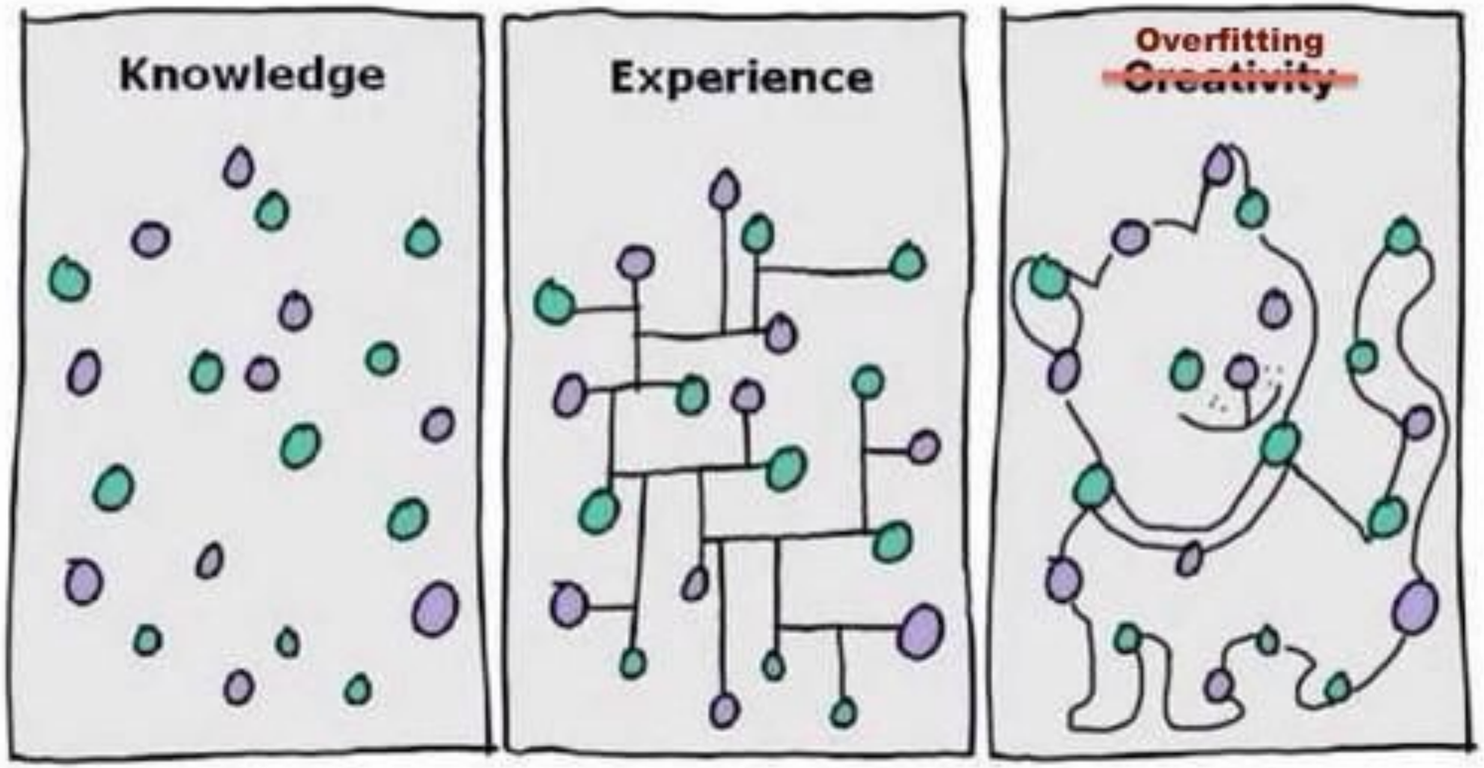
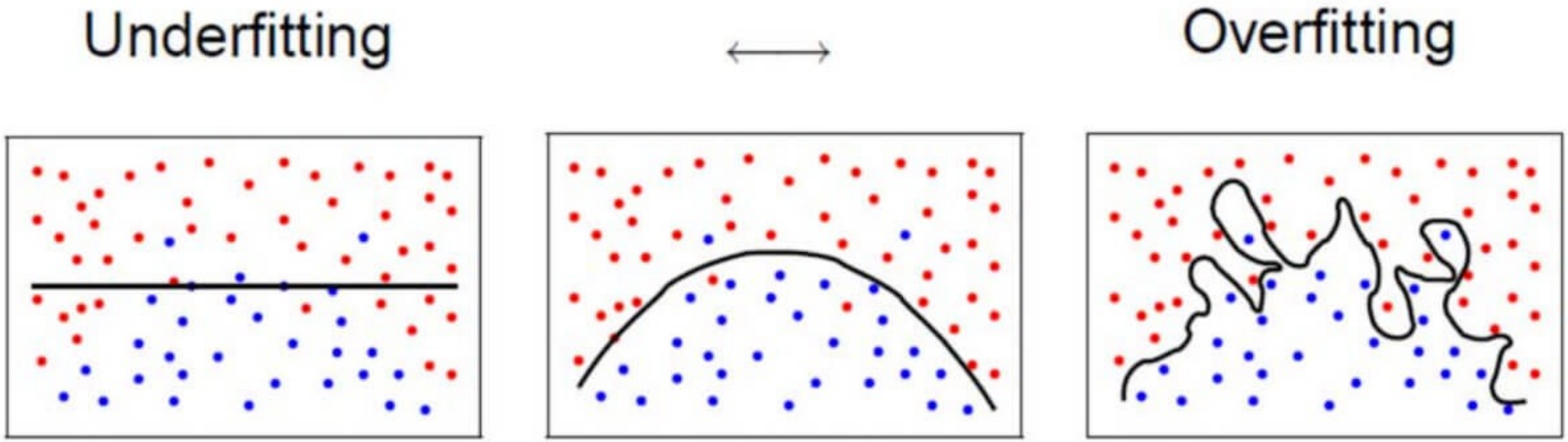

Knowledge
Experience
Overfitting
~~Creativity~~

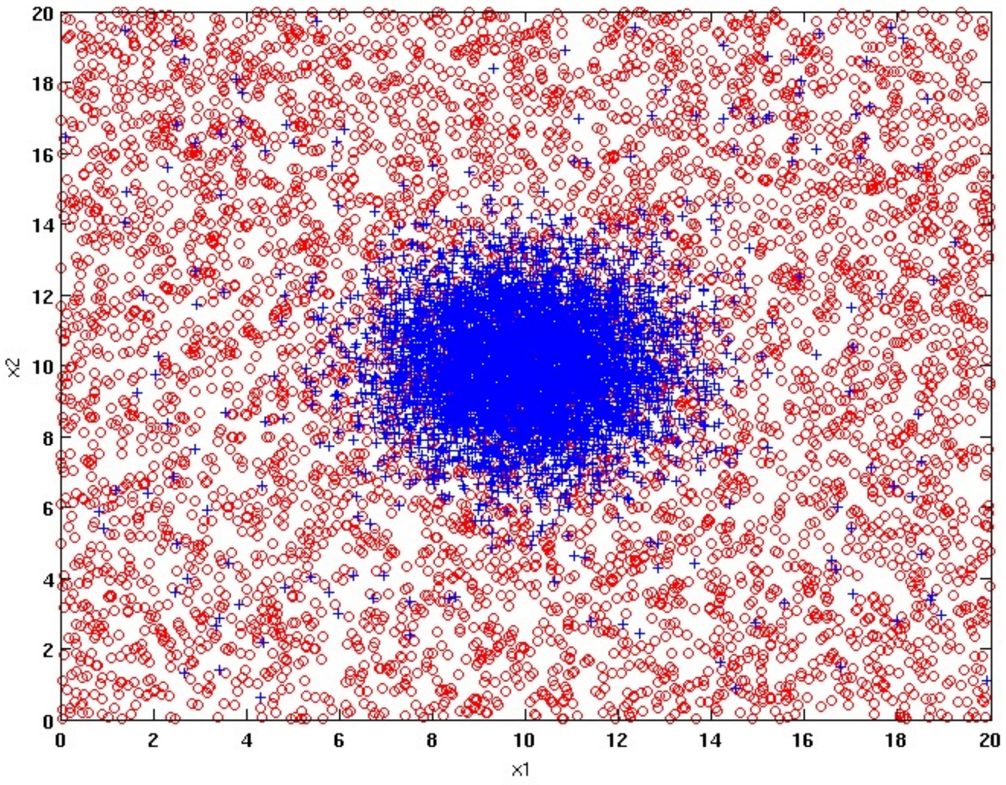
Cross-Validation & Overfitting

Underfitting and Overfitting Overview

- A model that fits the training data too well (has a low training error) may have a higher generalization error than a model with a higher training error



Example Data Set



Two class problem:

+ : 5400 instances

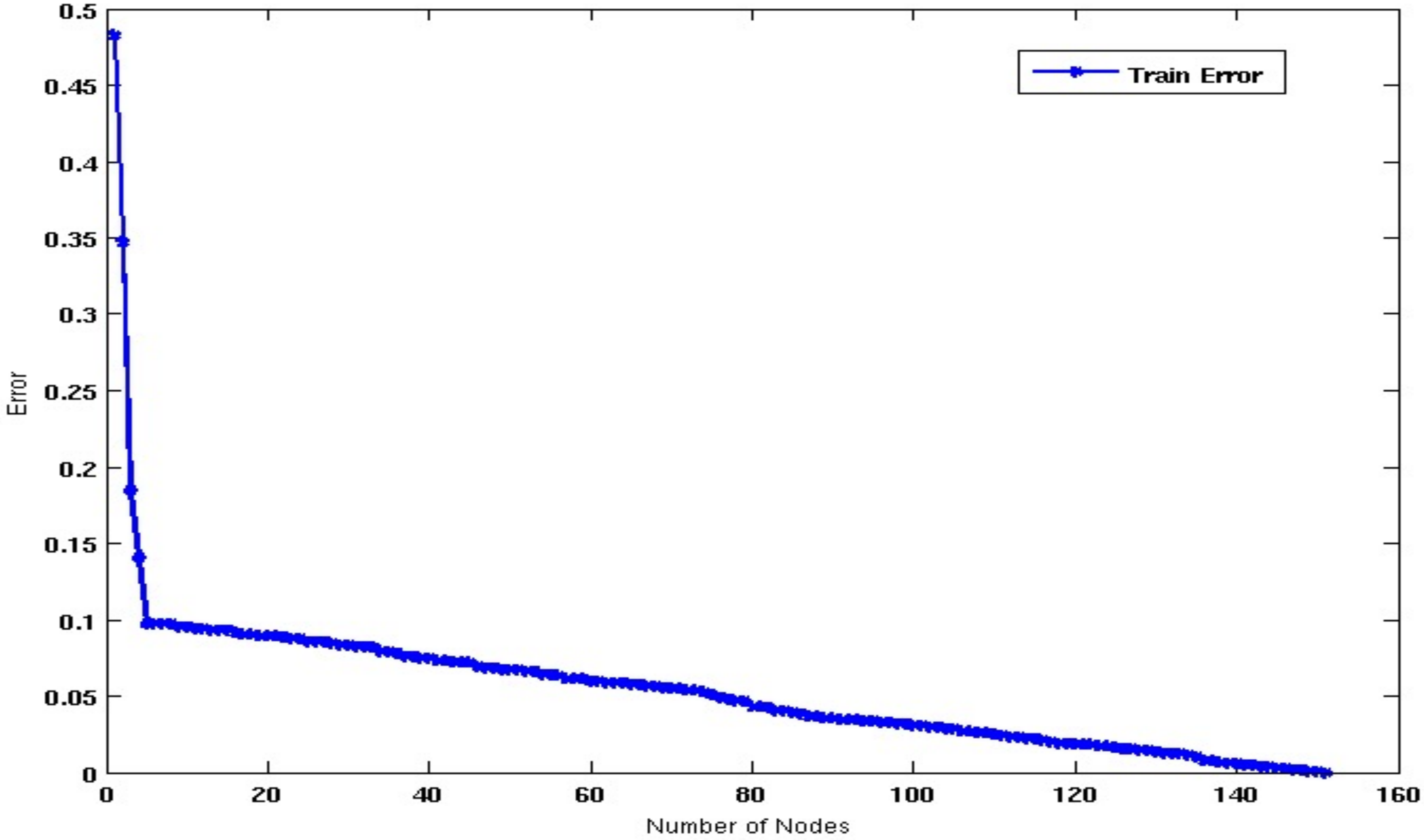
- **5000 instances generated from a Gaussian centered at (10,10)**
- **400 noisy instances added**

o : 5400 instances

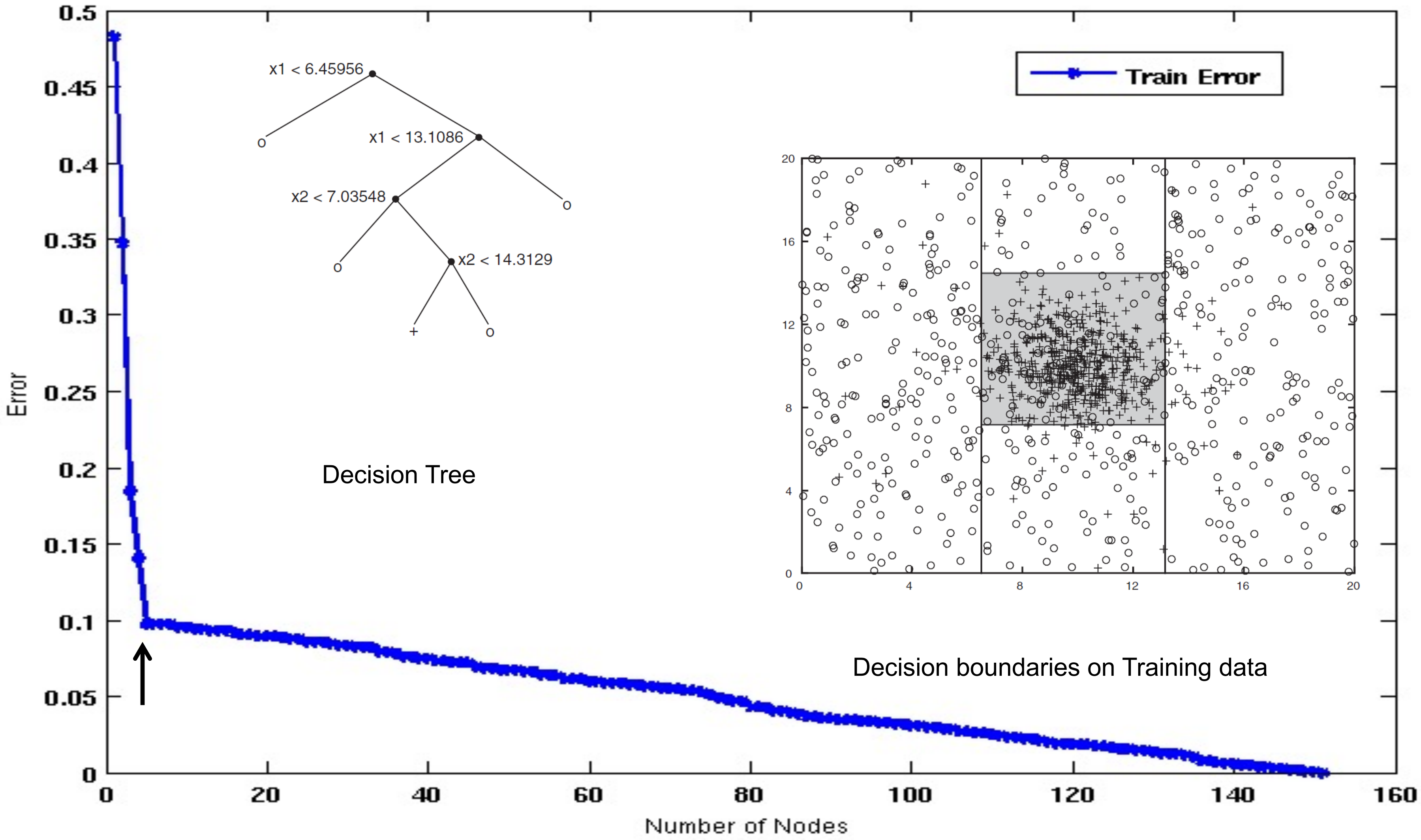
- **Generated from a uniform distribution**

10 % of the data used for training and 90% of the data used for testing

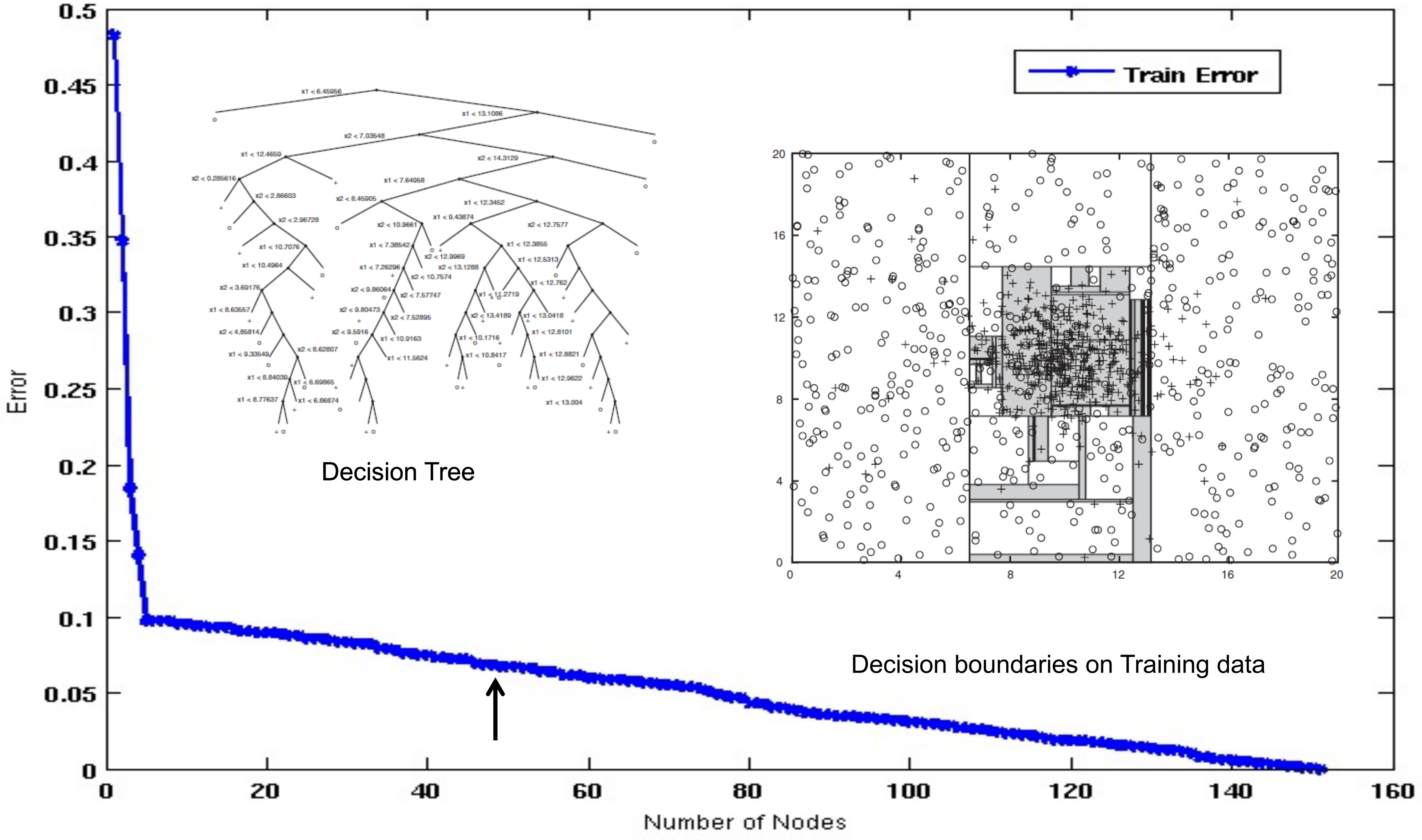
Learning Curve - Increasing number of nodes in Decision Trees



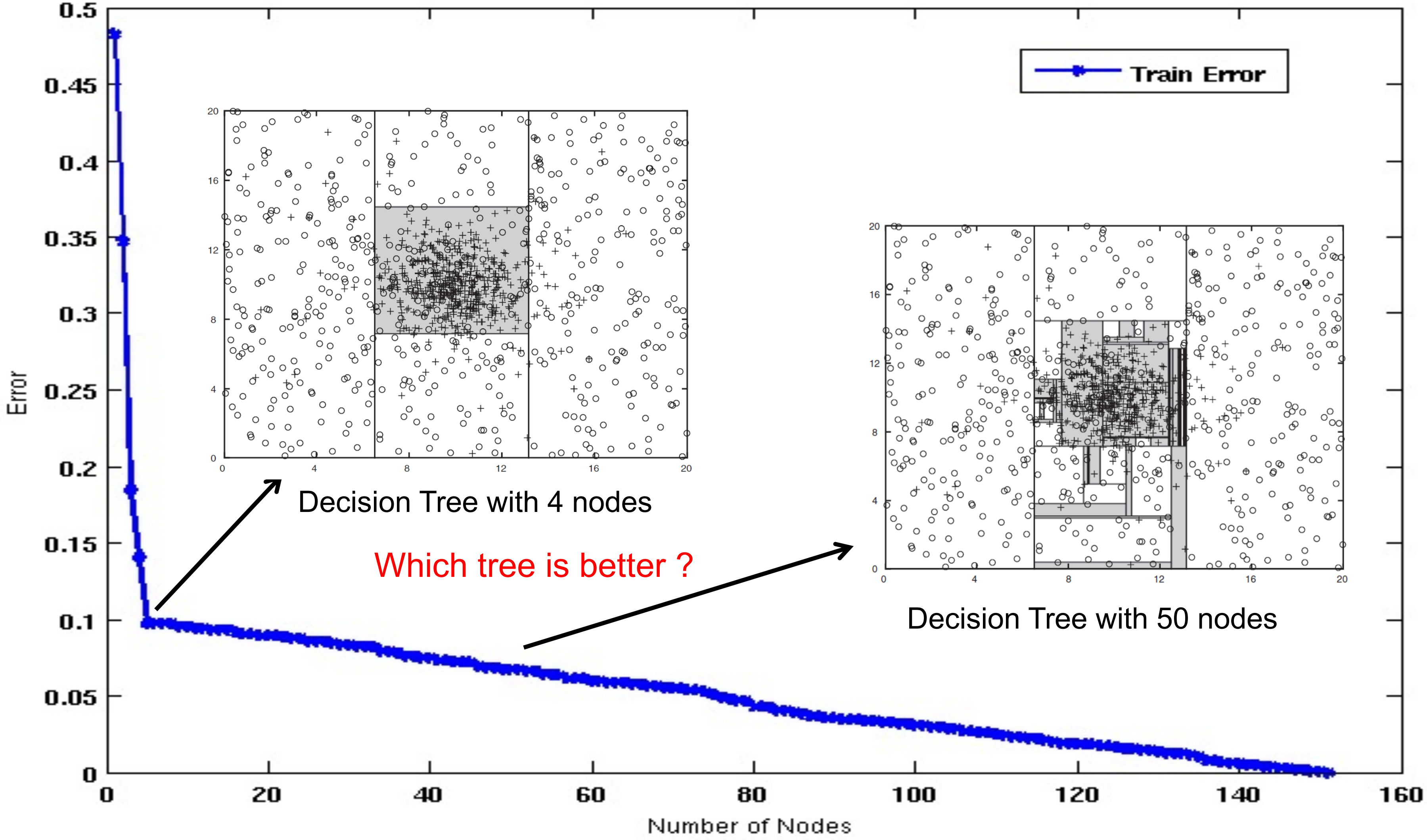
Decision Tree with 4 nodes



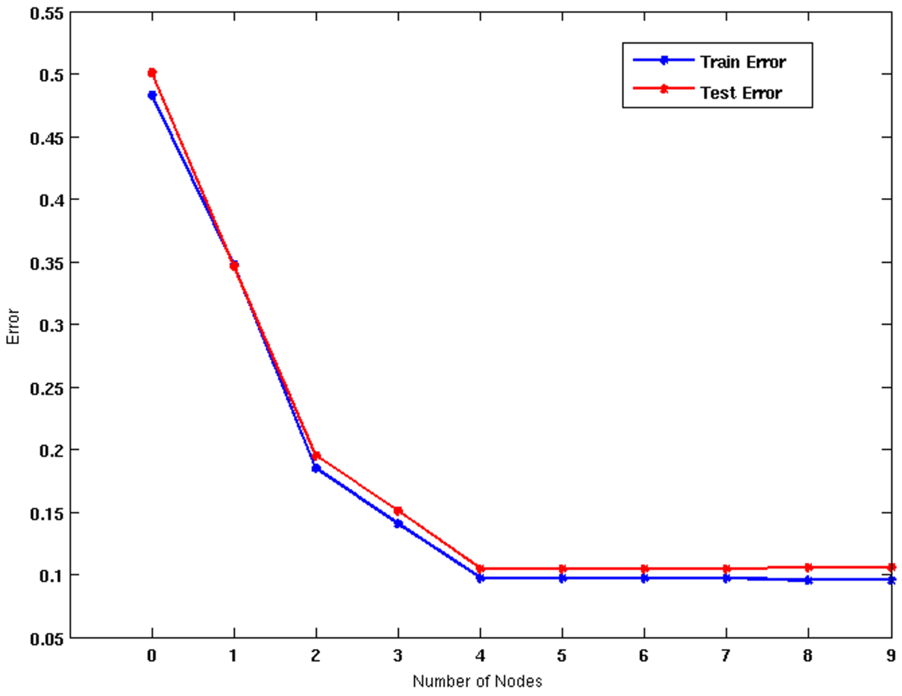
Decision Tree with 50 nodes

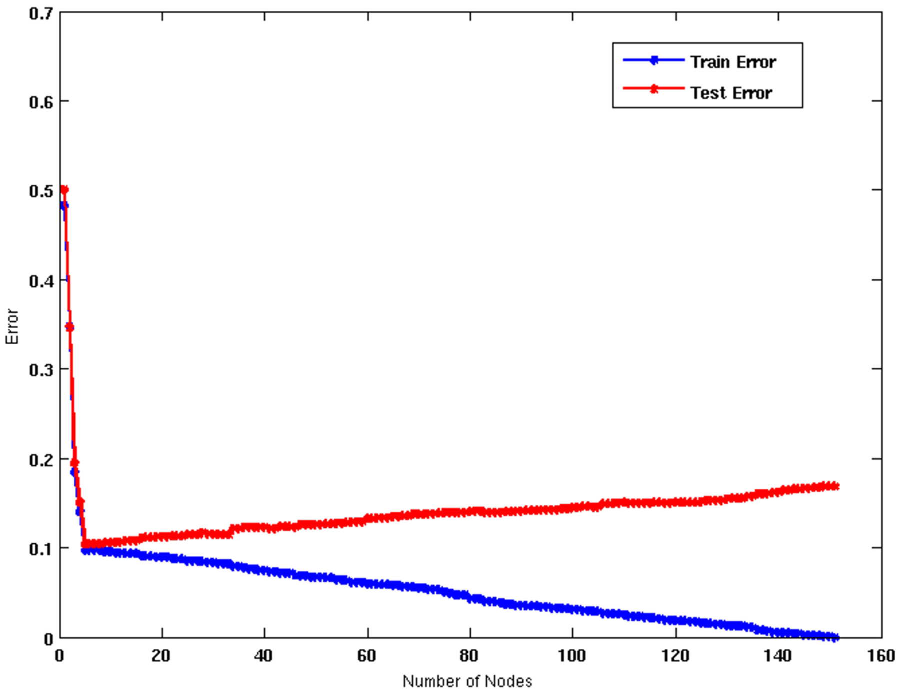


Which tree is better?



Model Overfitting



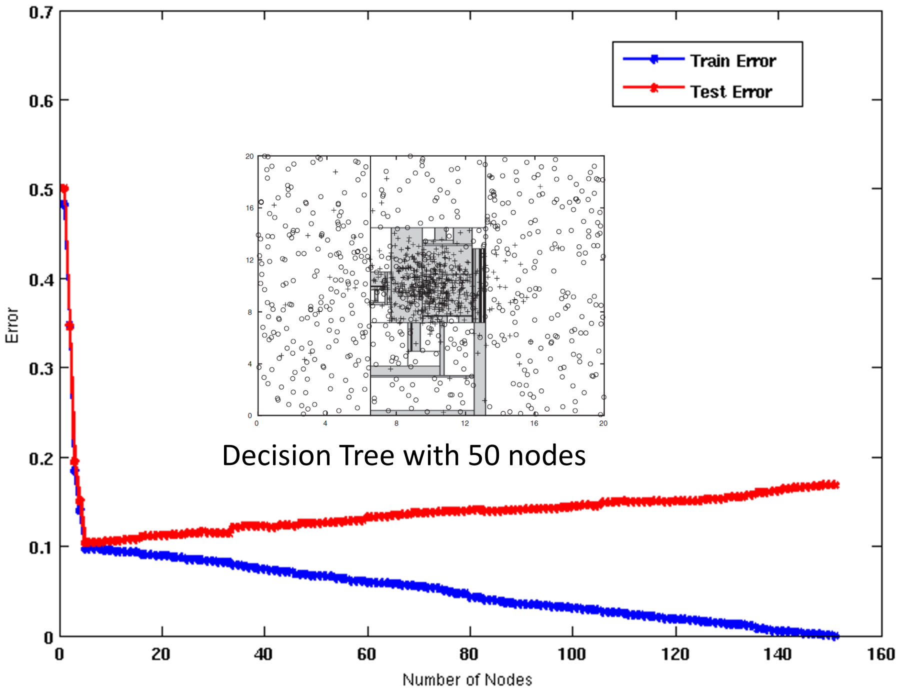


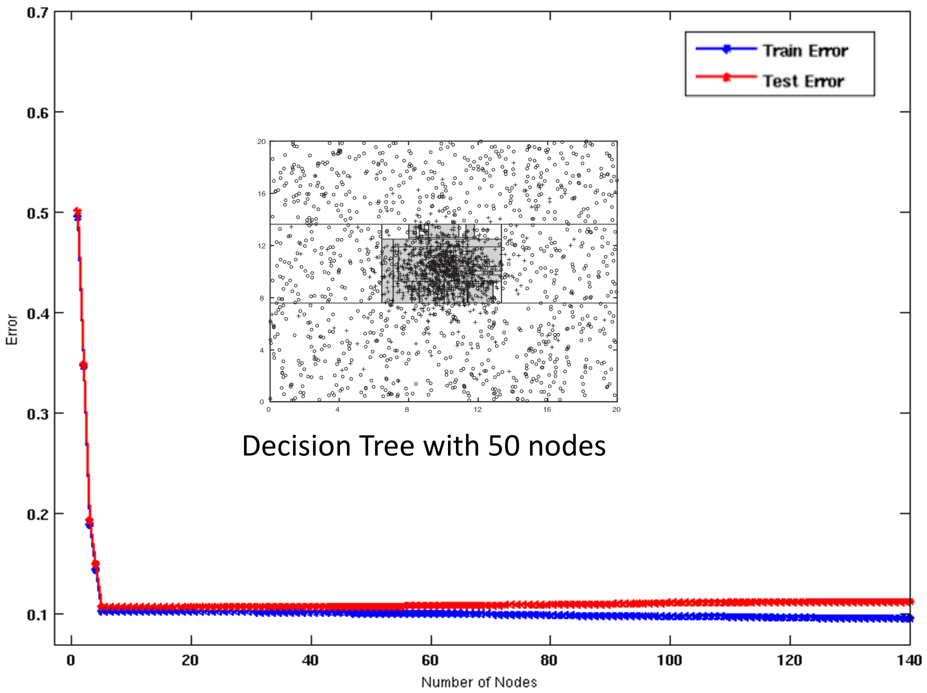
•As the model becomes more and more complex, test errors can start increasing even though training error may be decreasing

Underfitting: when model is too simple, both training and test errors are large

Overfitting: when model is too complex, training error is small but test error is large

Model Overfitting





Using twice the number of data instances

- Increasing the size of training data reduces the difference between training and testing errors at a given size of model

Reasons for Model Overfitting

- Limited Training Size

- High Model Complexity

Measuring Error

How to detect under/overfitting

Types of Error

Training Phase

- **Training Error:** the percent of misclassification errors on the training data set

Testing Phase

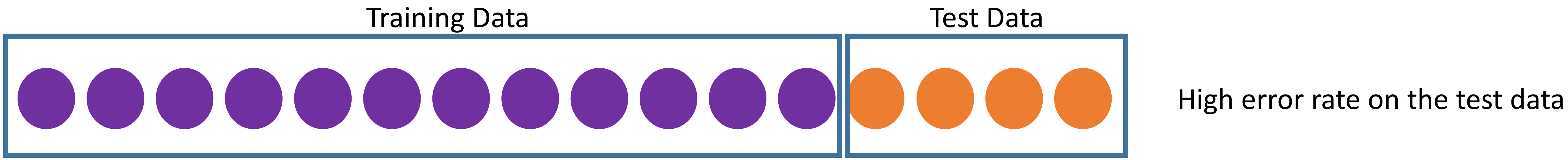
- **Test Error / Generalization Error:** the percent of misclassification errors on the test data set (previously unseen records)
- **Accuracy:** the percent of correct classifications on the test data set (100 – Generalization Error)

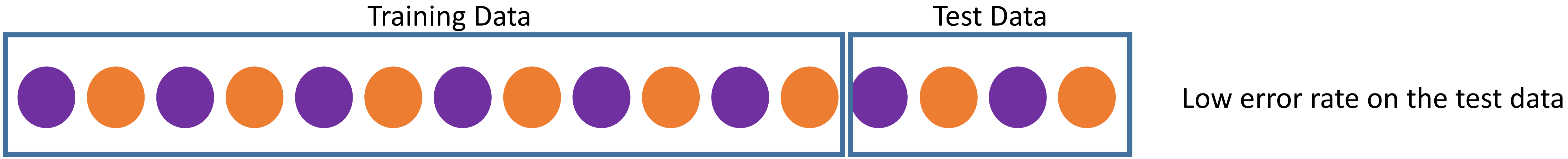
Partitioning Data to Predict Error

- Holdout Method: split data into training set and test set
- Build model on training set, evaluate generalization error on test set

Issues with Holdout Method

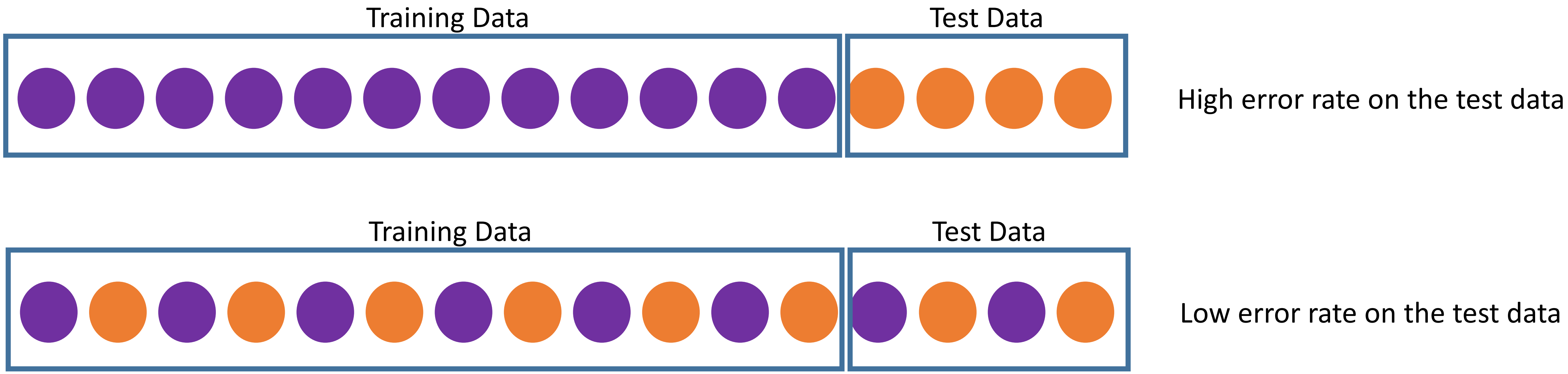
- Less data for training (and less data for testing)
- Some records never get trained on (and some never get tested on)
- A class overrepresented in one set will be underrepresented in the other set
- Varying performance of the model, depending on which records were held out for testing





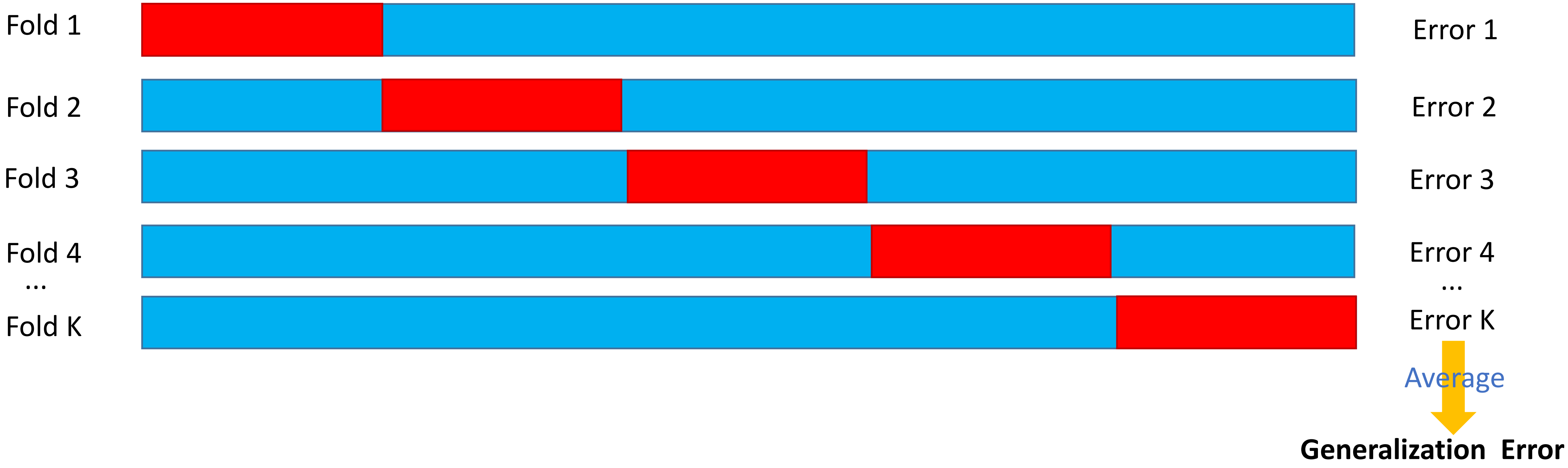
Issues with Holdout Method

- Less data for training (and less data for testing)
- Some records never get trained on (and some never get tested on)
- A class overrepresented in one set will be underrepresented in the other set
- Varying performance of the model, depending on which records were held out for testing

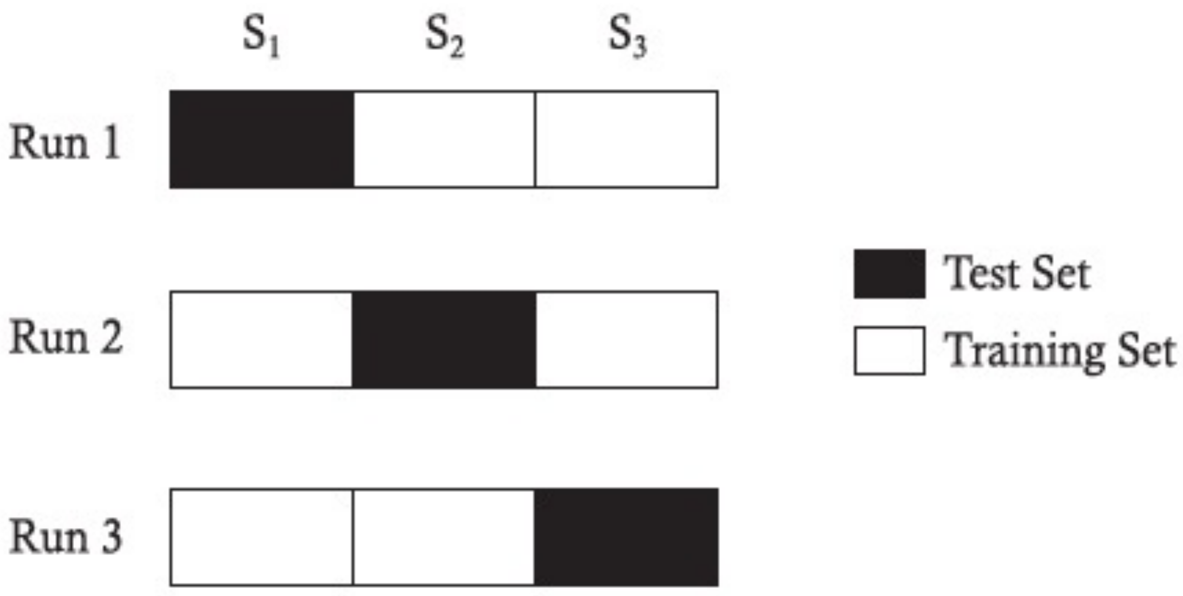


K-Fold Cross-Validation

- Split data into K folds
  - Model building (training) and error estimation (testing) is repeated K times
  - Each iteration, one of the folds is used for testing, the rest are used for training.
  - Get an error estimate from each fold. Average them to establish a final estimate of generalization error.



Example: Three Fold Cross Validation



Inside the Cross-Validation Loop

- Cross-validation is used to assess the performance of the process
- It is critical that the test data remains totally unseen and is not used for <u>any</u> part of the model creation. (That would be cheating! Also called: data leakage.)

Fold 1

Error 1

1. **On the training set only:** perform feature scaling (normalization/standardization), mathematical or algorithmic feature selection, dimensionality reduction (PCA, etc.)
The test data should be <u>unseen</u> data, so no decisions should be made based on it!
2. Build the model using the training data.

3. Transform the test data into the feature space that was found on the training data. (Scale using the values found on the training data, reduce dimensionality using the PCA found on the training data, etc.)

4. Make a prediction using the model. Evaluate the prediction.

Fold 2

Error 2

…

Building the Final Model

- A good solution is found (you got a high accuracy) - now what…?
- The cross-validation created K different models of the data – which one do you use?
- None of them! Cross-validation was necessary to evaluate the process. Once the process is deemed good, you run the whole process on your entire dataset to make a final model.
    - Perform scaling, feature selection, dimensionality reduction, etc. on the entire dataset, and train a final model on the entire dataset (there is no test set).
- This final model will be used in the real-world to make predictions on new data.

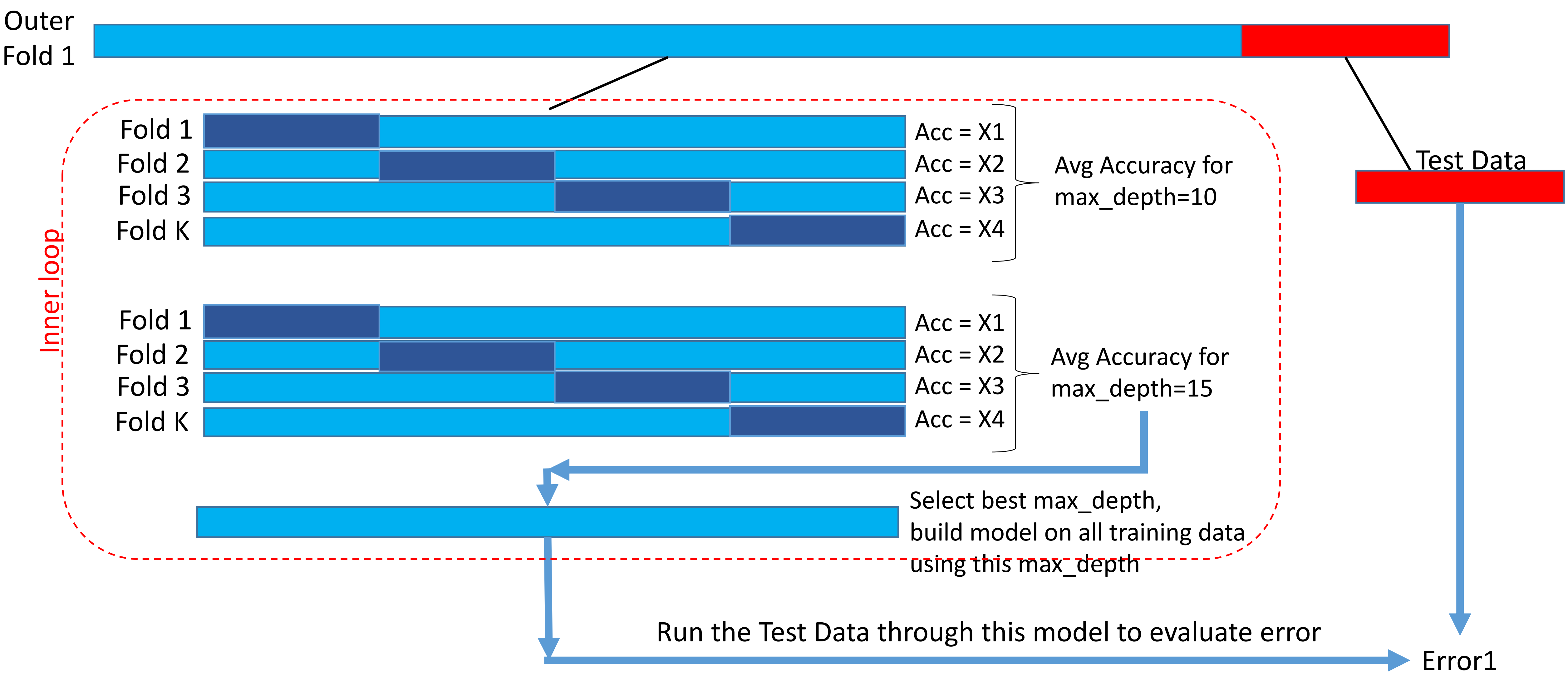
Finding the best Hyperparameters

Nested Cross Validation

Hyperparameters

- From the book:
- “Hyper-parameters are parameters of learning algorithms that need to be determined before learning the classification model”
- Look at the documentation for scikit-learn decision tree:
  - https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

Nested Cross Validation



Choose the Hyperparameter

- For each fold of your outer loop, you calculated the best hyperparameter, so you have *k* votes
  - In the best-case scenario, all folds vote for the same hyperparameter value
  - In the worst-case scenario, all folds vote for a different hyperparameter and more study is required.

- Use the consensus (most votes) to select the best hyperparameter to train the final model.