# Advanced Machine Learning

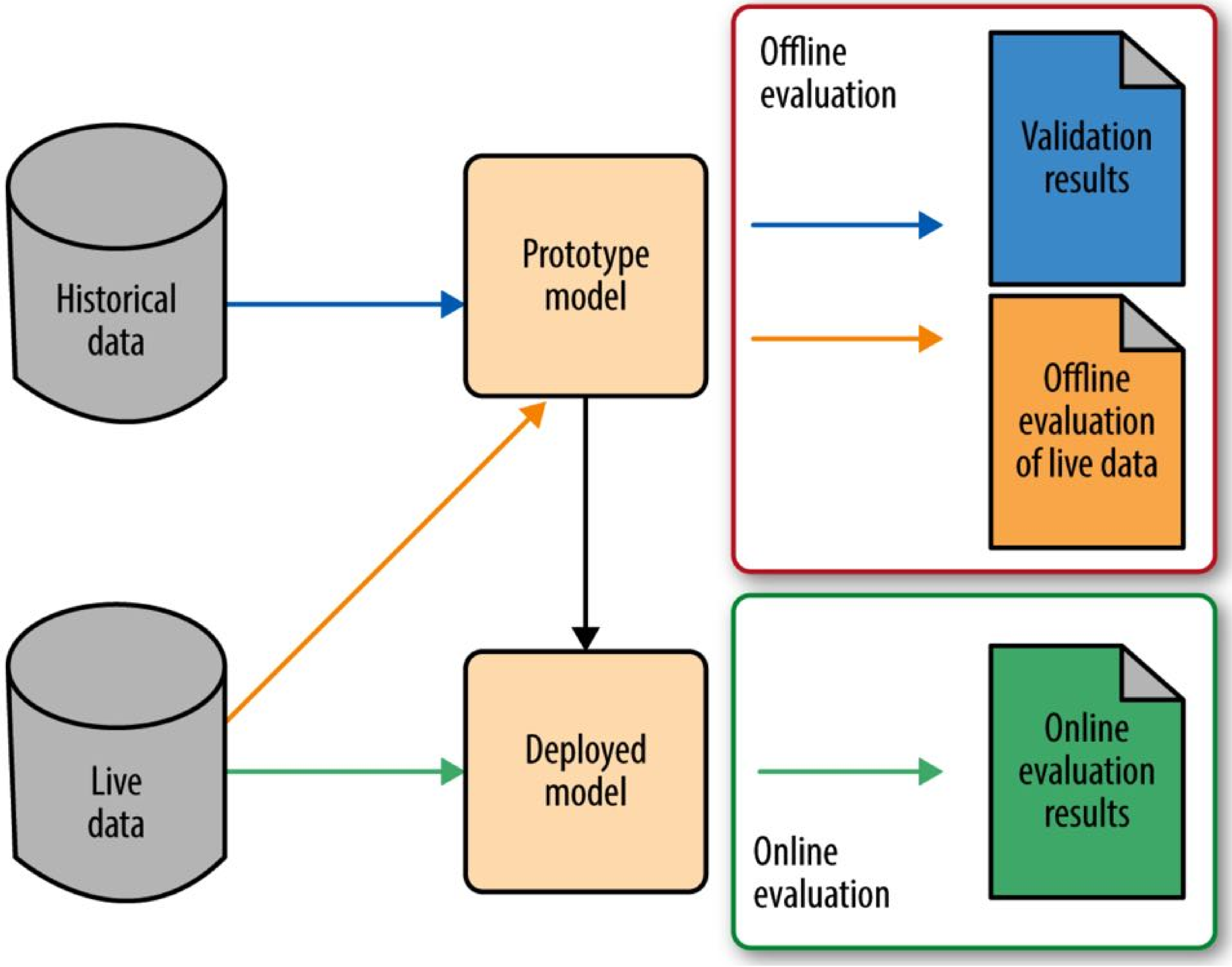
## Model Evaluation and Selection


Instructor: Rossano Schifanella

@UDD

# Recap

# Model Development and Evaluation Workflow



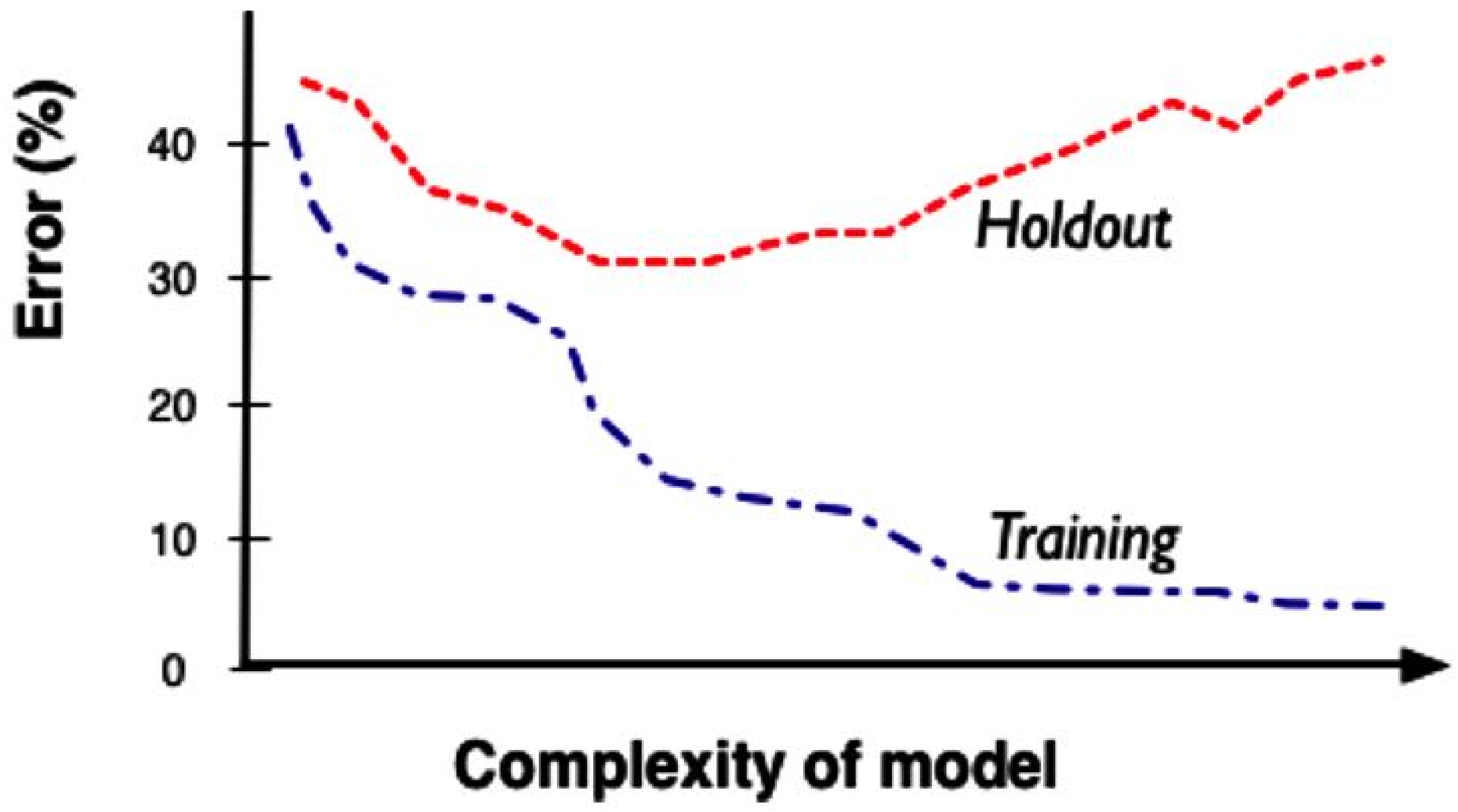
# Model Evaluation and Selection

# Some Important Concepts

- **Complexity** of a model
  - **model hyperparameters**
    - are the **tuning parameters** of a machine learning algorithm — for example, the maximum depth of a decision tree.
  - **model parameters**
    - are the **parameters that a learning algorithm fits to the training data** – the parameters of the model itself. For example, the weight coefficients (or slope) of a linear regression line.
- **Generalization**
- **Overfitting**

# Fitting Graph

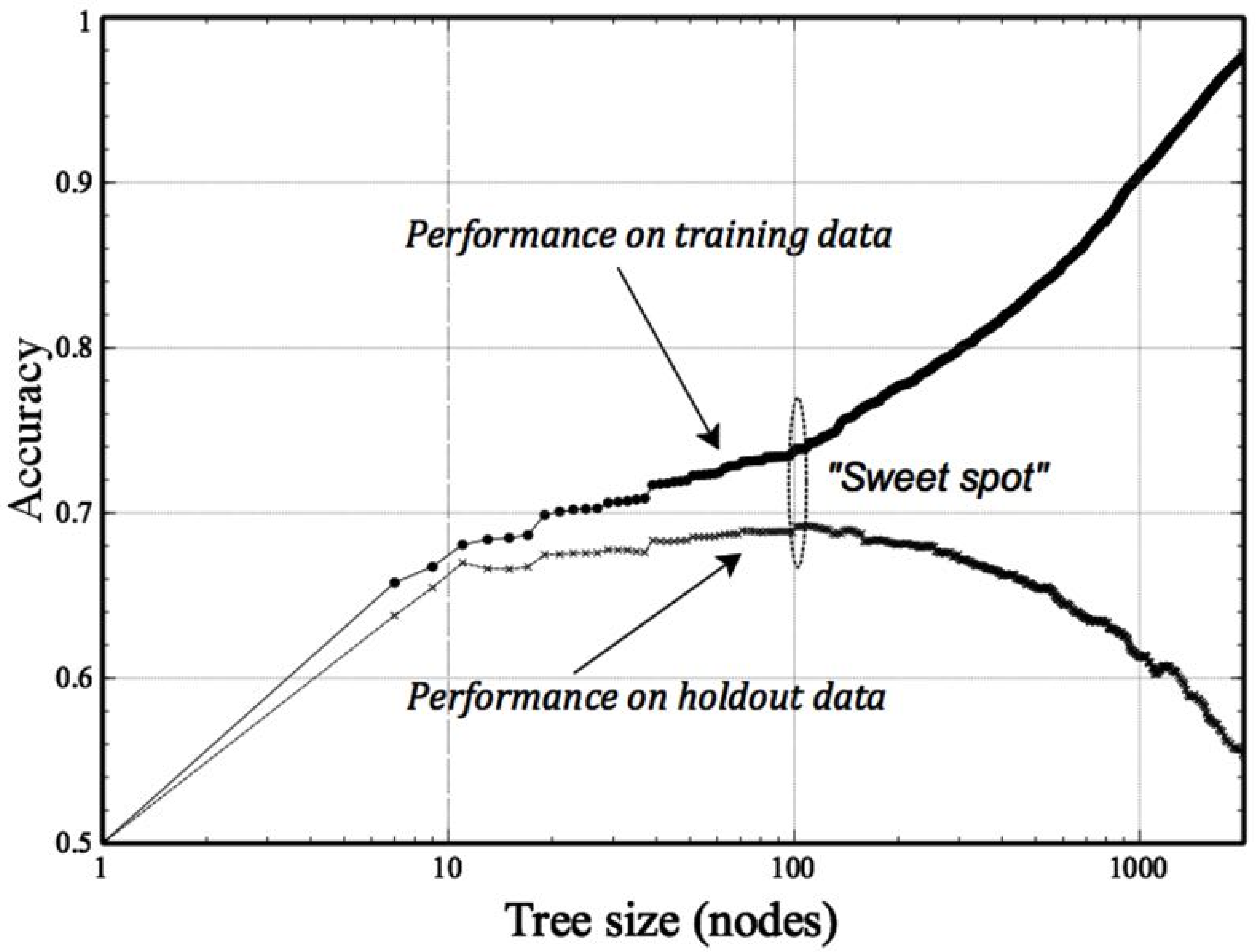
- Each point on a curve represents an accuracy estimation of a model with a **specified complexity**
  - **Holdout = Testing Set = Validation Set** (don't get confused by the terminology!)
- When the model is not allowed to be complex enough, it is not very accurate.
- As a model gets too complex, **it looks very accurate on the training data**
  - **Overfitting: training accuracy diverges from the holdout (generalization) accuracy.**



# Fitting Graph (decision tree example)

- Each point on a curve represents an accuracy estimation of a model with a **specified complexity**
  - **Holdout = Testing Set = Validation Set** (don't get confused by the terminology!)
- When the model is not allowed to be complex enough, it is not very accurate.
- As a model gets too complex, **it looks very accurate on the training data**
  - **Overfitting: training accuracy diverges from the holdout (generalization) accuracy.**

we measure **accuracy = (1-err)** in this example



# Evaluate the Predictive Performance of a Model

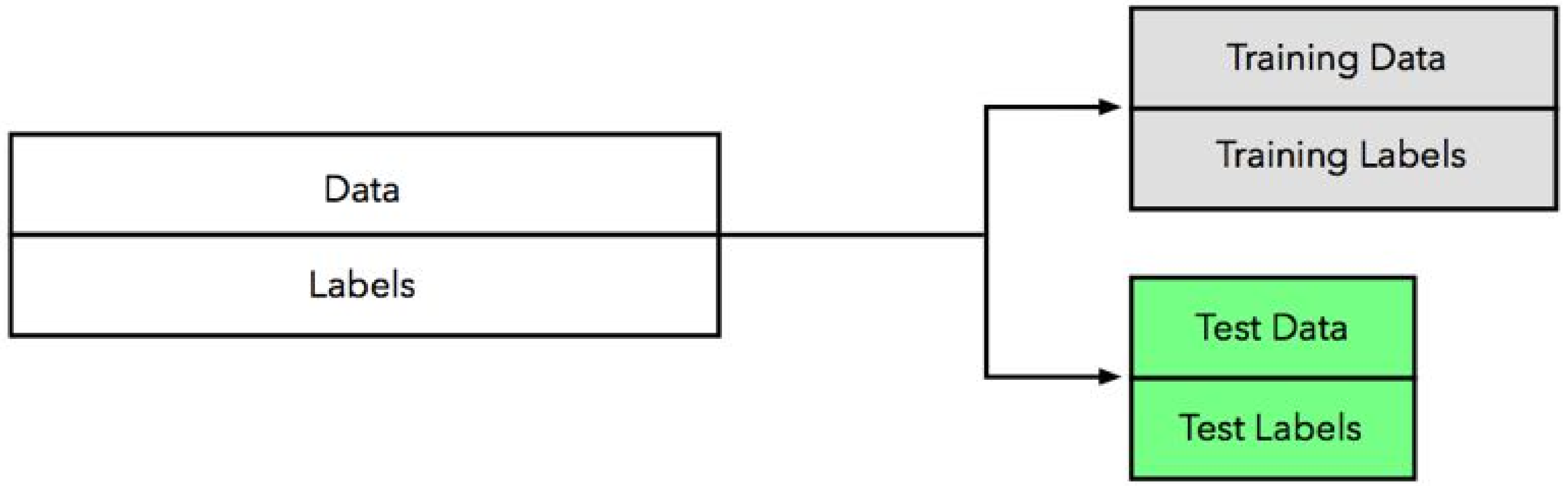
1. We want to estimate the **generalization performance**, the predictive performance of our model on future (unseen) data.
2. We want to **increase the predictive performance** by tweaking the learning algorithm and selecting the **best performing model from a given hypothesis space**.
3. We want to **identify the machine learning algorithm that is best-suited for the problem** at hand; thus, **we want to compare different algorithms**, selecting the best-performing one as well as the best performing model from the algorithm's hypothesis space.

# Model Evaluation

- **Hold-Out Validation**
- **Cross Validation**
  - k-cross fold validation
- **Nested Cross-Validation**

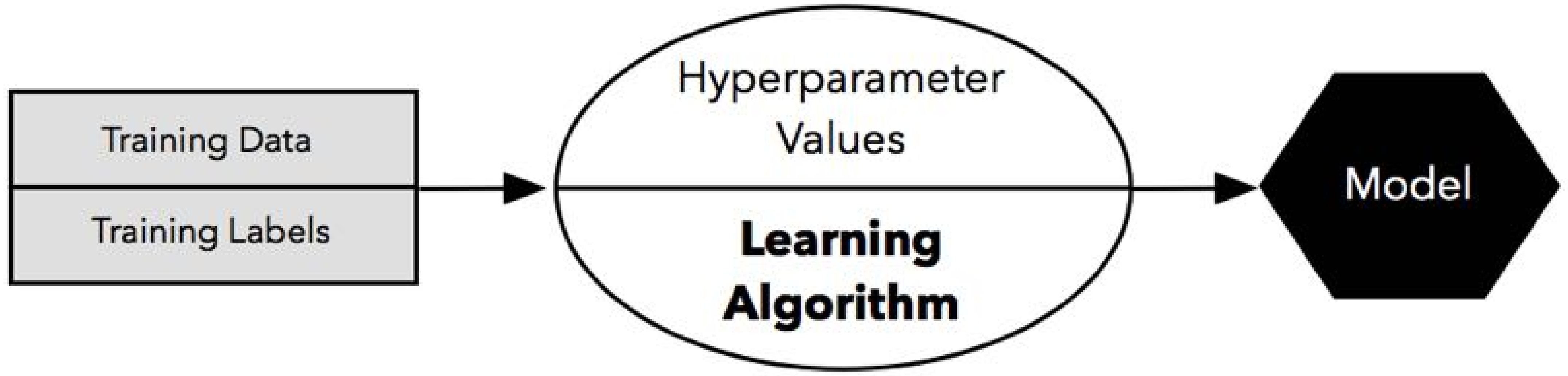
# HOLD-OUT

1



Data
Labels
Training Data
Training Labels
Test Data
Test Labels

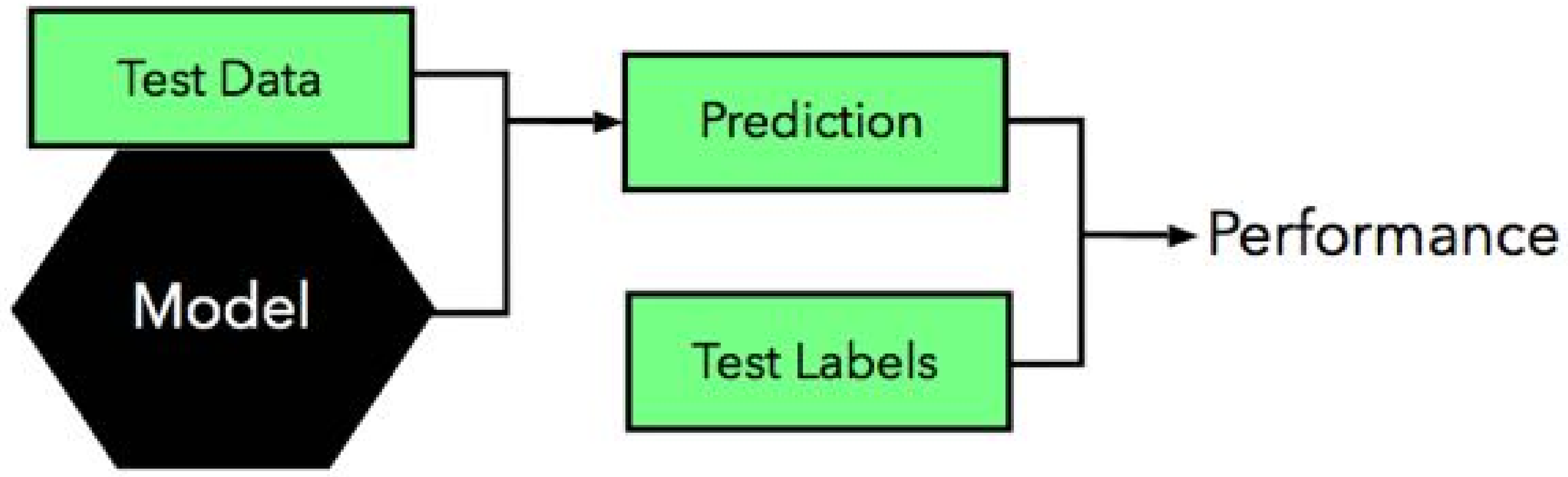

2



Training Data
Training Labels
Hyperparameter Values
Learning Algorithm
Model

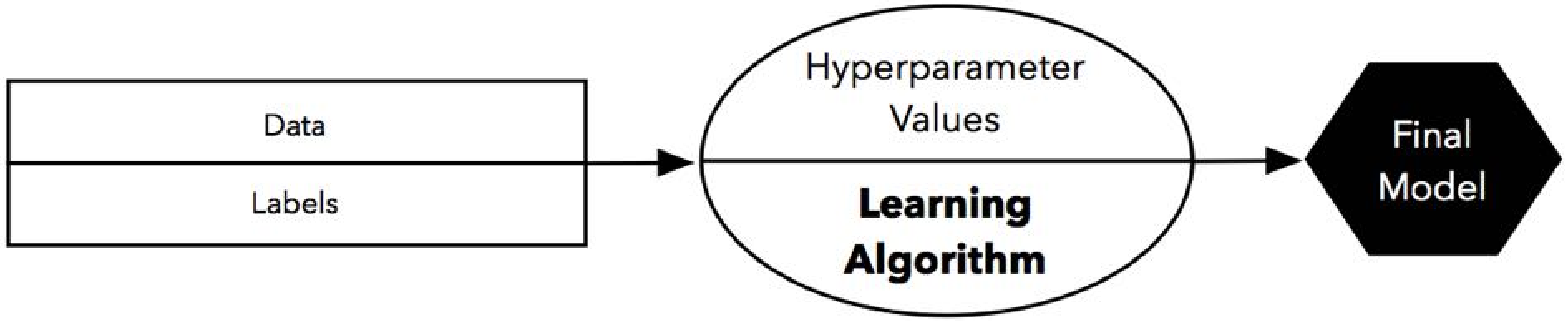
# HOLD-OUT

3



Test Data
Model
Prediction
Test Labels
Performance


4

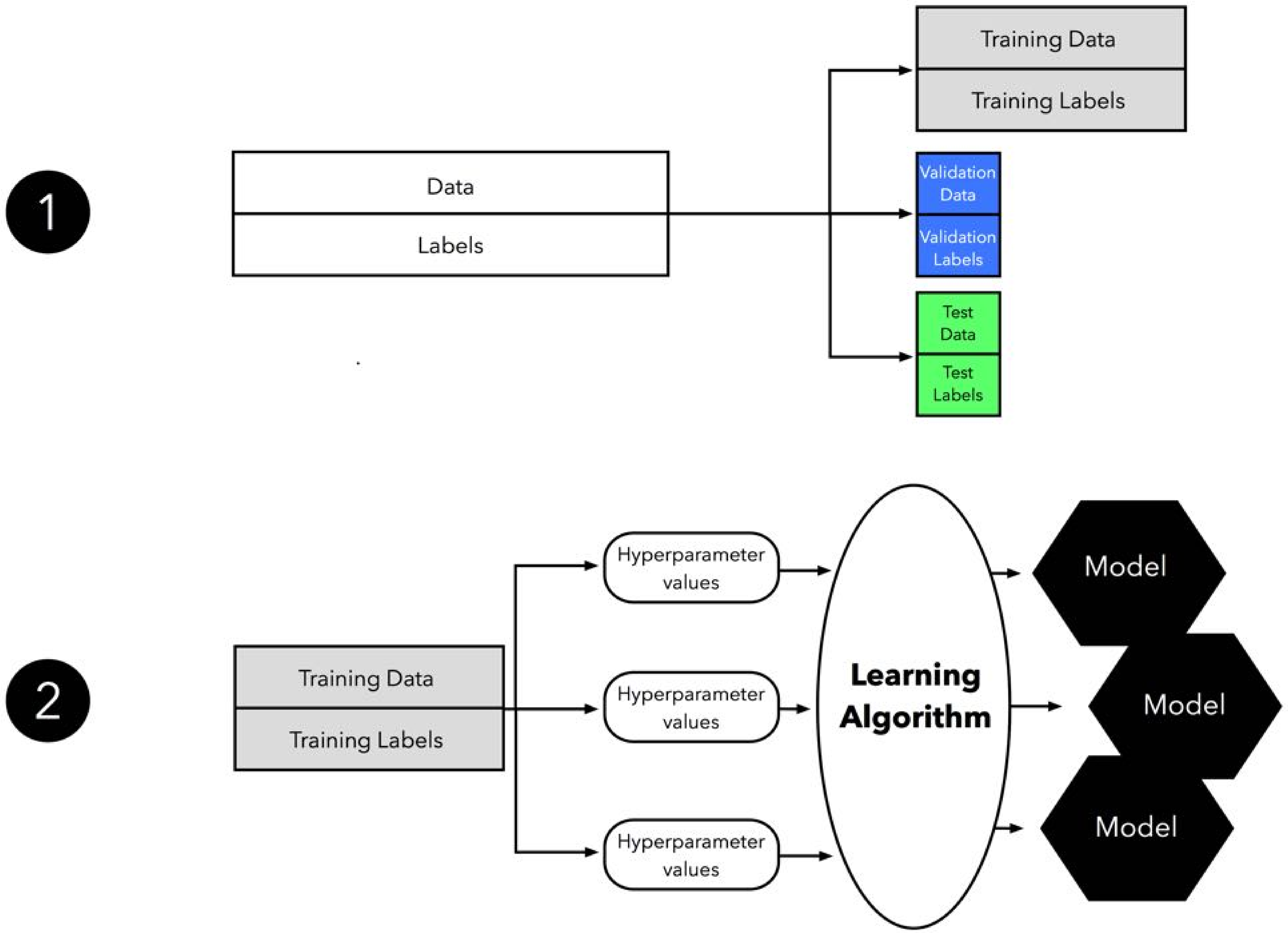


Data
Labels
Hyperparameter Values
Learning Algorithm
Final Model

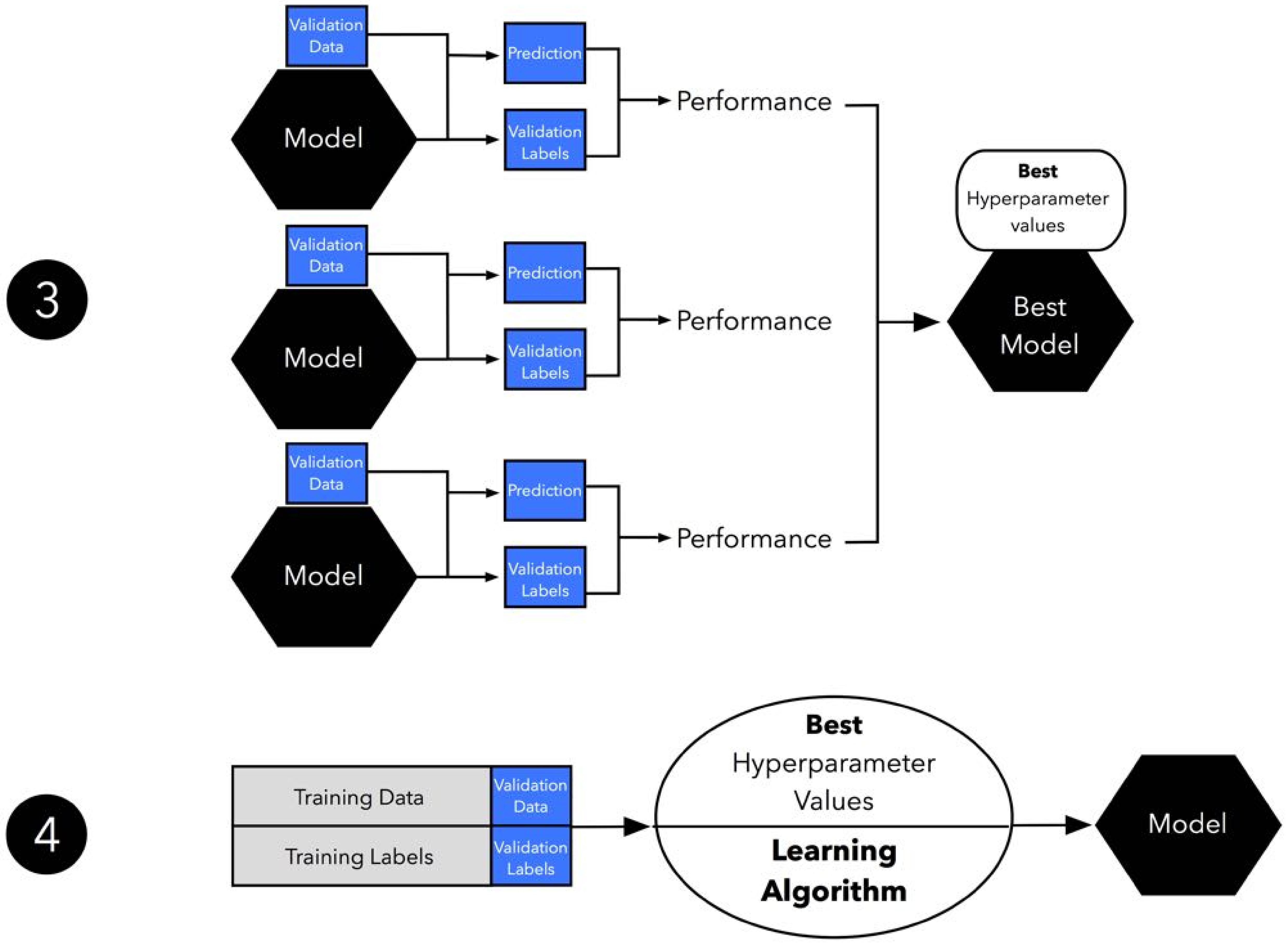
# Splitting in Training and Testing Data: Stratification

- Simple process of **random subsampling**.
  - Our dataset is a random sample drawn from a probability distribution; and we typically assume that this sample is **representative of the true population**
  - **Further subsampling without replacement** alters the statistic (mean, proportion, and variance) of the sample.
- **Stratification** randomly splits the dataset so that **each class is correctly represented in the resulting subsets**
- Not a big concern if we are working with large and balanced datasets
- Rule of thumb: in many occasions helps with k-fold cross validation (Kohavi 1995)

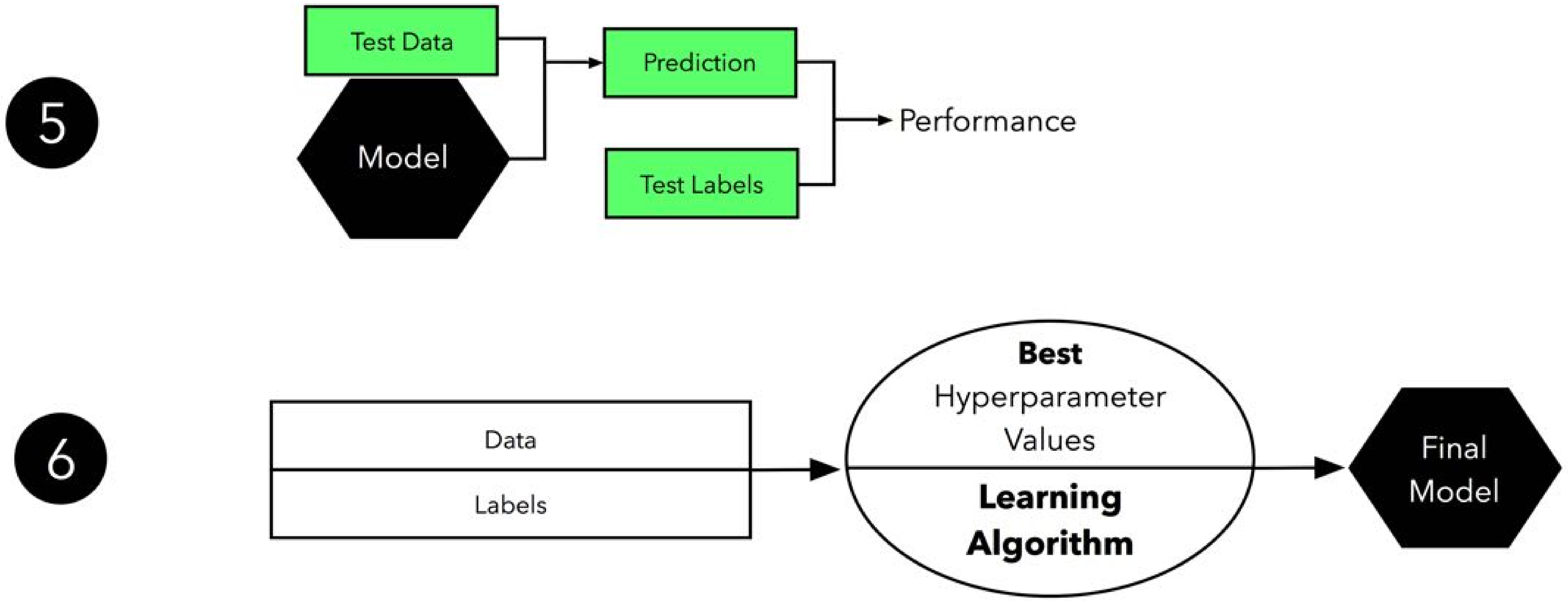
# Three-Way Holdout Method for Hyperparameter Tuning



# Three-Way Holdout Method for Hyperparameter Tuning



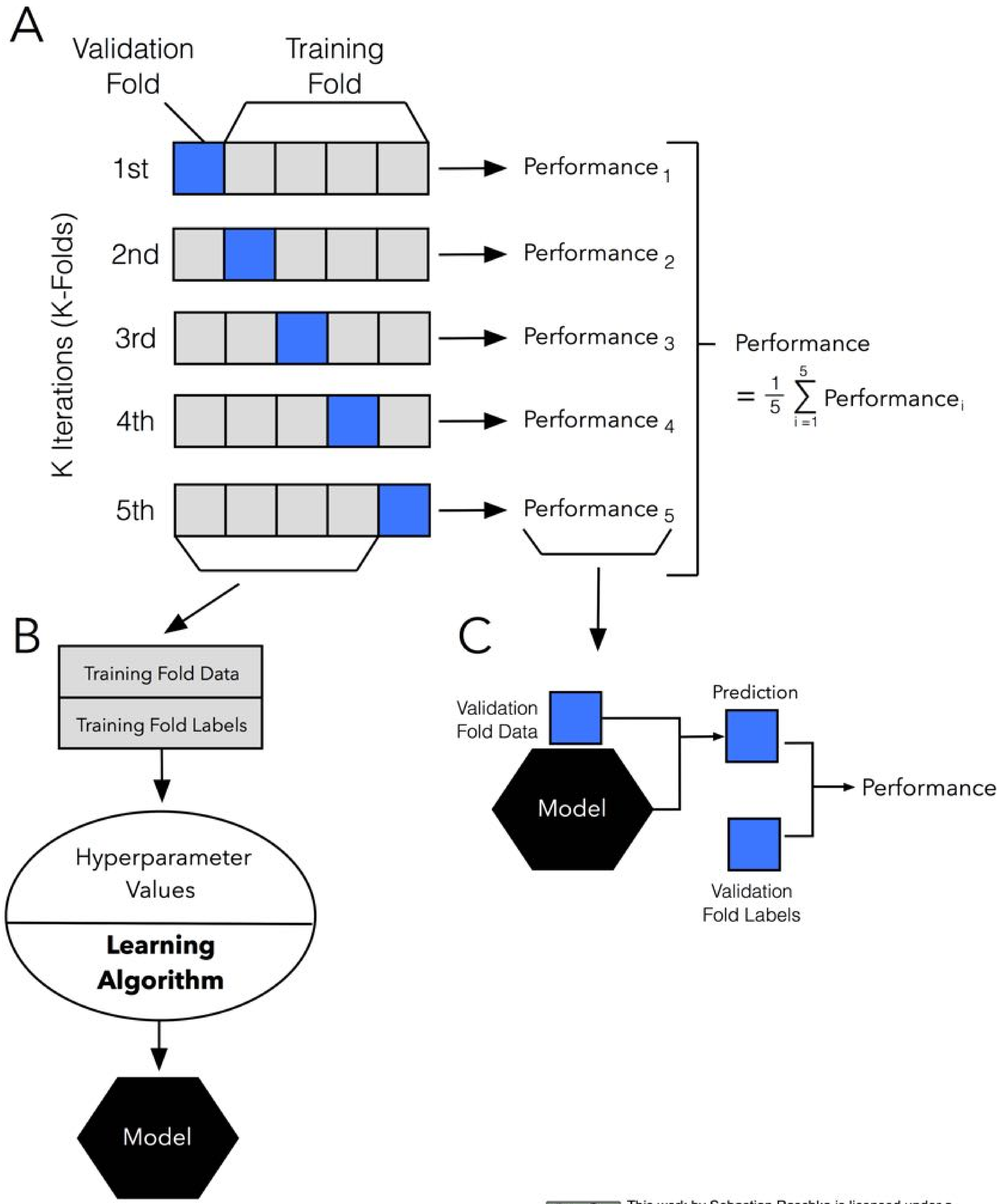
# Three-Way Holdout Method for Hyperparameter Tuning



# Hyperparameter Tuning Algorithms

- **Hyperparameter tuning is an optimization task**
  - **Grid Search**
  - **Random Search**
    - if at least 5% of the points on the grid yield a close-to-optimal solution, then random search with 60 trials will find that region with high probability
  - **Smart Hyperparameter Tuning**
    - **Not very parallelizable**
    - Many **automatic** techniques today (advanced), e.g.:
      - Derivative-free optimization
      - Bayesian optimization
      - Random forest smart tuning

# K-Fold Cross Validation






**All data for training and testing**

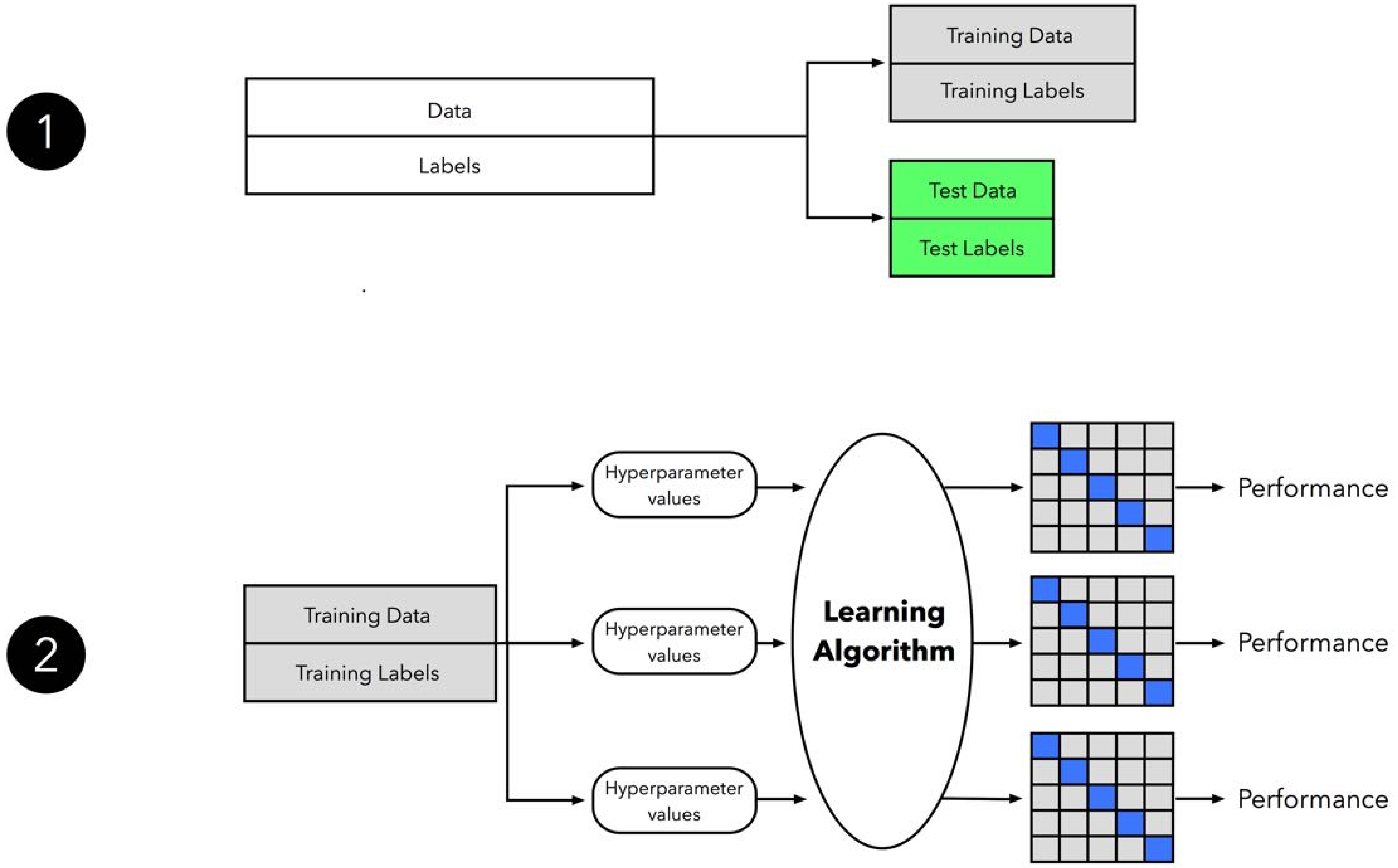
Test folds in k-fold cross-validation are **not overlapping**

In repeated holdout:

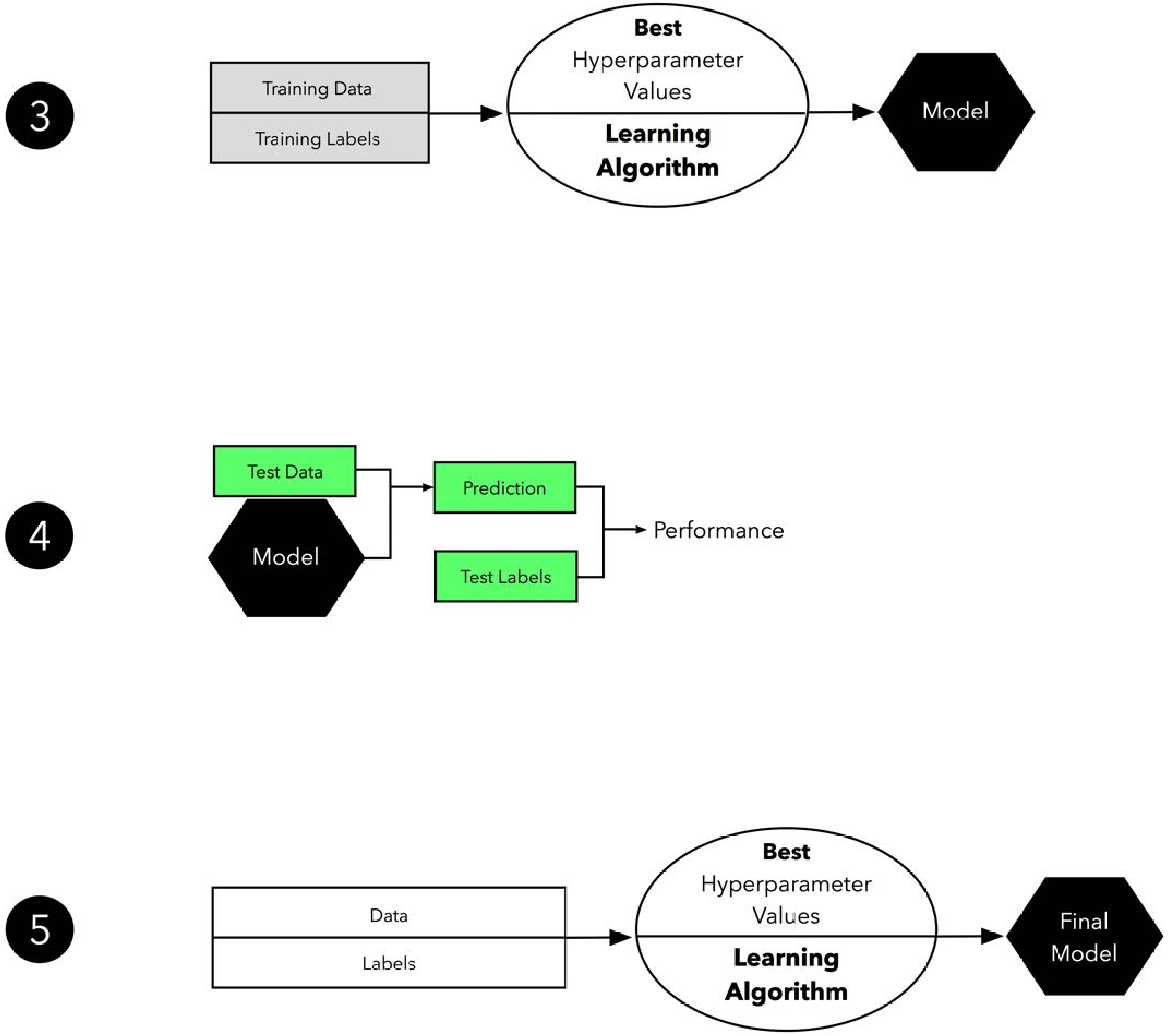
- performance estimates becomes dependent; it can be problematic for statistical comparisons

- some samples may never be part of the test set (k-fold cross validation guarantees that each sample is used for validation)

# Model Selection via K-fold Cross-Validation



# Model Selection via K-fold Cross-Validation



# Note (advanced)

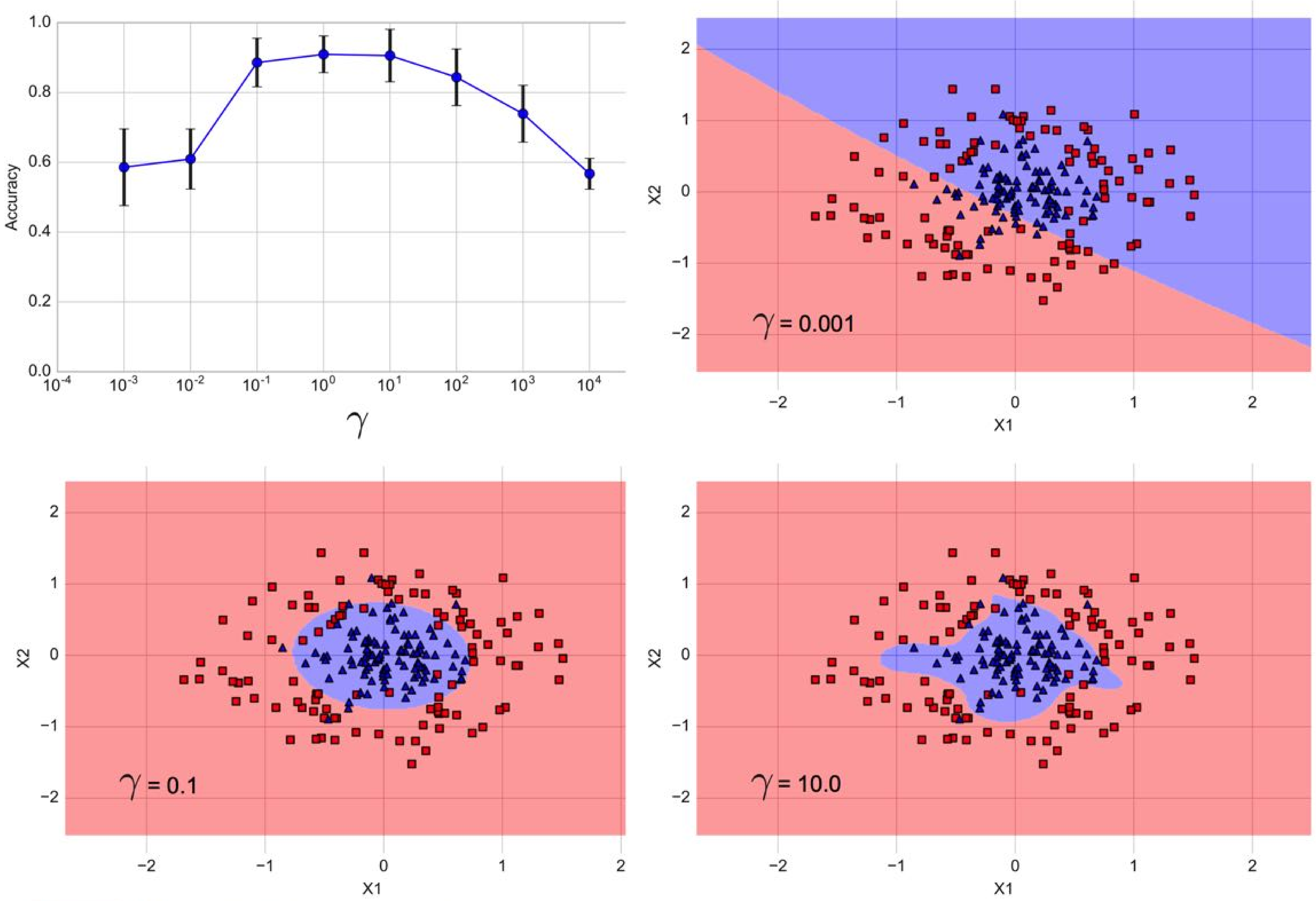
- If we normalize data or select features, we typically perform these operations **inside the k-fold cross-validation loop** in contrast to applying these steps to the whole dataset before splitting.
- Feature selection inside the cross-validation loop **reduces the bias through overfitting**, since we avoid peaking at the test data information during the training stage.
- However, feature selection inside the cross-validation loop may lead to an **overly pessimistic estimate**, since less data is available for training.
- For a more detailed discussion on this topic, read **"On Comparison of Feature Selection Algorithms"**

# Note (advanced)

- Law of Parsimony aka **Occam's Razor**
  - **Among competing hypotheses, the one with the fewest assumptions should be selected.**
- **One-standard error method** as follows:
- 1. Consider the numerically optimal estimate and its standard error.
- 2. Select the model whose performance is **within one standard error** of the value obtained in step 1 (Breiman and others, 1984).
- More details at this **link**

# Note

**γ=0.1** seems like a **good trade-off,** in fact  the performance of the corresponding model falls within **one standard error** of the best performing model with **γ=0** and **γ=10**



# Online Evaluation

- **A/B testing** (more **info**)
- **Multiarmed bandits** (more **info**)

# Metrics For Classification

- **Accuracy** (default in scikit-learn)
- **Per-Class Accuracy**
- **Confusion Matrix**
- **Precision**, **Recall**, **F1-score** and **Support**
  - (**classification_report** function in scikit-learn)
- **ROC Curves** and **AUC**

# Confusion Matrix

| | | ACTUAL | |
|---|---|---|---|
| | | POSITIVE | NEGATIVE |
| PREDICTED | POSITIVE | True Positive (TP) | False Positive (FP)<br>Type I Error |
| | NEGATIVE | False Negative (FN)<br>Type II Error | True Negative (TN) |

# Metrics For Classification

**sensitivity, recall, hit rate, or true positive rate (TPR)**

$$\mathrm{TPR} = \frac{\mathrm{TP}}{P} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

**specificity or true negative rate (TNR)**

$$\mathrm{TNR} = \frac{\mathrm{TN}}{N} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}$$

**precision**

$$\mathrm{PPV} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

# Metrics For Classification

**accuracy (ACC) = 1 - Error Rate**

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{P + N} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$$

**miss rate or false negative rate (FNR)**

$$\mathrm{FNR} = \frac{\mathrm{FN}}{P} = \frac{\mathrm{FN}}{\mathrm{FN} + \mathrm{TP}} = 1 - \mathrm{TPR}$$

**fall-out or false positive rate (FPR)**

$$\mathrm{FPR} = \frac{\mathrm{FP}}{N} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}} = 1 - \mathrm{TNR}$$

**F1 score is the harmonic mean of precision and sensitivity**

$$F_1 = 2 \cdot \frac{\mathrm{PPV} \cdot \mathrm{TPR}}{\mathrm{PPV} + \mathrm{TPR}} = \frac{2\mathrm{TP}}{2\mathrm{TP} + \mathrm{FP} + \mathrm{FN}}$$

# Metrics For Classification

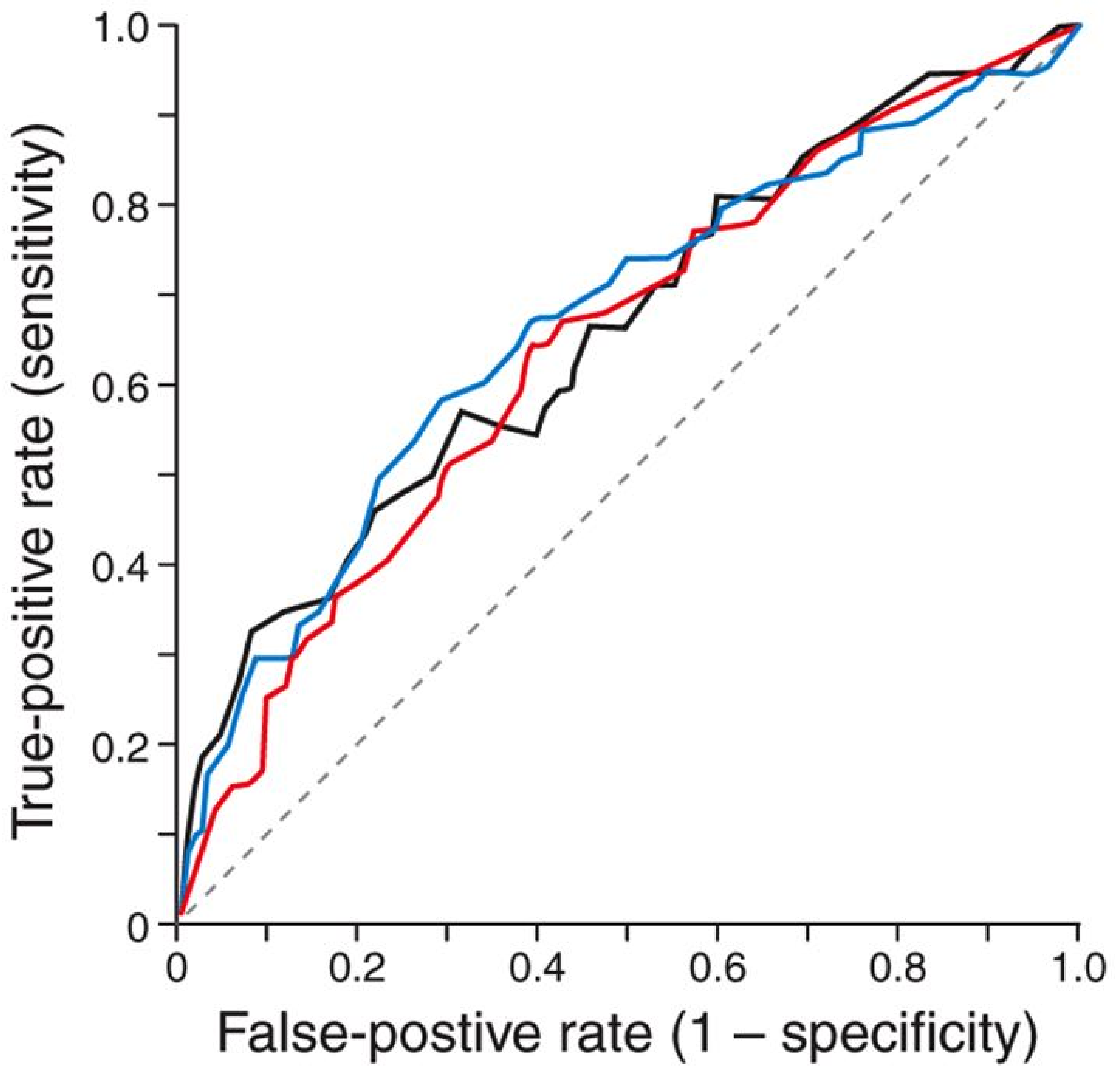
- The **support** is the number of occurrences of each class in the true data.
- The **per-class accuracy** computes the **accuracy** for each class independently
  - different from the average accuracy amongst all the classes
  - **useful in case of unbalanced classes** or when **the classifier performed very well/bad only for specific classes**

# ROC Curves

**Plot** the **True Positive Rate versus False Positive Rate** for various values of the threshold

Depicts relative **trade-offs** that a classifier makes between **benefits** (true positives) and **costs** (false positives)

**The diagonal is the baseline** – a random classifier

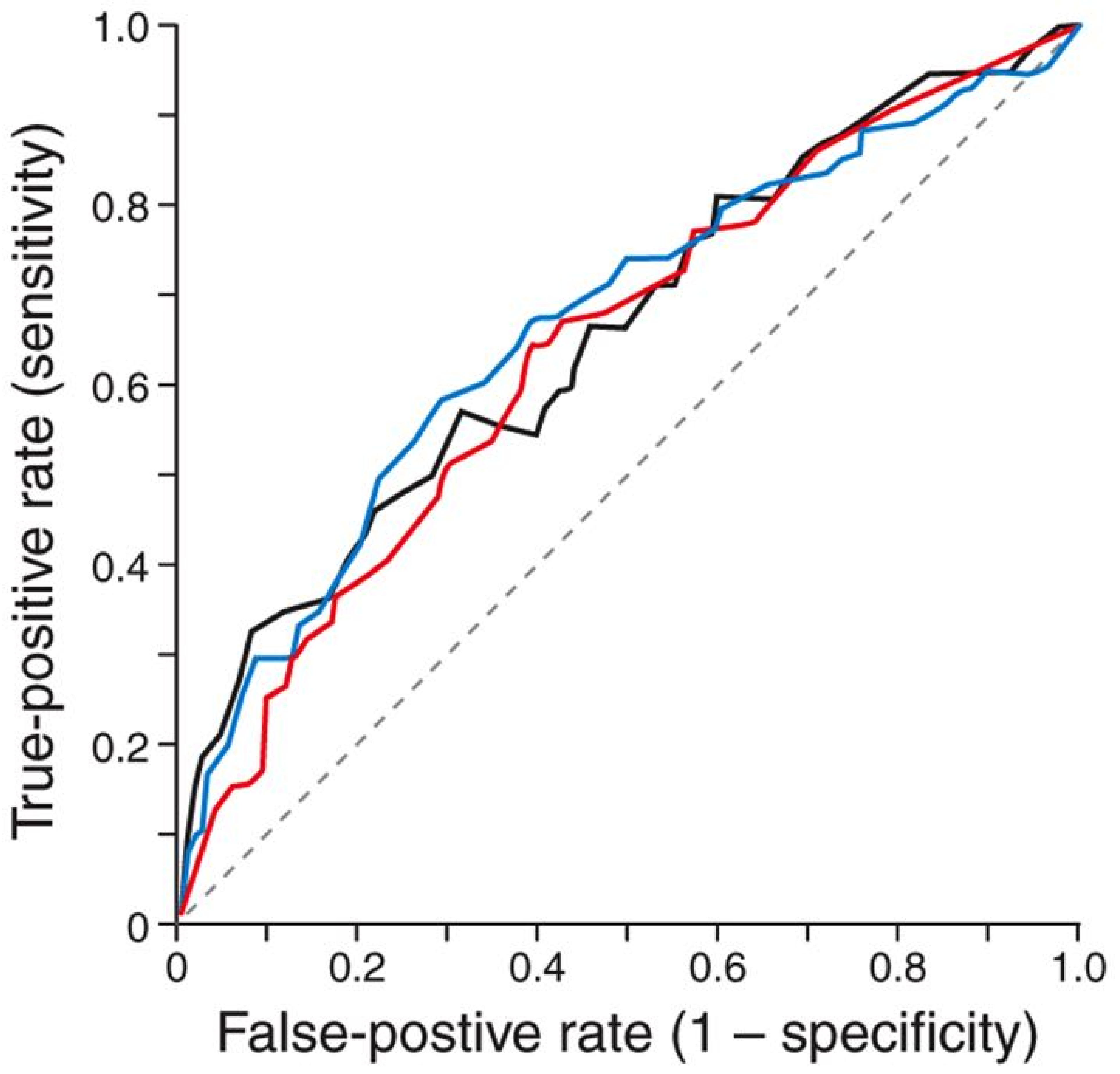


# ROC Curves

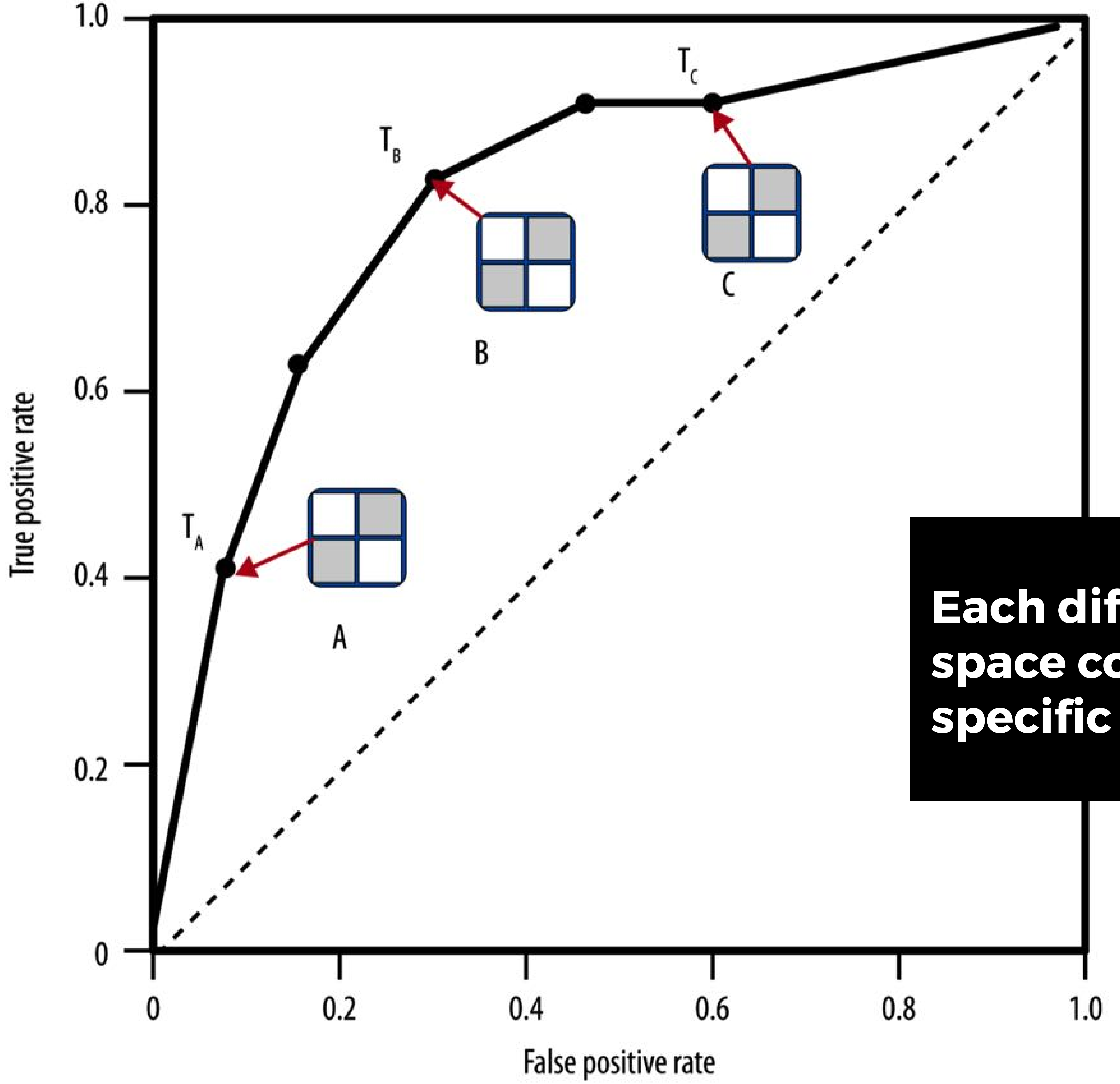
**Area under the ROC curve (AUC)**
- between 0 and 1

**Decouple classifier performance from the conditions under which the classifiers will be used**.

**Independent of the class proportions** as well as the costs and benefits.



# ROC Curves (advanced)




**Each different point in ROC space corresponds to a specific confusion matrix.**

# Questions?

@rschifan

schifane@di.unito.it

http://www.di.unito.it/~schifane