Supervised Learning

Chapter IX: Evaluation

Johannes Jurgovsky

Outline

Evaluation

1. Introduction

2. Performance Measures

3. Performance Estimation

1. Introduction

Introduction:

Motivation

Setup in Supervised Learning: Learn a classifier $h : X \to Y$ from training data such that we can accurately predict the target value of a new (unseen) observation.

Problem: The true (physical) process $y : X \to Y$ that *generates* observations is unknown to us. We only have access to a finite data set of observations.

Choices: Different learning algorithms produce different classifiers. They differ in:

- Inductive bias: e.g. linear decision boundary, axis-aligned decision boundary, etc.
- Set of hyper-parameters: e.g. splitting strategy, architecture of neural networks, etc.

Question: Which is the best classifier for a given problem?

Tools ...for answering the question:

1. Performance measures
2. Performance estimation methods
3. Comparison with baselines

Introduction:

True Misclassification Rate

Let X be a feature space and $Y = \{1 \ldots M\}$ the set of M class labels. Moreover, let $h : X \rightarrow Y$ be a classifier and $y : X \rightarrow Y$ be the target concept to be learned. Consider the true misclassification rate *Err*$^*(h)$:

$$\mathit{Err}^*(h) = \frac{|\{\mathbf{x} \in X : y(\mathbf{x}) \neq h(\mathbf{x})\}|}{|X|}$$

Problem:

- Usually the *function* y is unknown.

Solution:

- Estimation of *Err*$^*(h)$ with *Err*(h, D_s), i.e., evaluating h on *"some"* subset $D_s \subseteq D$ of the labeled data set D we are given.

2. Performance Measures

Performance Measures: Classification

Classification Quality

A performance measure is a function that assesses the *quality* as to which a classifier solves a classification problem.

Many such measures have been proposed, as the definition of “quality” may vary depending on:

- What we are interested in
- What we want to optimize
- The characteristics of the problem (e.g. class imbalance)

Example: Misclassification Rate

The missclassifcation rate *Err* calculates the average prediction accuracy over all classes.

- *Err* is not a good performance measure in the case of imbalanced data sets
- Does not yield insights on the distribution of the error

Performance Measures: Classification

Class Confusion Matrix

Consider a data set of observations $D = \{(x, y)_n\}$ and a classifier h in a multiclass classification setting. A confusion matrix $C \in \mathbb{N}^{|Y|\times|Y|}$ contains the class assignments made by the classifier on the data set D.

Each element c_{ij} denotes the number of data points with class label i that have been assigned to class j (predicted as j):

$$c_{ij} = |\{x_k | y_k = i \wedge h(x_k) = j\}|$$

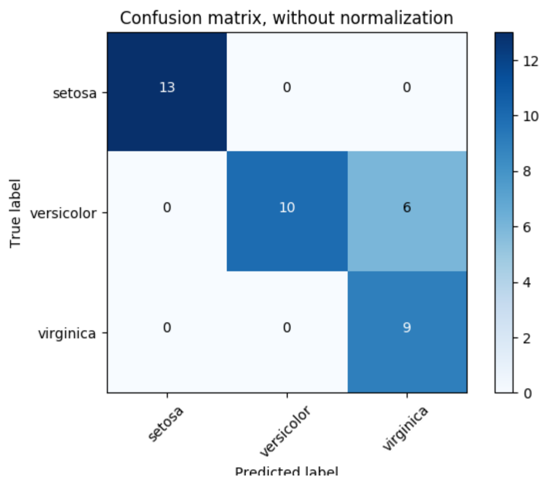


Image obtained from scikit learn demo [1]

[1] http://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html

Performance Measures: Classification

TP, FP, TN, FN

We can derive several measure from C for class k:

- **True Positives** TP$_k$**:** Number of data points **assigned to** k which **actually belong** to class k [2]

$$TP_k = C_{kk}$$

- **False Positives** FP$_k$**:** Number of data points **assigned to** k which **do not belong to** class k [3]

$$FP_k = \sum_{u \neq k} c_{uk}$$

- **False Negatives** FN$_k$**:** Number of data points **not assigned to** k but which **belong to** class k [4]

$$FN_k = \sum_{v \neq k} c_{kv}$$

- **True Negative** TN$_k$**:** Number of data points **not assigned to** k which **do not belong to** class k [5]

$$TN_k = \sum_{u \neq k} \sum_{v \neq k} c_{uv}$$

---

[2] h made a POSITIVE P decision and the decision was TRUE T $\Rightarrow$ TP
[3] h made a POSITIVE P decision but the decision was FALSE F $\Rightarrow$ FP
[4] h made a NEGATIVE N decision but the decision was FALSE F $\Rightarrow$ FN
[5] h made a NEGATIVE N decision and the decision was TRUE T $\Rightarrow$ TN

Performance Measures: Classification

TP, FP, TN, FN

Visualized in the class confusion Matrix for class $k = 2$

		Predicted Class $h(x)$			
		1	2	$\ldots$	$\lvert Y \rvert$
Actual Class y	1	TN	FP	TN	TN
	2	FN	TP	FN	FN
	3	TN	FP	TN	TN
	4	TN	FP	TN	TN
	$\vdots$	TN	FP	TN	TN
	$\lvert Y \rvert - 1$	TN	FP	TN	TN
	$\lvert Y \rvert$	TN	FP	TN	TN

Performance Measures: Classification

Precision, Recall & Co for a class k

- **Recall ρ_k of a class k (Sensitivity, True Positive Rate (TPR)):** Determines how complete the decisions of h regarding a class have been, i.e. the fraction of correct assignments to k among data points of class k:

$$\rho_k = \frac{\text{TP}_k}{\text{TP}_k + \text{FN}_k}$$

- **Precision π_k of a class k** determines how precise decisions in favor of class k are, i.e. the fraction of correct assignments to class k among all assignments to class k:

$$\pi_k = \frac{\text{TP}_k}{\text{TP}_k + \text{FP}_k}$$

- **False Positive Rate** FPR_k **of a class** k determines how likely a non class-k data point gets assigned to class k:

$$\text{FPR}_k = \frac{\text{FP}_k}{\text{FP}_k + \text{TN}_k}$$

Note: There are many more measures that can be derived from the Class Confusion Matrix.

Performance Measures: Classification

F$_\beta$-Score

Precision π_k and Recall ρ_k can be combined into a single score.

F$_\beta$-Score as weighted harmonic mean of π_k and ρ_k with β determining the importance of recall over precision:

$$(\mathsf{F}_\beta)_k = \frac{1+\beta^2}{\frac{1}{\pi_k} + \frac{\beta^2}{\rho_k}}$$

- $\beta > 1$ favors recall β times more over precision
- $\beta < 1$ favors precision $\frac{1}{\beta}$ times more over recall
- $\beta = 1$ harmonic mean between precision and recall, the so called F$_1$-Score (most common choice for β)

Performance Measures: Classification

Overall performance

Macro-Averaging:
Averages over all class measures

$$\pi^{(M)} = \frac{1}{|Y|}\sum_{k\in Y}\pi_k$$

$$\rho^{(M)} = \frac{1}{|Y|}\sum_{k\in Y}\rho_k$$

$$\mathsf{F}_\beta^{(M)} = \frac{1+\beta^2}{\frac{1}{\pi^{(M)}}+\frac{\beta^2}{\rho^{(M)}}}$$

The macro-average weighs all classes equally. There are also versions with class-specific weights.

Micro-Averaging: Calculates the performance measures over the individual assignments

$$\pi^{(\mu)} = \frac{\sum_{k\in Y}\mathsf{TP}_k}{\sum_{k\in Y}\mathsf{TP}_k + \mathsf{FP}_k}$$

$$\rho^{(\mu)} = \frac{\sum_{k\in Y}\mathsf{TP}_k}{\sum_{k\in Y}\mathsf{TP}_k + \mathsf{FN}_k}$$

$$\mathsf{F}_\beta^{(\mu)} = \frac{1+\beta^2}{\frac{1}{\pi^{(\mu)}}+\frac{\beta^2}{\rho^{(\mu)}}}$$

In the standard single-label scenario, $\pi^{(\mu)}$, $\rho^{(\mu)}$ and $\mathsf{F}_\beta^{(\mu)}$ are equal to the *accuracy* = $\frac{\sum_{k\in Y}\mathsf{TP}_k}{N}$ of the classifier (N data points).

Performance Measures: Classification

Intermediate Exercise

Exercise: You are working for a *Christmas gift insurance company* on the problem of identifying fraud[6]. You have trained two classifiers A and B and tested them on 10000 claims, the results are shown on the right.

- Compute the overall accuracy for both models. Which one has the higher accuracy and thus seems to be better?
- For both models, compute precision and recall for the "fraud" class. Which model would you choose and why?

Model A

		Predicted Class	
		no fraud	fraud
Actual Class	no fraud	9700	150
	fraud	50	100

Model B

		Predicted Class	
		no fraud	fraud
Actual Class	no fraud	9850	0
	fraud	100	50

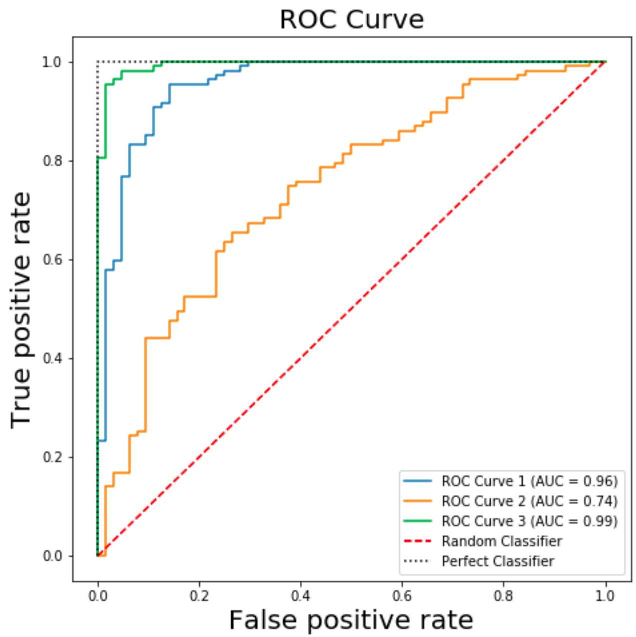
[6] Goes by the principle: "Wish one get two for free".

Performance Measures: Classification

ROC Curve

Receiver Operator Characteristic Curve (ROC-Curve)[7]: A method to visualize (summarize) the performance of a *binary classifier* ("$+$" or "$-$") across all possible classification thresholds.

- Requires classifier to output a predicted probability (score) for each observation
- Sort the observations according to the assigned score in descending order
  - Pick some score as decision threshold s_t and consider all observations with a score $> s_t$ as being classified as "$+$" and all observations with a score $<= s_t$ as "$-$".
  - Evaluate TPR and FPR for this situation to obtain one point on the ROC-curve.
- TPR and FPR are unaffected by imbalanced classes
- Area under ROC curve (AUC) is a robust performance summary for imbalanced classification problems.



[7] left: https://vitalflux.com/roc-curve-auc-python-false-positive-true-positive-rate/

Performance Measures: Real-valued Output

Common Error Measures

Let $y : X \to \mathbb{R}$ be the target function to be learned, mapping from feature space X to real values. Let $h : X \to \mathbb{R}$ be a predictor for y and let $D = \{(\mathbf{x}_i, y_i)\} \subseteq X \times \mathbb{R}$ be a set of examples.

Mean Squared Error (MSE): Preferred loss function due to mathematical convenience. Sensitive to outliers.

$$\text{MSE}(h) = \frac{1}{|D|} \sum_{(\mathbf{x}_i, y_i) \in D} (y_i - h(\mathbf{x}_i))^2$$

Mean Absolute Error (MAE): Interpretable error due to same units. Each deviation influences the MAE in direct proportion to its absolute value.

$$\text{MAE}(h) = \frac{1}{|D|} \sum_{(\mathbf{x}_i, y_i) \in D} |y_i - h(\mathbf{x}_i)|$$

The Mean Relative Approximation Error (MRAE) quantifies the deviation from the target value relative to the target value's magnitude:

$$\text{MRAE}(h) = \frac{1}{|D|} \sum_{(\mathbf{x}_i, y_i) \in D} \frac{|y_i - h(\mathbf{x}_i)|}{y_i}$$

3. Performance Estimation

Performance Estimation:
Recap

Challenge

1. Which is the best classification algorithm for a given problem?
2. Which is the most suitable configuration of hyper-parameters of a classification algorithm for a given problem?

➜ Select a performance measure and calculate a score s_h to quantify the classification quality

$D = \{(\mathbf{x}_1, y(\mathbf{x}_1)), \ldots, (\mathbf{x}_n, y(\mathbf{x}_n))\} \subseteq X \times Y$ is a set of examples.

Performance Estimation:
Recap

But we have limited information: A finite data set.

➜ s_h is a random variable - its true value is unknown.

Instead, calculate an estimate $\hat{s}_h$ of the score, based on the set of observations that have been sampled from the underlying process.

Choice of how to exploit the given dataset:

- Resubstitution
- Hold-Out
- Cross-Validation and Leave-One-Out
- Bootstrap

Performance Estimation:

Resubstitution [True Misclassification Rate]

A naive idea: Use full dataset for both training and score estimation

- $D_{tr} = D$ is the training set.
- $h : X \rightarrow Y$ is a classifier learned on the basis of D_{tr}.

Example: Calculate Misclassification rate on D_{tr} :

$$\hat{s}_h = \textit{Err}(h, D_{tr}) = \frac{|\{(\mathbf{x}, y_i) \in D_{tr} : y_i \neq h(\mathbf{x})\}|}{|D_{tr}|}$$

Bad estimator of the true score:

- $\hat{s}_h$ is based on examples that have been exploited to learn h.

➜ $\hat{s}_h$ quantifies memorization but not the generalization capability of h.

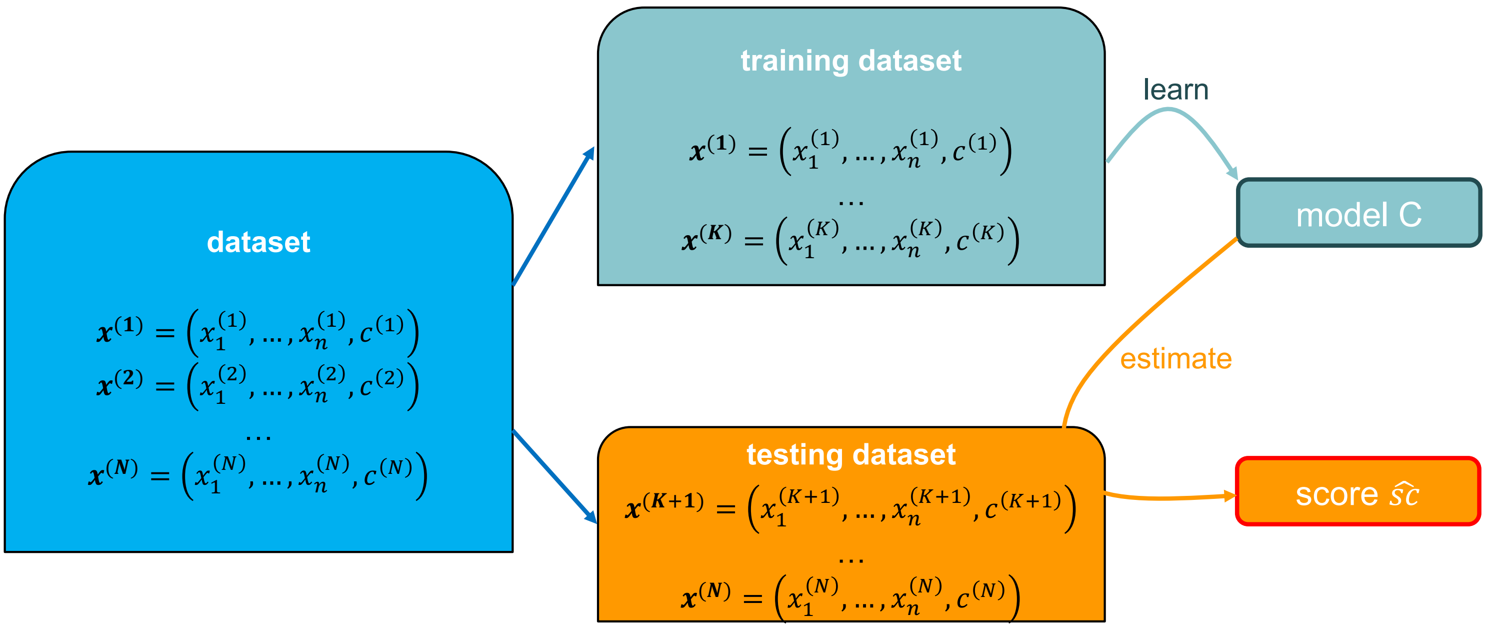
➜ $\hat{s}_h$ is too optimistic, i.e., it is constantly better than the score we observe when applying h in the wild.

Performance Estimation:

Hold-Out

- Split dataset into training and testing sets
- Often pessimistic estimation of true score

➜ Useful for large datasets, bad for small datasets

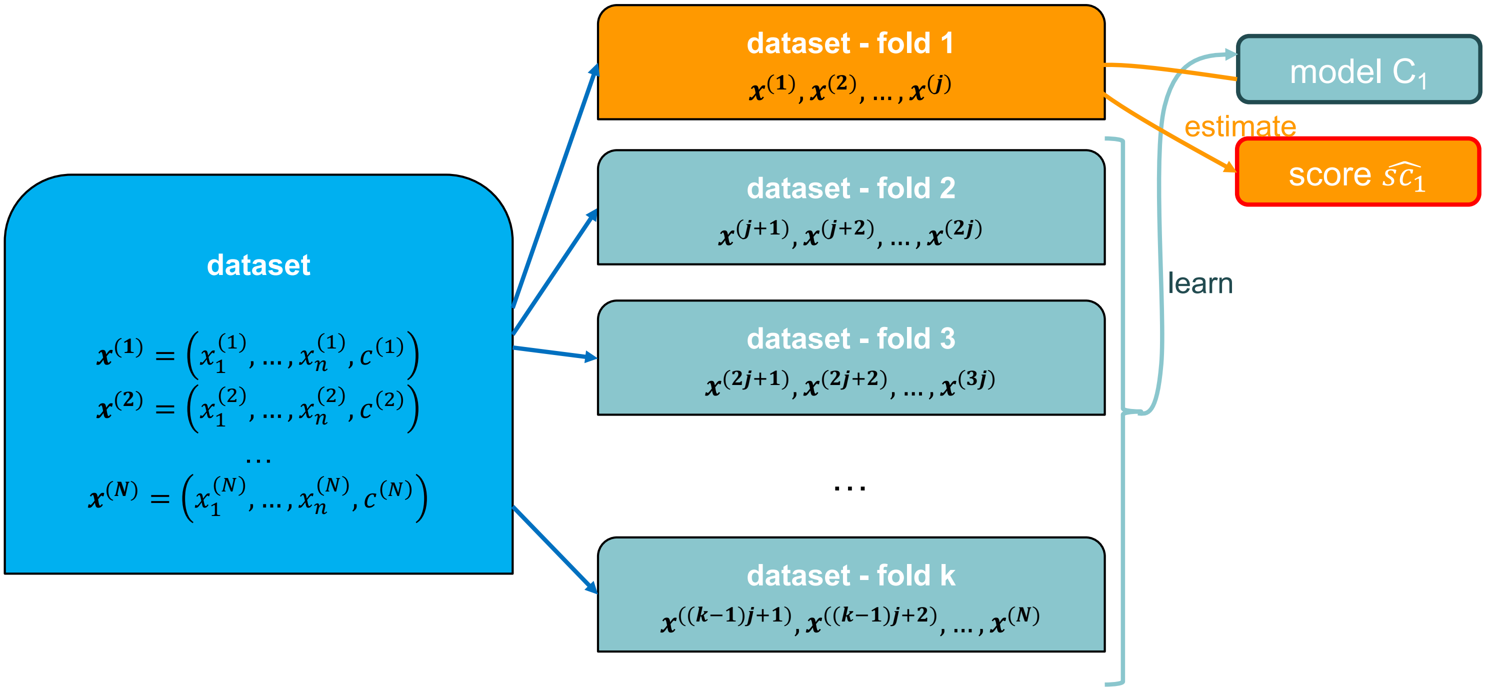


Performance Estimation:

k-fold Cross-Validation

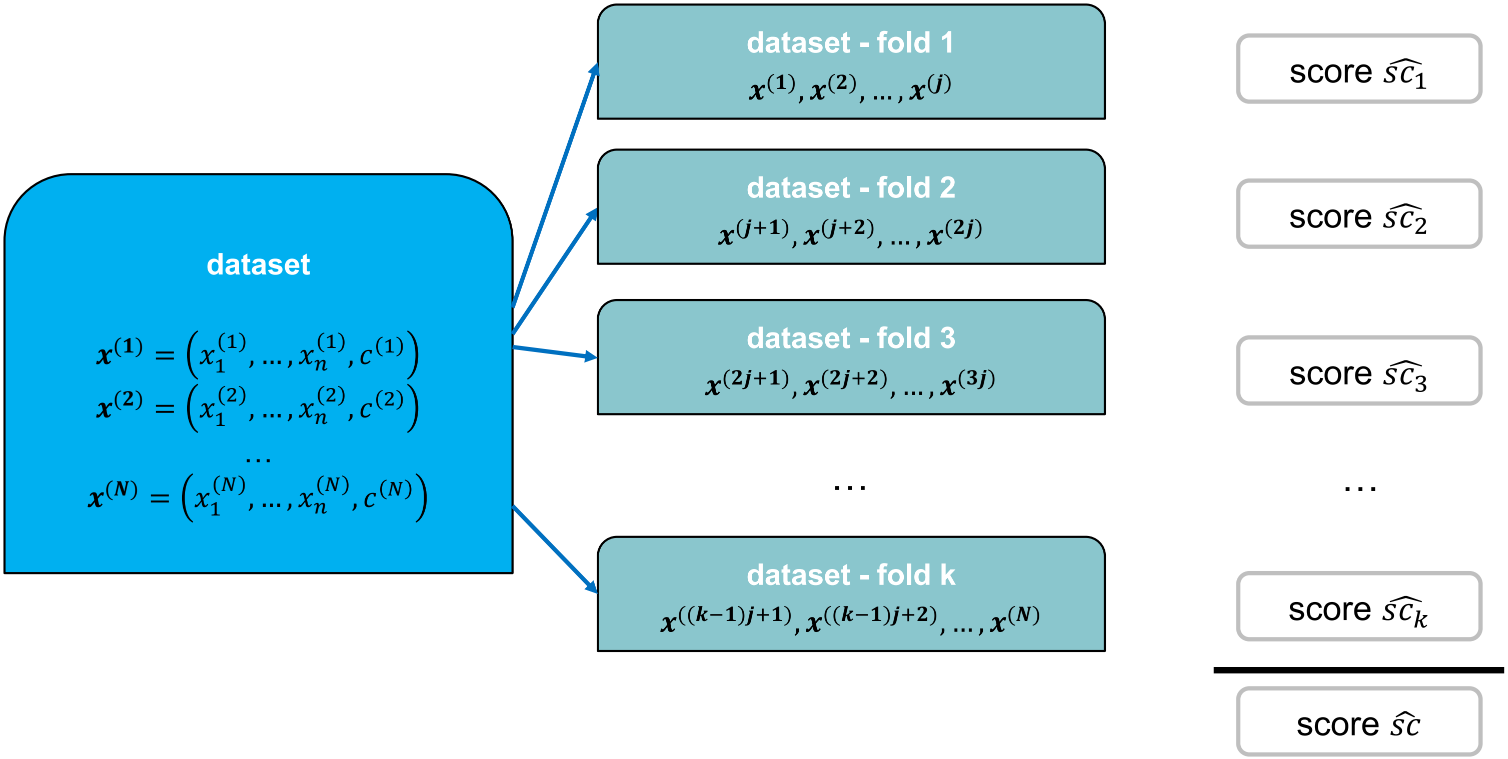
Partition the data set into k folds, each of size $j = \lfloor N/k \rfloor$. Repeat k times:

- Train on joint data from $k-1$ folds
- Calculate score on data from the held-out fold



Performance Estimation:

k-fold Cross-Validation



After k rounds, the average over k scores is reported as the classifier's performance.

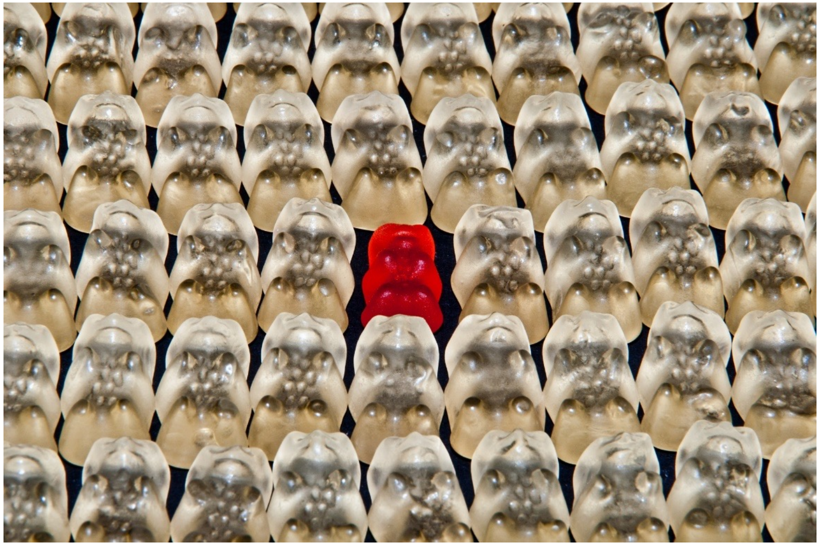
- Less pessimistic than Hold-Out
- Suitable for smaller datasets

Performance Estimation:

Leave-one-out Cross-Validation

Leave-one-out Cross-Validation: Special case of k-fold CV, where $k = N$

- Each fold contains only one observation
- Calculate N scores, each based on only one observation



- High computational cost
- Better estimation of true score, but less stable (=high variance)

Performance Estimation:
Bootstrap

- Repeat k times:
  - Create "bootstrap dataset" of j observations by randomly sampling *with* replacement
  - Train classifier on bootstrap dataset
  - Estimate score on *full dataset*

➜ Problematic: Uses data from the training phase. Optimistic estimation.

➜ Not recommended

Performance Estimation:

$.632$-Bootstrap

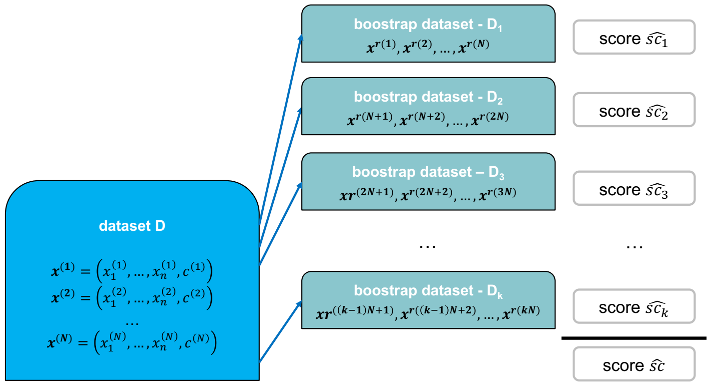
- Repeat k times:
  - Create "bootstrap dataset" D_b of N observations by randomly sampling *with* replacement
  - Train classifier on bootstrap dataset D_b
  - Estimate score on dataset $D \setminus D_b$

→ Expected number of distinct observations in D_b is approximately $0.632 * N$

→ No training data used for estimation

→ Pessimistic estimate

→ Useful for small data sets



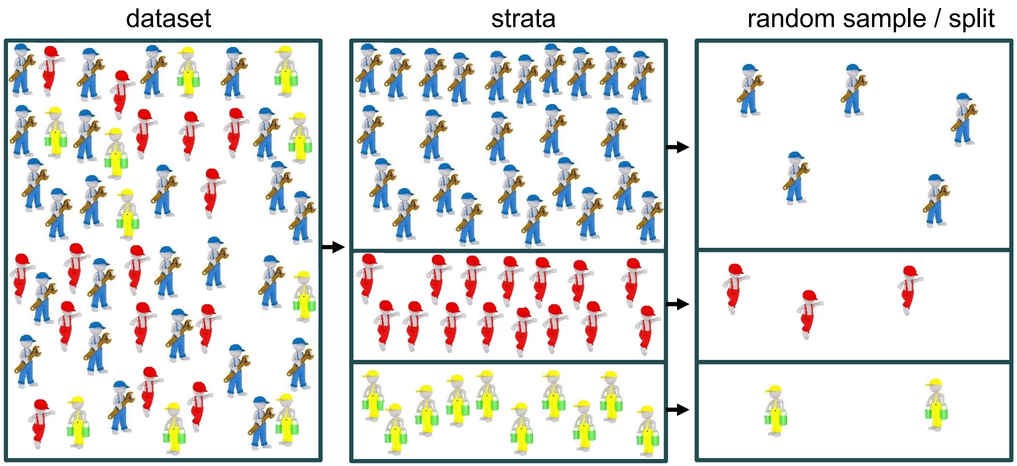
Performance Estimation:
Stratified Sampling

Problem: Imbalanced classes

- Splits may not contain any observations of the infrequent classes
- Leads to bad classifiers / estimations

Solution: *Stratification*. Keep the proportions of the classes in the splits equal to the full dataset. *Stratified sampling* can be incorporated in all estimation techniques:

- Hold-Out
- Cross-Validation
- Bootstrap



Further Details [8]

[8] https://en.wikipedia.org/wiki/Stratified_sampling

Performance Estimation:
Comparison to Baselines

Baselines are lower bounds on the achievable performance obtained from a trivial predictor.

Used as *"sanity"* check when approaching new problems. Examples:

- **Trivial Classification:** Always predict the majority class
- **Trivial Regression:** Use a central tendency (e.g. mean, median)
- **Random Guessing:** Randomly guess the outcome based on the marginal distribution $P(y)$

Example:

Unbalanced data set with two-classes: $D = \{(x_1, 0), (x_2, 0), \ldots (x_{99}, 0), (x_{100}, 1)\}$

Baseline for missclassification rate: *Err*$(h) = 1\%$

Performance Estimation:

Performance Estimation Summary

Resubstition

- Do not use.

Hold-Out

- Good for large data sets
- Computation expensive
- Frequent splits $80\%/20\%$ or $70\%/30\%$ (train / test)
- Can be done repeatedly to improve estimation (repeated hold-out)

Cross-Validation and Leave-One-Out

- Good for small datasets
- Computationally expensive

Bootstrap

- $.632 Bootstrap$ good for small datasets
- Computationally expensive