Predictive Bias and Calibration

DS-6030 | Spring 2026

pred-bias.pdf

Table of contents

1 Predictive Performance

1.1 Overall Performance Metrics

- For this lecture, let $\hat{f}(x)$ be predictions from a *fully tuned and trained* model.
  - Predictive performance is evaluated on the Test/Eval data

1.1.1 Default Data

default	student	balance	income
No	No	1384.9	40131
Yes	Yes	1889.3	22652
Yes	Yes	1740.8	18161
Yes	Yes	2123.4	23836
No	Yes	856.7	15523
No	No	310.1	31446
No	No	1248.9	31960
Yes	No	1823.6	44260

- Fit two models on training data
  - *lr*: Logistic Regression (Ridge Penalty)
  - *lgbm*: Boosted Trees (lightgbm)
- Made predictions on the testing data

model	log-loss	AUROC
lgbm	0.079	0.940
lr	0.078	0.953

Your Turn #1 : Which model would you pick?

Do you have sufficient information to choose the best model?

1.2 Performance as a function of feature space (X)

- Performance may vary over feature space. A single performance metric may mask serious issues.
- Why can performance vary over feature space?

1. Structural: pattern is stronger in some regions of feature space. E.g., some subgroups are just easier to predict given the features.
2. Training data sparsity: some regions of feature space has less training data.
3. Test data sparsity: performance is estimated from test data. We can't know the true performance, but only can get an estimate. Sparsity of samples in the test data can limit what we can conclude about actual performance.

1.2.1 Default prediction by *student*

student	n_test	log-loss	AUROC
Combined	2000	0.079	0.940
No	1398	0.067	0.953
Yes	602	0.108	0.913

Your Turn #2

1. If we thought that the number of students was going to increase in the future, what do you expect to happen to the overall performance?

2. Fairness: is it acceptable to have a model that has different predictive performance for student customers?

1.2.2 Performance Calculations

- For most metrics: overall average performance is a weighted aggregate of group-level performance.
  - $\widehat{\text{EPE}}$ is the estimated expected prediction error.

$$\begin{aligned}
\widehat{\text{EPE}} &= \frac{1}{n}\sum_{i=1}^{n} L(y_i, \hat{y}_i) \\
&= \frac{1}{n}\left(\sum_{i:\text{group 1}} L(y_i, \hat{y}_i) + \sum_{i:\text{group 2}} L(y_i, \hat{y}_i)\right) \\
&= \frac{n_{\text{group 1}}}{n}\widehat{\text{EPE}}(\text{group 1}) + \frac{n_{\text{group 2}}}{n}\widehat{\text{EPE}}(\text{group 2}) \\
&= p_{\text{group 1}}\widehat{\text{EPE}}(\text{group 1}) + p_{\text{group 2}}\widehat{\text{EPE}}(\text{group 2})
\end{aligned}$$

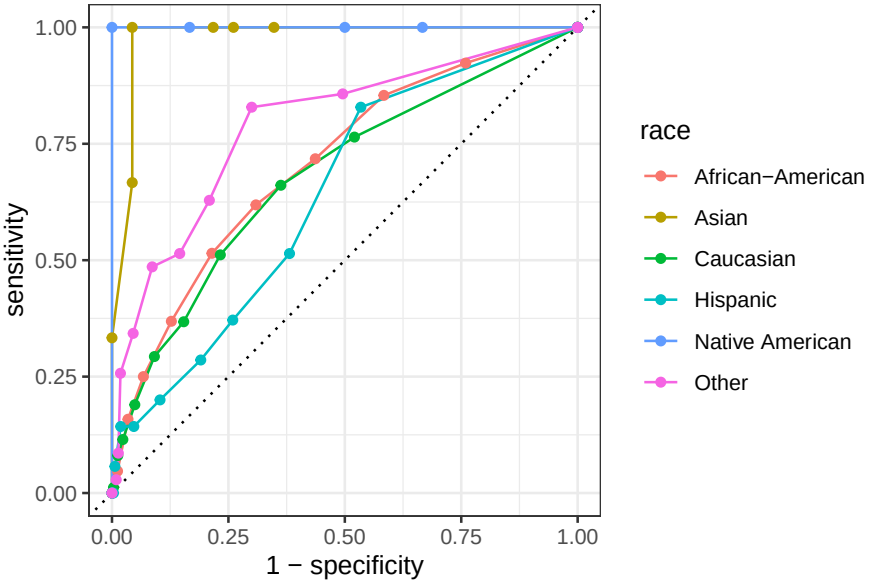
- For group based metrics, like AUC, this doesn't hold.

Your Turn #3 : Calculate Total Log-Loss

1. Use this equation to calculate the overall log-loss from the `student = Yes` and `student = No` losses.

2. Prove that the overall AUROC doesn't follow this relationship.

1.2.3 COMPAS risk scores



race	n	auroc
Native American	7	1.000
Asian	26	0.978
Other	255	0.792
Combined (All Races)	4020	0.719
African-American	1918	0.708
Caucasian	1459	0.683
Hispanic	355	0.641

1.3 Summary

- Aggregate performance metrics can indicate if the predictions are poor, but not how or why.
- Performance can vary over feature space.
  - This can be crucial to examine. The COMPAS model faced criticism because some performance metrics varied between races.
  - Note: estimated performance $\neq$ true performance. Check sample size and measure uncertainty.
- Aggregate performance metrics tell us how much predictive error exists on average, but not where, how, or why.
- Performance varies across feature space; overall metrics mask this heterogeneity.
  - Subgroup performance should always be examined, especially when predictions drive decisions that affect different groups differently.
  - The COMPAS recidivism model is a prominent example: aggregate performance metrics appeared acceptable, but performance varied substantially across racial groups leading to significant real-world criticism and legal scrutiny.

- Estimated performance $\neq$ true performance.
  - Check sample size and measure uncertainty.
  - Smaller subgroups produce noisier performance estimates. A performance gap between groups might be real, or it might be estimation noise.
- Aggregate metrics are a starting point. The rest of this lecture develops tools to understand the structure of prediction errors.

2 Predictive Bias Framework

- *Predictive bias* is a term representing how far off, on average, the predictions are from the truth.
- Let $f(x)$ be the optimal prediction.
  - $f(X) = E[Y \mid X = x]$ for squared error loss
  - $f(X) = \text{Median}[Y \mid X = x]$ for absolute error loss
  - $f(X) = \text{logit}(\Pr(Y = 1 \mid X = x))$ for log-loss
- Our fitted model produces $\hat{f}(x)$
- The predictive bias (loosely defined) is the error between our predictions and the true optimal predictions.
  - There are two ways to look at the bias/error.

1. Residual Analysis: condition on x

$$\begin{aligned} b(x) &= E[Y - \hat{f}(X) \mid X = x] \\ &= E[Y \mid X = x] - \hat{f}(x) \\ &= f(x) - \hat{f}(x) \end{aligned}$$

2. Calibration: condition on $\hat{f}(x)$

$$\begin{aligned} b(c) &= E[Y - c \mid \hat{f}(X) = c] \\ &= E[Y \mid \hat{f}(X) = c] - c \\ &= f(x) - c \qquad \text{when } \hat{f}(x) = c \end{aligned}$$

3 Predictive Residual Analysis: Detecting Bias Conditional on X

3.1 Notation

- True data generating process (for regression problems)

$$\begin{aligned} Y &= f(X) + \epsilon(X) \\ &= \hat{f}(X) + b(X) + \epsilon(X) \\ &= \hat{f}(X) + r(X) \end{aligned}$$

  - where $r(X) = b(X) + \epsilon(X)$ is error under our model.

- If $E[r(X)] = 0$, then our predictions are unbiased.
- We can search over X to discover regions where $E[r(X)] \neq 0$. This is residual analysis.

3.2 Graphical Residual Analysis:

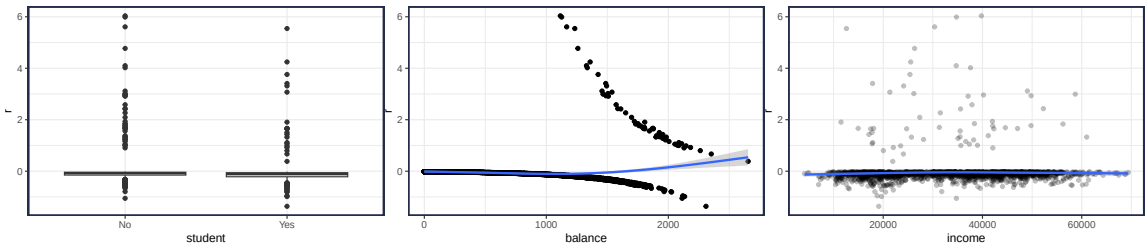
1. Form *predictive* residuals: $r_i = y_i - \hat{y}_i$ from *test* observations ($\hat{y}_i = \hat{f}(x_i)$).

- Note: if the variance is not equal, use scaled residuals. E.g., for logistic regression use Pearson residuals: $r_i = \frac{y_i - \hat{p}_i}{\sqrt{\hat{p}_i(1-\hat{p}_i)}}$.

2. Plot (x_i, r_i) over different regions of X. Look for deviations from 0.

3.2.1 Example: Default Data

default	student	balance	income	p_hat	r
No	No	75.83	42075	0.001	-0.024
No	No	515.62	39639	0.003	-0.054
No	Yes	736.24	13757	0.005	-0.072
No	No	678.02	59417	0.006	-0.077
Yes	Yes	1806.92	11506	0.216	1.907
Yes	No	1610.48	35590	0.146	2.419
Yes	No	2054.45	37367	0.475	1.052
Yes	Yes	1233.45	12586	0.032	5.544



3.3 Model Based Residual Analysis:

1. Form *predictive* residuals: $r_i = y_i - \hat{y}_i$ from *test* observations ($\hat{y}_i = \hat{f}(x_i)$).
2. Use r_i as the *outcome variable* and fit models using features x_i. Look for any significant terms or evaluation metrics better than baseline.

3.3.1 Connections to Boosting

- The model-based residual analysis has a strong connection to boosting.
- Recall in gradient boosting, a new model is fit to the (scaled) negative gradients. For many common losses (squared error, log-loss, poisson loss) the negative gradients are simple residuals.
- If we can fit a new model to the residuals and find either a) significant features or b) improved prediction, that is evidence that our model has structural issues.

4 Diagnostic Calibration: Detecting Bias Conditional on $\hat{f}(X)$

4.1 Overview

4.1.1 Continuous Outcome

A regression model is said to be *calibrated* if the true expected outcome is equal to the predicted outcome.

$$E[Y \mid \hat{f}(X) = c] = c \qquad \text{for all } c$$

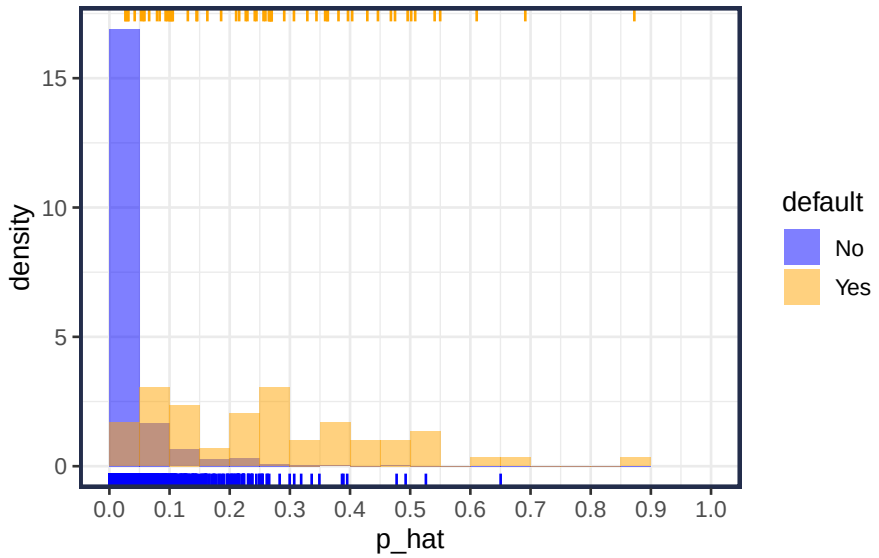
4.1.2 Risk Model (Binary Outcome)

A risk model is said to be *calibrated* if the true risk (probabilities) is equal to the predicted probabilities.

$$\Pr(Y = 1 \mid \hat{p}(x) = p) = p \qquad \text{for all } p$$

4.2 Calibration Plots (Reliability Diagrams)

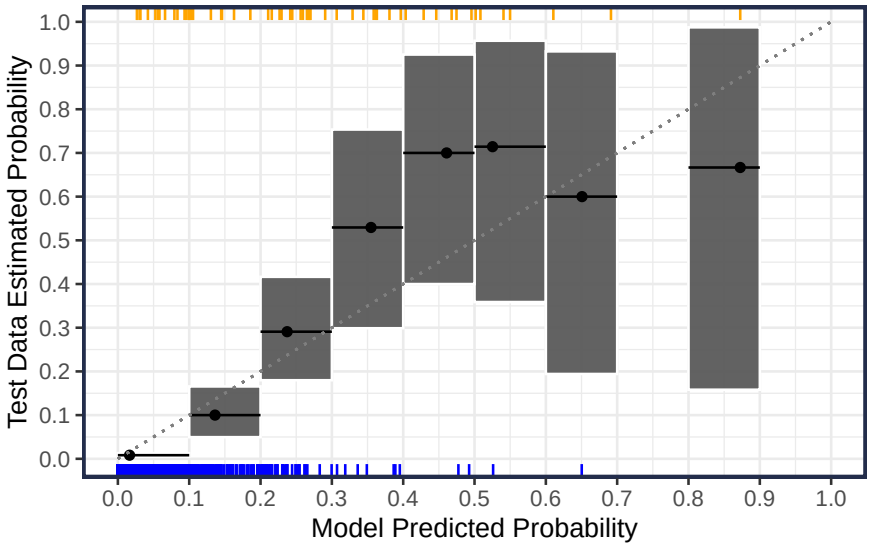
4.2.1 Densities



4.2.2 Binning

Here is an example of binning. I'll partition the predictions such that there are 10 groups of equal width.

bin	bin_lower	bin_upper	bin_avg	n	n_1	p_1	beta_lower	beta_upper	p_hat	moe
1	0.0	0.1	0.016	1817	14	0.008	0.005	0.013	0.008	0.004
2	0.2	0.3	0.237	53	15	0.291	0.180	0.416	0.283	0.121
3	0.1	0.2	0.136	98	9	0.100	0.050	0.166	0.092	0.057
4	0.3	0.4	0.355	15	8	0.529	0.299	0.753	0.533	0.252
5	0.5	0.6	0.525	5	4	0.714	0.359	0.957	0.800	0.351
6	0.4	0.5	0.461	8	6	0.700	0.400	0.925	0.750	0.300
7	0.6	0.7	0.651	3	2	0.600	0.194	0.932	0.667	0.533
8	0.8	0.9	0.873	1	1	0.667	0.158	0.987	1.000	0.000



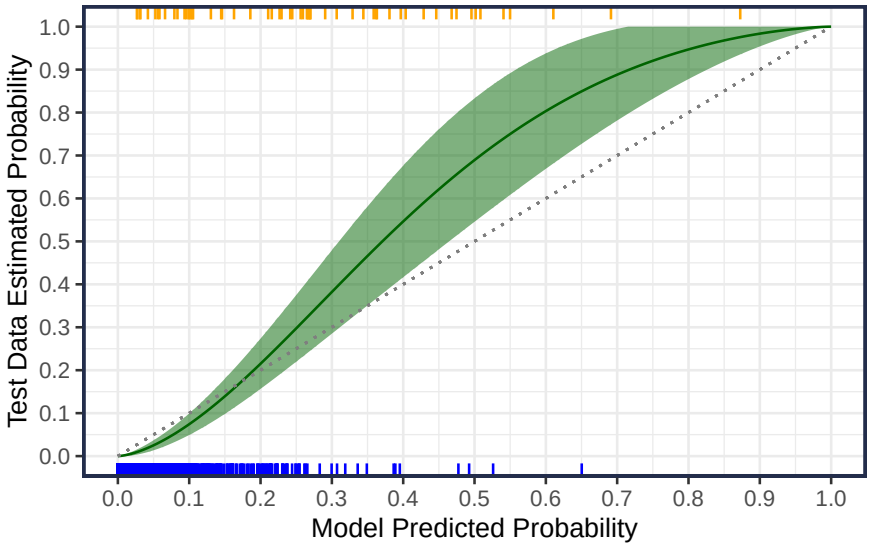
4.2.3 Smoothing

Instead of binning, we can use splines (or other smoothers) to estimate the relationship between $\hat{p}$ and p.

Use a logistic regression model with a smooth function of $\text{logit}(\hat{p})$.

- Recall $\text{logit}(\hat{p}) = \log \frac{\hat{p}}{1-\hat{p}}$

```
library(mgcv)
fit = mgcv::gam(
  default ~ s(gamma_hat, bs = "cs"),
  family = binomial(link = "logit"),
  data = preds %>% mutate(gamma_hat = log(p_hat) - log(1-p_hat))
)
```



4.3 Testing for Calibration

The main idea is to test the (null) hypothesis that

$$\Pr(Y = 1 \mid \hat{p}(x) = p) = p \qquad \text{for all } p$$

using *out-of-sample data* (e.g., *calibration/test data*).

This is equivalent to,

$$\Pr(Y = 1 \mid \hat{p}(x)) - \hat{p}(x) = 0 \qquad \text{for all } \hat{p}$$

4.4 Logistic Regression

For a calibrated model the estimated $\hat{p}(x)$ should be close to the true $p(x)$,

$$p(x) \approx \hat{p}(x)$$
$$\text{logit } p(x) \approx \text{logit } \hat{p}(x)$$

To test this, we introduce a bias term $b(x)$ and test for $b(x) = 0 \; \forall x$.

$$\text{logit } p(x) = b(x) + \text{logit } \hat{p}(x)$$

Notice the above expression is the same form as logistic regression.

This means we can use logistic regression to test for mis-calibration (predictive bias). To do this, use logit $\hat{p}(x)$ as an *offset* in the logistic regression model. An *offset* is a term that has a fixed weight/coefficient of 1.

We also need to specify the form of the bias term before we can test it. We give a few examples below.

4.5 Linear bias

To check for linear deviation, we specify the bias term as $b(x) = \beta_0 + \beta_1 \text{logit}\, \hat{p}(x)$.

$$\text{logit}\, p(x) = \beta_0 + \beta_1 \text{logit}\, \hat{p}(x) + (\text{logit}\, \hat{p}(x))$$

Fit on a hold-out set, and check how far β_0 and β_1 are from 0.

```
fit = glm(
        default ~ gamma_hat + offset(gamma_hat),
        family = binomial(link = "logit"),
        data = preds %>% mutate(gamma_hat = log(p_hat) - log(1-p_hat))
      )
# Note that gamma_hat = logit(p_hat) = log(p_hat) - log(1-p_hat)

fit %>% broom::tidy()
#> # A tibble: 2 x 5
#>   term        estimate std.error statistic  p.value
#>   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
#> 1 (Intercept)    0.927     0.307      3.02 0.00256
#> 2 gamma_hat      0.586     0.152      3.85 0.000116
```

4.6 Non-linear bias

We can introduce splines to detect non-linear deviations:

```
library(mgcv)
mgcv::gam(
  default ~ s(gamma_hat, bs = "bs") + offset(gamma_hat),
  family = binomial(link = "logit"),
  data = preds %>% mutate(gamma_hat = log(p_hat) - log(1-p_hat))
) %>%
  summary()
#>
#> Family: binomial
#> Link function: logit
#>
#> Formula:
#> default ~ s(gamma_hat, bs = "bs") + offset(gamma_hat)
#>
#> Parametric coefficients:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept)   -1.816      0.477   -3.81  0.00014 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Approximate significance of smooth terms:
#>               edf Ref.df Chi.sq p-value
#> s(gamma_hat)    1      1   14.8 0.00012 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> R-sq.(adj) =  0.316   Deviance explained = 6.45%
#> UBRE = -0.85354  Scale est. = 1         n = 2000
```

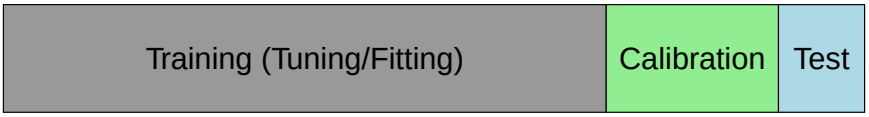
5 Corrective Recalibration: adjusting predictions

- A calibration model is a second-level model that use the predictions as a feature.
- We have already done this with risk scores (like COMPAS).
- We have also already done this with *stacking*.

Your Turn #4 : Connections to Stacking

Write out a stacking model with only 1 predictor.

5.1 Calibration Models



- $\hat{f}(x)$ is the tuned and fitted model.
- $\mathcal{D}_{cal}$ is the calibration data.
- We are trying to find a calibration function $b(\hat{y})$ that transforms the original predictions to work better on the test data.

$$\tilde{f}_{\text{cal}}(x) = \tilde{b}(\hat{f}(x))$$

- The function $b(\hat{y})$ should be monotonically increasing.

5.1.1 Monotonic Splines

- A good non-parametric option are monotonic splines. They produce smooth non-decreasing transformations of $\hat{f}(x)$.
  - by comparison, isotonic splines are usually step-functions (non-smooth)

Monotonic P-spline

```
library(mgcv)
calibrate_mpi = mgcv::gam(
  default ~ s(gamma_hat, bs = "mpi"),
  family = binomial(link = "logit"),
  data = preds %>% mutate(gamma_hat = log(p_hat) - log(1-p_hat))
)

summary(calibrate_mpi)
```

```
#>
#> Family: binomial
#> Link function: logit
#>
#> Formula:
#> default ~ s(gamma_hat, bs = "mpi")
#>
#> Parametric coefficients:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept)   -6.498      0.477   -13.6   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Approximate significance of smooth terms:
#>               edf Ref.df Chi.sq p-value
#> s(gamma_hat)    1      1    108  <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> R-sq.(adj) =  0.316   Deviance explained = 45.7%
#> UBRE = -0.85354  Scale est. = 1         n = 2000
```