Machine learning pitfalls

SEVERITY LOW ▼ ● ▲ HIGH

A non-exhaustive list of pitfalls in planning, executing, and implementing machine learning projects. Abbreviations: **SOTA** state of the art

CATEGORY	PITFALL		CONSEQUENCES	MITIGATION
Design	**Asking the wrong question**	▲	Useless model, low trust	Consult user/customer, ask more questions
	Identifying the wrong task	▲	Poor performance, low trust	Challenge assumptions
	No baseline/SOTA performance	●	Poor performance, low trust	Measure SOTA performance
	No success criterion	●	Poor performance, low trust	Ask user/customer what success means
Data	**Poor quality features**	●	Poor performance	Find better data
	Poor quality labels	●	Poor performance	Find or make better labels
	Unrecognized non-independent records	●	Leads to leakage via improper splitting	Careful splitting, exploit correlation
	Unrecognized class imbalance	●	Poor performance on minority classes	Balance classes, better evaluation metrics
	Hidden stratification	●	Poor performance on important examples	Ask user/customer, monitor performance
	Data not representative	●	Leads to out-of-distribution application	Find representative data
	Spurious/noncausal correlations	●	Leads to leakage	Pay attention to causality
	Missing explanatory variables	●	Poor performance	Pay attention to causality
Leakage	**Using features not available in application**	▲	Overoptimism	Examine real application data
	Improper splitting: scaling	●	Overoptimism	Code review
	Improper splitting: correlations	●	Overoptimism	Code review
	Improper splitting: augmentation	●	Overoptimism	Code review
Modeling	**Poor choice of algorithm**	●	Poor performance, poor explainability	Develop understanding, code review
	Lack of understanding of algorithm	●	Poor performance, poor explainability	Develop understanding
	Poor choice of hyperparameters	●	Poor performance, poor explainability	Develop understanding, code review
	No or inappropriate feature scaling	●	Poor performance, no convergence	Develop understanding, code review
	Inappropriate feature encoding	●	Poor performance	Develop understanding, code review
	Poor preprocessing: imputation	●	Poor performance	Develop understanding, code review
	Poor preprocessing: outlier elimination	●	Overoptimism, poor explainability	Develop understanding, code review
	Poor hyperparameter tuning	●	Poor performance, slow or no convergence	Develop understanding, code review
Underfitting	**No or inappropriate basis expansion**	●	Poor performance	Develop understanding, code review
	No data augmentation	●	Poor performance	Develop understanding, code review
Overfitting	**Too many parameters**	●	Overoptimism, poor explainability	Choose simpler algorithms
	Too many features	▼	Overoptimism, poor explainability	Dimensionality reduction
	No regularization	●	Overoptimism, poor performance	Use regularization, or equivalent
	Insufficient data	▼	Overoptimism, poor performance	Get more data
Evaluation	**Over-using test set**	●	Leads to overfitting	Use more splits
	Wrong metric	●	Overoptimism, overlooking minority classes	Choose better metric
	Not looking at variance	●	Poor understanding of performance	Use folded cross-validation
	Not evaluating residuals (in regression)	●	Spurious model	Examine residuals
	Not comparing train and test scores	●	Poor understanding of performance	Compute training scores
Application	**Unscaled input**	▲	Wildly spurious predictions	Integrate scaler into modeling pipeline
	Covariate (feature) shift: new P(X)	▲	Poor performance	Monitor incoming feature distribution
	Label shift: new P(y)	▲	Poor performance	Test future model performance
	Concept drift (posterior shift): new P(y|X)	▲	Poor performance	Monitor model performance
	Nonstationary input (in time or space)	●	Poor performance, drift over time or space	Test future or local model performance
Deployment	**No consideration of complex system**	▲	Unintended consequences, low trust	Talk to users/customers
	No contingency	●	Low trust	Plan ahead with users/customers
	No training of users	▲	Low impact, low trust	Train users/customers
	No documentation	●	Inappropriate application, low trust	Document the modeling process & product
Engineering	**No code or version control**	▲	Technical debt, high chance of error	Use version control
	No data versioning	▲	High chance of error	Use version control
	No tests of critical code	▲	High chance of error	Write tests
Governance	**No quality control**	▲	High chance of error	Integrate QC into development process
	Using protected features	▲	Potential unfair bias	Removed protected features
	Violates regulations or ethics	▲	Legal risk	Consult professional services