Machine Learning

Baseline, Bias, Overfitting, Cross Validation

ALBERTSCHOOL

MINES PARIS | PSL

Thomas Milcent - 2025

Machine Learning - DataScience Approach

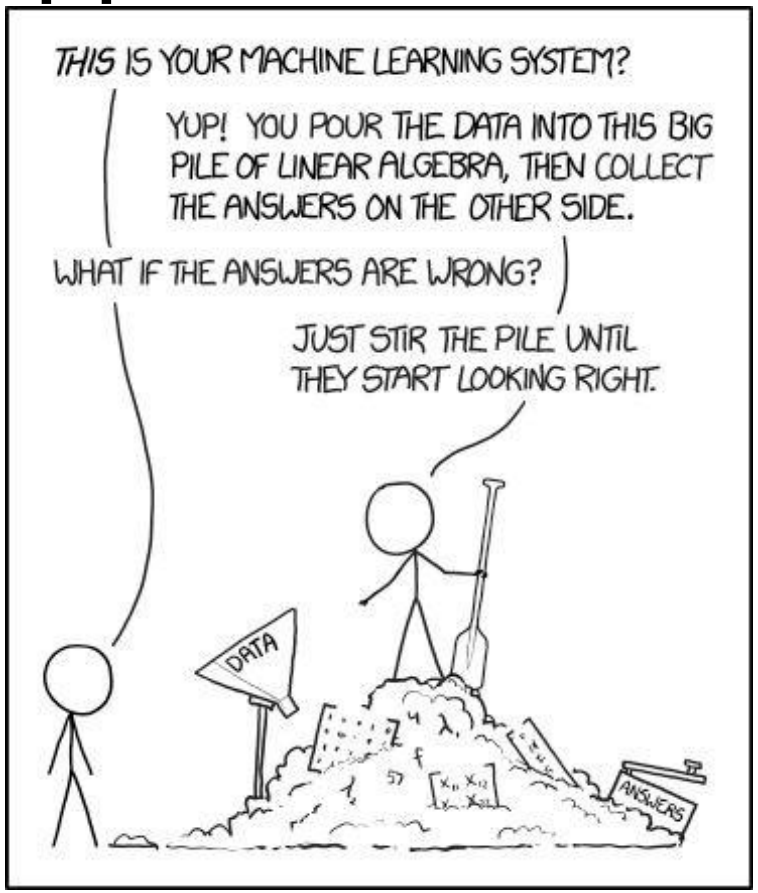
Noise in the data

Imprecision in **recording the features**

Errors in labelling the data points (**teacher noise**): especially the case on manually labeled data

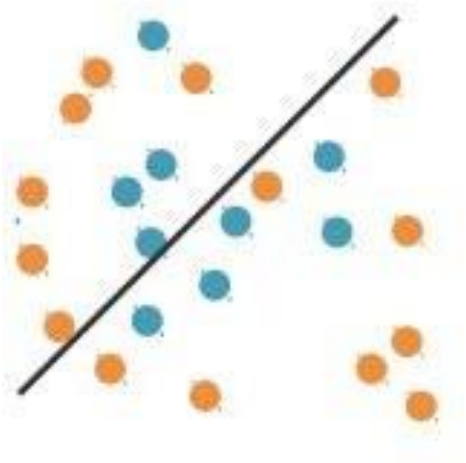
Missing features (**hidden** or **latent**)

Making errors on the training set is possible!

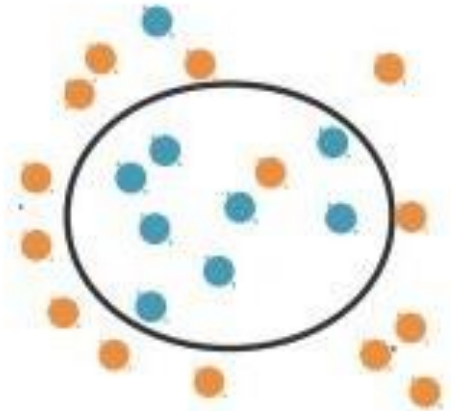


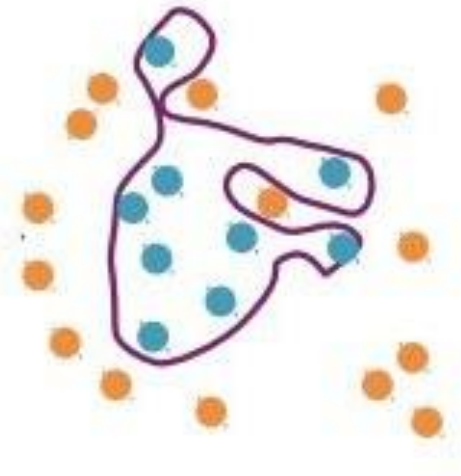
Machine Learning - DataScience Approach

Models in increasing complexity



Simple model
Eg linear models





Complex model
kNN, too many parameters models

Machine Learning - DataScience Approach

Noise and model complexity

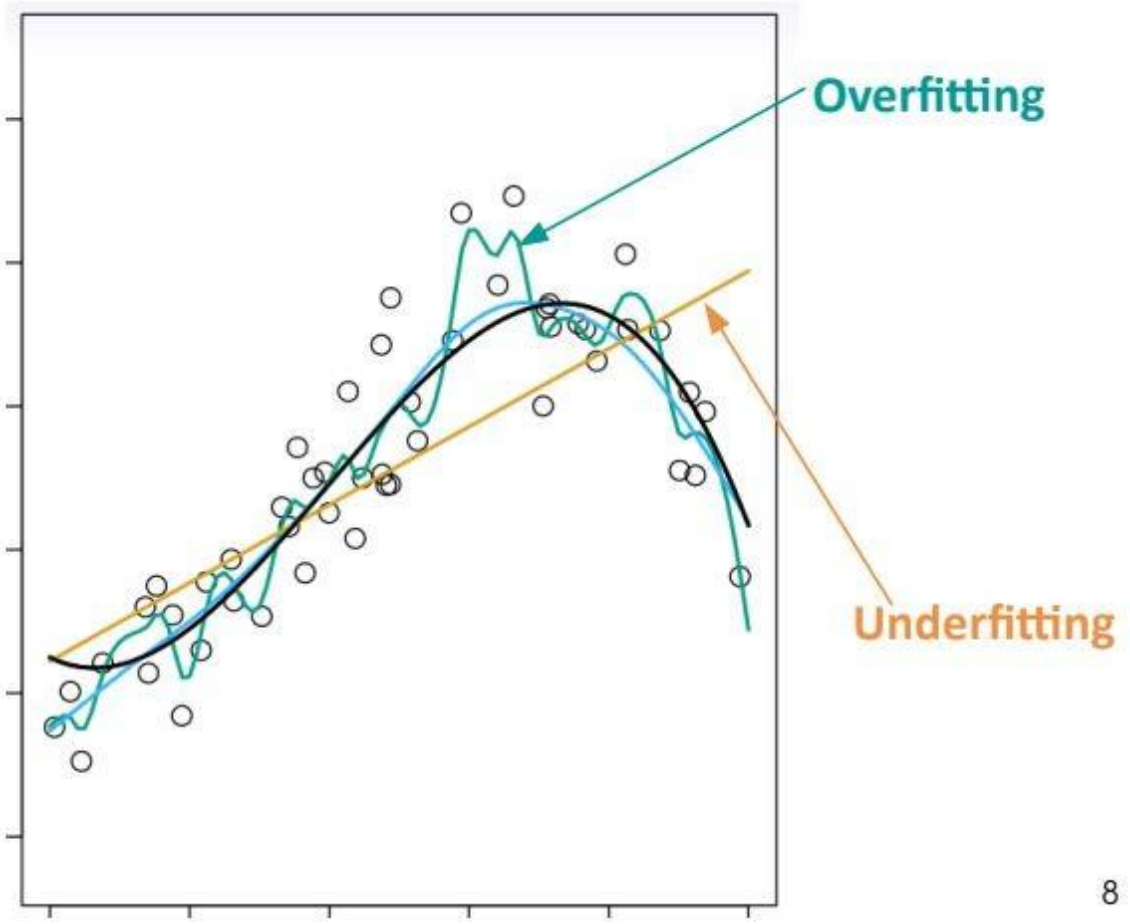
Use simple models!

- Easier to **use**
  - Lower computational complexity
- Easier to **train**
  - Lower space complexity
- Easier to **explain**
  - More interpretable/explainable
- **Generalize better**

Occam's razor: Simpler explanations are more plausible

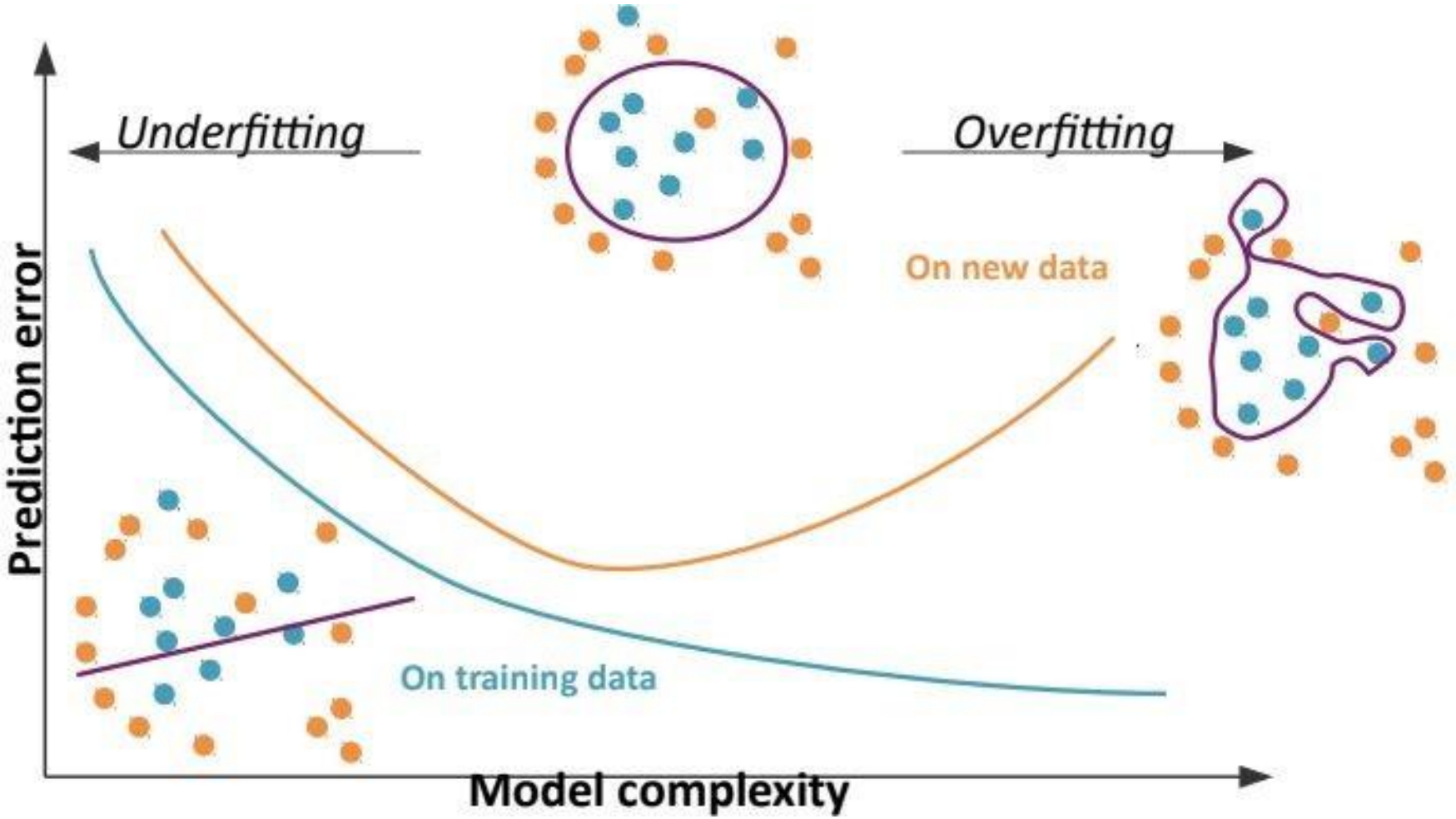
Flaws of Overfitting and Underfitting

Overfitting vs Underfitting



Flaws of Overfitting and Underfitting

Generalization error vs model complexity



Bias-Variance tradeoff

Bias-variance tradeoff

Bias: difference between the expected value of the estimator and the value being estimated.

$$\text{Bias}(f(\boldsymbol{x})) = \mathbb{E}[f(\boldsymbol{x}) - y]$$

- A simpler model has a higher bias
- **High bias can cause underfitting**

Variance : deviation from the execed value of the estimates

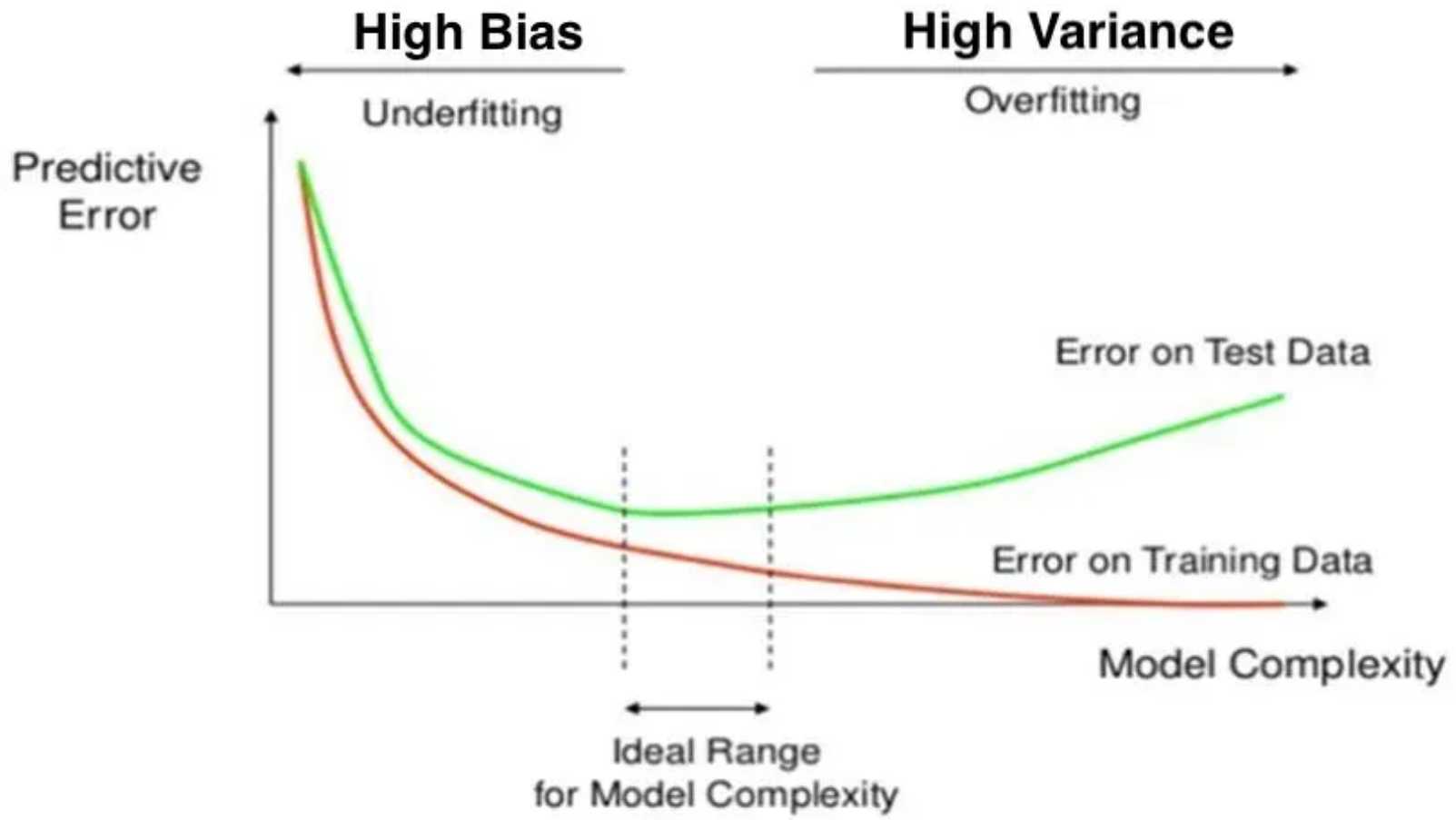
$$\text{Var}(f(\boldsymbol{x})) = \mathbb{E}[(f(\boldsymbol{x}) - \mathbb{E}(f(\boldsymbol{x}))^2]$$

- A more complex model has a higher variance
- **High variance can cause overfitting**

$$\begin{aligned} \text{MSE}(f(\boldsymbol{x})) &= \mathbb{E}[(f(\boldsymbol{x}) - y)^2] \\ &= \text{Var}(f(\boldsymbol{x})) + \text{Bias}^2(f(\boldsymbol{x})) \end{aligned}$$

Link between Bias-Variance tradeof and overfitting

Bias-variance tradeoff - Generalization error vs model complexity

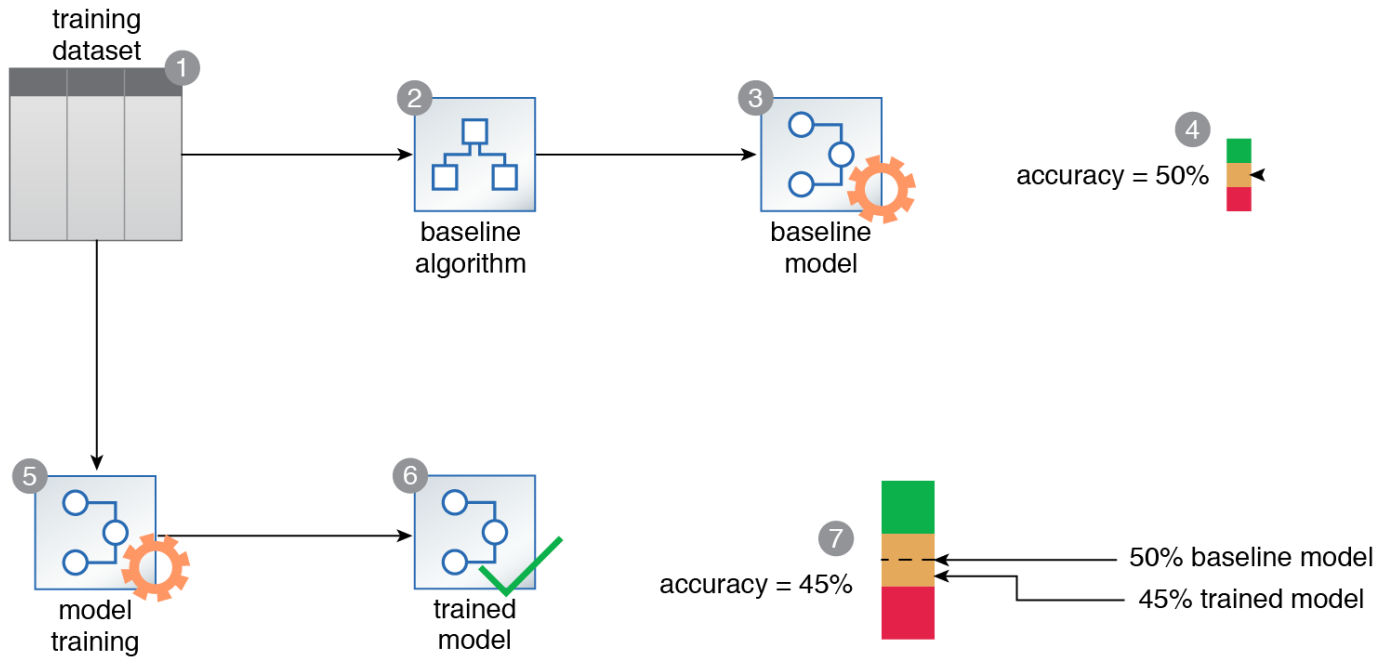


Introduction to Model Selection

A baseline model :
- provides a simple, transparent point of comparison
- allows to evaluate whether more complex models actually add value.

Getting a good baseline :
- easy to build
- Interpretable
- Fast

Ex : predicting the mean or median for regression, or the most frequent class for classification.



Introduction to Model Selection

Empirical error vs generalization error - Validation sets
The empirical error is the loss of the model on the training set $\mathcal{D} = \{x^i, y^i\}_{i=1,\ldots,n}$

The generalization error is the loss of the model on new data

The Empirical error is a poor estimate of the generalization error!

Choose the model that performs best on a validation set separate from the training set.

The error on the validation set is an estimation of the generalization error!

Introduction to Model Selection

Model selection

- What if we want to choose among k models?
  - Train each model on the train set
  - Compute the prediction error of each model on the validation set
  - Pick the model with the smallest prediction on the validation set

- What is generalization error?
  - We don’t know!
  - Validation data was used to select the model
  - We have “cheated” and looked at the validation data: it si not a good proxy for new, unseen data anymore.

From Validation to Test tests

Model selection - Validation sets

- Hence we need to set aside part of the data, the **test set**, that remains untouched during the entire procedure and on which we'll estimate the generalization error

Model **selection**: pick the best model
Model **assessment**: estimate its prediction error on new data

Going further : Cross-Validation

Sample re-use - Cross-Validation

- Cut the training set in k separate folds
- For each fold, train on the (k-1) remaining folds

Going further : Cross-Validation

Sample re-use - Cross-Validation

- Cut the training set in k separate folds
- For each fold, train on the (k-1) remaining folds

The mean prediction error on each folds estimated the expected prediction error!

Issue:

- **Training set size** become (k-1)n/k
  - Small training set leads to biased estimator of the error

- **Leave-one-out cross validation:** k=n
  - Approximately unbiased estimator of the expected prediction error
  - Computation can become burdensome (n repeats)

- In practice: set **k=5 or k=10**

Tuning your model with hyperparameters : Grid Search

Model selection - Grid Search

Hyperparameters are configuration settings for a model that are not learned from the data but need to be set prior to training.
Examples include the learning rate in a gradient boosting algorithm or the regularization parameter in Ridge/Lasso models.

Grid search involves defining a **grid of hyperparameter values** and then training and evaluating the model **for each combination of values** on the grid.
Basically, the grid can be compared to a **defined dictionary of hyperparameters and their values to test.**
The goal is to identify the combination of hyperparameters that results in the best model performance according to a specified metric

Sklearn has function to perform grid search
There are also other libraries for high performance/probabilistic search like **hyperopt : https://github.com/hyperopt/hyperopt**

Full model selection process

- **Splitting the Data:** Initially, the dataset is divided into training and validation sets. The training set is used to train different models with various hyperparameters, and the validation set is used to assess their performance.

- **Cross-Validation:** Cross-validation is employed to evaluate the model's performance across different subsets of the data. This helps in selecting the best model and its corresponding hyperparameters.

- **Selecting the Best Model:** Once the best model and hyperparameters are identified based on cross-validation performance, the next step is to use these to train a final model.

- **Training the Final Model:** The final model is then trained on the entire training dataset using the selected hyperparameters. This is often done to ensure that the model learns from the full diversity of the training data and captures the underlying patterns more effectively.

- **Evaluation:** After training the final model, it is typically evaluated on a separate test set that the model has never seen during training or cross-validation. This provides an unbiased assessment of the model's generalization performance on new, unseen data.

The art of feature engineering

Improving training performance - Feature engineering
Feature engineering is the process of creating new features or modifying existing ones in a dataset to improve the performance of a machine learning model.

Creating New Features: This involves generating additional features from the existing ones or external data. For example, A^2, AxB, A-B (on dates), extract information from text

Handling Missing Data: Dealing with missing values is an essential part of feature engineering. Strategies include **imputation** (replacing missing values with estimated values), **creating binary flags** indicating the presence of missing values, or **removing features** with too many missing values.

Scaling and **Normalization**: Standardizing or normalizing features ensures that they are on a consistent scale. This is important for models that are sensitive to the magnitude of input features (like linear models).

Encoding Categorical Variables: Many machine learning algorithms require numerical input, so categorical variables need to be encoded appropriately. Common methods include **one-hot encoding**, **label encoding**, or **target encoding**.

The art of feature engineering

Improving training performance - Feature engineering
Feature engineering is the process of creating new features or modifying existing ones in a dataset to improve the performance of a machine learning model.

Handling Outliers: Outliers can significantly impact the performance of a model. Identifying and appropriately handling outliers, such as through **transformation** or **removal**, can be part of feature engineering.

Binning and **Discretization**: Continuous variables can be converted into discrete bins or categories. This can help capture non-linear relationships and make the model more robust to outliers.

Feature Selection: Choosing the most relevant features for the model can improve efficiency and reduce overfitting. Techniques such as **recursive feature elimination** or **feature importance from tree-based models** can assist in feature selection.

Domain-Specific Features: Incorporating domain knowledge can lead to the creation of features that are more meaningful for the specific problem at hand.

Beware : Preprocessing and Feature Engineering has to be done **after splitting the dataset between train, validation and test sets** !! Otherwise you risk data leakage

The art of feature engineering

Improving training performance - Feature engineering
All of these techniques are to be found in scikit-learn.

A deep understanding of the data is quintessential for this crucial part.

Beware: More features will make your model less capable to fit the data! This is due to **Curse of dimensionality**

- As the number of dimensions increases, the available data points become more spread out in the high-dimensional space. In regions where there are few data points, it becomes challenging to make accurate predictions or estimations.
- With the increasing Volume Space comes more required computational power to cope with it!