# Model Evaluation and Hyperparameter Tuning

Dr. Sharu Theresa Jose

University of Birmingham

February 9, 2024

# Learning Outcomes

- Understand the significance of model evaluation
- Learn different methods of model evaluation
- Understand the pros and cons of the methods

(Slides adopted from Prof. Ata Kaban's lectures)

# Recap: Supervised Learning

Main ingredients of supervised learning:

- **Model**: Form of function that we want to learn characterized by free parameters
- **Cost function**: Measures the misfit of any particular function from the model, given a training set
- **Training algorithm**: For example, gradient descent that minimizes the cost function

Running the training algorithm on some training data learns the "best" values of the free parameters, yielding a **predictor**.

# Recap: Continued

**Hyperparameters** are "higher-level" free parameters.

- In neural networks:
  - Depth (number of hidden layers)
  - Width (number of hidden neurons in a hidden layer)
  - Activation function (choice of nonlinearity in non-input nodes)
  - Regularisation parameter (way to trade off simplicity vs. fit to the data)
- In polynomial regression:
  - Order of the polynomial (i.e., use of $x$, $x^2$, $x^3$, ..., $x^m$)
- In general
  - Model Choice

# Evaluation of a Predictor Before Deployment

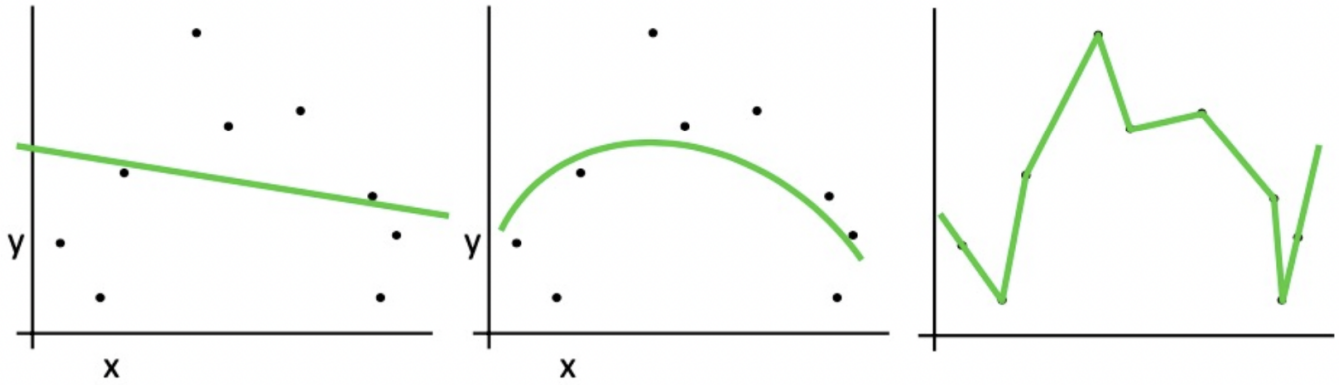
- Recall: A predictor is obtained by training the free parameters of the considered model using the available annotated data.
- Evaluation of a predictor serves to estimate its future performance, before deploying it in the real world.

- To do so, we always split the available annotated data randomly into:
  - A **training set**, used to estimate the free parameters, and
  - A **test set**, used to evaluate the performance of the trained predictor before deploying it.

# Which Model? How to set hyperparameters?

- Each hyperparameter value corresponds to a different model.
- We need methods that evaluate each model.
- For this evaluation, we can no longer use the cost function computed on the training set. Why?
  - The more complex (flexbile) the model, the better it will fit the training data.
  - But the goal is to predict well on future data.
  - A model that has capacity to fit any training data will overfit.

# Which Model to Choose?




Fit with Model 1 Fit with Model 2 Fit with Model 3

Note here:

- Even if the models only differ by one hyperparameter, they are different models.
- Choosing a particular value of a hyperparameter requires evaluating each model.

# Evaluating Models for Model Choice

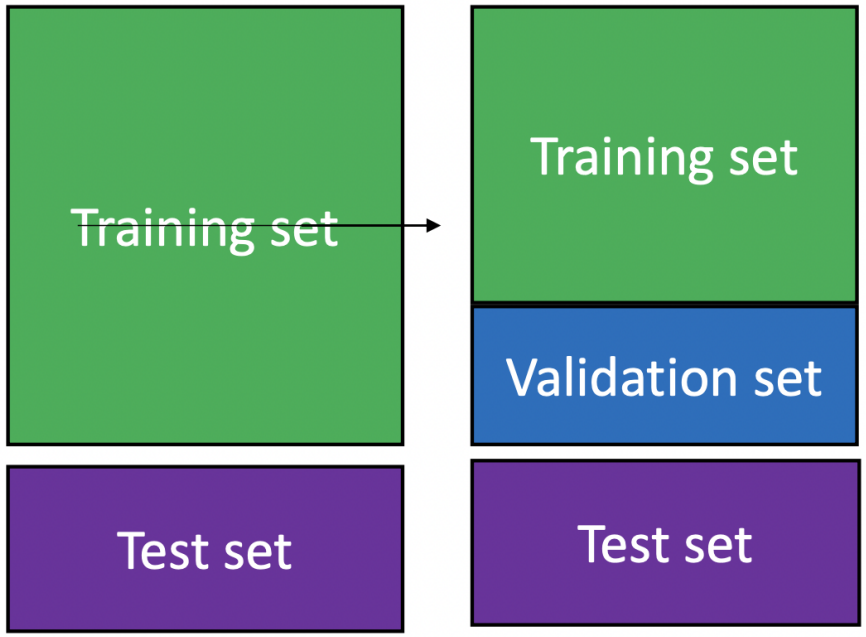
- Do not confuse this with evaluating a predictor (model already chosen).
- Training set is annotated data used for **training within a chosen model**.
- Test set is also annotated data but used for **evaluating the trained predictor before deploying it**.
- None of these can be used to choose the model!
  - Tempting to use the test set, but if we do, we no longer have an independent data set to evaluate the final predictor before deployment.

# Evaluating Models for Model Choice

Key Idea: To choose between models or hyperparameters, separate a subset from the training set to create **validation set**.

**Methods**

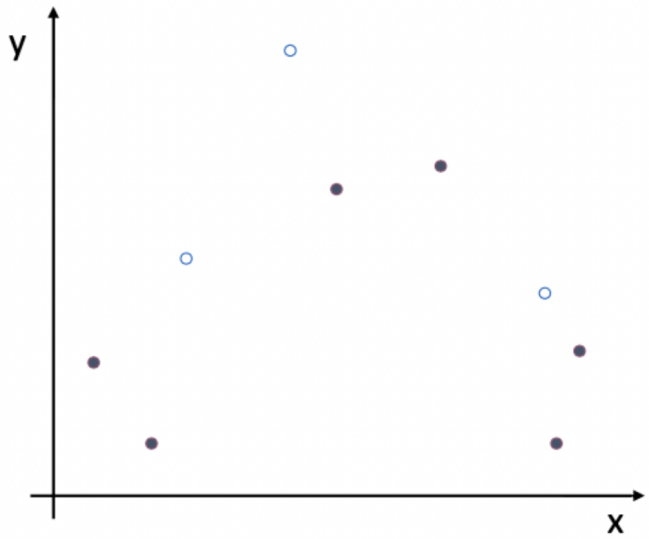
- Holdout validation
- Cross-validation
- Leave-one-out validation



# Method 1: Holdout Validation

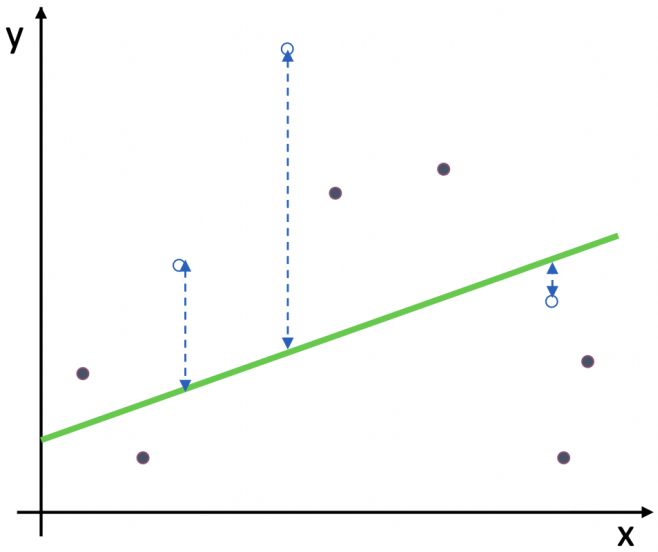
**Steps:**

1. Randomly choose 30% of data to form a validation set (blue data points)
2. Remaining data forms the training set (black data points)



**Steps:**

1. Randomly choose 30% of data to form a validation set
2. Remaining data forms the training set
3. Train *your model* on the training set
4. Estimate the test performance on the validation set

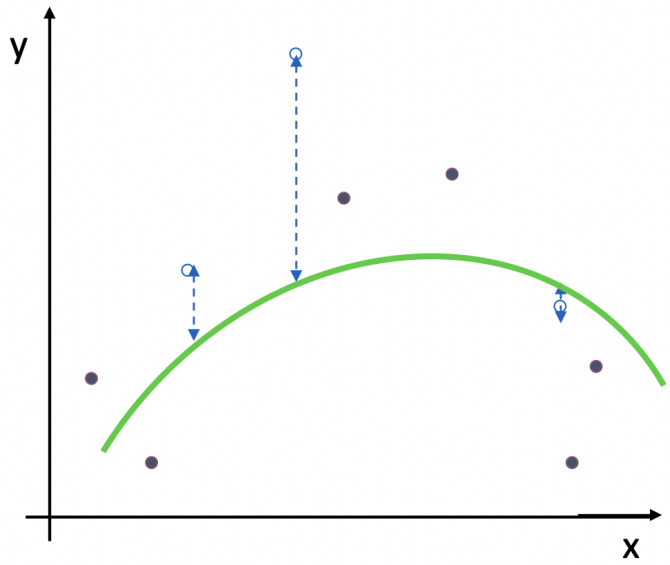



Model 1

Mean Squared Validation Error = 2.4

**Steps:**

1. Randomly choose 30% of data to form a validation set
2. Remaining data forms the training set
3. Train *your model* on the training set
4. Estimate the test performance on the validation set

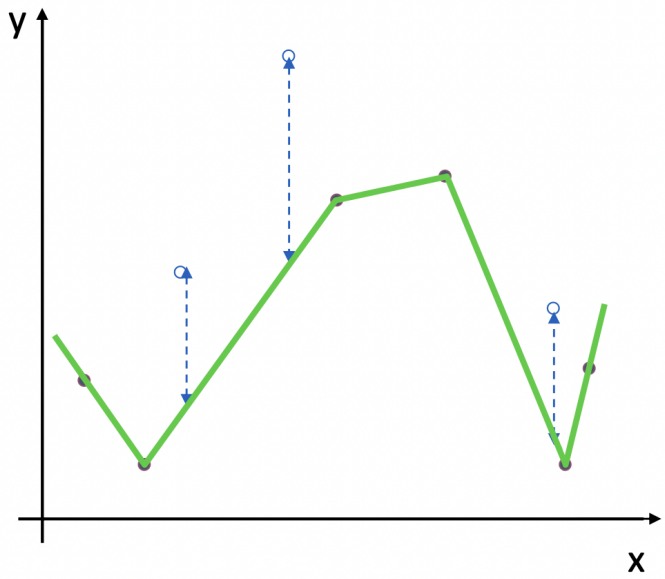



Model 2

Mean Squared Validation Error = 0.9

**Steps:**

1. Randomly choose 30% of data to form a validation set
2. Remaining data forms the training set
3. Train *your model* on the training set
4. Estimate the test performance on the validation set




Model 3

Mean Squared Validation Error = 2.2

**Steps:**

1. Randomly choose 30% of data to form a validation set
2. Remaining data forms the training set
3. Train *your model* on the training set
4. Estimate the test performance on the validation set
5. Choose the model with lowest validation error

- Model 1: Mean squared valiation error $= 2.4$
- Model 2: Mean squared valiation error $= 0.9$
- Model 3: Mean squared valiation error $= 2.2$

Lowest validation error: **Model 2**

**Steps:**

1. Randomly choose 30% of data to form a validation set
2. Remaining data forms the training set
3. Train *your model* on the training set
4. Estimate the test performance on the validation set
5. Choose the model with lowest validation error
6. Re-train with chosen model on joined training and validation to obtain predictor
7. Estimate future performance of the obtained predictor on test set
8. Ready to deploy the predictor

# A Final Note on Point 4

4: Estimate the test performance on the validation set

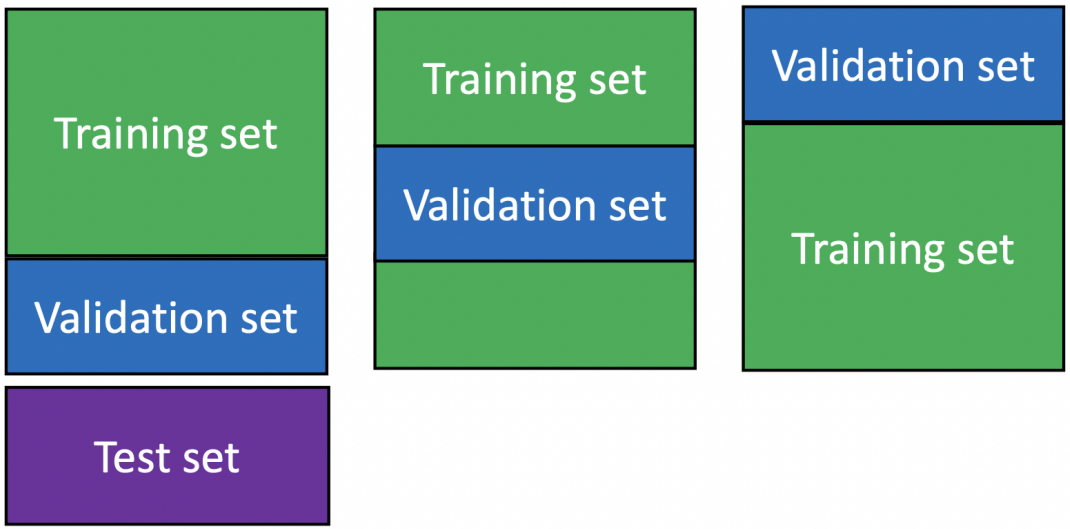
In practice, this is done differently in regression and in classification:

- Regression: We compute the cost function (mean squared error) on the examples of the validation set (instead of the training set)
- Classification: We do not compute the cross-entropy cost on the validation set. Instead, we compute the $0-1$ error metric:

$$0-1 \text{ error metric} = \frac{\text{number of wrong predictions}}{\text{number of predictions}} = 1 - \text{Accuracy}.$$

  - Other metrics besides Accuracy can also be employed.

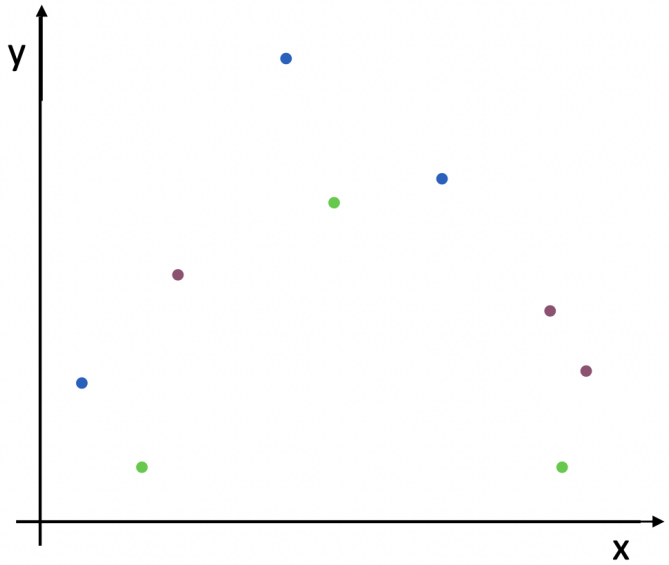
# Method 2: $k$-Fold Cross-Validation



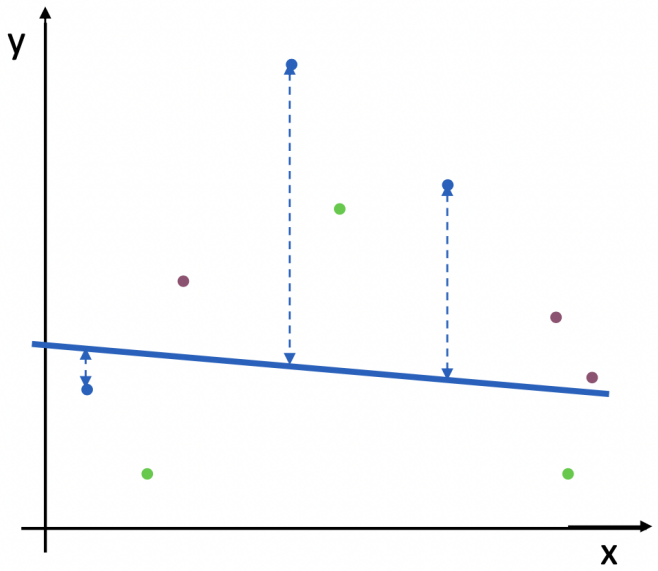

1. Split the training set randomly into $k$ (equal-sized) disjoint sets. (In this example, $k = 3$)
2. Use $k - 1$ of those together for training
3. Use the remaining one for validation
4. Permute the $k$ sets and repeat $k$ times
5. Average the performances on $k$ validation sets

# Explanation of the Steps

Randomly split the dataset into $k = 3$ partitions, denoted by blue, green and purple points.

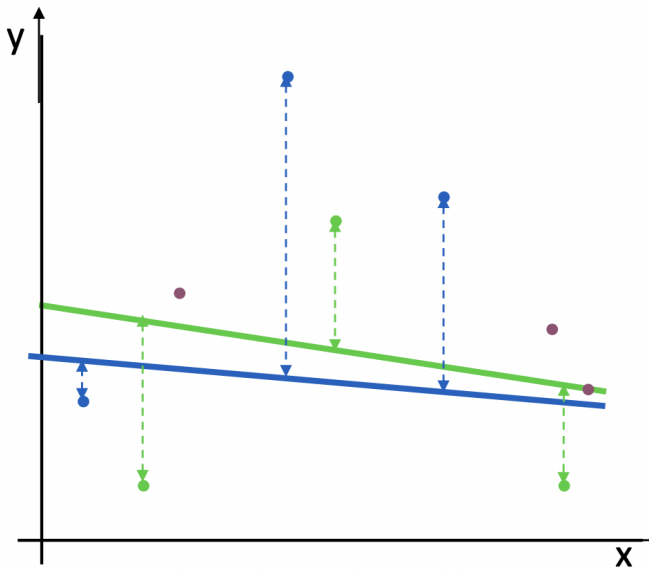


Model 1



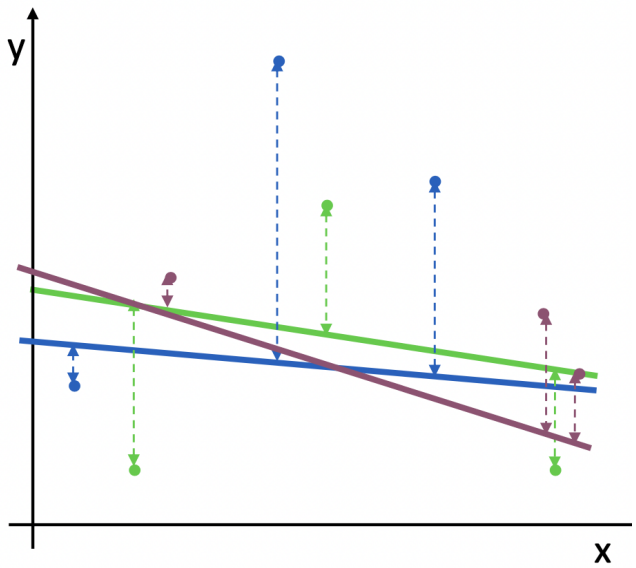

- For the blue partition: Train on all the points **except** the blue partition. Compute the validation error using the points in the blue partition

Model 1



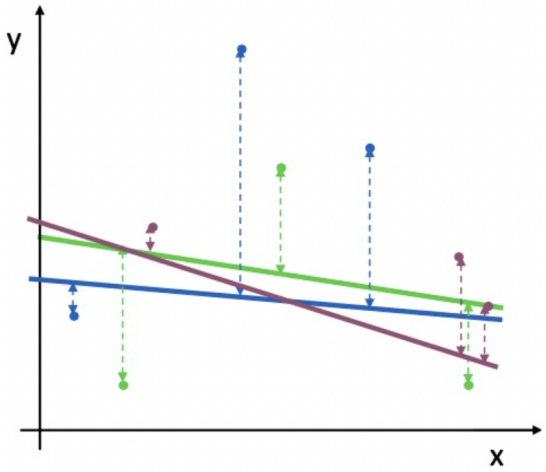

- For the blue partition: Train on all the points **except** the blue partition. Compute the validation error using the points in the blue partition
- For the green partition: Train on all the points **except** the green partition. Compute the validation error using the points in the green partition

Model 1

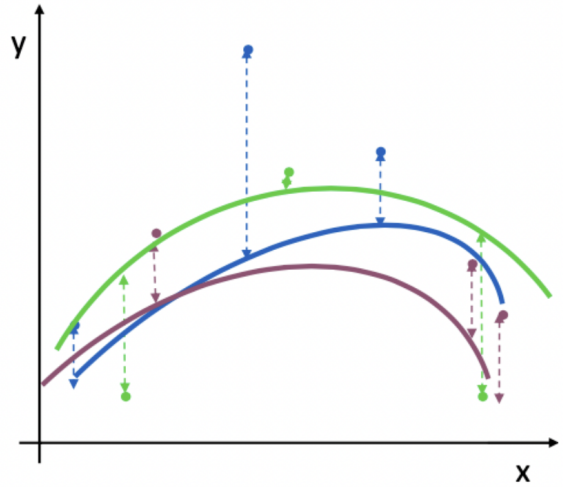



- For the blue partition: Train on all the points **except** the blue partition. Compute the validation error using the points in the blue partition
- For the green partition: Train on all the points **except** the green partition. Compute the validation error using the points in the green partition
- For the purple partition: Train on all the points **except** the purple partition. Compute the validation error using the points in the purple partition
- Take the mean of these errors
  - $\mathrm{MSE}_{\mathrm{3FOLD}} = 2.05$

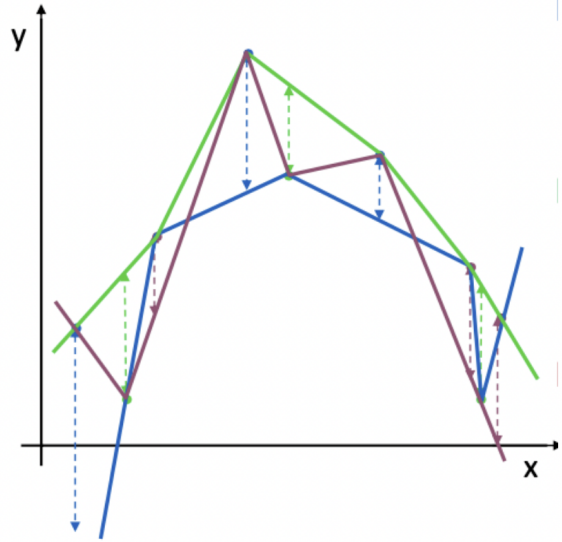
Repeat for other models:




Model 1
$MSE_{3FOLD}$=2.05




Model 2
$MSE_{3FOLD}$=1.11




Model 3
$MSE_{3FOLD}$=2.93

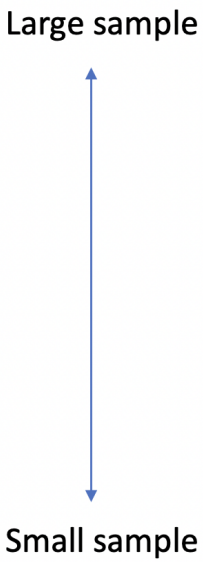
- Choose the model with the smallest average 3-fold cross validation error. Here, this is Model 2.
- Re-train with chosen model on joined training and validation to obtain the predictor
- Estimate future performance of the obtained predictor on test set
- Deploy the predictor in real-world

# Method 3: Leave-one-out Validation

- We leave out a single example for validation, and train on all the rest of the annotated data
- For a total of $N$ examples, we repeat this $N$ times, each time leaving out a single example
- Take the average of the validation errors as measured on the left-out points
- Same as $N$-fold cross-validation where $N$ is the number of labelled points

# Advantages and Disadvantages

| | Advantages | Disadvantages |
|---|---|---|
| **Holdout validation** | Computationally cheapest | Most unreliable if sample size is not large enough |
| **3-fold** | Slightly more reliable than holdout | • Wastes 1/3-rd annotated data.<br>• Computationally 3-times as expensive as holdout |
| **10-fold** | • Only wastes 10%<br>• Fairly reliable | • Wastes 10% annotated data<br>• Computationally 10-times as expensive as holdout |
| **Leave-one-out** | Doesn't waste data | Computationally most expensive |



# Summary

- Need for model validation methods
- Three methods for model validation:
  - Holdout validation
  - $k$-fold cross-validation
  - Leave-one-out validation
- Advantages and disadvantages of each method