STA 141c Project

Yikai Lu

2025-05-29

lasso and ridge regression

```
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────────────── tidyverse 2.0.0 ──
## ✔ dplyr     1.1.4     ✔ readr     2.1.5
## ✔ forcats   1.0.0     ✔ stringr   1.5.1
## ✔ ggplot2   3.5.2     ✔ tibble    3.2.1
## ✔ lubridate 1.9.3     ✔ tidyr     1.3.1
## ✔ purrr     1.0.2
## ── Conflicts ────────────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(glmnet)
```

```
## 载入需要的程序包：Matrix
##
## 载入程序包：'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
##
## Loaded glmnet 4.1-8
```

```
train <- read_csv("D:/Yikai university work/spring quarter 2025/STA 104/standardized_data.csv")
```

```
## Rows: 1460 Columns: 85
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (42): MSZoning, Street, Alley, LotShape, LandContour, Utilities, LotConf...
## dbl (43): Id, MSSubClass, LotFrontage, LotArea, OverallQual, OverallCond, Ye...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
y <- train$SalePrice
X <- train %>% select(-Id, -SalePrice)
cat("Summarizing missing values...\n")
```

```
## Summarizing missing values...
```

```
missing_counts <- colSums(is.na(X))
missing_pct <- (missing_counts / nrow(X)) * 100
missing_df <- data.frame(
  Variable = names(missing_counts),
  MissingCount = missing_counts,
  MissingPct = round(missing_pct, 2)
)
missing_df <- missing_df[missing_df$MissingCount > 0, ]
missing_df <- missing_df[order(-missing_df$MissingPct), ]
print(missing_df)
```

```
##                  Variable MissingCount MissingPct
## Alley               Alley         1369      93.77
## Fence               Fence         1179      80.75
## MasVnrType     MasVnrType          872      59.73
## FireplaceQu   FireplaceQu          690      47.26
## LotFrontage   LotFrontage          259      17.74
## GarageType     GarageType           81       5.55
## GarageYrBlt   GarageYrBlt           81       5.55
## GarageFinish GarageFinish           81       5.55
## GarageQual     GarageQual           81       5.55
## GarageCond     GarageCond           81       5.55
## BsmtExposure BsmtExposure           38       2.60
## BsmtFinType2 BsmtFinType2           38       2.60
## BsmtQual         BsmtQual           37       2.53
## BsmtCond         BsmtCond           37       2.53
## BsmtFinType1 BsmtFinType1           37       2.53
## MasVnrArea     MasVnrArea            8       0.55
## Electrical     Electrical            1       0.07
```

```
# Optional: Drop columns with >90% missing (customizable)
drop_cols <- missing_df %>% filter(MissingPct > 90) %>% pull(Variable)
if (length(drop_cols) > 0) {
  cat("🗑 Dropping high-missing columns:\n")
  print(drop_cols)
  X <- X %>% select(-all_of(drop_cols))
}
```

```
## 🗑 Dropping high-missing columns:
## [1] "Alley"
```

```
# Numeric → median imputation
num_vars <- sapply(X, is.numeric)
X[num_vars] <- lapply(X[num_vars], function(col) {
  col[is.na(col)] <- median(col, na.rm = TRUE)
  col
})
# Categorical → mode imputation
cat_vars <- sapply(X, is.character)
X[cat_vars] <- lapply(X[cat_vars], function(col) {
  mode_val <- names(sort(table(col), decreasing = TRUE))[1]
  col[is.na(col)] <- mode_val
  col
})
X <- X[, sapply(X, function(col) length(unique(col)) > 1)]
X <- as.data.frame(X)
names(X) <- make.names(names(X), unique = TRUE)
# Safety check
if (length(names(X)) == 0) {
  stop("No valid predictors left after filtering.")
}

X_formula <- as.formula(paste("~", paste(names(X), collapse = "+")))
X_model <- model.matrix(X_formula, data = X)[, -1]
cat("Final model matrix created with", nrow(X_model), "rows and", ncol(X_model), "columns.\n")
```

```
## Final model matrix created with 1460 rows and 245 columns.
```

```
# Fit Lasso regression
set.seed(123)
lasso_cv <- cv.glmnet(X_model, y, alpha = 1)  # Lasso uses alpha = 1
# Best lambda
best_lambda_lasso <- lasso_cv$lambda.min
# Predict and calculate RMSE
lasso_preds <- predict(lasso_cv, s = best_lambda_lasso, newx = X_model)
lasso_rmse <- sqrt(mean((lasso_preds - y)^2))

cat("lasso Results:\n")
```

```
## lasso Results:
```

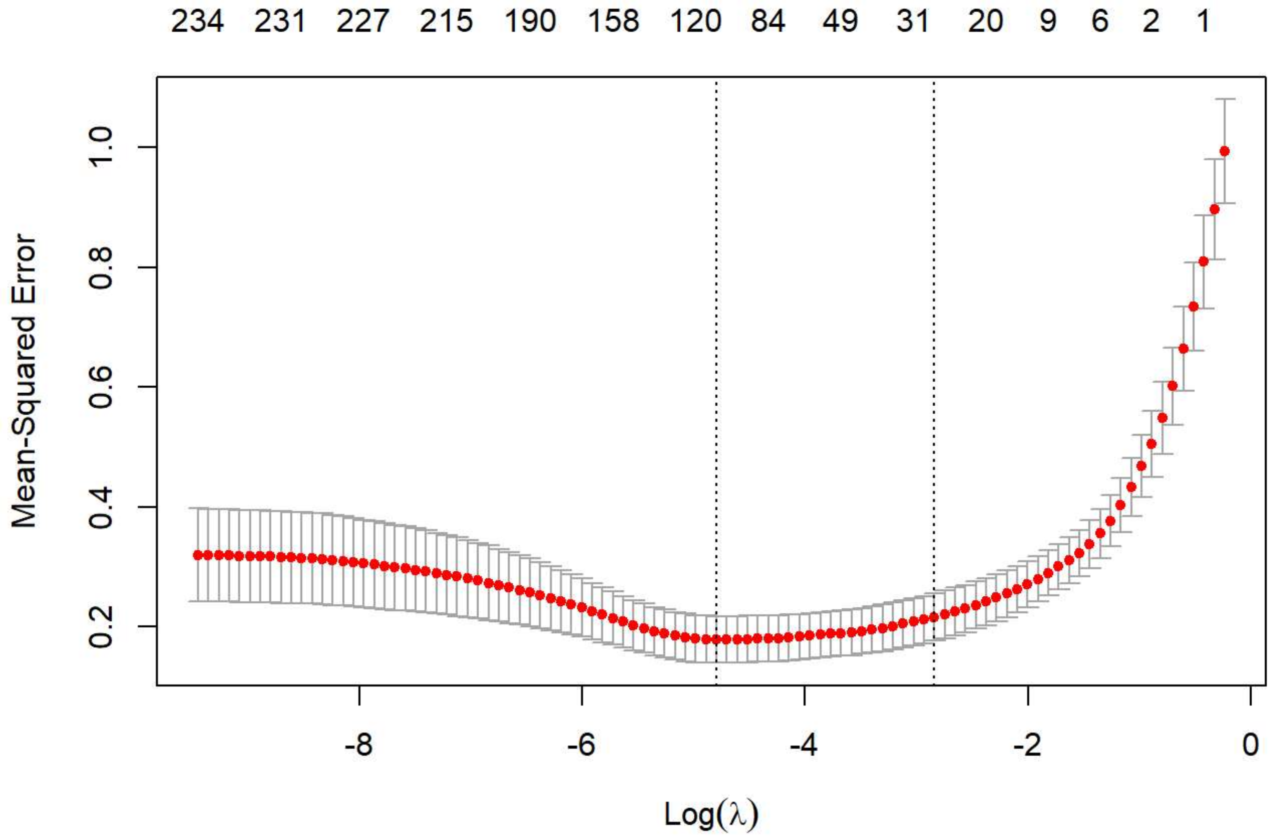
```
cat("Best lambda:", best_lambda_lasso, "\n")
```

```
## Best lambda: 0.00828361
```

```
cat("Lasso RMSE:", lasso_rmse, "\n")
```

```
## Lasso RMSE: 0.3166469
```

```
plot(lasso_cv)
```




```
# Fit Ridge regression-
set.seed(123)
ridge_cv <- cv.glmnet(X_model, y, alpha = 0)  # Ridge uses alpha = 0
# Best lambda
best_lambda_ridge <- ridge_cv$lambda.min
# Predict and calculate RMSE
ridge_preds <- predict(ridge_cv, s = best_lambda_ridge, newx = X_model)
ridge_rmse <- sqrt(mean((ridge_preds - y)^2))
cat("Ridge Results:\n")
```

```
## Ridge Results:
```

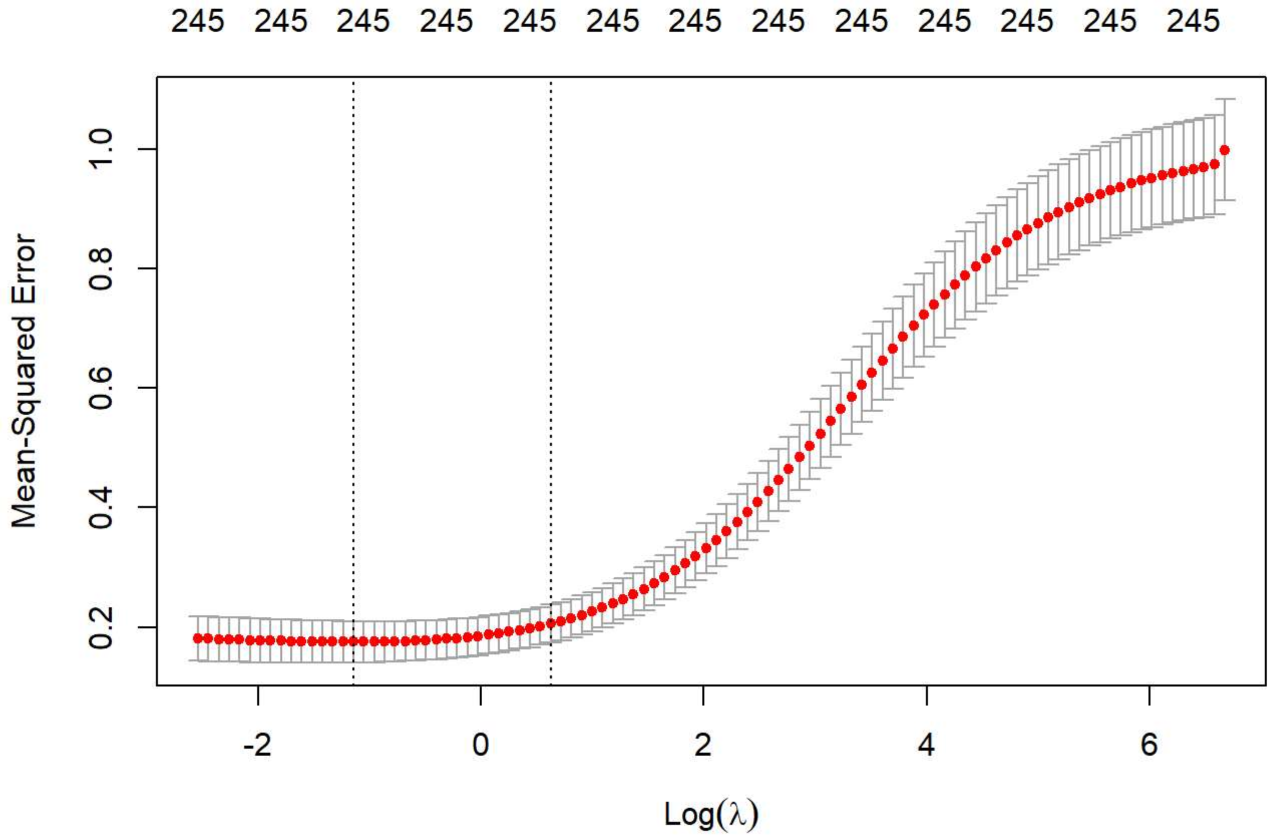
```
cat("Best lambda:", best_lambda_ridge, "\n")
```

```
## Best lambda: 0.3192113
```

```
cat("Ridge RMSE:", ridge_rmse, "\n")
```

```
## Ridge RMSE: 0.3212062
```

```
plot(ridge_cv)
```




```
#try to compare these two model
cat("lasso RMSE:", lasso_rmse, "\n")
```

```
## lasso RMSE: 0.3166469
```

```
cat(" Ridge RMSE:", ridge_rmse, "\n")
```

```
##  Ridge RMSE: 0.3212062
```

```
# RSS on training data
lasso_rss <- sum((lasso_preds - y)^2)
ridge_rss <- sum((ridge_preds - y)^2)
cat("Lasso Training RSS:", lasso_rss, "\n")
```

```
## Lasso Training RSS: 146.3873
```

```
cat("Ridge Training RSS:", ridge_rss, "\n")
```

```
## Ridge Training RSS: 150.6332
```

```
# Extract all Lasso coefficients
lasso_coef_full <- coef(lasso_cv, s = "lambda.min")
lasso_coef_df <- as.data.frame(as.matrix(lasso_coef_full))
colnames(lasso_coef_df) <- "Coefficient"
lasso_coef_df$Feature <- rownames(lasso_coef_df)

# Filter out  and select top 10 absolute coefficients
top10_lasso <- lasso_coef_df %>%
  filter(Feature != "(Intercept)") %>%
  mutate(abs_coef = abs(Coefficient)) %>%
  arrange(desc(abs_coef)) %>%
  slice(1:10)

cat(" Top 10 Lasso Features:\n")
```

```
##  Top 10 Lasso Features:
```

```
print(top10_lasso)
```

```
##                      Coefficient             Feature  abs_coef
## Condition2PosN       -1.8249514      Condition2PosN 1.8249514
## RoofMatlWdShngl       1.1670881     RoofMatlWdShngl 1.1670881
## NeighborhoodStoneBr   0.5822164 NeighborhoodStoneBr 0.5822164
## NeighborhoodNridgHt   0.4951974 NeighborhoodNridgHt 0.4951974
## NeighborhoodNoRidge   0.4898169 NeighborhoodNoRidge 0.4898169
## Condition2PosA        0.4623190      Condition2PosA 0.4623190
## UtilitiesNoSeWa      -0.3305870     UtilitiesNoSeWa 0.3305870
## GrLivArea             0.2754090           GrLivArea 0.2754090
## HeatingOthW          -0.2719468         HeatingOthW 0.2719468
## FunctionalSev        -0.2631412       FunctionalSev 0.2631412
```

```
# Use only top 10 features to refit the model and predict
top10_features <- top10_lasso$Feature
X_top10 <- X_model[, top10_features]

lasso_top10_model <- glmnet(X_top10, y, alpha = 1, lambda = best_lambda_lasso)
pred_top10 <- predict(lasso_top10_model, newx = X_top10)
rmse_top10 <- sqrt(mean((pred_top10 - y)^2))

cat("\n  RMSE using Top 10 Lasso Features:", rmse_top10, "\n")
```

```
##
##   RMSE using Top 10 Lasso Features: 0.596061
```