Linear regression

Vladislav Goncharenko
ML researcher

girafe ai

MSU, spring 2024

Recap

Lecture 1:
Intro to ML
Naïve Bayes

1. ML and AI overview
2. What is Machine Learning
3. Thesaurus and notation
4. Datasets
5. Exploratory data analysis
6. Maximum Likelihood Estimation
7. Some Machine Learning problems
8. Naïve Bayes classifier
9. Machine learning libraries

Outline

1. Regression model
2. Linear models overview
3. Linear regression
4. Gauss-Markov theorem
5. Regularizations
6. Model validation and evaluation

Regression model

girafe
ai

01

Regression model

Can be written if form

$$Y = f(X) + \varepsilon$$

or equivalently

$$\mathbb{E}(Y|X) = f(X)$$

here we split all the signal measured (Y) into parts containing:

1. Expectation with zero Variance (which we want to predict)
2. Variance with zero Expectation (aka noise)

Regression model assumptions

1. The sample is representative of the population at large
2. The independent variables are measured with no error
3. Deviations from the model have an expected value of zero, conditional on covariates: $E(ei|Xi)=0$
4. The variance of the residuals ei is constant across observations (homoscedasticity)
5. The residuals ei are uncorrelated with one another. Mathematically, the variance–covariance matrix of the errors is diagonal.

See https://en.wikipedia.org/wiki/Regression_analysis#Underlying_assumptions

Linear models overview

girafe
ai

02

Linear functions aka linear maps

In math is a mapping V → W between two vector spaces that preserves the operations of vector addition and scalar multiplication.

So we have two basic properties:

- additivity

$$f(\mathbf{u}+\mathbf{v}) = f(\mathbf{u}) + f(\mathbf{v})$$

- homogeneity

$$f(c\mathbf{u}) = cf(\mathbf{u})$$

Linear models

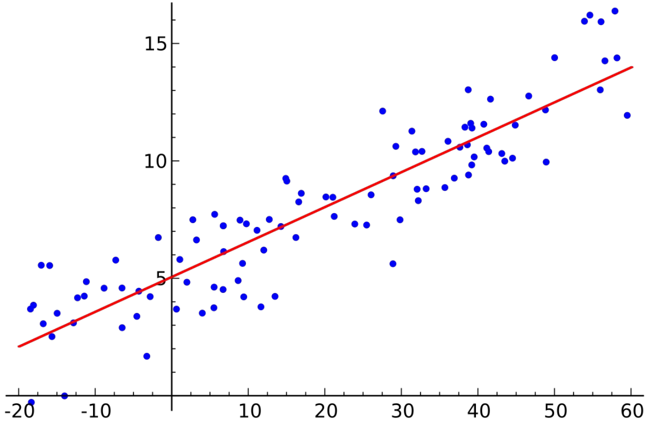
In applied statistics we use linear functions to predict different values

$$Y = X_1 + X_2 + X_3$$

Y	$X_1 + X_2 + X_3$
Dependent Variable	Independent Variable
Outcome Variable	Predictor Variable
Response Variable	Explanatory Variable

Linear models

- Regression models



Estimated (or predicted) Y value for observation i

Estimate of the regression intercept

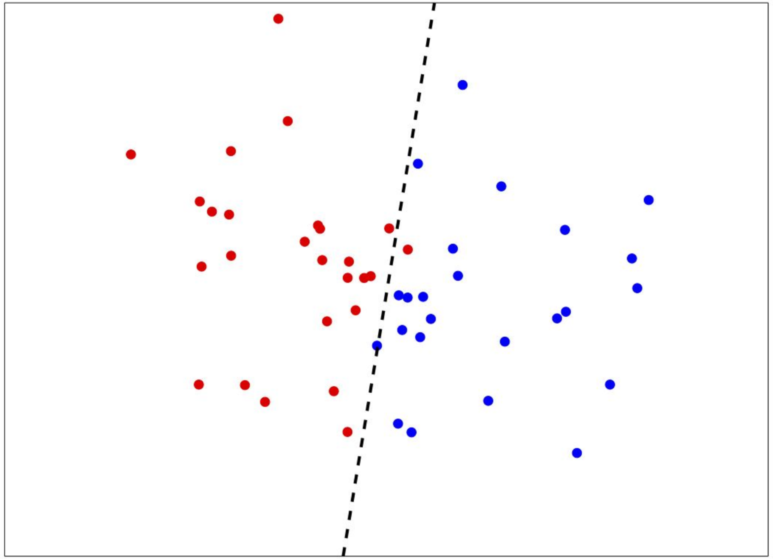
Estimate of the regression slope

Value of X for observation i

$$\hat{Y}_i = b_0 + b_1 X_i$$

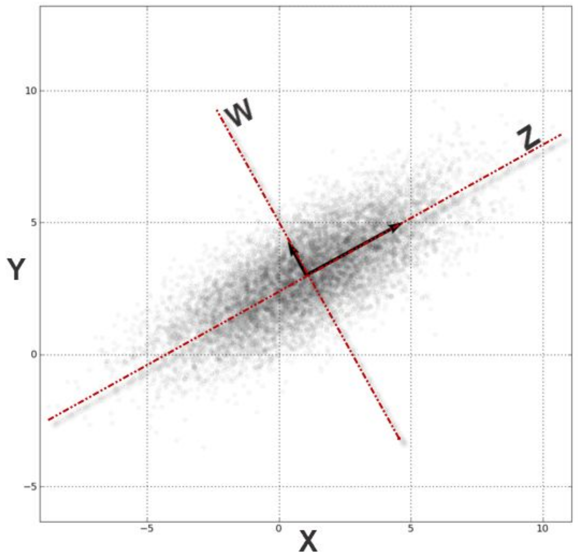
Linear models

- Regression models
- Classification models



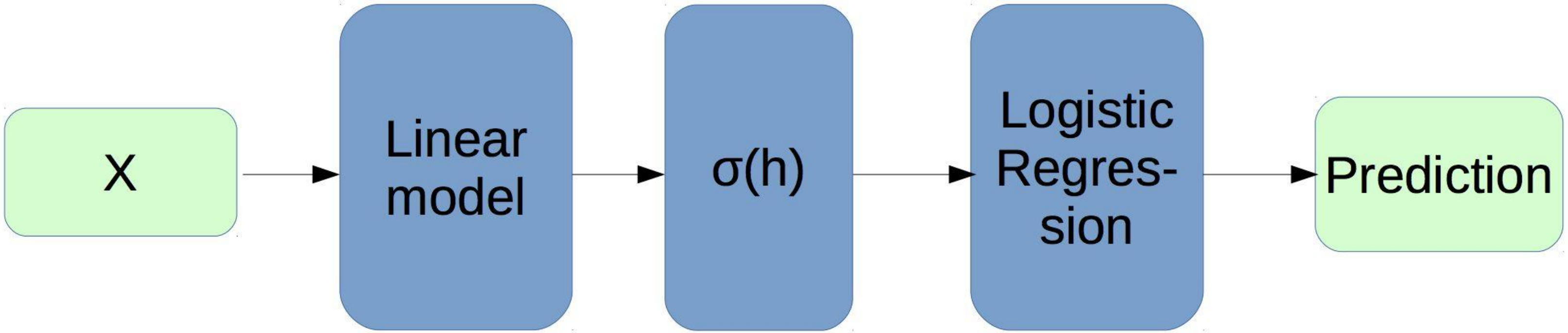
Linear models

- Regression models
- Classification models
- Unsupervised models (e.g. PCA)



Linear models

- Regression models
- Classification models
- Unsupervised models (e.g. PCA)
- Building block of other models (ensembles, NNs, etc.)

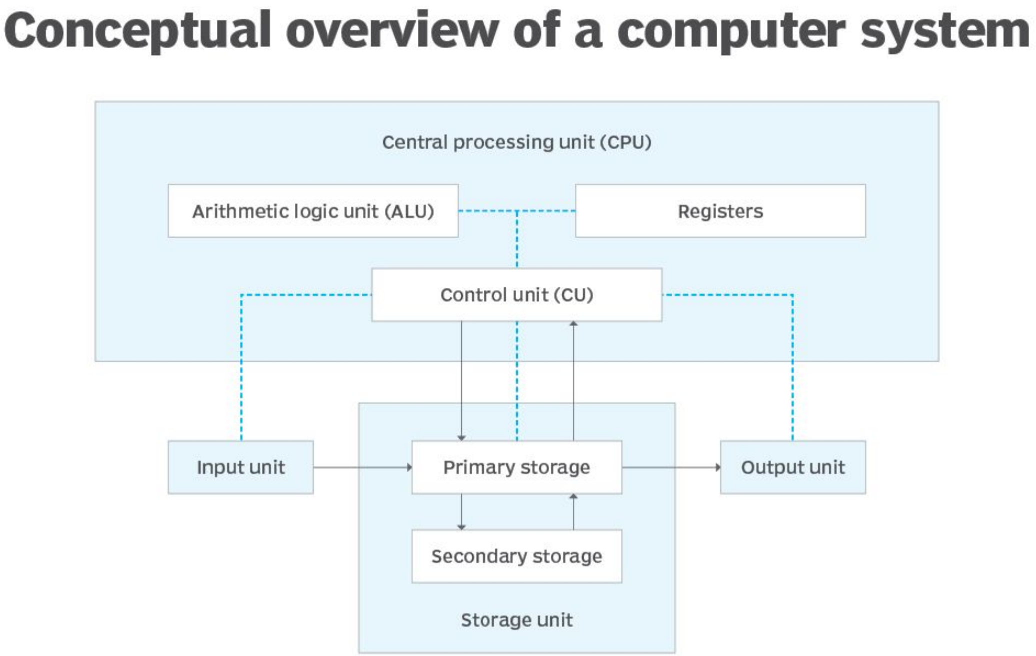


A simple neural network

The only efficiently calculated

For computers the only one model we can calculate natively is linear model.

That’s why it is so important to study it in depth!



Linear regression

girafe
ai

03

Linear Regression model

When estimator is linear

$$f_w(x) = w_0 + \sum_{i=1}^{p} w_i x_i \equiv x^T w$$

estimation gets linear

Note: x and w are supposed to include bias term (conventional notation)

$$w = (w_0, w_1, \ldots, w_n)^T$$
$$x = (1, x_1, \ldots, w_n)^T$$

Linear Regression problem

Observed samples

$$(x^i, y^i), i = 1, \ldots, n$$

$$x^i \in R^p, y^i \in R$$

Matrix form of dataset

$$X = [x^1, \ldots, x^n]^T, X \in R^{n \times p}$$

$$Y = [y^1, \ldots, y^n]^T, Y \in R^n$$

Linear Regression

$$f_w(X) = Xw = \hat{Y} \approx Y$$

Linear Regression problem

How to choose weights?
With MLE!

$$\text{Empirical risk} = \sum_{\text{by objects}} \text{Loss on object} \to \min_{\text{model params}}$$

$$Q(X) = \sum_{i=1}^{n} L(y^i, f_w(x^i)) \to \min_w$$

Loss functions

$$\text{MSE:} \quad L(y_t, y_p) = (y_t - y_p)^2$$

$$\text{MAE:} \quad L(y_t, y_p) = |y_t - y_p|$$

MSE minimization equivalents Maximum Likelihood Estimation in certain conditions (e.g. Gaussian noise)

Linear Regression analytical solution

For MSE closed form solution exists

$$Q_{\text{MSE}}(X) = \sum_{i=1}^{n}(y^i - f_w(x^i))^2 = ||Y - Xw||^2 = (Y - Xw)^T(Y - Xw) \to \min_w$$

$$\nabla_w Q(X) = \nabla_w (Y^T Y - (Xw)^T Y - Y^T Xw + (Xw)^T Xw) =$$
$$= 0 - Y^T X - Y^T X + 2X^T X w^T = 0$$

$$w^* = (X^T X)^{-1} X^T Y$$

Gauss-Markov theorem

girafe
ai

04

Linear Regression analytical solution

$$Y = f(X) + \varepsilon$$

$$f_w(x) = w_0 + \sum_{i=1}^{p} w_i x_i$$

$$\mathbb{E}(\varepsilon_i) = 0 \quad \forall i$$

$$\mathrm{Var}(\varepsilon_i) = \sigma^2 < \inf \quad \forall i$$

$$\mathrm{Cov}(\varepsilon_i, \varepsilon_j) = 0 \quad \forall i \neq j$$

Minimizing MSE loss gives

Best Linear Unbiased Estimation (BLUE)

Estimator with minimal Variance from all unbiased linear estimators

$$w^* = (X^T X)^{-1} X^T Y$$

$$\mathbb{E}(w^*) = w_{\mathrm{true}}$$

$$Var(w^*) = min$$

Problems with matrix solution

- Computation complexity
  - ~n^3 for n*n matrix
- Unstable inversion result
  - fixed point arithmetics
  - multicollinear features

$$w^* = \boxed{(X^T X)^{-1}} X^T Y$$

So no one uses such way to find solution

Unstable solution

In case of multicollinear features the matrix $X^T X$ is almost singular .

It leads to unstable solution:

```
w_true
```

```
array([ 2.68647887, -0.52184084, -1.12776533])
```

```
w_star = np.linalg.inv(X.T.dot(X)).dot(X.T).dot(Y)
w_star
```

```
array([   2.68027723, -186.0552577 ,  184.41701118])
```

corresponding features are almost collinear

Unstable solution

In case of multicollinear features the matrix $X^T X$ is almost singular .

It leads to unstable solution:

```
w_true
```

```
array([ 2.68647887, -0.52184084, -1.12776533])
```

```
w_star = np.linalg.inv(X.T.dot(X)).dot(X.T).dot(Y)
w_star
```

```
array([   2.68027723, -186.0552577 ,  184.41701118])
```

the coefficients are huge and sum up to almost 0

Regularizations

girafe
ai

05

L2 regularization

To make the matrix nonsingular, we can add a diagonal matrix:

$$w = (X^T X + \lambda^2 I)^{-1} X^T Y$$

It is also called Tikhonov regularization or Ridge regression.

Regularization constraints model weights and decrease them.
Resulting solution is **biased** (doesn't converge to true value of w).

Actually, it's a solution for the following loss function:

$$L_2 = ||Y - Xw||_2^2 + \lambda^2 ||w||_2^2$$

exercise: derive it by yourself

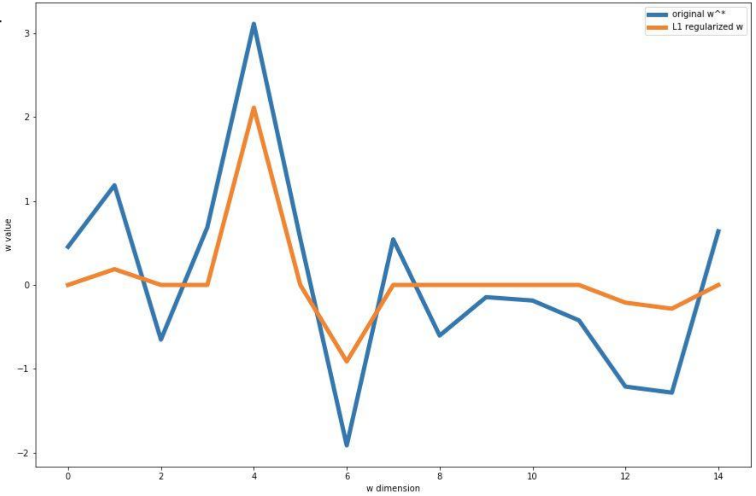
L1 regularization

What if we add L1 norm of w to our loss? This technique is called LASSO: least absolute shrinkage and selection operator

$$L_1 = ||Y - Xw||_2^2 + \lambda^2 ||w||_1$$

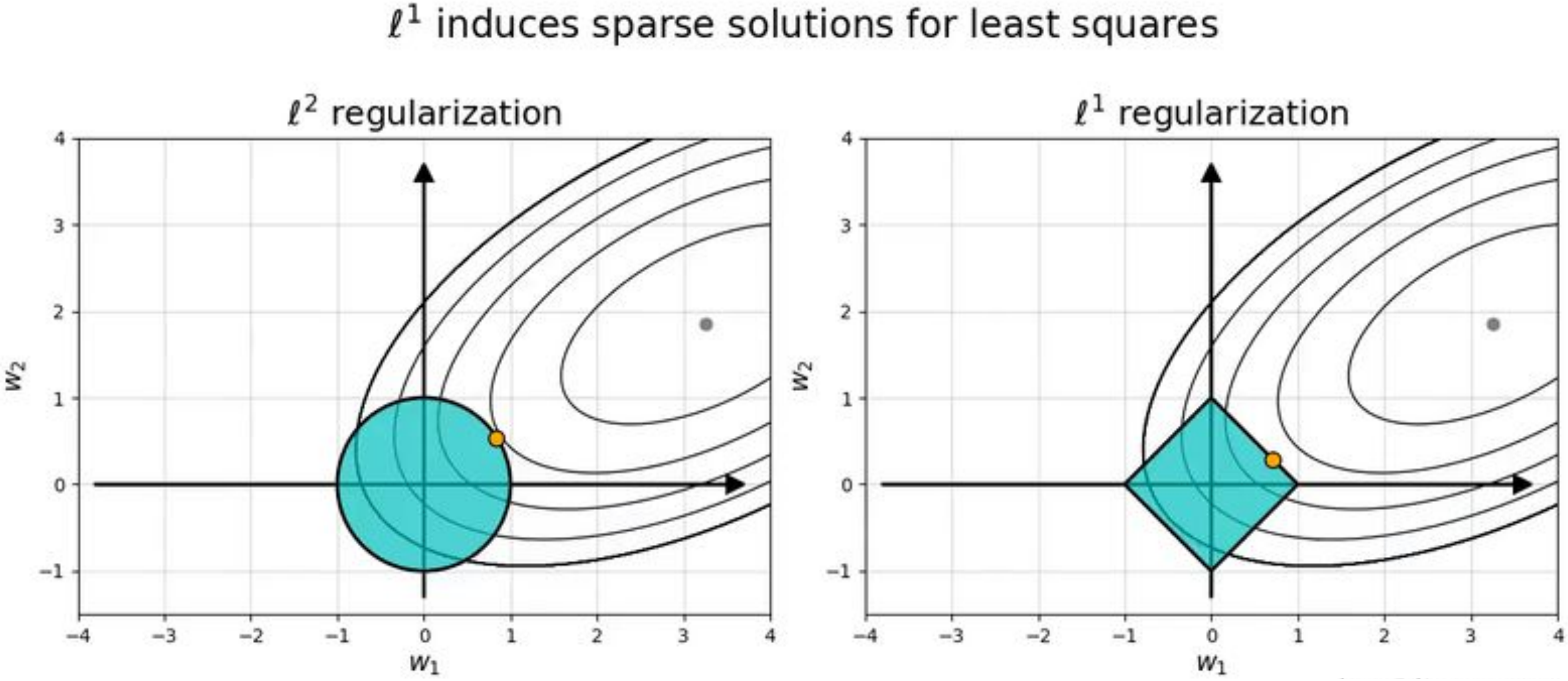
For this case there is no such an elegant solution as for L2 regularization, however solution exists for orthonormal design (see Spokoiny's book p.173)

$$\widehat{\theta}_j = \begin{cases} (\widetilde{\theta}_j - \lambda)_+ & \widetilde{\theta}_j \geq 0, \\ -(|\widetilde{\theta}_j| - \lambda)_+ & \widetilde{\theta}_j < 0 \end{cases}$$



Thus this type of regularization performs implicit feature selection

L1 vs L2 regularization



ElasticNet regularization

Applying both types of regularization also works

$$L_{EN} = ||Y - Xw||_2^2 + \lambda_1^2||w||_1 + \lambda_2^2||w||_2^2$$

Loss functions in regression

$$MSE(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{N}\|\mathbf{y} - \hat{\mathbf{y}}\|_2^2 = \frac{1}{N}\sum_i (y_i - \hat{y}_i)^2$$

$$MAE(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{N}\|\mathbf{y} - \hat{\mathbf{y}}\|_1 = \frac{1}{N}\sum_i |y_i - \hat{y}_i|$$

Different norms

Once more: loss functions:

$$\boxed{MSE = \frac{1}{n}\|\mathbf{x}^T\mathbf{w} - \mathbf{y}\|_2^2}$$

only works for Gauss-Markov theorem

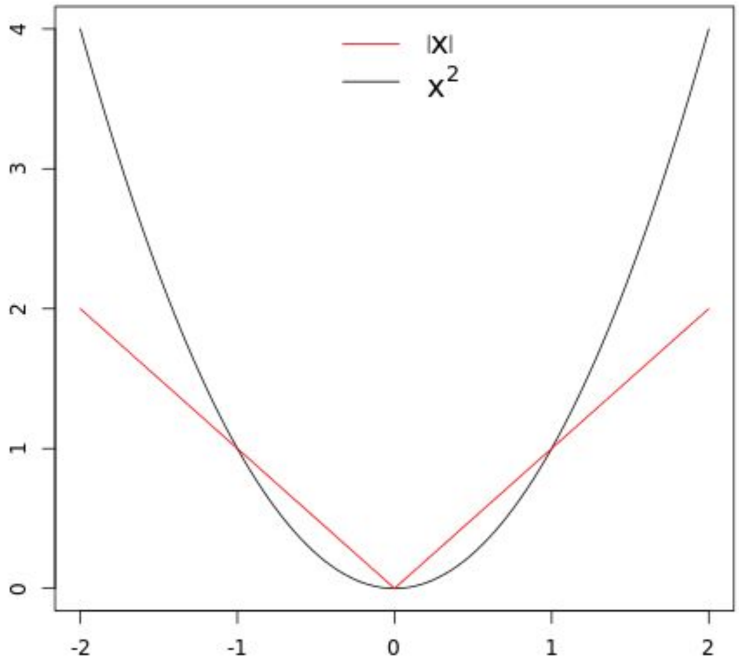
$$MAE = \frac{1}{n}\|\mathbf{x}^T\mathbf{w} - \mathbf{y}\|_1$$

Regularization terms:

- L2 $\|\mathbf{w}\|_2^2$
- L1 $\|\mathbf{w}\|_1$

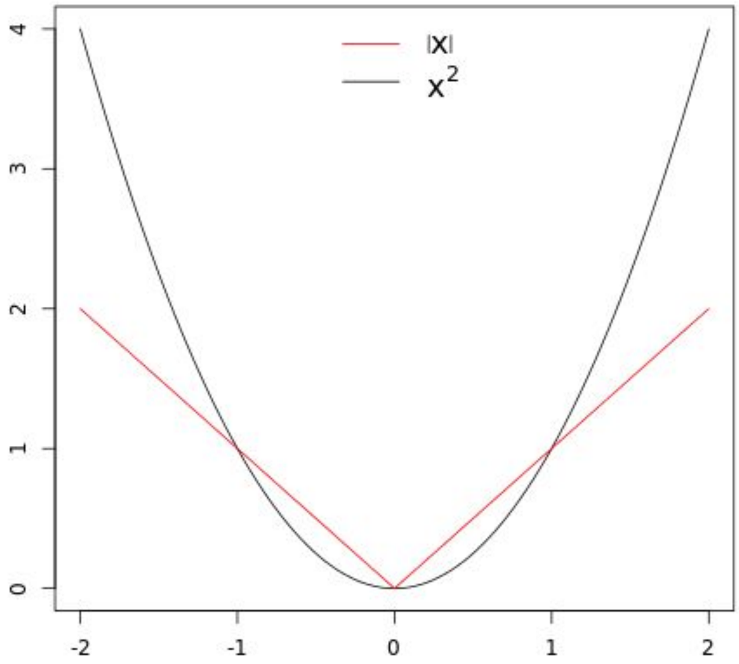
Loss function properties

- MSE (L_2)
  - delivers BLUE according to Gauss-Markov theorem
  - differentiable
  - sensitive to noise
- MAE (L_1)
  - non-differentiable
    - not a problem
  - much more prone to noise



Regularization properties

- L_2 regularization
  - constraints weights
  - delivers more stable solution
  - differentiable
- L_1 regularization
  - non-differentiable
  - not a problem
  - selects features



Metrics in regression

- MSE - Mean Square Error
- MAE - Mean Absolute Error
- RMSE - Root Mean Square Error
- MAPE - Mean Absolute Percentage Error
- SMAPE - Symmetric Mean Absolute Percentage Error
- R2 - “R squared” aka coefficient of determination
- etc... (any combination you like)

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n} \left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

$$\text{SMAPE} = \frac{1}{n}\sum_{i=1}^{n} \frac{2 \cdot |y_i - \hat{y}_i|}{|y_i| + |\hat{y}_i|}$$

Model validation and evaluation

girafe
ai

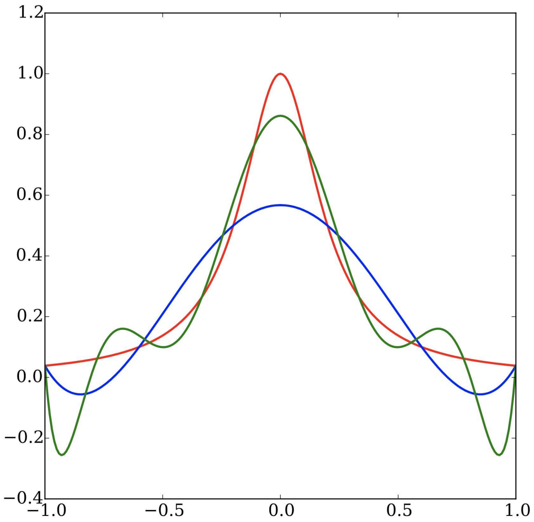
06

Runge's phenomenon

Bigger model is not always better!

Runge function interpolation on uniform grid

$$f(x) = \frac{1}{1+25x^2}, x \in (-1, 1) \quad x_i = \frac{2i}{n} - 1, i = 0, \ldots, n$$



by polynomials of n-th degree

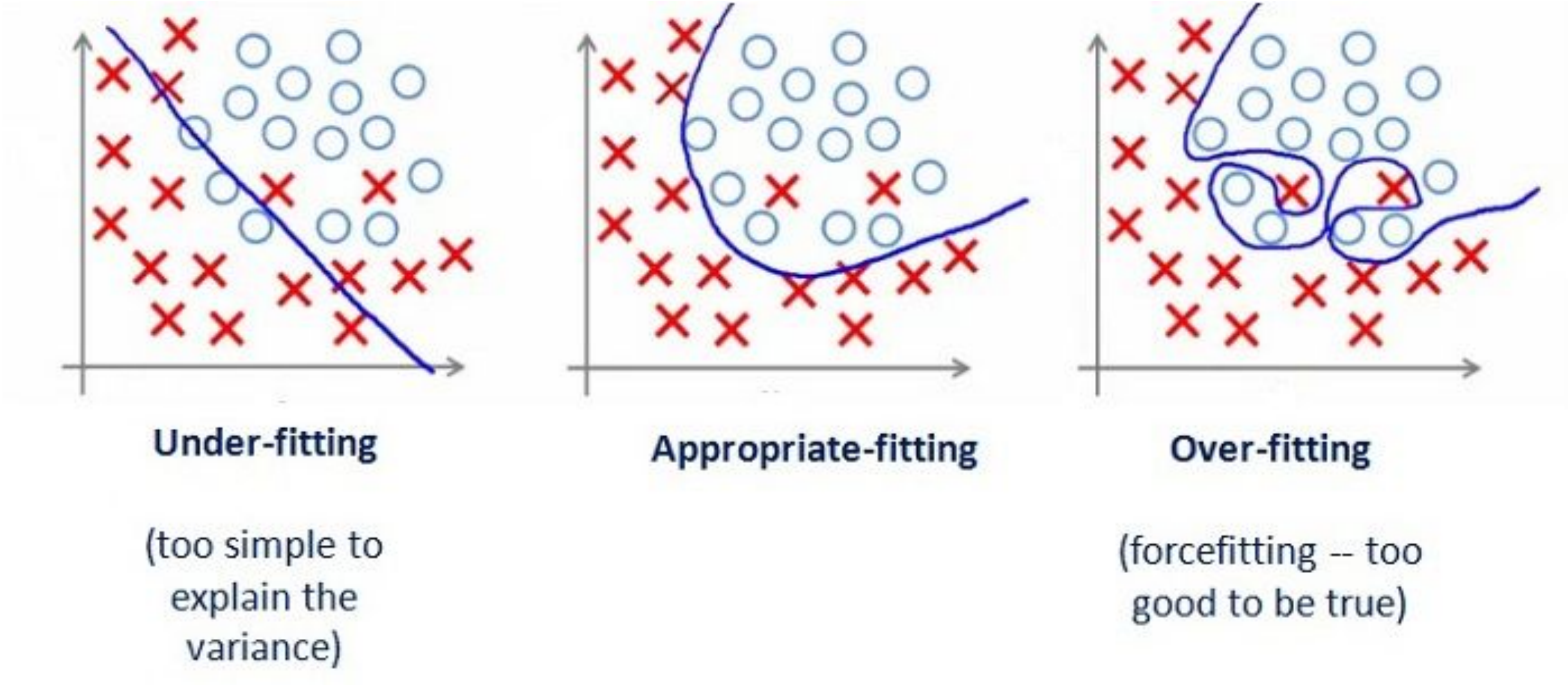
$$P_n(x) = p_n x^n + \cdots + p_1 x + p_0$$

$$P_n(x_i) = f(x_i)$$

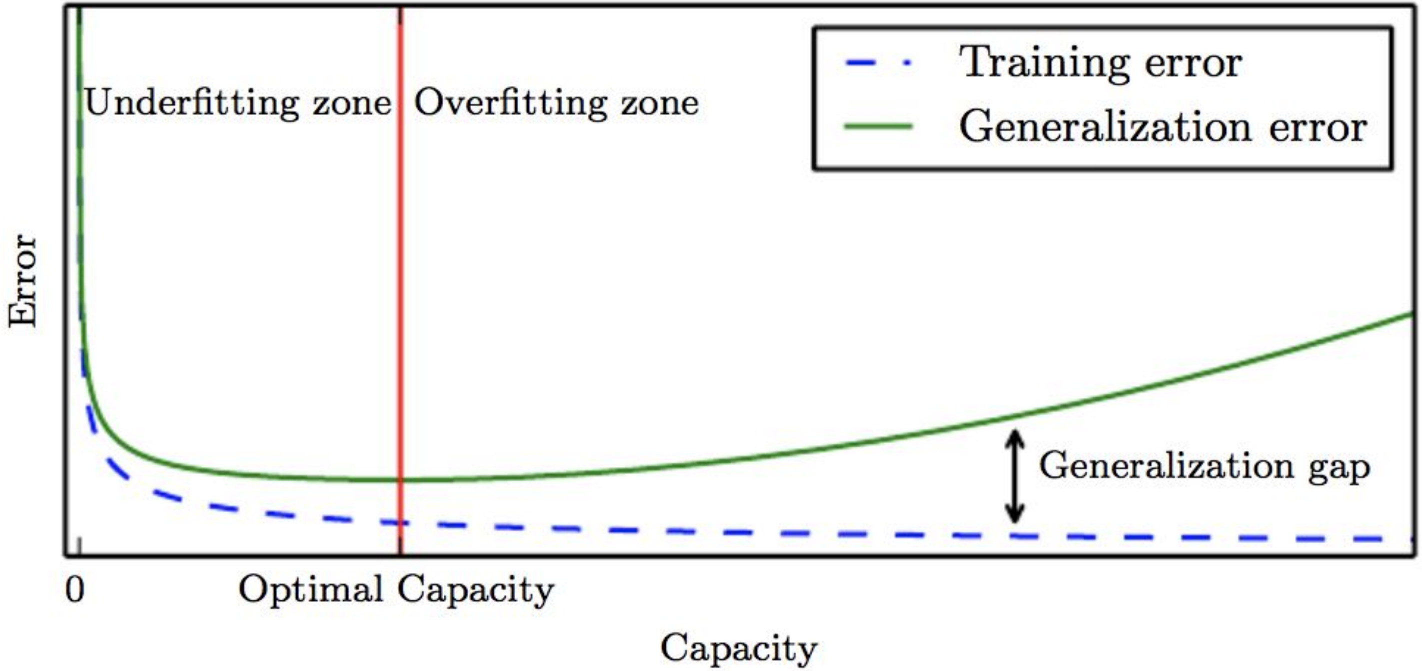
is infinitely bad on the whole interval

$$\lim_{n\to\infty} \left(\max_{-1\le x\le 1} |f(x) - P_n(x)| \right) = +\infty$$

Overfitting vs. underfitting



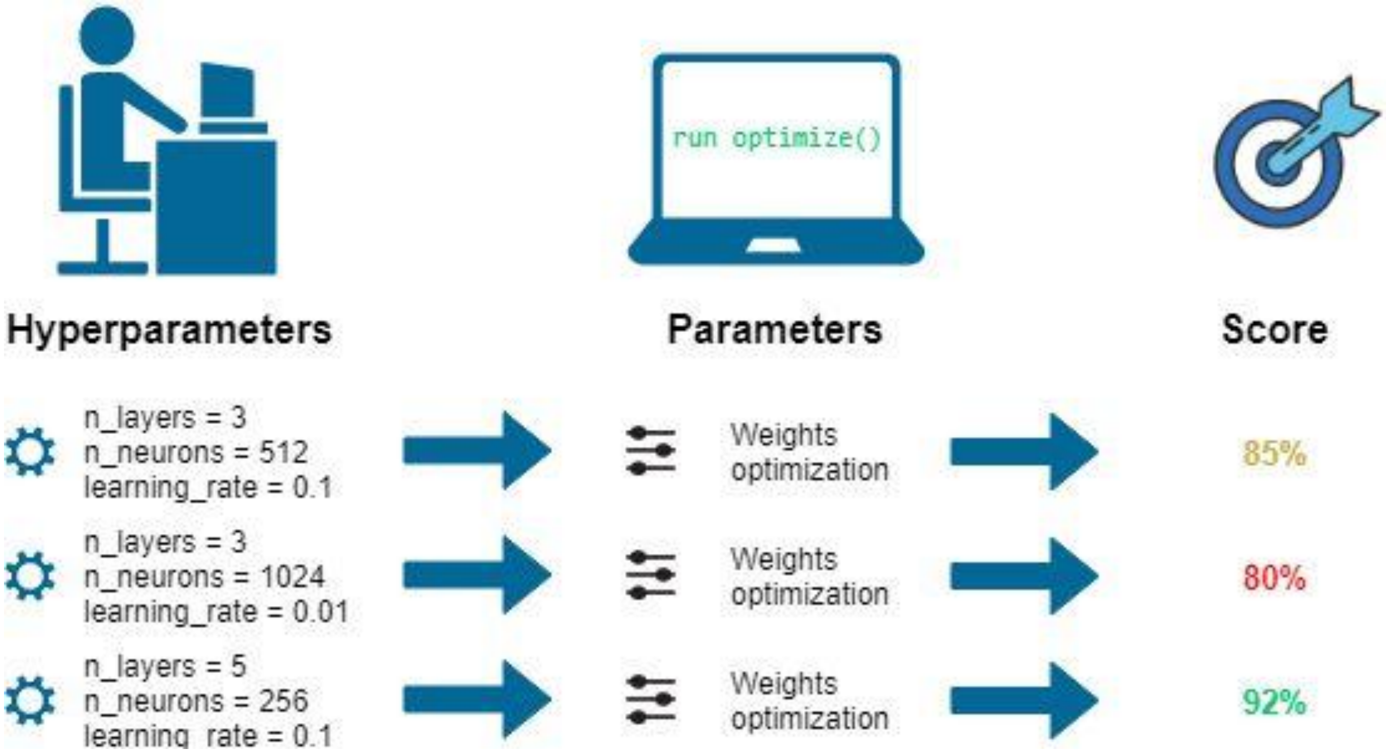
Overfitting vs. underfitting



Evaluating the quality

Is it good enough?

Parameters and hyperparameters



Parameters and hyperparameters

The values of parameters are derived via learning. In other words parameter values are result of optimization process.

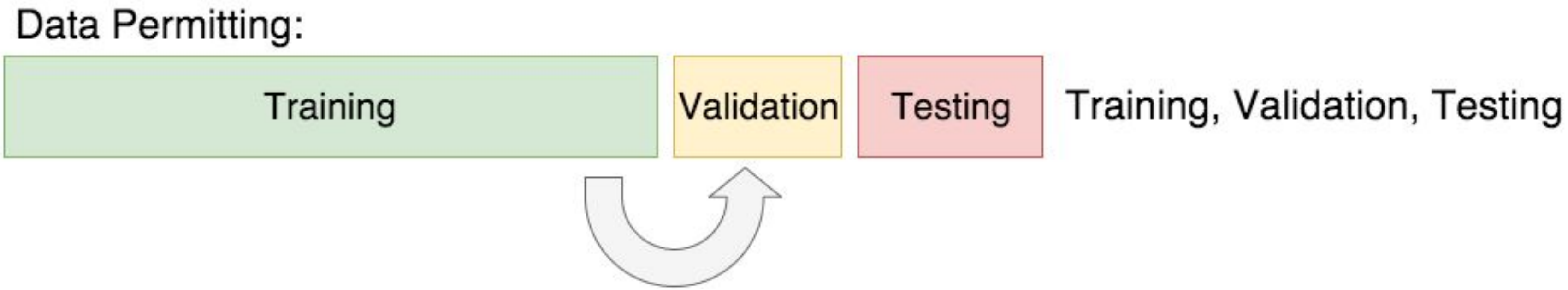
A hyperparameter is a constant parameter whose value is set before the learning process begins.

Technically any parameter can be made hyperparameter (by fixing it) and vise versa (by estimating hyperparameter from data anyhow).
But for mainstream models there is a stable setup on what is what.

Comparison

	Defined by	Depend on the training data	Order of optimization methods	Required for	Affect the complexity of the model
Parameters	during the training	yes	first (gradient)	predictions	no
Hyperparameters	before the start of training	no	zero (manual, Bayesian)	training	yes

Dataset splits

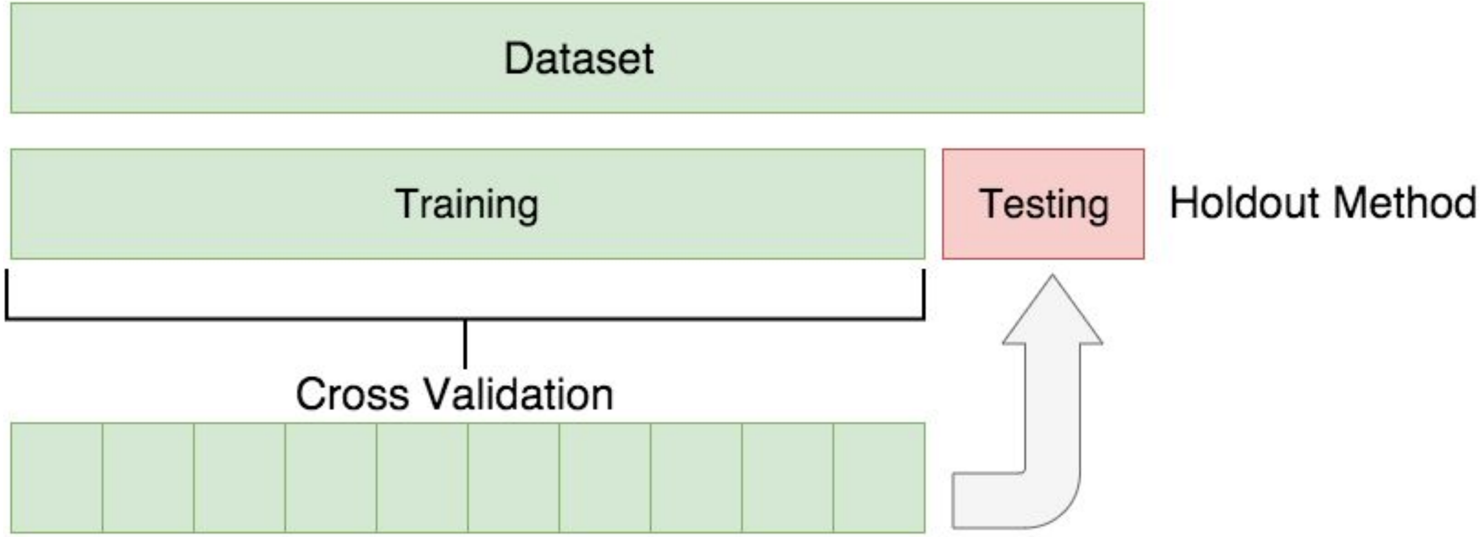


Fixed dataset split is the most used in practice nowadays
because sizes of dataset are usually big enough.

Stages of model training

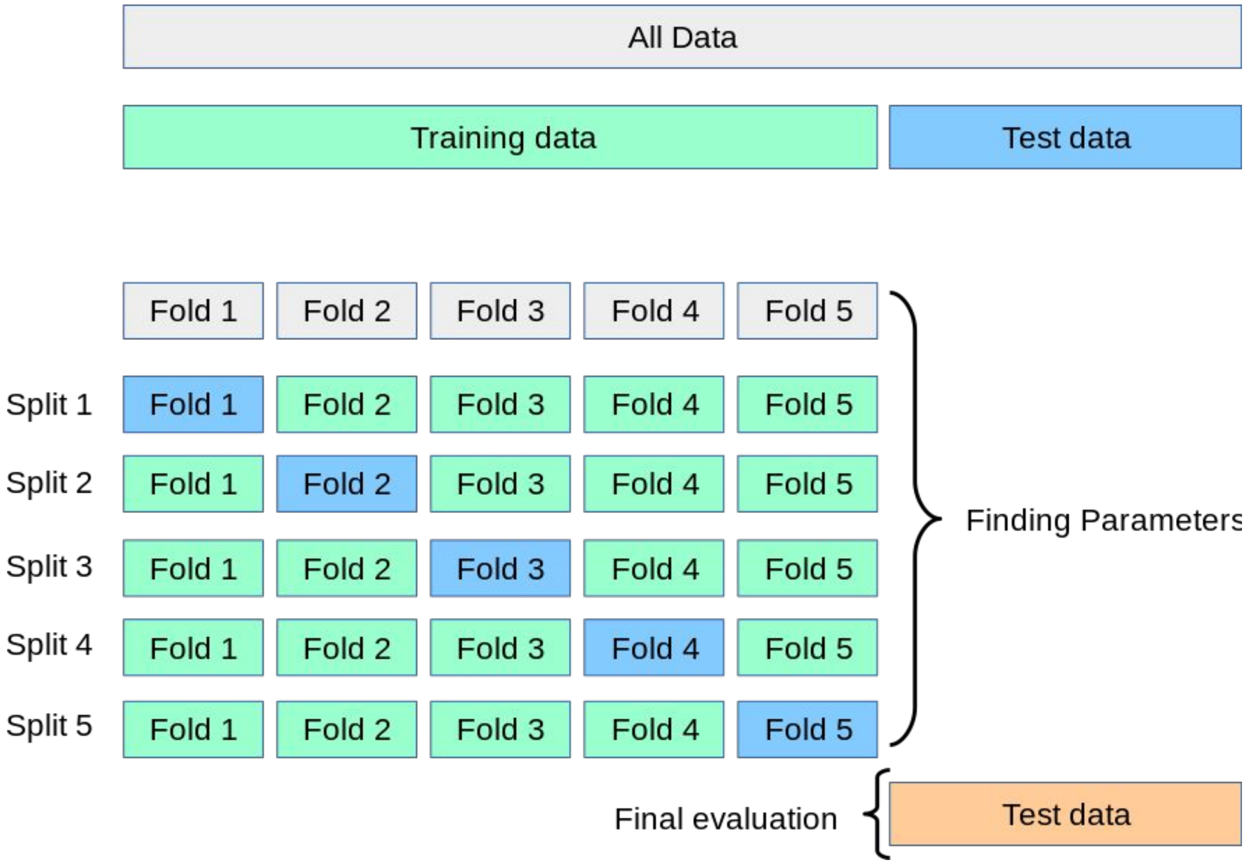
Split	training	validation	test
Used for	parameters optimization	hyperparameters selection	quality measurement
Overfitting level	high	average	low

Cross-validation

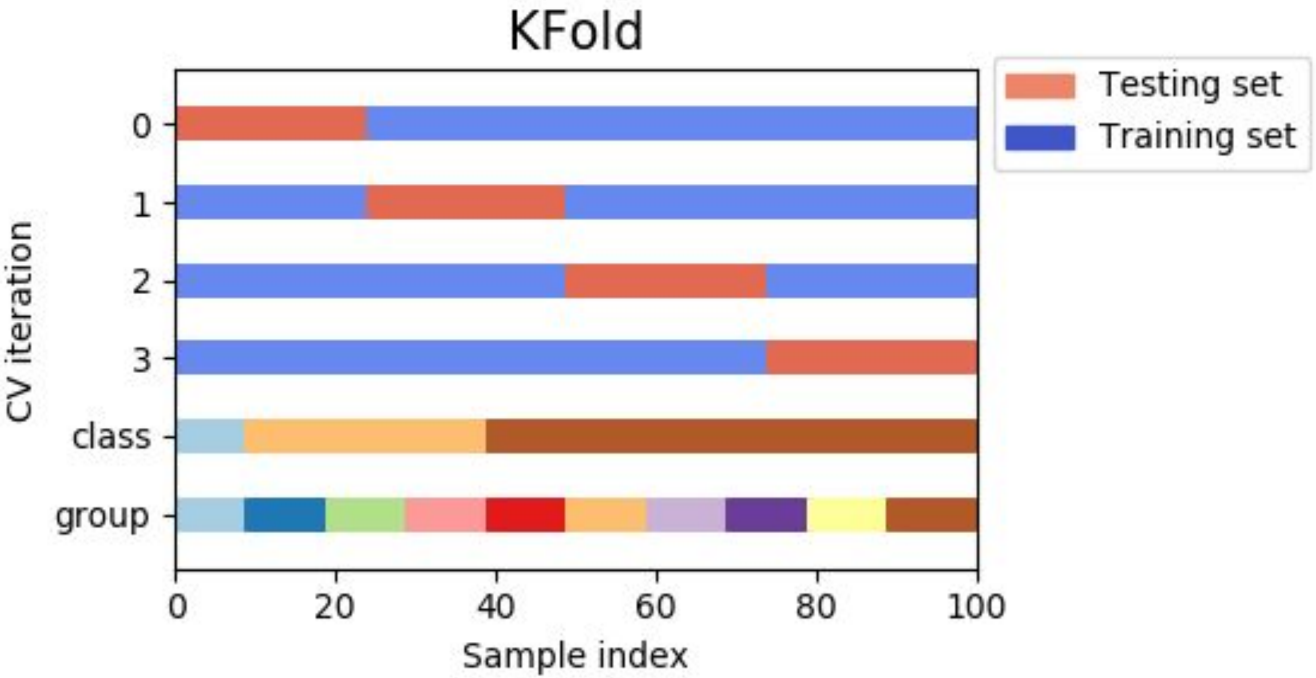


In real life is used only on **small datasets** (<10^4 samples)

Cross-validation

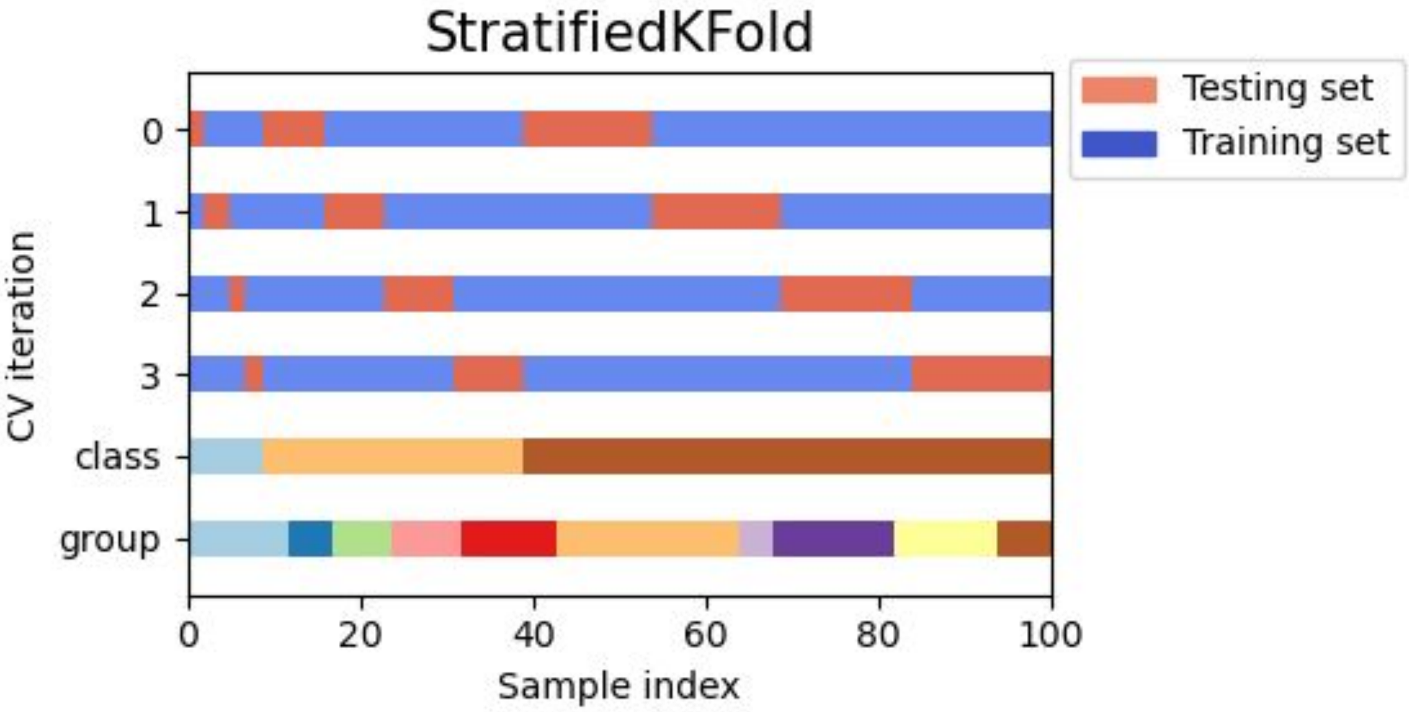


More validations



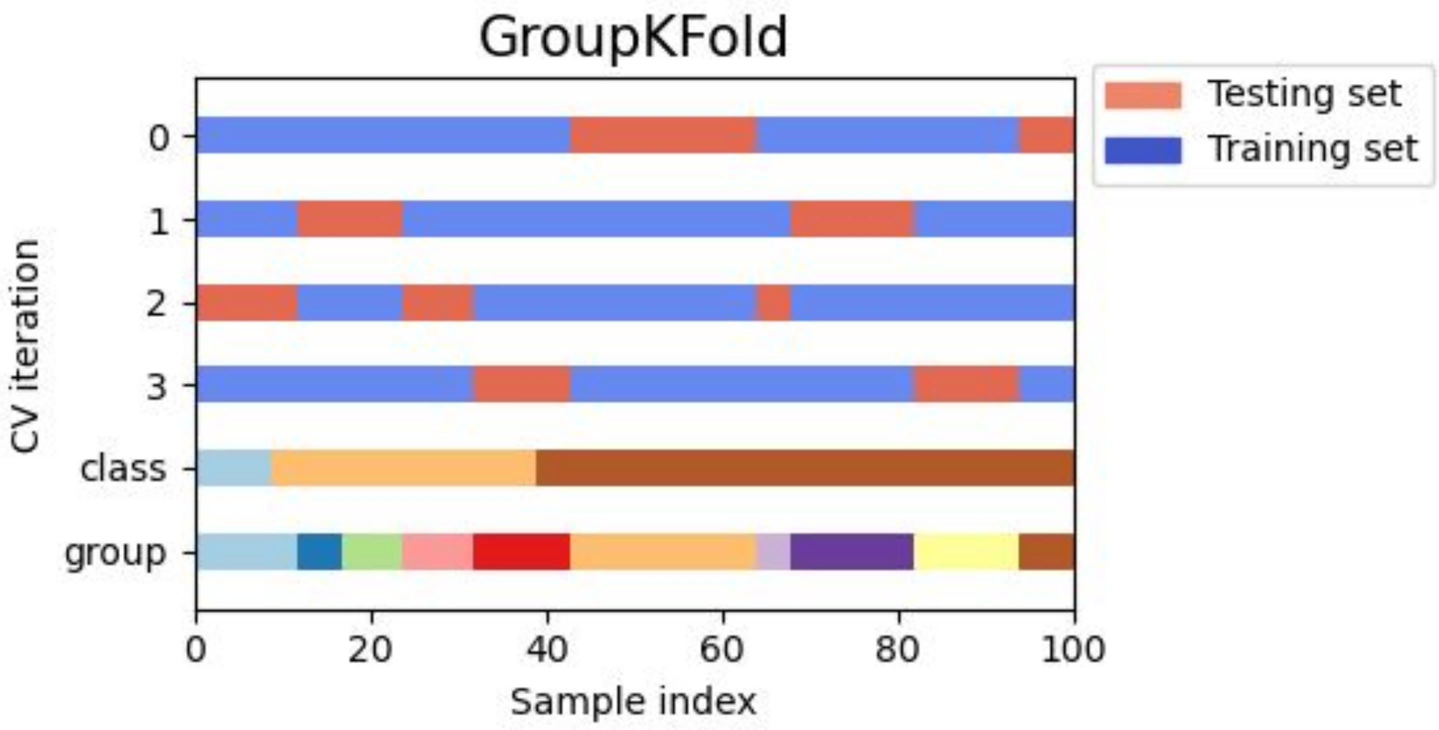
Special case: Leave One Out (LOO) - good for tiny datasets

More validations



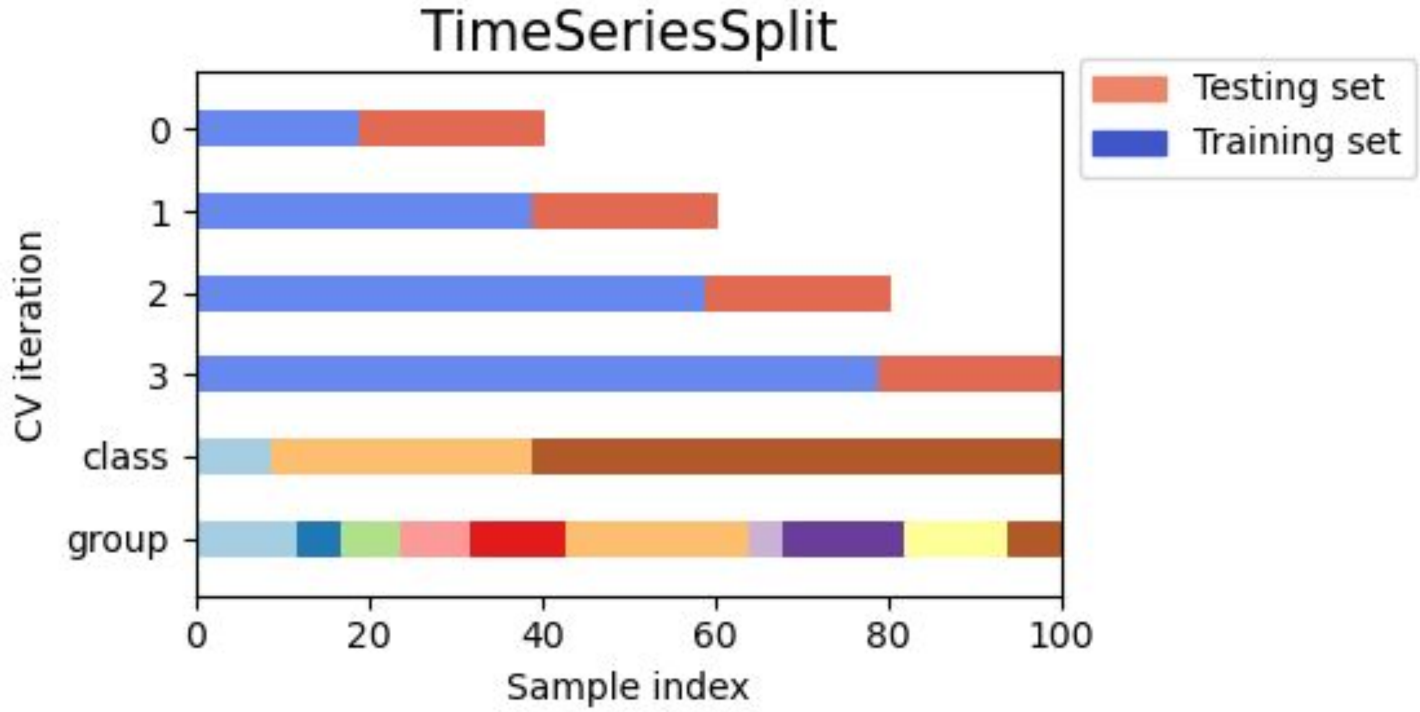
Preserve class ratio for each split. **Default for sklearn** methods

More validations



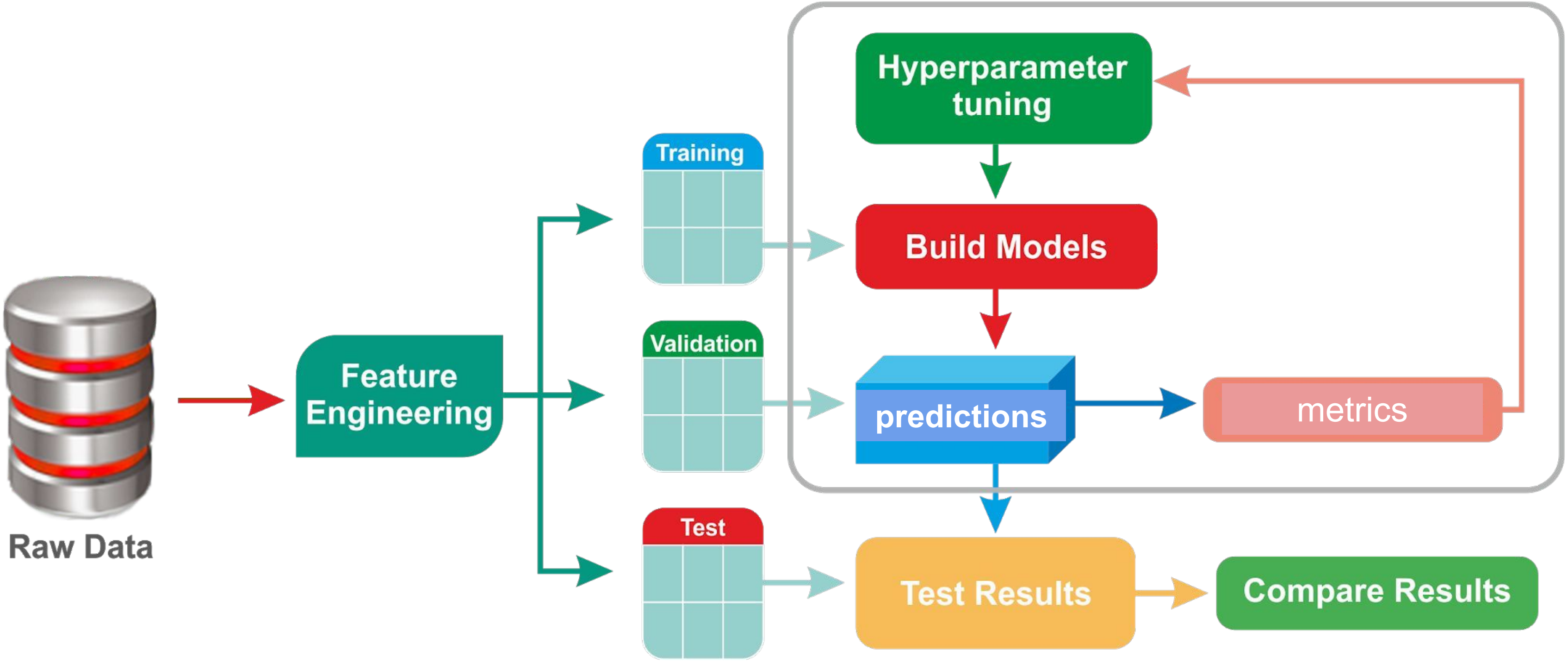
Set whole group either to train or validation.
Used in medicine and ranking (search, recsys)

More validations



Never use train_test_split in this case!!!

Stages of model training



Although feature engineering better be done before split!

Data leaks

TODO: cases of data leaks

Revise

1. Linear models overview
2. Linear Regression under the hood
3. Gauss-Markov theorem
4. Regularization in Linear regression
5. Model validation and evaluation

Thanks for attention!

Questions?

girafe
ai