



NOTRE DAME  
OF MARYLAND  
UNIVERSITY



JOHNS HOPKINS  
WHITING SCHOOL  
of ENGINEERING

# Artificial Intelligence in Predictive Health and Laboratory Research

Somdatta Goswami  
Civil and Systems Engineering  
Johns Hopkins University

Rosy Joshi-Mukherjee and Sherry Moore  
School of Pharmacy  
Notre Dame of Maryland University

# Exciting Recent Developments!

Cute owl professor writing drug names on a computer, photorealistic style.

**DALL.E 2**



Write a Drug Interaction Checker code to check if two drugs interact with each other using a small predefined list of known interactions.

python

**ChatGPT**

```
# Predefined list of drug interactions (very simplified)
interactions = {
    ("aspirin", "warfarin"): "Increased risk of bleeding.",
    ("ibuprofen", "lisinopril"): "May reduce kidney function.",
    ("metformin", "alcohol"): "Risk of lactic acidosis."
}

# Get input from user
drug1 = input("Enter the first drug: ").lower()
drug2 = input("Enter the second drug: ").lower()

# Check both orders (aspirin + warfarin or warfarin + aspirin)
if (drug1, drug2) in interactions:
    print("⚠️ Interaction Warning:", interactions[(drug1, drug2)])
elif (drug2, drug1) in interactions:
    print("⚠️ Interaction Warning:", interactions[(drug2, drug1)])
else:
    print("✅ No known interactions between these drugs.")
```

# Exciting Recent Developments!

Cute owl professor writing drug names on a computer, photorealistic style.

**DALL.E 2**



## What do these all have in common?

**ChatGPT coding:** Write a Drug Interaction Checker code to check if two drugs interact with each other using a small predefined list of known interactions.

## Machine Learning... 😊

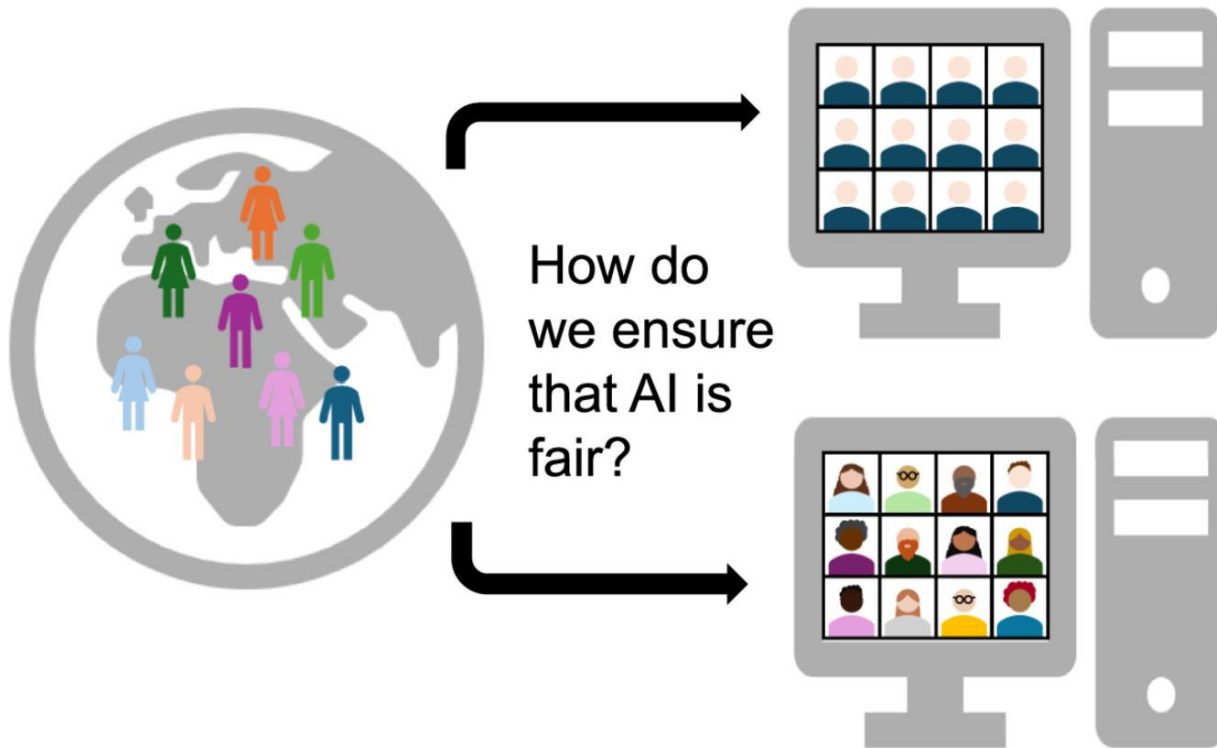
python

```
# Predefined list of drug interactions (very simplified)
interactions = {
    ("aspirin", "warfarin"): "Increased risk of bleeding.",
    ("ibuprofen", "lisinopril"): "May reduce kidney function.",
    ("metformin", "glimepiride"): "Risk of lactic acidosis."
}

# Get input from user
drug1 = input("Enter the first drug: ").lower()
drug2 = input("Enter the second drug: ").lower()

# Check both orders (aspirin + warfarin or warfarin + aspirin)
if (drug1, drug2) in interactions:
    print("⚠️ Interaction Warning:", interactions[(drug1, drug2)])
elif (drug2, drug1) in interactions:
    print("⚠️ Interaction Warning:", interactions[(drug2, drug1)])
else:
    print("✅ No known interactions between these drugs.")
```

# Worrisome Recent Development



Fairness in AI for healthcare?

Artificial intelligence / Machine learning

## Training a single AI model can emit as much carbon as five cars in their lifetimes

Deep learning has a terrible carbon footprint.

by **Karen Hao**

June 6, 2019

### Need for Efficient AI Models?

# Worrisome Recent Development

What do these all have in common?  
**Machine Learning...** 🙄

# Our Goals in this Course

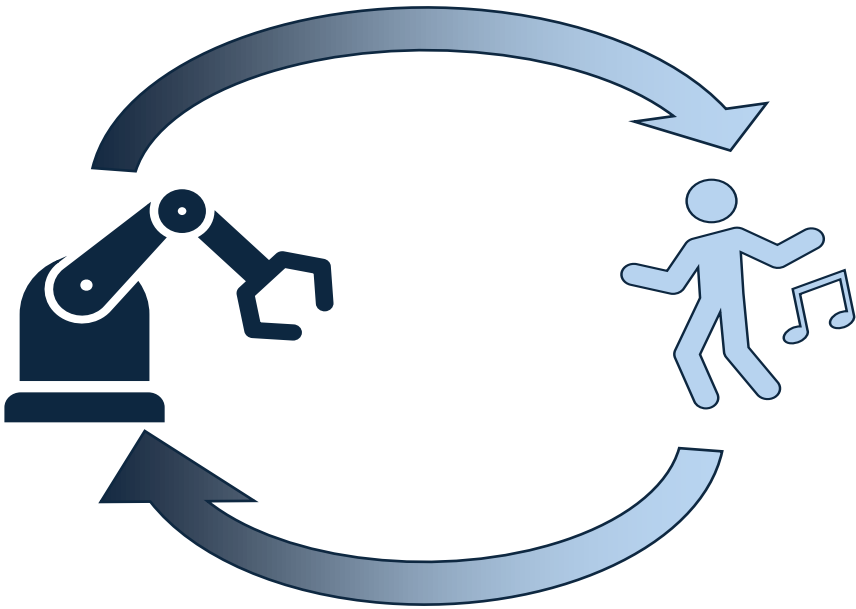
---

- Understand the core high-level ideas that lead to the success of machine learning-based methods.
- Gain hands-on experience implementing deep learning models for solving problems in pharmacy/healthcare.

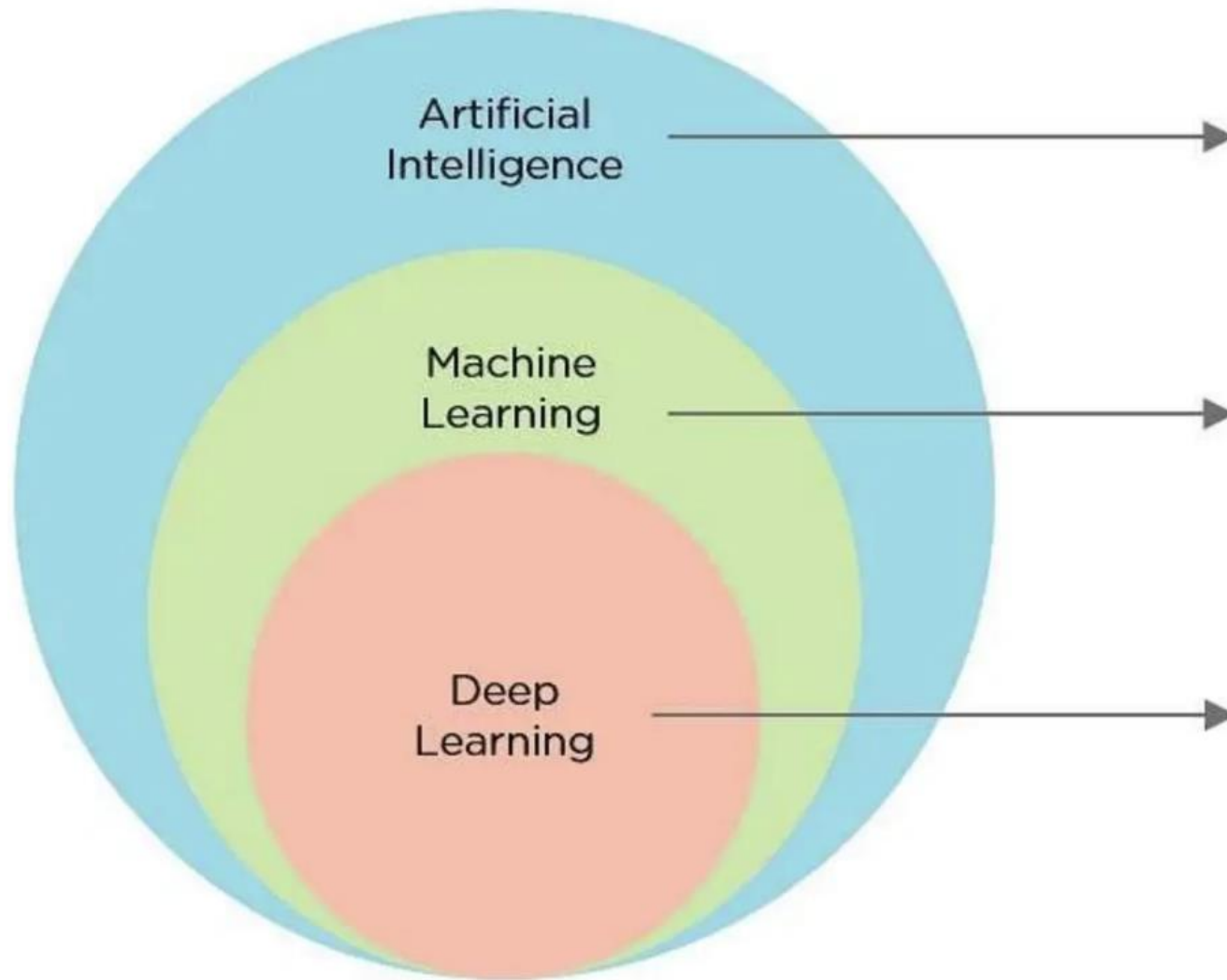


# Human Vs Machine Learning

Relaxed Alertness



Human	Machine
Rest and Digest	Training
Fight-or-flight	Prediction



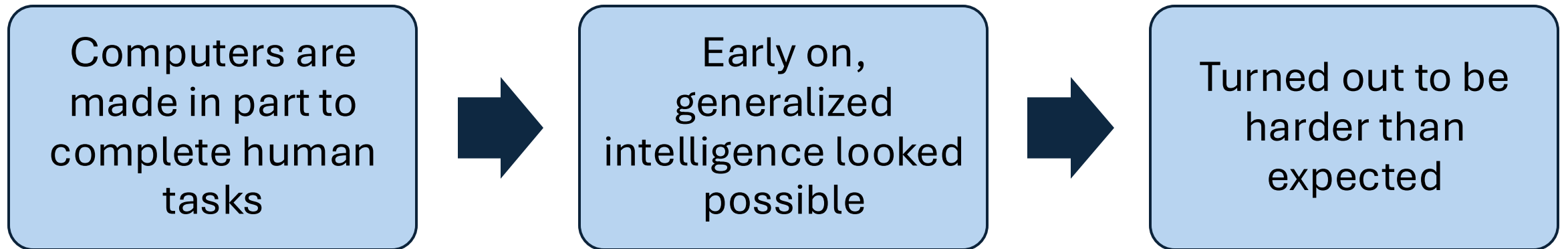
Ability of a machine to imitate intelligent human behavior

Application of AI that allows a system to automatically learn and improve from experience

Application of Machine Learning that uses complex algorithms and deep neural nets to train a model

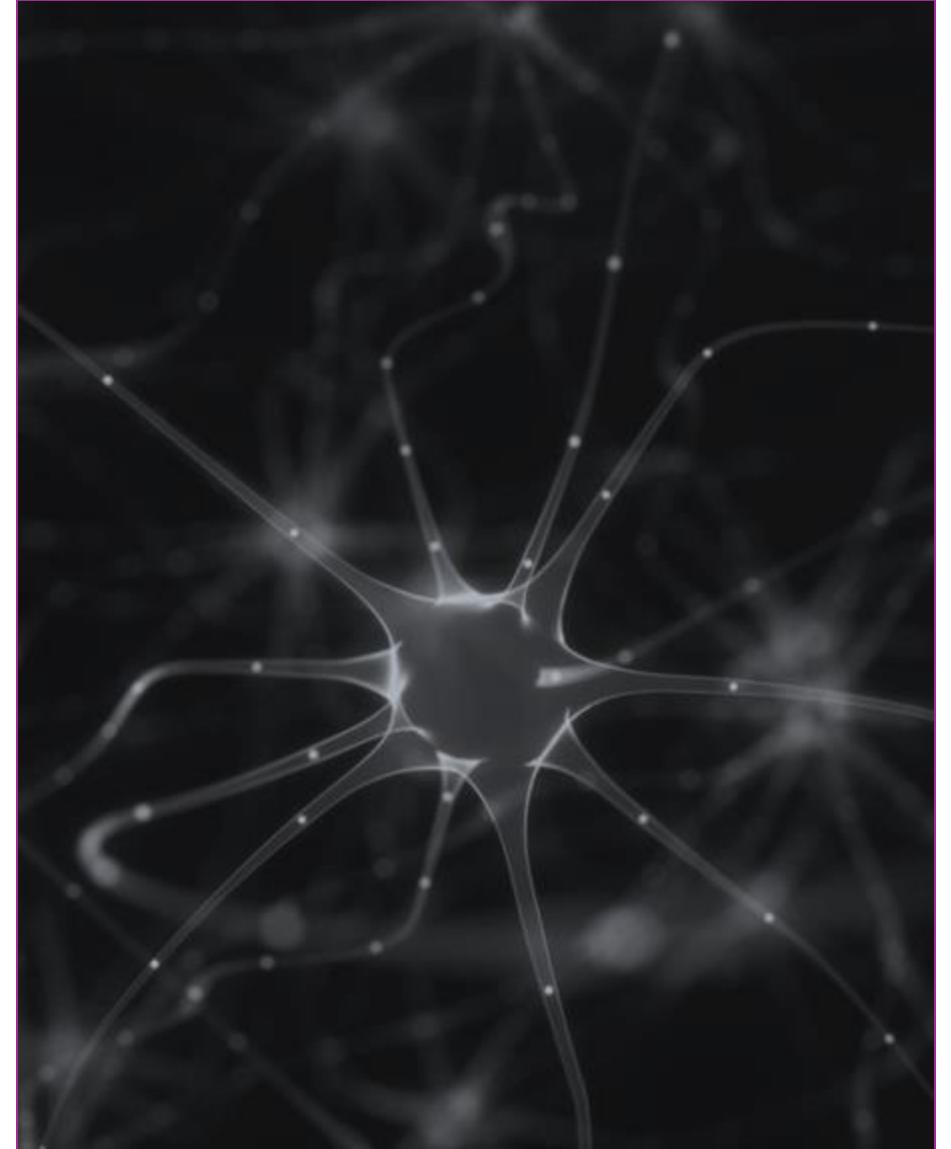


# Beginning of Artificial Intelligence



# Early Neural Networks

- Inspired by biology
- Created in the 1950's
- Outclassed by Von Neumann Architecture

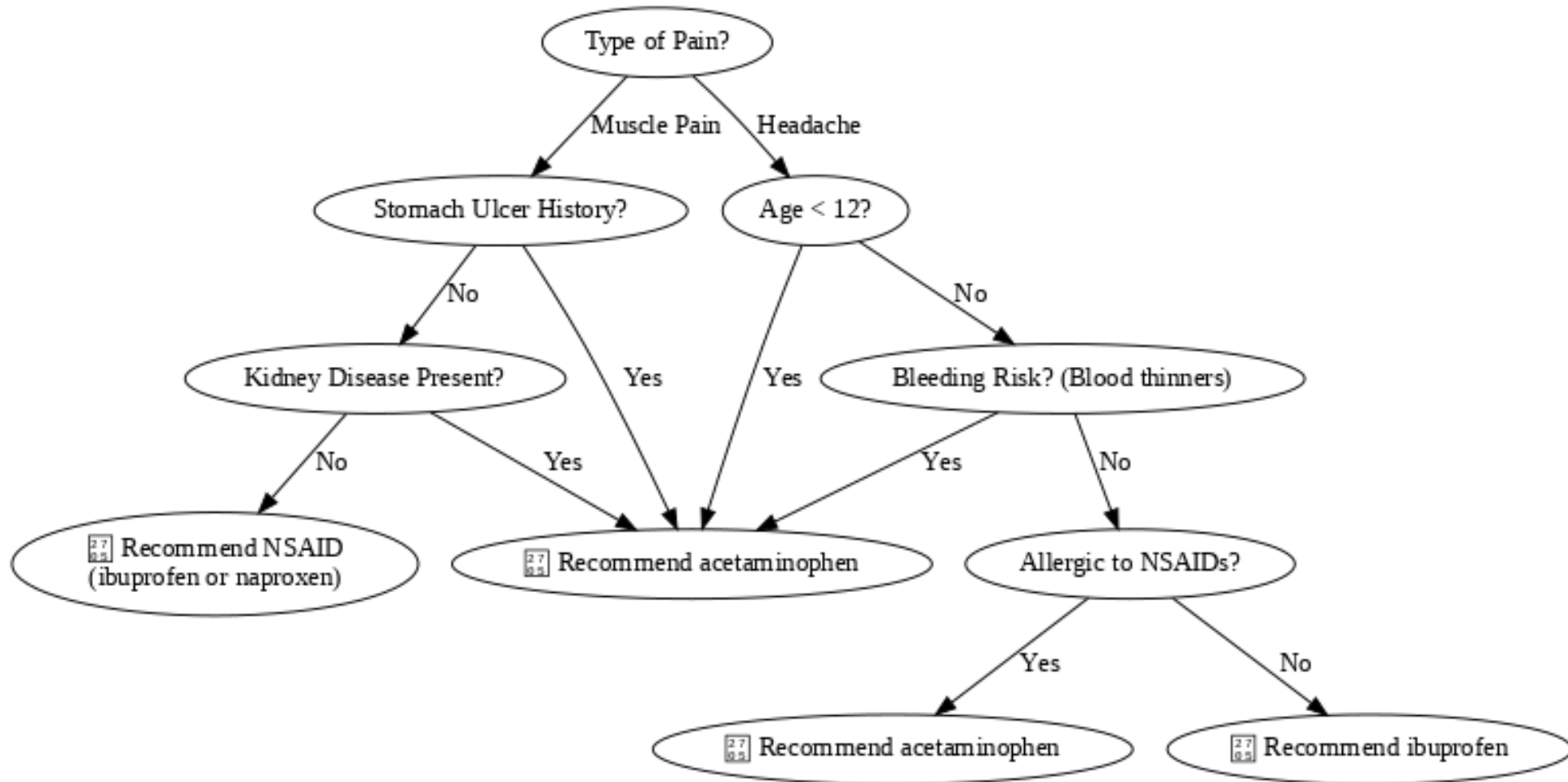


# Artificial Intelligence

“AI is ... the science and engineering of making intelligent machines.” -  
John McCarthy



# Symbolic Artificial Intelligence



Symbolic AI: Based on high-level representations of problem, logic, and search.

# Symbolic AI



Highly complex

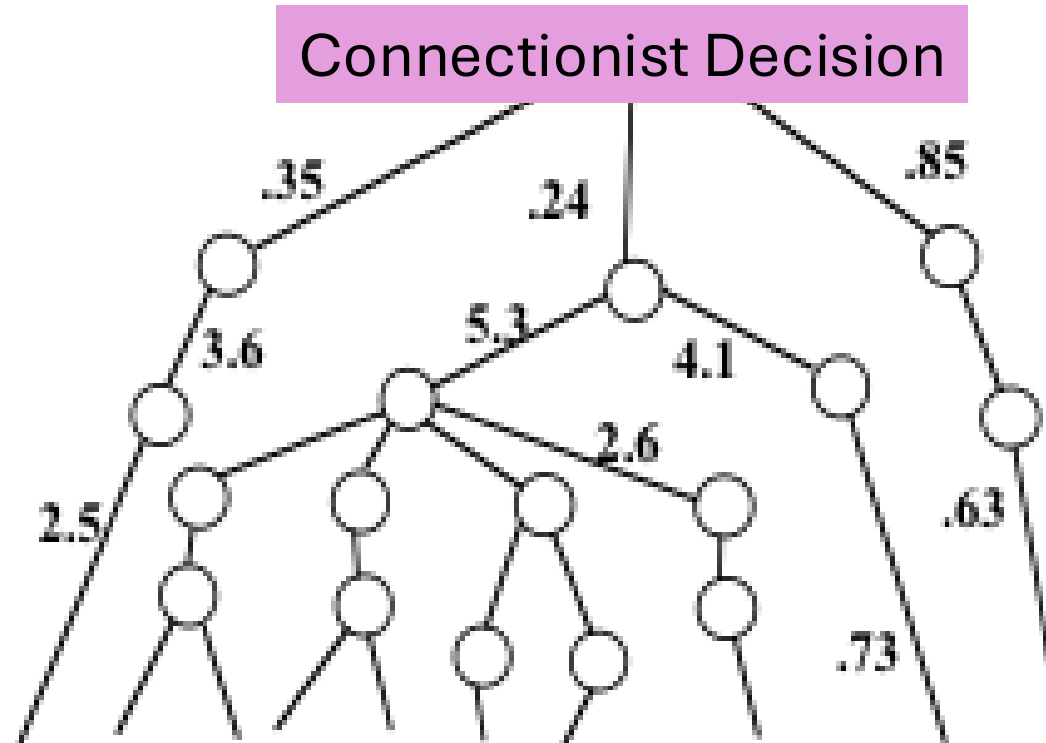


Programmed by hundreds of engineers



Rigorous programming of many rules

# Connectionist Artificial Intelligence



Using Artificial Neural Networks to build intelligent machines.



# How Do Children Learn?



- Expose them to lots of data
- Give them the “correct answer”
- They will pick up the important patterns on their own

# *the* **DEEP LEARNING Revolution**



# Data

- Networks need a lot of information to learn from.
- The digital era and the internet has supplied that data.



# Computing Power

Need a way for our artificial “brain” to observe lots of data within a practical amount of time.



# What is deep learning?



“

Deep learning flips traditional programming on its head

”

# Traditional Programming for Building a Classifier

- 1 Define a set of rules for classification
- 2 Program those rules into the computer
- 3 Feed it examples, and the program uses the rules to classify



# Machine Learning for Building a Classifier

- 1 Show model the examples with the answer of how to classify
- 2 Model takes guesses, we tell it if it's right or not
- 3 Model learns to correctly categorize as it's training. The system learns the rules on its own



This is a Fundamental Shift

# When to choose deep learning?

Classic Programming

If rules are clear and straightforward, often better to program it

Deep Learning

If rules are nuanced, complex, difficult to discern, use deep learning

# Problem Statement



## **Problem: Is this patient at risk for diabetes?**

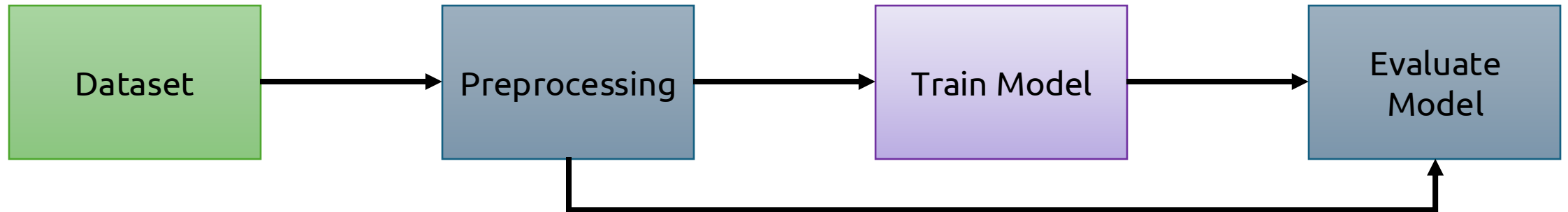
You have data on 1000 patients with just two pieces of information:

- Age
- BMI (Body Mass Index)

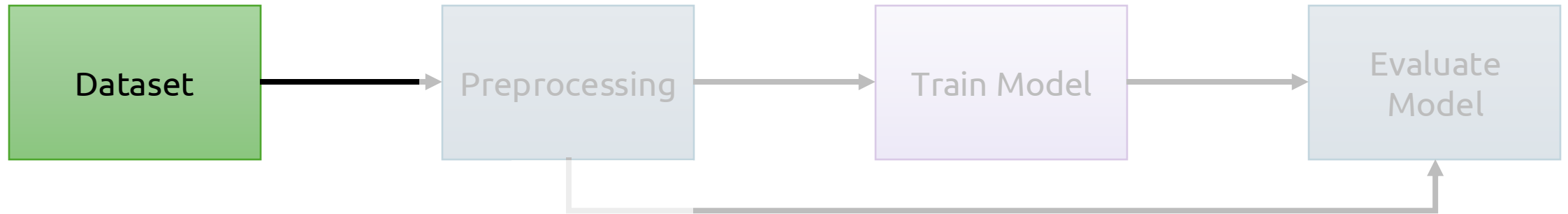
And you know which ones developed diabetes within 5 years.

**The Classification Task:** When a new patient walks in (age 45, BMI 28), predict: Will they develop diabetes? YES, or NO?

# Machine Learning Pipeline



# Machine Learning Pipeline

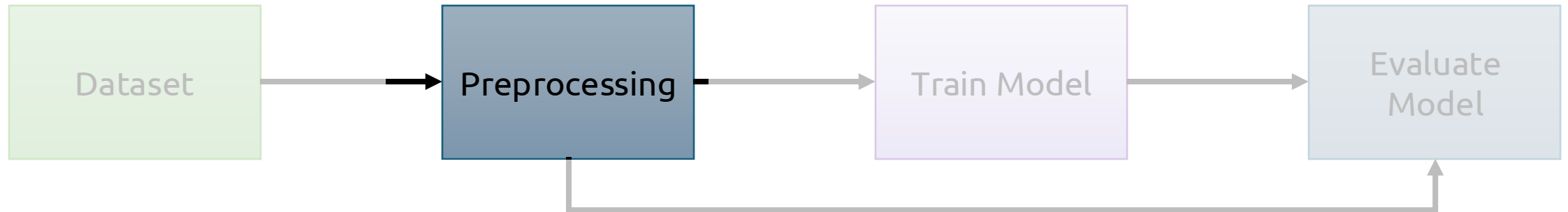




# Data

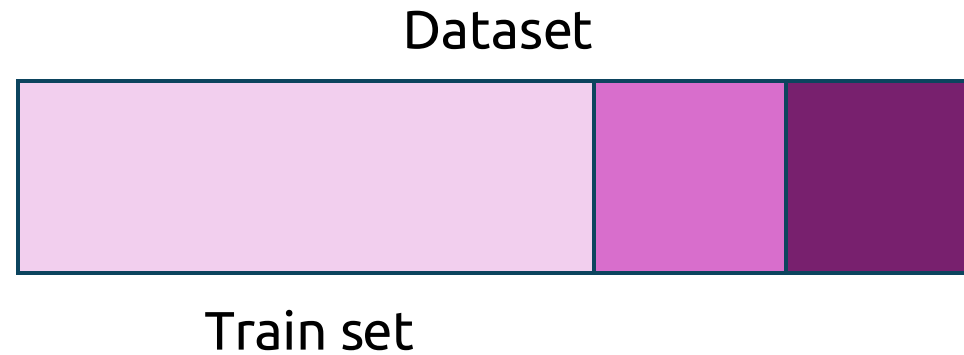
#	Age	BMI	Developed Diabetes?
0001	23	22.1	No
0002	34	24.8	No
0003	45	29.3	<b>Yes</b>
0004	28	21.5	No
0005	52	31.2	<b>Yes</b>
0006	38	26.7	No
0007	61	33.8	<b>Yes</b>
...	...	...	...
...	...	...	...
1000	29	23.4	No

# Machine Learning Pipeline



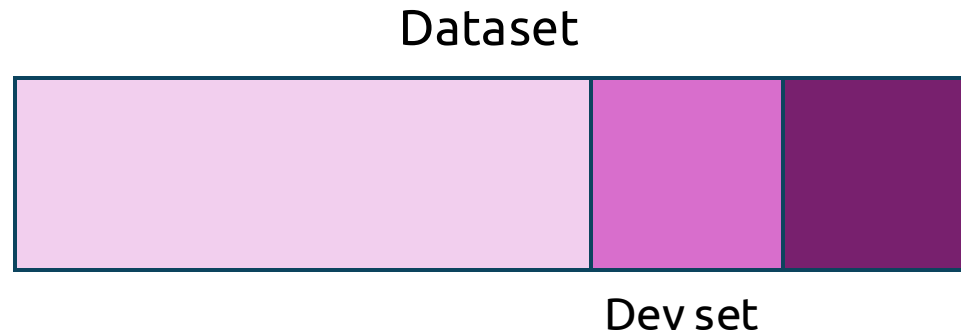
# Train, validation, and test sets

- ***Train set*** — used to adjust the parameters of the model



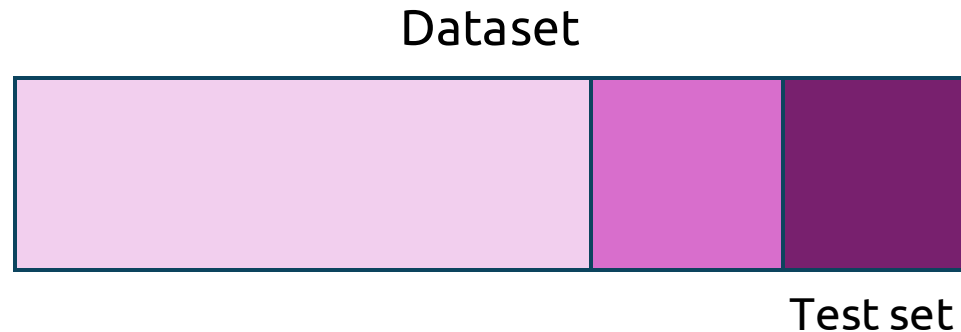
# Train, validation, and test sets

- **Train set** — used to adjust the parameters of the model
- **Validation set** — used to test how well we're doing as we develop
  - Prevents **overfitting**, something you will learn later!
  - Also known as the **development set**



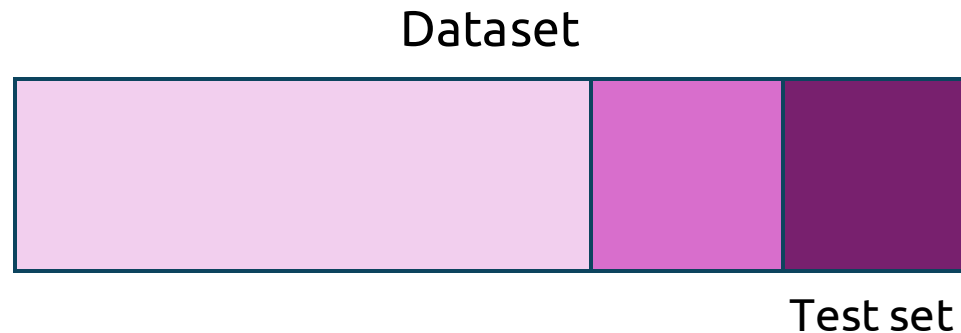
# Train, validation, and test sets

- **Train set** — used to adjust the parameters of the model
- **Validation set** — used to test how well we're doing as we develop
  - Prevents **overfitting**, something you will learn later!
- **Test set** — used to evaluate the model once the model is done



# Train, validation, and test sets

- **Train set** — used to adjust the parameters of the model
- **Validation set** — used to test how well we're doing as we develop
  - Prevents **overfitting**, something you will learn later!
- **Test set** — used to evaluate the model once the model is done



Question: Why do we need a separate test set?

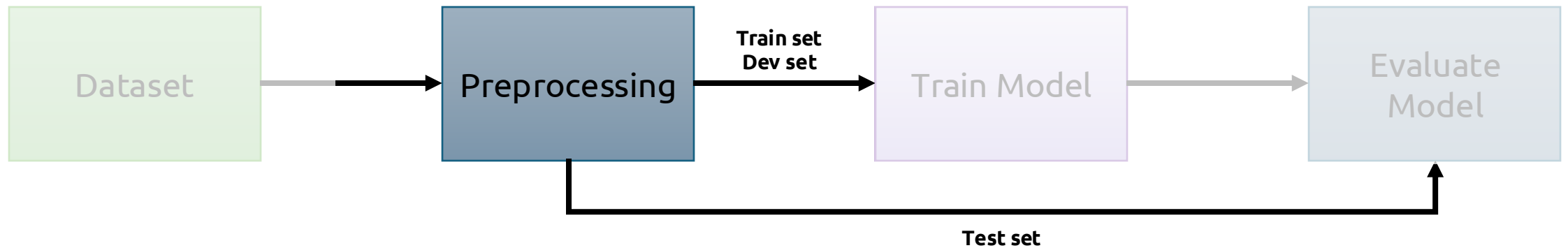


# Train, validation, and test sets

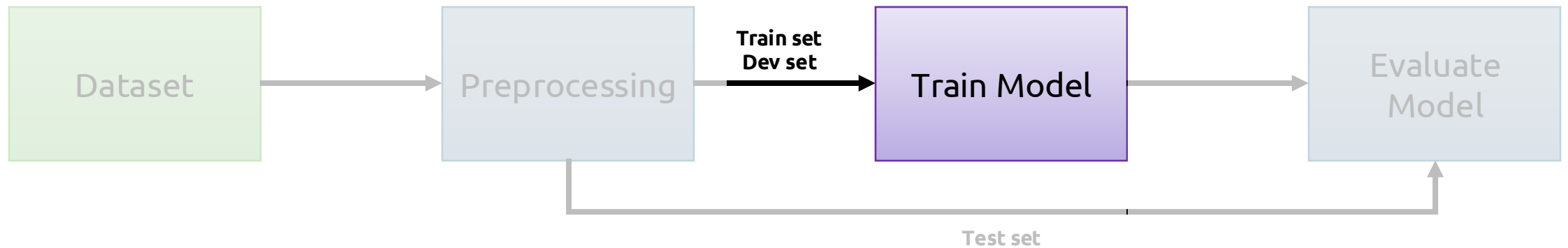
For the patient records

- 800 train records
- 100 validation records
- 100 test records

# Machine Learning Pipeline

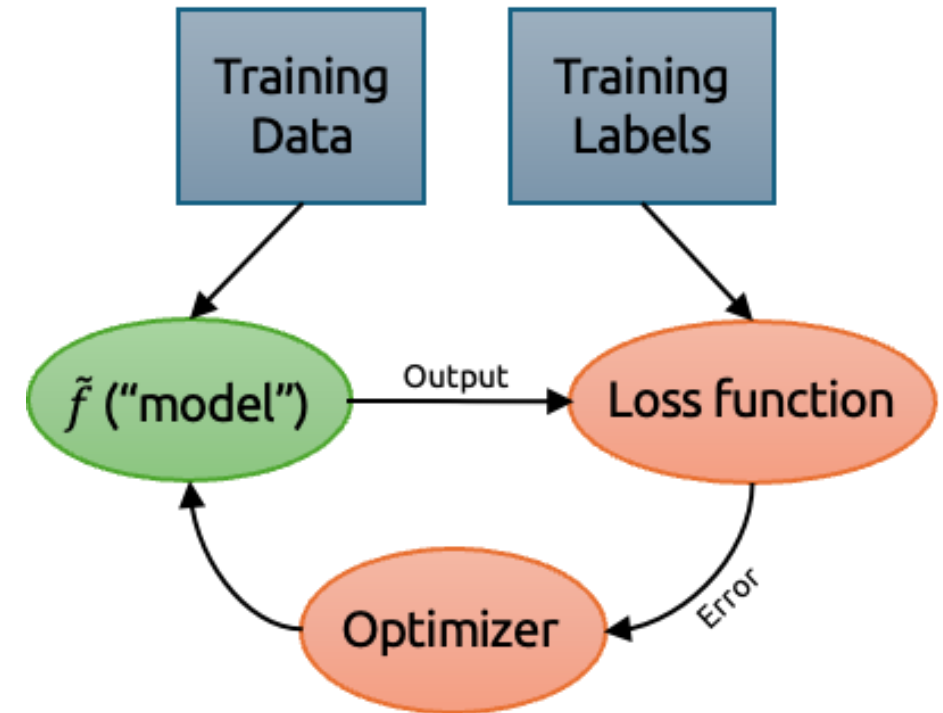


# Machine Learning Pipeline



# Classification: Problem Setup

- Each  $x$  in our dataset is called an **input**
  - $x$  is represented by 2 features:  $[a1, a2]$ .  
a1 = age, a2 is BMI
- Each  $y$  in our dataset is called a **label**
  - $y$  is the corresponding answer/classification, one of two possibilities
- We refer to each  $(x, y)$  as an **example**
- This is a **supervised learning** task



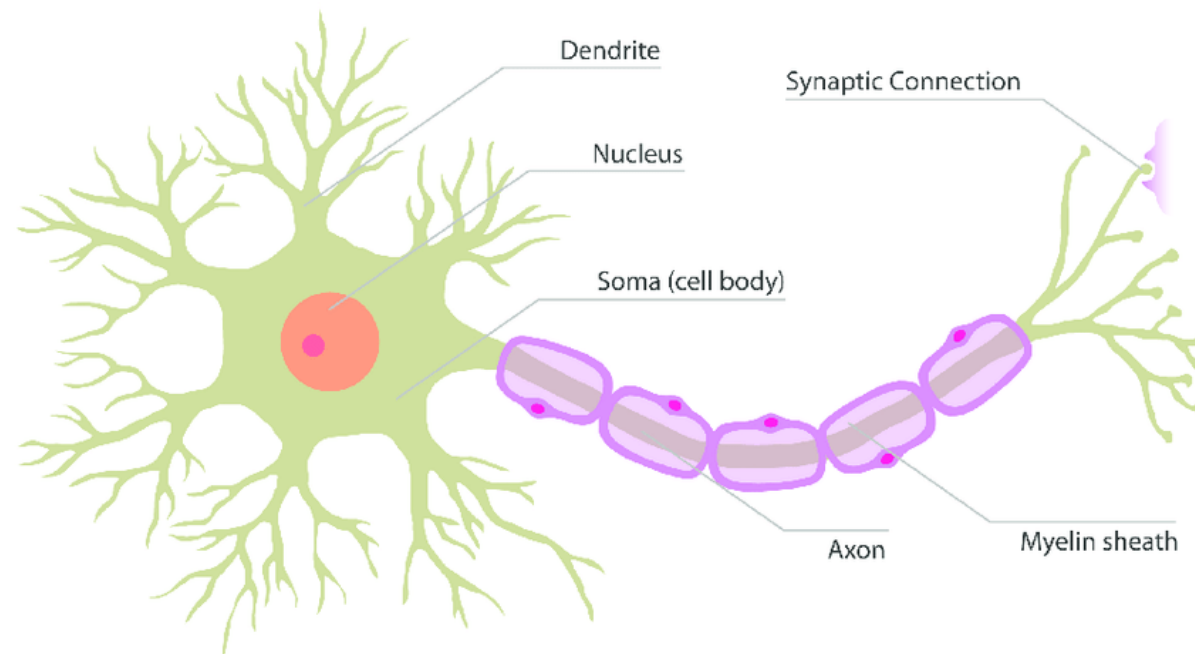
# Perceptron: Our first neural network!

- “An electronic device which was constructed in accordance with biological principles and showed an ability to **learn**”
- Proposed by Frank Rosenblatt in 1958 (Cornell Aeronautical Laboratory)
- Initially simulated on an IBM 704 computer

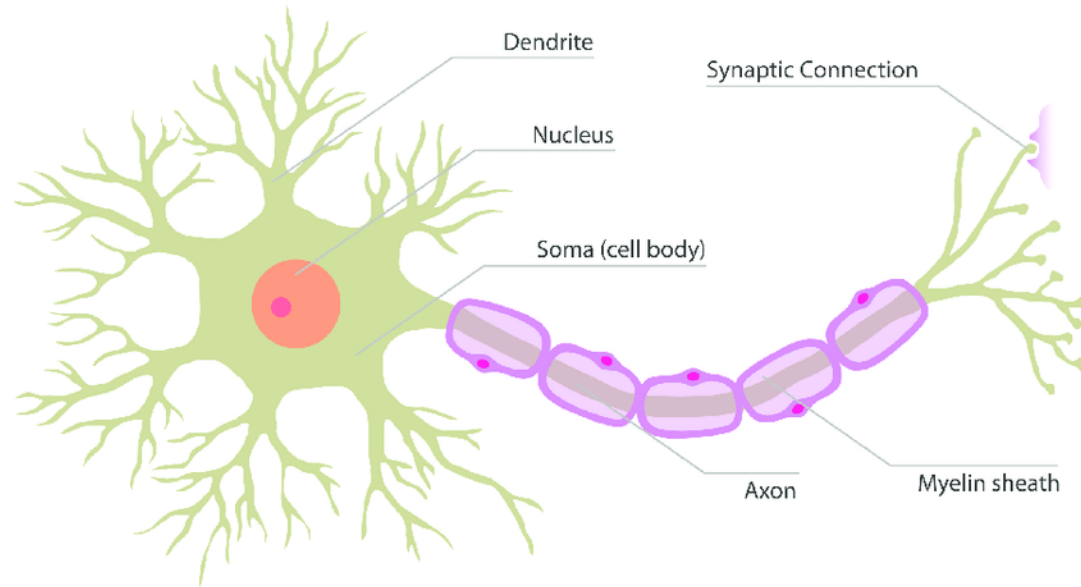


# Biological motivation

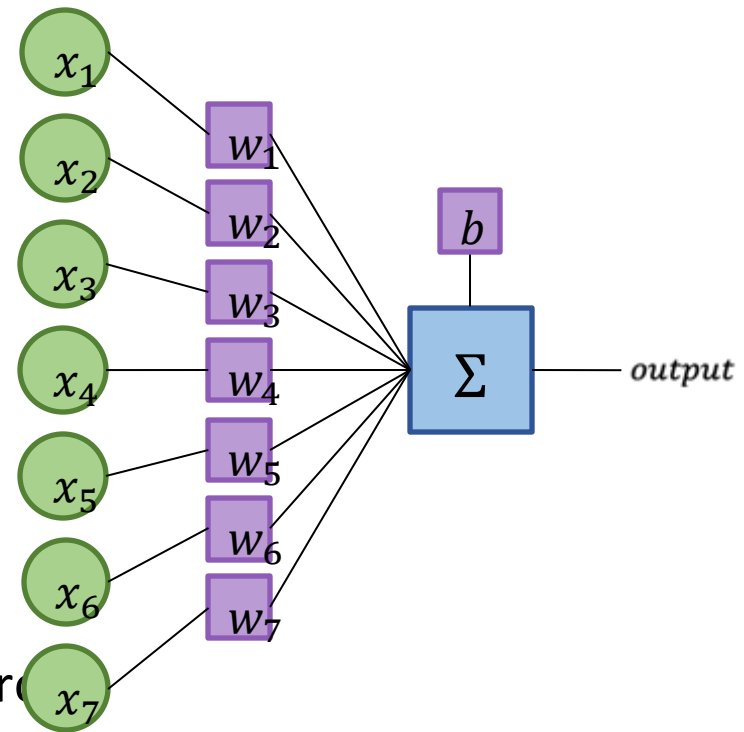
- Loosely inspired by neurons, basic working unit of the brain
- Serve to transmit information between cells



# The Perceptron



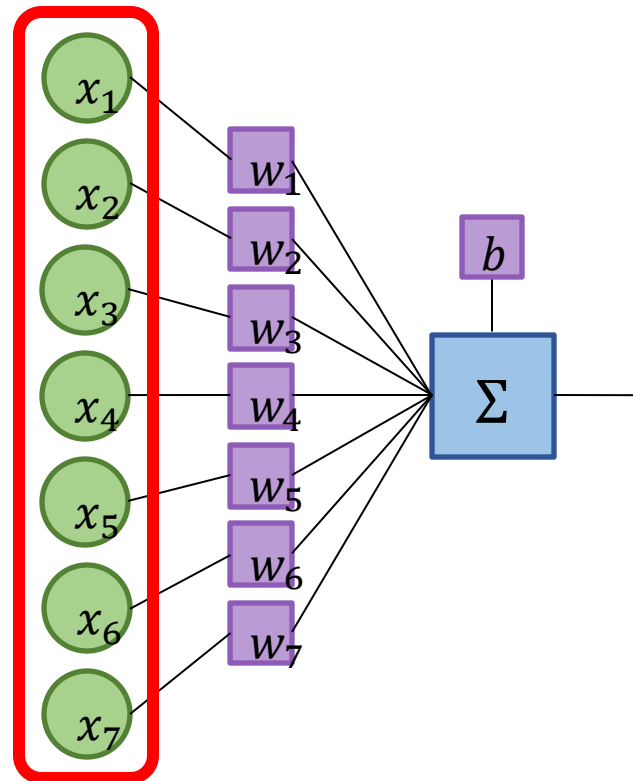
Biological Neuron



Artificial Neuron (Perceptron)

# Input

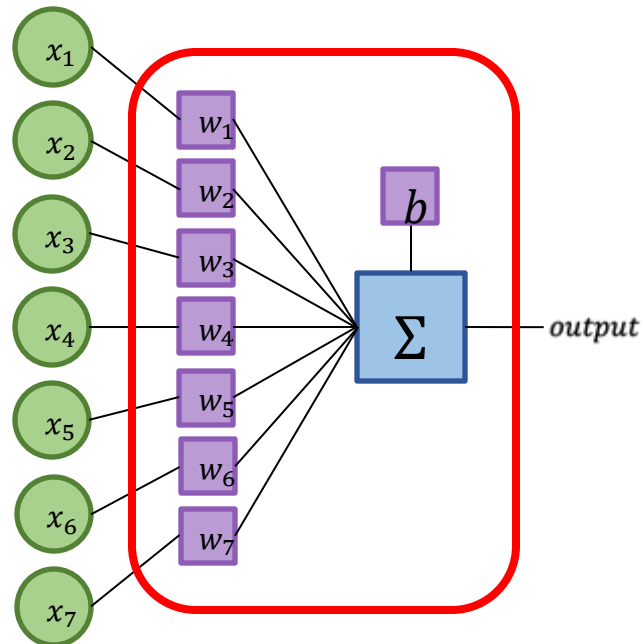
- Input: a vector of numbers  $\mathbf{x} = [x_1, x_2, \dots, x_n]$





# Predicting with a Perceptron

1. Multiply each input  $x_i$  by its corresponding weight  $w_i$ , sum them up.
2. Add the bias  $b$



# Predicting with a Perceptron

1. Multiply each input  $x_i$  by its corresponding weight  $w_i$ , sum them up.
2. Add the bias  $b$
3. If the result value is greater than 0, return 1, otherwise return 0

$$f_{\Phi}(x) = \begin{cases} 1, & \text{if } b + \sum_{i=0}^n w_i x_i > 0 \\ 0, & \text{otherwise} \end{cases}$$

4. As a binary classifier, 1 indicates that  $x$  is a member of the class and 0, not a member

# Linear Neurons

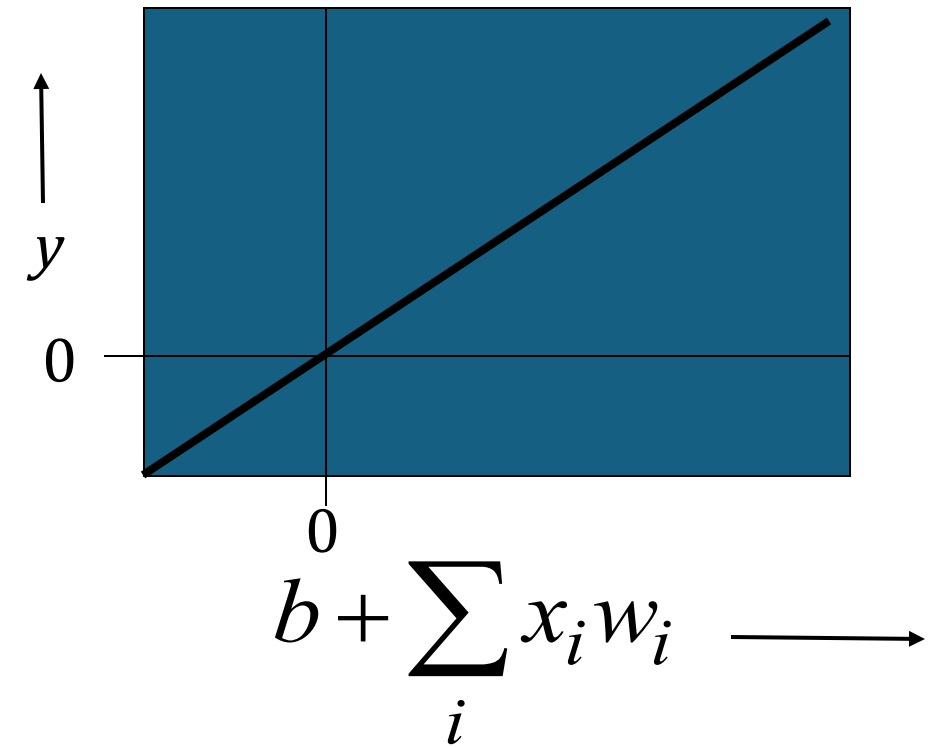
These are simple but computationally limited.

**i-th Input**

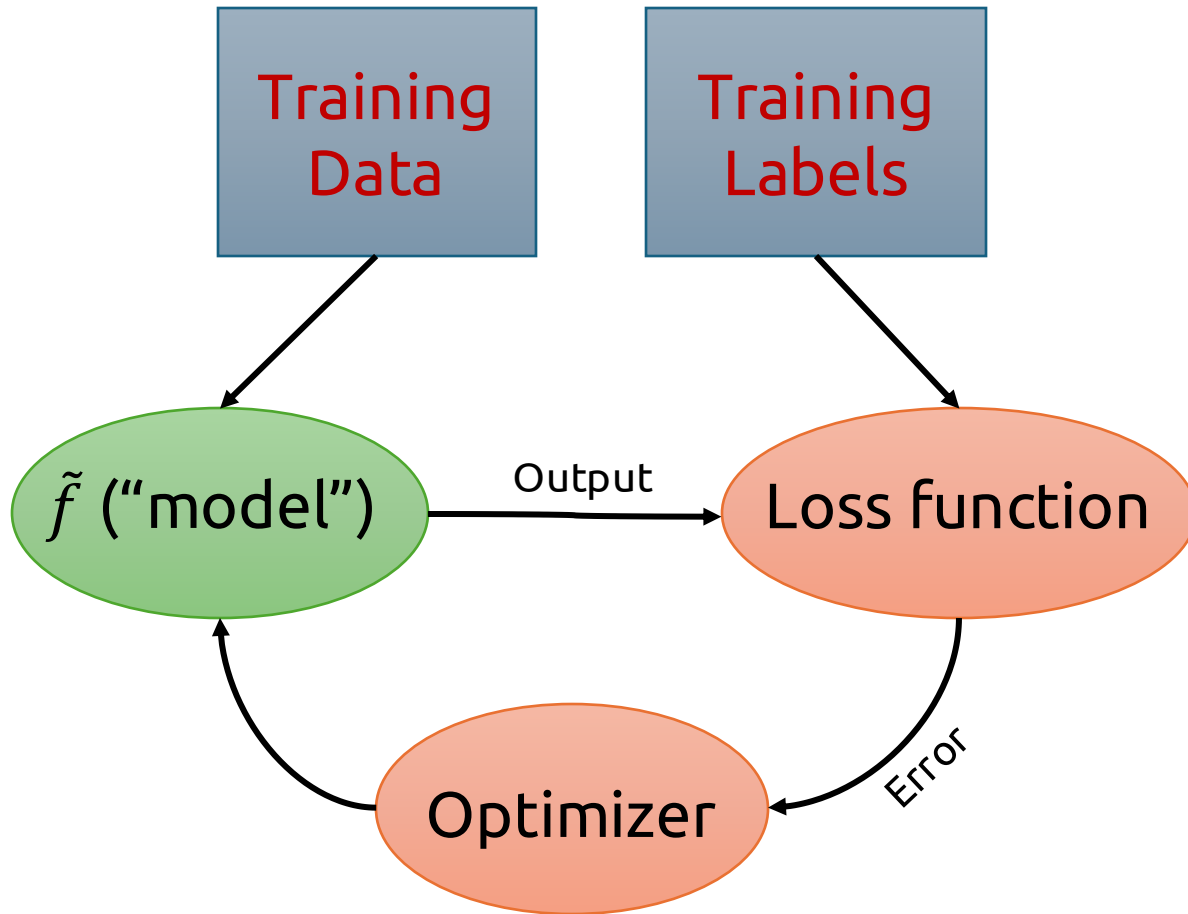
**output**  $y = \sum_i x_i w_i$

**Index over input connections**

**Weights on i-th neuron**

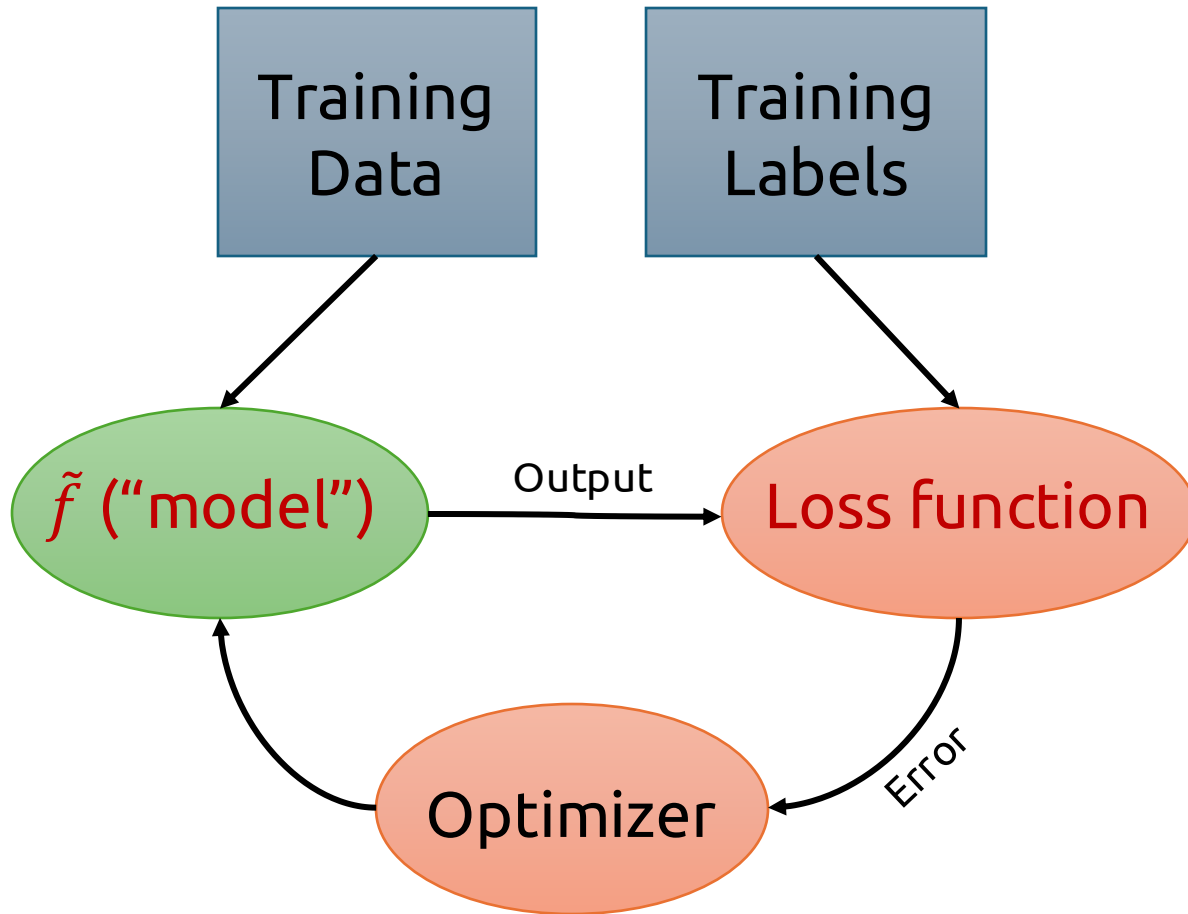


# Supervised Learning



Collect training data and labels

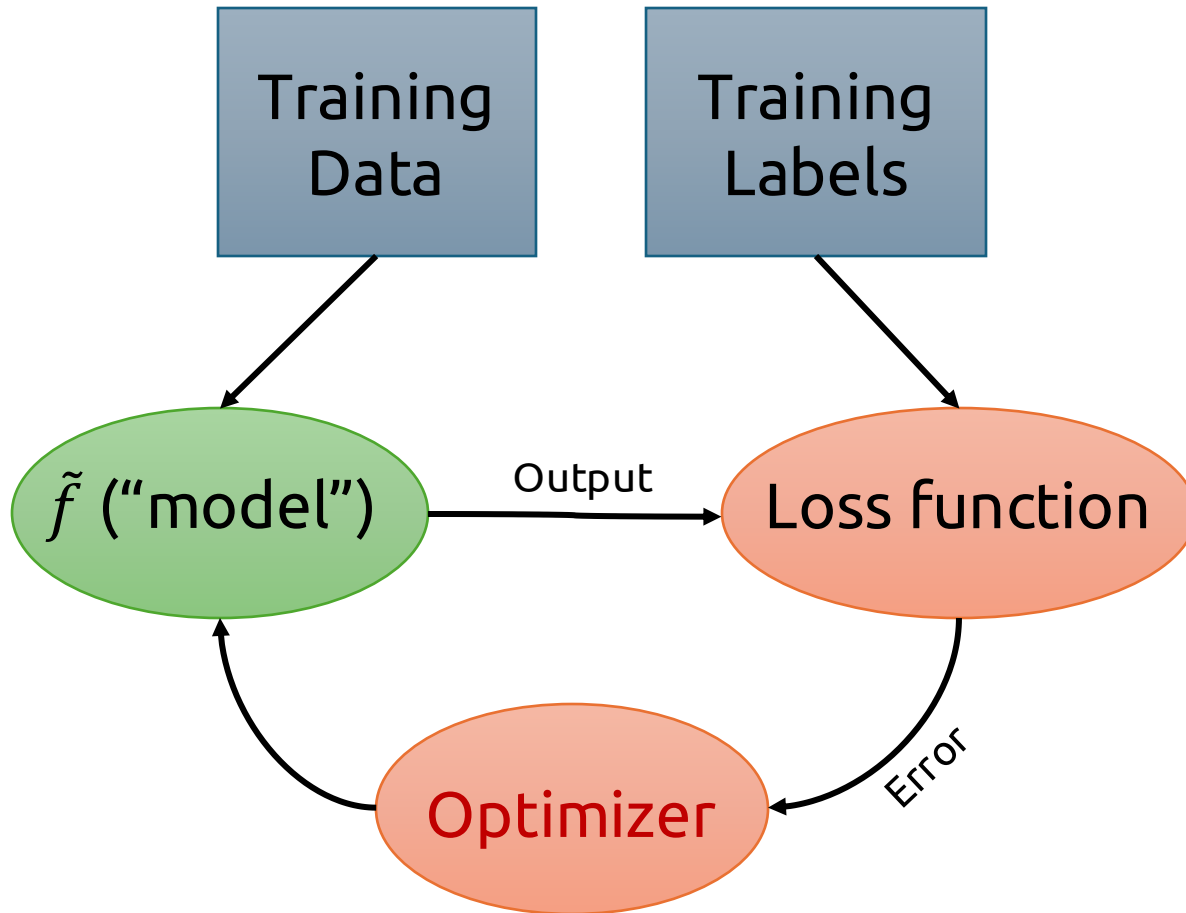
# Supervised Learning



Collect training data and labels

Specify hypothesis class  $\mathcal{H}$  and loss function

# Supervised Learning

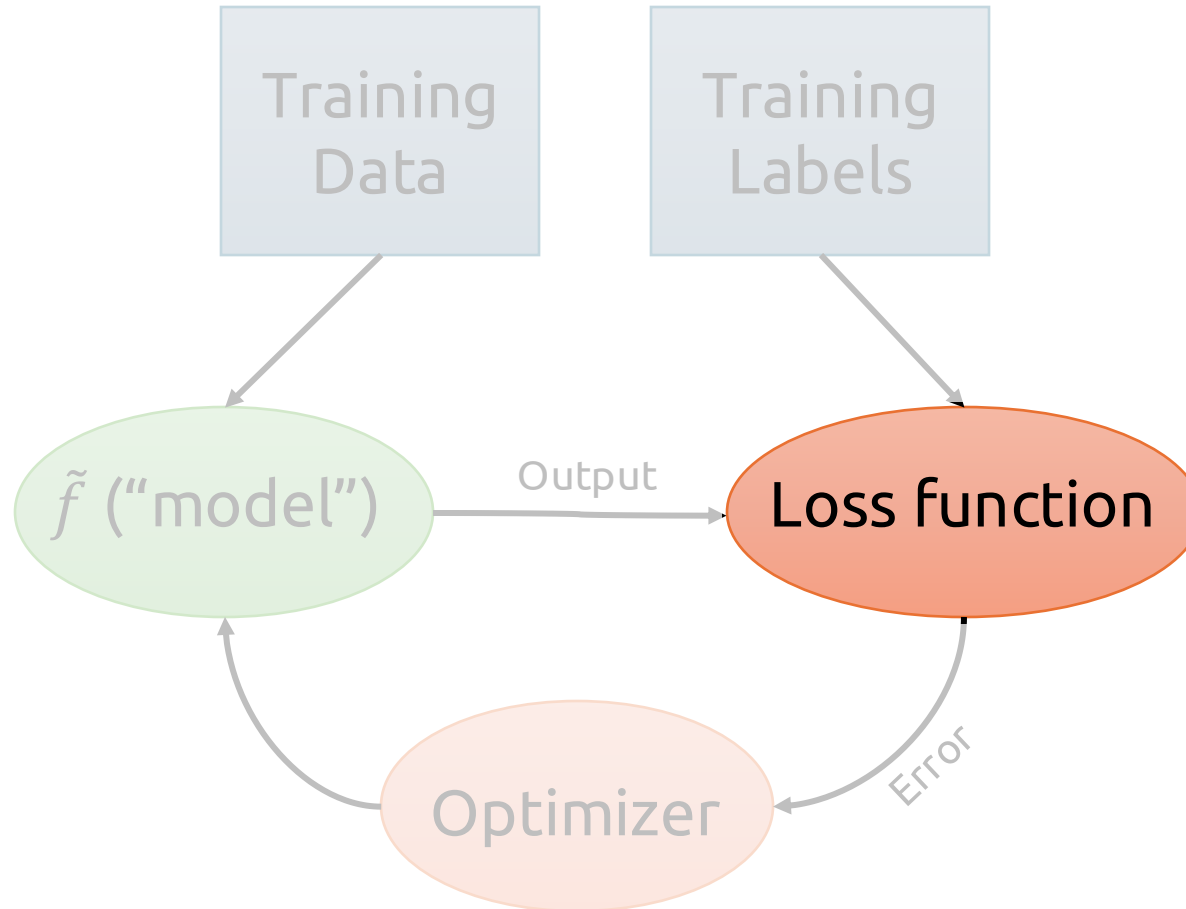


Collect training data and labels

Specify hypothesis class  $\mathcal{H}$  and loss function

Learning: Find  $f \in \mathcal{H}$  that minimizes the *empirical loss* on *training* data

# A Critical Ingredient: Loss function



Loss function: How far  $f(x)$  is from  $y$

We want to minimize the loss.

# Loss function – Measure of How Wrong Are We?



Imagine you're a pharmacy student doing clinical rotations. Every time you predict whether a patient will develop diabetes, you're either **right** or **wrong**. But how do we measure "how wrong" the computer is so it can improve?

## Simple Scoring System

Let's say our algorithm doesn't just say "YES" or "NO" - it gives a **confidence score** between 0 and 1:

- 0.1 = "Very unlikely to get diabetes"
- 0.9 = "Very likely to get diabetes"
- 0.5 = "50-50 chance"



# Loss function – Measure of How Wrong Are We?

## Real Example with Our Patients:

Patient	Age	BMI	Actually Got Diabetes?	Algorithm's Confidence	How Wrong?
John	45	28	<b>YES</b> ✓	0.8 (80% sure YES)	Small error
Mary	32	23	<b>NO</b> ✓	0.2 (20% sure YES)	Small error
Bob	50	30	<b>YES</b> ✓	0.3 (30% sure YES)	<b>BIG ERROR!</b>

## The Loss Function is Like a Penalty System

Think of it like losing points in a video game:

### When the algorithm is confident AND correct:

- Patient has diabetes, algorithm says 90% chance → **Small penalty** (like -1 point)

### When the algorithm is confident BUT wrong:

- Patient has diabetes, algorithm says 10% chance → **HUGE penalty** (like -100 points)

### When the algorithm is uncertain:

- Patient has diabetes, algorithm says 50% chance → **Medium penalty** (like -20 points)

# Loss function – Measure of How Wrong Are We?

## Real Example with Our Patients:

Patient	Age	BMI	Actually Got Diabetes?	Algorithm's Confidence	How Wrong?
John	45	28	<b>YES</b> ✓	0.8 (80% sure YES)	Small error
Mary	32	23	<b>NO</b> ✓	0.2 (20% sure YES)	Small error
Bob	50	30	<b>YES</b> ✓	0.3 (30% sure YES)	<b>BIG ERROR!</b>

The **binary loss function** calculates these penalties mathematically. The algorithm's goal is to **minimize total penalties** across all patients.

**Key insight:** The algorithm doesn't just want to be right - it wants to be **confidently right** and **uncertainly wrong** (rather than confidently wrong).

This is exactly like studying for an exam - you don't just want to know the right answer, you want to be confident when you know it and admit uncertainty when you don't!

# Loss Function in Classification

$$\hat{L}(f) = -\frac{1}{N} \sum_{i=1}^N y_i \cdot \log(p(y_i)) + (1 - y_i) \cdot \log(1 - p(y_i))$$

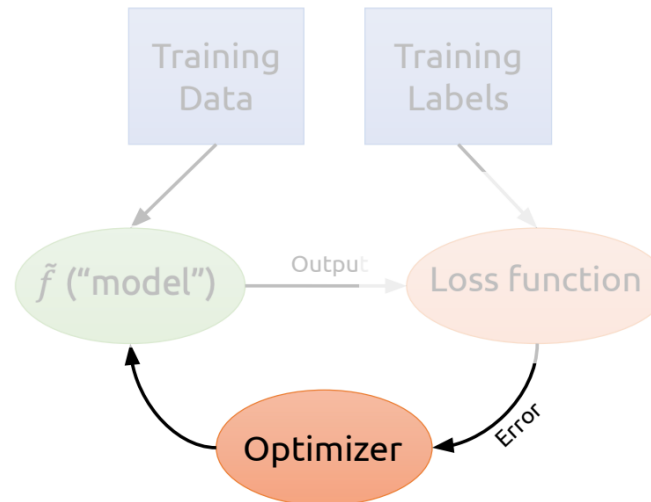
$y_i$  is the **label** (**1** for yes to diabetes and **0** for no to diabetes) and  $p(y_i)$  is the predicted **probability of the cases being yes** for all **N** points.

# Linear Regression: Optimization

- We want to minimize

$$\hat{L}(f) = -\frac{1}{N} \sum_{i=1}^N y_i \cdot \log(p(y_i)) + (1 - y_i) \cdot \log(1 - p(y_i))$$

- The algorithm to minimize the loss function is called an **optimizer**.



# Hands on Exercise

---



# Exercise



In this project, we develop a binary classification model using PyTorch to predict a patient's gender (Male or Female) based on common blood test values:

- Hemoglobin
- Platelet Count
- White Blood Cells
- Red Blood Cells
- MCV (Mean Corpuscular Volume)
- MCH (Mean Corpuscular Hemoglobin)
- MCHC (Mean Corpuscular Hemoglobin Concentration)

Link: <https://github.com/Centrum-IntelliPhysics/Artificial-Intelligence-in-Predictive-Health-and-Laboratory-Research/tree/main> for the code and the dataset

# Test case

---

- Hemoglobin: 11.8
- Platelet Count: 280000
- White Blood Cells: 7300
- Red Blood Cells: 4.3
- MCV: 85
- MCH: 28
- MCHC: 32

Can you predict the patient's gender based on the given blood test values?





# Thank you