ECE 57000

Basics of (supervised) Machine Learning

Chaoyue Liu

Fall 2024

Overview

First half of the semester: Supervised Learning

- Basics of machine Learning
- Linear model, linear regression, logistic regression
- Gradient descent, SGD
- Other non-neural networks models: kernel model, SVM (time-permitting)
- Neural network (fully-connected)
- Computer vision & Convolutional NN (CNN)
- Natural Language Processing (NLP) & Attention models

Second half of the semester: Unsupervised Learning

Supervised Machine Learning

- Data
  - Dataset: a set of input-label pairs. “Supervised” because of existence of labels.
  - Example: ImageNet -- more than **1 million** pairs of (image, label).

“goldfish”

terminology	input	label
notation	$\mathbf{x}$	y

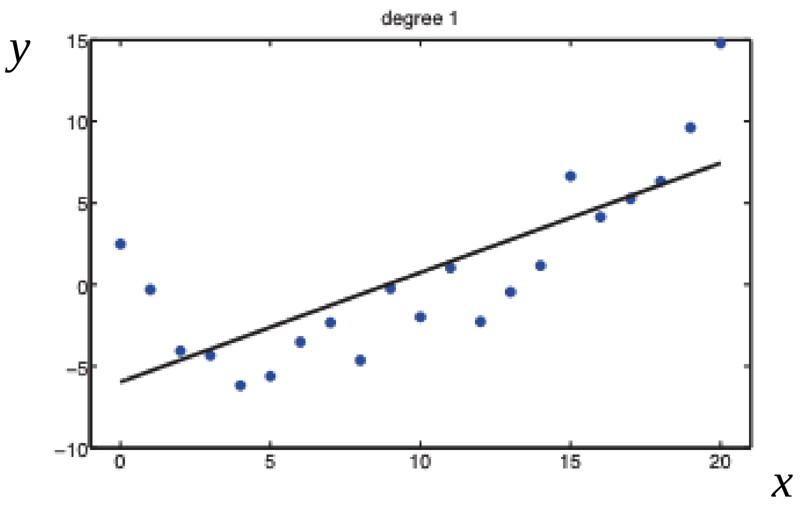
An input can be: a vector (a list of numbers), an image, a paragraph of text, …

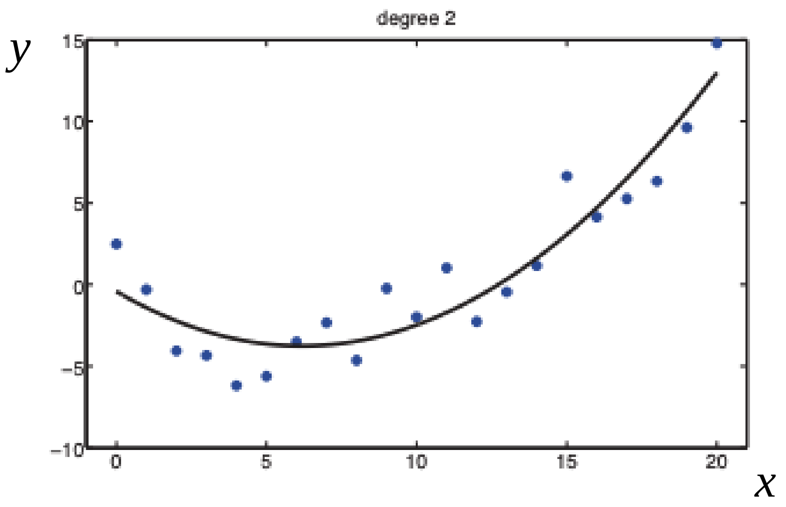
A label can be: real-valued (a number), categorical (an element of a finite set)

Regression

If the label y is <u>numeric</u> (a real valued number), then the problem is known as <u>regression</u>.

- Example 1: given inputs (area, age, # of bedrooms, location, …), predict/estimate the house's price.
- Example 2:





NOTE: Input x does not have to be numeric. Only the label y must be numeric.

Classification

If output is categorical,
then the problem is known as classification.

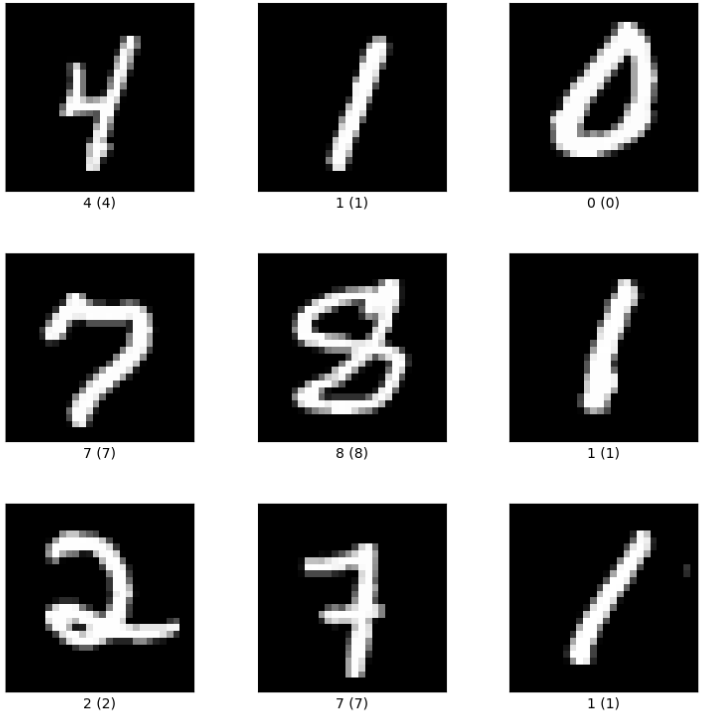
- Example 1: given height x, predict "male" (y=0) or "female" (y=1)
- Example 2: given salary x_1 and mortgage payment x_2, predict defaulting on loan ("yes" or "no")
- Example 3: given images, predict "cat" or "dog"

Question

MNIST:
- inputs: images with hand-written digits
- labels: digits from 0 to 9

Q: regression or classification?

A: classification.
- Predicting one digit: one element of a finite set
  - e.g., “5.5” makes no sense
- No ordering among digits
  - “4” is not closer to “5” than “1”



Supervised Learning

- Data
- Model

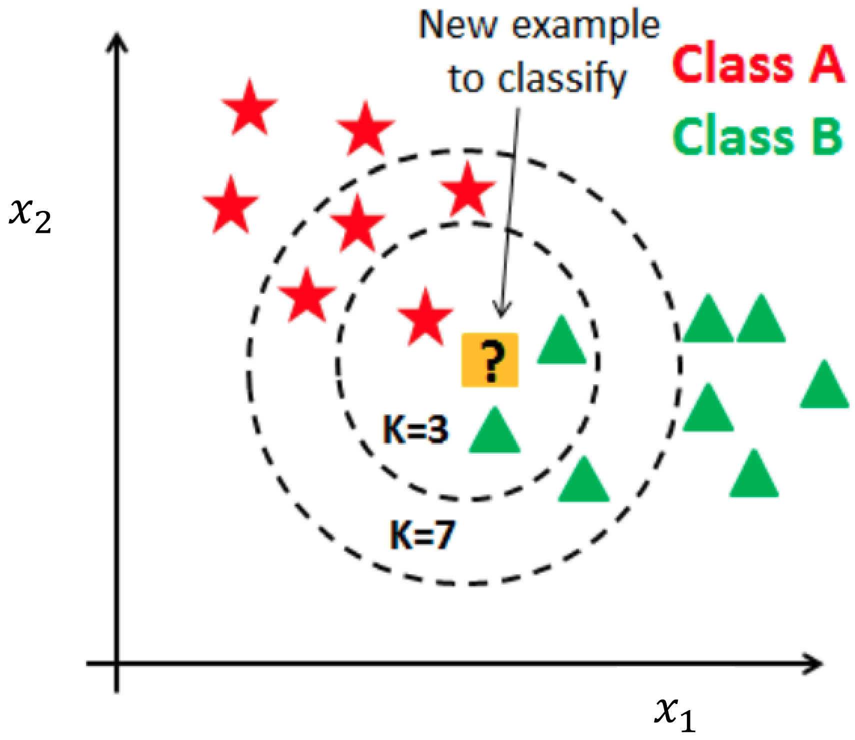
Build the model	**Train** the model	**Test/evaluate** the model
"machine"	"learning"	
Examples: Neural network, Kernel model, SVM, Linear model …	Use **training data** and **algorithm** to update the model	Use **test/dev data** to evaluate the learned model(s)

Now, let's use a simple model to demonstrate this procedure

k-NN: k-Nearest Neighbors

- a very simple and intuitive supervised learning model
- Intuition: data with similar inputs tend to have similar or the same labels

1. Choose k, k=1,3,5… (for binary)
2. Find the k nearest neighbors in the training dataset.
3. Select most common class as the predicted label.



k-NN

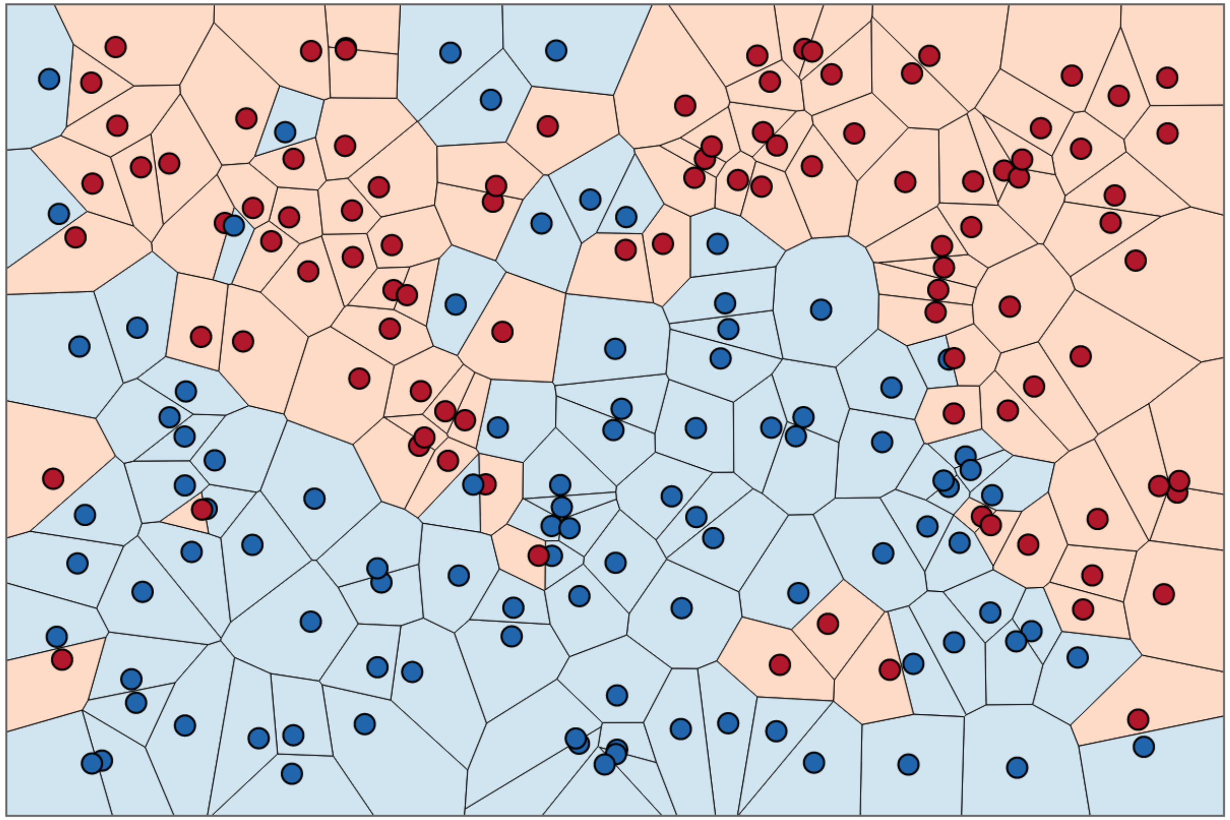
Input: Test point x_0, training data $D = \{(x_i, y_i)\}_{i=1}^{n}$

Output: Predicted class y_0

1. Compute distance to all training points:
$$d_i = distance(x_0, x_i), \forall i$$
2. Sort distances where π is a permutation:
(e.g., $\pi(1)$ is the index of the closest point)
$$d_{\pi(1)} \leq d_{\pi(2)} \leq d_{\pi(3)} \leq \cdots \leq d_{\pi(n)}$$
3. Return the most common class of the top k (vote)
$$y_0 = mode\left(\left\{y_{\pi(1)}, y_{\pi(2)}, y_{\pi(3)}, \cdots, y_{\pi(k)}\right\}\right)$$

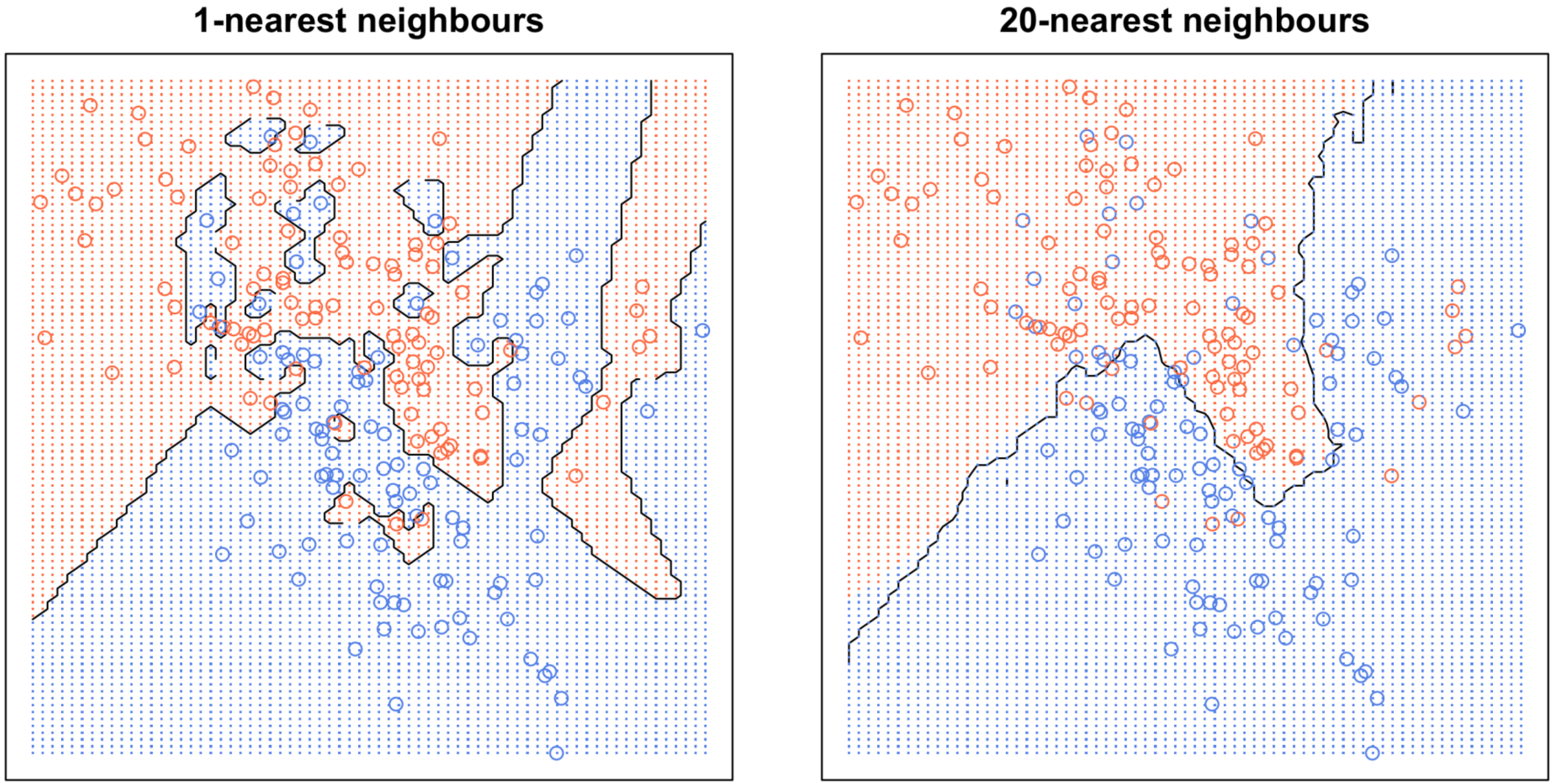
1-NN (k=1)

1-NN partitions the space into Voronoi cells based on the training data



http://scott.fortmann-roe.com/docs/BiasVariance.html

The k-NN boundary gets "smoother" as k increases



https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/

Now, let's go back to the procedure

	Build the model	**Train** the model	**Test/evaluate** the model
k-NN	Choose the *k*-NN model and Set *k*	no need to train for *k*-NN (a key part for other models)	We talk about this next…

Test/evaluate the model

Goal: make sure the model works well on **unseen** data, so that we can safely deploy the model

Q: can we use the same training data to test the model?

- If we train and evaluate on the same dataset, the model may only **memorize the training data** and **not generalize** well.
- Analogy to class
    - Training dataset is like homework, sample problems, and sample exams
    - Test dataset is like the real exam

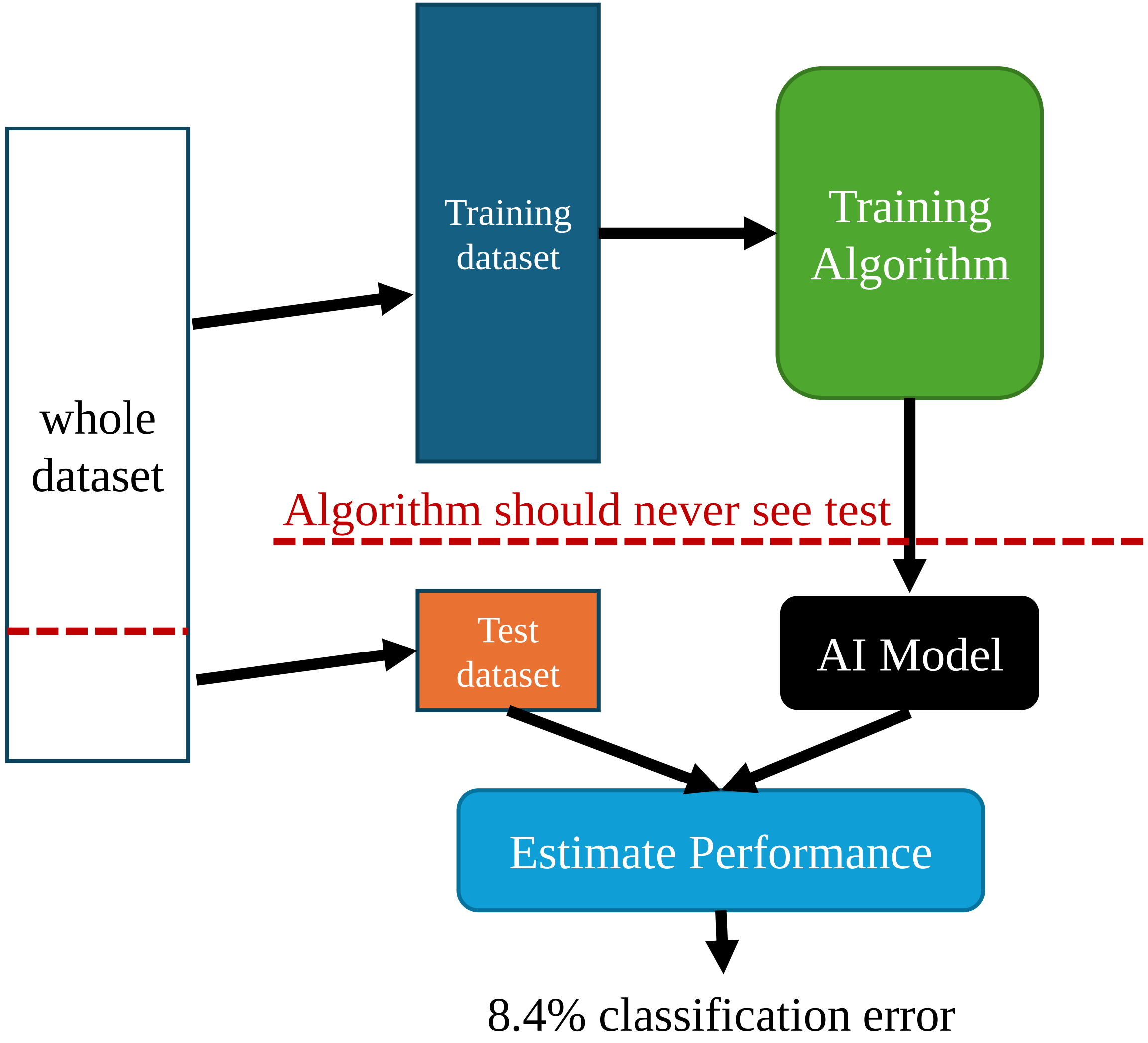
But, we don’t have the unseen data…

- estimate by splitting the known dataset

- Split the dataset

1. The <u>training dataset</u> is used to train the model only

2. The <u>test dataset</u> (or <u>held-out dataset</u>) is used to estimate generalization error.



k-NN: test performance

Input: Test dataset $D_{test} = \{(x_i, y_i)\}_{i=1}^{m}$, training dataset D_{train}

Output: generalization accuracy a

correct = 0

for each (x, y) in D_{test}:

  prediction $\tilde{y} = kNN(x, D_{train})$

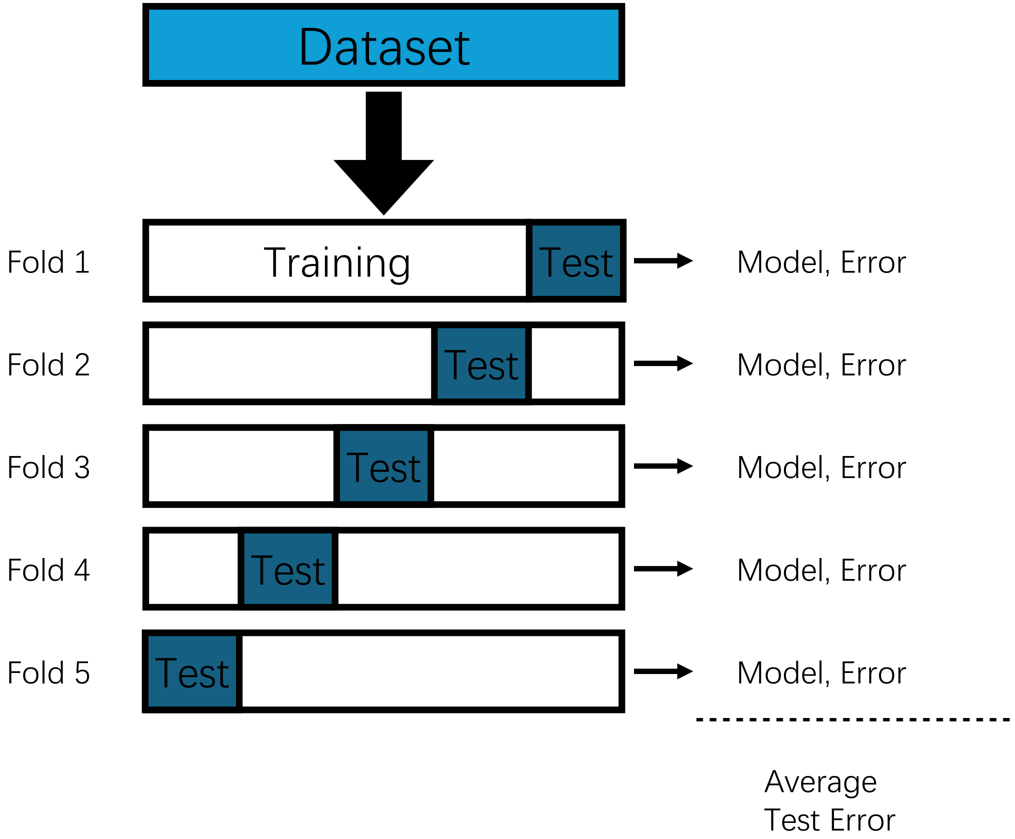
  if $\tilde{y}=y$, then correct = correct +1

generalization accuracy a = correct/m

generalization error e = 1-a.

Cross-validation (CV) generalizes the simple train/test split to M disjoint splits (effectively reusing data)

- Repeat the split process M times
  - Fit new model on train
  - Evaluate model on test
- Note: M models are fitted throughout process
- Final error estimate is average over all folds

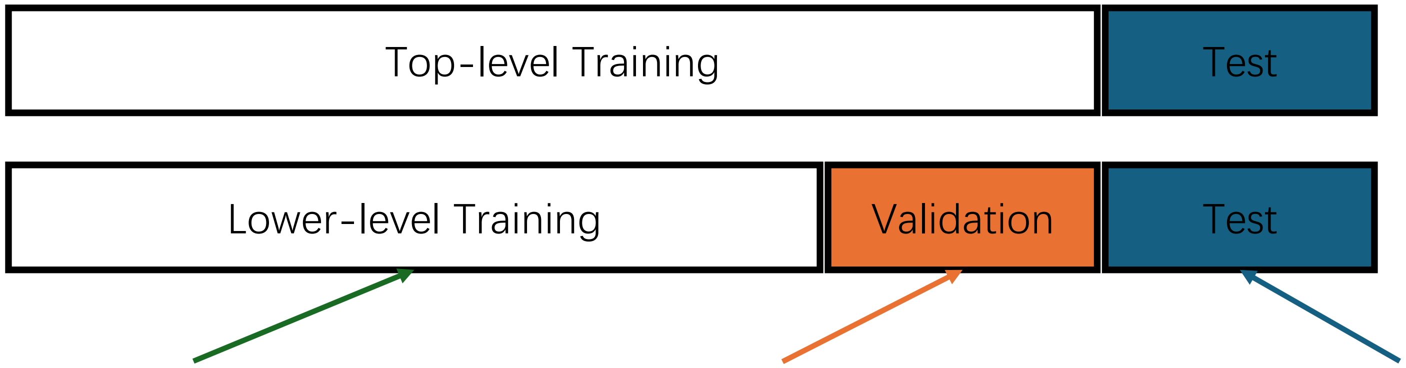


$M = 3, M = 5, M = 10$ are common

Cross-validation (CV) generalizes the simple train/test split to M disjoint splits

- Cross-validation is suitable for small dataset and small models
  - effectively reusing data, beneficial when dataset is small
  - requires running of M models, which is computationally expensive when model is large (e.g., deep neural networks).

But what if we want to select a model AND estimate the model's performance?



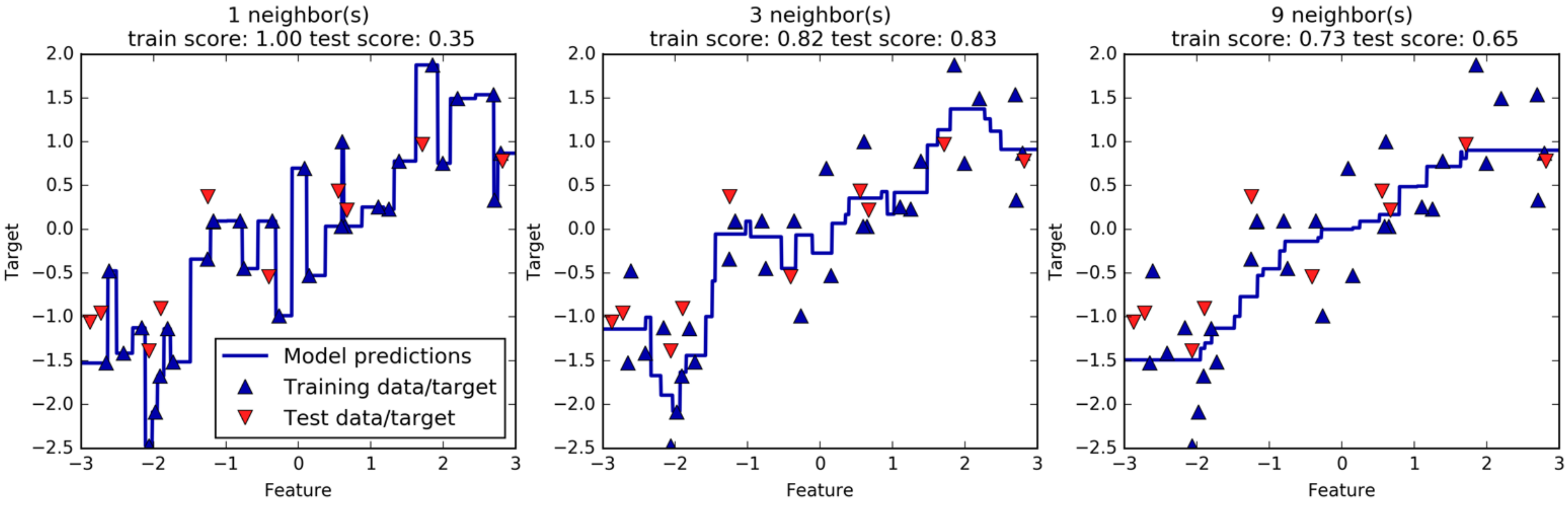
Used for training model during model selection

Used for selecting model (e.g., hyperparameter selection -- *k* for *k*-NN)

Used for estimating performance

k-NN can be used for regression problems (predict continuous values)

1. Find k nearest neighbors
2. Predict average of k nearest neighbors
   (possibly weighted by distance)

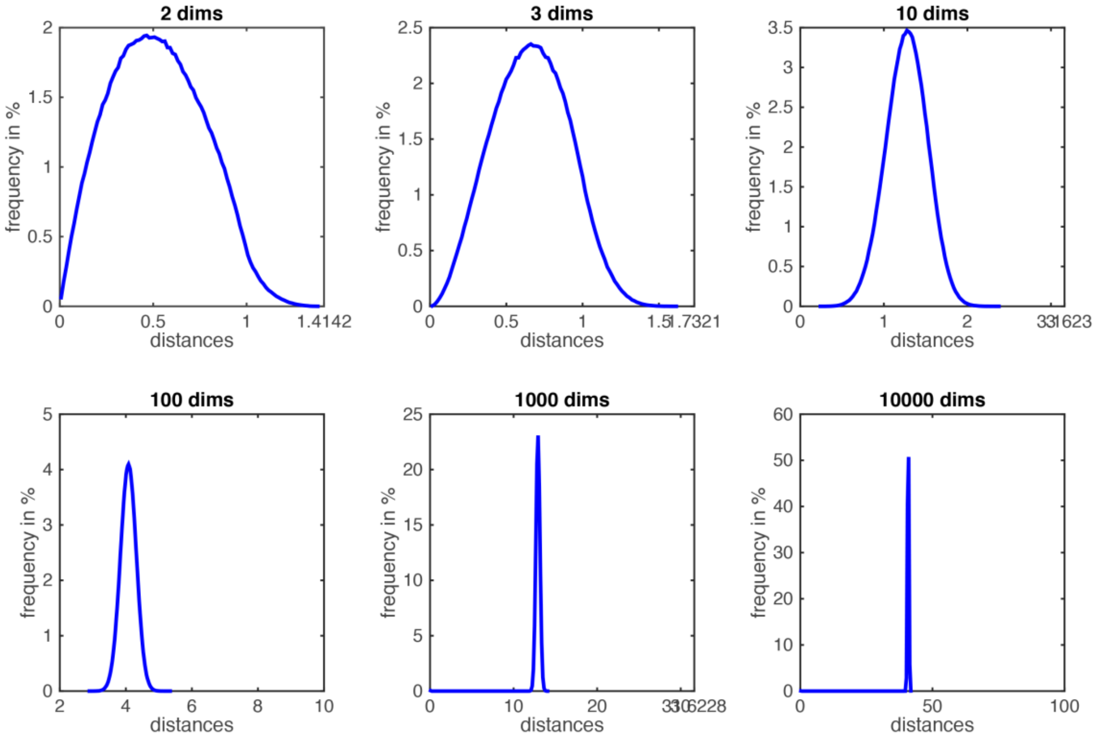


https://medium.com/analytics-vidhya/k-neighbors-regression-analysis-in-python-61532d56d8e4

Curse of dimensionality:

The performance of *k*-NN degrade significantly in high dimensions.

- The distances between <u>**any two points**</u> in high dimensions is nearly the same



Distance between two **random points** concentrate around a single value

Related reading and source for KNN curse of dimensionality illustrations

- https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote02_kNN.html