# A Practical Introduction to Data Science

Part 4

# Advanced Supervised Learning

Gergely Zsombor Haász

haasz.zsombi@gmail.com

# Course Agenda

# Advanced Supervised Learning

- Tree Ensembles
- Overfitting and Generalization
- Hyperparameter Optimization

## Tree Ensembles

# Random Forest

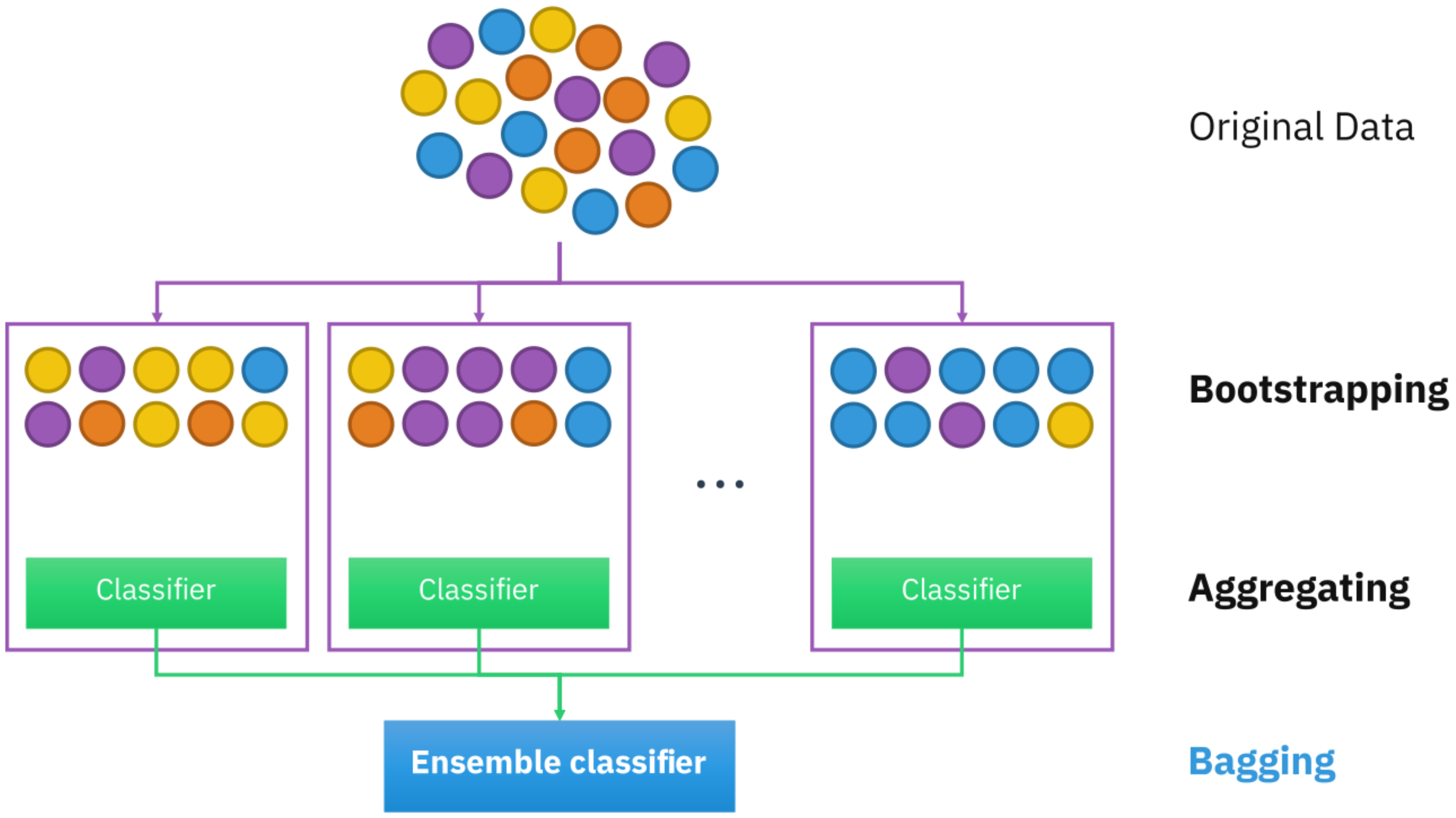
- Base learner: Decision Tree
- Bagging + random subspace method

**Advantages:**

- More accurate
- More robust

**Disadvantages:**

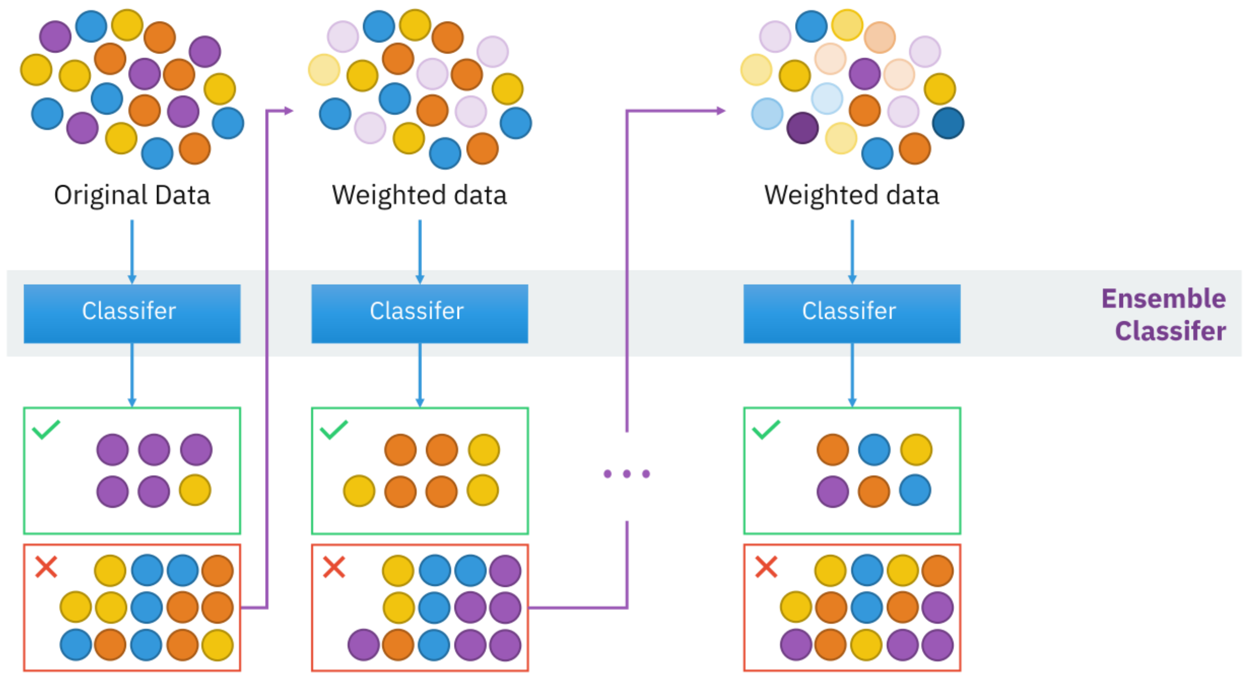
- Less interpretable
- More computation



Source: Bootstrap aggregating – Wikipedia

# Adaptive Boosting (AdaBoost)

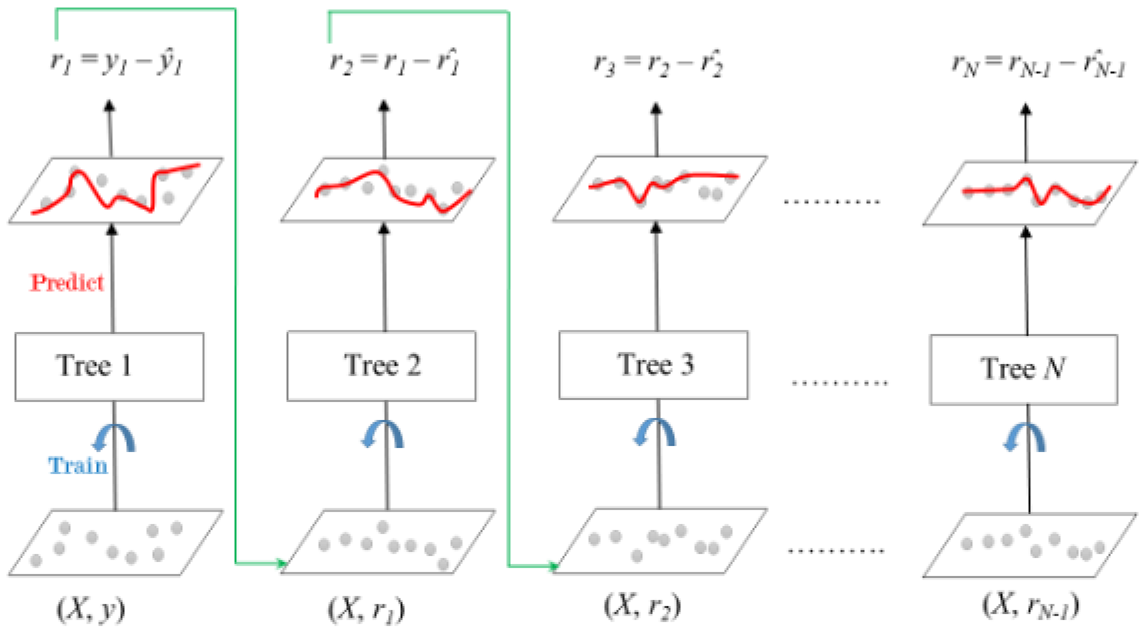
- Boosting: Sequentially trains classifiers
- Weight Adjustment: Emphasizes misclassified instances.
- Final Prediction: weighted sum of the predictions of all weak classifiers.



Source: Boosting (machine learning) – Wikipedia

# Gradient Boosting

- Boosting: Sequentially trains classifiers
- Target: the residuals of the previous tree
- Final Prediction: sum of the initial prediction and the contributions of all the weak learners, each multiplied by the learning rate



Source: Gradient Boosting in ML – GeeksforGeeks

# Wide Range of Boosting Algorithms

Same main idea, different details and implementations:

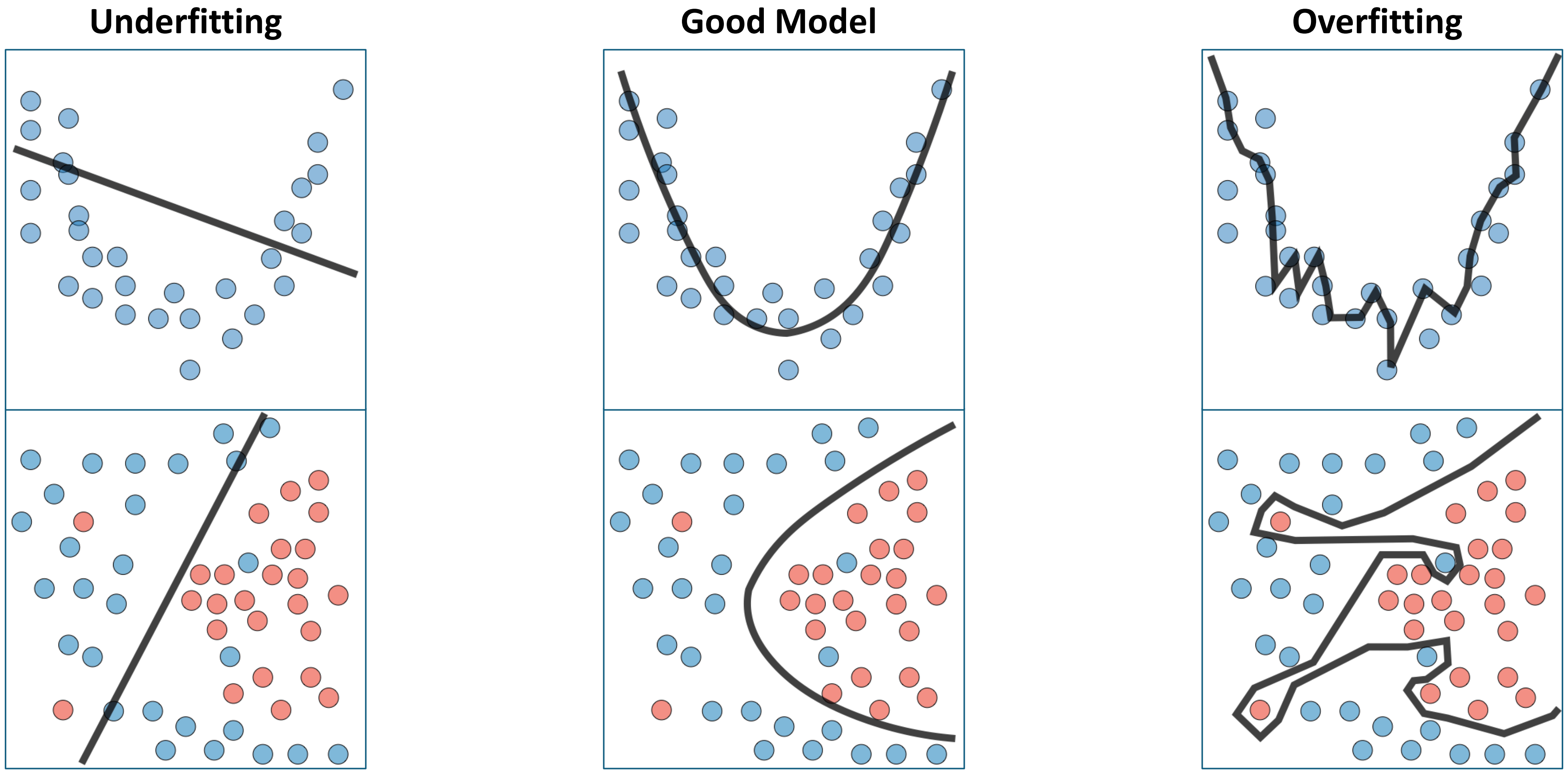
- Speed and memory usage
- Handling missing values
- Handling categorical values
- Regularization
- Tree pruning
- Parallelism
- Community

Python packages:

- sklearn.ensemble
- CatBoost
- LightGBM
- XGBoost

## Overfitting and Generalization

# Overfitting and Generalization



Source: CS 229 - Machine Learning Tips and Tricks Cheatsheet

# Overfitting and Generalization

| | | |
|---|---|---|
| **Sample** | Small | Big |
| **Features** | Few | Many |
| **Model** | Simple | Complex |

Possible solutions:

- Collect more data (observations and/or features)
- Enhance data quality (outliers, noise)
- Feature selection, engineering and regularization
- Control model complexity by tuning hyperparameters
- Better evaluation with cross-validation

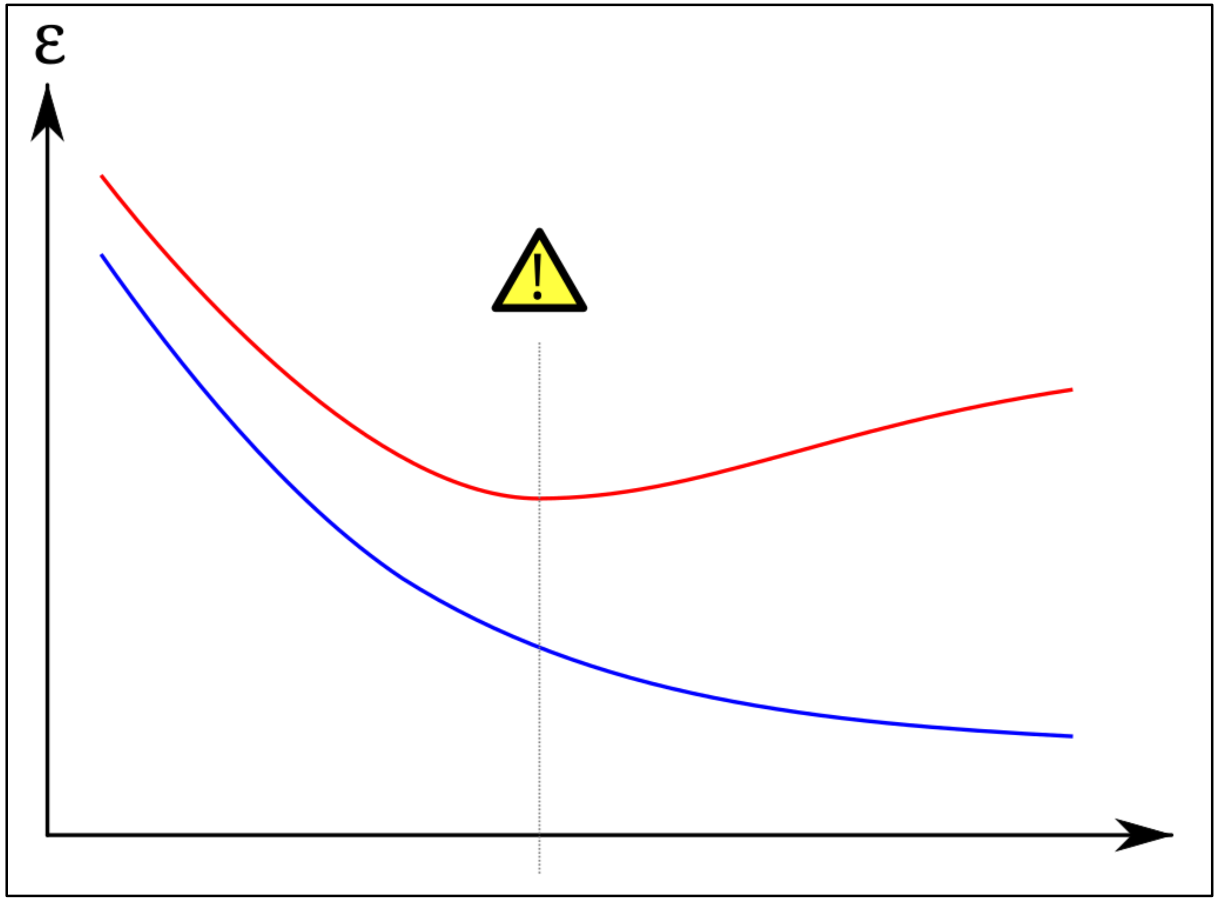
# Overfitting and Generalization

Reasons:

- Learning noise
- Memorization
- Sampling issue

Steps:

1. Build a simple model
2. Try to overfit
3. Tune the hyperparameters



Source: Overfitting – Wikipedia

# Feature Selection

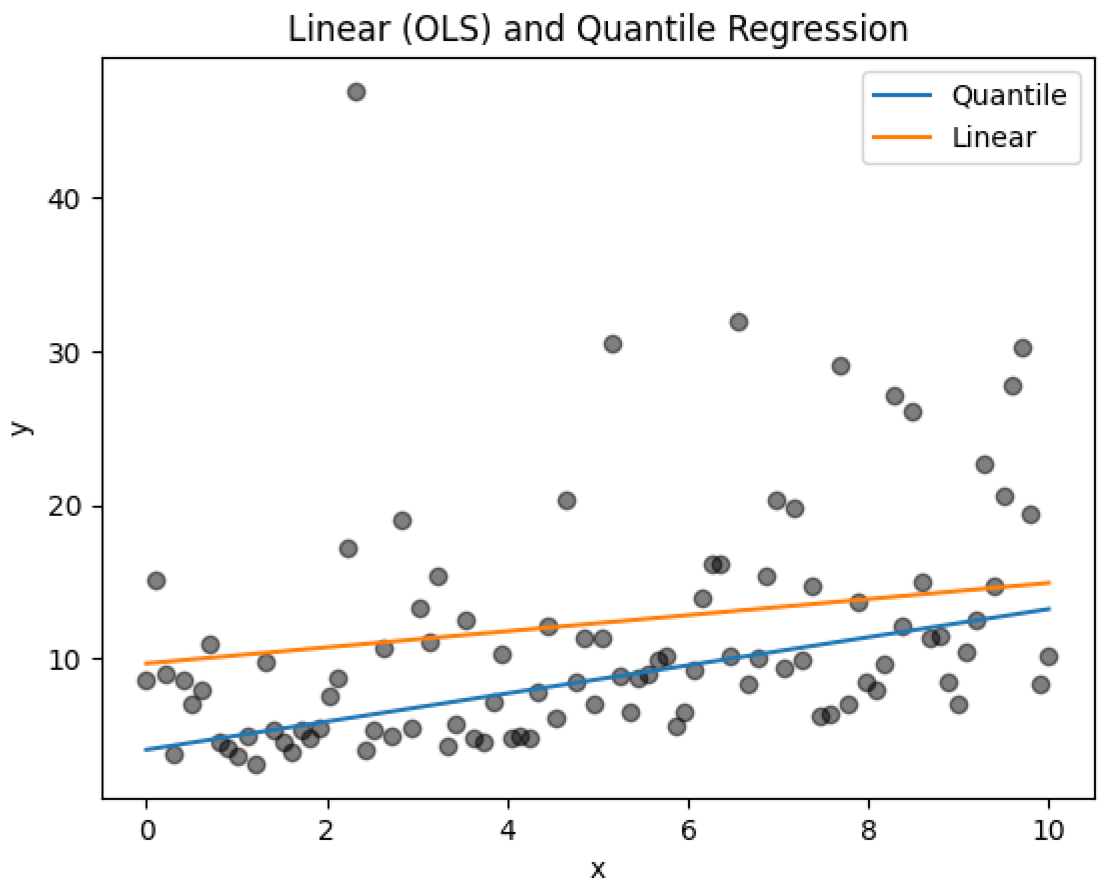
1. Initial feature selection
    - Business meaning
    - Target leakage
    - Predictive power
    - Multicollinearity
    - Stability

2. Training-time feature selection
    - Stepwise feature selection
    - L1 Regularization (LASSO regression)
    - Feature importances

# Quantile Regression

- We can change the loss function from MSE to MAE for some algorithms
- Estimates the median instead of the mean
- Less sensitive to outliers

| | RMSE | MAE |
|---|---|---|
| OLS Linear Regression | **7.2** | 5.4 |
| Quantile Regression | 7.5 | **4.6** |



# $L_1$ – $L_2$ Regularization

Lasso (L1) $Loss_{L_1} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \alpha \sum_{i=1}^{p}|\beta_i|$

Ridge (L2) $Loss_{L_2} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \alpha \sum_{i=1}^{p}\beta_i^2$

Elastic net $Loss_{L_1+L_2} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \alpha_1 \sum_{i=1}^{p}|\beta_i| + \alpha_2 \sum_{i=1}^{p}\beta_i^2$

- The error on the train sample will be higher with L1 and L2 penalties, but we might avoid overfitting and reduce the test error
- Lasso is a technique for feature selection as well
- Feature standardization is beneficial
- The same can be applied for logistic regression, neural networks and XGBoost

## Hyperparameter Optimization

## Hyperparameter Optimization

**Hyperparameters:** Parameters that are not learned from the data but set prior to the training process

**Hyperparameter Optimization:** The process of finding the best set of hyperparameters for a machine learning model to improve its performance.

**Importance:** The wrong hyperparameters can cause **underfitting** or **overfitting.**

# Hyperparameter Optimization

kNN:
- Number of neighbors (k)
- Weights
- Distance metric

SVM:
- C (regularization parameter)
- Kernel
- Kernel parameters (e.g. gamma)

Decision Tree:
- Max depth
- Min samples on leafs
- Max features

Random Forest:
- Tree parameters
- Number of estimators

Gradient Boosting:
- Tree parameters
- Number of estimators
- Learning rate
- Subsample

*…and many more*

# Hyperparameter Optimization

**Grid Search:** exhaustive search over a specified parameter grid

- Pros: Simple and comprehensive
- Cons: Computationally expensive

**Random Search:** randomly samples hyperparameters from a specified distribution

- Pros: More efficient than grid search
- Cons: May miss optimal combinations

**Bayesian Optimization:** uses probabilistic models to predict the performance of hyperparameters.

- Pros: Efficient and can find better hyperparameters.
- Cons: More complex to implement.

# Cross-Validation

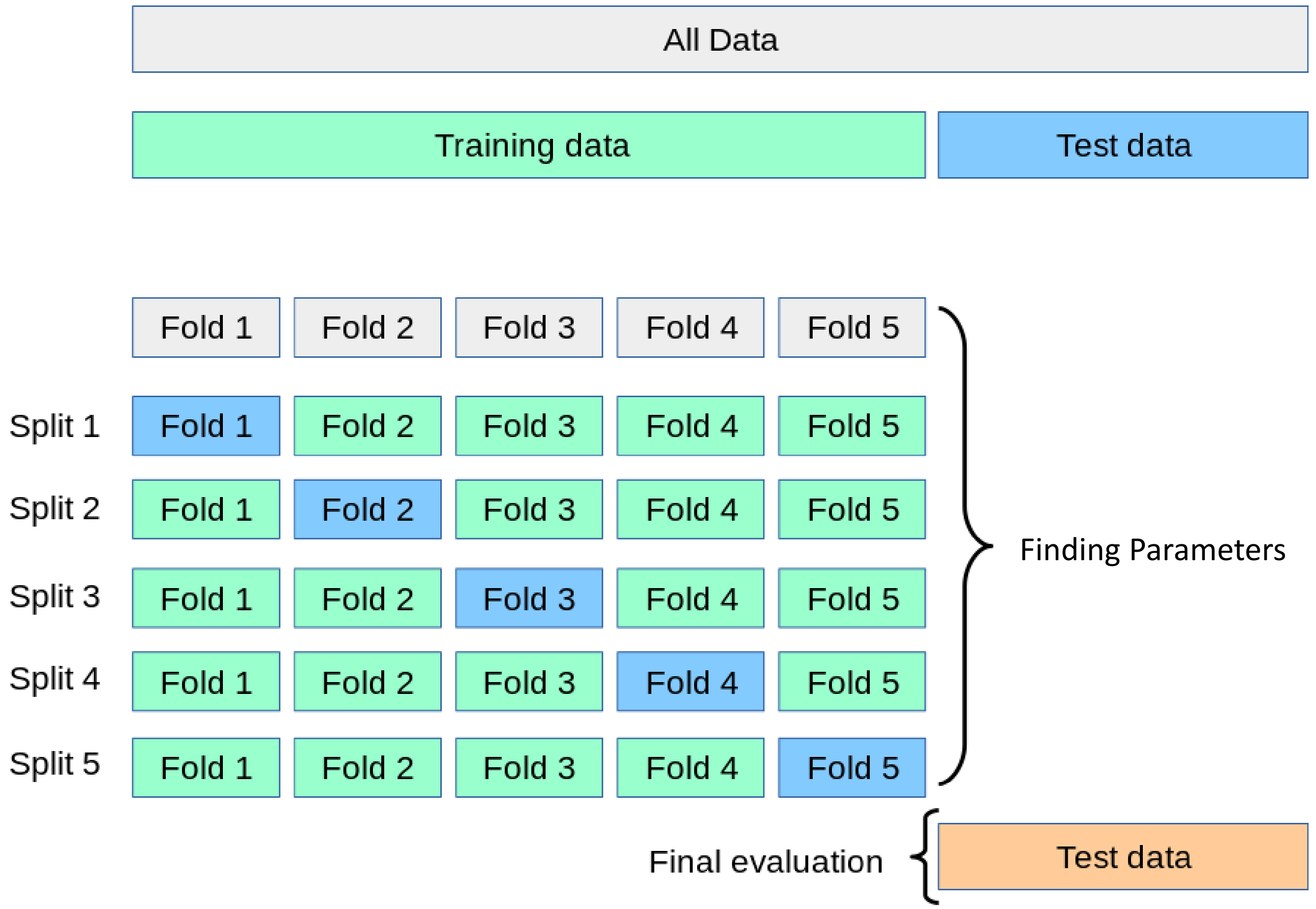
*More reliable evaluation, prevents overfitting to the validation sample*

**K-Fold CV**:

- data is divided into k (5-10) equal-sized folds.
- model is trained on k-1 folds and validated on the remaining fold.
- this is repeated k times, with each fold used exactly once as the validation set.

**Repeated K-Fold CV:** Perform multiple runs of K-Fold CV

**Leave-One-Out CV**: Each data point is used once as a validation set, and the model is trained on the remaining data



Source: 3.1. Cross-validation: evaluating estimator performance — scikit-learn 1.6.1 documentation

## Model Selection – No free lunch theorem

*“All models are wrong, but some are useful”*

Things to consider:

- Sample size, number of features, problem complexity
- Training/Inference runtime, memory, compute
- Robustness, extrapolation capabilities
- Interpretability

*Start simple. Experiment. Use domain knowledge.*

## Overfitting and Model Tuning Summary

1. Sufficient Sample Size
2. Data Completeness (having the right features)
3. Data Quality (outliers, noise)
4. Feature Selection (no irrelevant features)
5. Feature Engineering (helping the model with domain knowledge)
6. Model Selection (according to the data and the problem)
7. Loss function and evaluation metric according to the business problem
8. Regularization
9. Hyperparameter Search
10. Reliable Evaluation with Cross-Validation

## Thank you for your attention!

Your feedback would be much appreciated:

## Any Questions?

Gergely Zsombor Haász

haasz.zsombi@gmail.com