ML – Model Fitting

CPSC 383: Explorations in Artificial Intelligence and Machine Learning
Fall 2025

Jonathan Hudson, Ph.D.
Associate Professor (Teaching)
Department of Computer Science
University of Calgary

August 27, 2025

UNIVERSITY OF CALGARY

Steps of machine learning

We will be splitting this into three parts:

First, you need to figure out **what** you want to do.

Second, you need to figure out **how** you can do it efficiently.

Third, you need to assess **whether** your solution performs well.

Design

1. Study the problem you are trying to solve (What?)
2. Choose a model class, hyperparameters (How?)
3. Prepare data (Do.)
4. Run learning algorithm to train the model (Do.)
5. Evaluate trained model (Did it work?)

Models

ML models

A **model** in machine learning is a hypothesis used for making predictions about data.

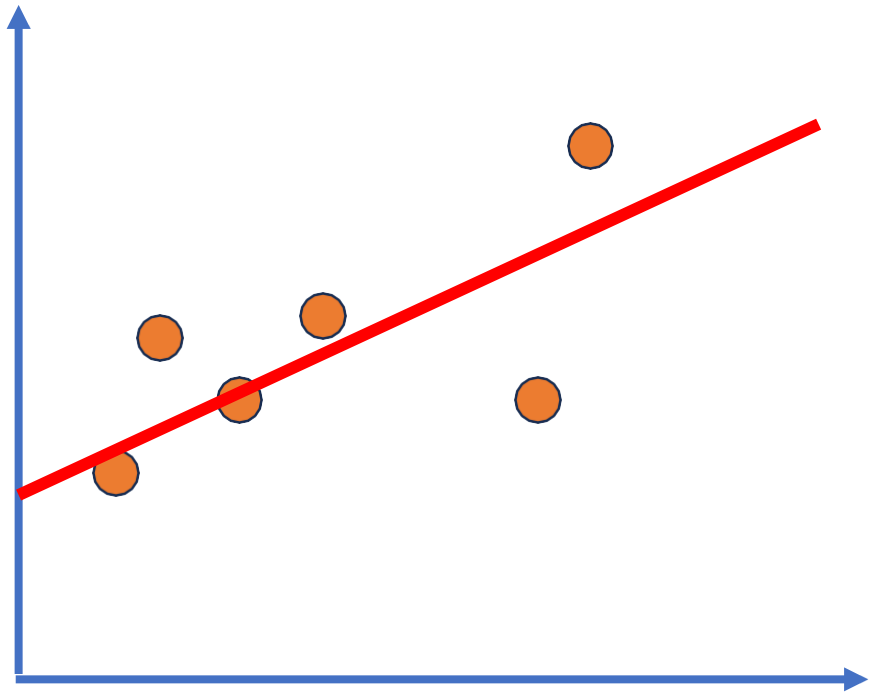
By choosing a **model class**, you are selecting a space of possible solutions for your learning algorithm to explore.

ML models

A model contains **parameters** which are adjusted by the training algorithm to best fit to your problem data.

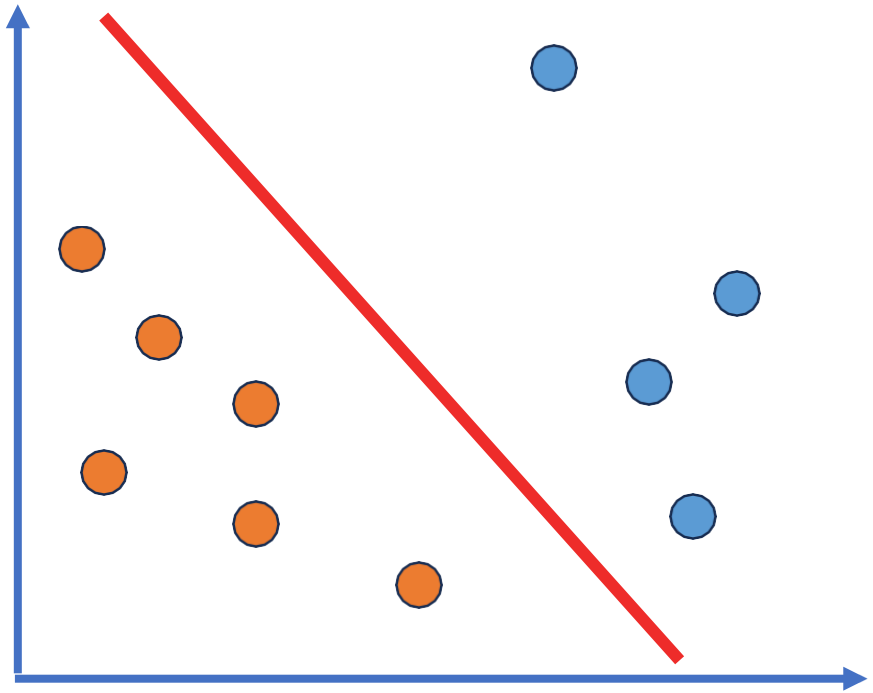
- e.g. linear regression:

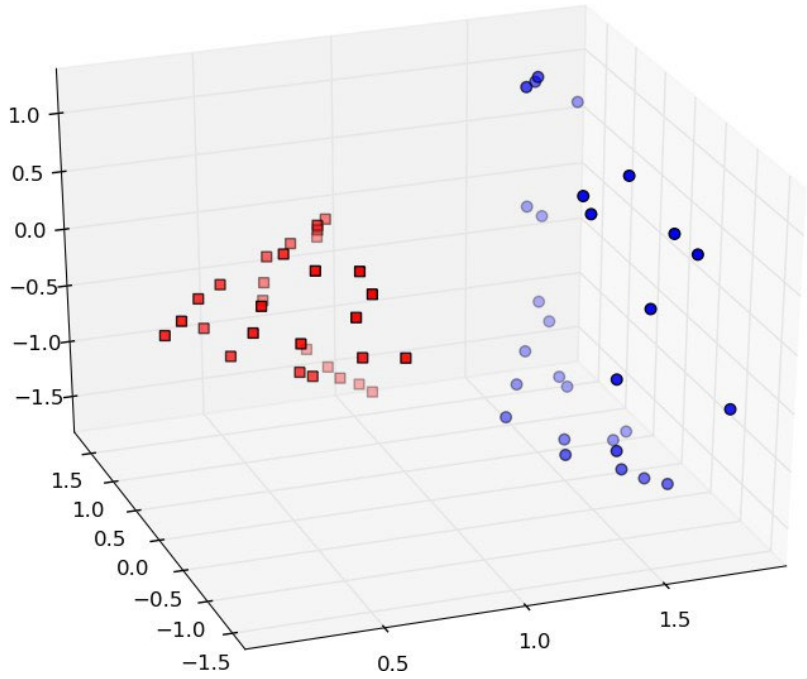
$$y = mx + b$$



ML models

A **linear classifier** is a model for binary classification that separates the dataset into the two possible outcomes using a hyperplane (line).

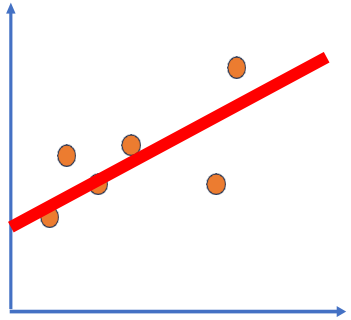




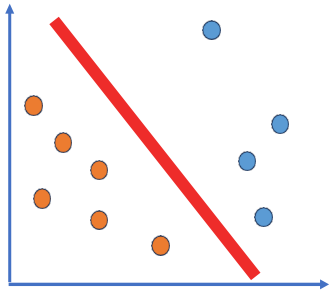
Question

What do we mean by the “best fit” in this case?

A **linear regression** is a model for regression in which a hyperplane (line) is **best fit** to your problem data.



A **linear classifier** is a model for binary classification that separates (**best fits a separation of**) the dataset into the two possible outcomes using a hyperplane (line).

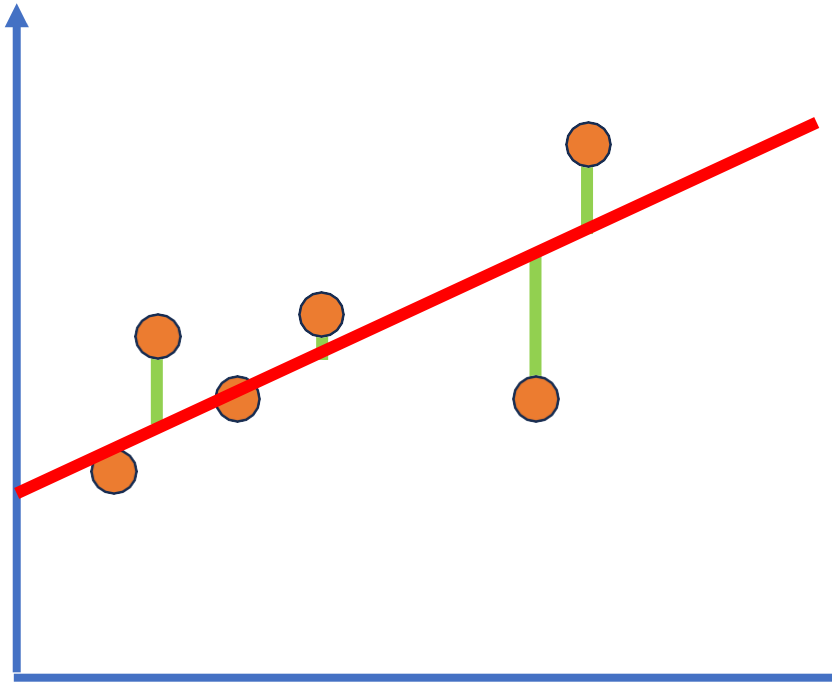


Best Fit

Example

In least-squares linear regression, the goal is to find a line that "best fits" a data set, i.e. one that minimizes the sum of the squared distances of the points to the line.

$$\min \sum \left(y_i - (mx_i + b)\right)^2$$



Loss functions

We usually measure the performance of a model on a data point in terms of a **loss function**, $L(g, a)$.

This captures how unhappy we are if our model makes a guess g when the actual correct answer is a.

Loss functions

0-1 loss: $L(g, a) = \begin{cases} 0 & \text{if } g = a \\ 1 & \text{otherwise} \end{cases}$

Squared loss: $L(g, a) = (g - a)^2$

Linear loss: $L(g, a) = |g - a|$

Cross-entropy: $L(g, a) = -a \log g$

Asymmetric loss: $L(g, a) = \begin{cases} 1 & \text{if } g = 1 \text{ and } a = 0 \\ 10 & \text{if } g = 0 \text{ and } a = 1 \\ 0 & \text{otherwise} \end{cases}$

Example

- Suppose you are training a facial recognition model to determine whether someone matches the photo scanned from their passport. Eventually, this model will be deployed at a high-security airport to check whether passengers are eligible to board.
- 1 means **eligible**, 0 means **ineligible**

- Which of the following loss functions might be more appropriate?

$$L_1(g,a) = \begin{cases} 10 & \text{if } g = 1 \text{ and } a = 0 \\ 1 & \text{if } g = 0 \text{ and } a = 1 \\ 0 & \text{otherwise} \end{cases}$$

$$L_2(g,a) = \begin{cases} 1 & \text{if } g = 1 \text{ and } a = 0 \\ 10 & \text{if } g = 0 \text{ and } a = 1 \\ 0 & \text{otherwise} \end{cases}$$

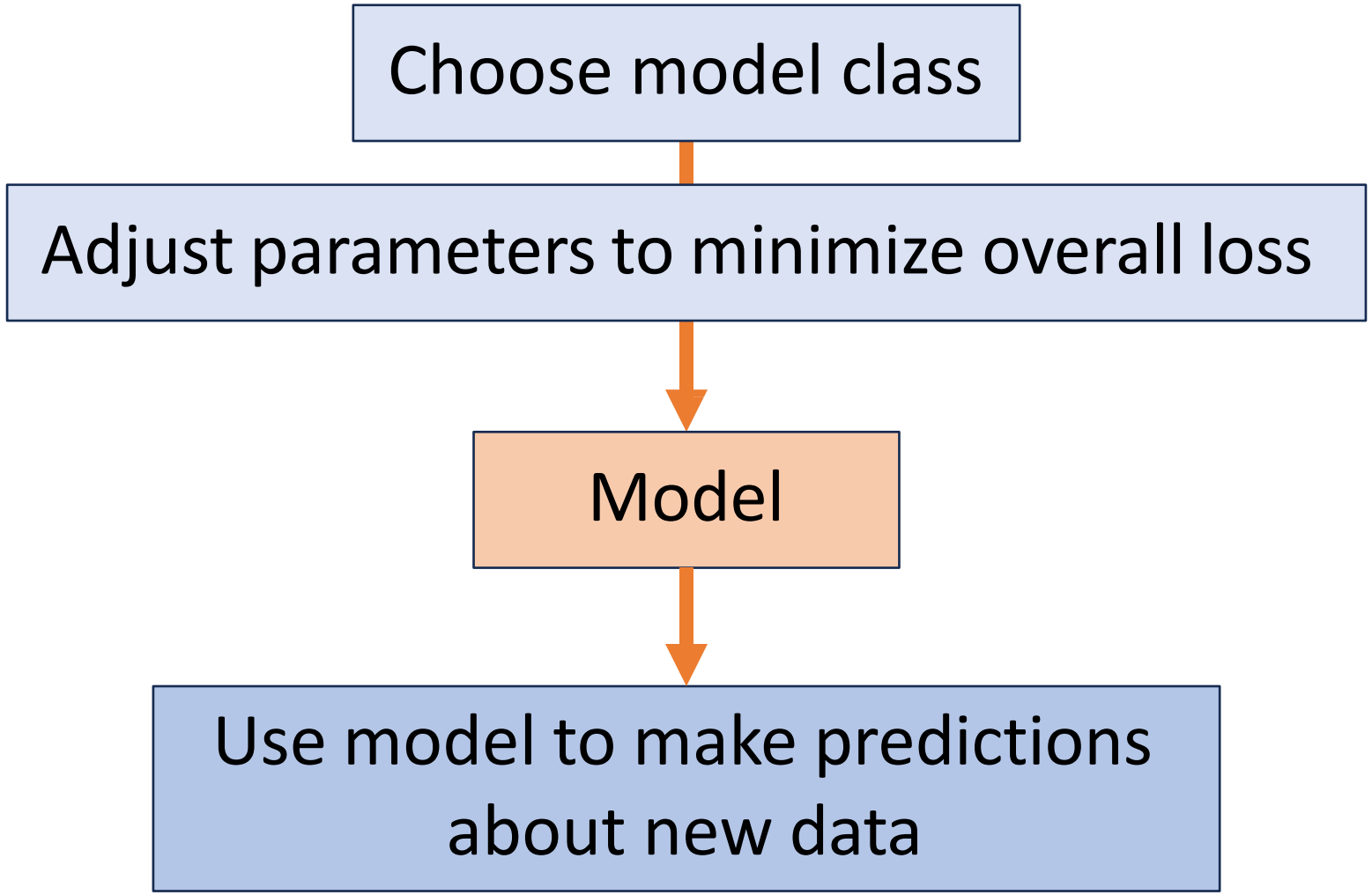
Overall loss

Given a loss function, there are many criteria we can choose to assess the model across the *entire* dataset, such as:

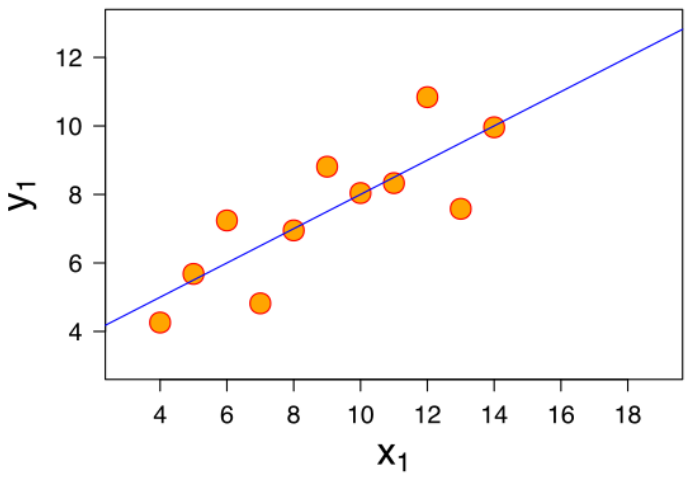
- Minimize the expected/average/total loss over all points
- Minimize the maximum loss
- Minimize regret, i.e. comparison to best model from some class
- Characterize how well the model performs asymptotically as the amount of training data increases

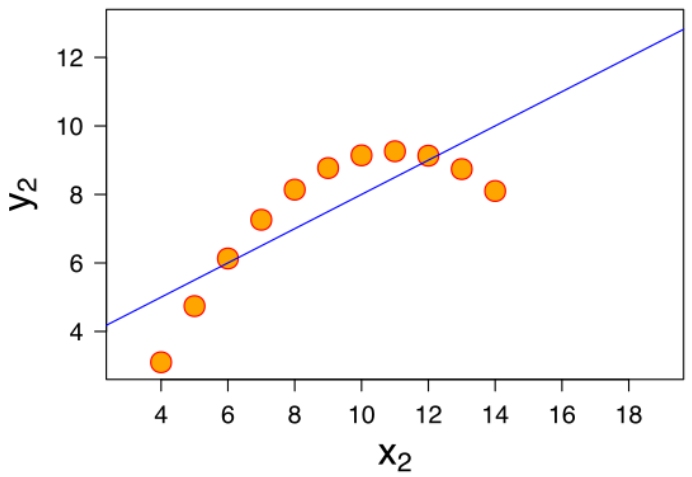
Considerations

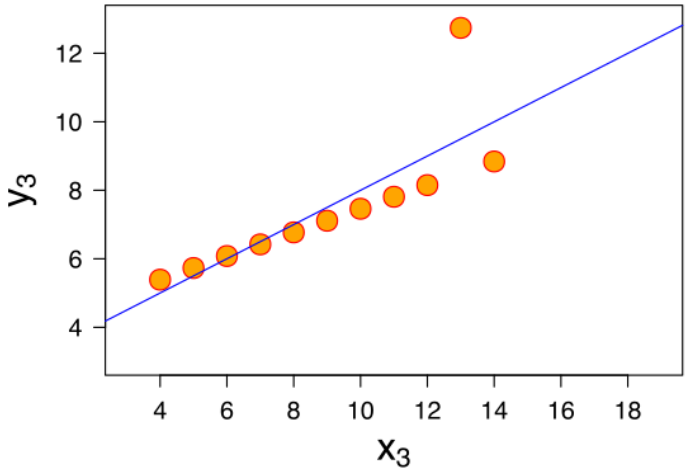
Current workflow

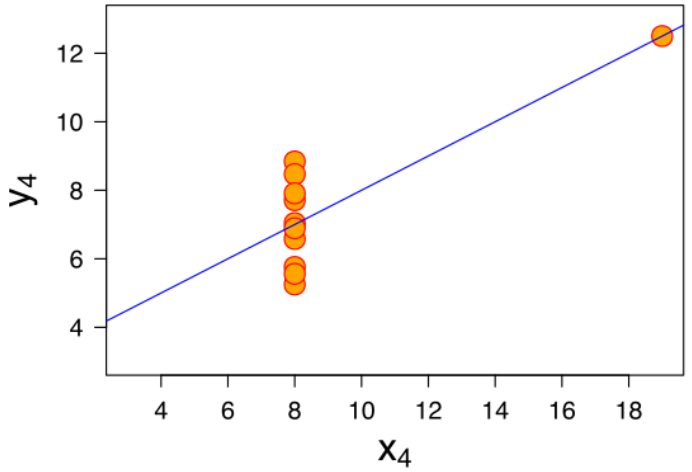


Example





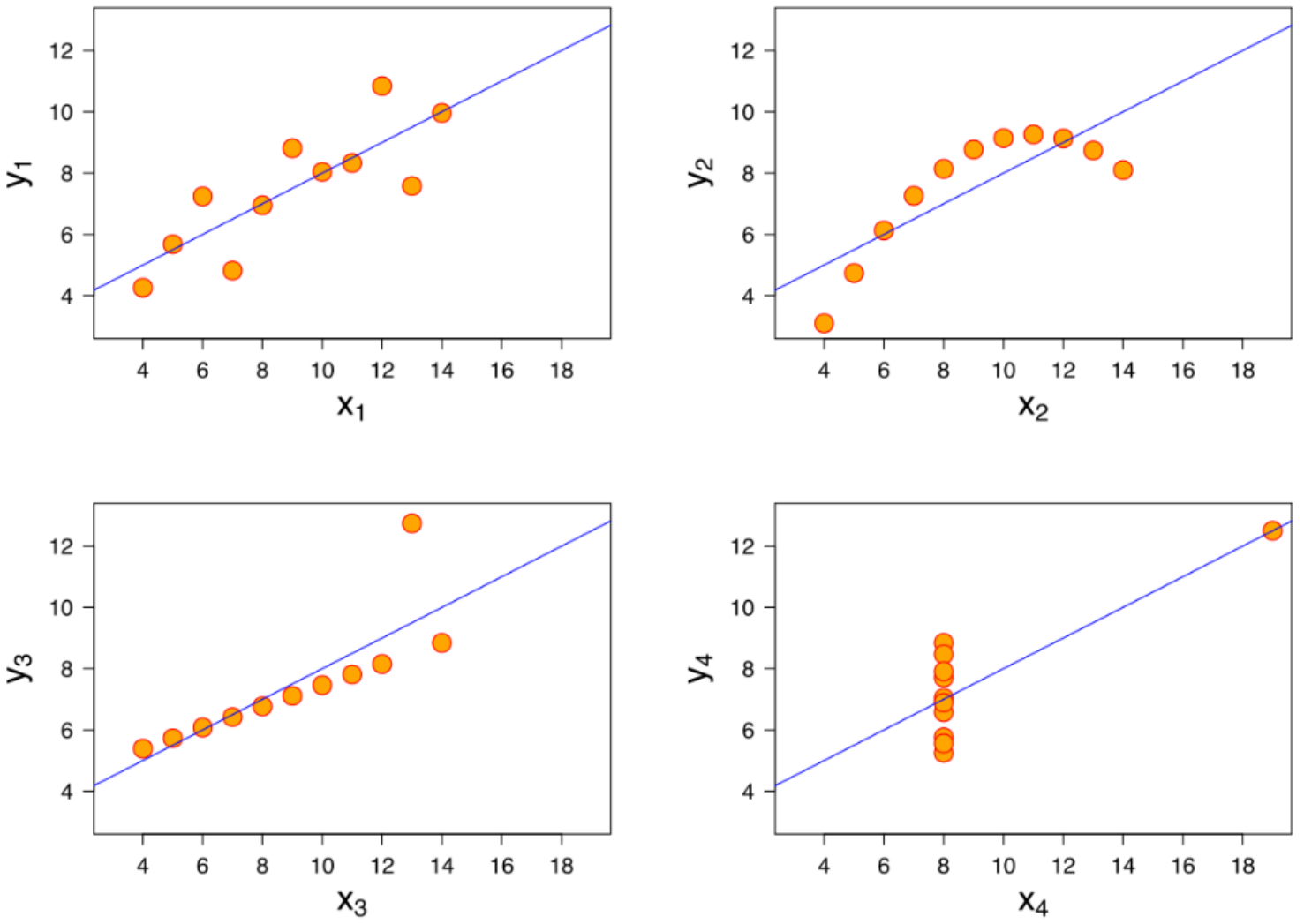




https://en.wikipedia.org/wiki/Anscombe%27s_quartet

Challenges

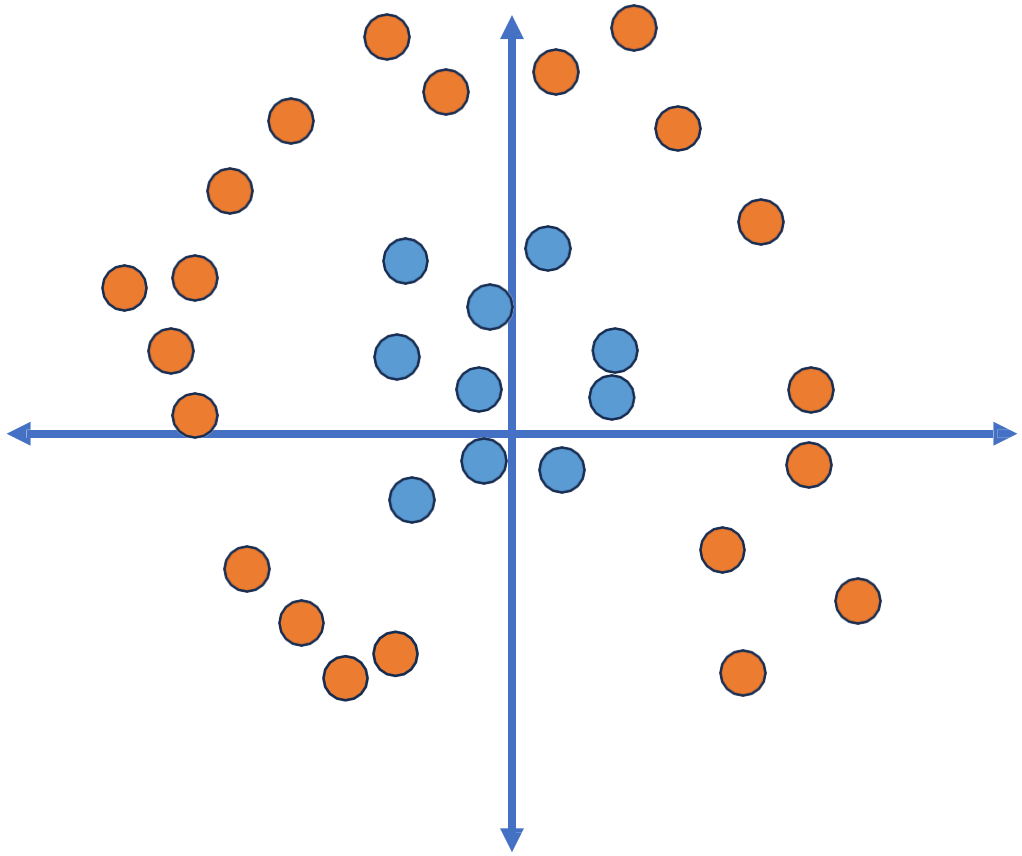
- The four data sets have identical statistics
  - Same mean, and variance
- You might produce same model for all 4 inputs
- But we might disagree that that model will actually predict future y points on the line for new x values as we desire



https://en.wikipedia.org/wiki/Anscombe%27s_quartet

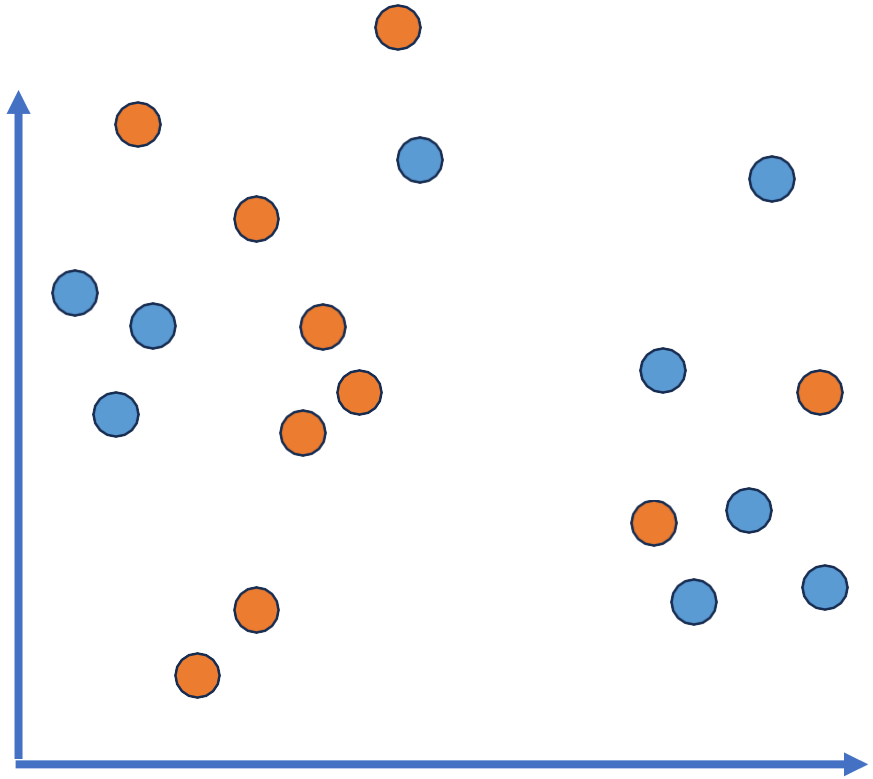
Example

How would a linear classifier perform on the following dataset?



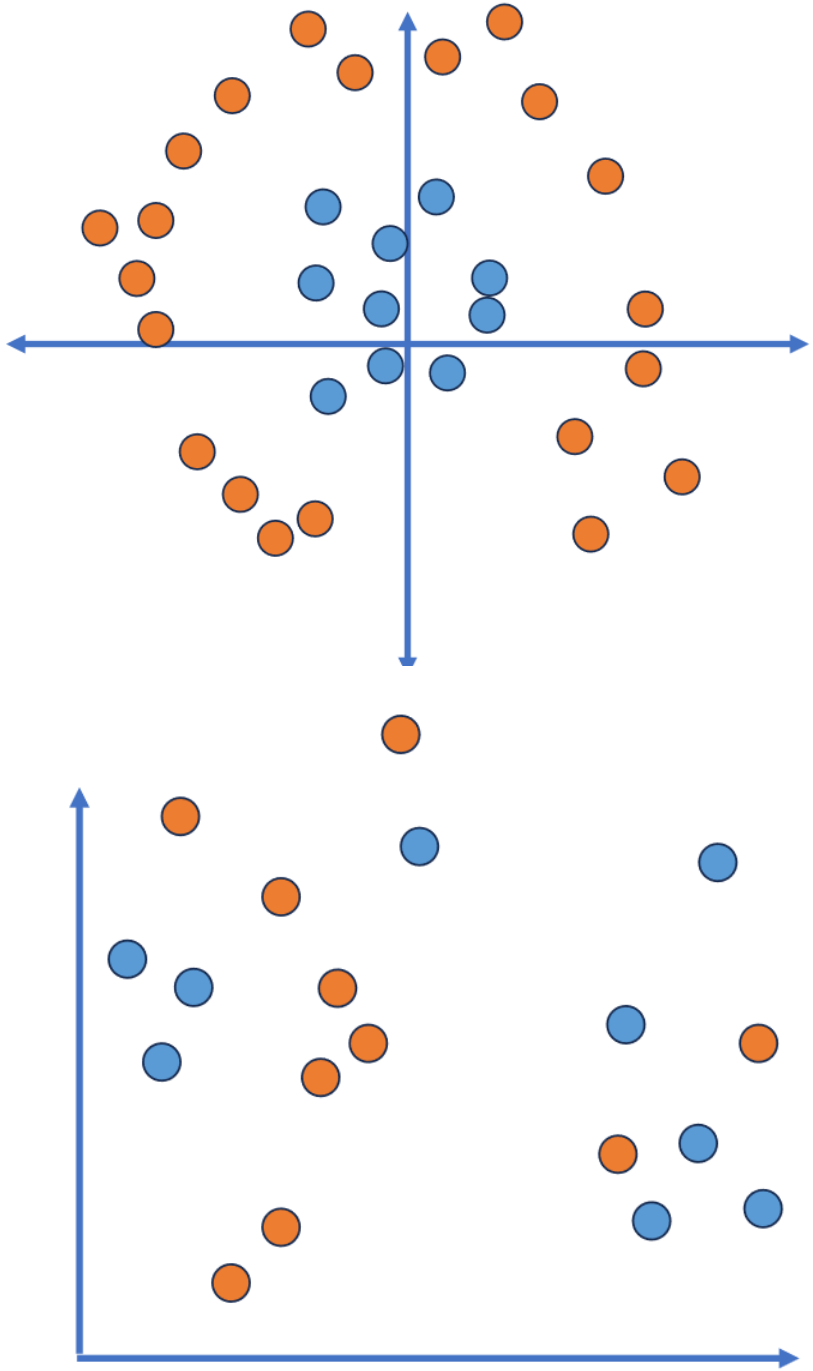
Example

What would be a good model for this dataset?



Challenges

- Some data sets don't have natural linear…
- For some like the top-right it could be pre-processed into linear space
  - using polar coordinate system
- Some like bottom-left may have no reasonable re-mapping
- For shapes like the first, we could also make our function more complex by making the model non-linear (Neural networks often chosen here)
  - http://playground.tensorflow.org



Overfit/Underfit

Underfitting

In machine learning, **underfitting** occurs where the model is not able to accurately capture a pattern in the data.

This could be due to insufficient training, lack of a pattern to be found, poor choice of data encoding, or poor choice of model class.

More complex model classes allow for a wider (more expressive) range of models, which are able to capture more complex patterns in the data.

Overfitting

In machine learning, **overfitting** occurs when a model fits too closely to the training data, and as a result does not perform well on new data.

You can think of this as the model being too specialized.

Remember, our goal is to train a model that generalizes well to new data.

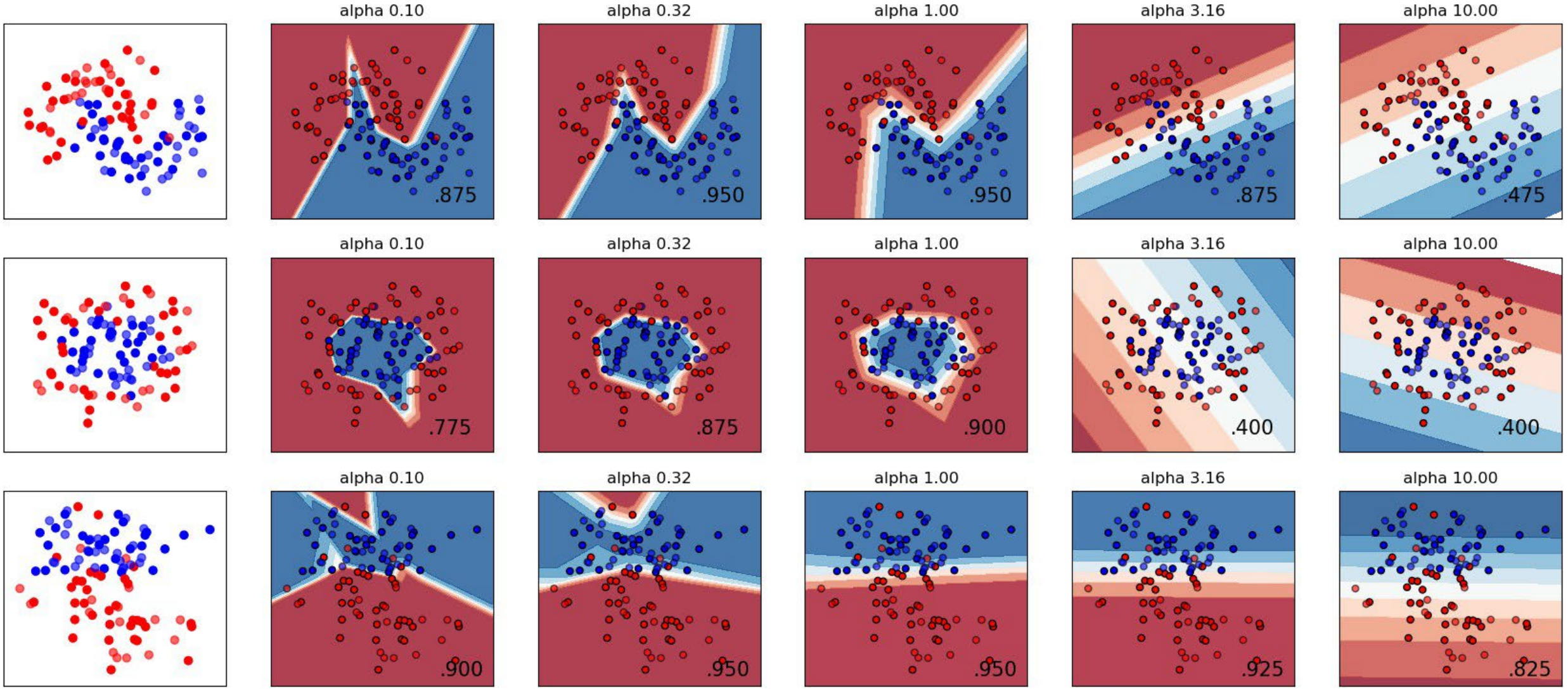
Neural networks often use drop-out for overfitting.
(Only use X% of your brain each time!)

Regularization

Recall that our goal so far is to find a model from a model class that minimizes the overall loss.

One way that overfitting can be handled is by adding a **regularization** parameter (α) to the optimization problem.

This adds pressure to the algorithm to keep the model less “extreme”, at the cost of some extra loss.



https://scikit-learn.org/stable/auto_examples/neural_networks/plot_mlp_alpha.html

Testing

Model testing

How do we detect whether our model is overfitting or not?

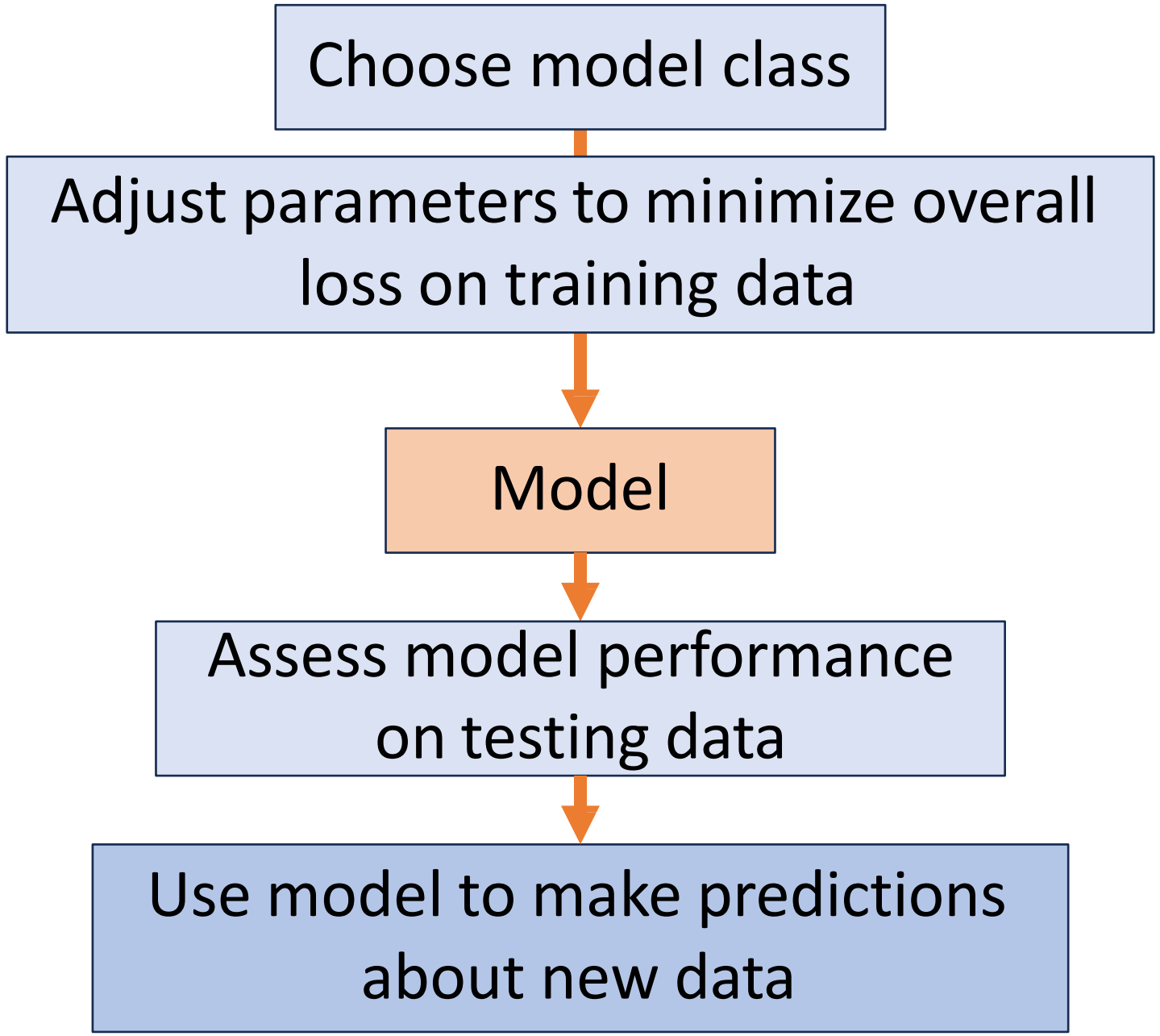
Remember, we can't "eyeball it" in 10,000 dimensions…

Model testing

When training machine learning models, we usually split our initial dataset into **training** and **testing** data.

We train the model on the training data, and then use the testing data to assess whether the trained model also performs well on new data.

- If the testing doesn't go well, we need to adjust our setup and try again


Choose model class
Adjust parameters to minimize overall loss on training data
Model
Assess model performance on testing data
Use model to make predictions about new data

Splitting data

Suppose you have a dataset of 10,000 labeled samples. How many should you use for training, and how many for testing?

- Ratio is usually 6:1

It is very important that the training and testing data come from the same underlying distribution.

- Make sure you shuffle your data first!

Next…neural networks

Jonathan Hudson, Ph.D.
jwhudson@ucalgary.ca
https://cspages.ucalgary.ca/~jwhudson/

UNIVERSITY OF CALGARY