Solution Validation

Data Science Project: An Inductive Learning Approach

Prof. Dr. Filipe A. N. Verri

About these slides

These slides are companion material for the book

Data Science Project: An Inductive Learning Approach

Prof. Dr. Filipe A. N. Verri

`https://leanpub.com/dsp`

All intellectual content comes from the book and is not AI-generated. Slides were produced with the assistance of Claude Code.

All models are wrong, but some are useful.

— George E. P. Box, Robustness in Statistics

Overview

Contents

- Evaluation
- An experimental plan for data science

Objectives

- Understand the importance of experimental planning
- Learn the main evaluation metrics
- Learn how to design an experimental plan

Evaluation

Evaluation setup

1. **Preprocessing**: apply F to training set $\rightarrow$ fitted preprocessor f_ϕ
2. **Learning**: train machine M on adjusted training data $\rightarrow$ model f_θ
3. **Transformation**: apply f_ϕ to test set (no access to target y)
4. **Prediction**: $\widehat{y}_i = f_\theta(f_\phi(\vec{x}_i))$
5. **Evaluation**: compare $\widehat{y}_i$ with y_i on test set

Training and test sets must be **disjoint**: $\mathcal{I}_{\text{train}} \cap \mathcal{I}_{\text{test}} = \emptyset$

Confusion matrix

		Predicted	
		1	0
Expected	1	TP	FN
	0	FP	TN

- **TP**: true positives
- **TN**: true negatives
- **FP**: false positives
- **FN**: false negatives

Accuracy and balanced accuracy

Accuracy — proportion of correct predictions:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Balanced accuracy — average of per-class rates:

$$\text{Balanced Accuracy} = \frac{\text{TPR} + \text{TNR}}{2}$$

where $\text{TPR} = \frac{\text{TP}}{\text{TP}+\text{FN}}$ and $\text{TNR} = \frac{\text{TN}}{\text{TN}+\text{FP}}$.

Accuracy can be misleading with imbalanced classes; balanced accuracy gives equal weight to each class.

Precision and recall

Both focus on the **positive class**:

Precision — confidence in positive predictions:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Recall (TPR) — completeness of positive retrieval:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Precision: avoid false alarms. Recall: avoid missing positives.

F-score and specificity

F-score — weighted harmonic mean of precision and recall:

$$\mathrm{F}_\beta\text{-score} = \frac{(1+\beta^2)\cdot \mathrm{Precision}\cdot \mathrm{Recall}}{\beta^2\cdot \mathrm{Precision} + \mathrm{Recall}}$$

$\beta = 1$: equal weight. $\beta > 1$: favor precision. $0 < \beta < 1$: favor recall.

Specificity (TNR) — focuses on the negative class:

$$\mathrm{Specificity} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}$$

Summary of classification metrics

Metric	Focus	Interpretation
Accuracy	Symmetrical	Penalizes all
Balanced Accuracy	Symmetrical	Penalizes all (weighted)
Recall (TPR)	Positive	Penalizes FN
Precision	Positive	Penalizes FP
F-score	Positive	Penalizes all (weighted)
Specificity (TNR)	Negative	Penalizes FP

Metrics for trivial classifiers

Metric	Guess 1	Guess 0	Random
Accuracy†	π	$1-\pi$	0.5
Balanced Accuracy	0.5	0.5	0.5
Recall (TPR)	1	0	0.5
Precision†	π	—	π
F_1-score†	$\frac{2\pi}{1+\pi}$	0	$\frac{2\pi}{1+2\pi}$
Specificity (TNR)	0	1	0.5

π = ratio of positive samples. † = affected by class imbalance.

Prefer asymmetric metrics when the positive class is the **minority**.

Regression metrics — absolute errors

Residual: $\epsilon_i = \hat{y}_i - y_i$

Mean Absolute Error: $\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |\epsilon_i|$

Mean Squared Error: $\text{MSE} = \frac{1}{n}\sum_{i=1}^{n} \epsilon_i^2$

Root Mean Squared Error: $\text{RMSE} = \sqrt{\text{MSE}}$

MAE and RMSE are in the same unit as the target variable. MSE/RMSE penalize large errors more.

Regression metrics — relative errors

For strictly positive targets ($y_i > 0$, $\hat{y}_i > 0$):

Mean Absolute Percentage Error: $\text{MAPE} = \dfrac{1}{n}\sum_{i=1}^{n} \dfrac{|\epsilon_i|}{y_i}$

Mean Absolute Logarithmic Error: $\text{MALE} = \dfrac{1}{n}\sum_{i=1}^{n} |\ln \hat{y}_i - \ln y_i|$

Useful when target has large range. MAPE punishes overestimation more. MALE is symmetric in multiplicative terms: $|\ln \frac{\hat{y}}{y}| = \ln \max\left(\frac{\hat{y}}{y}, \frac{y}{\hat{y}}\right)$.

MAPE vs. MALE

$\widehat{y}$	y	$\lvert\epsilon\rvert$	MAPE	exp(MALE)
100	10	90	9.0	10
1	10	9	0.9	10

MAPE gives 9.0 for overestimation and 0.9 for underestimation.
MALE treats both as a factor of 10.

Probabilistic classification

Regressor output $f_R(\vec{x}) \in [0, 1]$ converted to classifier:

$$f_C(\vec{x}; \tau) = \begin{cases} 1 & \text{if } f_R(\vec{x}) \geq \tau \\ 0 & \text{otherwise} \end{cases}$$

- Low $\tau \approx 0$: high recall, low specificity
- High $\tau \approx 1$: high specificity, low recall
- Default: $\tau = 0.5$

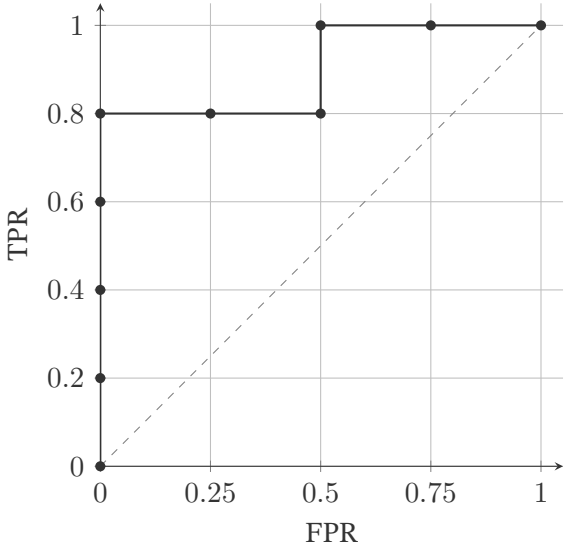
Example: probability regressor output

Expected	Predicted
0	0.1
0	0.5
0	0.2
0	0.6
1	0.4
1	0.9
1	0.7
1	0.8
1	0.9

Sort by predicted value, then compute TPR and FPR for each threshold:

Exp.	Threshold	TPR	FPR
—	∞	$0/5$	$0/4$
1	0.9	$2/5$	$0/4$
1	0.8	$3/5$	$0/4$
1	0.7	$4/5$	$0/4$
0	0.6	$4/5$	$1/4$
0	0.5	$4/5$	$2/4$
1	0.4	$5/5$	$2/4$
0	0.2	$5/5$	$3/4$
0	0.1	$5/5$	$4/4$

ROC curve



Points above diagonal — better than random AUC = 0.9 in this example

Area under the ROC curve (AUC)

- Summarizes the performance of all possible thresholds
- Ranges from 0 to 1 (1 = perfect)
- **Scale invariant**: measures how well predictions are ranked
- **Robust to class imbalance**: considers both recall and specificity

An experimental plan for data science

Why experimental planning?

- Data science is **experimental** — we lack theoretical models predicting algorithm performance
- Data and learning are **stochastic**
- Robust planning ensures reliable, decision-worthy results

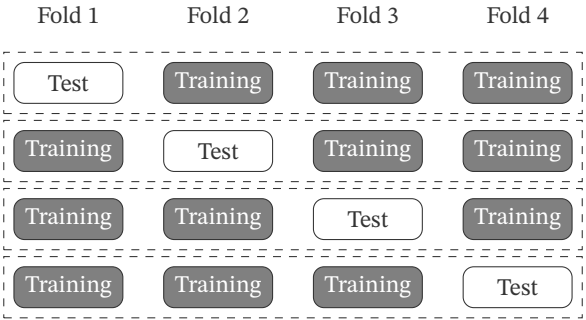
Key elements:

- **Hypothesis**: what do we want to validate?
- **Data**: representative dataset
- **Solution search**: preprocessing + learning
- **Performance metric**: how do we measure success?

Performance as a random variable

- Training and testing on the same data $\Rightarrow$ **optimistic**
- Single hold-out $\Rightarrow$ depends on the random split
- Solution: treat performance as a **random variable**
- Generate multiple training/test splits to study the distribution

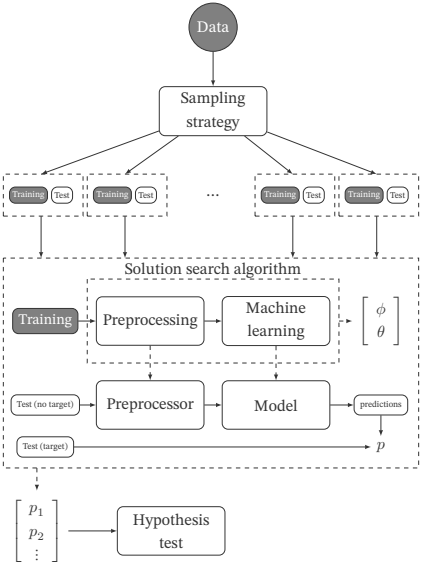
Cross-validation



r folds; each fold is the test set once, training set $r-1$ times.

Prefer: **repeated** and **stratified** cross-validation.

Experimental plan



Collecting evidence — summary

For each sampling D_k:

1. Fit preprocessor ϕ_k and model θ_k on $D_{k,\text{train}}$
2. Predict $\hat{y}_i = f_{\phi_k,\theta_k}(x_i)$ on $D_{k,\text{test}}$
3. Compute performance $p_k = R(\vec{y}, \hat{\vec{y}})$

By definition, p_k is free of **leakage**: parameters found without test data, predictions use only x_i (no target y_i).

Evaluation vs. validation

- **Evaluation**: assessing performance using a test set
- **Validation**: interpreting the evaluation results to determine how well the solution generalizes to unseen data

The sampled performance values $p_1, p_2, \ldots$ can be analyzed statistically to prove (or disprove) the hypothesis.

Estimating expected performance

Goal: reach performance threshold p^*.

Simple approach:

- Run 10×10-fold cross-validation
- Compute $\bar{p}$ and σ
- If $\bar{p} - \sigma \gg p^*$, the solution is likely good enough

More sophisticated: **Bayesian analysis** to estimate the probability distribution of performance.

Bayesian correlated t-test

Let $z_k = p_k - p^*$ (performance gain over goal).

Generative model[1]:

$$\vec{z} = \vec{1}\mu + \vec{v}, \quad \vec{v} \sim \text{MVN}(0, \Sigma)$$

with $\Sigma_{ii} = \sigma^2$ and $\Sigma_{ij} = \sigma^2\rho$ for $i \neq j$.

Posterior of μ is a Student distribution:

$$\mu \mid \vec{z} \sim \text{St}\Big(n{-}1,\ \bar{z},\ \Big(\tfrac{1}{n} + \tfrac{\rho}{1-\rho}\Big)\, s^2\Big)$$

Validate: $\text{P}(\mu > 0 \mid \vec{z}) > \gamma$ (confidence level).

[1] A. Benavoli et al. (2017). **"Time for a Change: a Tutorial for Comparing Multiple Classifiers Through Bayesian Analysis"**. In: *Journal of Machine Learning Research* 18.77, pp. 1–36. URL: http://jmlr.org/papers/v18/16-305.html.

Comparing strategies

Baseline $A(\lambda_0)$ vs. candidate $A(\lambda)$:

- Use the **same samplings** (paired performance samples)
- Let $\vec{z} = \vec{p}(\lambda) - \vec{p}(\lambda_0)$
- Validate: $\mathrm{P}(\mu > 0 \mid \vec{z}) > \gamma$
- $\mu > 0$: candidate is better

Iterate: compare $A(\lambda_1)$ vs. best so far, keep the winner.

If confidence is not reached but $\mu > 0$: consider interpretability, computational cost, or ease of implementation. Always check if $\mathrm{P}(\mu < 0 \mid \vec{z})$ is acceptable.

Nesting experiments

- Hyperparameters λ can be optimized with a **nested** experimental plan (e.g., grid search)
- Inner loop finds best λ; outer loop estimates performance
- Never use inner choices for production decisions

Trade-offs:

- Computationally expensive (combinations multiply)
- Inner dataset is smaller $\Rightarrow$ less reliable estimates
- Alternative: unnest by comparing options two by two

Final remarks

- Framework assumes data is **i.i.d.**
- Time-series, spatial data $\Rightarrow$ different sampling strategies
- Changing the sampling strategy also changes the validation method

Takeaways

- Evaluation metrics should be chosen according to project goals
- The experimental plan gathers performance data systematically
- A hypothesis test validates the results
- Performance is a **random variable** — study its distribution, not a single value
- Same samplings must be used when comparing algorithms

Questions?