MODEL LIFECYCLE

FÁBRICA DE MODELOS

DIEGO RODRIGUES DSC

INFNET

MODEL LIFECYCLE : FÁBRICA DE MODELOS

PARTE 1 : TEORIA

- FÁBRICA DE MODELOS
  - FRAMEWORK ESTATÍSTICO
  - HEURÍSTICAS
  - ALGORITMO DE TREINAMENTO
    - CARACTERÍSTICAS DO DATASET
    - SPLITTER
    - OTIMIZAÇÃO DE HIPERPARÂMETROS
    - RETREINO
    - REVISÃO DOS ALGORITMOS IMPLEMENTADOS

Produzir Ação

CICLO DE VIDA DO MODELO

Baseado em Dados

AMBIENTE PYTHON

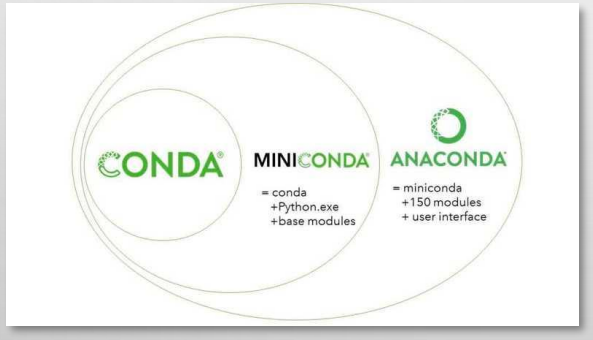
5. Visualização

6. Estimação e Inferência

7. Machine Learning

4. Variáveis Aleatórias

1. Editor de Código



2. Gestor de Ambiente

3. Ambiente Python do Projeto

3. Notebook Dinâmico

Cross Industry Standard Process for Data Mining - IBM

1) Requerimentos e Análise de Negócio

Entendimento do problema decisório, dados relacionados & revisão bibliográfica.

2) Preparação dos Dados

Entendimento das fontes de dados, dos tipos e elaboração da representação.

3) Modelagem

Análise Exploratória, Seleção de atributos e treinamento.

4) Avaliação

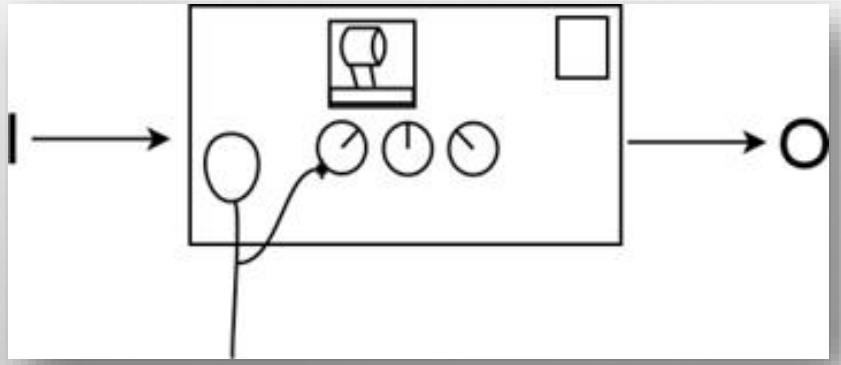
Seleção do melhor modelo.

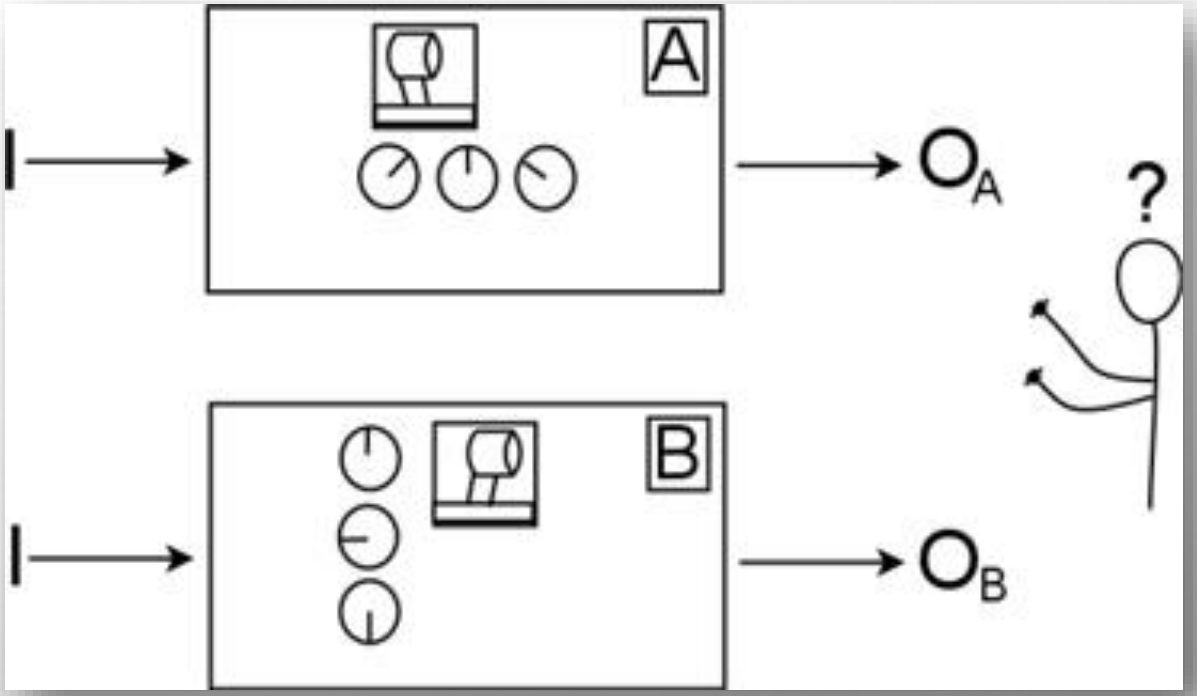
5) Liberação

Liberação do modelo no ambiente de produção.

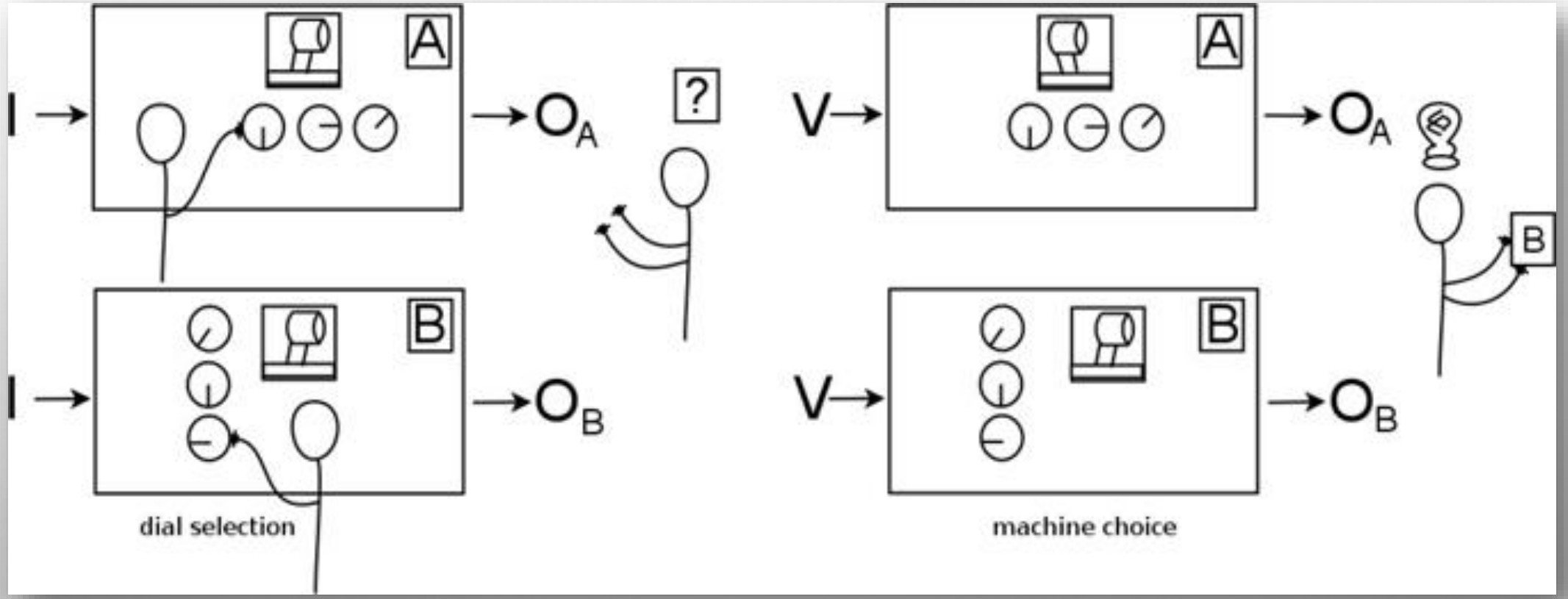
FÁBRICA DE MODELOS

PARÂMETROS E HIPERPARÂMETROS





PARÂMETROS E HIPERPARÂMETROS



1.1 Statistical framework

Assume that some data $\xi_1, \ldots, \xi_n \in \Xi$ with common distribution P are observed. Throughout the paper — except in Section 8.3 — the ξ_i are assumed to be independent. The purpose of statistical inference is to estimate from the data $(\xi_i)_{1 \leq i \leq n}$ some target feature s of the unknown distribution P, such as the mean or the variance of P. Let $\mathbb{S}$ denote the set of possible values for s.

The quality of $t \in \mathbb{S}$, as an approximation of s, is measured by its loss $\mathcal{L}(t)$, where $\mathcal{L} : \mathbb{S} \mapsto \mathbb{R}$ is called the *loss function*, and is assumed to be minimal for $t = s$. Many loss functions can be chosen for a given statistical problem.

Several classical loss functions are defined by

$$\mathcal{L}(t) = \mathcal{L}_P(t) := \mathbb{E}_{\xi \sim P}[\gamma(t;\xi)] \ , \tag{1}$$

where $\gamma : \mathbb{S} \times \Xi \mapsto [0, \infty)$ is called a *contrast function*. Basically, for $t \in \mathbb{S}$ and $\xi \in \Xi$, $\gamma(t;\xi)$ measures how well t is in accordance with observation of ξ, so that the loss of t, defined by (1), measures the average accordance between t and new observations ξ with distribution P. Therefore, several frameworks such as transductive learning do not fit definition (1). Nevertheless, as detailed in Section 1.2, definition (1) includes most classical statistical frameworks.

Another useful quantity is the *excess loss*

$$\ell(s,t) := \mathcal{L}_P(t) - \mathcal{L}_P(s) \geq 0 \ ,$$

which is related to the risk of an estimator $\widehat{s}$ of the target s by

$$R(\widehat{s}) = \mathbb{E}_{\xi_1, \ldots, \xi_n \sim P}[\ell(s, \widehat{s})] \ .$$

Classification corresponds to finite $\mathcal{Y}$ (at least discrete). In particular, when $\mathcal{Y} = \{0, 1\}$, the prediction problem is called *binary (supervised) classification*. With the 0-1 contrast function $\gamma(t;(x,y)) = \mathbb{1}_{t(x) \neq y}$, the minimizer of the loss is the so-called Bayes classifier s defined by

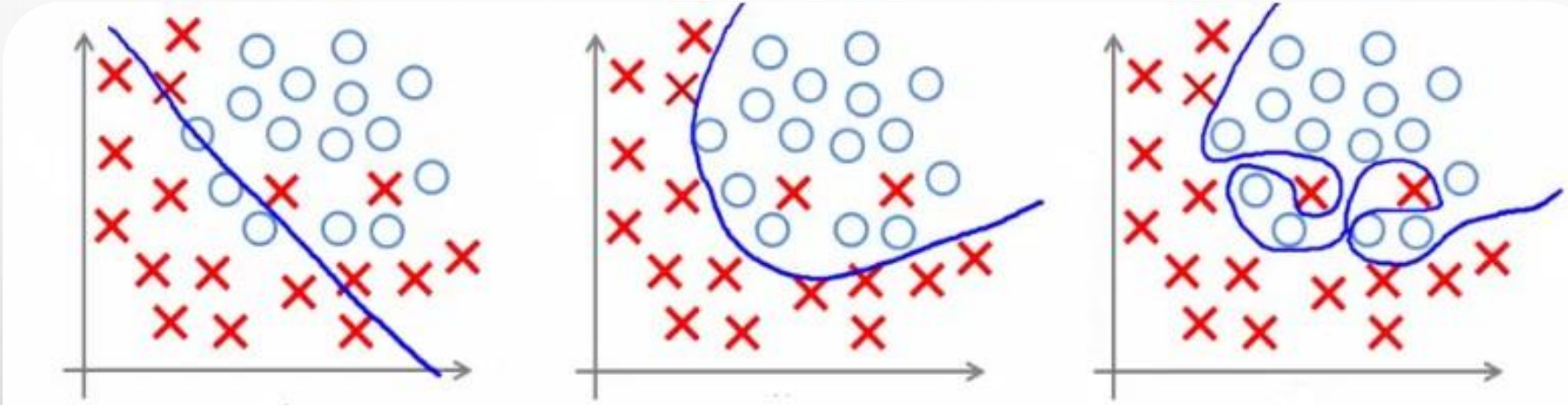
$$s(x) = \mathbb{1}_{\eta(x) \geq 1/2} \ ,$$

where η denotes the regression function $\eta(x) = \mathbb{P}_{(X,Y) \sim P}(Y = 1 \mid X = x)$.

Remark that a slightly different framework is often considered in binary classification. Instead of looking only for a classifier, the goal is to estimate also the confidence in the classification made at each point: $\mathbb{S}$ is the set of measurable mappings $\mathcal{X} \mapsto \mathbb{R}$, the classifier $x \mapsto \mathbf{1}_{t(x) \geq 0}$ being associated to any $t \in \mathbb{S}$. Basically, the larger $|t(x)|$, the more confident we are in the classification made from $t(x)$. A classical family of losses associated with this problem is defined by (1) with the contrast $\gamma_\phi(t;(x,y)) = \phi(-(2y-1)t(x))$ where $\phi : \mathbb{R} \mapsto [0, \infty)$ is some function. The 0-1 contrast corresponds to $\phi(u) = \mathbb{1}_{u \geq 0}$. The convex loss functions correspond to the case where ϕ is convex, nondecreasing with $\lim_{-\infty} \phi = 0$ and $\phi(0) = 1$. Classical examples are $\phi(u) = \max\{1 + u, 0\}$ (hinge), $\phi(u) = \exp(u)$, and $\phi(u) = \log_2(1 + \exp(u))$ (logit). The corresponding losses are used as objective functions by several classical learning algorithms such as support vector machines (hinge) and boosting (exponential and logit).

Many references on classification theory, including model selection, can be found in the survey by Boucheron et al. (2005).

ESCOLHA DA COMPLEXIDADE



Under-fitting

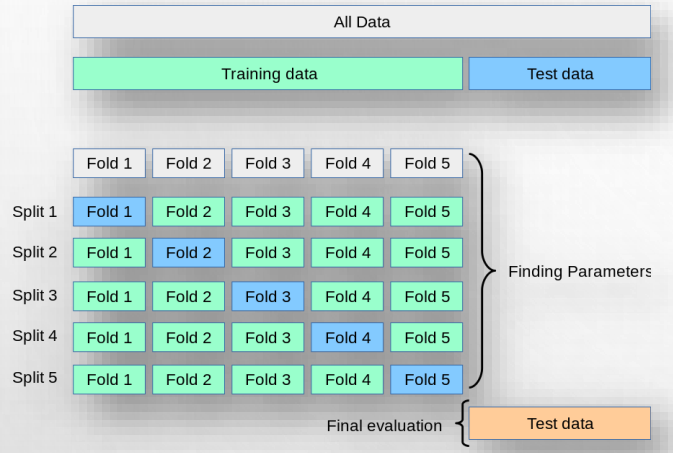
(too simple to explain the variance)

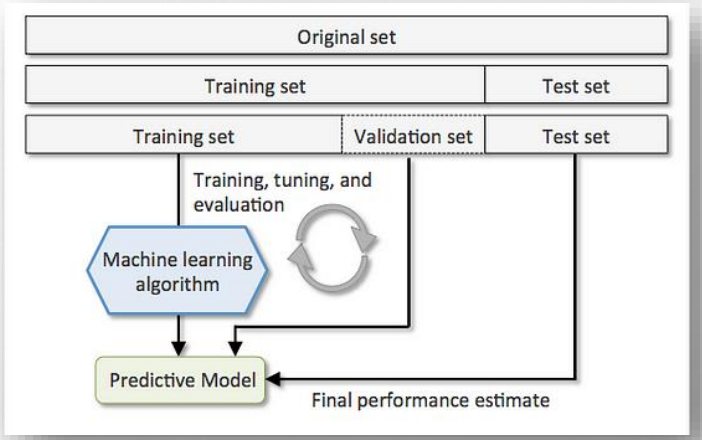
Appropriate-fitting

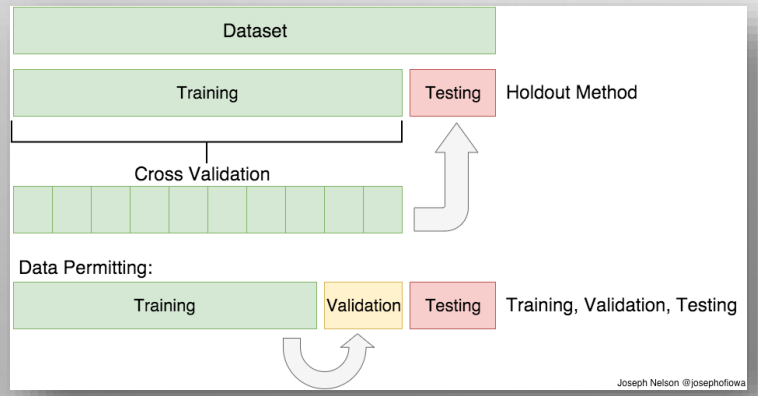
Over-fitting

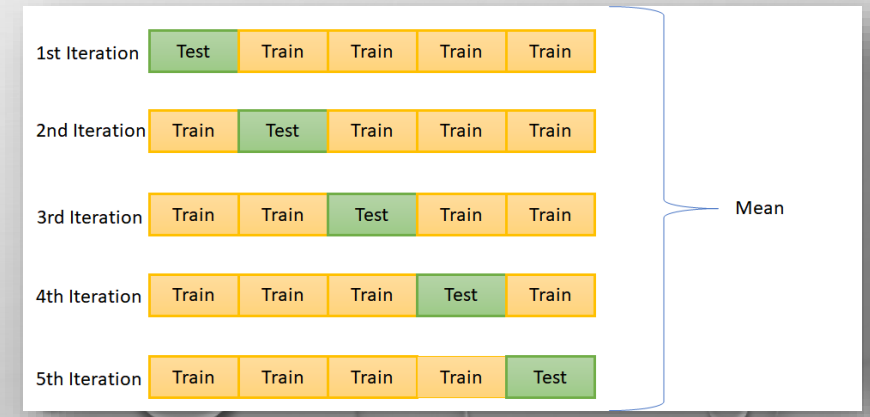
(forcefitting -- too good to be true)

GOOGLE: TRAIN, VALIDATION AND TEST DATASETS









HEURÍSTICAS

No contexto de otimização, uma heurística é uma estratégia prática usada para encontrar soluções suficientemente boas em tempo viável, especialmente quando encontrar a solução ótima seria muito caro ou inviável.

Em Machine Learning
Heurísticas de treinamento são **procedimentos empíricos** adotados para:

- Avaliar a capacidade de generalização de um modelo;
- Evitar overfitting/underfitting;
- Guiar decisões de otimização, como escolha de modelo ou hiperparâmetros.

CARACTERÍSTICA DO DATASET (VOLUME, TESTE SEPARADO)

+

ESTRATÉGIA DE SPLIT

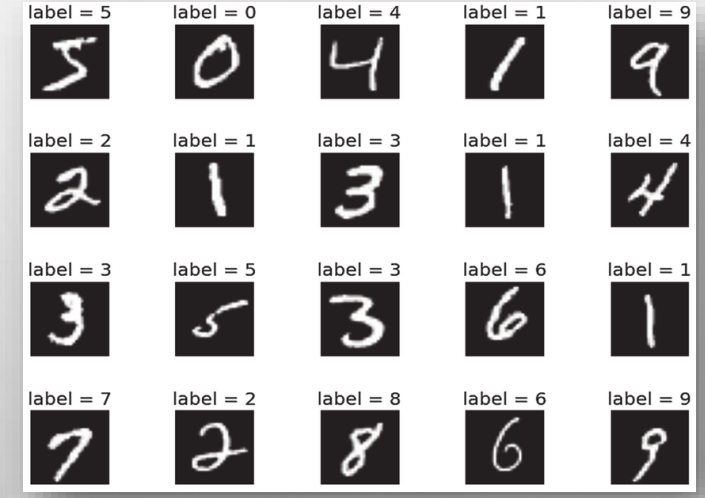
+

BUSCA NO ESPAÇO DE HIPERPARÂMETROS

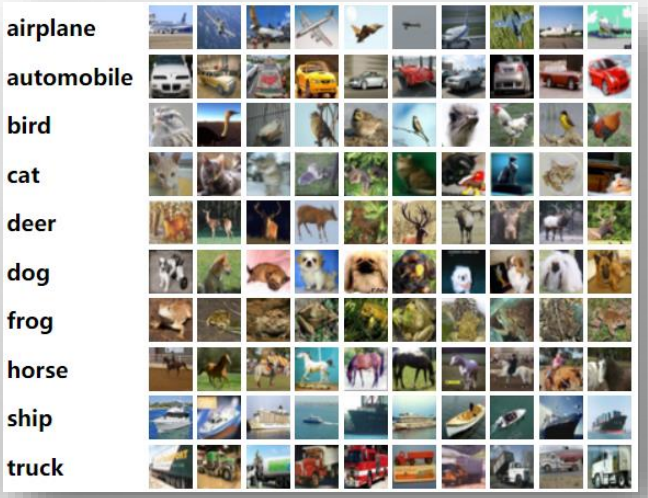
=

ALGORITMO DE TREINAMENTO!

DATASETS



MNIST



CIFAR

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. The...	positive
2	I thought this was a wonderful way to spend ti...	positive
3	Basically there's a family where a little boy ...	negative
4	Petter Mattei's "Love in the Time of Money" is...	positive
5	Probably my all-time favorite movie, a story o...	positive
6	I sure would like to see a resurrection of a u...	positive
7	This show was an amazing, fresh & innovative i...	negative
8	Encouraged by the positive comments about this...	negative
9	If you like original gut wrenching laughter yo...	positive

IMDB

CONJUNTOS

5.2.2.1 Learning Phases and Training Sets

Each of these three phases has a component of evaluation in it. In turn, each different evaluation makes use of a specific set of data containing different known input-output pairs. Let's give the phases and the datasets some useful names. Remember, the term *model* stands for our metaphorical factory machine. The phases are

1. *Assessment*: final, last-chance estimate of how the machine will do when operating in the wild
2. *Selection*: evaluating and comparing different machines which may represent the same broad type of machine (different k in k-NN) or completely different machines (k-NN and Naive Bayes)
3. *Training*: setting knobs to their optimal values and providing auxiliary side-tray information

The datasets used for these phases are:

1. Hold-out test set
2. Validation test set
3. Training set

We can relate these phases and datasets to the factory machine scenario. This time, I'll work from the inside out.

1. The training set is used to adjust the knobs on the factory machine.
2. The validation test set is used to get a non-taught-to-the-test evaluation of that finely optimized machine and help us pick between different optimized machines.
3. The hold-out test set is used to make sure that *the entire process of building one or more factory machines, optimizing them, evaluating them, and picking among them* is evaluated fairly.

SPLITTERS

LEAVE ONE OUT

- Uma única observação é deixada de fora a cada treinamento. N treinamentos são realizados para calcular a estatística de erro.

SINGLE SPLIT (GRUPO DE CONTROLE)

- Amostra é dividida entre treino e teste, mantendo um percentual das observações como grupo de teste externo ao treinamento.

K FOLDS

- Amostra é dividida em K conjuntos. K treinamentos são realizados, mantendo um conjunto como fora-da-amostra.

SHUFFLESPLIT

- Amostra é dividida em M conjuntos / treino e teste obedecendo uma proporção.

BOOTSTRAPPING

- O algoritmo itera, amostrando aleatoriamente M observações, para a quantidade Q desejada de treinamentos.

Splitters

`GroupKFold`	K-fold iterator variant with non-overlapping groups.
`GroupShuffleSplit`	Shuffle-Group(s)-Out cross-validation iterator.
`KFold`	K-Fold cross-validator.
`LeaveOneGroupOut`	Leave One Group Out cross-validator.
`LeaveOneOut`	Leave-One-Out cross-validator.
`LeavePGroupsOut`	Leave P Group(s) Out cross-validator.
`LeavePOut`	Leave-P-Out cross-validator.
`PredefinedSplit`	Predefined split cross-validator.
`RepeatedKFold`	Repeated K-Fold cross validator.
`RepeatedStratifiedKFold`	Repeated Stratified K-Fold cross validator.
`ShuffleSplit`	Random permutation cross-validator.
`StratifiedGroupKFold`	Stratified K-Fold iterator variant with non-overlapping groups.
`StratifiedKFold`	Stratified K-Fold cross-validator.
`StratifiedShuffleSplit`	Stratified ShuffleSplit cross-validator.
`TimeSeriesSplit`	Time Series cross-validator.
`check_cv`	Input checker utility for building a cross-validator.
`train_test_split`	Split arrays or matrices into random train and test subsets.

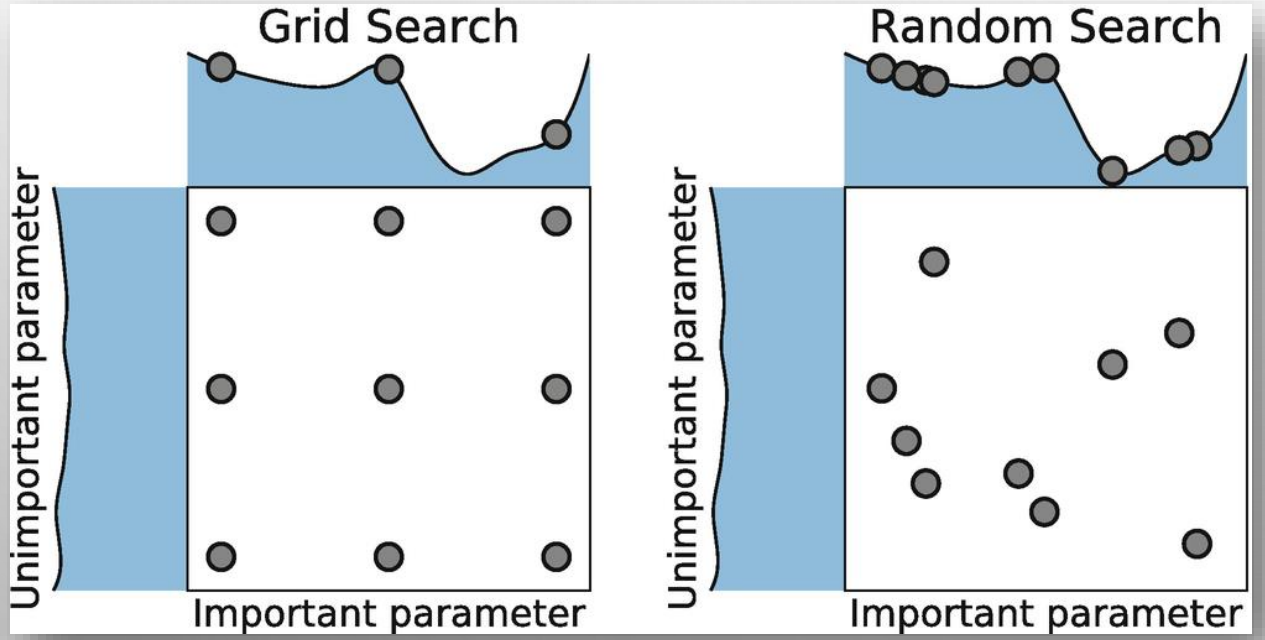
SKLEARN

QUAL SPLITTER USAR?

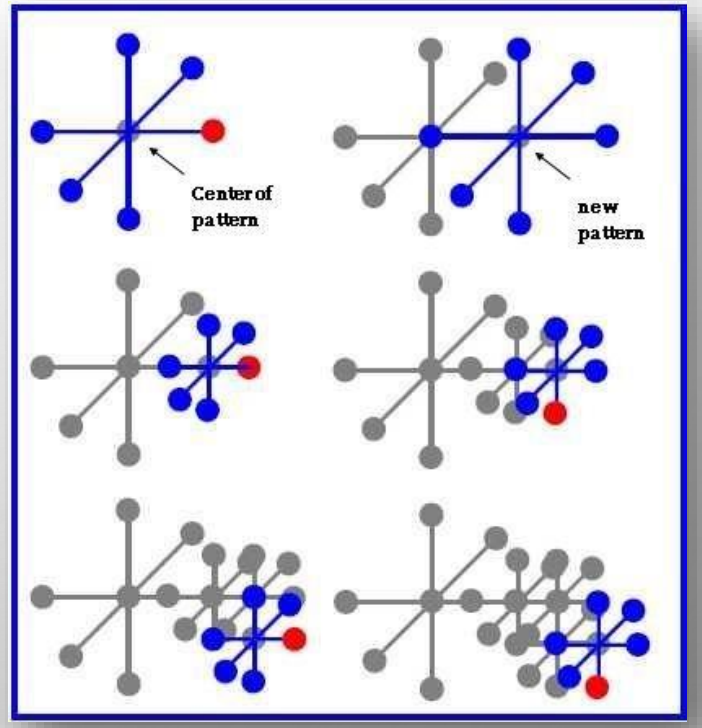
🎓 Estratégias baseadas no tamanho do dataset:

Volume de Dados	Estratégia Recomendada	Por quê?
Muito pequeno (≤ 1.000 amostras)	**Leave-One-Out** ou **K-Fold com K alto (ex: 10)**	Usa o máximo de dados para treino, sem desperdiçar informação.
Pequeno a médio (1.000 – 10.000)	**K-Fold Cross-Validation (K=5 ou 10)**	Equilíbrio entre avaliação estável e custo computacional.
Grande (> 10.000)	**Holdout simples (ex: 80% treino, 10% validação, 10% teste)**	Avaliação rápida, pouca variância, dados suficientes em cada parte.
Muito grande (> 1 milhão)	**Amostragem inteligente + Holdout**	Validação cruzada pode ser inviável; amostras bem escolhidas são suficientes.

OTIMIZAÇÃO DE HIPERPARÂMETROS



GRID SEARCH



PATTERN SEARCH

Hyper-parameter optimizers

`GridSearchCV`	Exhaustive search over specified parameter values for an estimator.
`HalvingGridSearchCV`	Search over specified parameter values with successive halving.
`HalvingRandomSearchCV`	Randomized search on hyper parameters.
`ParameterGrid`	Grid of parameters with a discrete number of values for each.
`ParameterSampler`	Generator on parameters sampled from given distributions.
`RandomizedSearchCV`	Randomized search on hyper parameters.

GridSearchCV

```
class sklearn.model_selection.GridSearchCV(estimator, param_grid, *, scoring=None,
n_jobs=None, refit=True, cv=None, verbose=0, pre_dispatch='2*n_jobs',
error_score=nan, return_train_score=False)
```

[source]

Exhaustive search over specified parameter values for an estimator.

Important members are fit, predict.

GridSearchCV implements a “fit” and a “score” method. It also implements “score_samples”, “predict”, “predict_proba”, “decision_function”, “transform” and “inverse_transform” if they are implemented in the estimator used.

The parameters of the estimator used to apply these methods are optimized by cross-validated grid-search over a parameter grid.

Read more in the User Guide.

RandomizedSearchCV

```
class sklearn.model_selection.RandomizedSearchCV(estimator, param_distributions, *,
n_iter=10, scoring=None, n_jobs=None, refit=True, cv=None, verbose=0,
pre_dispatch='2*n_jobs', random_state=None, error_score=nan,
return_train_score=False)
```

[source]

Randomized search on hyper parameters.

RandomizedSearchCV implements a “fit” and a “score” method. It also implements “score_samples”, “predict”, “predict_proba”, “decision_function”, “transform” and “inverse_transform” if they are implemented in the estimator used.

The parameters of the estimator used to apply these methods are optimized by cross-validated search over parameter settings.

In contrast to GridSearchCV, not all parameter values are tried out, but rather a fixed number of parameter settings is sampled from the specified distributions. The number of parameter settings that are tried is given by n_iter.

If all parameters are presented as a list, sampling without replacement is performed. If at least one parameter is given as a distribution, sampling with replacement is used. It is highly recommended to use continuous distributions for continuous parameters.

Read more in the User Guide.

RETREINO

- USAR TREINO + (VALIDAÇÃO) + MELHORES HIPERPARÂMETROS
- VALIDAR USANDO A BASE DE TESTES OU TODOS OS DADOS, CASO NÃO TENHA HOLDOUT

HEURÍSTICAS JÁ IMPLEMENTADAS EM SALA

- CLASSIFICAÇÃO BINÁRIA IRIS SINGLE SPLIT K SELECIONADO MANUALMENTE
  - CLASSIFICAÇÃO IRIS KNN
- CLASSIFICAÇÃO BINÁRIA IRIS KFOLDS (TREINO/TESTE) K GRID UNIDIMENSIONAL (HALF LEARNING)
  - CLASSIFICAÇÃO IRIS KNN KFOLDS
- CLASSIFICAÇÃO MULTICLASSE IRIS LEAVE ONE OUT K GRID UNIDIMENSIONAL X NÚMERO DE FEATURES COM RE-TREINO SINGLE SPLIT
  - CLASSIFICAÇÃO IRIS KNN LOO

Scenario	Example	Good	Bad	Risk
high bias & low variance	more neighbors	resists noise	misses pattern	un-derfit
	low-degree polynomial	forced to generalize		
	smaller or zero lin-ear regression coefficients			
	more indepen-dence assumptions			
low bias & high variance	fewer neighbors	follows complex patterns	follows noise	over-fit
	high-degree polynomial		memorizes training data	
	bigger linear re-gression coefficients			
	fewer indepen-dence assumptions			

BATALHA DE VIZINHOS II