Instituto de Computação

UNIVERSIDADE ESTADUAL DE CAMPINAS

Capacitação profissional em tecnologias de Inteligência Artificial

Machine Learning Overview

Prof. Edson Borin

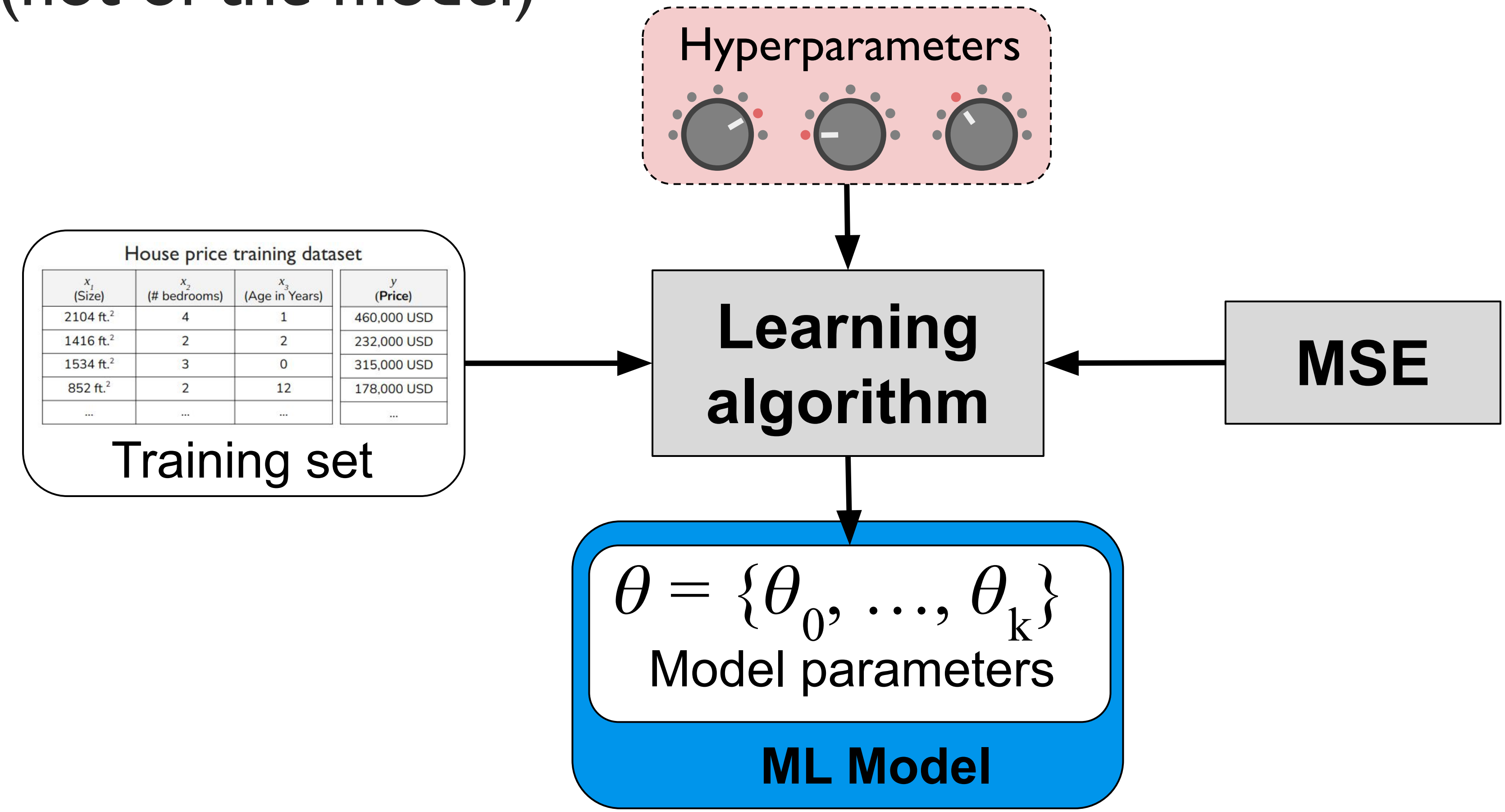
https://www.ic.unicamp.br/~edson

Institute of Computing - UNICAMP

Hyperparameters tuning

ML Process - Hyperparameters tuning

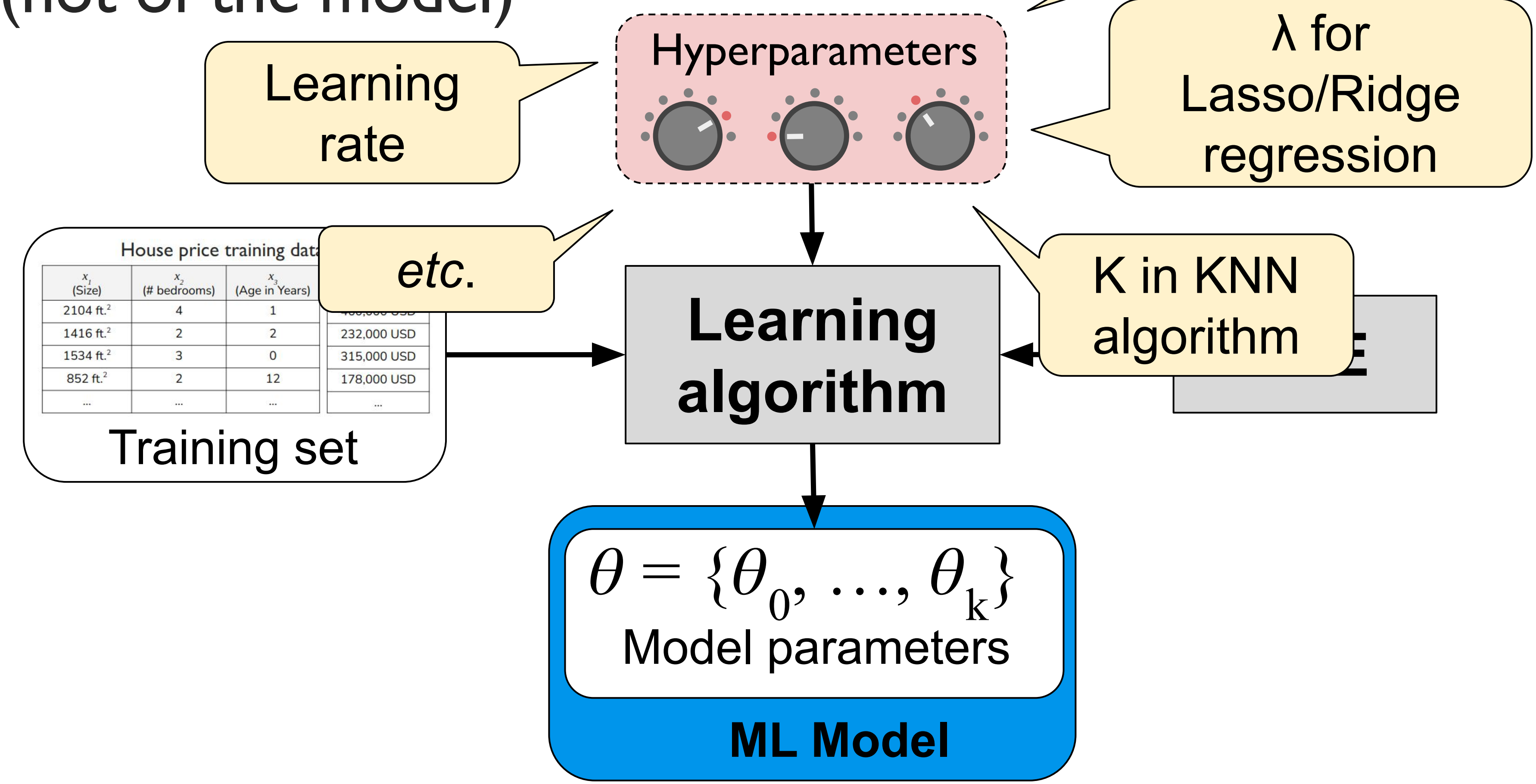
Hyperparameters: parameters of the learning process (not of the model)



x_1 (Size)	x_2 (# bedrooms)	x_3 (Age in Years)	y (Price)
2104 ft.^{2}	4	1	460,000 USD
1416 ft.^{2}	2	2	232,000 USD
1534 ft.^{2}	3	0	315,000 USD
852 ft.^{2}	2	12	178,000 USD
...	...	...	...

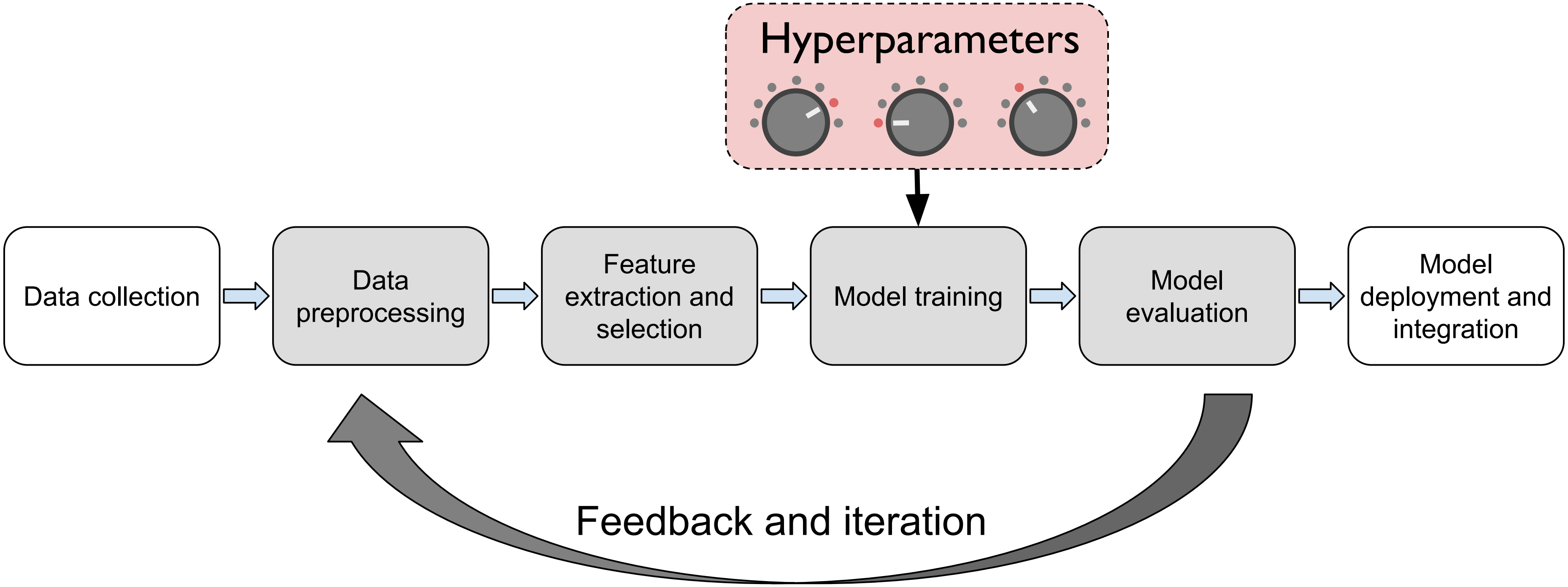
ML Process - Hyperparameters tuning

<u>Hyperparameters</u>: parameters of the le ess (not of the model)



ML Process - Hyperparameters tuning

<u>Hyperparameter tuning</u>: finding the best combination of hyperparameters that causes the learning process to produce the best model!



ML Process - Hyperparameters tuning

Hyperparameter tuning: finding the best combination of hyperparameters that causes the learning process to produce the best model!

- Example: scikit learn SVC models with RBF kernel
  - C: regularization parameter
  - γ: Kernel coefficient
  - Some hyperparameters combinations:
    - $(C, \gamma) \in \{ (10, 0.1), (10, 0.2), (100, 0.1), (100, 1.0)\}$

ML Process - Hyperparameters tuning

Search approach: strategy to evaluate the combinations of hyperparameters

- Several approaches
    - Grid search
    - Random search
    - Bayesian optimization
    - …

Search approach: **Grid Search**

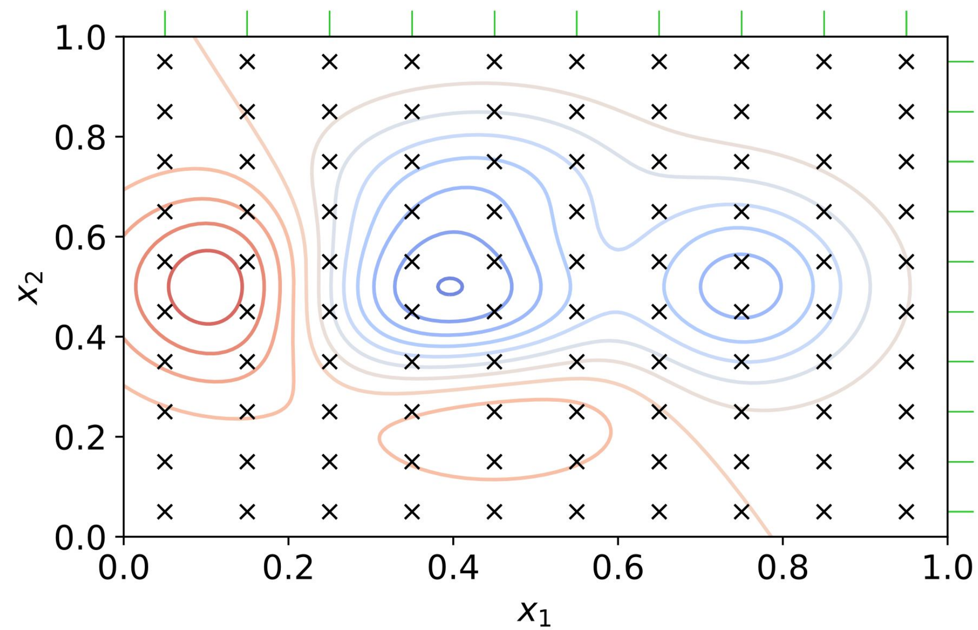
- Grid search (or parameter sweep) consists on a exhaustive search on a grid defined by the cartesian product of all parameters candidate values
- Example 1:
    - For $C \in \{10, 50, 100\}$, γ = {0.1, 0.2, 0.5, 1.0}, defined by the practitioner
    - $C \times \gamma$ = { (10, 0.1), (10, 0.2), (10, 0.5), (10, 1.0),
      (50, 0.1), (50, 0.2), (50, 0.5), (50, 1.0),
      (100, 0.1), (100, 0.2), (100, 0.5), (100, 1.0) }

ML Process - Hyperparameters tuning

Search approach: **Grid Search**

- Grid search (or parameter sweep) consists on a exhaustive search on a grid defined by the cartesian product of all parameters candidate values

- Example:
  - x_1 = np.arange(0.05, 1.0, 0.1)
  - x_2 = np.arange(0.05, 1.0, 0.1)

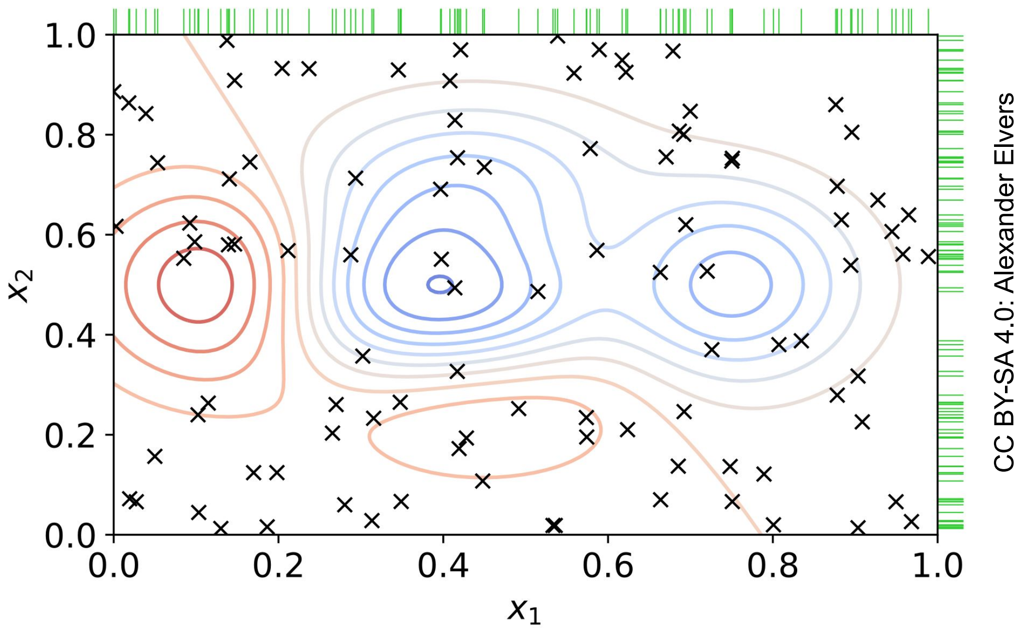


ML Process - Hyperparameters tuning

Search approach: **Random Search**

- Randomly selects values for hyperparameters
  - Bounds (max, min) values are defined by the user

- Example:
  - $x_1 \in = [0.0, 1.0]$
  - $x_2 \in = [0.0, 1.0]$

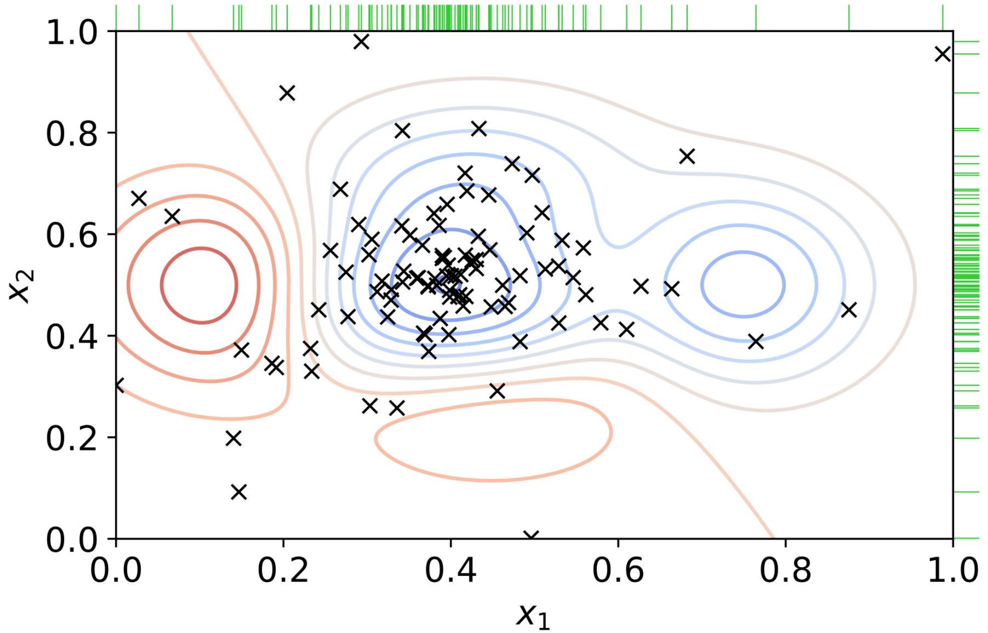


ML Process - Hyperparameters tuning

<u>Search approach</u>: **Bayesian optimization**

- Selects next set of hyperparameters to evaluate based on the performance of previous ones
  - Can be adjusted to favor exploring unknown regions or to focus on best regions found so far

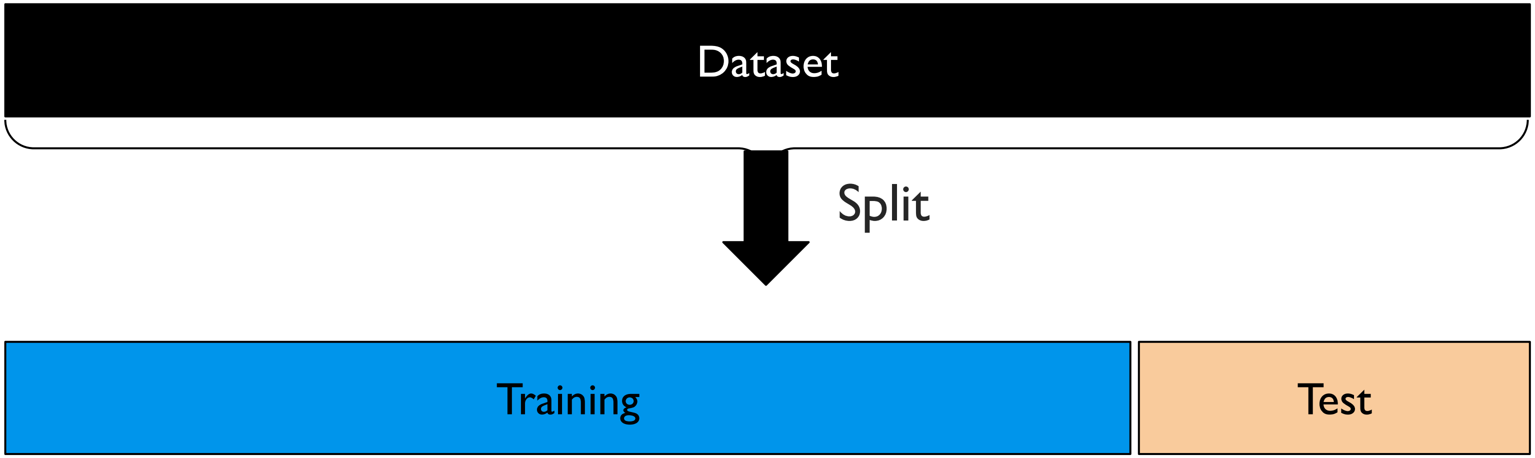
- Example:
  - $x_1 \in = [0.0, 1.0]$
  - $x_2 \in = [0.0, 1.0]$



Dataset splitting

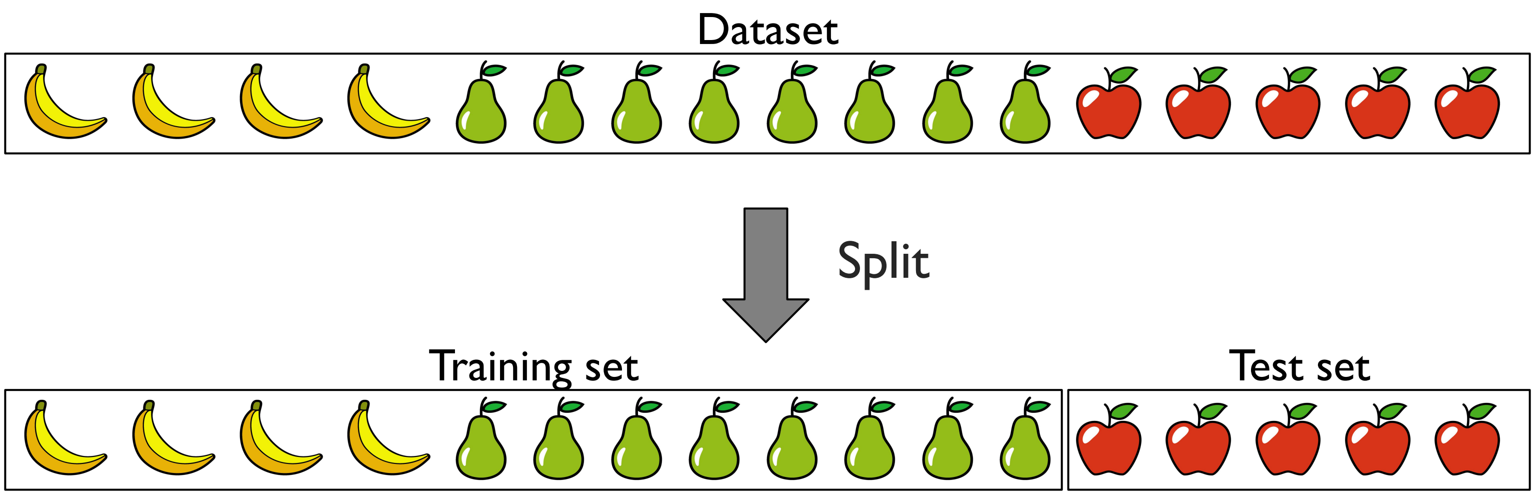
ML Process - Dataset splitting

- On supervised learning tasks, the dataset is usually split into two subsets: training and test
    - Training set: used to train the model (i.e., adjust θ)
    - Test set: check the model generalization
        - Represents new/unseen data



ML Process - Dataset splitting

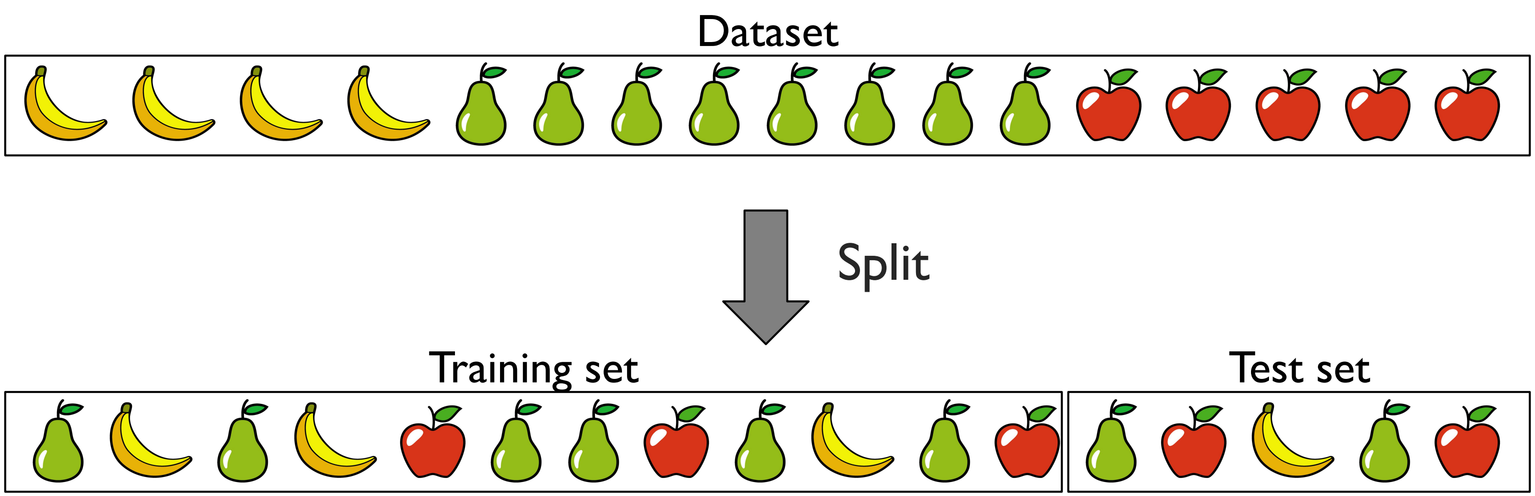
- On supervised learning tasks, the dataset is usually split into two subsets: training and test
  - Training set: used to train the model (i.e., adjust θ)
  - Test set: check the model generalization
    - Represents new/unseen data



WARNING: Bad split!

ML Process - Dataset splitting

- On supervised learning tasks, the dataset is usually split into two subsets: training and test
    - Training set: used to train the model (i.e., adjust θ)
    - Test set: check the model generalization
        - Represents new/unseen data



Random is better!

ML Process - Dataset splitting

- On supervised learning tasks, the dataset is usually split into two subsets: training and test
  - Training set: used to train the model (i.e., adjust θ)
  - Test set: check the model generalization
    - Represents new/unseen data

```
from sklearn.model_selection import train_test_split
X = ['y', 'y', 'y', 'r', 'r', 'r', 'r', 'g', 'g', 'g', 'g', 'g', 'g']
y = ['B', 'B', 'B', 'A', 'A', 'A', 'A', 'P', 'P', 'P', 'P', 'P', 'P']

# Split arrays or matrices into random train and test subsets.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.2)

print(X_train, X_test)
print(y_train, y_test)
```

```
['r', 'y', 'g', 'r', 'g', 'g', 'y', 'r', 'g', 'g'] ['r', 'y', 'g']
['A', 'B', 'P', 'A', 'P', 'P', 'B', 'A', 'P', 'P'] ['A', 'B', 'P']
```

ML Process - Dataset splitting

- On supervised learning tasks, the dataset is usually split into two subsets: training and test
  - Training set: used to train the model (i.e., adjust θ)
  - Test set: check the model generalization
    - Represents new/unseen data

```
from sklearn.model_selection import train_test_split
X = ['y', 'y', 'y', 'r', 'r', 'r', 'r', 'g', 'g', 'g', 'g', 'g', 'g']
y = ['B', 'B', 'B', 'A', 'A', 'A', 'A', 'P', 'P', 'P', 'P', 'P', 'P']

# Split arrays or matrices into random train and test subsets.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.2)

print(X_train, X_test)
print(y_train, y_test)
```

```
['r', 'y', 'g', 'r', 'g', 'g', 'y', 'r', 'g', 'g
['A', 'B', 'P', 'A', 'P', 'P', 'B', 'A', 'P', 'P
```

Must be chosen carefully!

ML Process - Dataset splitting

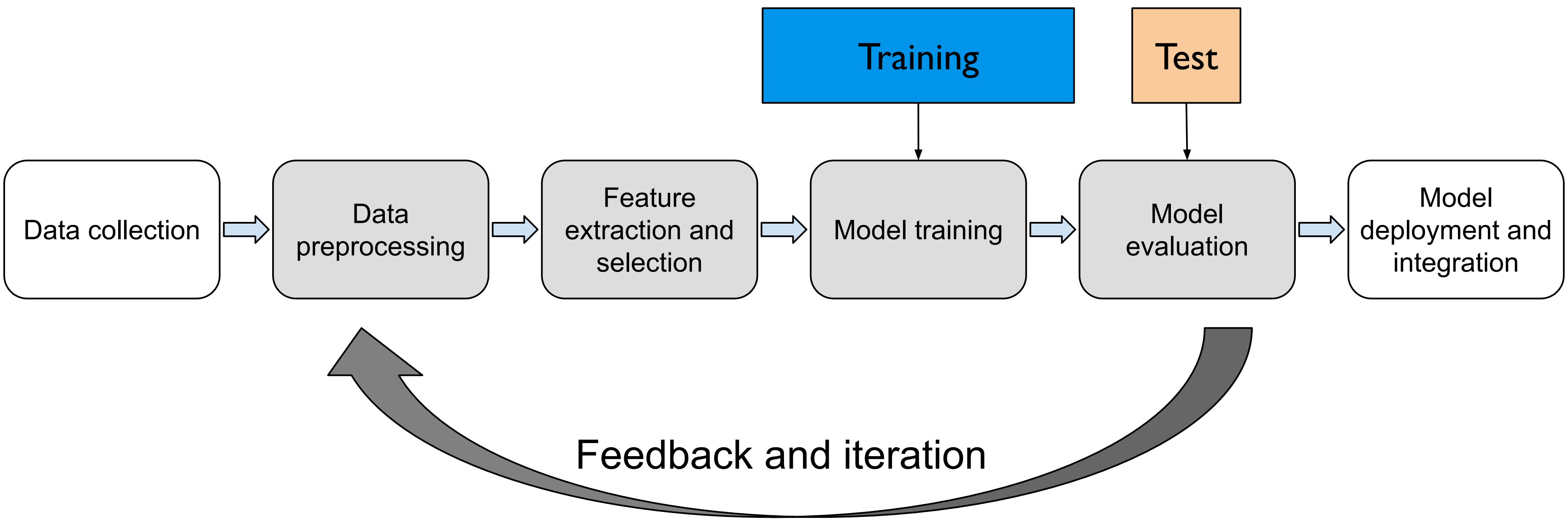
- On supervised learning tasks, the dataset is usually split into two subsets: training and test
  - Training set: used to train the model (i.e., adjust θ)
  - Test set: check the model generalization
    - Represents new/unseen data

Never train your model using the test data!

ML Process - Dataset splitting

<u>Test set and ML process iterations.</u>

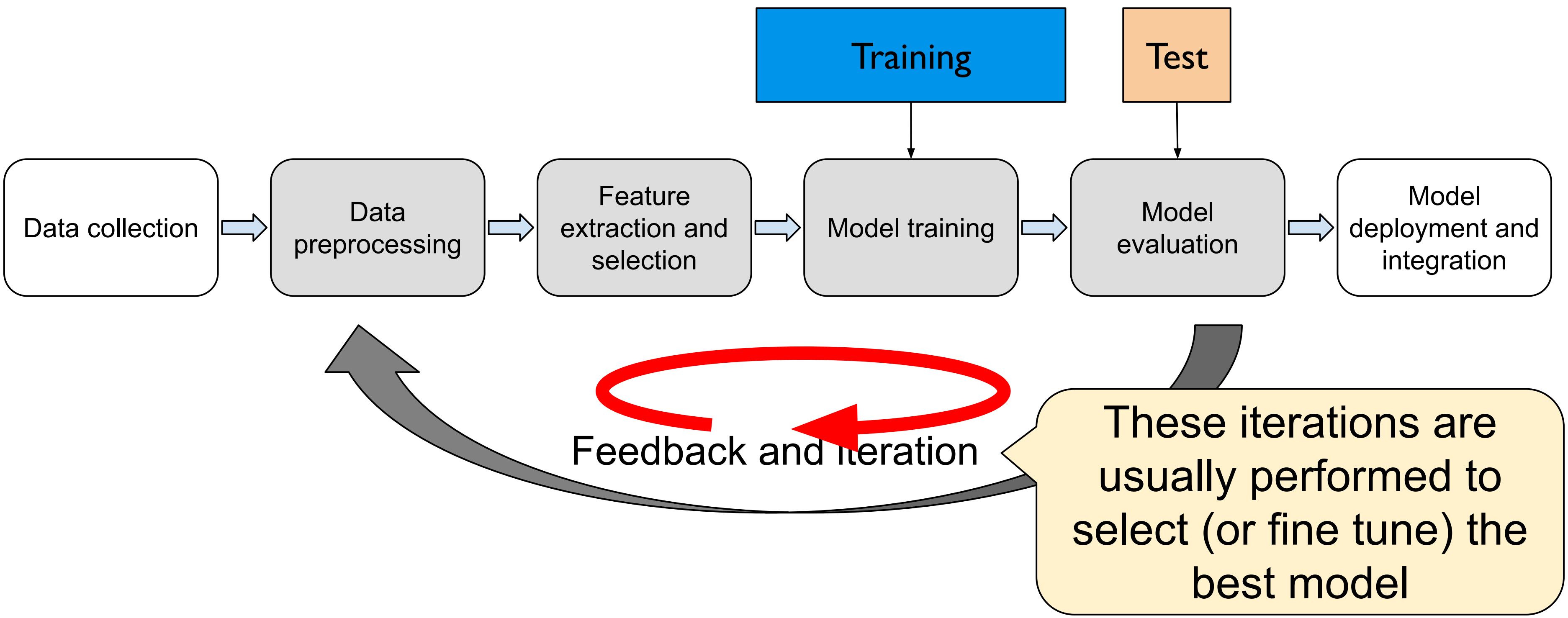
Previously…



ML Process - Dataset splitting

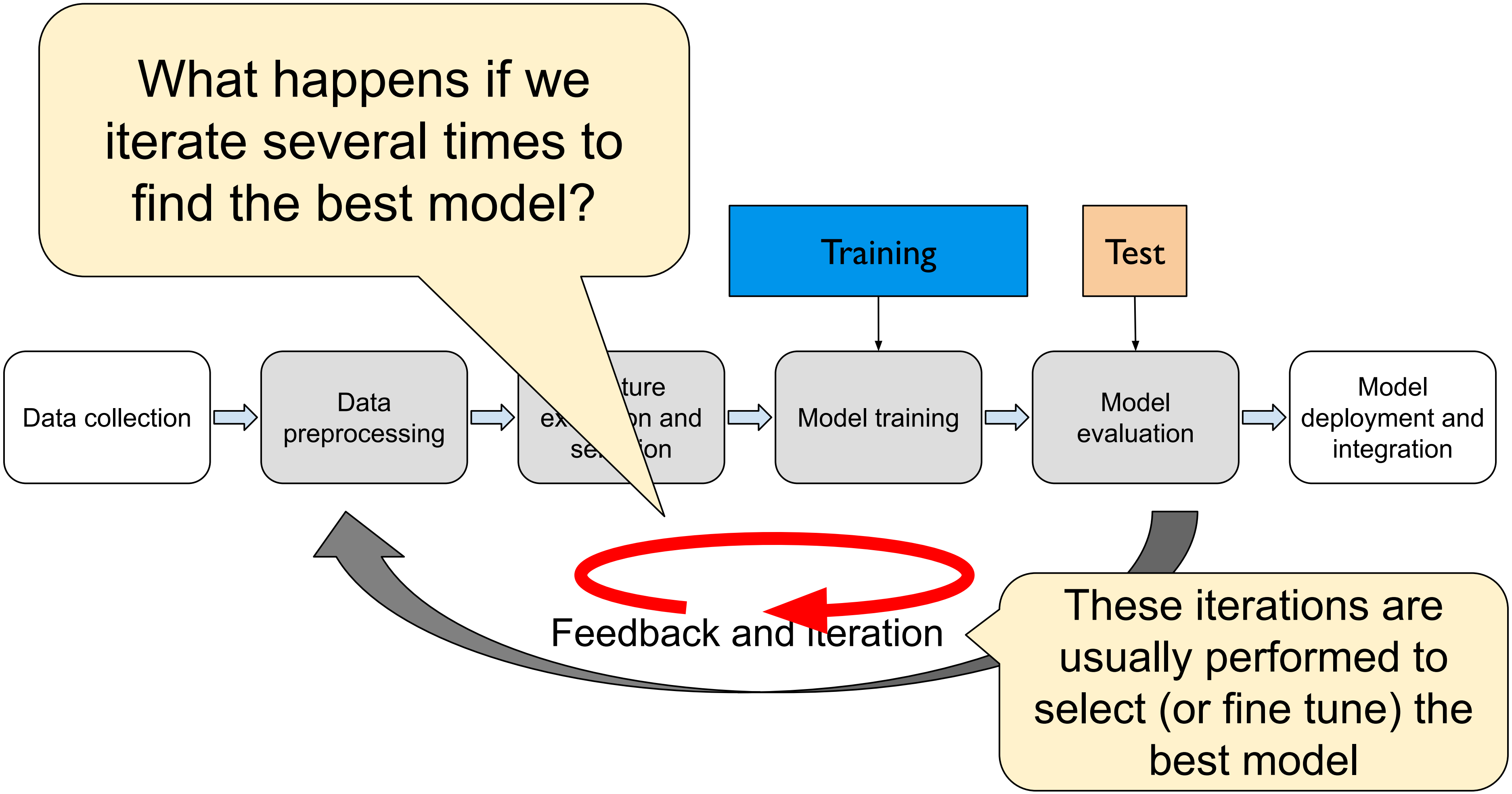
Test set and ML process iterations.

Previously…



ML Process - Dataset splitting

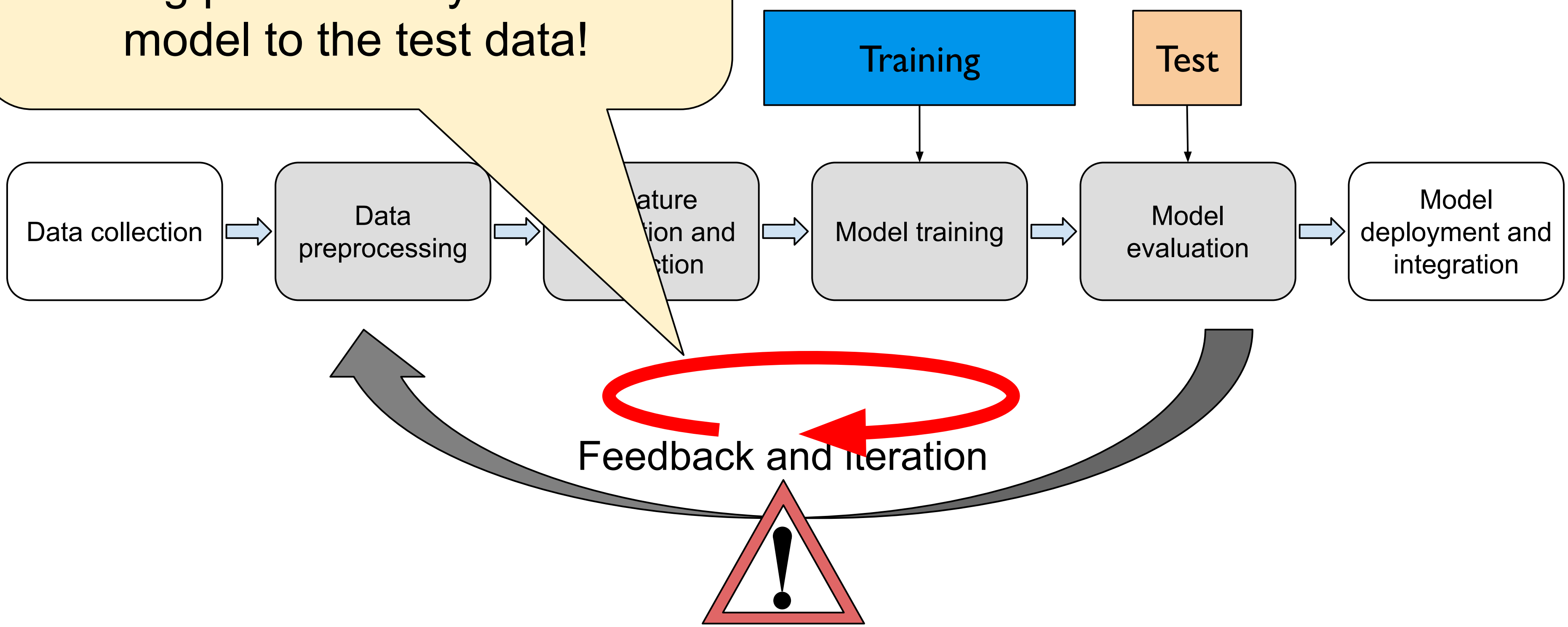
Test set and ML process iterations.



ML Process - Dataset splitting

Test set and ML process iterations.

Using the test results to repeatedly change/improve your learning process may overfit the model to the test data!

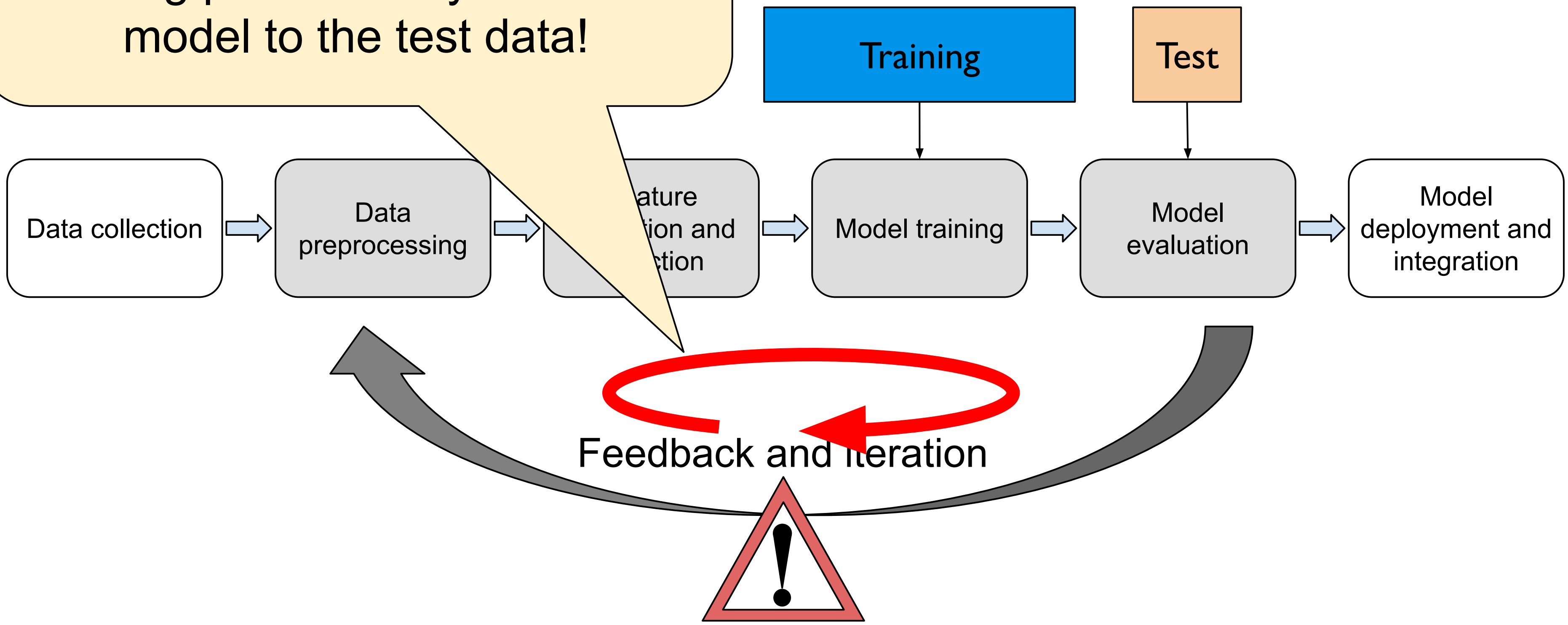


ML Process - Dataset splitting

Test set and ML process itera

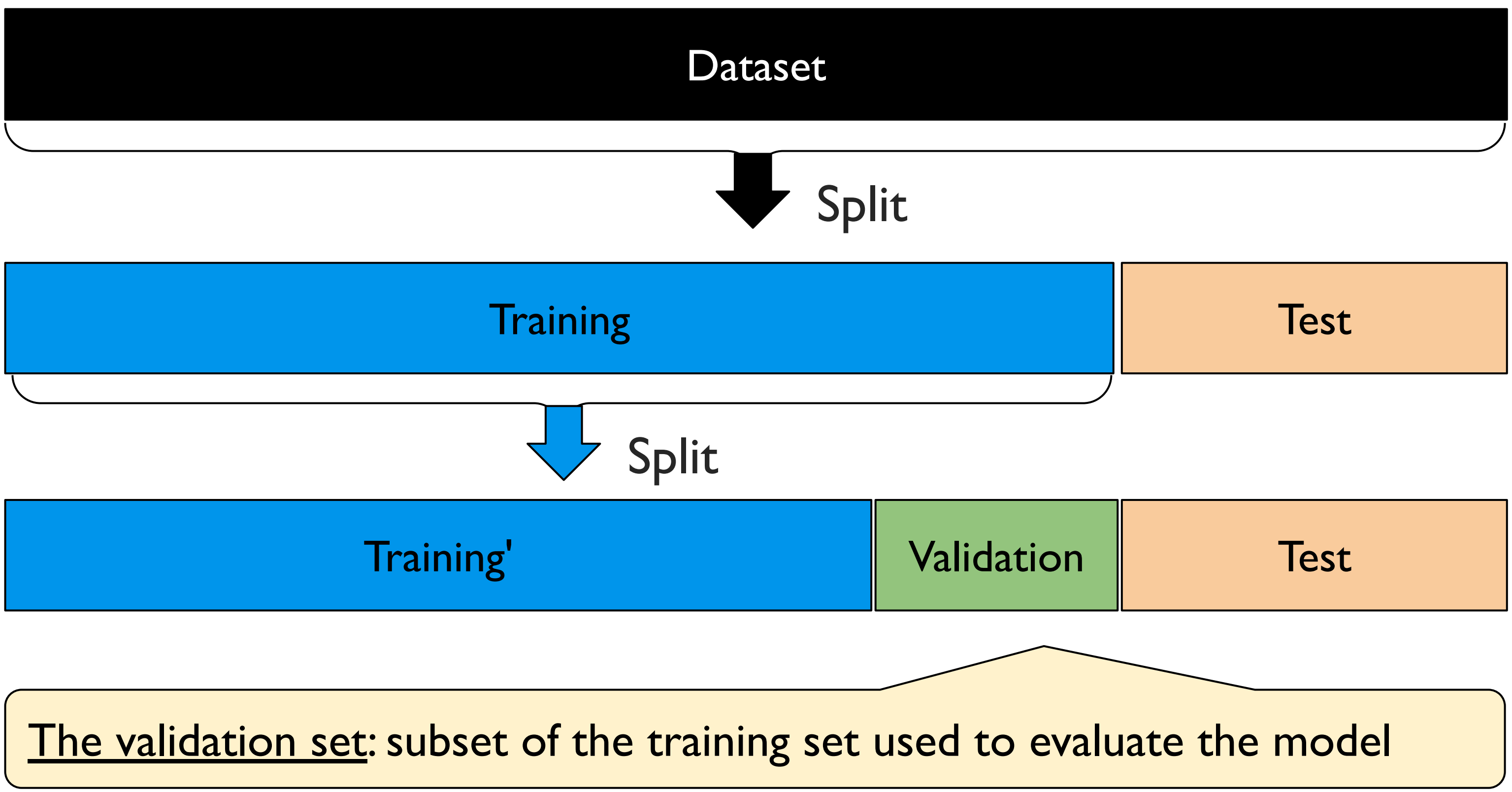
Using the test results to repeatedly change/improve your learning process may overfit the model to the test data!

Test set wears out and cannot be used to evaluate generalization anymore



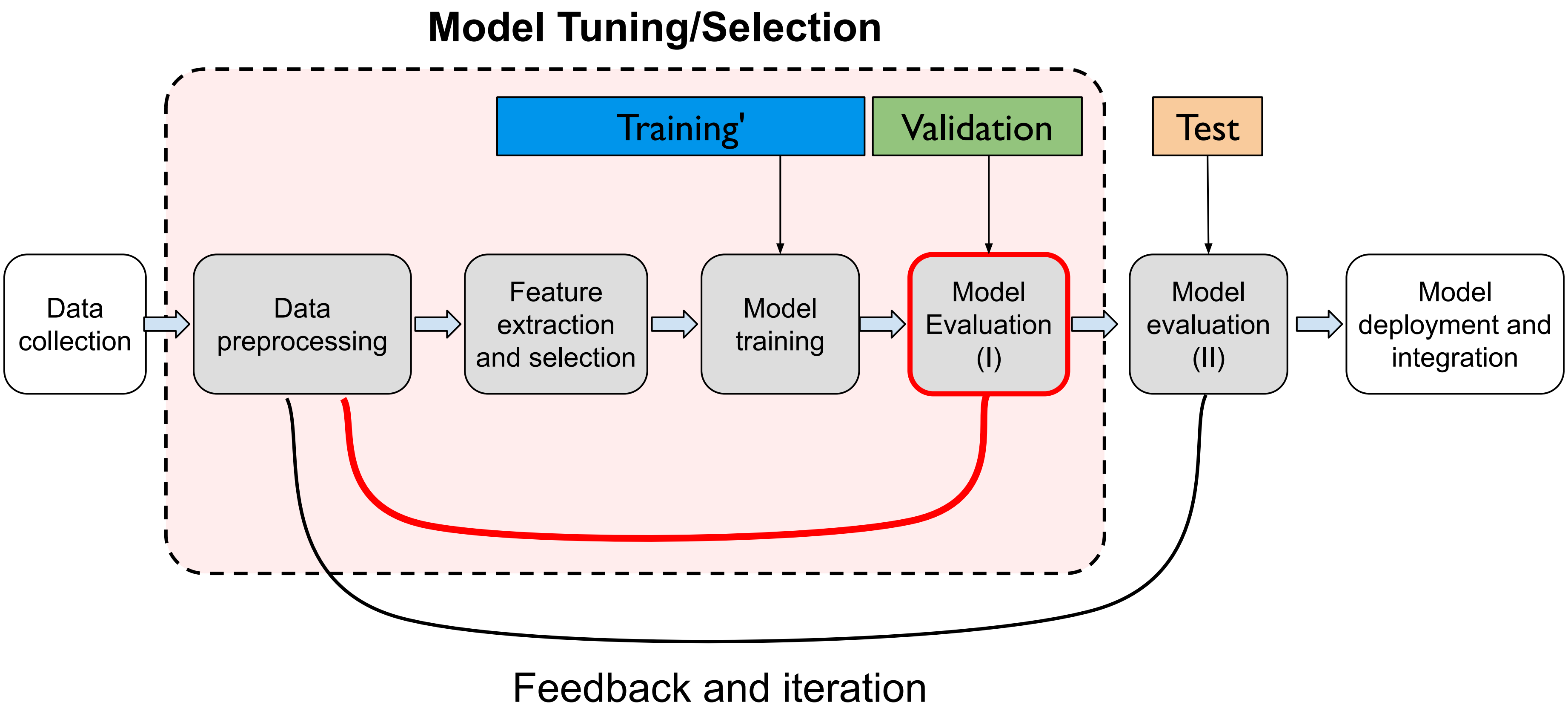
ML Process - Dataset splitting

Cross-validation: use different portions of the training set to train and to evaluate the model



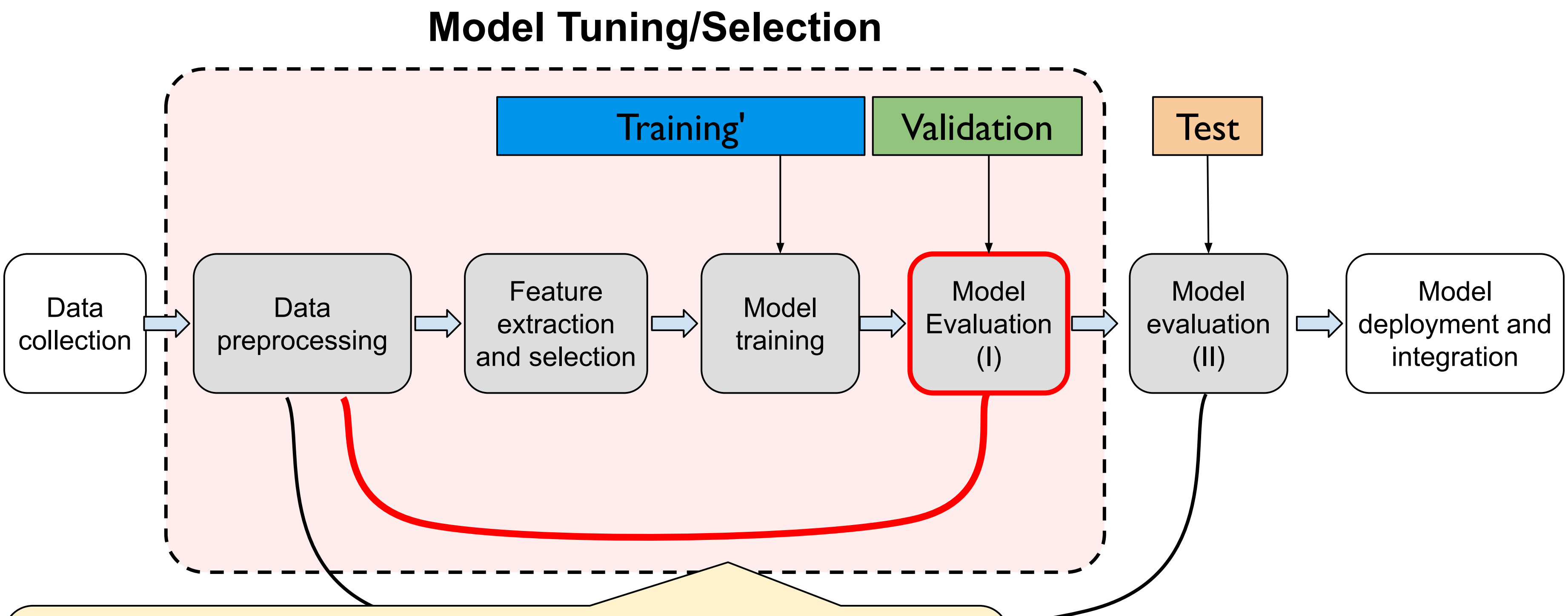
ML Process - Dataset splitting

<u>Cross-validation</u>: use different portions of the training set to train and to evaluate the model



ML Process - Dataset splitting

Cross-validation: use different portions of the training set to train and to evaluate the model

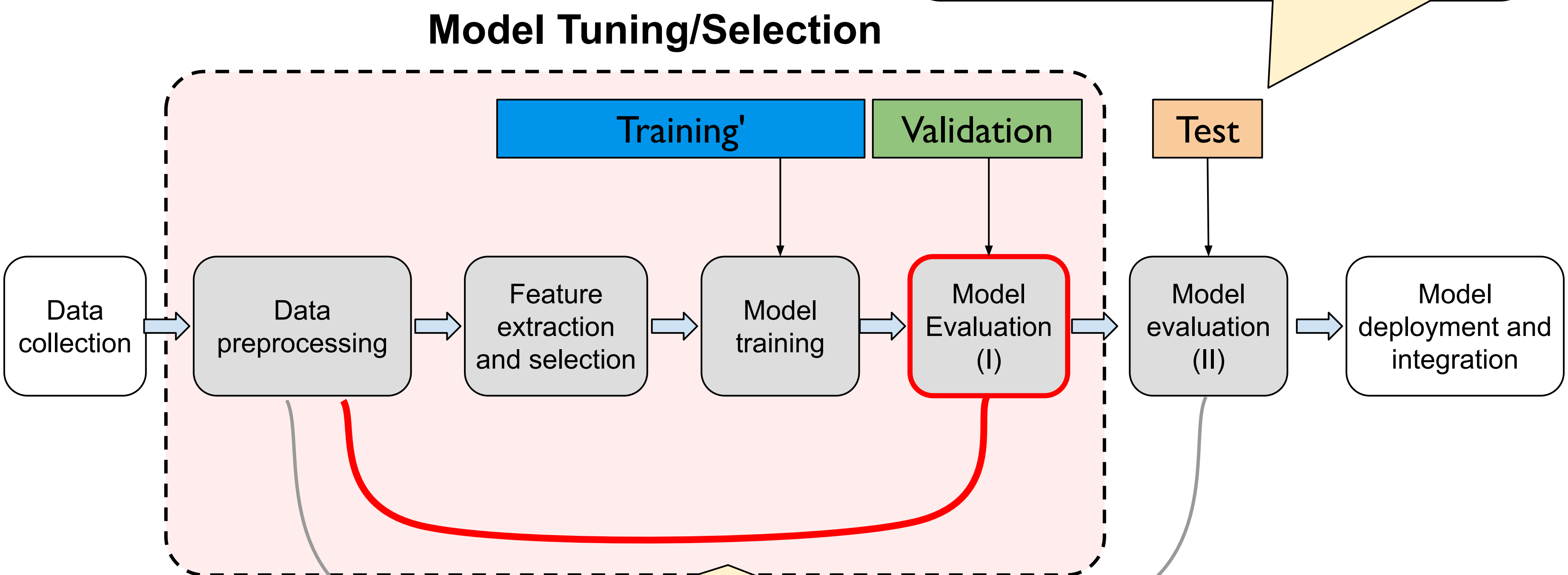


Pick model using validation set results and "double-check" it using the test set.

ML Process - Dataset splitting

Cross-validation: use different p... set to train and to evaluate the...

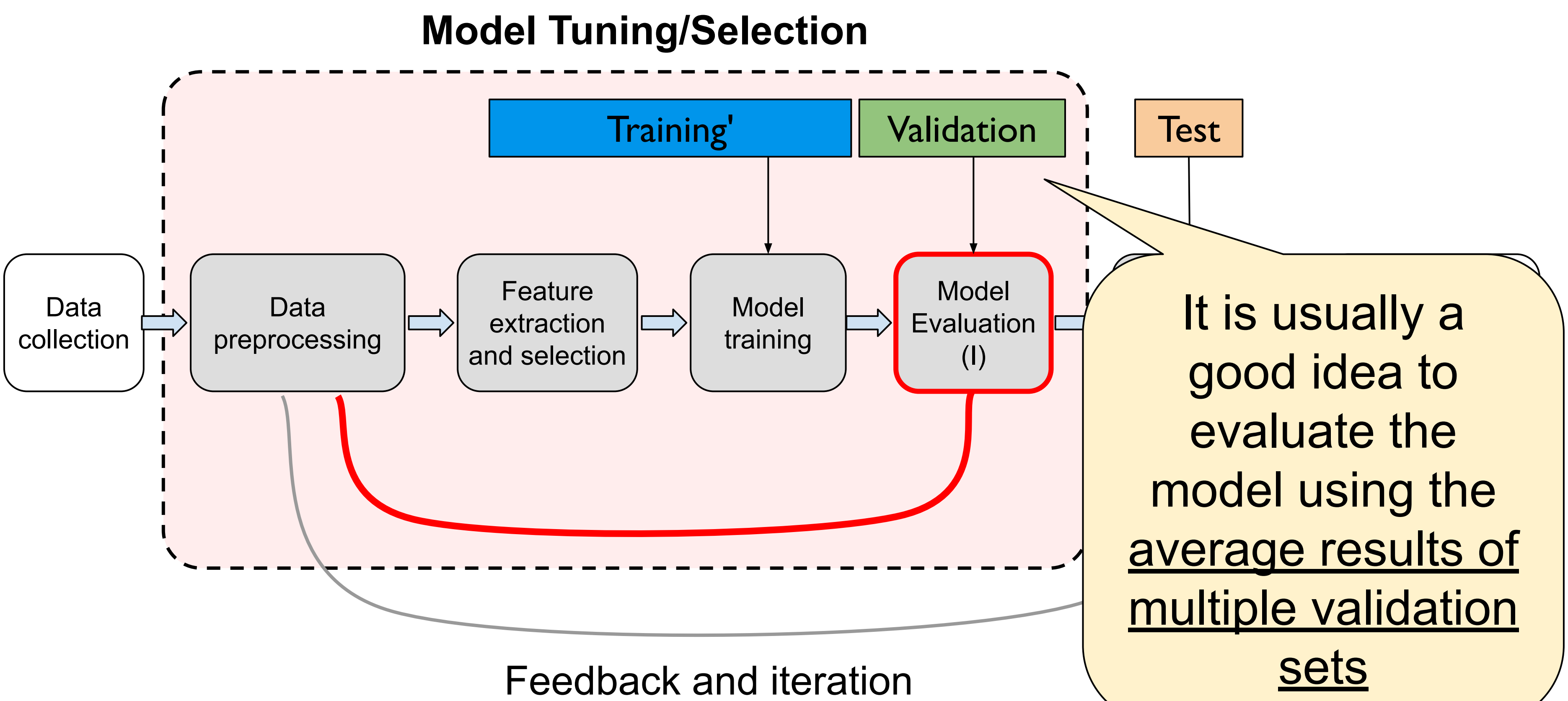
Unfrequently used!
Ideally, only once!



Pick model using validation set results and "double-check" it using the test set.

ML Process - Dataset splitting

Cross-validation: use different portions of the training set to train and to evaluate the model



ML Process - Dataset splitting

Cross-validation: use different portions of the training set to train and to evaluate the model

Several approaches:

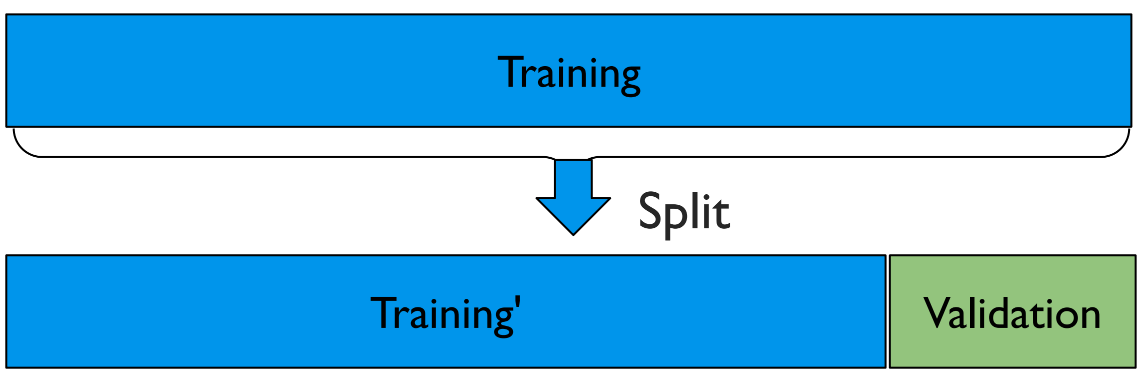
- **Holdout method**
- **Leave-one-out cross-validation**
- **k-fold cross-validation**
- Leave-p-out cross-validation
- repeated random sub-sampling validation
- k*l-fold cross validation
- …

ML Process - Dataset splitting

Cross-validation: use different portions of the training set to train and to evaluate the model

Several approaches:

- **Holdout method**: single train/validation partition randomly selected

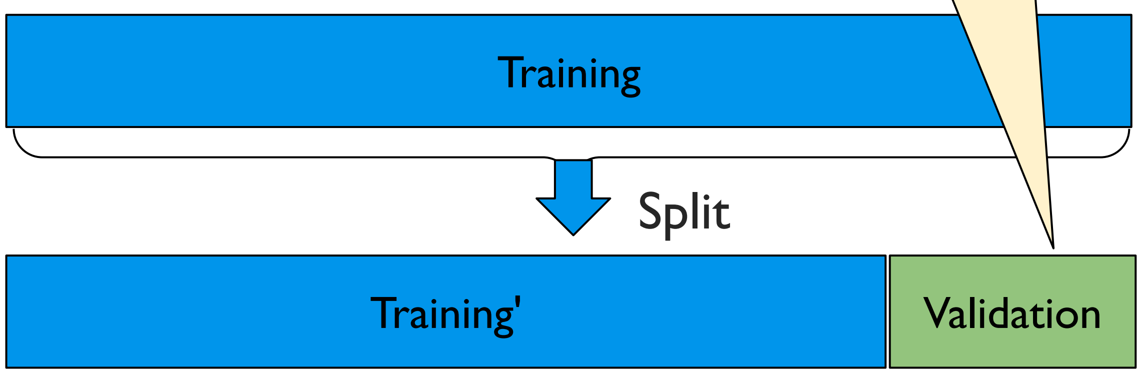


ML Process - Dataset splitting

<u>Cross-validation</u>: use different p g
set to train and to evaluate the
Several approaches:

- **Holdout method**: single train/validatio tion randomly selected

Single partition may cause evaluation bias.

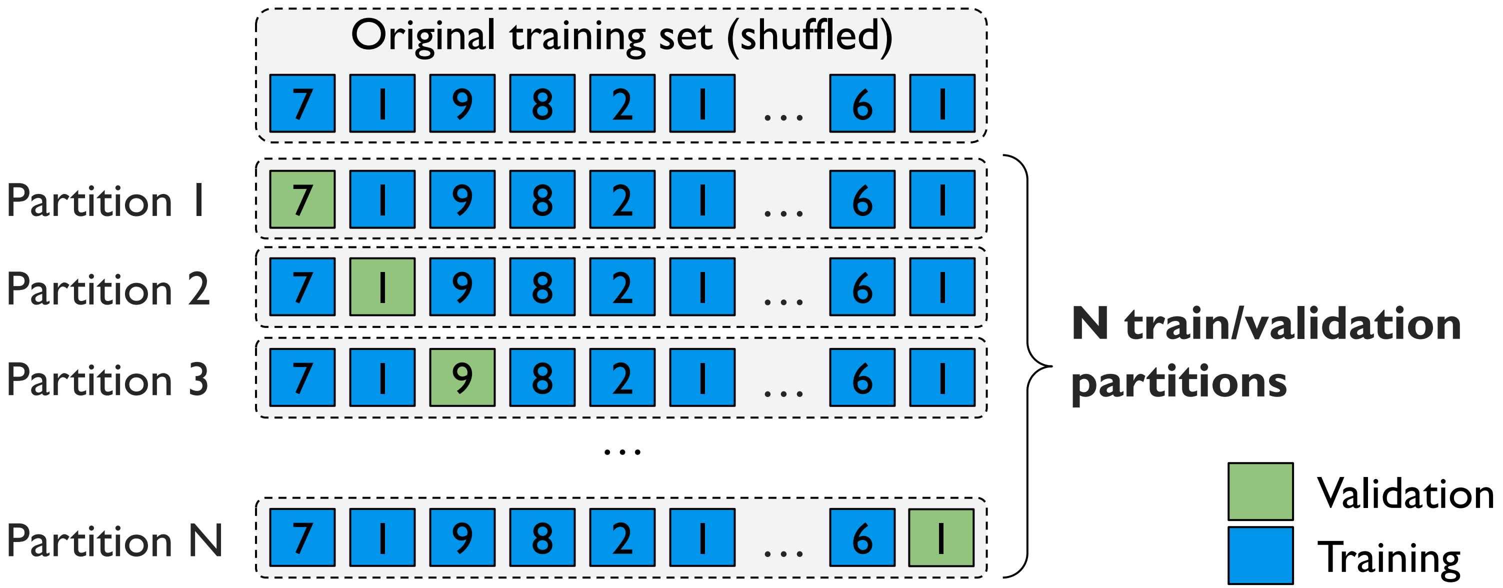


ML Process - Dataset splitting

<u>Cross-validation</u>: use different portions of the training set to train and to evaluate the model

Several approaches:

- **Leave-one-out cross-validation**: 1 partition per item

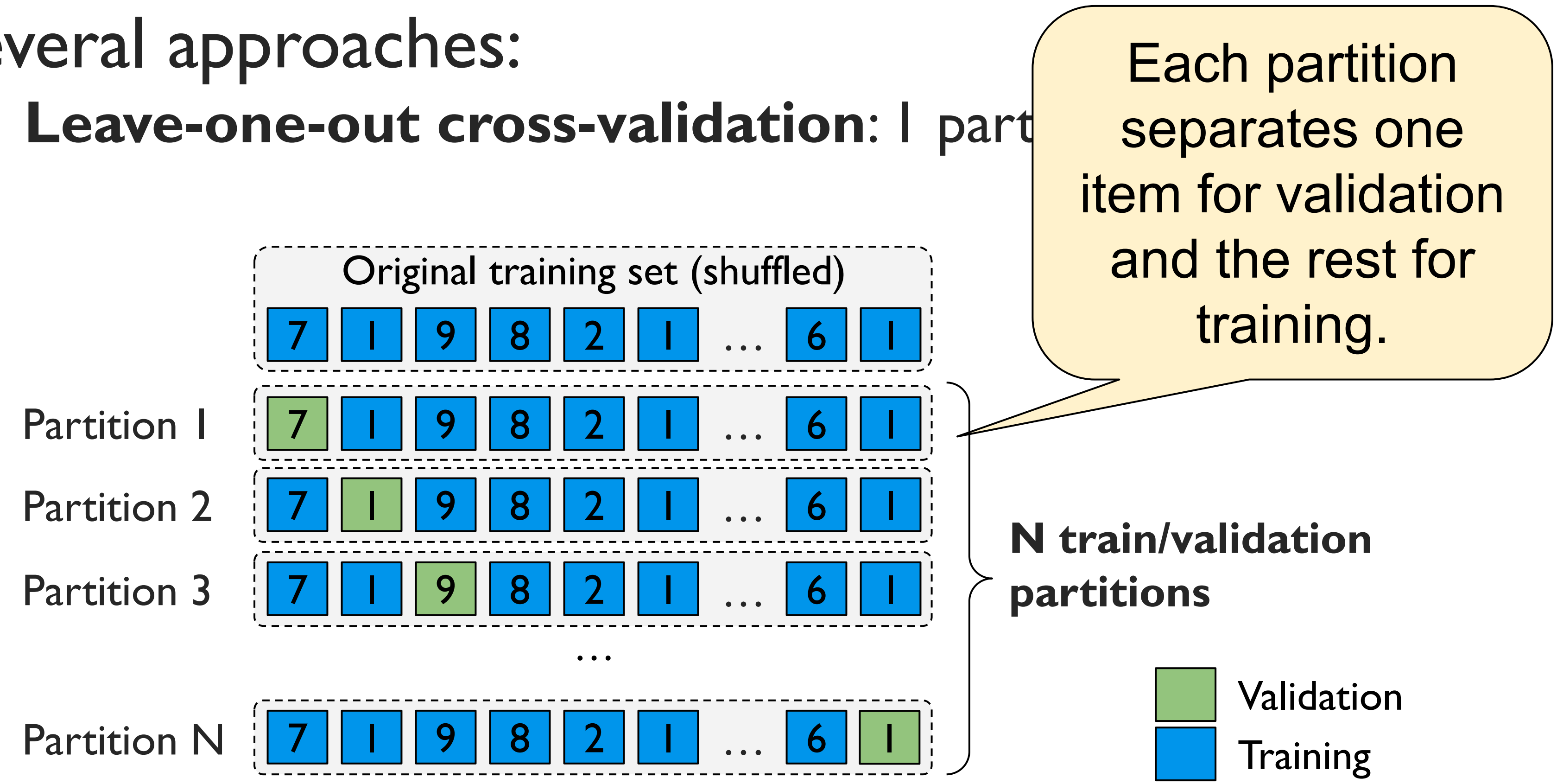


ML Process - Dataset splitting

Cross-validation: use different portions of the training set to train and to evaluate the model

Several approaches:

- **Leave-one-out cross-validation**: 1 part

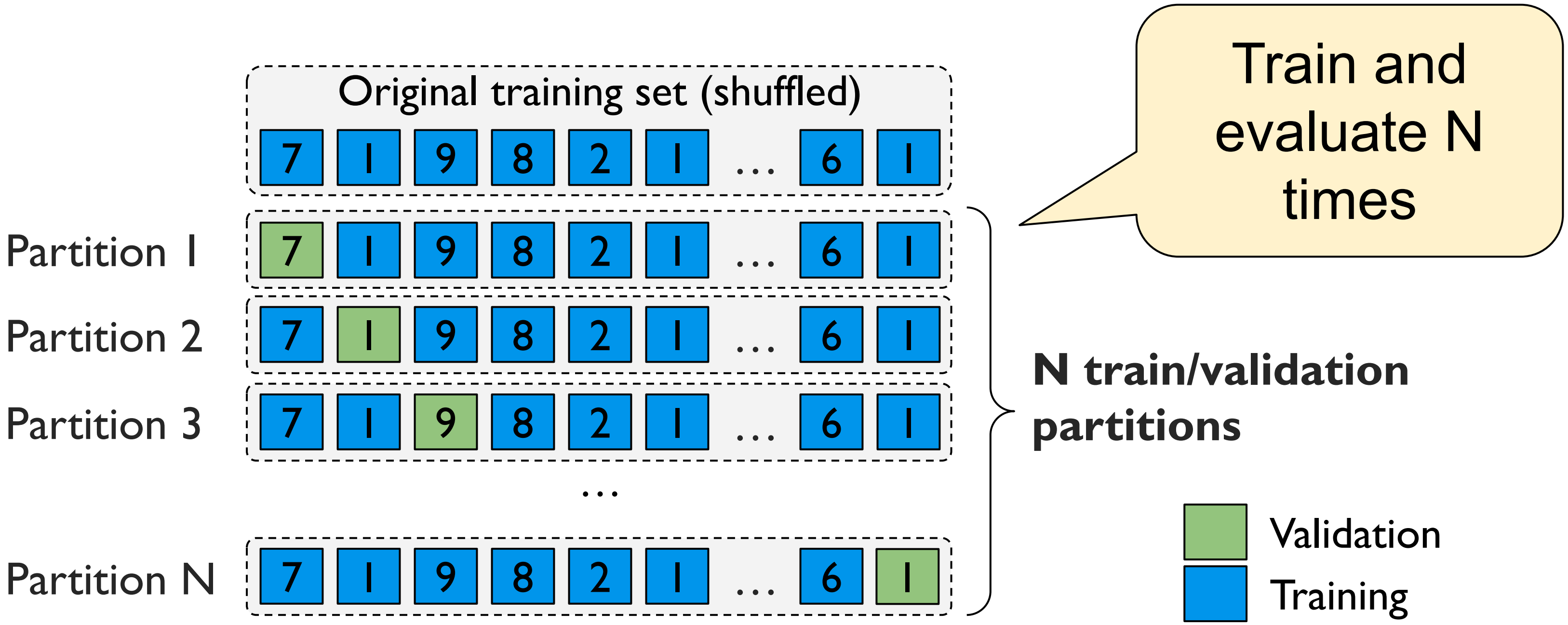


ML Process - Dataset splitting

Cross-validation: use different portions of the training set to train and to evaluate the model

Several approaches:

- **Leave-one-out cross-validation**: 1 partition per item



ML Process - Dataset splitting

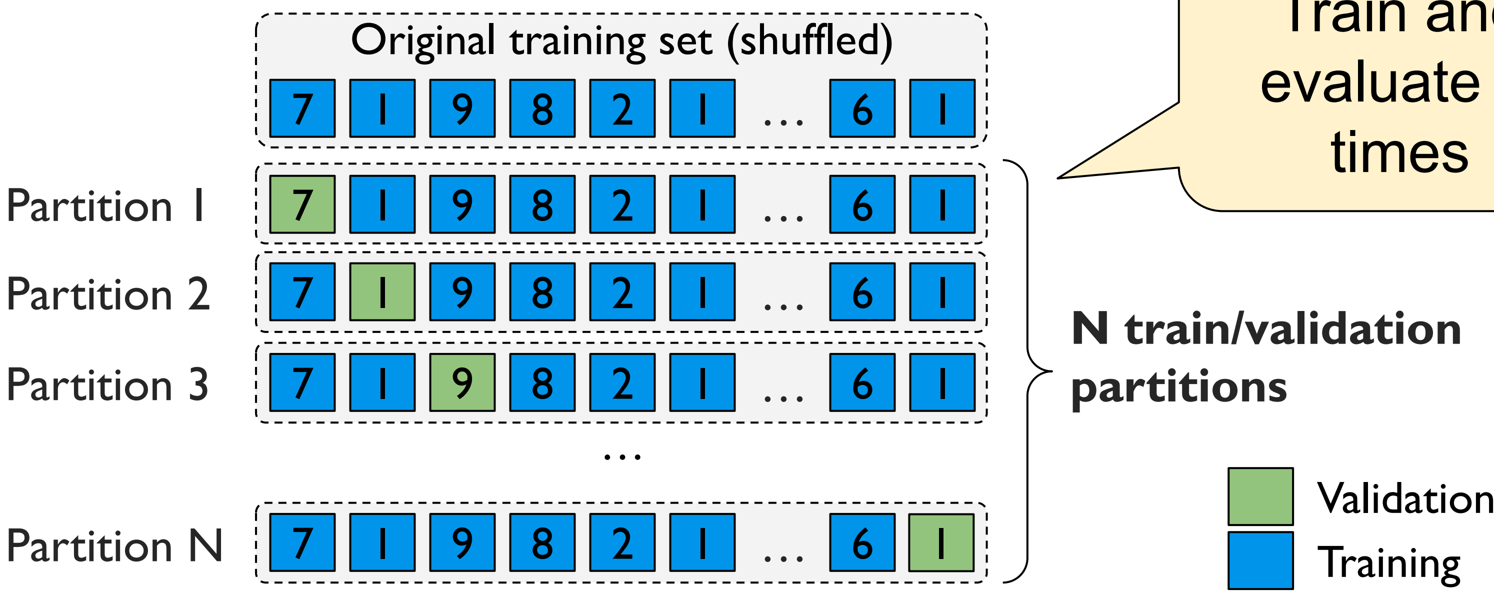
<u>Cross-validation</u>: use different portions g set to train and to evaluate the model

Several approaches:

- **Leave-one-out cross-validation**: 1 partition per item

Report average and stdev

Train and evaluate N times



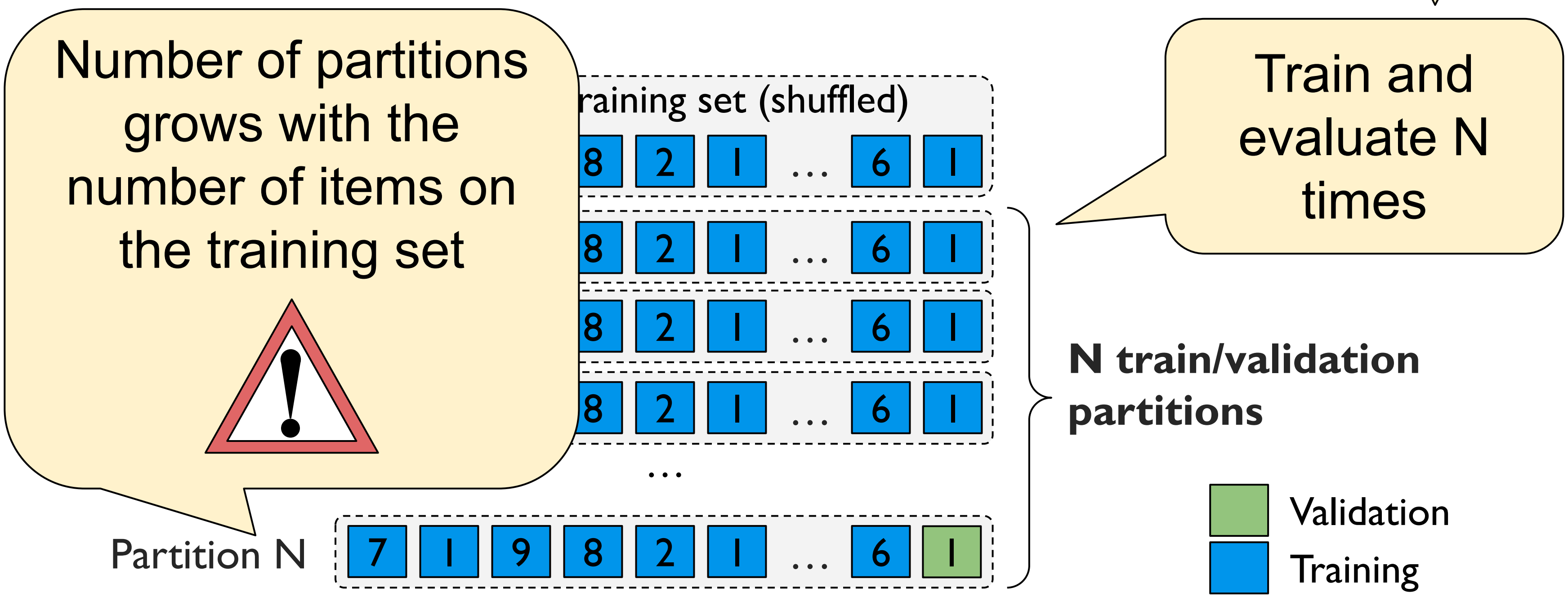
ML Process - Dataset splitting

Cross-validation: use different portions g set to train and to evaluate the model

Report average and stdev

Several approaches:

- **Leave-one-out cross-validation**: 1 partition per item

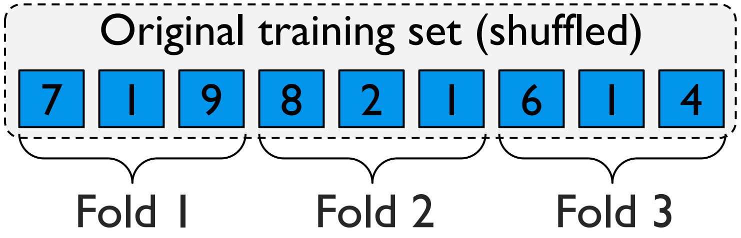


ML Process - Dataset splitting

Cross-validation: use different portions of the training set to train and to evaluate the model

Several approaches:

- **k-fold cross-validation**: split the data in K folds and generate 1 partition per fold
- Example: 3-fold cross-validation

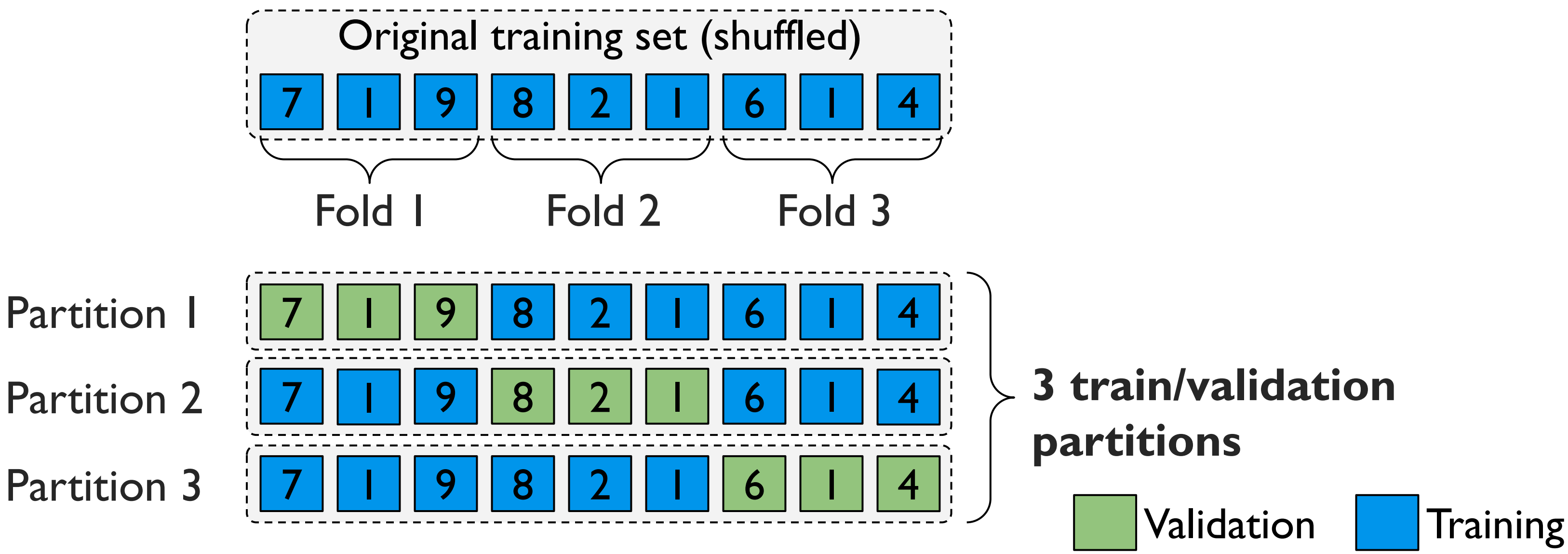


ML Process - Dataset splitting

Cross-validation: use different portions of the training set to train and to evaluate the model

Several approaches:

- **k-fold cross-validation**: split the data in K folds and generate 1 partition per fold
- Example: 3-fold cross-validation



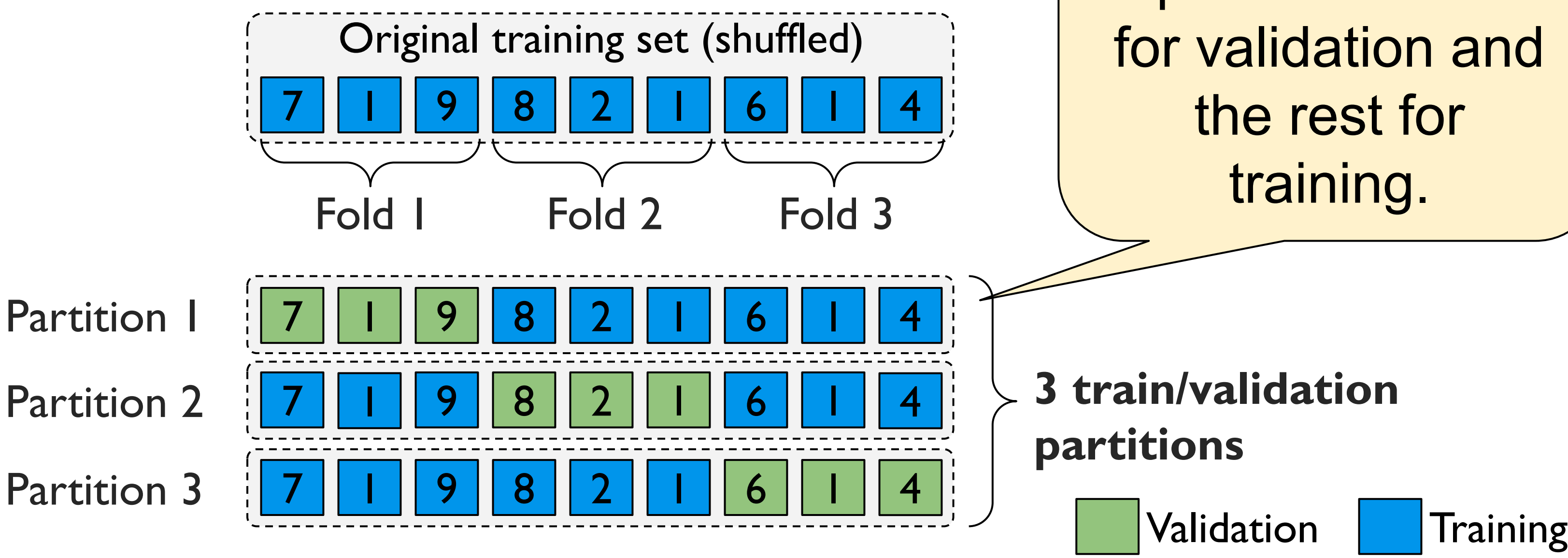
ML Process - Dataset splitting

Cross-validation: use different portions of the training set to train and to evaluate the model

Several approaches:

- **k-fold cross-validation**: split the data in K folds and generate 1 partition per fold
- Example: 3-fold cross-validation

Each partition separates one fold for validation and the rest for training.

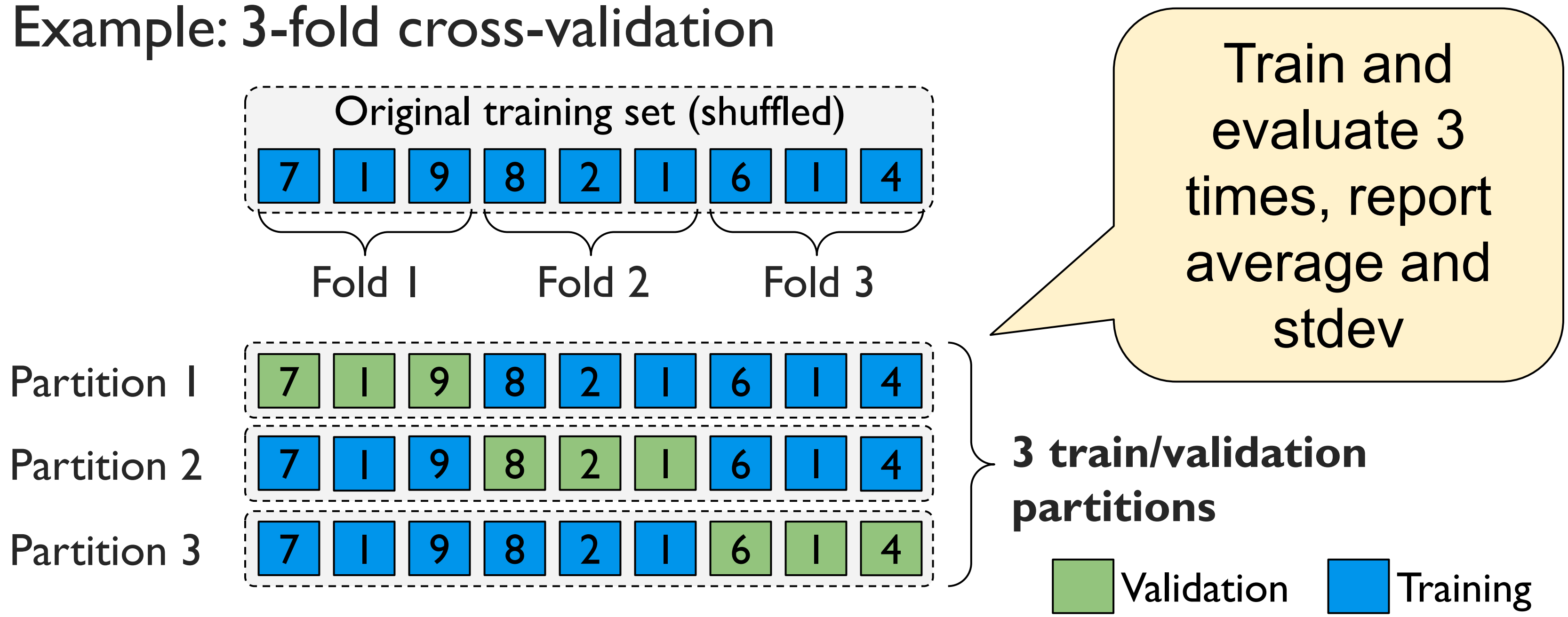


ML Process - Dataset splitting

Cross-validation: use different portions of the training set to train and to evaluate the model

Several approaches:

- **k-fold cross-validation**: split the data in K folds and generate 1 partition per fold
- Example: 3-fold cross-validation



ML Process - Dataset splitting

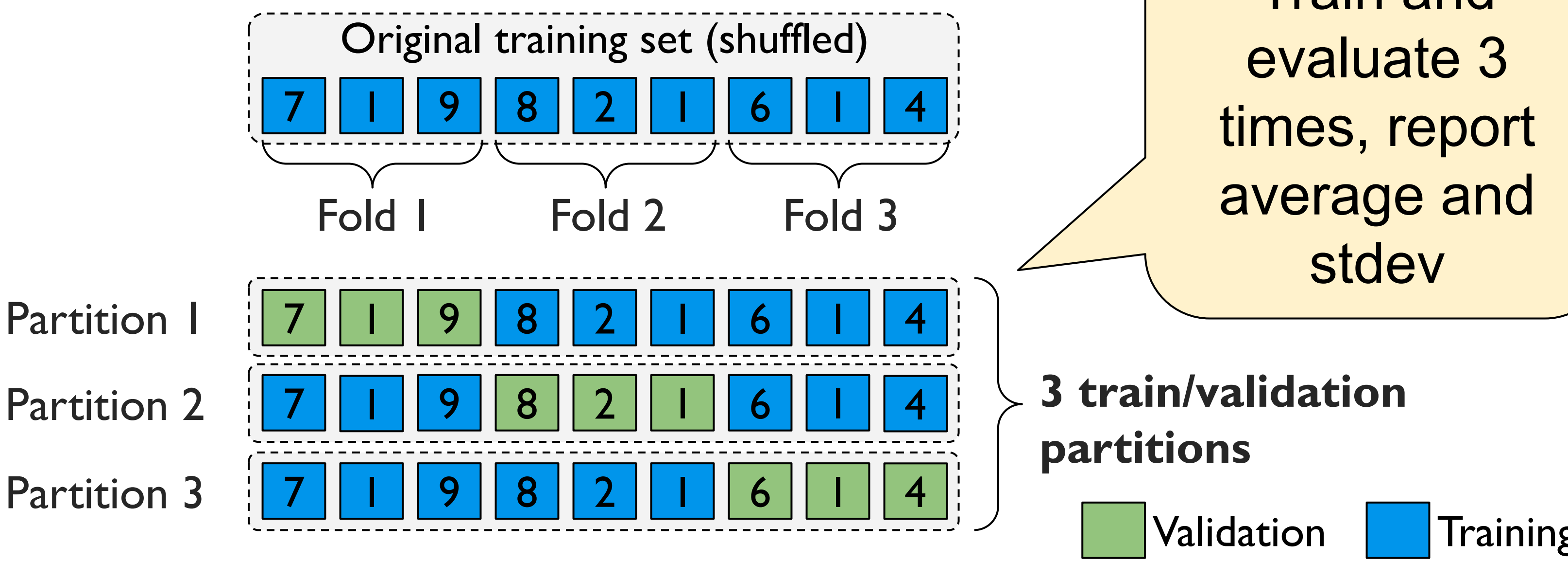
Cross-validation: use different portions of the training set to train and to evaluate the

Several approaches:

- **k-fold cross-validation**: split the ... nerate 1 partition per fold
- Example: 3-fold cross-validation

Number of partitions and training/validation operations = K

Train and evaluate 3 times, report average and stdev



ML Process - Dataset splitting

Key takeaways

- Training Set: part of the dataset used to train the model
- Validation Set: part of the dataset used to evaluate the model when searching for the best model or best set of hyperparameters
- Test set: part of the dataset set aside for final model evaluation. Ideally, should be used only once!
- Cross-validation: resampling method that uses different portions of the training set to train and evaluate models on different iterations
  - k-fold cross-validation: split the data in K folds and generate k partitions - each one using a different fold for validation and the remaining ones for training

Instituto de Computação
UNIVERSIDADE ESTADUAL DE CAMPINAS

Capacitação profissional em tecnologias de Inteligência Artificial

Machine Learning Overview

Prof. Edson Borin
https://www.ic.unicamp.br/~edson
Institute of Computing - UNICAMP