STATS 507
Data Analysis in Python

Week8: More on SciPy stats, and a crash course on learning

Dr. Xian Zhang

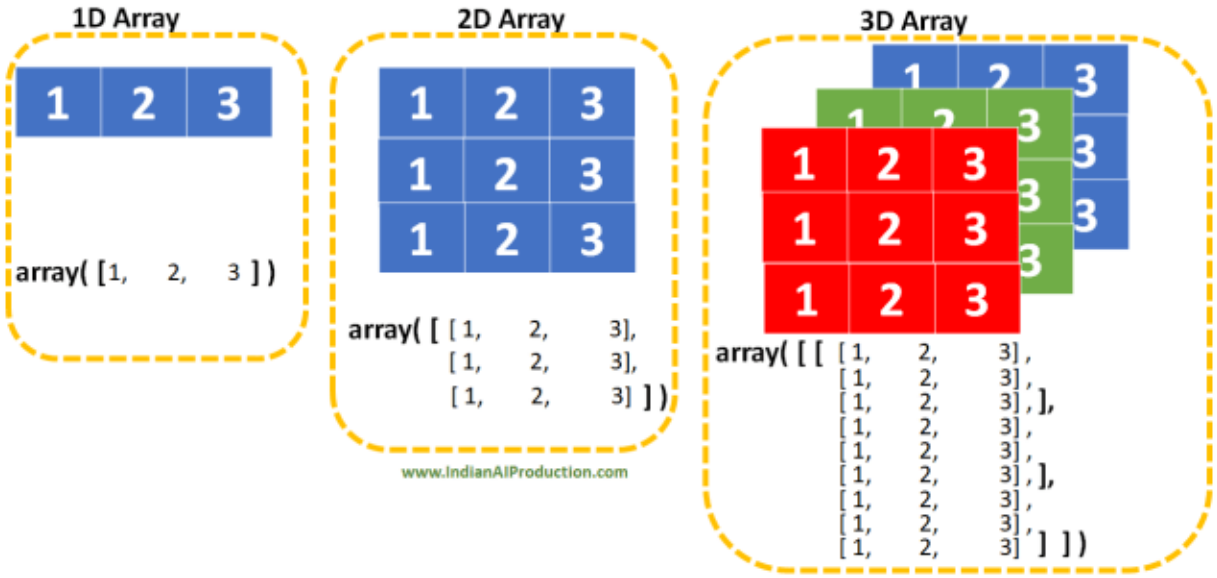
Adapted from slides by Prof. Simon Prince

NumPy EcoSystem…

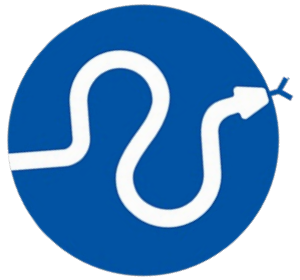
Recap: NumPy as numerical Python

One of the key data type of NumPy is its **N-dimensional array object** (also referred as: array, NumPy array, np.ndarray).

- A NumPy array is a grid of values, **all of the same type**
- *Rank* of the array: the dimension of the arrays
- *Shape*: a tuple of integers giving the size of the array along each dimension



Recap: SciPy as scientific Python



SciPy: Scientific Python

Designed and used for scientific and technical computing

SciPy is a library (collection) of **algorithms** and **mathematical** tools built to work with NumPy arrays.

- Mathematics, science, and engineering.
- Organized into **subpackages** covering different scientific domains.

https://docs.scipy.org/doc/scipy/tutorial/index.html#user-guide

Recap: `matplotlib`

Matplotlib is a comprehensive **library** for creating static, animated, and interactive **visualizations** in Python.

matplotlib.pyplot.plot

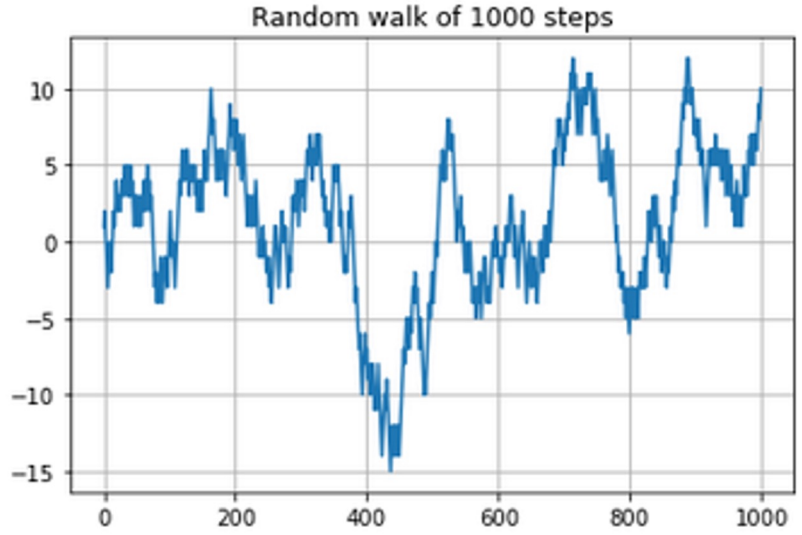
```
matplotlib.pyplot.plot(*args, **kwargs)
```

positional argument

Optional keyword arguments for **customizing** the plot

```
1 random_signs = np.sign(np.random.rand(1000)-0.5)
2 plt.grid(True)
3 plt.title('Random walk of 1000 steps')
4 # cumsum() returns cumulative sums
5 _ = plt.plot(np.cumsum(random_signs))
6 plt.savefig('random_walk.svg')
```

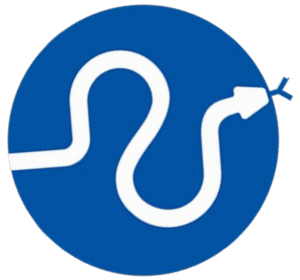
Save pics…



1. More on SciPy

2. Crash course on learning
   (before scikit learn)

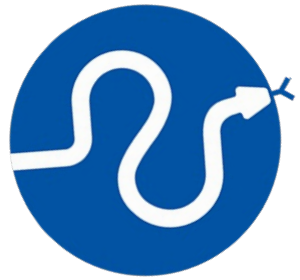
SciPy Stats



- A subpackage for stats.
- For **statistical analysis** and **probability distributions**.
- It provides a wide range of **functions and classes** for various statistical calculations, hypothesis testing, and generating random variables from different distributions.

https://docs.scipy.org/doc/scipy/reference/stats.html

SciPy Stats



Generating random variables from **probability distributions**.
Fit and analyze data…

```
from scipy.stats import norm
# Example: Probability Distributions - Normal distribution
# Generate random numbers
data = norm.rvs(loc=0, scale=1, size=1000)
# Fit distribution to data
mean, variance = norm.fit(data)
print("The mean is", mean)
print("The variance is", variance)
```

```
The mean is 0.011924257579800323
The variance is 1.0029911229469624
```

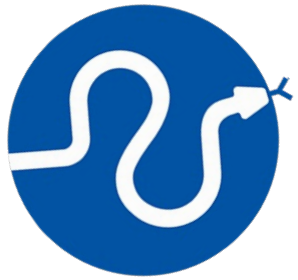
Random variates of given type.

`uniform.rvs(…)`

`expon.rvs(…)`

https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.rv_continuous.rvs.html

SciPy Descriptive Statistics



Mean, Minimum, Maximum, Variance, Skewness, Kurtosis, etc.

```python
from scipy.stats import describe, norm
import numpy as np

# Generate some random data from a normal distribution
data = norm.rvs(loc=0, scale=1, size=100)
# Calculate descriptive statistics using describe
desc = describe(data)
print(type(desc))
# Print the descriptive statistics
print("Descriptive Statistics:")
print("Mean:", desc.mean)
print("Minimum:", desc.minmax[0])
print("Maximum:", desc.minmax[1])
print("Variance:", desc.variance)
print("Skewness:", desc.skewness)
print("Kurtosis:", desc.kurtosis)
```

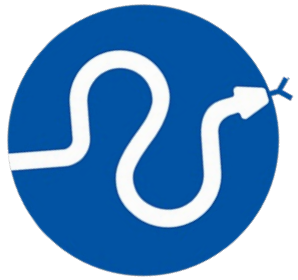
```
<class 'scipy.stats._stats_py.DescribeResult'>
Descriptive Statistics:
Mean: 0.071167075585717
Minimum: -2.532763719494357
Maximum: 2.204149247115657
Variance: 1.0921318987224702
Skewness: -0.07905470434813142
Kurtosis: -0.25947596996269695
```

Skewness: measure the asymmetry of a distribution with 0 indicating a perfectly symmetric distribution

Kurtosis: measure the ”tailedness” of a distribution

https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.describe.htm

SciPy Statistics Models



Statistics Models that can fit/analyze data

```
from scipy.stats import linregress
# Example: Linear regression
math_score = np.array([70, 92, 45, 48, 76, 65, 67, 83, 54, 29])
phys_score = np.array([87, 58, 64, 56, 88, 43, 72, 53, 72, 55])
slope, intercept, r_value, p_value, std_err = linregress(math_score, phys_score)
print("slope: ", slope)
print("intercept: ", intercept)
print("p_value: ", p_value)
```

```
slope:  0.12980489448375143
intercept:  56.635272136972034
p_value:  0.6444585968786516
```

$$\text{Physics Score} = (0.1298 \times \text{Math Score}) + 56.6353$$

With a measure of statistical significance …

https://en.wikipedia.org/wiki/Statistical_significance

Scikit learn

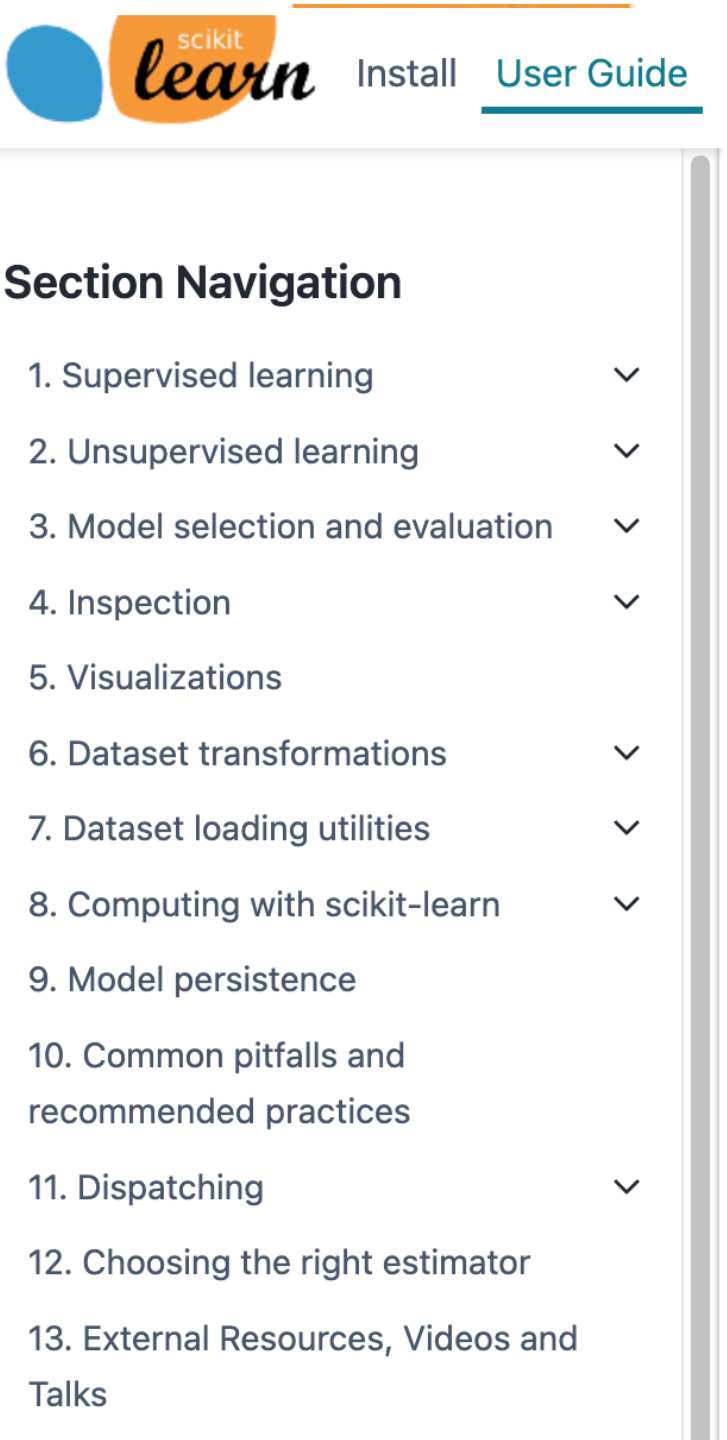
SciPy Tool**kit**
Open source machine learning library

- Built atop NumPy, SciPy and Matplotlib.

Makes many common ML/stats models easily available.

- API supports simple modeling fitting, prediction, cross-validation, etc.

https://scikit-learn.org/stable/user_guide.html#user-guide

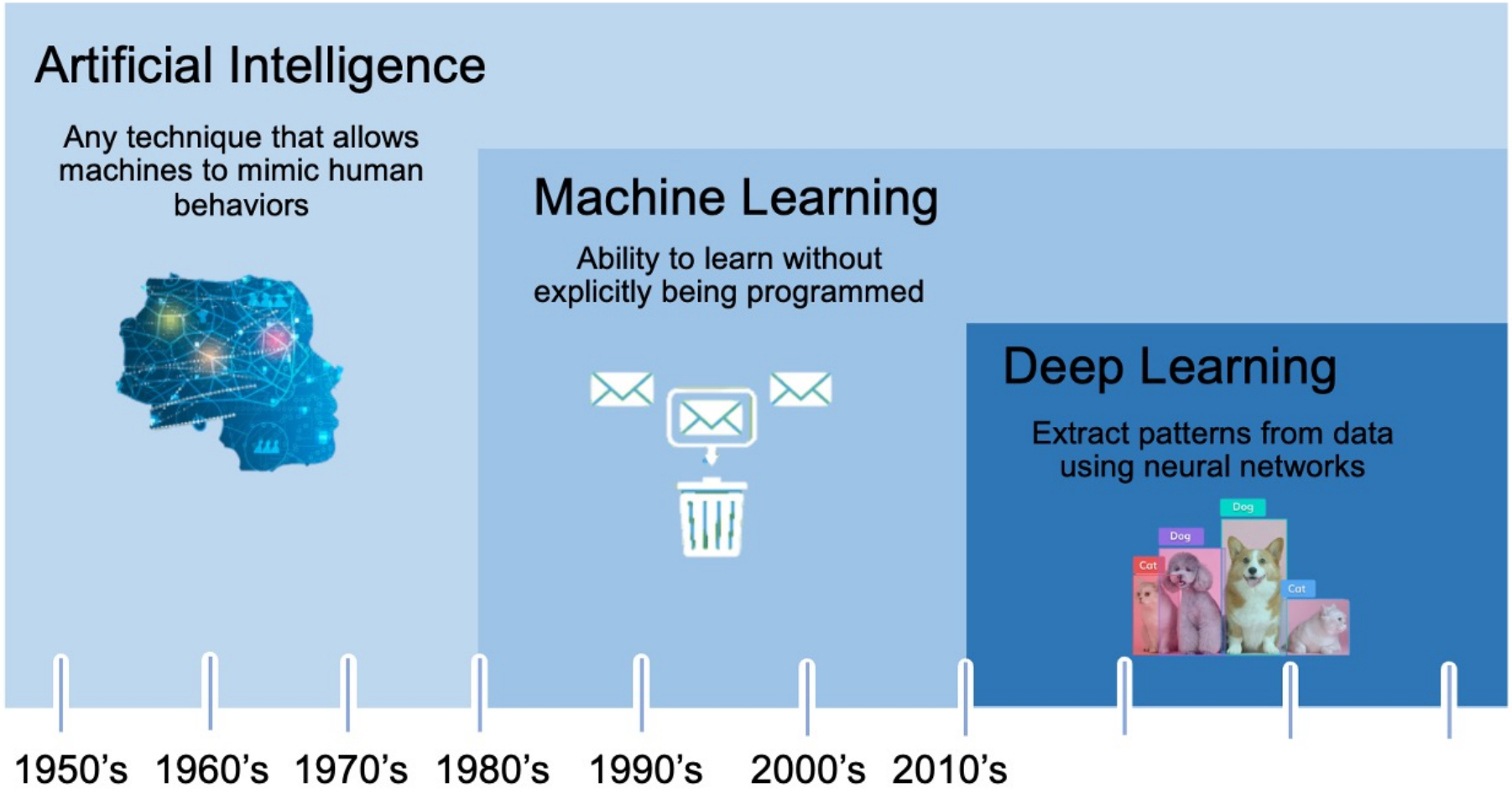


Before the tool…

A crash course on learning.

Ref: https://udlbook.github.io/udlbook/

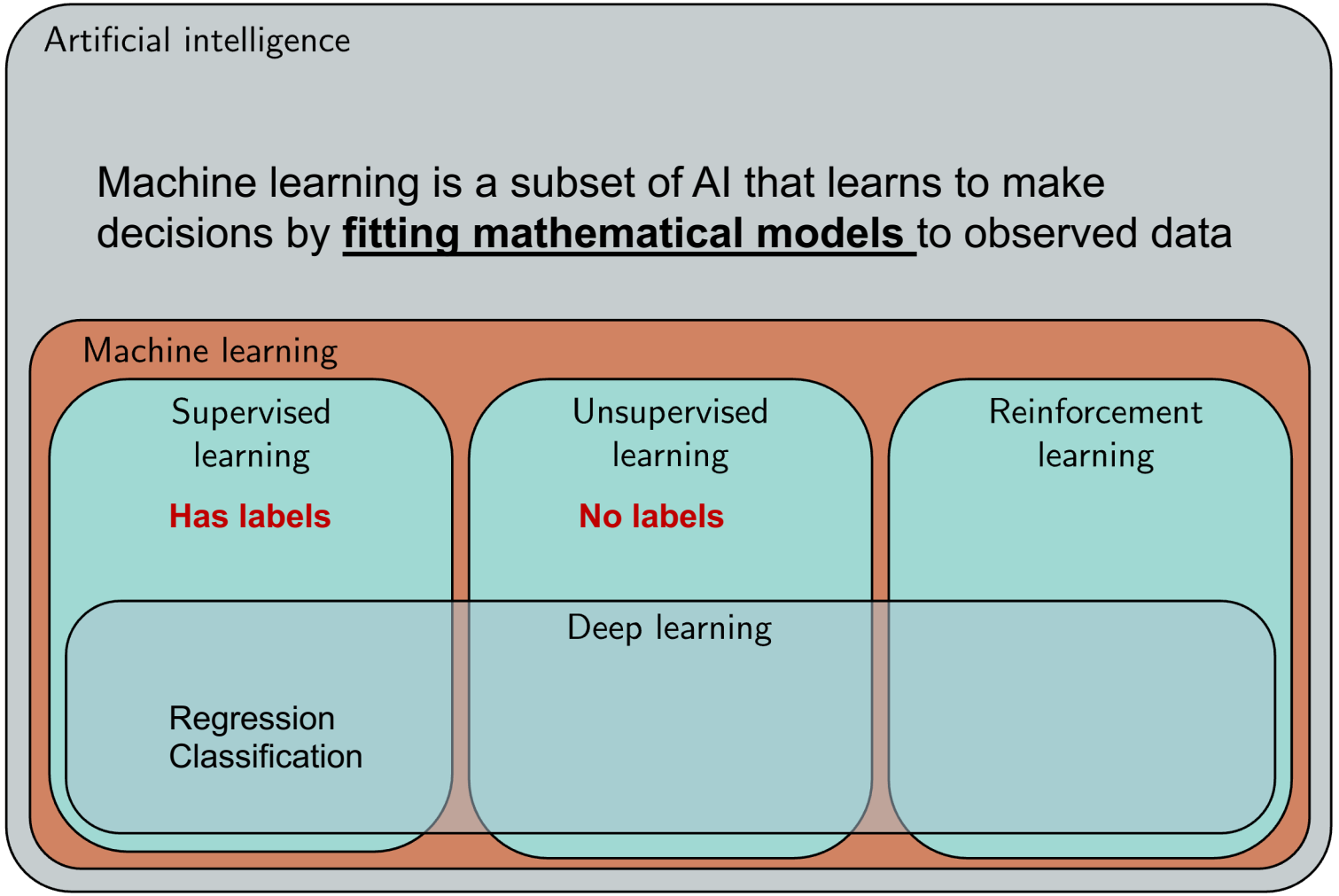
What is Learning?



Slide Credit: Alexander Amini

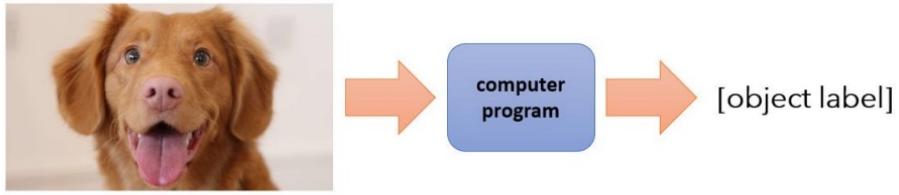
Modified from MIT open course: 6.S191 and Nvidia blog

Machine Learning



Supervised Learning

supervised learning



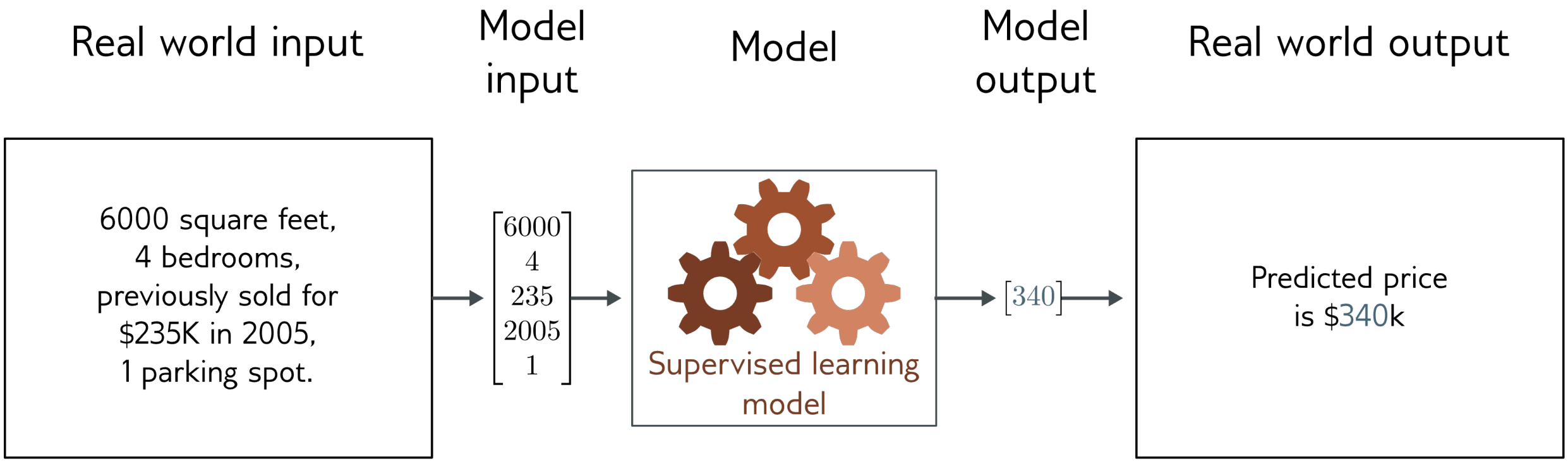
$$\text{input: } \mathbf{x}$$
$$\text{output: } \mathbf{y}$$
$$\text{data: } \mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}$$
$$\text{goal: } f_\theta(\mathbf{x}_i) \approx \mathbf{y}_i$$

someone gives this to you

- Classification

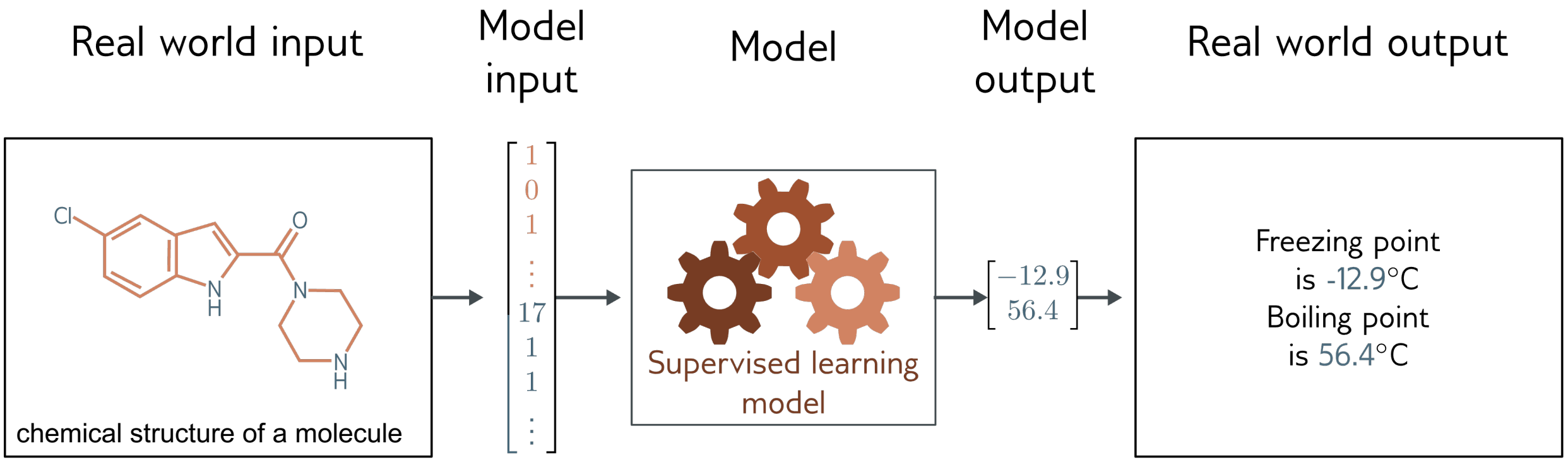
- Define a mapping from **input** to **output**
- Learn this **mapping** from paired output/output data examples.

Univariate Regression problem



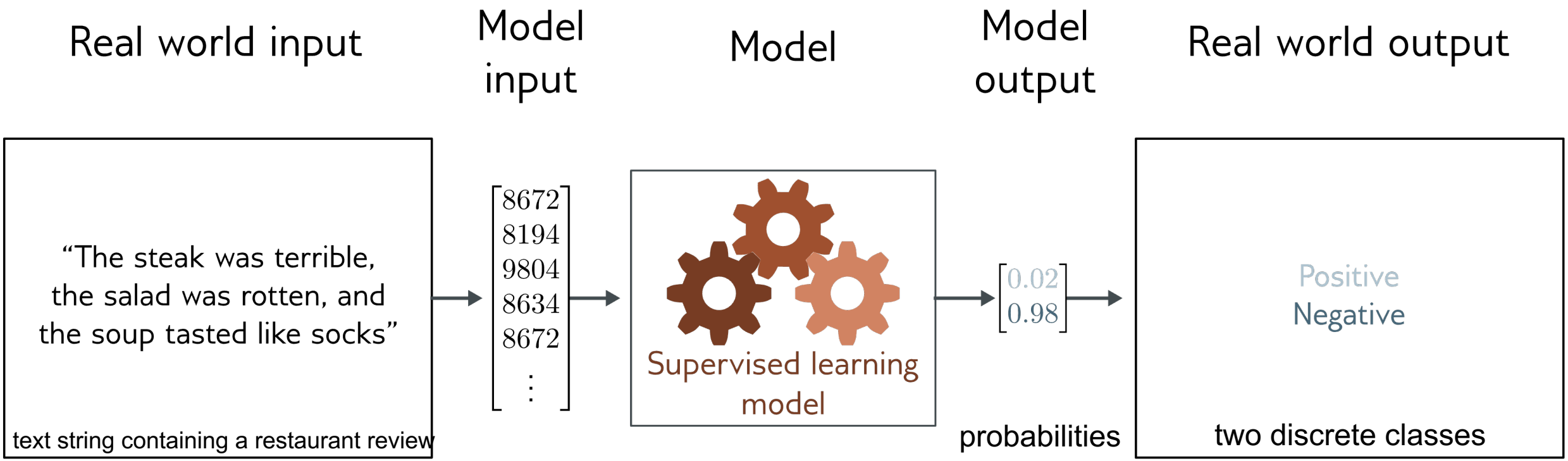
- Univariate regression problem (one output, real value)
- The model: fully connected network

Multivariate Regression problem



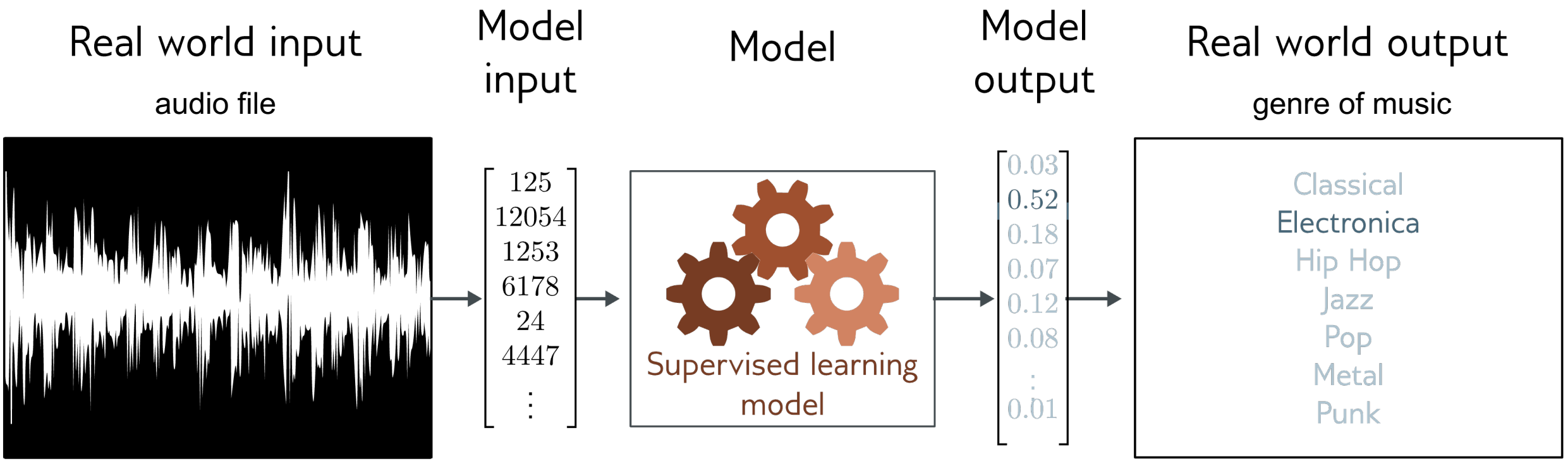
- Multivariate regression problem (>1 output, real value)
- The model: Graph neural network

Binary classification problem



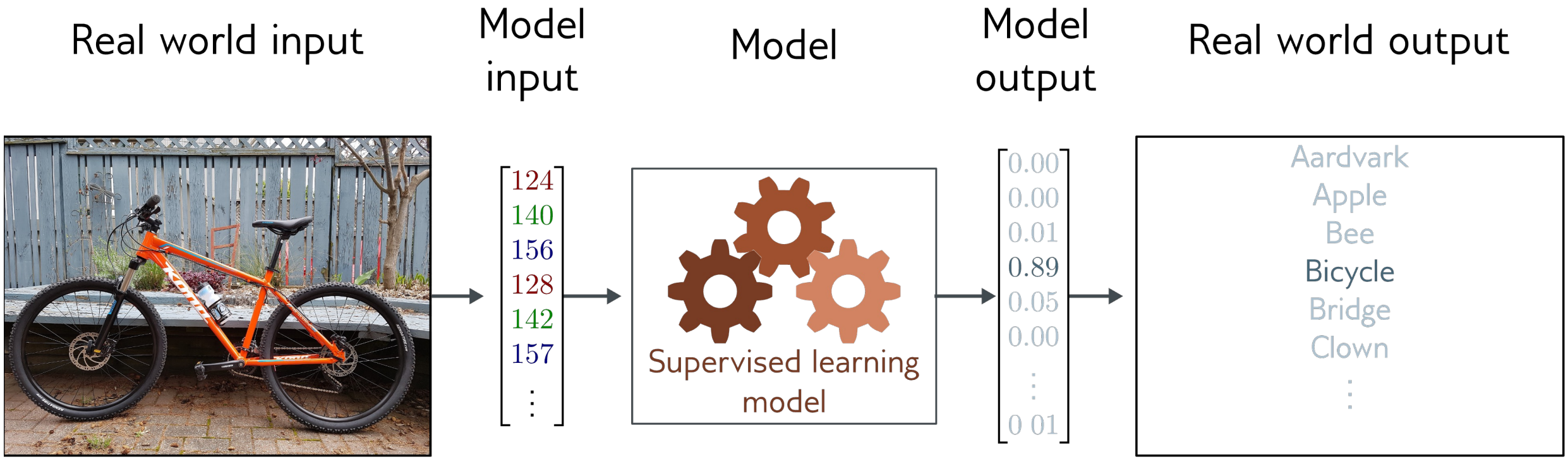
- Binary classification problem (two discrete classes)
- Transformer network

Multi-class classification problem



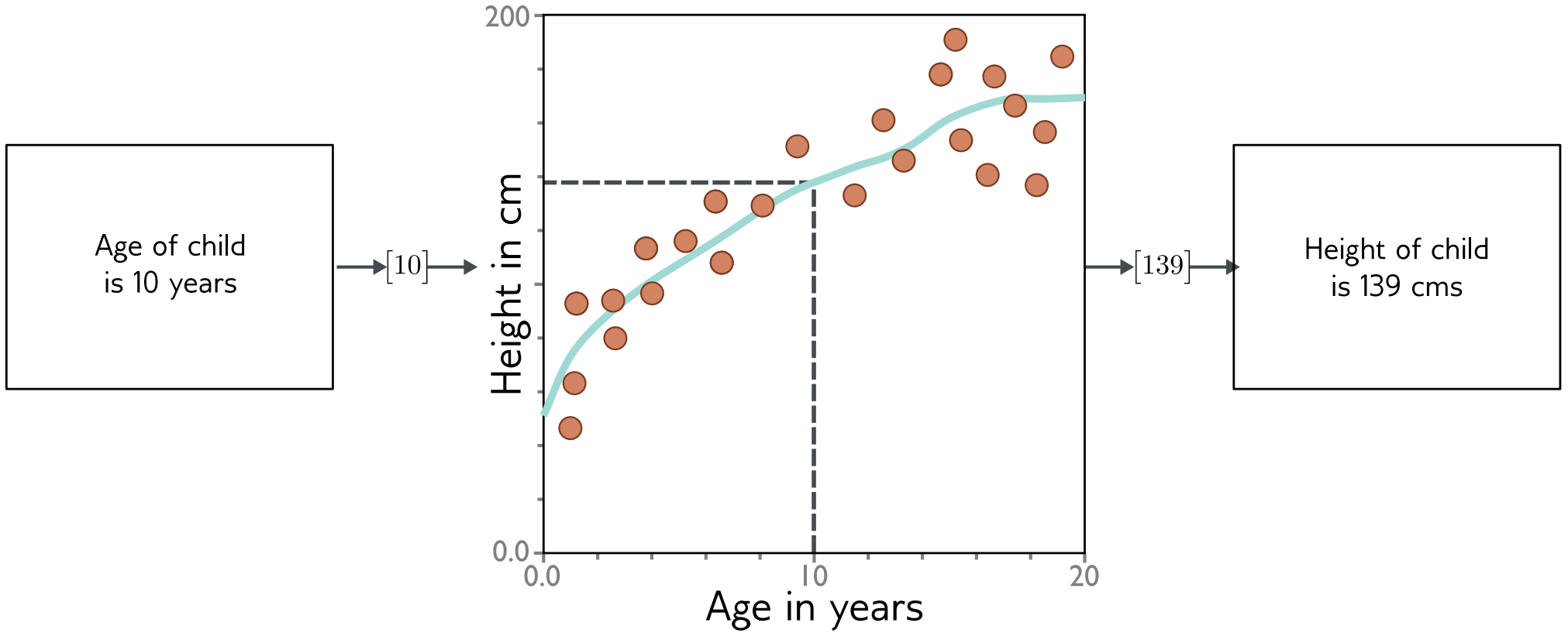
- Multiclass classification problem (discrete classes, >2 possible values)
- Recurrent neural network (RNN)

Multi-class classification problem



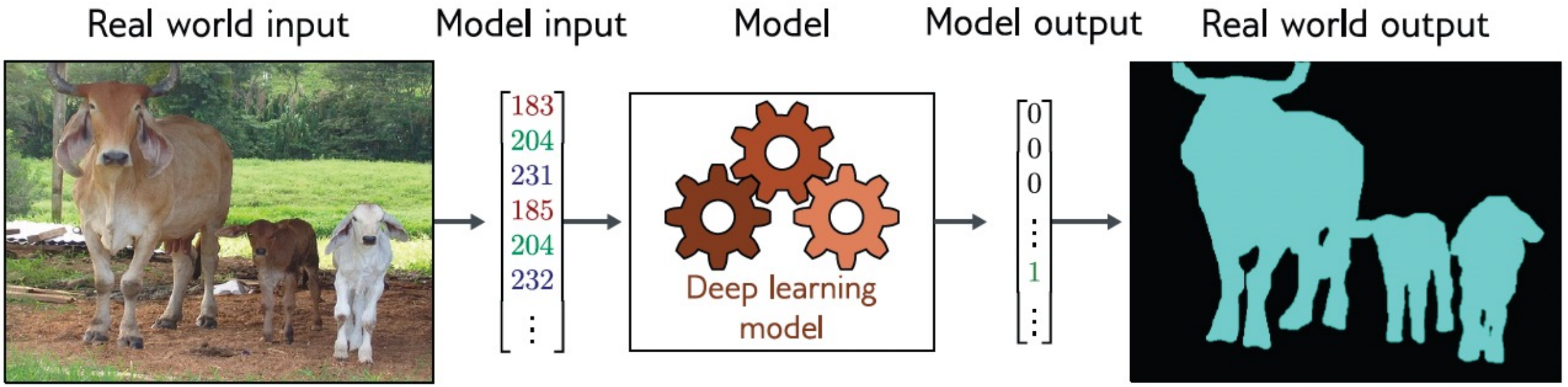
- Multiclass classification problem (discrete classes, >2 possible classes)
- Convolutional network

What exactly is in this black box?



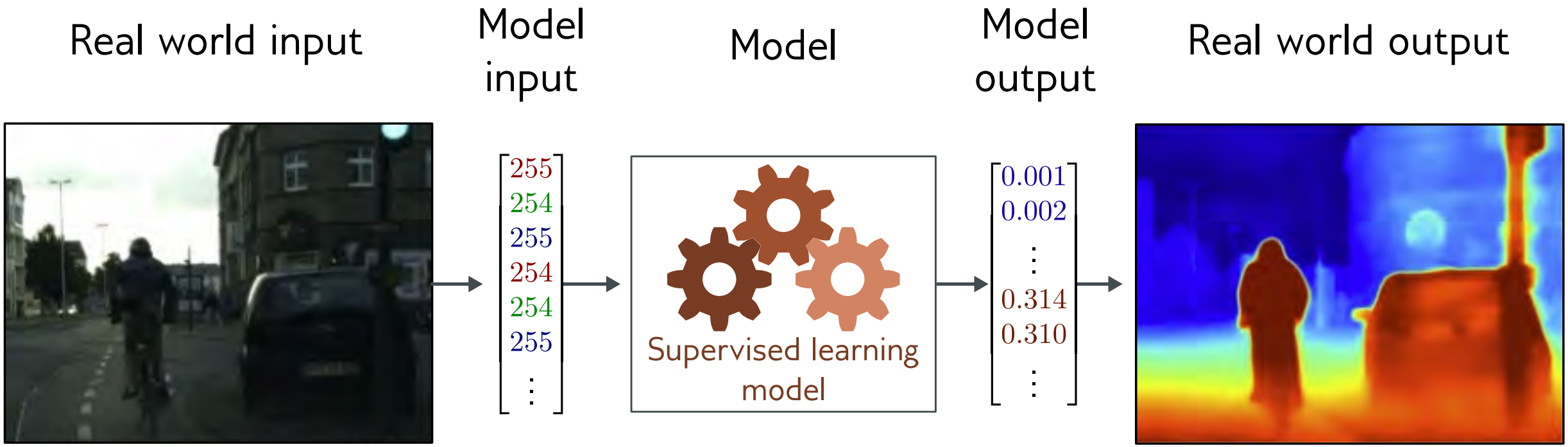
- A family of mathematical equations relating input (age) to output (height)
- Search through family of possible equations to find one that fits training data well

Image segmentation



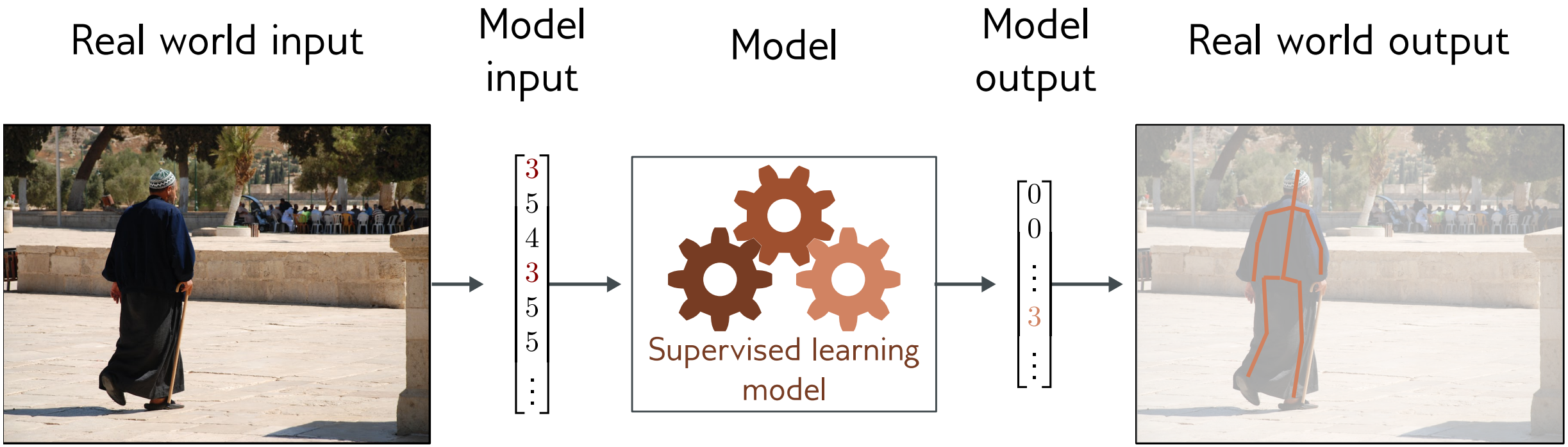
- Multivariate binary classification problem (many outputs, two discrete classes)
- Convolutional encoder-decoder network

Depth Estimation



- Multivariate regression problem (many outputs, continuous)
- Convolutional encoder-decoder network

Pose estimation



- Multivariate regression problem (many outputs, continuous)
- Convolutional encoder-decoder network

Terms for supervised problems

- Regression = continuous numbers as output
- Classification = discrete classes as output
- Two class and multiclass classification treated differently
- Univariate = one output
- Multivariate = more than one output

Some more difficult tasks…

Translation

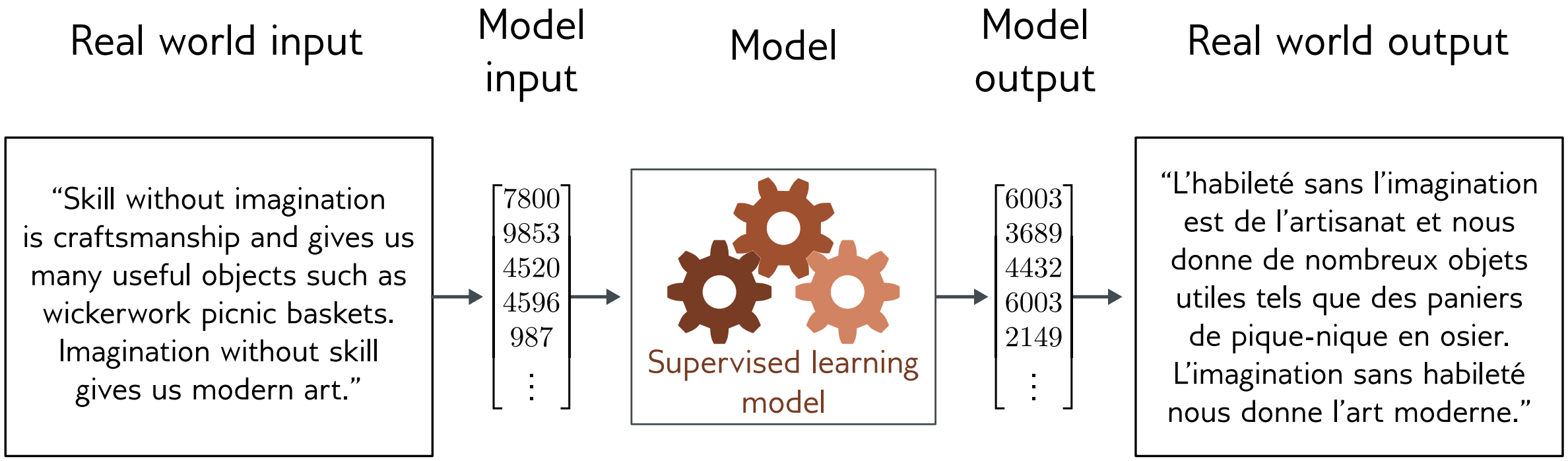


Image caption

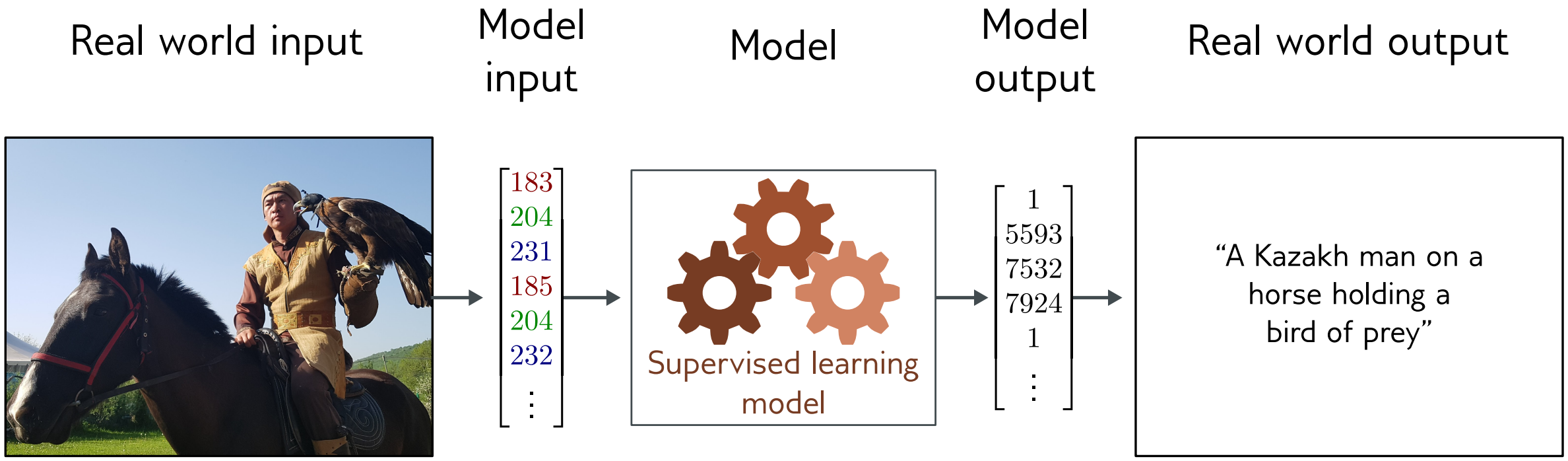
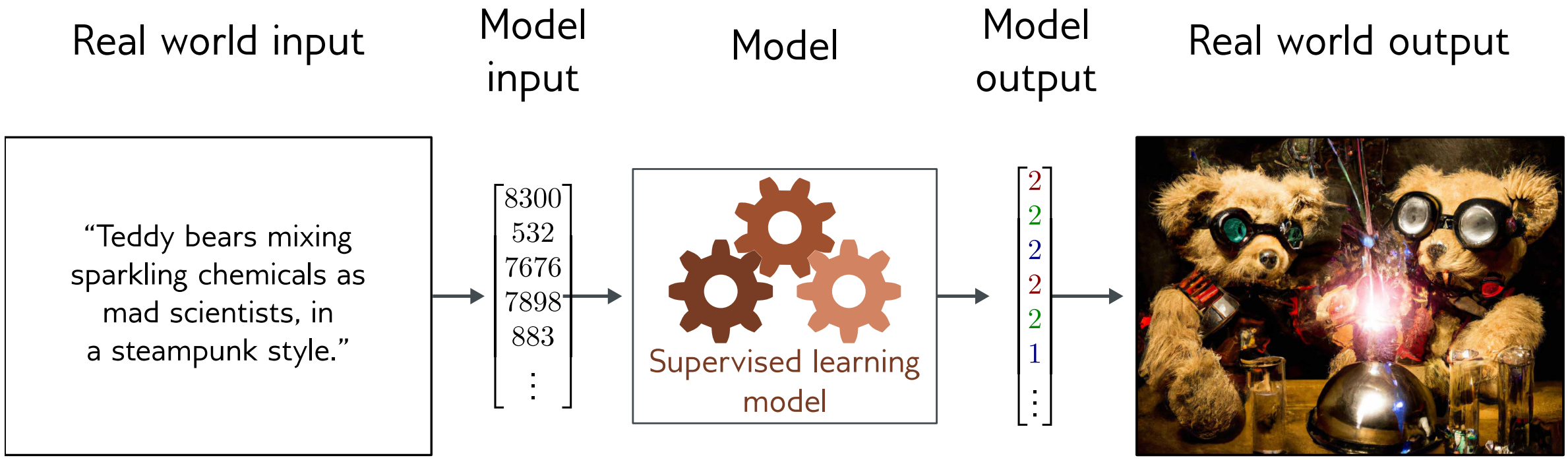


Image generation from text



What do those examples have in common?

- Very complex relationship between input and output
- Sometimes may be **many** possible valid answers
- But outputs (and sometimes inputs) obey rules

"A Kazakh man on a horse holding a bird of prey"

Language obeys grammatical rules

Natural images also have "rules"

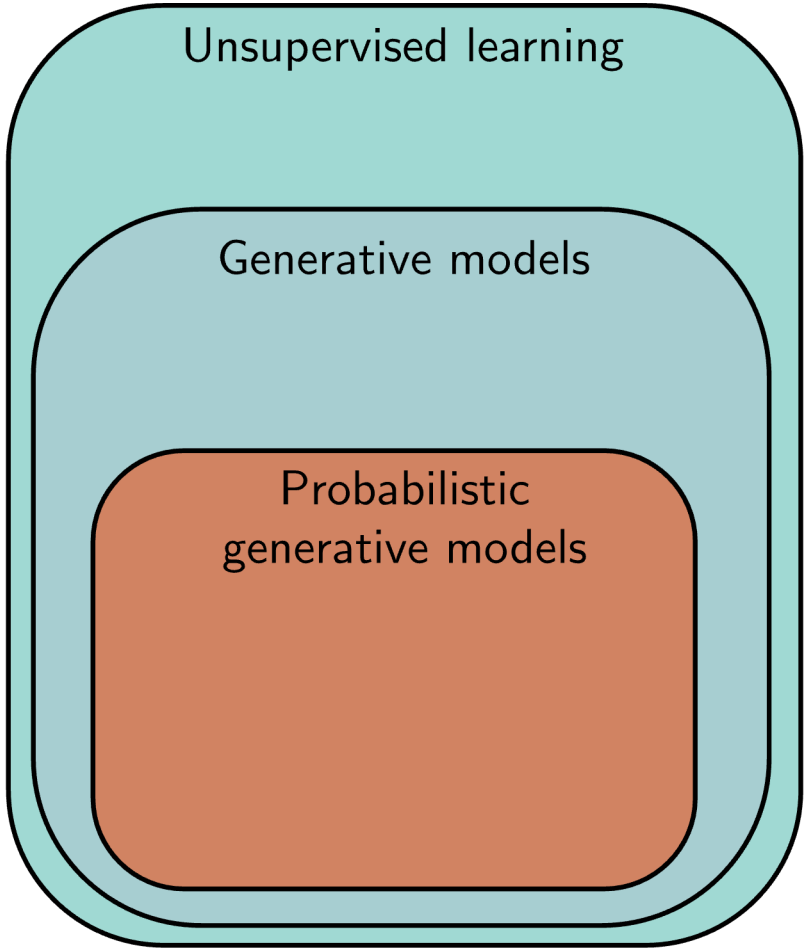
Idea

- Learn the “grammar” of the data from unlabeled examples
- Can use a gargantuan amount of data to do this (as unlabeled)
- Make the supervised learning task earlier by having a lot of knowledge of possible outputs

Unsupervised Learning

Unsupervised Learning

- Learning about a dataset **without labels**
  - e.g., clustering
- Generative models can **create examples**
  - e.g., generative adversarial networks
- PGMs learn **distribution** over data
  - e.g., variational autoencoders,
  - e.g., normalizing flows,
  - e.g., diffusion models



Generative Models

Generate images

ChatGPT 4o

write a poem about Michigan Fall

Michigan Fall

Golden leaves in whispered flight,
A tapestry of autumn light,
Lake's cool breath on every shore,
As trees ignite in crimson lore.

Maple's fire, oak's embrace,
A symphony of nature's grace,
Cider's warmth, the harvest sweet,
Crunching paths beneath our feet.

Winds that hum through orchard rows,
As apples blush in sunset glows,
The air is crisp, the skies are clear,
October sings the end of year.

Generate texts

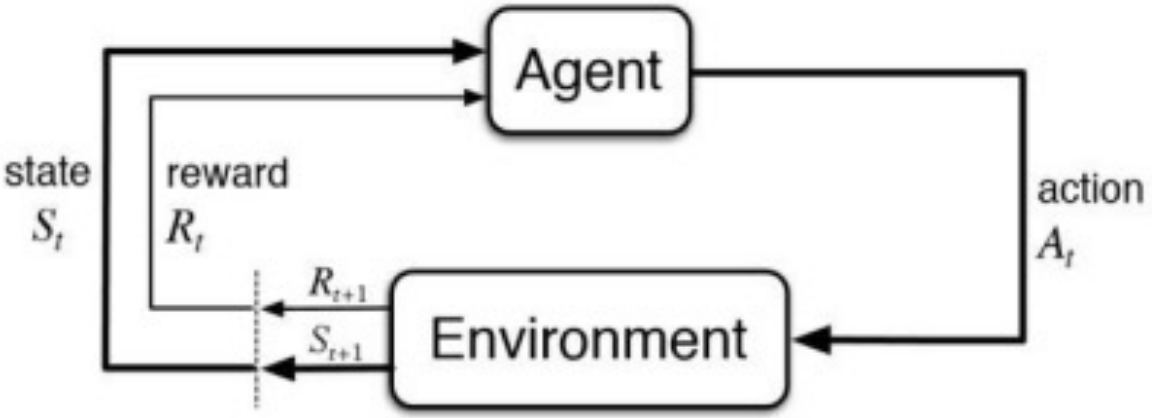
Reinforcement learning

Ref: https://udlbook.github.io/udlbook/

https://rail.eecs.berkeley.edu/deeprlcourse/

Reinforcement learning

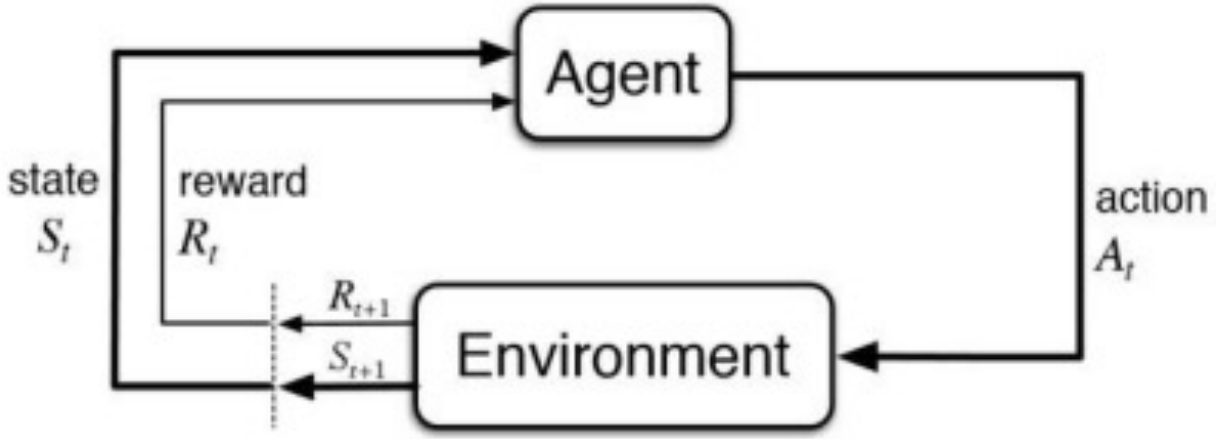
Goal: **Agent** take actions (decision making) to change the state so that you receive rewards from the environment.



https://www.nature.com/articles/nature16961

How is this different from other machine learning topics?

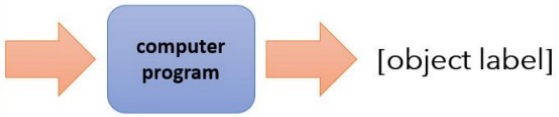
Reinforcement learning



- Data is not **i.i.d.** (independent and identically distributed): previous outputs influence future inputs.
- Ground truth answer is not known, only know we succeeded or failed. More generally, we only know the **reward**.
- Sometimes agents don't receive any data – agents have to explore the environment to **gather data** as you go.

Reinforcement learning

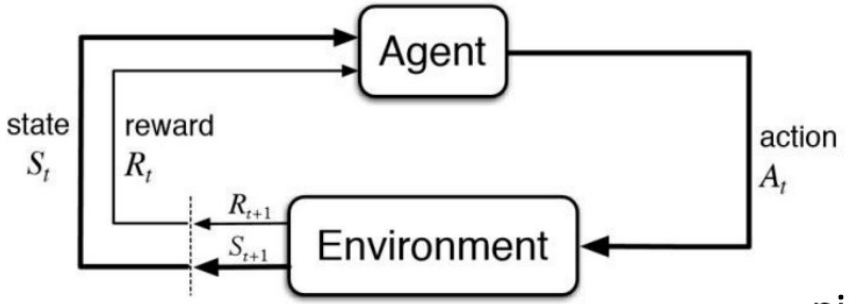
supervised learning



$$\text{input: } \mathbf{x}$$
$$\text{output: } \mathbf{y}$$
$$\text{data: } \mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}$$
$$\text{goal: } f_\theta(\mathbf{x}_i) \approx \mathbf{y}_i$$

someone gives this to you

reinforcement learning



$$\text{input: } \mathbf{s}_t \textit{ at each time step}$$
$$\text{output: } \mathbf{a}_t \textit{ at each time step}$$
$$\text{data: } (\mathbf{s}_1, \mathbf{a}_1, r_1, \ldots, \mathbf{s}_T, \mathbf{a}_T, r_T)$$
$$\text{goal: learn } \pi_\theta : \mathbf{s}_t \rightarrow \mathbf{a}_t$$
$$\text{to maximize } \textstyle\sum_t r_t$$

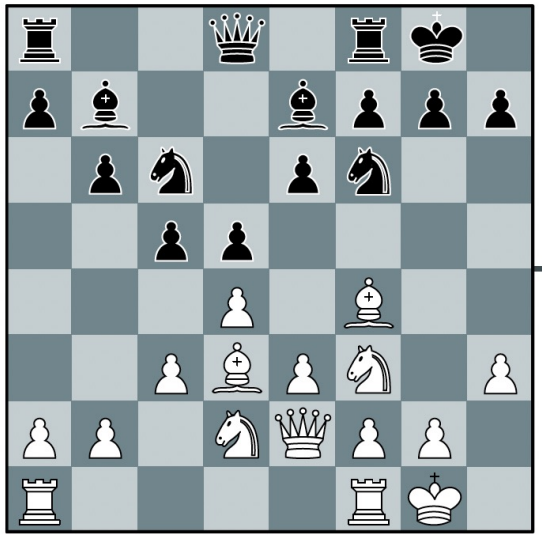
pick your own actions

Ref: https://rail.eecs.berkeley.edu/deeprlcourse/

Example: chess

- States are valid states of the chess board
- Actions at a given time are valid possible moves
- Positive rewards for taking pieces, negative rewards for losing them

State



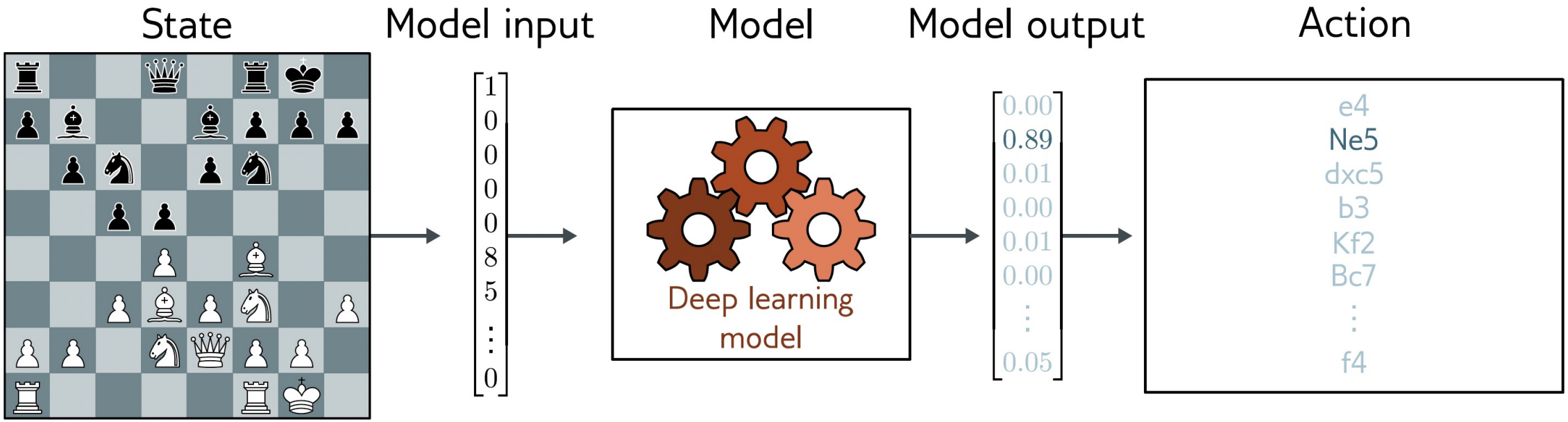
:

Action

e4
Ne5
dxc5
b3
Kf2
Bc7
⋮
f4

Example: chess

- States are valid states of the chess board
- Actions at a given time are valid possible moves
- Positive rewards for taking pieces, negative rewards for losing them



2018 Turing award winners

The Nobel Prize in Physics 2024

Ill. Niklas Elmehed © Nobel Prize Outreach

John J. Hopfield

Prize share: 1/2

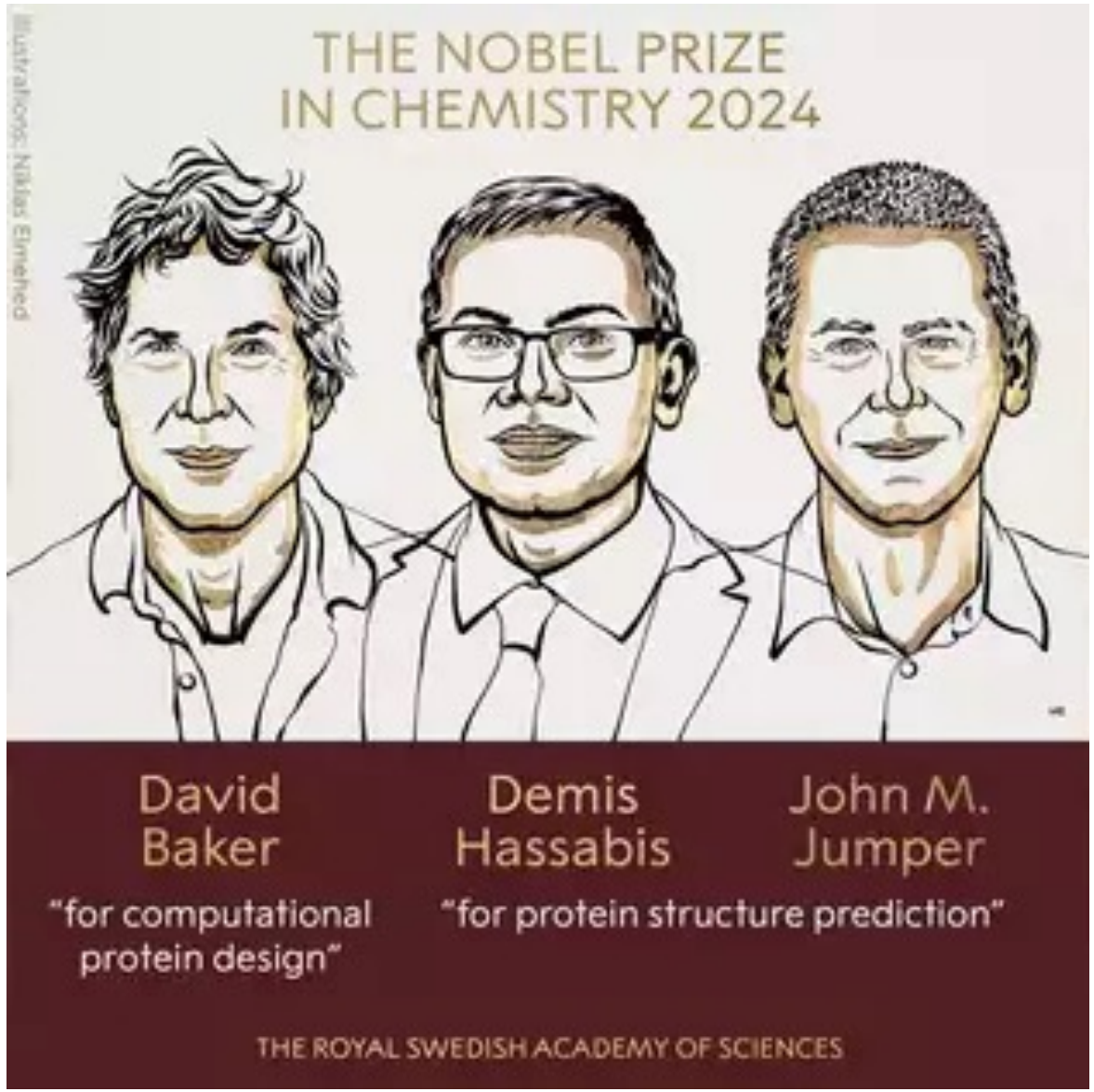
Ill. Niklas Elmehed © Nobel Prize Outreach

Geoffrey E. Hinton

Prize share: 1/2

The Nobel Prize in Physics 2024 was awarded jointly to John J. Hopfield and Geoffrey E. Hinton "for foundational discoveries and inventions that enable machine learning with artificial neural networks"

https://www.nobelprize.org/prizes/physics/2024/summary/



"Demis Hassabis and John Jumper have developed an AI model to solve a 50-year-old problem: predicting proteins' complex structures."

https://www.nobelprize.org/prizes/chemistry/2024/summary/

Coming next: scikit-learn for learning problems

The **MNIST database** (*Modified National Institute of Standards and Technology database*) is a large database of handwritten digits

Other things

HW6 is due next week.