Sampling and Weighting

DS-6030 | Spring 2026

sampling-weighting.pdf

Table of contents

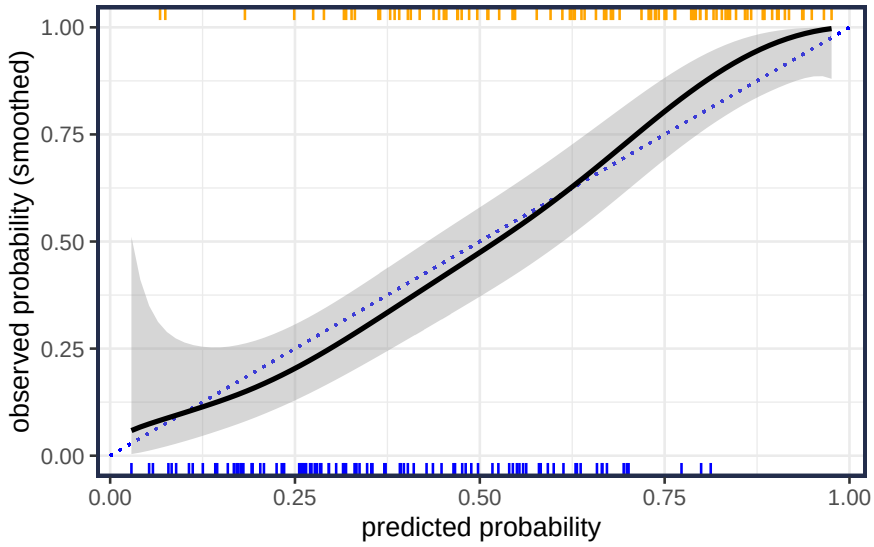
1 Case-Control Sampling

1.1 The Problem

1.1.1 Model building

A hospital research team builds a classifier to predict a rare disease from blood markers. They collect a *balanced* dataset: 1000 positive, 1000 negative. The model predicts well on held-out test data (over many metrics).

n	auroc	brier	log_loss
200	0.828	0.171	0.517



1.1.2 Deployment Problems

- Based on the good results, the model is deployed in a general screening where true prevalence is around 2% (2 out of 100 have disease).
- In the trial period, the model flags **12 of the first 20 patients** as having the disease using the $\hat{p} > 0.20$ threshold the hospital want to use (1:4 cost ratio).
  - Only 1 patient actually has the disease!
  - There are 11 false positives! (57.9% False Alarm Rate)
  - Avg Cost = (11 x 1 + 0 x 4) / 20.
- This is a 500 bed hospital that runs the model on about 40 to 80 suspected cases per day, or roughly 1,200 to 2,400 per month.
  - Hospital leadership is furious and says they can't use your model with so many false alarms!

Table 1: CRP: C-reactive protein (mg/L; normal < 3). WBC: white blood cell count ($10^3/\mu$L; normal 4.5–11).

Patient	Age	Sex	Race	CRP (mg/L)	WBC ($10^3/\mu$L)	$\hat{p}$	disease
P16	37	M	White	34.8	8.6	0.918	1
P18	68	M	Hispanic	11.1	10.7	0.828	0
P08	53	M	Black	5.5	12.7	0.777	0
P12	51	F	White	9.7	7.3	0.627	0
P05	57	M	White	5.2	9.5	0.584	0
P14	30	M	White	3.2	10.8	0.522	0
P09	53	M	Hispanic	3.8	9.8	0.509	0
P19	44	F	Asian	3.9	8.6	0.438	0
P15	60	M	White	1.8	12.1	0.433	0
P03	54	F	White	3.4	9.0	0.423	0
P17	56	M	White	3.1	7.4	0.301	0
P07	48	F	Black	3.9	4.5	0.210	0
P06	48	F	White	2.8	4.6	0.154	0
P01	61	F	Black	1.8	6.1	0.137	0
P11	30	F	White	3.5	2.8	0.130	0
P02	48	M	White	1.7	5.8	0.120	0
P10	50	M	Hispanic	0.8	9.2	0.118	0
P13	56	F	White	1.5	6.3	0.118	0
P04	47	M	Hispanic	1.6	3.2	0.060	0
P20	45	M	White	1.0	4.2	0.045	0

Your Turn #1 : Can you help?

- What went wrong?
- How to fix it before the research team loses their jobs?

1.2 Base Rate Correction

1.2.1 Bayes Theorem

Bayes Theorem to the rescue!

Bayes Theorem

$$p_k(x) = \Pr(Y = k \mid X = x) = \frac{\Pr(X = x \mid Y = k)\Pr(Y = k)}{\Pr(X = x)}$$
$$= \frac{f_k(x)\pi_k}{\sum_j f_j(x)\pi_j}$$

- $f_k(x)$ is the *class conditional density* (pdf/pmf)
- $0 \leq \pi_k \leq 1$ are the *prior class probabilities*
- $\sum \pi_k = 1$
- X is distributed as a finite mixture model: $f(x) = \sum_j f_j(x)\pi_j$

1.2.2 Special case when $K = 2$ (binary classification)

$$p(x) = \Pr(Y = 1 \mid X = x) = \frac{\Pr(X = x \mid Y = 1)\Pr(Y = 1)}{\Pr(X = x)}$$
$$= \frac{f_1(x)\pi}{f_1(x)\pi + f_0(x)(1-\pi)}$$

Recall from the generative classifiers lecture that the odds and log-odds (logit) decomposes neatly.

Odds

$$\text{odds}(x) = \frac{p(x)}{1-p(x)}$$
$$= \frac{f_1(x)\pi}{f_0(x)(1-\pi)}$$
$$= \underbrace{\left(\frac{\pi}{1-\pi}\right)}_{\text{prior odds}} \underbrace{\left(\frac{f_1(x)}{f_0(x)}\right)}_{\text{density ratio}}$$

- The **density ratio** $f_1(x)/f_0(x)$ measures how much more consistent x is with class 1 than class 0, *independent of how common either class is*.
- The prior odds $\pi/(1-\pi)$ is a **constant** that doesn't depend on x.

logit (log odds)

Define $\gamma(x) = \log \frac{p(x)}{1-p(x)}$:

$$\begin{aligned}\gamma(x) &= \log\left(\frac{p(x)}{1-p(x)}\right) \\ &= \log\left(\frac{f_1(x)\pi}{f_0(x)(1-\pi)}\right) \\ &= \underbrace{\log\left(\frac{\pi}{1-\pi}\right)}_{\text{log prior odds}} + \underbrace{\log\left(\frac{f_1(x)}{f_0(x)}\right)}_{\text{log density ratio}}\end{aligned}$$

- The log density ratio is sometimes called the *weight of evidence*.

1.2.3 The problem

- The model was fit using the wrong prior ($\pi_{\text{train}} = 0.5$). We want to swap in the right prior ($\pi_{\text{deploy}} = 0.02$) while keeping the rest of the model intact.
  - We know that $\pi_{\text{train}} = 0.50$, since that was how the data was collected.
  - We want to adjust the model for reality ($\hat{\pi}_{\text{deploy}} = 0.02$).

$$\begin{aligned}\widehat{\text{odds}}_{\text{train}}(x) &= \frac{\hat{p}_{\text{train}}(x)}{1-\hat{p}_{\text{train}}(x)} \\ &= \underbrace{\left(\widehat{\frac{f_1(x)}{f_0(x)}}\right)}_{\text{density ratio}} \underbrace{\left(\frac{\pi_{\text{train}}}{1-\pi_{\text{train}}}\right)}_{\text{prior odds}}\end{aligned}$$

$$\begin{aligned}\widehat{\text{odds}}_{\text{deploy}}(x) &= \frac{\hat{p}_{\text{deploy}}(x)}{1-\hat{p}_{\text{deploy}}(x)} \\ &= \underbrace{\left(\widehat{\frac{f_1(x)}{f_0(x)}}\right)}_{\text{density ratio}} \underbrace{\left(\frac{\pi_{\text{deploy}}}{1-\pi_{\text{deploy}}}\right)}_{\text{prior odds}}\end{aligned}$$

- The model learned the density ratio (weight of evidence) $\widehat{f_1(x)/f_0}(x)$. This captures the how the feature values influence the predictions.
- We just have to correct the prior odds, which is a **constant** (not a function of x).
- Instead of working with the odds, we can consider the log-odds (logits)

Writing the training log-odds:

$$\begin{aligned}\hat{\gamma}_{\text{train}}(x) &= \log\frac{\hat{p}_{\text{train}}(x)}{1-\hat{p}_{\text{train}}(x)} \\ &= \log\widehat{\frac{f_1(x)}{f_0(x)}} + \log\frac{\pi_{\text{train}}}{1-\pi_{\text{train}}}\end{aligned}$$

And deployment log-odds:

$$\begin{aligned}\hat{\gamma}_{\text{deploy}}(x) &= \log \frac{\hat{p}_{\text{deploy}}(x)}{1-\hat{p}_{\text{deploy}}(x)} \\ &= \log \widehat{\frac{f_1(x)}{f_0(x)}} + \log \frac{\pi_{\text{deploy}}}{1-\pi_{\text{deploy}}}\end{aligned}$$

1.2.4 The fix

Your Turn #2 : Let's work it out

Define correction weights:

$$w_1 = \frac{\pi_{\text{deploy}}}{\pi_{\text{train}}}$$
$$w_0 = \frac{1 - \pi_{\text{deploy}}}{1 - \pi_{\text{train}}}$$

Write out correction in odds:

$$\begin{aligned}
\widehat{\text{odds}}_{\text{deploy}}(x) &= \frac{\hat{p}_{\text{deploy}}(x)}{1 - \hat{p}_{\text{deploy}}(x)} \\
&= \underbrace{\left(\widehat{\frac{f_1(x)}{f_0(x)}}\right)}_{\text{density ratio}} \underbrace{\left(\frac{\pi_{\text{deploy}}}{1 - \pi_{\text{deploy}}}\right)}_{\text{prior odds}} \\
&= \widehat{\text{odds}}_{\text{train}}(x) \left(\frac{\pi_{\text{deploy}}}{\pi_{\text{train}}}\right) \left(\frac{1 - \pi_{\text{train}}}{1 - \pi_{\text{deploy}}}\right) \\
&= \widehat{\text{odds}}_{\text{train}}(x) \frac{w_1}{w_0}
\end{aligned}$$

And correction in logits (log odds):

$$\begin{aligned}
\hat{\gamma}_{\text{deploy}}(x) &= \log \widehat{\frac{f_1(x)}{f_0(x)}} + \log \frac{\pi_{\text{deploy}}}{1 - \pi_{\text{deploy}}} \\
&= \hat{\gamma}_{\text{train}}(x) - \log \frac{\pi_{\text{train}}}{1 - \pi_{\text{train}}} + \log \frac{\pi_{\text{deploy}}}{1 - \pi_{\text{deploy}}} \\
&= \hat{\gamma}_{\text{train}}(x) + \log w_1 - \log w_0 \\
&= \hat{\gamma}_{\text{train}}(x) + \log \frac{w_1}{w_0}
\end{aligned}$$

And correction in probability space:

$$\hat{p}_{\text{deploy}}(x) = \frac{\hat{p}_{\text{train}}(x) \cdot w_1}{\hat{p}_{\text{train}}(x) \cdot w_1 + \left(1 - \hat{p}_{\text{train}}(x)\right) \cdot w_0}$$

Correction Equations

Correction Weights:

$$w_1 = \frac{\pi_{\text{deploy}}}{\pi_{\text{train}}}$$
$$w_0 = \frac{1 - \pi_{\text{deploy}}}{1 - \pi_{\text{train}}}$$

Correction Equations:

$$\hat{p}_{\text{deploy}}(x) = \frac{\hat{p}_{\text{train}}(x)\, w_1}{\hat{p}_{\text{train}}(x)\, w_1 + \left(1 - \hat{p}_{\text{train}}(x)\right) w_0}$$

$$\widehat{\text{odds}}_{\text{deploy}}(x) = \widehat{\text{odds}}_{\text{train}}(x)\, \frac{w_1}{w_0}$$

$$\hat{\gamma}_{\text{deploy}}(x) = \hat{\gamma}_{\text{train}}(x) + \log \frac{w_1}{w_0}$$

1.2.5 Summary

- The model was not completely wrong. It correctly learned how features separate the classes (the density ratio). The mistake was asking it to make decisions using a prior it was never told.
- **Connection to calibration (hw6):** In hw6, you fit a model to recalibrate predictions after a distribution shift. You had to fit a new model because the source of miscalibration was unknown. Here, we know exactly why calibration is off: the wrong prior was baked in. Here we are doing calibration, with a known correction (specifically, change in intercept).

1.2.6 The clinic patients

Your Turn #3

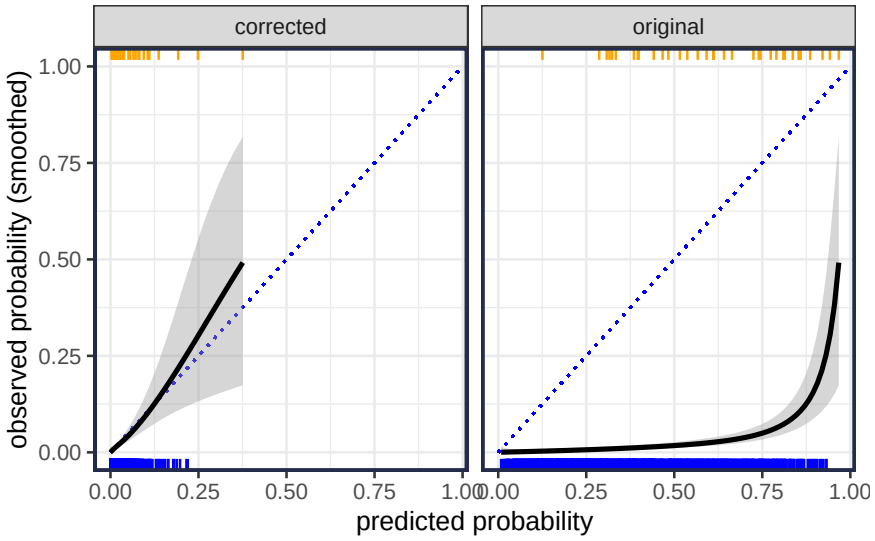
The hospital refers patients for follow-up biopsy when $\hat{p} > 0.20$ (cost ratio 1:4).

1. Mark which patients are referred using the raw $\hat{p}$. How many? Does that seem right for a 2% prevalence population?
2. Apply the correction. Compute $\hat{p}_{\text{corrected}}$. Who is referred now?
3. Order the patients from highest to lowest risk. Is the model's rank ordering of patients changed by the correction. Which standard performance metric captures ranking? Would you expect it to change?

1.2.7 Performance on deployment data

Here, we consider $n_{\text{deploy}} = 2000$ new patients.

n	model	auroc	brier	log_loss
2000	corrected	0.781	0.017	0.079
2000	original	0.781	0.188	0.551



- Note that rank ordering is completely preserved by the correction; the fix only changes the intercept.

2 Weighted Likelihoods

2.1 Weights in the loss function

The base rate fix corrected predictions *after* training. A similar result can be achieved *during* training using weighted likelihoods.

Using weighted fitting is useful in many situations beyond base rate correction.

The weighted log-likelihood for logistic regression is:

$$\ell(\beta;\, \mathbf{w}) = \sum_{i=1}^{n} w_i \Big[y_i \log \hat{p}(x_i;\beta) + (1-y_i) \log\big(1-\hat{p}(x_i;\beta)\big) \Big]$$

What do weights actually do?

- $w_i = 0$: row i is **excluded** from training
- $w_i = 2$: row i is counted **twice**, as if the row were duplicated
- $w_i = k$ (positive integer): row i is counted k times

Bootstrap

- You have been using weighted likelihoods all semester!
- A bootstrap sample is equivalent to using random integer weights.
- Each observation receives weight $w_i \sim \text{Multinomial}(n;\, 1/n, \ldots, 1/n)$, with $w_i = 0$ for out-of-bag observations and $w_i \geq 1$ for selected ones.
- The **Bayesian bootstrap** replaces integer weights with continuous weights drawn from $\text{Dirichlet}(1, 1, \ldots, 1)$. You don't get out of bag, but has some nice properties.

2.2 Two paths to the same answer

For the hospital case-control problem, there are two equivalent approaches:

2.2.1 Path 1 — Post-hoc correction (what we just did):

$$\text{Train on balanced data} \longrightarrow \hat{p}_{\text{train}}(x) \longrightarrow \text{Bayes correction} \longrightarrow \hat{p}_{\text{deploy}}(x)$$

2.2.2 Path 2 — Weighted training (IPW):

Assign weights to the training data before fitting:

$$w_i = \begin{cases} w_1 = \pi_{\text{deploy}}/\pi_{\text{train}} & \text{if } y_i = 1 \\ w_0 = (1-\pi_{\text{deploy}})/(1-\pi_{\text{train}}) & \text{if } y_i = 0 \end{cases}$$

Your Turn #4 : Weighted Fitting

1. Refit the model to the training data, but use weights.

- The original fit was logistic regression using features `log(crp)` and `wbc`.
- In R,

```
fit_original = glm(disease ~ log(crp) + wbc,
                   family = binomial(),
                   data = training)
```

2. Make predictions on the test deployment.

3. Compare to the Bayesian corrected probabilities.

3 Strategic Subsampling

3.1 When your data is too big to fit

Suppose you have 50 million insurance claims, 0.1% fraudulent. Fitting a model on all 50 million rows is slow or infeasible.

Naive approach (1% random subsample): ~500,000 rows, ~500 fraud cases; but you've lost most of the rare class signal.

Strategic approach: Keep *all* positive cases (50,000 frauds); randomly subsample negatives at rate r.

- Sample size roughly: $50{,}000 + r \times 49{,}950{,}000$ with r tunable to your compute budget
- Sample prevalence: far higher than the true 0.1%
- **Problem:** model trained on this sample will predict inflated probabilities

Now we know of two ways to fix this:

3.1.1 Option 1. Base rate correction.

3.1.2 Option 2. Inverse probability weights (IPW)

IPW fix: positives were sampled with probability 1; negatives with probability r.

$$w_i = \begin{cases} 1 & y_i = 1 \\ 1/r & y_i = 0 \end{cases}$$

Sampling

- Setting $w_i = 0$ in the loss function is identical to removing row i from the training set.
- Subsampling is just IPW where some weights are zero and others are $1/r$.
- **Sampling and weighting are the same operation**; sampling is just the special case where weights are 0 or $1/r$.

3.2 What rate r to use?

Smaller r: faster, but the estimated decision boundary has more variance (fewer negatives to calibrate against).

Your Turn #5

You have 10 million rows with 0.1% positive class. You keep all positives and subsample negatives at $r = 0.01$.

1. How many rows are in your subsample? What is the sample prevalence?
2. What weight should each negative receive in the weighted loss?
3. If a model trained *without* weights on this subsample predicts $\hat{p} = 0.30$ for a test case, what would the corrected prediction be? (Use the base rate correction formula with π_{train}

equal to the sample prevalence from part 1, and $\pi_{\text{deploy}} = 0.001$.)

---

4 Group-Weighted Modeling

4.1 Building models that adapt to subgroups

Consider building a fraud detection model for a bank with customers across 50 states. State populations vary enormously - California has millions of customers; Wyoming has thousands.

Three options:

1. **Separate model per state:** high variance for Wyoming (too little data), no borrowing of information across states.
2. **Pooled model:** low variance, but may not adapt to Wyoming's distinct fraud patterns.
3. **Group-weighted model:** a middle path.

For group j, weight group j observations at 1 and all others at $\lambda \in [0, 1]$:

$$w_i = \begin{cases} 1 & \text{observation } i \text{ is in group } j \\ \lambda & \text{otherwise} \end{cases}$$

- $\lambda = 0$: separate model (group j data only)
- $\lambda = 1$: fully pooled model (all groups equally weighted)
- $\lambda \in (0, 1)$: **borrows strength** from other groups while adapting to group j

The weight λ controls a **bias-variance tradeoff**: small λ reduces bias for group j at the cost of higher variance; large λ reduces variance but may introduce bias.

Choosing λ: cross-validate *within group j*. Evaluate held-out performance on group j observations for each candidate λ. Let the data answer: "How much should group j trust data from other groups?"

```
fit_group <- function(data, j, lambda) {
  data %>%
    mutate(w = if_else(group == j, 1, lambda)) %>%
    fit(x, y, weights = w, data = .)
}
```

Applications:

- **Demographic subgroups:** adapt a model to underrepresented groups without discarding majority-group information
- **Regional models:** adapt a national model to local conditions (regional economic patterns, climate zones)
- **Rare disease subtypes:** learn from related subtypes while adapting to a rare one
- **Client adaptation:** personalize a general model to a specific client using their limited historical data

Your Turn #6

A bank has four regional groups:

Region	n
Northeast	50,000

South	40,000
Midwest	8,000
Mountain West	300

You want to build a model for the Mountain West (300 observations).

1. Why would a fully separate model ($\lambda = 0$) be problematic for this region?
2. Why might the fully pooled model ($\lambda = 1$) also be unsatisfactory?
3. How would you choose the optimal λ? What concern arises when performing cross-validation with only $n = 300$ observations?

5 Synthesis

5.1 Weights are everywhere

Every section of this lecture used the same operation (**reweighting observations in the loss function**) to solve a different problem:

Problem	Weights	IPW interpretation
Base rate mismatch	$w_k = \pi_{\text{deploy},k} / \pi_{\text{train},k}$	Correct selection into the balanced training set
Strategic subsampling	$w_i = 1$ (positives), $1/r$ (negatives)	Correct selection into the subsample
Group adaptation	$w_i = 1$ (group j), λ (others)	Soft selection into group-specific training data
Bootstrap	$w_i \sim \text{Multinomial}(n, 1/n)$	Random resampling = random reweighting

The unifying idea: any time your training distribution differs from your target distribution, the fix is to reweight observations so the weighted training distribution matches your target. Subsampling is just weighting with some $w_i = 0$.