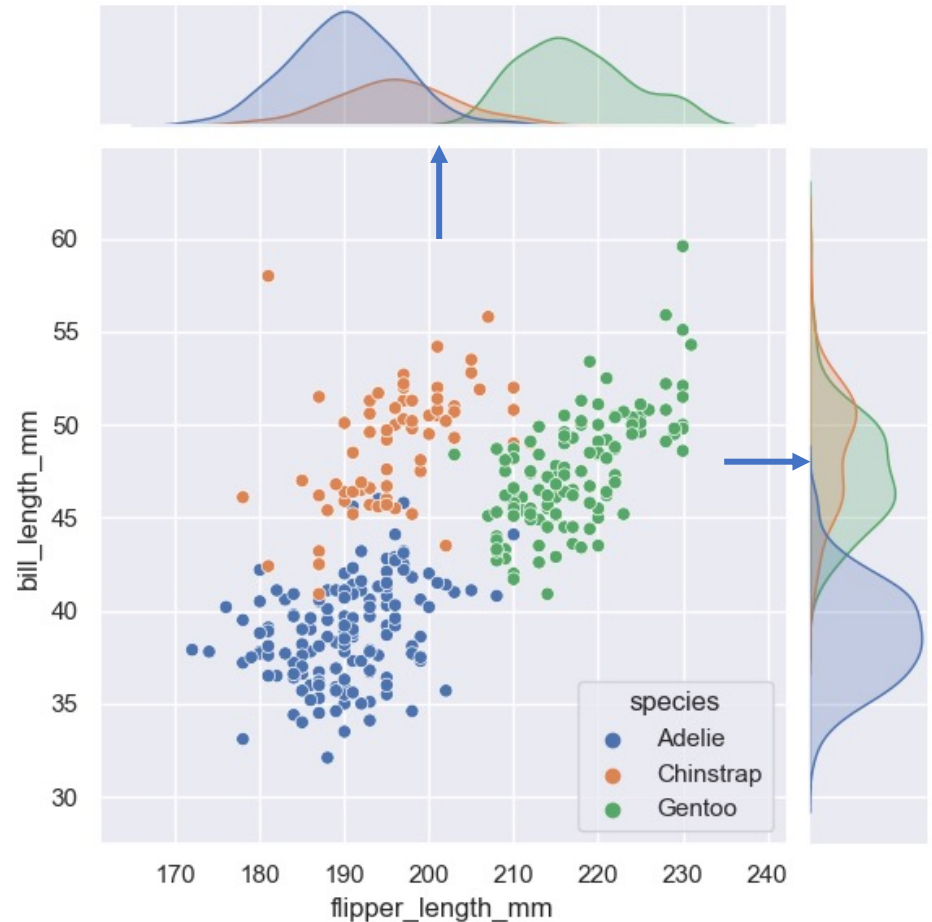
# From 2-dimensions to 1-dimension

Imagine what happens if we only had 1 feature (either flipper or bill\_length)

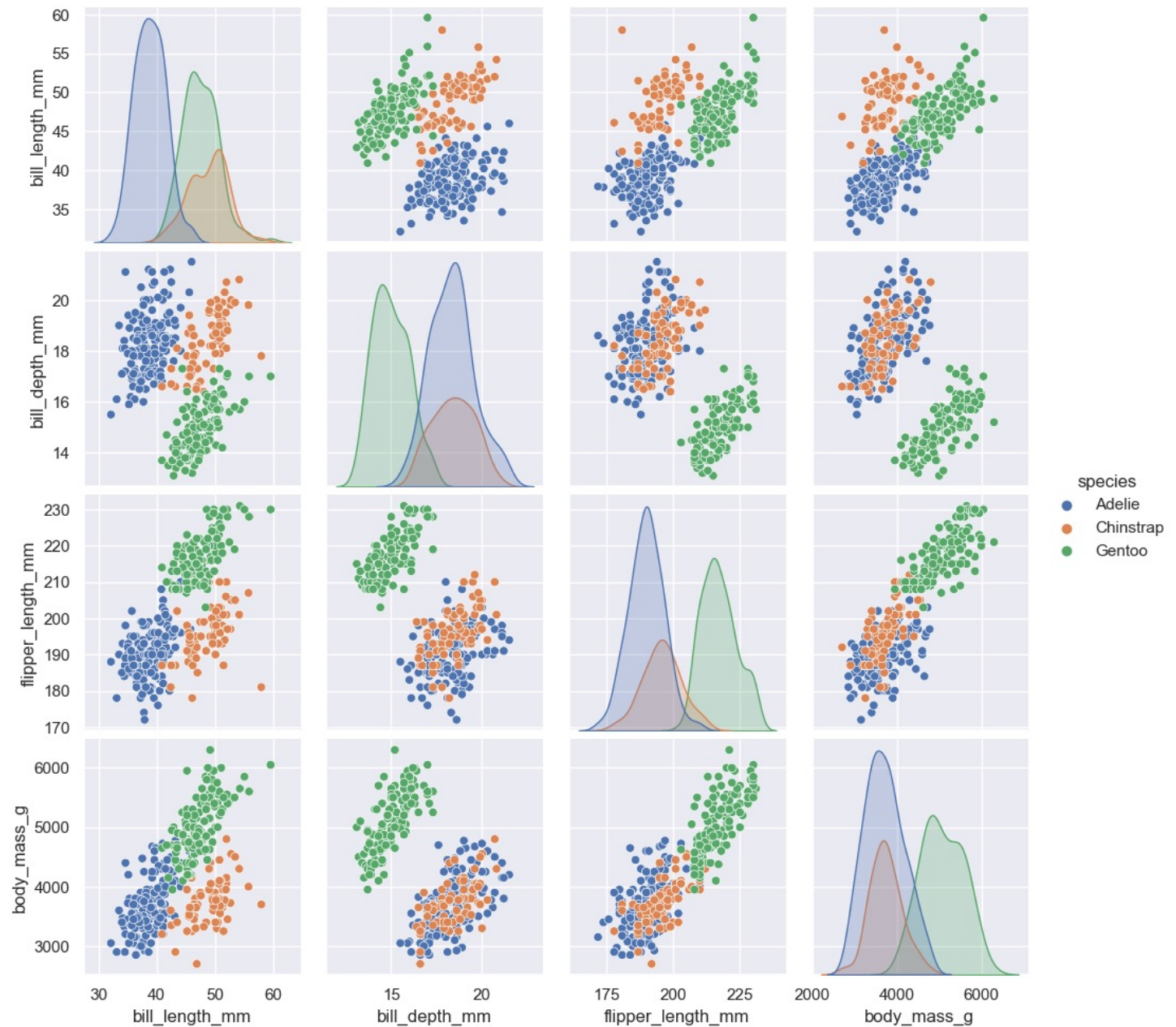
The densities on top and to the right show what the data looks like in each of 1-dimensional feature spaces

We see that in each case that two of the classes are not separated

-> 2 dimensions will be much better than either single feature for classifying this data



# Pairplot with class labels



Questions?

# Today's Lecture

---

Types of Machine Learning Problems

Representing and Exploring Data

Mathematical Notation

# Feature Vectors

---

$$\mathbf{x} = (x_1, x_2, \dots, x_d)$$

Bold font  
for a vector

A component of the  
vector, corresponding to  
the value of “feature 2”

$d$  = dimensionality  
of the vector

## Example 1:

Feature vector for a medical patient:  $\mathbf{x} = (21.4, 6.1, 200)$   
age height weight

## Example 2:

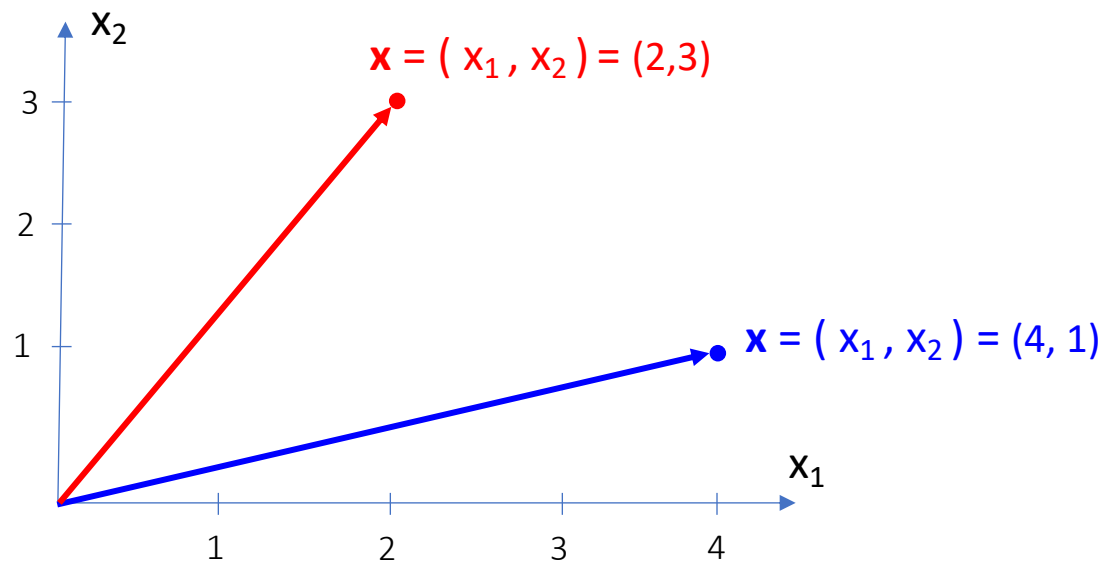
Feature vector for a loan applicant:  $\mathbf{x} = (21.4, 92697, 65k, 7.5k)$   
age zipcode income debt

# Feature Vectors as “Data Points”

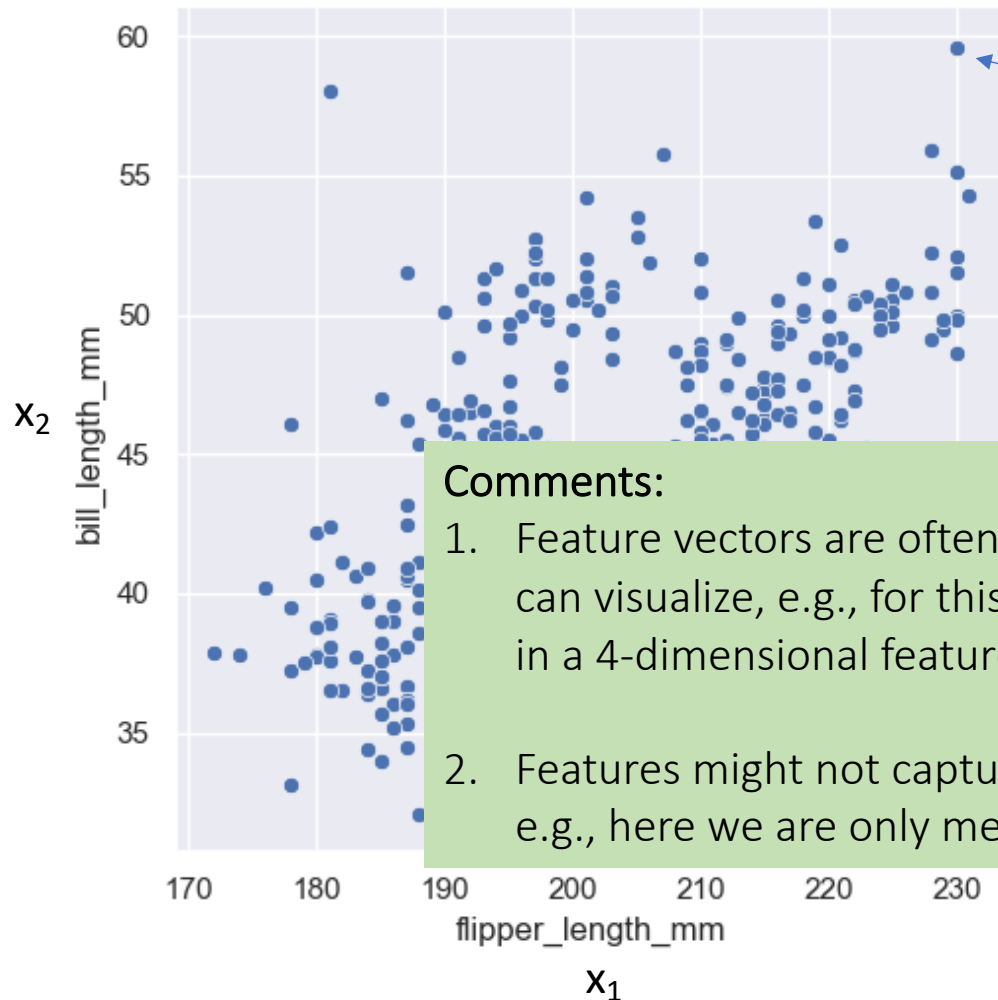
When we say “feature vector” we are referring to a point (in some d-dimensional space)

For example, if  $d = 2$ , we have  $\mathbf{x} = (x_1, x_2)$

Here are two examples of vectors (red and blue)  
representing two different datapoints in this 2-dimensional space



# 2-Dim Feature Space for Penguins



This feature vector, with coordinates (230,59.5), represents a particular datapoint (a penguin)

## Comments:

1. Feature vectors are often in higher-dimensional spaces than we can visualize, e.g., for this dataset the penguins are represented in a 4-dimensional feature space
2. Features might not capture all important aspects of a problem, e.g., here we are only measuring certain properties of penguins



# Notation for Multiple Feature Vectors

- When we speak of a single “generic” feature vector we will refer to

$$\mathbf{x} = (x_1, x_2, \dots, x_d)$$

and  $x_j$  refers to the  $j$ th component of the vector  $\mathbf{x}$

- But we often need multiple feature vectors, e.g.,  $n$  of them  
.....so we introduce a subscript  $i$  to index the feature vectors

$$\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{id}), \quad i = 1, 2, \dots, n$$

feature vector  $i$

The index  $i$  is an index over feature vectors, here going from 1 to  $n$

$x_{i2}$  is the 2<sup>nd</sup> component (value of feature 2)  
for the  $i$ th feature vector

# Data Matrix

---

Caps and bold font for a matrix

$\mathbf{X} = \begin{pmatrix} X_{11} & X_{12} & \dots & X_{1d} \\ X_{21} & X_{22} & \dots & X_{2d} \\ \dots & \dots & \dots & \dots \\ X_{n1} & X_{n2} & \dots & X_{nd} \end{pmatrix}$

n rows: each one is a feature vector for a different individual or object

d columns = d features

$X_{ij}$  is the  $j$ th feature value (column) for the  $i$ th row

$\mathbf{X}_i$  is the feature vector for the  $i$ th row

# Data Matrix

Caps and bold font for a matrix

$$\mathbf{X} = \begin{pmatrix} X_{11} & X_{12} & \dots & X_{1d} \\ X_{21} & X_{22} & \dots & X_{2d} \\ \dots & \dots & \dots & \dots \\ X_{n1} & X_{n2} & \dots & X_{nd} \end{pmatrix}$$

n rows: each one is a feature vector for a different individual or object

d columns = d features

For supervised learning problems, we also have a vector of **targets**  $\mathbf{y}$ :

Bold font for a vector

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

n rows: one target for each datapoint

# Example of a Data Matrix

---

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix} = \begin{pmatrix} 21.4 & 6.1 & 200 \\ 28.1 & 5.5 & 145 \end{pmatrix}$$

3 columns = 3 features  
↔

age          height      weight

↕  
2 rows, e.g.,  
2 patients

Examples of particular elements of the data matrix

$$x_{12} = 6.1$$

$$x_{23} = 145$$

# Example of a Data Matrix

---

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix} = \begin{pmatrix} 21.4 & 6.1 & 200 \\ 28.1 & 5.5 & 145 \end{pmatrix}$$

3 columns = 3 features  
↔

age      height      weight

↕  
2 rows, e.g.,  
2 patients

Example: If this is a binary classification problem,

$$\mathbf{y} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

↖ First patient *does not* have the disease ( $y_1 = 0$ )

↖ Second patient *does* have the disease ( $y_2 = 1$ )

# Example of a Data Matrix

---

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix} = \begin{pmatrix} 21.4 & 6.1 & 200 \\ 28.1 & 5.5 & 145 \end{pmatrix}$$

3 columns = 3 features  
↔  
age      height      weight

↕  
2 rows, e.g.,  
2 patients

Example: If this is a regression problem,

$$\mathbf{y} = \begin{pmatrix} 121 \\ 143 \end{pmatrix}$$

First patient has a systolic blood pressure of 121 ( $y_1 = 121$ )

Second patient has a systolic blood pressure of 143 ( $y_2 = 143$ )

# Data Matrix for Penguin Data

---

		bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g
$\mathbf{x}_1$	→	39.1	18.7	181.0	3750.0
$\mathbf{x}_2$	→	39.5	17.4	186.0	3800.0
$\mathbf{x}_3$	→	40.3	18.0	195.0	3250.0
$\mathbf{x}_4$	→	NaN	NaN	NaN	NaN
$\mathbf{x}_5$	→	36.7	19.3	193.0	3450.0

Here are the first 5 feature vectors from the 344 x 4 data matrix for the penguins

# Summary on Notation

---

Notation can be challenging to follow until you are familiar with it

Research papers and books on ML are full of notation....its an important aspect of understanding the details of how ML works

A few things to keep in mind in terms of notation for this class:

Bold capitals like  **$X$**  usually refer to matrices (arrays)

Bold lower-case like  **$x$**  usually refer to feature vectors (row vectors)

The subscript  $j$  will often be used to index over features (columns)

The subscript  $i$  will often be used to index over examples (rows)

For classification problems  $y$  will take values  $\{0,1\}$  or  $\{1,2, \dots, C\}$



Questions?

# Summary and Wrapup

---

- Machine learning problems come in different types
  - Input data  $\mathbf{x}$  (“features”) and outputs  $y$  (“targets” or “labels”)
  - Classification vs. regression
  - Supervised, Unsupervised, Reinforcement Learning, ...
- Data exploration and visualization is a key first step in any ML project
  - Summary statistics, histograms, scatter plots, etc.
- Notation for this course
  - Please review!
- Next lecture
  - Linear regression
  - Gradient descent