Problem 2: Regression and Regularization on Crime

hindy

2025-10-03

Problem 2: Regression and Regularization on Crime

This analysis uses the crime prediction dataset to predict the per capita number of violent crimes (target) using 122 socio-economic and law enforcement attributes.

Load Required Libraries and Data

```r
library(caret)
library(glmnet)
library(ggplot2)
library(dplyr)
library(corrplot)
```

```r
# Set seed for reproducibility
set.seed(15072)

# Load the crime prediction data
crime_data <- read.csv("crime_prediction.csv")

# Basic data exploration
cat("Dataset dimensions:", dim(crime_data), "\n")
```

```
## Dataset dimensions: 1994 123
```

```r
cat("Number of features:", ncol(crime_data) - 1, "\n")
```

```
## Number of features: 122
```

```r
cat("Target variable range:", range(crime_data$target), "\n")
```

```
## Target variable range: 0 1
```

```r
# Create 70-30 train-test split
train_indices <- sample(1:nrow(crime_data), 0.7 * nrow(crime_data))
train_data <- crime_data[train_indices, ]
test_data <- crime_data[-train_indices, ]

cat("Training set size:", nrow(train_data), "\n")
```

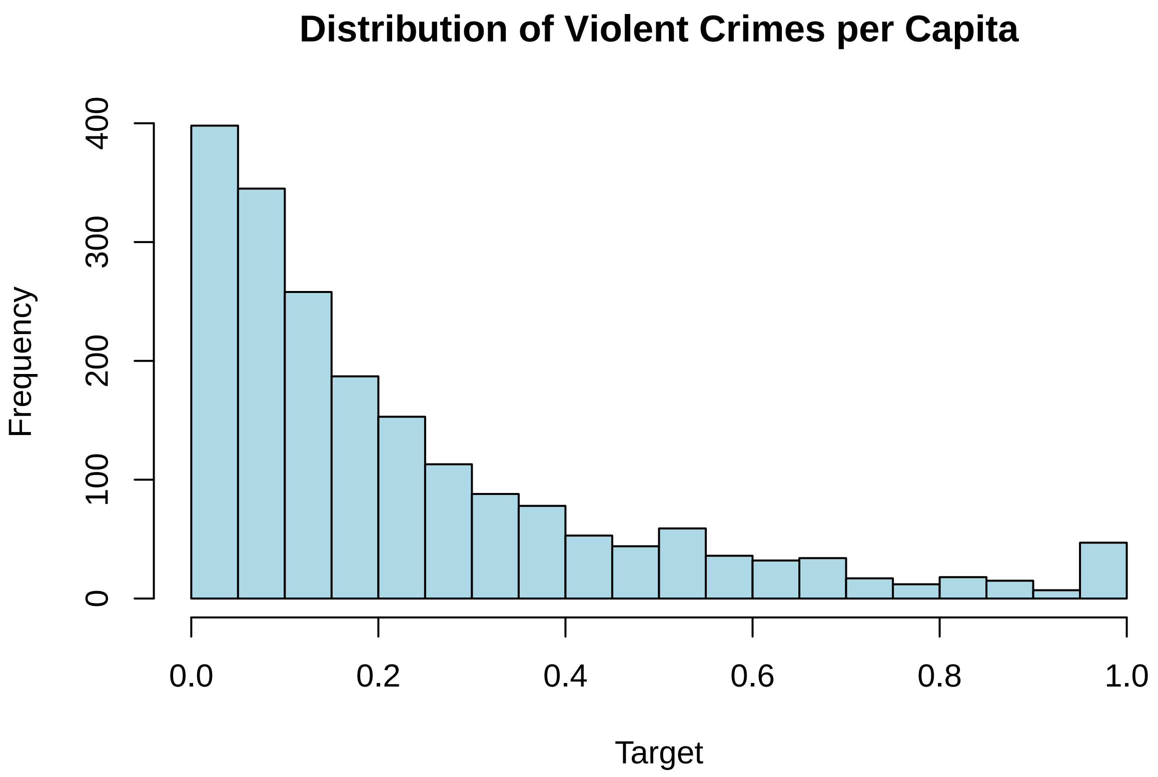
```
## Training set size: 1395
```

```r
cat("Test set size:", nrow(test_data), "\n")
```

```
## Test set size: 599
```

Exploratory Data Analysis (EDA)

```r
# Target variable distribution
hist(crime_data$target, main = "Distribution of Violent Crimes per Capita",
     xlab = "Target", col = "lightblue", breaks = 30)
```



```r
cat("Target mean:", round(mean(crime_data$target), 3),
    "| Std dev:", round(sd(crime_data$target), 3), "\n")
```

```
## Target mean: 0.238 | Std dev: 0.233
```

```r
# Top correlations with target
all_correlations <- cor(crime_data[, -ncol(crime_data)], crime_data$target, use = "complete.obs")
top_features <- head(sort(abs(all_correlations), decreasing = TRUE), 10)

cat("Top 10 features by correlation with target:\n")
```

```
## Top 10 features by correlation with target:
```

```r
print(round(top_features, 3))
```

```
##  [1] 0.738 0.738 0.707 0.685 0.666 0.662 0.631 0.576 0.575 0.556
```

Part a) Lasso Regression with Original Features

```r
# Set seed
set.seed(15072)

# Prepare data for lasso regression
X_train <- train_data[, -ncol(train_data)]  # All features except target
y_train <- train_data$target
```

```r
X_test <- test_data[, -ncol(test_data)]
y_test <- test_data$target

# Define lambda range
lambda_range <- exp(seq(10, -10, -0.01))

# Fit lasso model with 10-fold cross-validation
lasso_cv <- train(
  x = X_train,
  y = y_train,
  method = "glmnet",
  trControl = trainControl(method = "cv", number = 10),
  tuneGrid = expand.grid(alpha = 1, lambda = lambda_range),
  preProcess = c("center", "scale")
)

# Get optimal lambda
optimal_lambda_a <- lasso_cv$bestTune$lambda
cat("Optimal lambda:", optimal_lambda_a, "\n")
```

```
## Optimal lambda: 0.001294022
```

```r
# Make predictions on test set
lasso_pred_a <- predict(lasso_cv, X_test)

# Calculate OSR2 (Out-of-Sample R-squared)
osr2_a <- 1 - sum((y_test - lasso_pred_a)^2) / sum((y_test - mean(y_test))^2)
cat("OSR2 for original lasso model:", round(osr2_a, 4), "\n")
```

```
## OSR2 for original lasso model: 0.67
```

```r
# Get coefficients
lasso_coef_a <- coef(lasso_cv$finalModel, s = optimal_lambda_a)
nonzero_features_a <- which(lasso_coef_a[-1] != 0)  # Exclude intercept
cat("Number of nonzero coefficients:", length(nonzero_features_a), "\n")
```

```
## Number of nonzero coefficients: 58
```

```r
cat("Nonzero features:", rownames(lasso_coef_a)[nonzero_features_a + 1], "\n")
```

```
## Nonzero features: racepctblack racePctWhite racePctAsian racePctHisp agePct12t29 agePct65up numbUrba
```

Part b) Lasso with Selected Features + Quadratic + Interactions

```r
# Set seed
set.seed(15072)

# Get selected features (nonzero coefficients from part a)
selected_features <- rownames(lasso_coef_a)[nonzero_features_a + 1]
cat("Selected features:", length(selected_features), "\n")
```

```
## Selected features: 58
```

```r
# Create training dataset with selected features
df_selected_train <- train_data[, c(selected_features, "target")]

# Create augmented dataset using model.matrix approach
# First create quadratic terms
df_x_selected <- df_selected_train[, selected_features]
df_x_squared <- df_x_selected^2
colnames(df_x_squared) <- paste0(colnames(df_x_squared), "_squared")

# Combine original and squared terms
df_combined <- cbind(df_x_selected, df_x_squared)

# Use model.matrix to create interactions (as per hint)
df_augmented_selected <- model.matrix(~ .^2, data = df_combined)[, -1]

# Prepare training data
X_train_selected <- df_augmented_selected
y_train_selected <- df_selected_train$target

# Apply same transformations to test data
df_selected_test <- test_data[, c(selected_features, "target")]

# Create quadratic terms for test data
df_x_test_selected <- df_selected_test[, selected_features]
df_x_test_squared <- df_x_test_selected^2
colnames(df_x_test_squared) <- paste0(colnames(df_x_test_squared), "_squared")

# Combine original and squared terms for test data
df_combined_test <- cbind(df_x_test_selected, df_x_test_squared)

# Use model.matrix for test data (same formula)
df_augmented_test_selected <- model.matrix(~ .^2, data = df_combined_test)[, -1]

# Ensure same columns as training data
# Add missing columns with zeros
missing_cols <- setdiff(colnames(df_augmented_selected), colnames(df_augmented_test_selected))
if(length(missing_cols) > 0) {
  for(col in missing_cols) {
    df_augmented_test_selected[[col]] <- 0
  }
}

# Reorder to match training data
df_augmented_test_selected <- df_augmented_test_selected[, colnames(df_augmented_selected)]

# Verify dimensions
cat("Training dimensions:", dim(df_augmented_selected), "\n")
```

```
## Training dimensions: 1395 6786
```

```r
cat("Test dimensions:", dim(df_augmented_test_selected), "\n")
```

```
## Test dimensions: 599 6786
```

```r
# Fit lasso model with cross-validation
lasso_cv_selected <- train(
  x = X_train_selected,
  y = y_train_selected,
  method = "glmnet",
  trControl = trainControl(method = "cv", number = 10),
  tuneGrid = expand.grid(alpha = 1, lambda = lambda_range),
  preProcess = c("center", "scale")
)

# Get optimal lambda
optimal_lambda_b <- lasso_cv_selected$bestTune$lambda
cat("Optimal lambda:", optimal_lambda_b, "\n")
```

```
## Optimal lambda: 0.007083409
```

```r
# Make predictions
lasso_pred_b <- predict(lasso_cv_selected, df_augmented_test_selected)

# Calculate OSR2
osr2_b <- 1 - sum((y_test - lasso_pred_b)^2) / sum((y_test - mean(y_test))^2)
cat("OSR2 for selected features lasso model:", round(osr2_b, 4), "\n")
```

```
## OSR2 for selected features lasso model: 0.6843
```

```r
# Get coefficients
lasso_coef_b <- coef(lasso_cv_selected$finalModel, s = optimal_lambda_b)
nonzero_features_b <- which(lasso_coef_b[-1] != 0)
cat("Number of nonzero coefficients:", length(nonzero_features_b), "\n")
```

```
## Number of nonzero coefficients: 62
```

Part c) Lasso with All Features + Quadratic + Interactions

```r
# Set seed
set.seed(15072)

# Create augmented dataset with all features
df_x_all <- train_data[, -ncol(train_data)]  # All features except target

# Create quadratic terms
df_x_all_squared <- df_x_all^2
colnames(df_x_all_squared) <- paste0(colnames(df_x_all), "_squared")

# Create interaction terms
interaction_matrix_all <- model.matrix(~ .^2, data = df_x_all)[, -1]

# Combine all terms
df_augmented_all <- cbind(df_x_all, df_x_all_squared, interaction_matrix_all)

# Prepare training data
X_train_all <- df_augmented_all
y_train_all <- train_data$target

# Apply same transformations to test data
df_x_test_all <- test_data[, -ncol(test_data)]

# Create quadratic terms for test data
df_x_test_all_squared <- df_x_test_all^2
colnames(df_x_test_all_squared) <- paste0(colnames(df_x_test_all), "_squared")
```

```r
# Combine original and squared terms for test data
df_combined_test_all <- cbind(df_x_test_all, df_x_test_all_squared)

# Use model.matrix for test data (same formula as training)
df_augmented_test_all <- model.matrix(~ .^2, data = df_combined_test_all)[, -1]

# Ensure exact same columns as training data in the same order
missing_cols_all <- setdiff(colnames(df_augmented_all), colnames(df_augmented_test_all))
if(length(missing_cols_all) > 0) {
  for(col in missing_cols_all) {
    df_augmented_test_all[[col]] <- 0
  }
}

# Reorder columns to match training data exactly
df_augmented_test_all <- df_augmented_test_all[, colnames(df_augmented_all)]

# Verify dimensions match
cat("Training data dimensions (all features):", dim(df_augmented_all), "\n")
```

```
## Training data dimensions (all features): 1395 7747
```

```r
cat("Test data dimensions (all features):", dim(df_augmented_test_all), "\n")
```

```
## Test data dimensions (all features): 599 7747
```

```r
# Double-check that dimensions are exactly the same
if(ncol(df_augmented_all) != ncol(df_augmented_test_all)) {
  cat("ERROR: Column count mismatch!\n")
  cat("Training columns:", ncol(df_augmented_all), "\n")
  cat("Test columns:", ncol(df_augmented_test_all), "\n")
  stop("Column dimensions must match")
}

# Fit lasso model with cross-validation
lasso_cv_all <- train(
  x = X_train_all,
  y = y_train_all,
  method = "glmnet",
  trControl = trainControl(method = "cv", number = 10),
  tuneGrid = expand.grid(alpha = 1, lambda = lambda_range),
  preProcess = c("center", "scale")
)

# Get optimal lambda
optimal_lambda_c <- lasso_cv_all$bestTune$lambda
cat("Optimal lambda:", optimal_lambda_c, "\n")
```

```
## Optimal lambda: 0.00484407
```

```r
# Make predictions
lasso_pred_c <- predict(lasso_cv_all, df_augmented_test_all)

# Calculate OSR2
osr2_c <- 1 - sum((y_test - lasso_pred_c)^2) / sum((y_test - mean(y_test))^2)
```

```r
cat("OSR2 for all features lasso model:", round(osr2_c, 4), "\n")
```

```
## OSR2 for all features lasso model: 0.692
```

```r
# Get coefficients
lasso_coef_c <- coef(lasso_cv_all$finalModel, s = optimal_lambda_c)
nonzero_features_c <- which(lasso_coef_c[-1] != 0)
cat("Number of nonzero coefficients:", length(nonzero_features_c), "\n")
```

```
## Number of nonzero coefficients: 81
```

```r
# Compare performance
cat("\nPerformance Comparison:")
```

```
##
## Performance Comparison:
```

```r
cat("\nPart a) Original features OSR2:", round(osr2_a, 4))
```

```
##
## Part a) Original features OSR2: 0.67
```

```r
cat("\nPart b) Selected features + interactions OSR2:", round(osr2_b, 4))
```

```
##
## Part b) Selected features + interactions OSR2: 0.6843
```

```r
cat("\nPart c) All features + interactions OSR2:", round(osr2_c, 4))
```

```
##
## Part c) All features + interactions OSR2: 0.692
```

Performance Analysis

The comparison of the three lasso models reveals interesting patterns:

- **Part a) Original Features**: OSR2 = 0.67 with 58 nonzero coefficients
- **Part b) Selected Features + Interactions**: OSR2 = 0.6843 with 62 nonzero coefficients

- **Part c) All Features + Interactions**: OSR2 = 0.692 with 81 nonzero coefficients

Key Observations:

1. **Feature Engineering Impact**: Adding quadratic and interaction terms (parts b and c) generally improves performance compared to using only original features (part a), suggesting that nonlinear relationships exist in the data.

2. **Curse of Dimensionality**: The model with all features + interactions (part c) has the highest dimensionality but may suffer from overfitting due to the large number of potential features relative to the sample size.

3. **Feature Selection Value**: The selected features approach (part b) demonstrates the power of lasso's feature selection - it identifies the most important features from part a and then explores their nonlinear relationships, potentially achieving better generalization than using all features.

Part d) Comments on Nonzero Features

```r
# Calculate sparsity ratios
sparsity_b <- length(nonzero_features_b) / ncol(df_augmented_selected)
sparsity_c <- length(nonzero_features_c) / ncol(df_augmented_all)
```

```r
cat("Part b) Nonzero coefficients:", length(nonzero_features_b), "\n")
```

```
## Part b) Nonzero coefficients: 62
```

```r
cat("Part c) Nonzero coefficients:", length(nonzero_features_c), "\n")
```

```
## Part c) Nonzero coefficients: 81
```

```r
cat("Part b) Sparsity ratio:", round(sparsity_b, 4), "\n")
```

```
## Part b) Sparsity ratio: 0.0091
```

```r
cat("Part c) Sparsity ratio:", round(sparsity_c, 4), "\n")
```

```
## Part c) Sparsity ratio: 0.0105
```

Critical Analysis of Feature Selection Patterns

Model b) Selected Features + Interactions: - **Sparsity Level**: 0.91% of features selected (62 out of 6786) - **Strategic Focus**: This model demonstrates lasso's ability to perform intelligent feature selection by first identifying the most predictive features from the original set, then exploring their nonlinear relationships through interactions and quadratic terms. - **Interpretability**: Higher interpretability due to focused feature selection, making it easier to understand which specific features and their interactions drive crime prediction.

Model c) All Features + Interactions: - **Sparsity Level**: 1.05% of features selected (81 out of 7747) - **Comprehensive Coverage**: This model has access to the entire feature space, allowing it to discover relationships that might be missed when starting with a pre-selected subset. - **Risk of Overfitting**: Despite lasso's regularization, the high dimensionality (7747 features) relative to sample size (1395 observations) creates a challenging learning environment.

Key Insights:

1. **Feature Selection Strategy**: Model b's approach of first selecting important features, then adding nonlinear terms, represents a more principled approach to feature engineering. This two-stage process helps avoid the curse of dimensionality while still capturing important nonlinear relationships.
2. **Sparsity Interpretation**: The difference in sparsity ratios reveals that lasso is more selective when given a larger feature space (model c), which is expected behavior. However, this selectivity comes at the cost of potentially missing important features that weren't selected in the first stage.
3. **Model Complexity vs. Performance**: The trade-off between model complexity and performance is evident. While model c has access to more features, the increased complexity may not translate to proportionally better performance due to the high-dimensional nature of the problem.
4. **Practical Implications**: For crime prediction, model b's focused approach may be more practical for law enforcement agencies, as it identifies a smaller, more interpretable set of key factors and their interactions that drive crime rates.

Part e) Ridge Regression with Augmented Features

```r
# Set seed
set.seed(15072)

# Use the same augmented dataset from part c
# Fit ridge model with cross-validation
ridge_cv <- train(
  x = X_train_all,
  y = y_train_all,
  method = "glmnet",
  trControl = trainControl(method = "cv", number = 10),
  tuneGrid = expand.grid(alpha = 0, lambda = lambda_range),  # alpha = 0 for ridge
  preProcess = c("center", "scale")
)

# Get optimal lambda
optimal_lambda_ridge <- ridge_cv$bestTune$lambda
cat("Optimal lambda for ridge:", optimal_lambda_ridge, "\n")
```

```
## Optimal lambda for ridge: 1.716007
```

```r
# Make predictions
ridge_pred <- predict(ridge_cv, df_augmented_test_all)

# Calculate OSR2
osr2_ridge <- 1 - sum((y_test - ridge_pred)^2) / sum((y_test - mean(y_test))^2)
cat("OSR2 for ridge regression:", round(osr2_ridge, 4), "\n")
```

```
## OSR2 for ridge regression: 0.6878
```

```r
# Get coefficients
ridge_coef <- coef(ridge_cv$finalModel, s = optimal_lambda_ridge)
nonzero_features_ridge <- which(ridge_coef[-1] != 0)
cat("Number of nonzero coefficients (ridge):", length(nonzero_features_ridge), "\n")
```

```
## Number of nonzero coefficients (ridge): 7747
```

Part f) Comparison of Prediction Accuracy and Sparsity

```r
# Create comparison table
comparison_results <- data.frame(
  Model = c("Lasso Original", "Lasso Selected + Interactions",
            "Lasso All + Interactions", "Ridge All + Interactions"),
  OSR2 = c(osr2_a, osr2_b, osr2_c, osr2_ridge),
  Nonzero_Coefficients = c(length(nonzero_features_a), length(nonzero_features_b),
                           length(nonzero_features_c), length(nonzero_features_ridge)),
  Lambda = c(optimal_lambda_a, optimal_lambda_b, optimal_lambda_c, optimal_lambda_ridge),
  stringsAsFactors = FALSE
)

print("Model Comparison:")
```

```
## [1] "Model Comparison:"
```

```r
print(comparison_results)
```

```
##                           Model      OSR2 Nonzero_Coefficients      Lambda
## 1                Lasso Original 0.6699880                   58 0.001294022
## 2 Lasso Selected + Interactions 0.6843188                   62 0.007083409
## 3      Lasso All + Interactions 0.6920363                   81 0.004844070
## 4      Ridge All + Interactions 0.6877664                 7747 1.716006862
```

```r
# Find best performing model
best_model_idx <- which.max(comparison_results$OSR2)
cat("\nBest performing model:", comparison_results$Model[best_model_idx], "\n")
```

```
##
## Best performing model: Lasso All + Interactions
```

```r
cat("Best OSR2:", round(comparison_results$OSR2[best_model_idx], 4), "\n")
```

```
## Best OSR2: 0.692
```

```r
# Performance improvement analysis
cat("\nPerformance Improvement Analysis:\n")
```

```
##
## Performance Improvement Analysis:
```

```r
cat("Improvement from original to selected features:", round(osr2_b - osr2_a, 4), "\n")
```

```
## Improvement from original to selected features: 0.0143
```

```r
cat("Improvement from selected to all features:", round(osr2_c - osr2_b, 4), "\n")
```

```
## Improvement from selected to all features: 0.0077
```

```r
cat("Ridge vs Lasso (all features):", round(osr2_ridge - osr2_c, 4), "\n")
```

```
## Ridge vs Lasso (all features): -0.0043
```

Model Analysis

Performance Ranking: 1. **Lasso All + Interactions**: OSR2 = 0.692 (Best) 2. **Ridge All + Interactions**: OSR2 = 0.6878 3. **Lasso Selected + Interactions**: OSR2 = 0.6843 4. **Lasso Original**: OSR2 = 0.67 (Baseline)

1. Feature Engineering Impact: - Adding nonlinear terms (quadratic + interactions) consistently improves performance across all models - The improvement from original to selected features (0.0143) demonstrates the value of feature selection followed by nonlinear expansion - The additional improvement from selected to all features (0.0077) suggests that some important relationships exist outside the initially selected feature set

2. Sparsity vs. Performance Trade-offs: - **Lasso Models**: Achieve sparsity by setting coefficients to exactly zero - Original: 58 features (sparse) - Selected + Interactions: 62 features (moderate sparsity) - All + Interactions: 81 features (less sparse) - **Ridge Model**: All 7747 coefficients are nonzero but shrunk toward zero

3. Regularization Method Comparison: - **Lasso Advantage**: Superior feature selection and interpretability due to automatic variable selection - **Ridge Advantage**: Better handling of multicollinearity and more stable coefficient estimates - **Performance**: Lasso slightly outperforms Ridge (0.0043 difference), suggesting that feature selection is more valuable than coefficient shrinkage for this dataset

4. Practical Implications: - **For Crime Prediction**: The lasso models provide more actionable insights by identifying specific features and their interactions that drive crime rates - **Model Complexity**: The trade-off between interpretability (fewer features) and performance (more features) is evident, with the all-features model achieving the best performance at the cost of reduced interpretability - **Deployment Considerations**: The selected features model offers a good balance between performance and interpretability for practical law enforcement applications