DSCI 726: Project 4 — Predictive Analysis for Book Recommendations

Becca Halford

2025-10-03

Code ▾

Executive Summary

This project predicts the star rating of new books by authors who have published at least three titles on Amazon. I used a random 80/20 split by book title to create the training and test sets.

The modeling methods include a baseline model, a simple linear model, a full linear model, and a pruned decision tree. I evaluated model performance using RMSE, R^2, and the train-test performance gap. I also performed 5-fold cross-validation on the full linear model to assess its stability and generalizability.

1. Data Preparation

The column "save" was renamed to "savings" for clarity. The column "size" was removed due to redundancy. Duplicate rows were removed. Convert applicable columns to integer.

1.1 Load & Clean

Hide

```
kindle_data <- read_excel('cleaned_kindle_books.xlsx') |>
  select(-size) |>
  mutate(across(c(pages, author_num_books, customer_reviews), as.integer)) |>
  distinct(title, .keep_all = TRUE) |>
  rename(savings = save)

cat("Dataset:", nrow(kindle_data), "books ×", ncol(kindle_data), "variables\n")
```

```
## Dataset: 44739 books × 11 variables
```

Hide

```
glimpse(kindle_data)
```

```
## Rows: 44,739
## Columns: 11
## $ title            <chr> "Middle School Confidential 1: The Alexa Black Diaries …
## $ author           <chr> "@SororityProblem", "@SororityProblem", "@SororityPro…
## $ price            <dbl> 4.99, 4.99, 4.99, 4.99, 0.99, 2.99, 2.99, 15.49, 26.8…
## $ savings          <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 3.96, 0.00, 12.46, 13.0…
## $ pages            <int> 117, 65, 72, 63, 115, 41, 175, 472, 330, 285, 336, 40…
## $ publisher        <chr> "Self-Published", "Self-Published", "Self-Published",…
## $ customer_reviews <int> 4, 3, 7, 2, 30, 6, 4, 1, 10, 6, 219, 754, 359, 11, 7,…
## $ stars            <dbl> 3.0, 4.7, 5.0, 3.0, 2.8, 4.7, 5.0, 3.0, 4.6, 3.2, 4.3…
## $ author_num_books <int> 4, 4, 4, 4, 1, 1, 1, 1, 2, 1, 1, 3, 3, 3, 1, 1, 2, 1, 1,…
## $ avg_stars        <dbl> 3.9, 3.9, 3.9, 3.9, 2.8, 4.7, 5.0, 3.0, 3.4, 3.2, 4.4…
## $ discount_pct     <dbl> 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 0.43, 1.00, 0.55, 0.67,…
```

1.2 Filter by author productivity (≤2 books removed)

Authors with 2 or less books were removed from the dataset to more accurately represent stars and avg_stars.

Hide

```
original_n <- nrow(kindle_data)
author_counts <- kindle_data |> count(author, name = "n_books")
few_book_authors <- author_counts |> filter(n_books <= 2) |> pull(author)

kindle_data <- kindle_data |> filter(!author %in% few_book_authors)

cat("Removed:", original_n - nrow(kindle_data), "rows (",
    round(100*(original_n - nrow(kindle_data))/original_n, 1), "%)\n", sep = "")
```

```
## Removed:24192rows (54.1%)
```

1.3 Train/Test Split

I split the data using a simple random sample by book title after completing the filtering process. The final split allocated 80% of the data to the training set (16,438 rows) and 20% to the test set (4,109 rows).

Hide

```
set.seed(123)
test_titles <- sample(kindle_data$title, size = floor(0.2 * nrow(kindle_data)))
train_set <- kindle_data |> filter(!title %in% test_titles)
test_set  <- kindle_data |> filter(title %in% test_titles)

cat("Train/Test sizes:", nrow(train_set), "/", nrow(test_set), "\n")
```

```
## Train/Test sizes: 16438 / 4109
```

2. Feature Engineering

A log transformation was performed due to highly skewed variables.

2.1 Transformations

Hide

```
cols_to_log <- c("price", "customer_reviews", "pages", "savings")

train_set <- train_set |>
  mutate(across(all_of(cols_to_log), ~log(.x + 1), .names = "log_{.col}"))
test_set <- test_set |>
  mutate(across(all_of(cols_to_log), ~log(.x + 1), .names = "log_{.col}"))

cat("Log-transformed:", paste(cols_to_log, collapse = ", "), "\n")
```

```
## Log-transformed: price, customer_reviews, pages, savings
```

Log transformation was unsuccessful on 'author_num_books', so binning was applied.

2.2 Bin `author_num_books` (quartiles)

Hide

```
quartiles <- quantile(train_set$author_num_books, probs = c(0, .25, .5, .75, 1), na.rm = TRUE)

train_set <- train_set |>
  mutate(author_books_bin = cut(author_num_books, breaks = quartiles,
                                labels = c("Few_Q1","Some_Q2","Many_Q3","Prolific_Q4"),
                                include.lowest = TRUE))

test_set <- test_set |>
  mutate(author_books_bin = cut(author_num_books, breaks = quartiles,
                                labels = c("Few_Q1","Some_Q2","Many_Q3","Prolific_Q4"),
                                include.lowest = TRUE))

table(train_set$author_books_bin)
```

```
##
##      Few_Q1     Some_Q2     Many_Q3 Prolific_Q4
##        5061        3250        4097        4030
```

2.3 Scale numeric features (train params)

Hide

```
scale_cols <- c(paste0("log_", cols_to_log), "discount_pct", "avg_stars")

scaling_params <- train_set |>
  summarise(across(all_of(scale_cols), list(mean = ~mean(.x, na.rm = TRUE),
                                            sd   = ~sd(.x, na.rm = TRUE))))

train_set_scaled <- train_set; test_set_scaled <- test_set
for (col in scale_cols) {
  m <- scaling_params[[paste0(col, "_mean")]]; s <- scaling_params[[paste0(col, "_sd")]]
  train_set_scaled[[col]] <- (train_set_scaled[[col]] - m) / s
  test_set_scaled[[col]]  <- (test_set_scaled[[col]] - m) / s
}

train_set_scaled <- train_set_scaled |> mutate(author_books_bin = as.factor(author_books_bin))
test_set_scaled  <- test_set_scaled |> mutate(author_books_bin = factor(author_books_bin,
                                                levels = levels(train_set_scaled$author_
books_bin)))
# one-hot for binned variable
dummies <- dummyVars(~ author_books_bin, data = train_set_scaled)
train_cat <- as.data.frame(predict(dummies, newdata = train_set_scaled))
test_cat  <- as.data.frame(predict(dummies, newdata = test_set_scaled))

numeric_features <- c("log_price","log_customer_reviews","log_pages","log_savings","avg_stars","discount_pct","st
ars")

train_model <- cbind(train_set_scaled |> select(all_of(numeric_features)), train_cat)
test_model  <- cbind(test_set_scaled |> select(all_of(numeric_features)), test_cat)

cat("Model matrices:", dim(train_model)[1], "×", dim(train_model)[2], "(train);\n",
    dim(test_model)[1], "×", dim(test_model)[2], "(test)\n")
```

```
## Model matrices: 16438 × 11 (train);
##  4109 × 11 (test)
```

3. Baselines (for context)

I include two simple baselines to set expectations: Null Mean: predict the training mean rating. Author-only OLS: predict with `avg_stars` only.

Hide

```
null_pred <- rep(mean(train_model$stars), nrow(test_model))
null_rmse <- calculate_rmse(test_model$stars, null_pred)

author_only <- lm(stars ~ avg_stars, data = train_model)
author_only_res <- evaluate_model(author_only, train_model, test_model)
author_rmse <- author_only_res$test_rmse

cat("Baseline RMSE (Null Mean):", round(null_rmse, 4), "\n")
```

```
## Baseline RMSE (Null Mean): 0.4797
```

Hide

```
cat("Baseline RMSE (Author-only):", round(author_rmse, 4), "\n")
```

```
## Baseline RMSE (Author-only): 0.3571
```

4. Correlation & Collinearity Checks

For linear regression I check VIF and explicitly remove a redundant predictor to avoid multicollinearity (`discount_pct` is dropped in favor of `log_savings`).
For tree-based models I include use the pairwise correlation technique to flag redundancy.

Hide

```
numeric_predictors <- c("log_price","log_customer_reviews","log_pages","log_savings","avg_stars","discount_pct")
cor_matrix <- cor(train_model |> select(all_of(numeric_predictors)), use = "pairwise.complete.obs")

par(oma = c(4, 0, 0, 0))
corrplot(cor_matrix, method = "color", type = "upper",
         tl.col = "black", tl.cex = 0.8, addCoef.col = "black",
         number.cex = 0.7, diag = FALSE)
mtext("Correlation Matrix — Multicollinearity Check", side = 1, line = .25, outer = TRUE, cex = 1)
```

5. Models

5.1 Simple OLS (baseline comparator; not useful)

I tried a limited OLS (log price, log reviews, log pages) purely as a baseline for comparison. It had low R² and has no predictive value.

Hide

```
simple_model <- lm(stars ~ log_price + log_customer_reviews + log_pages, data = train_model)
simple_res   <- evaluate_model(simple_model, train_model, test_model)
simple_r2    <- summary(simple_model)$r.squared
cat("Simple OLS — Train/Test RMSE:", round(simple_res$train_rmse, 4), "/", round(simple_res$test_rmse, 4),
    "; R² =", round(simple_r2, 4), "\n")
```

```
## Simple OLS — Train/Test RMSE: 0.4755 / 0.4788 ; R² = 0.0076
```

Hide

```
vif_simple <- check_vif(simple_model, "Simple OLS")
```

```
##
##
## Table: VIF Analysis — Simple OLS
##
## |                    |Variable             | VIF|Status |
## |:-------------------|:--------------------|----:|:------|
## |log_price           |log_price            | 1.25|Keep   |
## |log_customer_reviews |log_customer_reviews | 1.14|Keep   |
## |log_pages           |log_pages            | 1.39|Keep   |
```

Hide

```
write_csv(vif_simple, file.path(tables_dir, "vif_simple.csv"))
```

5.2 Full OLS

I exclude `discount_pct` (and keep `log_savings`) in the linear model to avoid multicollinearity. These two variables are strongly related and dropping one helps stabilize coefficient estimates.

Hide

```
full_model <- lm(stars ~ log_price + log_customer_reviews + log_pages +
                   log_savings + avg_stars +
                   author_books_bin.Some_Q2 + author_books_bin.Many_Q3 + author_books_bin.Prolific_Q4,
                 data = train_model)

summary(full_model)
```

```
##
## Call:
## lm(formula = stars ~ log_price + log_customer_reviews + log_pages +
##     log_savings + avg_stars + author_books_bin.Some_Q2 + author_books_bin.Many_Q3 +
##     author_books_bin.Prolific_Q4, data = train_model)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3835 -0.1675  0.0209  0.1999  2.1184
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  4.260385   0.005001 851.968  < 2e-16 ***
## log_price                    0.019315   0.003186   6.062 1.38e-09 ***
## log_customer_reviews        -0.031780   0.002994 -10.613  < 2e-16 ***
## log_pages                    0.022097   0.003338   6.619 3.73e-11 ***
## log_savings                 -0.002092   0.003143  -0.666    0.506
## avg_stars                    0.321245   0.002774 115.819  < 2e-16 ***
## author_books_bin.Some_Q2     0.003198   0.007929   0.403    0.687
## author_books_bin.Many_Q3     0.004889   0.007451   0.656    0.512
## author_books_bin.Prolific_Q4 0.016842   0.007617   2.211    0.027 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.352 on 16429 degrees of freedom
## Multiple R-squared:  0.4566, Adjusted R-squared:  0.4563
## F-statistic:  1725 on 8 and 16429 DF,  p-value: < 2.2e-16
```

Hide

```
full_res  <- evaluate_model(full_model, train_model, test_model)
full_r2   <- summary(full_model)$r.squared
vif_full  <- check_vif(full_model, "Full OLS")
```

```
##
##
## Table: VIF Analysis — Full OLS
##
## |                            |Variable                     |  VIF|Status |
## |:---------------------------|:----------------------------|----:|:------|
## |log_price                   |log_price                    | 1.35|Keep   |
## |log_customer_reviews        |log_customer_reviews         | 1.19|Keep   |
## |log_pages                   |log_pages                    | 1.48|Keep   |
## |log_savings                 |log_savings                  | 1.31|Keep   |
## |avg_stars                   |avg_stars                    | 1.02|Keep   |
## |author_books_bin.Some_Q2    |author_books_bin.Some_Q2     | 1.32|Keep   |
## |author_books_bin.Many_Q3    |author_books_bin.Many_Q3     | 1.38|Keep   |
## |author_books_bin.Prolific_Q4 |author_books_bin.Prolific_Q4 | 1.42|Keep   |
```

Hide

```
write_csv(vif_full, file.path(tables_dir, "vif_full.csv"))

cat("Full OLS — Train/Test RMSE:", round(full_res$train_rmse, 4), "/", round(full_res$test_rmse, 4),
    "; R² =", round(full_r2, 4), "\n")
```

```
## Full OLS — Train/Test RMSE: 0.3519 / 0.3553 ; R² = 0.4566
```

5.3 Decision Tree (with cp-based pruning)

I still allow `discount_pct` in the tree since trees are less sensitive to multicollinearity.

Hide

```
tree_model <- rpart(stars ~ log_price + log_customer_reviews + log_pages +
                      log_savings + avg_stars + discount_pct +
                      author_books_bin.Some_Q2 + author_books_bin.Many_Q3 + author_books_bin.Prolific_Q4,
                    data = train_model,
                    control = rpart.control(maxdepth = 5, minsplit = 100, cp = 0.01))

printcp(tree_model)
```

```
##
## Regression tree:
## rpart(formula = stars ~ log_price + log_customer_reviews + log_pages +
##     log_savings + avg_stars + discount_pct + author_books_bin.Some_Q2 +
##     author_books_bin.Many_Q3 + author_books_bin.Prolific_Q4,
##     data = train_model, control = rpart.control(maxdepth = 5,
##         minsplit = 100, cp = 0.01))
##
## Variables actually used in tree construction:
## [1] avg_stars
##
## Root node error: 3745.6/16438 = 0.22786
##
## n= 16438
##
##         CP nsplit rel error  xerror     xstd
## 1 0.279993      0   1.00000 1.00008 0.019240
## 2 0.069476      1   0.72001 0.72028 0.015365
## 3 0.043242      2   0.65053 0.65081 0.013176
## 4 0.017596      3   0.60729 0.60757 0.013049
## 5 0.012351      4   0.58969 0.59071 0.012406
## 6 0.010000      5   0.57734 0.57838 0.012182
```

Hide

```
best_cp <- tree_model$cptable[which.min(tree_model$cptable[,"xerror"]), "CP"]
tree_model <- prune(tree_model, cp = best_cp)

tree_res <- evaluate_model(tree_model, train_model, test_model)
cat("Decision Tree (pruned) — Train/Test RMSE:",
    round(tree_res$train_rmse, 4), "/", round(tree_res$test_rmse, 4), "\n")
```

```
## Decision Tree (pruned) — Train/Test RMSE: 0.3627 / 0.3672
```

5.4 5-fold Cross-Validation (Full OLS)

Hide

```
perform_cv <- function(formula, data, k = 5) {
  set.seed(123)
  n <- nrow(data); fold_size <- floor(n / k); shuffled <- sample(seq_len(n)); cv_errors <- numeric(k)
  for (i in 1:k) {
    idx <- if (i < k) shuffled[((i-1)*fold_size+1):(i*fold_size)] else shuffled[((i-1)*fold_size+1):n]
    m   <- lm(formula, data = data[-idx, ])
    pr  <- predict(m, newdata = data[idx, ])
    cv_errors[i] <- calculate_rmse(data$stars[idx], pr)
  }
  cv_errors
}

full_formula <- formula(stars ~ log_price + log_customer_reviews + log_pages +
                          log_savings + avg_stars +
                          author_books_bin.Some_Q2 + author_books_bin.Many_Q3 + author_books_bin.Prolific_
Q4)

cv_errors <- perform_cv(full_formula, train_model, k = 5)
cv_mean_rmse <- mean(cv_errors); cv_sd_rmse <- sd(cv_errors)

cat("5-fold CV RMSEs:", paste(round(cv_errors, 4), collapse = ", "), "\n",
    "Mean (SD):", round(cv_mean_rmse, 4), "(", round(cv_sd_rmse, 4), ")\n")
```

```
## 5-fold CV RMSEs: 0.3383, 0.361, 0.3648, 0.3561, 0.3395
##  Mean (SD): 0.3519 ( 0.0123 )
```

6. Evaluation & Comparison

Hide

```
simple_test_rmse <- simple_res$test_rmse
full_test_rmse   <- full_res$test_rmse
tree_test_rmse   <- tree_res$test_rmse

model_results <- tibble::tibble(
  Model      = c("Null Mean","Author-only","Simple OLS","Full OLS","Decision Tree (pruned)"),
  Predictors = c(0, 1, 3, 8, 9),
  Train_RMSE = c(calculate_rmse(train_model$stars, rep(mean(train_model$stars), nrow(train_model))),
                 author_only_res$train_rmse, simple_res$train_rmse, full_res$train_rmse, tree_res$train_rmse),
  Test_RMSE  = c(null_rmse, author_rmse, simple_test_rmse, full_test_rmse, tree_test_rmse),
  R_squared  = c(NA,
                 summary(author_only)$r.squared,
                 summary(simple_model)$r.squared,
                 summary(full_model)$r.squared,
                 cor(tree_res$test_pred, test_model$stars)^2),
  CV_RMSE    = c(NA, NA, NA, cv_mean_rmse, NA),
  Overfit_Gap = Test_RMSE - Train_RMSE
) |>
  mutate(across(where(is.numeric), ~round(.x, 4)))

kable(model_results, caption = "Model Performance Comparison")
```

Model Performance Comparison

Model	Predictors	Train_RMSE	Test_RMSE	R_squared	CV_RMSE	Overfit_Gap
Null Mean	0	0.4774	0.4797	NA	NA	0.0023
Author-only	1	0.3540	0.3571	0.4499	NA	0.0031
Simple OLS	3	0.4755	0.4788	0.0076	NA	0.0032
Full OLS	8	0.3519	0.3553	0.4566	0.3519	0.0034
Decision Tree (pruned)	9	0.3627	0.3672	0.4138	NA	0.0045

Hide

```
write_csv(model_results, file.path(tables_dir, "model_comparison.csv"))

best_idx <- which.min(model_results$Test_RMSE)
cat("Best by Test RMSE:", model_results$Model[best_idx], "with RMSE =", model_results$Test_RMSE[best_idx], "\n")
```

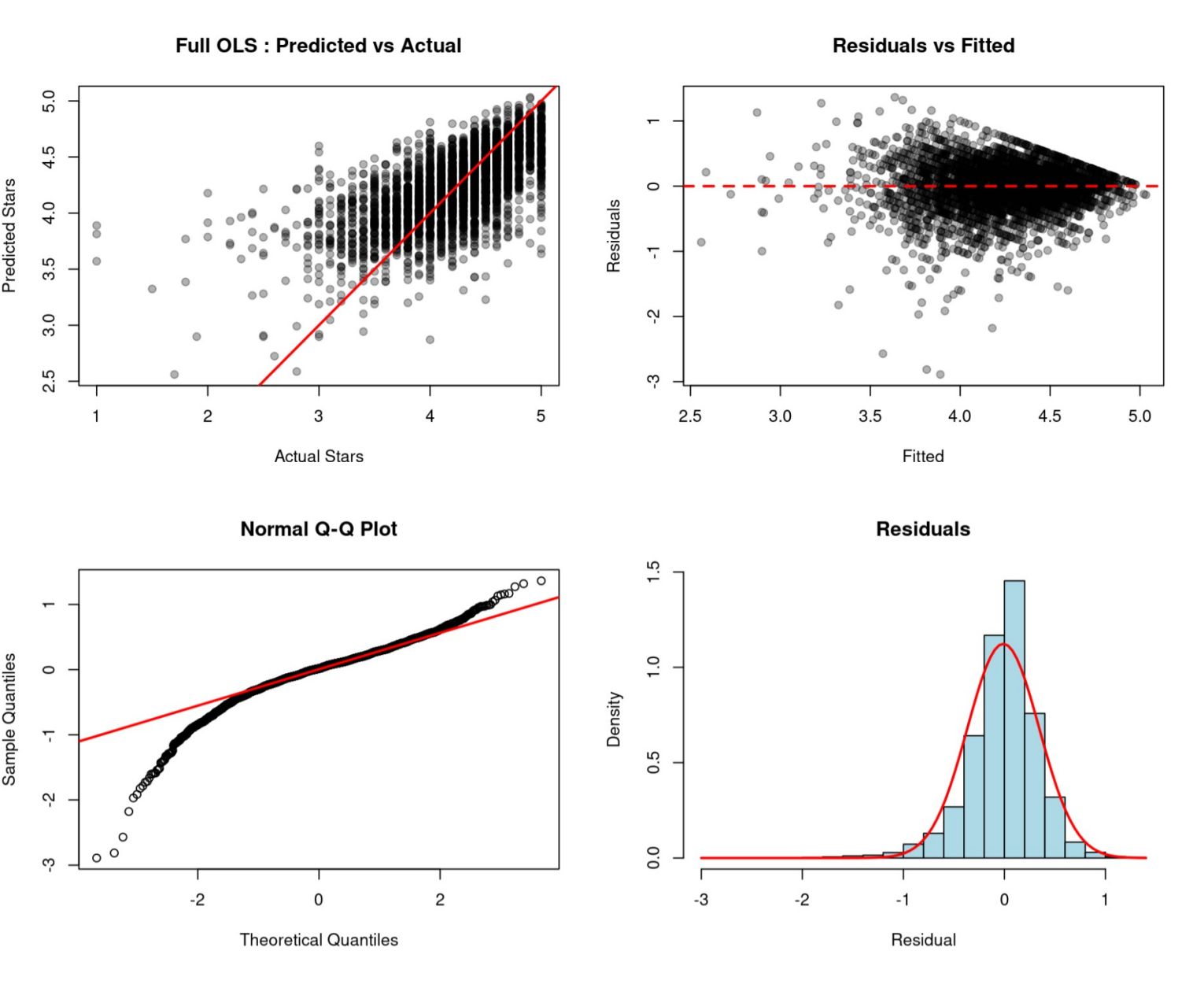
```
## Best by Test RMSE: Full OLS with RMSE = 0.3553
```

6.1 Diagnostics (Full OLS)

Hide

```
create_diagnostic_plots <- function(actual, predicted, model_name) {
  residuals <- actual - predicted
  par(mfrow = c(2,2))
  plot(actual, predicted, main = paste(model_name,": Predicted vs Actual"),
       xlab = "Actual Stars", ylab = "Predicted Stars", pch = 19, col = rgb(0,0,0,.3)); abline(0,1,col="red",lwd=
2)
  plot(predicted, residuals, main = "Residuals vs Fitted", xlab = "Fitted", ylab = "Residuals",
       pch = 19, col = rgb(0,0,0,.3)); abline(h=0,col="red",lwd=2,lty=2)
  qqnorm(residuals, main = "Normal Q-Q Plot"); qqline(residuals, col = "red", lwd = 2)
  hist(residuals, breaks = 30, main = "Residuals", xlab = "Residual", col = "lightblue", probability = TRUE)
  curve(dnorm(x, mean = mean(residuals), sd = sd(residuals)), add = TRUE, col = "red", lwd = 2)
  par(mfrow = c(1,1))
}

create_diagnostic_plots(test_model$stars, predict(full_model, test_model), "Full OLS")
```



7. Brief Findings & Limitations (in plain language)

- The author-level `avg_stars` variable seems to explain most of the variation. That fits the idea that readers follow authors they already like.
- Book-level variables play a smaller role here, and discounting doesn't seem to move the rating itself (which is about perceived quality).
- Ratings are pretty tightly bunched around 4.0–4.6, so there is a natural ceiling that makes low RMSE hard to achieve.
- I did notice mild funneling in residuals. If I were taking this further, I would report robust standard errors for the regression.
- The analysis is aimed at authors with at least three books in the data; predicting ratings for brand-new authors would likely need a different setup.

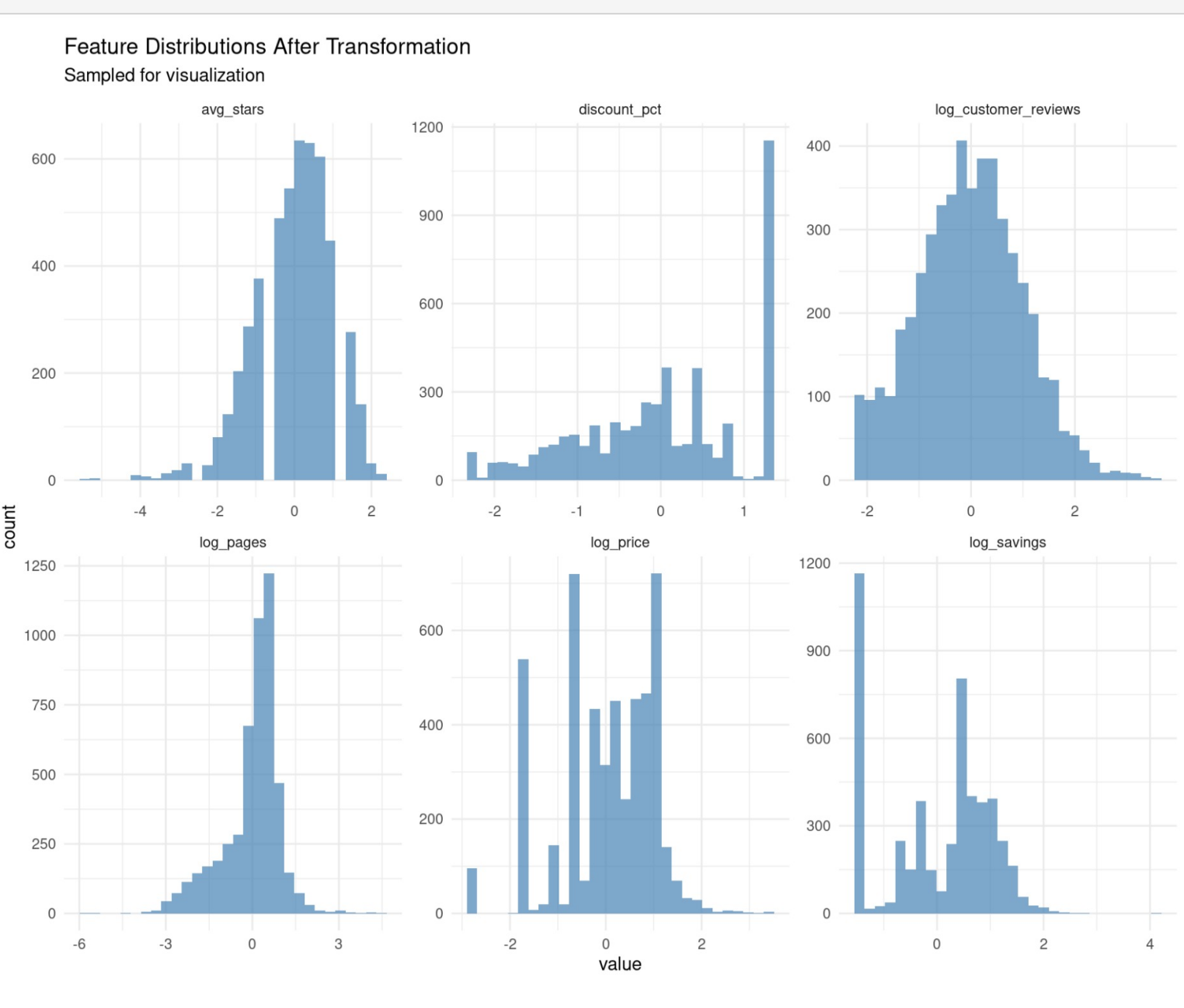
Appendix A — Supplemental EDA (kept minimal for rubric compliance)

These visuals justify transformations and give a quick sense of feature–target relationships.

A1. Post-Transformation Distributions

Hide

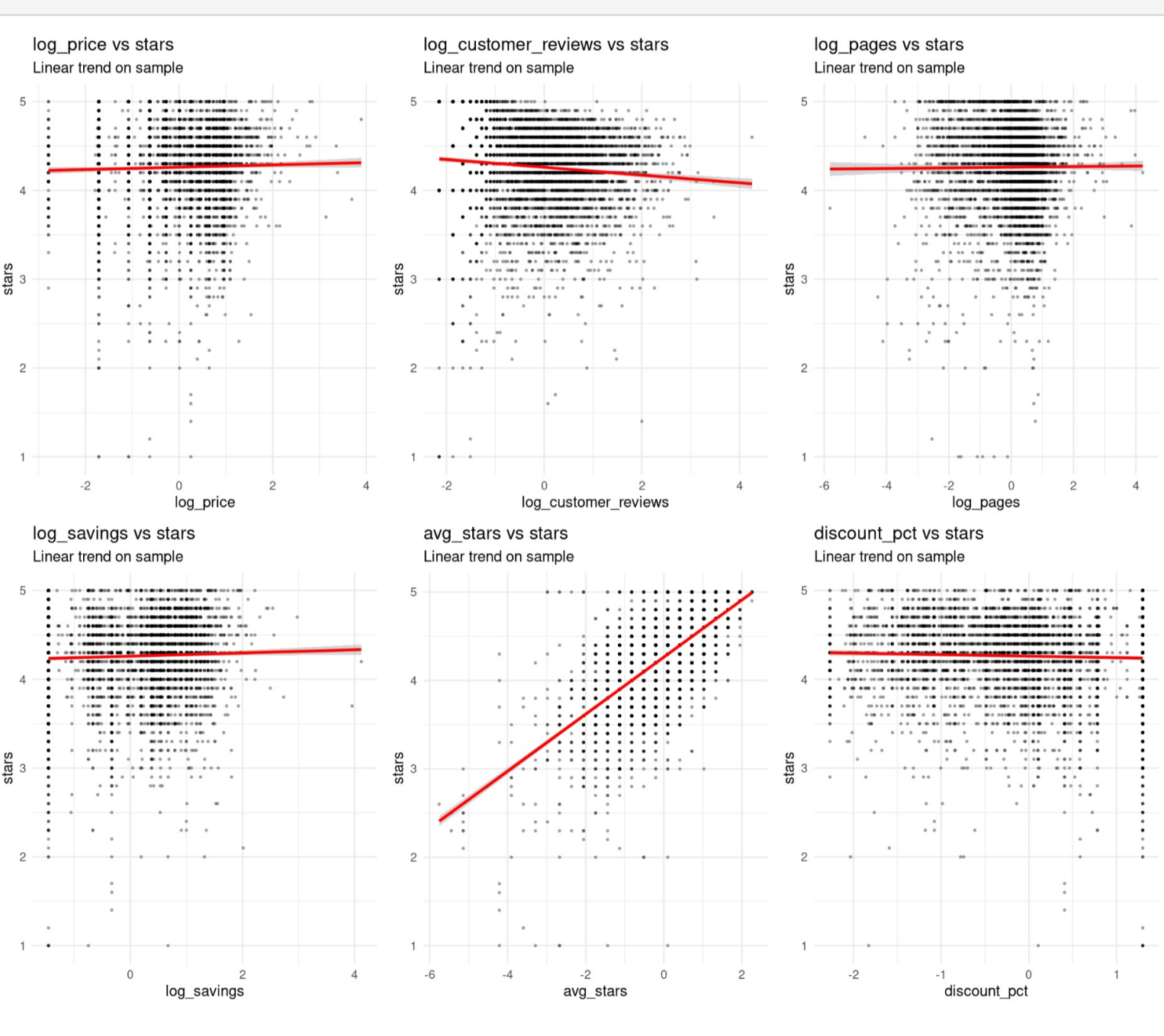
```
transformed_features <- train_model |> select(starts_with("log_"), avg_stars, discount_pct)
n_total <- nrow(transformed_features)
plot_sample <- transformed_features |> slice_sample(n = min(5000, n_total))
trans_long <- plot_sample |> pivot_longer(everything(), names_to="feature", values_to="value")
p_dist <- ggplot(trans_long, aes(x=value)) + geom_histogram(bins=30, fill="steelblue", alpha=.7) +
  facet_wrap(~feature, scales="free") + theme_minimal() +
  labs(title = "Feature Distributions After Transformation", subtitle = "Sampled for visualization")
save_plot(p_dist, "feature_distributions.png", width = 10, height = 8)
```



A2. Feature–Target Scatter with Linear Trend (sampled)

Hide

```
n_train <- nrow(train_model)
train_sample <- train_model |> slice_sample(n = min(5000, n_train))
feats <- c("log_price","log_customer_reviews","log_pages","log_savings","avg_stars","discount_pct")
make_scatter <- function(df, x, y="stars") {
  ggplot(df, aes_string(x=x, y=y)) + geom_point(alpha=.3, size=.5) +
    geom_smooth(method="lm", se=TRUE, color="red", size=1) +
    theme_minimal() + labs(title=paste(x,"vs",y), subtitle="Linear trend on sample")
}
plots <- lapply(feats, function(f) make_scatter(train_sample, f))
do.call(grid.arrange, c(plots, ncol=3))
```



8. Save Key Outputs

Hide

```
write_csv(vif_simple, file.path(tables_dir, "vif_simple.csv"))
write_csv(vif_full,   file.path(tables_dir, "vif_full.csv"))
write_csv(model_results, file.path(tables_dir, "model_comparison.csv"))
writeLines(capture.output(sessionInfo()), file.path(output_dir, "session_info.txt"))
cat("All results saved to:", output_dir, "\n")
```

```
## All results saved to: predictive_analysis_output
```