Model Selection and Regularization - R Version

Overview

In this section, we'll use the **Ames Housing** dataset to demonstrate model selection and regularization.

We'll cover:

1. Splitting data into training and testing sets
2. Performing **stepwise regression** with cross-validation
3. Running **Ridge** and **Lasso** regression, and visualizing how affects RMSE and model complexity
4. Exploring **Elastic Net**, varying to balance Ridge and Lasso
5. Comparing all models on the test set

Step 0: Setup and Data Preparation

We'll use a reduced set of variables for speed and clarity.

```
# Suppress messages and warnings globally
knitr::opts_chunk$set(echo = TRUE, message = FALSE, warning = FALSE)

library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.1     v stringr   1.5.2
v ggplot2   4.0.0     v tibble    3.3.0
v lubridate 1.9.4     v tidyr     1.3.1
v purrr     1.1.0
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becor
```

```r
library(caret)
```

```
Loading required package: lattice

Attaching package: 'caret'

The following object is masked from 'package:purrr':

    lift
```

```r
library(glmnet)
```

```
Loading required package: Matrix

Attaching package: 'Matrix'

The following objects are masked from 'package:tidyr':

    expand, pack, unpack

Loaded glmnet 4.1-10
```

```r
library(AmesHousing)
library(GGally)

set.seed(123)

# Load data and split into 70% training, 30% testing
ames <- make_ordinal_ames() %>% mutate(id = row_number())
```

```r
train <- ames %>% sample_frac(0.7)
test  <- anti_join(ames, train, by = "id")

# Select a manageable subset of predictors
keep <- c("Sale_Price", "Bedroom_AbvGr", "Year_Built", "Mo_Sold", "Lot_Area",
          "Street", "Central_Air", "First_Flr_SF", "Second_Flr_SF", "Full_Bath",
          "Half_Bath", "Fireplaces", "Garage_Area", "Gr_Liv_Area", "TotRms_AbvGrd")

train <- train %>% select(all_of(keep))
test  <- test  %>% select(all_of(keep))

# Convert categorical variables to factors
train <- train %>% mutate(across(c(Street, Central_Air), as.factor))
test  <- test %>% mutate(across(c(Street, Central_Air), as.factor))
```

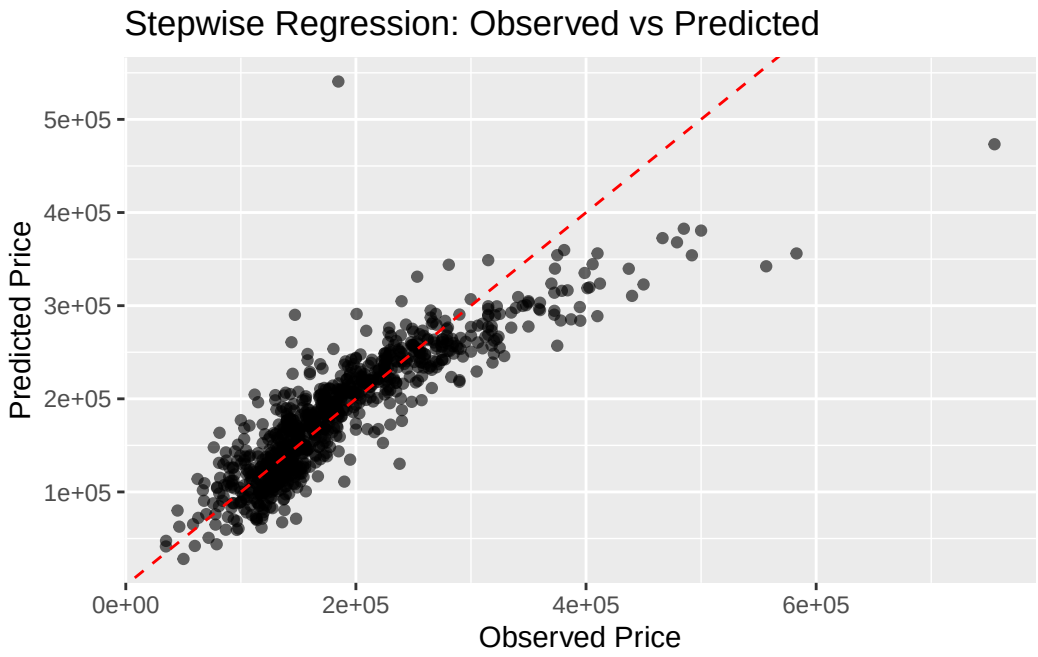
Step 1: Stepwise Regression with 10-Fold Cross Validation

We'll use backward stepwise regression as a traditional benchmark.

```r
ctrl <- trainControl(method = "cv", number = 10)

step_model <- train(
  Sale_Price ~ ., data = train,
  method = "lmStepAIC",
  trControl = ctrl,
  trace = FALSE,
  direction = "backward"
)

# Evaluate on the test set
step_pred <- predict(step_model, newdata = test)
step_perf <- postResample(step_pred, test$Sale_Price)
step_perf
```

```
        RMSE     Rsquared          MAE
3.820776e+04 7.670269e-01 2.613098e+04
```

```r
# Visualize predicted vs observed
ggplot(data.frame(obs = test$Sale_Price, pred = step_pred), aes(obs, pred)) +
  geom_point(alpha = 0.6) +
  geom_abline(linetype = "dashed", color = "red") +
  labs(title = "Stepwise Regression: Observed vs Predicted",
       x = "Observed Price", y = "Predicted Price")
```



Note:
Stepwise regression is intuitive and fast, but it can be unstable.

Stepwise regression (forward, backward, or both) selects predictors one at a time based on how much they improve a criterion (like AIC or adjusted R^2).

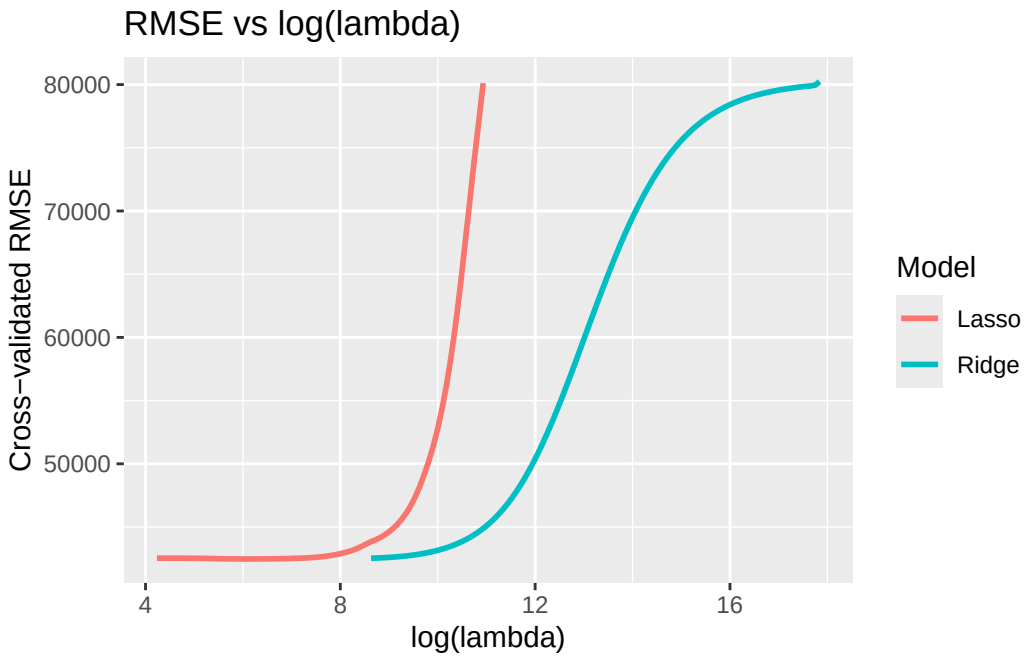
Step 2: Ridge and Lasso Regression

Ridge and Lasso both shrink coefficients, but in different ways: - Ridge shrinks all coefficients toward zero. - Lasso can set some coefficients *exactly* to zero, performing variable selection.

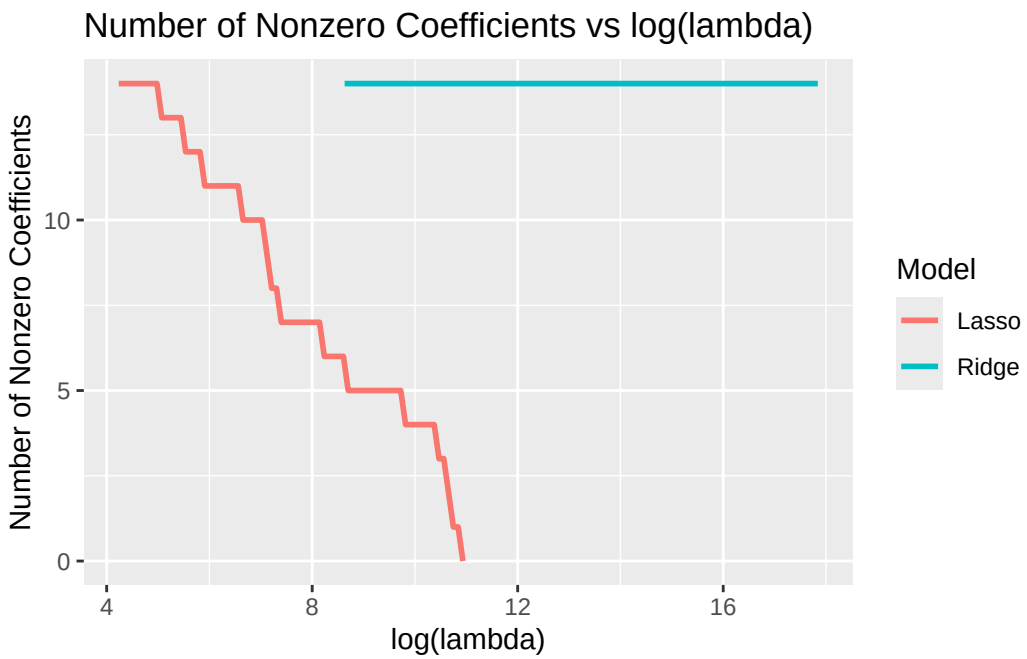
```r
x_train <- model.matrix(Sale_Price ~ ., train)[, -1]
y_train <- train$Sale_Price
x_test  <- model.matrix(Sale_Price ~ ., test)[, -1]
y_test  <- test$Sale_Price

set.seed(123)
cv_ridge <- cv.glmnet(x_train, y_train, alpha = 0, nfolds = 10)
cv_lasso <- cv.glmnet(x_train, y_train, alpha = 1, nfolds = 10)
```

Visualizing RMSE and Model Complexity

```r
build_path_df <- function(cvfit, label) {
  fit <- cvfit$glmnet.fit
  tibble(
    lambda = fit$lambda,
    log_lambda = log(fit$lambda),
    RMSE = sqrt(cvfit$cvm),
    nonzero = colSums(abs(fit$beta) > 0),
    Model = label
  )
}

ridge_df <- build_path_df(cv_ridge, "Ridge")
lasso_df <- build_path_df(cv_lasso, "Lasso")
df <- bind_rows(ridge_df, lasso_df)

ggplot(df, aes(log_lambda, RMSE, color = Model)) +
  geom_line(size = 1) +
  labs(title = "RMSE vs log(lambda)", y = "Cross-validated RMSE", x = "log(lambda)")
```




```
ggplot(df, aes(log_lambda, nonzero, color = Model)) +
  geom_line(size = 1) +
  labs(title = "Number of Nonzero Coefficients vs log(lambda)",
       y = "Number of Nonzero Coefficients", x = "log(lambda)")
```



Note:
- Increasing increases regularization.
- Ridge never eliminates variables, Lasso can.
- There's a sweet spot where RMSE is minimized.

Step 3: Elastic Net (Balancing Ridge and Lasso)

Elastic Net introduces to control the mix between Ridge (= 0) and Lasso (= 1).

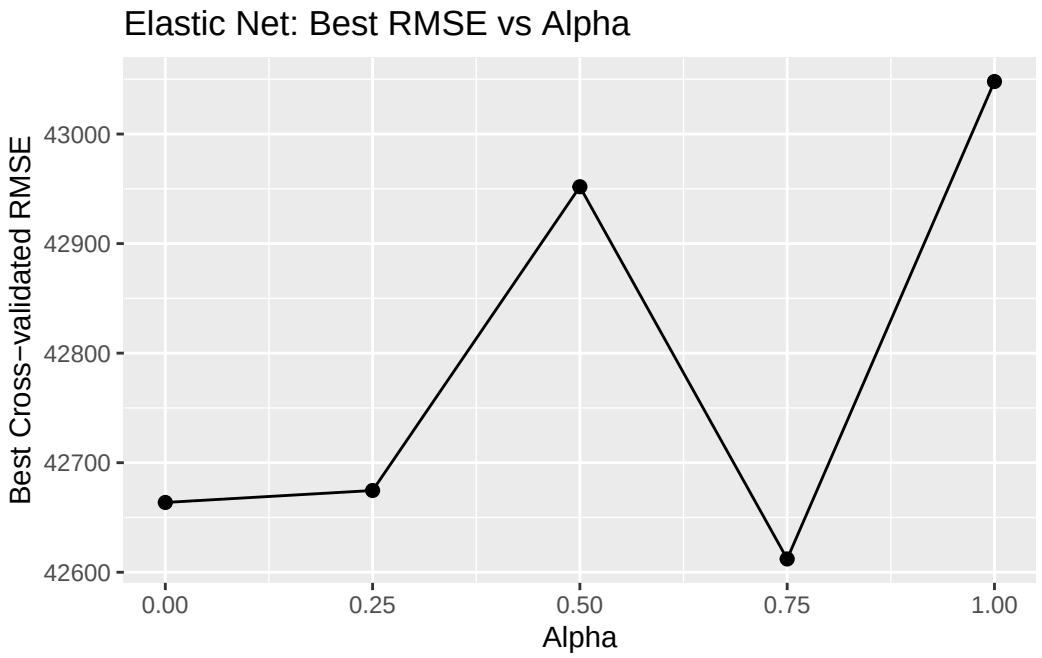
```
alpha_grid <- seq(0, 1, by = 0.25)

elastic_results <- map_df(alpha_grid, function(a) {
  cv_fit <- cv.glmnet(x_train, y_train, alpha = a, nfolds = 10)
  tibble(
    alpha = a,
    best_lambda = cv_fit$lambda.min,
    best_RMSE = sqrt(min(cv_fit$cvm)),
    nonzero = sum(abs(coef(cv_fit, s = "lambda.min")[-1]) > 0)
  )
})
```

```
elastic_results
```

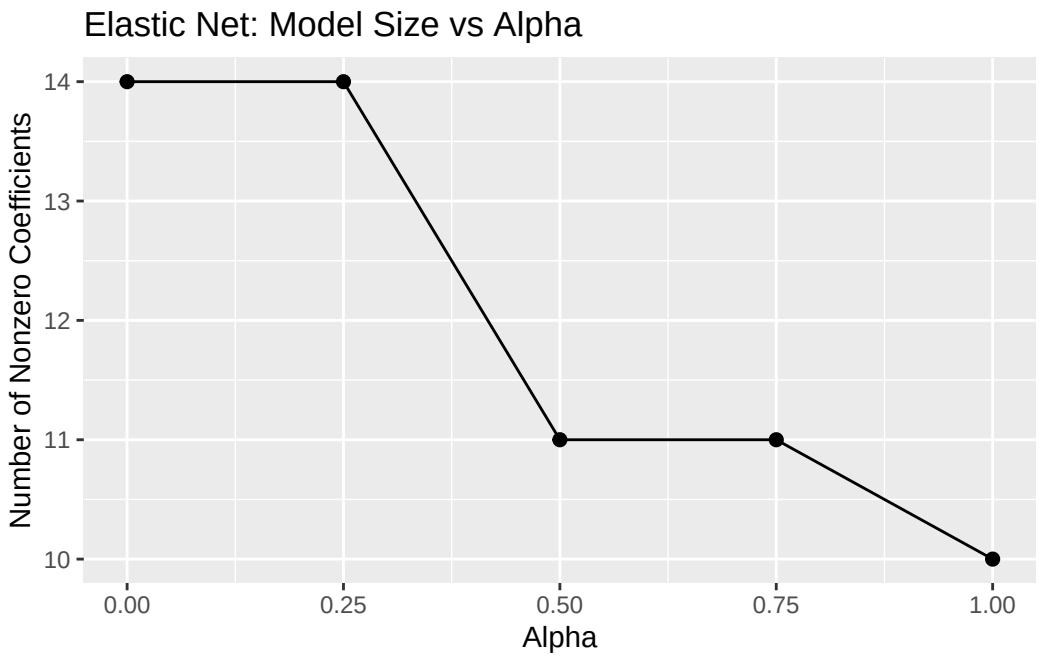
```
# A tibble: 5 x 4
  alpha best_lambda best_RMSE nonzero
  <dbl>       <dbl>     <dbl>   <int>
1  0          5603.    42664.      14
2  0.25       1224.    42675.      14
3  0.5        1552.    42952.      11
4  0.75        650.    42612.      11
5  1          1126.    43048.      10
```

Visualizing Effects

```
ggplot(elastic_results, aes(alpha, best_RMSE)) +
  geom_line() + geom_point(size = 2) +
  labs(title = "Elastic Net: Best RMSE vs Alpha",
       y = "Best Cross-validated RMSE", x = "Alpha")
```



```r
ggplot(elastic_results, aes(alpha, nonzero)) +
  geom_line() + geom_point(size = 2) +
  labs(title = "Elastic Net: Model Size vs Alpha",
       y = "Number of Nonzero Coefficients", x = "Alpha")
```



Note:
Elastic Net combines the strengths of Ridge and Lasso: - Ridge for stability. - Lasso for sparsity. - Often performs best at intermediate values.

Step 4: Comparing All Models on the Test Set

We'll compare Stepwise, Ridge, Lasso, and the best Elastic Net.

```r
ridge_pred <- predict(cv_ridge, newx = x_test, s = "lambda.min")
lasso_pred <- predict(cv_lasso, newx = x_test, s = "lambda.min")

best_alpha <- elastic_results %>% arrange(best_RMSE) %>% slice(1) %>% pull(alpha)
cv_en_best <- cv.glmnet(x_train, y_train, alpha = best_alpha, nfolds = 10)
elastic_pred <- predict(cv_en_best, newx = x_test, s = "lambda.min")
```

```r
compare_tbl <- bind_rows(
  Stepwise = as_tibble_row(postResample(step_pred, test$Sale_Price)),
  Ridge = as_tibble_row(postResample(ridge_pred, y_test)),
  Lasso = as_tibble_row(postResample(lasso_pred, y_test)),
  Elastic = as_tibble_row(postResample(elastic_pred, y_test)),
  .id = "Model"
)

colnames(compare_tbl) <- c("Model", "RMSE", "Rsquared", "MAE")
compare_tbl <- compare_tbl %>% arrange(RMSE)
compare_tbl
```

```
# A tibble: 4 x 4
  Model      RMSE Rsquared    MAE
  <chr>     <dbl>    <dbl>  <dbl>
1 Stepwise 38208.    0.767 26131.
2 Lasso    38297.    0.767 26045.
3 Elastic  38301.    0.767 26039.
4 Ridge    38412.    0.766 26022.
```

Key Takeaways

- Stepwise selection is simple but can lead to unstable models.
- Ridge adds stability via coefficient shrinkage.
- Lasso enforces sparsity by removing weak predictors.
- Elastic Net balances both effects.
- Regularization often produces models that generalize better than purely stepwise ones.