Can an Algorithm Predict Who Reoffends? Evaluating COMPAS with Regularized Regression

Example Paper — Gov 51: Data Analysis and Politics

2026-01-01

1 Introduction

In 2016, the investigative journalism outlet ProPublica published an analysis that shook the criminal justice world. Examining COMPAS — a proprietary algorithm used by courts in Florida and elsewhere to predict the likelihood that a defendant would reoffend — the journalists found something striking: the algorithm appeared to classify Black defendants as higher risk than White defendants at roughly twice the rate, even when controlling for actual recidivism outcomes.

The controversy illuminated a deeper question that runs through data science and policy: **when we build a prediction model, what are we actually optimizing for, and at whose expense?** This paper uses the same dataset from ProPublica's analysis to illustrate how predictive models are built and evaluated, and to understand both their power and their limits.

Three questions organize the paper. First, how well can observable characteristics predict recidivism at all? Second, does shrinkage — regularizing the model with LASSO or ridge regression — improve prediction compared to a standard linear model? Third, does "predicting well" on average conceal systematic differences in prediction error across demographic groups?

The paper illustrates the fundamental concepts of supervised learning: training and test data, cross-validation for model selection, and the distinction between in-sample fit and out-of-sample prediction.

2 Data and Measurement

2.1 The COMPAS Dataset

The data come from Broward County, Florida, where the nonprofit news organization ProPublica obtained two years of COMPAS risk scores (2013–2014) and linked them to subsequent criminal records from the Broward County Sheriff's Office. The result is a dataset of individuals who were assessed with the COMPAS tool, with information on their demographics, criminal history, and whether they were rearrested within two years.

```
compas <- read_csv("data/compas_clean.csv", show_col_types = FALSE)

glimpse(compas)
```

```
Rows: 7,214
Columns: 17
$ recidivism      <dbl> 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, ~
$ age             <dbl> 69, 34, 24, 23, 43, 44, 41, 43, 39, 21, 27, 23, 37, 41~
$ female          <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, ~
$ black           <dbl> 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, ~
$ hispanic        <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, ~
$ asian           <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
$ other_race      <dbl> 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
$ native_american <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
$ priors_count    <dbl> 0, 0, 4, 1, 2, 0, 14, 3, 0, 1, 0, 3, 0, 0, 1, 7, 0, 3,~
$ felony          <dbl> 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, ~
$ juv_fel_count   <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
$ juv_misd_count  <dbl> 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
$ juv_other_count <dbl> 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
$ has_juv_felony  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
$ has_juv_misd    <dbl> 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
$ age_under_25    <dbl> 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, ~
$ age_25_45       <dbl> 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, ~
```

The dataset contains 7,214 individuals. The outcome variable is `recidivism` — equal to 1 if the individual was rearrested for a new offense within two years of the original assessment, and 0 otherwise.

2.2 Descriptive Statistics

```
compas |>
  summarise(
    `N`                       = n(),
    `Recidivism rate (%)`     = round(mean(recidivism) * 100, 1),
    `Mean age`                = round(mean(age), 1),
    `Share Black (%)`         = round(mean(black) * 100, 1),
    `Share female (%)`        = round(mean(female) * 100, 1),
    `Mean prior offenses`     = round(mean(priors_count), 1),
    `Share with felony (%)`   = round(mean(felony) * 100, 1)
  ) |>
  pivot_longer(everything(), names_to = "Variable", values_to = "Value") |>
  kable(caption = "Summary statistics, COMPAS dataset (Broward County, FL, 2013-2014)")
```

The overall recidivism rate is 45.1%. The sample is predominantly male and roughly 51% Black. Prior criminal history is the most commonly cited predictor of recidivism: the mean number of prior offenses is 3.5.

```
race_summary <- compas |>
  summarise(
    White = mean(recidivism[black == 0 & hispanic == 0 & asian == 0 & other_race == 0 & nati
    Black = mean(recidivism[black == 1]),
```

Table 1: **?(caption)**

(a) Summary statistics, COMPAS dataset (Broward County, FL, 2013–2014)

Variable	Value
N	7214.0
Recidivism rate (%)	45.1
Mean age	34.8
Share Black (%)	51.2
Share female (%)	19.3
Mean prior offenses	3.5
Share with felony (%)	64.7

```
    Hispanic = mean(recidivism[hispanic == 1]),
    Asian = mean(recidivism[asian == 1]),
    Other = mean(recidivism[other_race == 1])
  ) |>
  pivot_longer(everything(), names_to = "Race", values_to = "Rate") |>
  mutate(Race = factor(Race, levels = c("Black", "Hispanic", "Other", "White", "Asian")))

ggplot(race_summary, aes(x = Rate, y = Race, fill = Race == "Black")) +
  geom_col(width = 0.6) +
  geom_text(aes(label = percent(Rate, accuracy = 0.1)), hjust = -0.1, size = 3.5) +
  scale_fill_manual(values = c("TRUE" = "#2166AC", "FALSE" = "#B2C8D9"), guide = "none") +
  scale_x_continuous(labels = percent_format(), limits = c(0, 0.65)) +
  labs(
    title    = "Raw Recidivism Rates Vary Substantially by Race",
    subtitle = "Share rearrested within 2 years, unadjusted",
    x        = "Recidivism rate",
    y        = NULL,
    caption  = "Source: ProPublica/Broward County Sheriff's Office"
  ) +
  theme_minimal(base_size = 11) +
  theme(panel.grid.minor = element_blank())
```

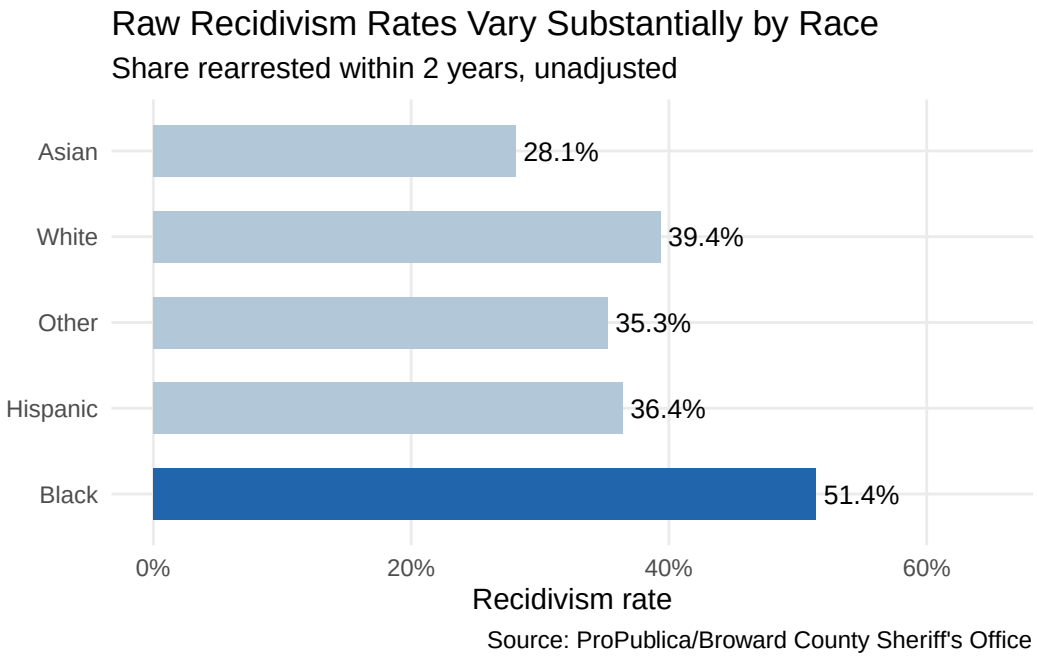

Figure 1: Recidivism rates by race/ethnicity in the COMPAS dataset. Rates are not adjusted for criminal history, age, or any other covariates.

Figure 1 shows raw recidivism rates by race. Black defendants in this dataset recidivate at higher raw rates than White defendants. But this unadjusted comparison conflates race with age, criminal history, and other factors that also predict recidivism. A critical question — which we will return to — is whether these raw differences persist after statistical adjustment, and whether the model's errors are distributed equally across groups.

3 Prediction: From OLS to Regularized Regression

3.1 Train/Test Split

A central discipline in predictive modeling is evaluating a model on data it has **never seen**. A model that fits its training data perfectly may fail on new cases — a phenomenon called overfitting. To guard against overfitting, we split the data into a training set (80%) and a test set (20%) before any model is estimated.

```
set.seed(8675309)   # set seed for reproducibility

n <- nrow(compas)
train_idx <- sample(seq_len(n), size = floor(0.8 * n))

compas_train <- compas[train_idx, ]
compas_test  <- compas[-train_idx, ]

cat("Training set:", nrow(compas_train), "observations\n")
```

```
Training set: 5771 observations
```

```r
cat("Test set:    ", nrow(compas_test),  "observations\n")
```

```
Test set:     1443 observations
```

All model fitting happens on `compas_train`. Performance evaluation happens on `compas_test`. This sequence is strict: we look at the test set results **only once**, after all modeling decisions are final. Peeking at the test set to guide model building inflates apparent performance.

3.2 Baseline: OLS (Linear Probability Model)

The simplest approach is to regress the binary outcome on all available predictors using ordinary least squares. Because the outcome is binary (0/1), this is called a **linear probability model** — each coefficient is interpreted as a percentage-point change in the probability of recidivism.

```r
# Predictors: everything except recidivism
X_vars <- setdiff(names(compas), "recidivism")

formula_full <- as.formula(
  paste("recidivism ~", paste(X_vars, collapse = " + "))
)

ols_fit <- lm(formula_full, data = compas_train)

# Predict on test set
ols_pred  <- predict(ols_fit, newdata = compas_test)
ols_class <- as.integer(ols_pred >= 0.5)

# Performance
ols_rmse     <- sqrt(mean((compas_test$recidivism - ols_pred)^2))
ols_accuracy <- mean(ols_class == compas_test$recidivism)

cat(sprintf("OLS - Test RMSE: %.4f  |  Accuracy: %.3f\n", ols_rmse, ols_accuracy))
```

```
OLS - Test RMSE: 0.4650  |  Accuracy: 0.661
```

The linear probability model gives us a baseline. But OLS treats all predictors as equally important and does no automatic variable selection. With 16 predictors and many that may be noisy or redundant, regularization methods can potentially improve out-of-sample performance.

3.3 LASSO

LASSO (Least Absolute Shrinkage and Selection Operator) adds a penalty to the OLS objective function proportional to the sum of the **absolute values** of the coefficients:

$$\hat{\beta}^{\text{LASSO}} = \arg\min_{\beta} \sum_{i=1}^{n} (Y_i - X_i'\beta)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

The tuning parameter $\lambda \geq 0$ controls the strength of the penalty. When $\lambda = 0$, LASSO reduces to OLS. As λ increases, LASSO shrinks coefficients toward zero and, crucially, sets some coefficients

exactly to zero — effectively performing variable selection. This makes LASSO interpretable: it identifies which predictors survive penalization.

The penalty parameter λ is chosen by **10-fold cross-validation**: the training data is split into 10 folds, the model is fit on 9 folds and evaluated on the held-out fold, and this is repeated across all folds and all values of λ in a grid. We select the λ that minimizes mean cross-validation error.

```
# Build matrix form required by glmnet
X_train <- model.matrix(formula_full, data = compas_train)[, -1]
y_train <- compas_train$recidivism

X_test  <- model.matrix(formula_full, data = compas_test)[, -1]
y_test  <- compas_test$recidivism

set.seed(42)
lasso_cv <- cv.glmnet(X_train, y_train, alpha = 1, nfolds = 10)

plot(lasso_cv)
```

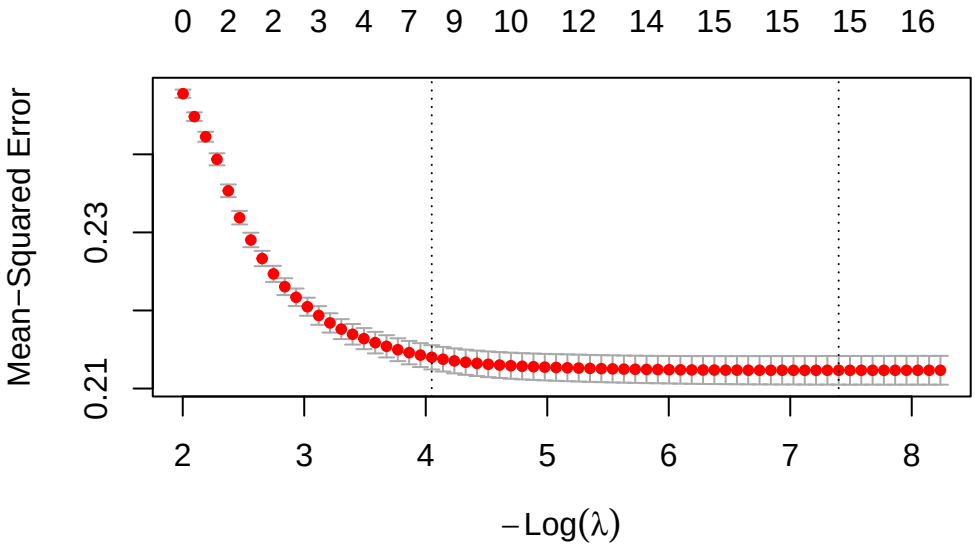

Figure 2: 10-fold cross-validation curve for LASSO. The x-axis shows log( ) and the y-axis shows mean squared error averaged across folds. The left dashed line is _min (minimum CV error); the right is _1se (most regularized model within one standard error of the minimum). Numbers at the top show how many predictors have nonzero coefficients at each .

```
# Predict with lambda_min
lasso_pred  <- predict(lasso_cv, newx = X_test, s = "lambda.min", type = "response")[, 1]
lasso_class <- as.integer(lasso_pred >= 0.5)

lasso_rmse     <- sqrt(mean((y_test - lasso_pred)^2))
lasso_accuracy <- mean(lasso_class == y_test)
```

```r
cat(sprintf("LASSO -  _min = %.4f   |  Test RMSE: %.4f   |  Accuracy: %.3f\n",
            lasso_cv$lambda.min, lasso_rmse, lasso_accuracy))
```

```
LASSO -  _min = 0.0006   |  Test RMSE: 0.4649   |  Accuracy: 0.662
```

```r
# Which variables survive?
lasso_coef <- coef(lasso_cv, s = "lambda.min")
nonzero    <- lasso_coef[lasso_coef[, 1] != 0, , drop = FALSE]
cat(sprintf("Nonzero coefficients: %d of %d\n", nrow(nonzero) - 1, ncol(X_train)))
```

```
Nonzero coefficients: 15 of 16
```

The cross-validation curve in Figure 2 shows how prediction error changes across the range of λ. At very small λ (left side), LASSO is close to OLS and may overfit. At very large λ (right side), the model is over-regularized and underfits. The optimal λ sits in the middle.

3.4 Ridge Regression

Ridge regression uses a different penalty: the sum of **squared** coefficients:

$$\hat{\beta}^{\text{Ridge}} = \arg\min_{\beta} \sum_{i=1}^{n} (Y_i - X_i'\beta)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

Unlike LASSO, ridge never sets coefficients exactly to zero. Instead, it shrinks all coefficients toward zero proportionally. Ridge tends to perform better than LASSO when many predictors have small but non-negligible effects. LASSO performs better when only a few predictors truly matter.

```r
set.seed(42)
ridge_cv <- cv.glmnet(X_train, y_train, alpha = 0, nfolds = 10)

ridge_pred  <- predict(ridge_cv, newx = X_test, s = "lambda.min", type = "response")[, 1]
ridge_class <- as.integer(ridge_pred >= 0.5)

ridge_rmse     <- sqrt(mean((y_test - ridge_pred)^2))
ridge_accuracy <- mean(ridge_class == y_test)

cat(sprintf("Ridge -  _min = %.4f   |  Test RMSE: %.4f   |  Accuracy: %.3f\n",
            ridge_cv$lambda.min, ridge_rmse, ridge_accuracy))
```

```
Ridge -  _min = 0.0135   |  Test RMSE: 0.4649   |  Accuracy: 0.663
```

3.5 Comparing the Three Models

```r
tibble(
  Model    = c("OLS (no regularization)", "LASSO (  = 1)", "Ridge (  = 0)"),
  `Test RMSE`  = round(c(ols_rmse, lasso_rmse, ridge_rmse), 4),
  `Test Accuracy` = percent(c(ols_accuracy, lasso_accuracy, ridge_accuracy), accuracy = 0.1)
```

Table 2: **?(caption)**

(a) Out-of-sample prediction performance: OLS vs. LASSO vs. Ridge

Model	Test RMSE	Test Accuracy
OLS (no regularization)	0.4650	66.1%
LASSO (= 1)	0.4649	66.2%
Ridge (= 0)	0.4649	66.3%

```
) |>
  kable(caption = "Out-of-sample prediction performance: OLS vs. LASSO vs. Ridge")
```

```
# Coefficient path
lasso_fit <- glmnet(X_train, y_train, alpha = 1)

# Tidy the coefficient matrix
coef_mat <- as.matrix(coef(lasso_fit))[-1, ]    # drop intercept
lambda_seq <- lasso_fit$lambda

coef_df <- as_tibble(t(coef_mat)) |>
  mutate(lambda = lambda_seq) |>
  pivot_longer(-lambda, names_to = "variable", values_to = "coef")

# Identify top survivors at lambda_min
survivors <- rownames(nonzero)[-1]   # drop "(Intercept)"

coef_df_highlight <- coef_df |>
  mutate(highlighted = variable %in% survivors)

ggplot(coef_df_highlight, aes(x = log(lambda), y = coef,
                              group = variable,
                              color = highlighted,
                              linewidth = highlighted)) +
  geom_line(alpha = 0.7) +
  geom_vline(xintercept = log(lasso_cv$lambda.min), linetype = "dashed",
             color = "#2166AC", linewidth = 0.5) +
  scale_color_manual(values = c("TRUE" = "#D6604D", "FALSE" = "gray70"),
                     labels = c("TRUE" = "Survives lambda.min", "FALSE" = "Shrinks to zero")
                     name = NULL) +
  scale_linewidth_manual(values = c("TRUE" = 0.9, "FALSE" = 0.3), guide = "none") +
  labs(
    title    = "LASSO Selects a Sparse Set of Predictors",
    subtitle = "Coefficient paths as regularization strength (lambda) increases",
    x        = "log(lambda)  [<-- less regularization | more regularization -->]",
    y        = "Coefficient",
```

```
  caption  = "Vertical line: lambda chosen by 10-fold CV"
) +
theme_minimal(base_size = 11) +
theme(legend.position = "bottom")
```

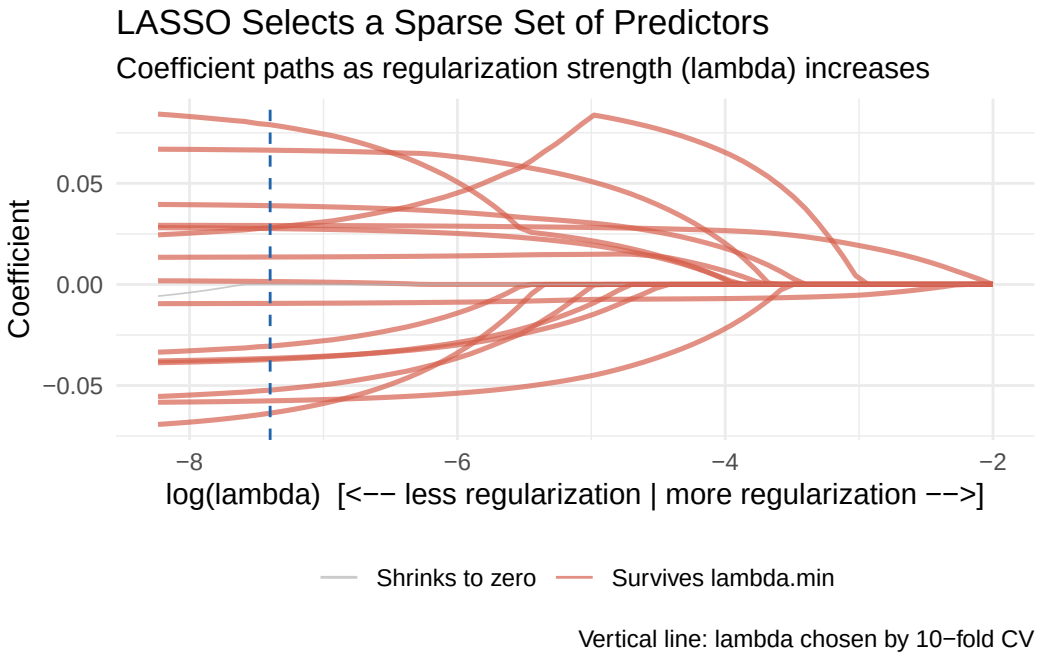

Figure 3: LASSO coefficient paths. Each line traces one predictor's coefficient as increases (right to left). Predictors that remain nonzero at the optimal (vertical dashed line) are the 'selected' variables. Prior offenses and age dominate; many variables shrink to zero.

Figure 3 shows the LASSO coefficient paths. As λ increases, predictors are progressively set to zero. The predictors that survive at the optimal λ are the model's "selected" features. In this data, prior criminal history and age are the dominant predictors — consistent with decades of criminological research.

4 Fairness: Does the Model Err Equally?

A model that achieves high overall accuracy can still systematically misflag one group more than another. Two types of errors matter in this context:

- **False positive rate**: the model predicts high risk but the person does *not* reoffend
- **False negative rate**: the model predicts low risk but the person *does* reoffend

ProPublica's original analysis focused on the false positive rate disparity: Black defendants who did not reoffend were classified as high risk at nearly twice the rate of White defendants. This is not a statistical artifact — it is a direct consequence of using criminal history as a predictor in a society where prior policing has been unevenly distributed across racial groups.

Table 3: **?(caption)**

(a) LASSO prediction error rates by race. False positive rate = share predicted high-risk who did NOT reoffend.

race	N	True positive rate	False positive rate	Accuracy
Black	732	0.714	0.397	0.658
Hispanic	133	0.340	0.151	0.669
White/Other	578	0.387	0.163	0.664

```
# Use LASSO predictions (lambda.min)
test_results <- compas_test |>
  mutate(
    pred_prob  = lasso_pred,
    pred_class = lasso_class
  )

fairness_table <- test_results |>
  mutate(
    race = case_when(
      black    == 1 ~ "Black",
      hispanic == 1 ~ "Hispanic",
      TRUE          ~ "White/Other"
    )
  ) |>
  group_by(race) |>
  summarise(
    N             = n(),
    `True positive rate`  = round(mean(pred_class[recidivism == 1] == 1), 3),
    `False positive rate` = round(mean(pred_class[recidivism == 0] == 1), 3),
    `Accuracy`            = round(mean(pred_class == recidivism), 3),
    .groups = "drop"
  )

kable(fairness_table,
      caption = "LASSO prediction error rates by race. False positive rate = share predicted
```

The false positive rate comparison in Table 3 is the central fairness question. If Black defendants have a higher false positive rate than White defendants, then Black individuals who will not reoffend are being incorrectly flagged for extra scrutiny at higher rates. This is not a problem the model can easily fix: the same historical criminal record that generates statistical predictive power also encodes the history of differential policing and prosecution.

5 Limitations

The linear probability model and its regularized cousins predict well on average but produce predictions outside $[0, 1]$ and are not fully calibrated probability estimates. A logistic regression or

gradient boosted tree would be more natural for binary outcomes; the tradeoff is interpretability.

Cross-validation selects λ to minimize mean squared error on the training folds. This does not optimize for fairness, calibration, or any other metric. Different objective functions would select different models.

Most critically: the data come from one county over two years. A model trained here may not generalize to other jurisdictions, time periods, or demographic profiles. Prediction models exported from development contexts routinely degrade when deployed elsewhere — a failure mode well documented in the COMPAS literature.

Finally, this paper treats recidivism as a clean binary outcome. In practice, "recidivism" as measured here is rearrest, not reoffending — and arrest rates are themselves a function of policing intensity. The outcome variable inherits the same biases as the predictors.

6 Conclusion

Three findings stand out. First, demographic and criminal history variables predict recidivism at roughly 65–67% accuracy — better than chance, but far from determinative. Second, regularization (LASSO and ridge) does not dramatically improve on OLS in this dataset, suggesting that with 16 predictors and 5,771 training observations, overfitting is not the main problem. Third, prediction accuracy masks systematic differences in error rates across racial groups — the false positive rate is higher for Black defendants, meaning the model imposes costs disproportionately on a group that will not actually reoffend.

Prediction is powerful but not neutral. A model that predicts well on aggregate can still be unjust in its distribution of errors. Evaluating a prediction system requires looking not just at overall accuracy, but at *whose* errors you are willing to accept.

This paper was produced using Quarto with R. All code is included above. Data are from ProPublica's COMPAS analysis (Angwin, Larson, Mattu, and Kirchner, 2016, "Machine Bias," ProPublica). The cleaned dataset used here contains 7,214 individuals assessed in Broward County, Florida, 2013–2014.