



What Is Machine Learning? Framing a Problem

Machine learning | Supervised, unsupervised, self-supervised and reinforcement learning, framing a task, baselines, bias-variance in plain words, no free lunch

#ML #ProblemFraming #Baselines #BiasVariance #scikit-learn

What machine learning is

Machine learning



Instead of writing the rules, you show the computer many examples with the right answers and it works out a rule that also works on new examples.

Pick a model family, a loss to minimise and data split into train, validation and test; fit on train, choose on validation, report on test.

- features in, prediction out
- judged on unseen data only



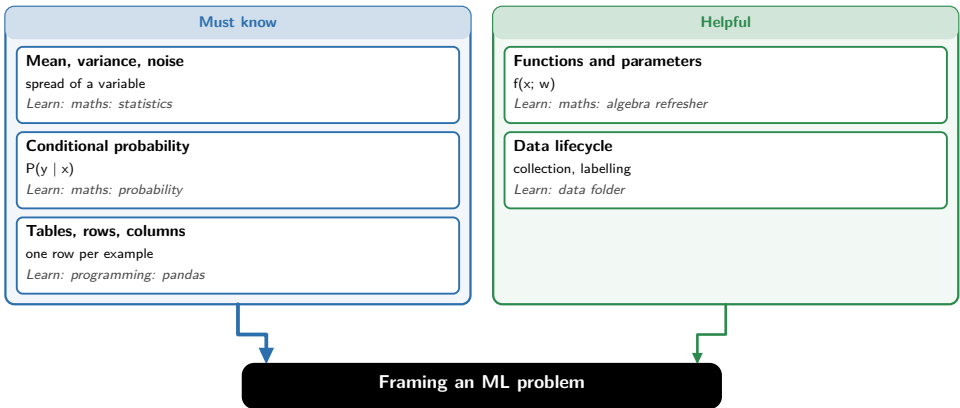
Empirical risk minimisation over a hypothesis class with an inductive bias; generalisation depends on capacity, sample size and how well training data matches deployment.

- representation + evaluation + optimisation
- distribution shift breaks the guarantee

PICTURE IT

Like learning to recognise ripe fruit by handling hundreds of them rather than reading a rulebook: you end up with a sense you cannot fully write down, and it fails on a fruit you have never seen.

Machine learning told three ways. Start with the top bubble; the others add what practitioners and experts care about



What to know before this sheet

Hand-written rules

if 'free money' in text: spam

Expert writes and maintains each rule

Breaks when spammers change wording

Fully explainable, brittle

Best when rules are few, stable and known

Machine learning

`clf.fit(emails, labels)`; the weights of 50 000 words are learned

Expert labels examples and picks features and metric

Retrain on new examples; still drifts

Handles thousands of weak signals, needs monitoring

Best when the pattern is fuzzy, large or changing

The four learning paradigms

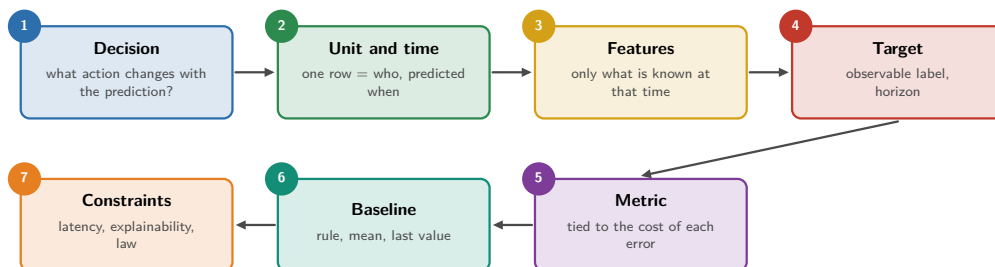
Category	Signal	Typical tasks	Classical tools
Supervised	▪ labels given by people or logs	▪ classification ▪ regression ▪ ranking	▪ linear models ▪ trees, boosting ▪ SVM, k-NN
Unsupervised	▪ the data's own structure	▪ clustering ▪ dimension reduction ▪ anomaly detection	▪ k-means, HDBSCAN ▪ PCA, UMAP ▪ Isolation Forest
Self-supervised	▪ labels cut from the data itself	▪ next-token prediction ▪ masked filling ▪ contrastive pairs	▪ word2vec ▪ BERT, GPT pretraining ▪ SimCLR, CLIP
Reinforcement	▪ a reward after actions	▪ games, control ▪ recommendation loops ▪ LLM post-training	▪ Q-learning ▪ policy gradients, PPO ▪ bandits

Where the learning signal comes from decides the paradigm

Classification output is a category Tell: Can the answer be averaged meaningfully (a price) or only counted (a class)?	vs	Regression output is a number
Self-supervised labels derived automatically Tell: Is there a prediction target, even a manufactured one?	vs	Unsupervised no target at all
Multiclass exactly one of k labels Tell: Can one example carry two labels at once?	vs	Multilabel any subset of labels
Reinforcement learning reward after actions, delayed Tell: Is the right answer shown, or only a score for what you did?	vs	Supervised learning correct answer given per example
Parameter learned by fit Tell: Does the training algorithm set it, or do you?	vs	Hyperparameter chosen before fit

Look-alike notions and the one question that separates them

© Framing a problem



Seven decisions before any model. Skipping step 3 causes most leakage; skipping step 6 makes success unprovable

Worked example: framing late-delivery prediction

PROBLEM	An online shop wants to warn customers whose parcel will arrive late. Frame it as an ML task.
GIVEN	2 years of orders, 1.2 M rows - late = delivered after the promised date (9 % of orders) - warnings are sent at checkout
1 Decision and unit	one row per order, scored at checkout <i>Why: the action (warn, offer express) happens at checkout</i>
2 Features	carrier, warehouse, distance, weekday, stock, weather forecast <i>Why: known at checkout; 'actual pickup time' is not and would leak</i>
3 Target and horizon	y = 1 if delivered > promised date <i>Why: observable within 2 weeks, so labels arrive fast</i>
4 Task type	binary classification with probabilities <i>Why: the warning needs a yes/no, the threshold a probability</i>
5 Metric	PR-AUC; recall at 5 % alert rate <i>Why: positives are 9 %; too many warnings annoy customers</i>
6 Baseline	rule: carrier X on Fridays → late; recall 0.31 <i>Why: the model must beat what operations already know</i>
ANSWER	Binary classifier at checkout on checkout-time features, judged by PR-AUC and recall at a 5 % alert rate against the carrier-day rule
CHECK	✓ Every feature passes the test: could I compute it at checkout time on the day?

```

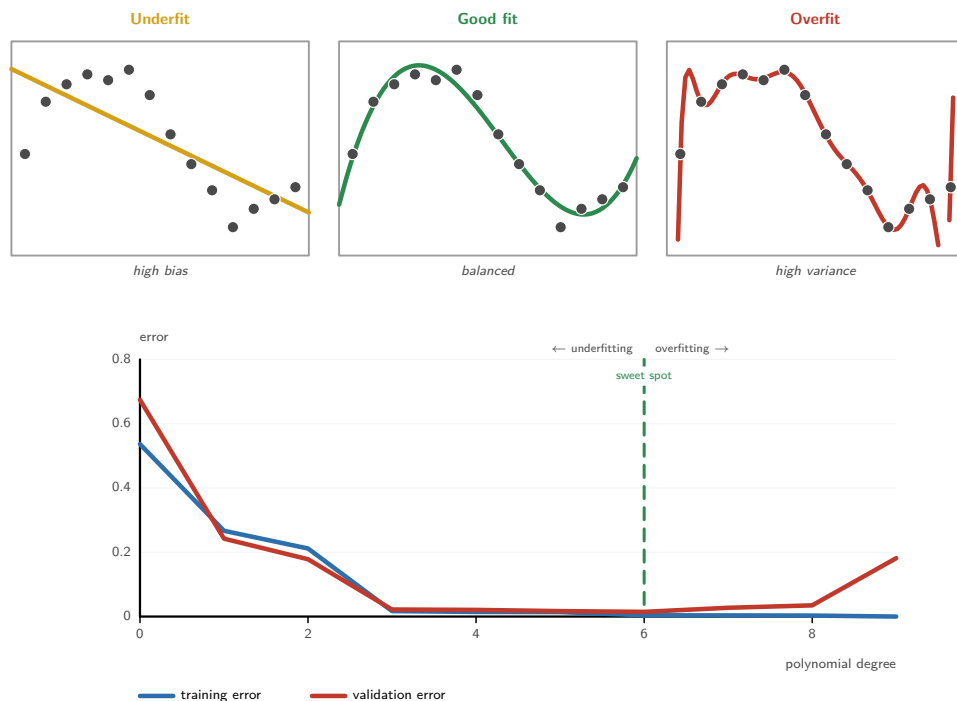
Baselines in two lines: DummyClassifier and DummyRegressor
from sklearn.dummy import DummyClassifier, DummyRegressor
from sklearn.model_selection import cross_val_score

base = DummyClassifier(strategy="most_frequent") # or "prior",
print(cross_val_score(base, X, y, cv=5,          # "stratified"
                      scoring="average_precision").mean())
reg = DummyRegressor(strategy="median")          # or "mean"
  
```

Task	Baseline	What it tells you
Classification	majority class; class prior	the score of knowing nothing (accuracy = majority share)
Regression	mean or median of the train target	$R^2 = 0$ by definition; MAE of the median
Time series	last value; same value last season	the 'naive' and 'seasonal naive' forecasts (sheet 13)
Ranking, recommendation	most popular items	popularity is hard to beat (sheet 14)
Any	simple domain rule; logistic regression	how much the complex model really adds

Baselines by task: report them next to every model score

📊 Bias, variance and generalisation



Same noisy points, growing model complexity. Left: high bias (misses the curve). Right: high variance (chases noise). The validation curve bottoms out in between

Quantity	Formula	Plain words
Expected error	$E[(y - \hat{f}(x))^2] = \text{Bias}^2 + \text{Var} + \sigma^2$	averaged over training sets and noise
Bias	$\text{Bias} = E[\hat{f}(x)] - f(x)$	how far the average model is from the truth
Variance	$\text{Var} = E[(\hat{f}(x) - E[\hat{f}(x)])^2]$	how much the model moves between training sets

where y = observed target, $y = f(x) + \text{noise}$ (target unit); $f(x)$ = true (unknown) function (target unit); $\hat{f}(x)$ = prediction of a model trained on one random training set (target unit); $E[\cdot]$ = average over random training sets (and noise) (-); σ^2 = variance of the noise, the irreducible error (target unit squared)
Bias-variance decomposition of the expected squared error at a point x (squared loss only)

📖 Reading the decomposition with numbers

Price model, errors in (k EUR)². Linear model: bias² 9 + variance 1 + noise 4 = 14. Deep unpruned tree: bias² 1 + variance 10 + noise 4 = 15. Random forest of 500 trees: bias² 1.5 + variance 2 + noise 4 = 7.5. Averaging trees cut the variance; nothing can remove the noise term 4 except new, informative features.

Symptom	Diagnosis	What helps
train and validation error both high, close	high bias (underfit)	more features, more flexible model, less regularisation, train longer
train error low, validation much higher	high variance (overfit)	more data, regularisation, simpler model, bagging, early stopping
validation error good, production bad	distribution shift or leakage	time-based split, monitor drift, audit features (sheet 08)
both errors stuck near the same floor	noise or missing information	better labels, new data sources; not a bigger model

Diagnosis and remedies

💡 Beyond the classical U-curve

Very over-parameterised models (large neural nets, boosted ensembles with many trees and a small learning rate) can keep improving past the interpolation point: the 'double descent' effect, covered in deep-learning. For the models of this folder, the U-shaped validation curve and early stopping remain the working rule.

⚡ No free lunch and inductive bias

No free lunch (Wolpert, 1996, for supervised learning): averaged uniformly over every possible target function, all learning algorithms have the same expected error on unseen points. Practical meaning: there is no universally best model; an algorithm wins when its **inductive bias** matches the structure of real problems, which are far from uniform. That is why you compare a few model families on validation data instead of trusting a ranking.

Model	Inductive bias (what it assumes)	Wins when
Linear / logistic regression	output is a weighted sum of features	relations roughly additive, few rows, need to explain
k-NN	similar inputs have similar outputs	low dimension, good distance, smooth target
Naive Bayes	features independent given the class	text, very many sparse features, little data
SVM with RBF kernel	smooth boundary with a wide margin	medium data, clean features
Decision trees, boosting	axis-aligned thresholds, interactions	tabular data, mixed types, non-linear
CNN, transformer	locality, translation, attention patterns	images, text, audio at scale

Each model family bets on a different structure

MYTH

✗ The model learns the truth from the data

✗ Overfitting means too many features

✗ Unsupervised learning needs no human judgement

✗ A better algorithm fixes bad labels

✗ No free lunch means all algorithms are equal in practice

FACT

✓ It learns correlations in the sample it saw
If the training data differs from deployment (other country, season, product), the learned rule may not hold.

✓ Overfitting means too much capacity for the data
Many features with strong regularisation can generalise; a deep tree on 3 features can overfit.

✓ It needs more
Without labels nobody tells you a clustering is right; the choice of features and distance decides the result.

✓ Label noise sets the error floor
If annotators disagree on 10 % of cases, no model reliably beats about 90 % agreement with them.

✓ Real problems have structure, so some algorithms are much better on them
The theorem averages over all possible functions, most of which look like noise.

Misconceptions that derail first projects

✓ Recap and self-test

Remember

01

Learning signal sets the paradigm

Labels: supervised; none: unsupervised; derived: self-supervised; reward: RL

02

Frame before you model

unit, prediction time, features known then, target, metric, baseline

03

Always beat a baseline

DummyClassifier, mean, last value, a business rule

04

Error = bias² + variance + noise

diagnose with train vs validation error

05

No universally best model

match the inductive bias to the data, compare on validation

Q1 Give the learning signal of each paradigm: supervised, unsupervised, self-supervised, RL.

A Human or logged labels; the data's own structure; labels cut from the data (masked word, next token); a reward after actions.

Q2 Name the seven framing decisions.

A Decision supported, unit and prediction time, features known at that time, target and horizon, metric, baseline, constraints.

Q3 Training error 2 %, validation error 15 %: diagnosis and two fixes?

A High variance (overfit): more data, regularisation, simpler model, bagging or early stopping.

Q4 Training 14 %, validation 15 %, target 5 %: diagnosis?

A High bias (underfit): add features or a more flexible model, reduce regularisation.

Q5 Write the bias-variance decomposition.

A Expected squared error = bias² + variance + σ^2 (irreducible noise), for squared loss.

Q6 What does the no free lunch theorem say, and what does it not say?

A Averaged over all possible problems all algorithms tie; it does not say they tie on real, structured problems.

Q7 Which DummyClassifier strategy gives the accuracy of always predicting the majority class?

A strategy="most_frequent"; "prior" also predicts it but returns the class priors as probabilities.

Q8 Why is 'actual pickup time' a bad feature for a checkout-time delay model?

A It is only known after checkout, so using it is leakage: great offline scores, impossible in production.