CV.LASSO, RIDGE, and NAIVE LASSO

2024-05-19

General Analysis

```r
data_scale$cleaning_fee <- as.logical(data_scale$cleaning_fee)

x <- model.matrix(log_price ~ ., data_scale)[, -1]
y <- data_scale$log_price

# Set up cross-validation for LASSO and Ridge Regression
set.seed(123) # For reproducibility
cv_lasso <- cv.glmnet(x, y, alpha = 1, family = 'gaussian', standardize = TRUE)
cv_ridge <- cv.glmnet(x, y, alpha = 0, family = 'gaussian', standardize = TRUE)

# Best lambda for each model
best_lambda_lasso <- cv_lasso$lambda.min
best_lambda_ridge <- cv_ridge$lambda.min

# Train the final models using the best lambda
lasso_model <- glmnet(x, y, alpha = 1, lambda = best_lambda_lasso, family = 'gaussian',
                      standardize = TRUE)
ridge_model <- glmnet(x, y, alpha = 0, lambda = best_lambda_ridge, family = 'gaussian',
                      standardize = TRUE)

set.seed(123)
trainIndex <- createDataPartition(data_scale$log_price, p = 0.8, list = FALSE, times = 1)
loft_train <- data_scale[trainIndex, ]
loft_test <- data_scale[-trainIndex, ]

# Prepare training and test sets for prediction
x_train <- model.matrix(log_price ~ ., loft_train)[, -1]
y_train <- loft_train$log_price
x_test <- model.matrix(log_price ~ ., loft_test)[, -1]
y_test <- loft_test$log_price

lasso_pred <- predict(lasso_model, s = best_lambda_lasso, family = 'gaussian', newx = x_test)
ridge_pred <- predict(ridge_model, s = best_lambda_ridge, faimily = 'gaussian', newx = x_test)

lasso_mse <- mean((y_test - lasso_pred)^2)
lasso_rmse <- sqrt(lasso_mse)
lasso_mae <- mean(abs(y_test - lasso_pred))
lasso_r2 <- cor(y_test, lasso_pred)^2

ridge_mse <- mean((y_test - ridge_pred)^2)
ridge_rmse <- sqrt(ridge_mse)
ridge_mae <- mean(abs(y_test - ridge_pred))
```

```r
ridge_r2 <- cor(y_test, ridge_pred)^2

comparison <- data.frame(
  Model = c("LASSO", "Ridge"),
  RMSE = c(lasso_rmse, ridge_rmse),
  MSE = c(lasso_mse, ridge_mse),
  MAE = c(lasso_mae, ridge_mae),
  R_squared = c(lasso_r2, ridge_r2)
)
comparison
```

```r
set.seed(123)
train_index <- createDataPartition(data_scale$log_price, p = 0.8, list = FALSE)
train_data <- data_scale[train_index, ]
test_data <- data_scale[-train_index, ]

X <- as.matrix(train_data[, -which(names(train_data) == "log_price")])
Y <- train_data$log_price

cv.fit <- cv.glmnet(x_train, y_train, family = "gaussian", alpha = 1)

# Coefficients for the best lambda value
coefficients_best_lambda <- coef(cv.fit, s = "lambda.min")
head(coefficients_best_lambda, 20)
nonzero_coef_best_lambda_count <- sum(coefficients_best_lambda[-1] != 0)
print(nonzero_coef_best_lambda_count)

# Coefficients for the lambda value selected by 1 standard error rule
coefficients_1se <- coef(cv.fit, s = "lambda.1se")
head(coefficients_1se, 20)
nonzero_coef_1se_count <- sum(coefficients_1se[-1] != 0)
print(nonzero_coef_1se_count)
```

```r
#Top 10 coefficients for general analysis with LASSO
lasso_coefs <- coef(lasso_model, s = best_lambda_lasso)
lasso_coefs <- as.data.frame(as.matrix(lasso_coefs))
colnames(lasso_coefs) <- c("Coefficient")
lasso_coefs <- lasso_coefs %>%
  rownames_to_column(var = "Feature") %>%
  arrange(desc(abs(Coefficient)))
head(lasso_coefs, 10)
```

```r
# Extract coefficients for general analysis with Ridge
ridge_coefs <- coef(ridge_model, s = best_lambda_ridge)
ridge_coefs <- as.data.frame(as.matrix(ridge_coefs))
colnames(ridge_coefs) <- c("Coefficient")
ridge_coefs <- ridge_coefs %>%
  rownames_to_column(var = "Feature") %>%
  arrange(desc(abs(Coefficient)))
head(ridge_coefs, 10)
```

```
#In_sample_R2 vs OOS_R2 comparison
lasso_pred_train <- predict(lasso_model, s = best_lambda_lasso, family = 'gaussian', newx = x_train)
ridge_pred_train <- predict(ridge_model, s = best_lambda_ridge, family = 'gaussian', newx = x_train)

lasso_pred_test <- predict(lasso_model, s = best_lambda_lasso, newx = x_test)
ridge_pred_test <- predict(ridge_model, s = best_lambda_ridge, newx = x_test)

in_sample_r2 <- function(y_true, y_pred) {
  cor(y_true, y_pred)^2
}
out_of_sample_r2 <- function(y_true, y_pred) {
  1 - sum((y_true - y_pred)^2) / sum((y_true - mean(y_true))^2)
}

lasso_in_sample_r2 <- in_sample_r2(y_train, lasso_pred_train)
ridge_in_sample_r2 <- in_sample_r2(y_train, ridge_pred_train)

lasso_out_sample_r2 <- out_of_sample_r2(y_test, lasso_pred_test)
ridge_out_sample_r2 <- out_of_sample_r2(y_test, ridge_pred_test)

comparison <- data.frame(
  Model = c("LASSO", "Ridge"),
  In_sample_R2 = c(lasso_in_sample_r2, ridge_in_sample_r2),
  Out_sample_R2 = c(lasso_out_sample_r2, ridge_out_sample_r2)
)
comparison
```

Model with Treatment Variables

```
X_treatment <- model.matrix(log_price~.-1,train_data)
Y_treatment <- train_data$log_price

X_test <- model.matrix(log_price~.-1,test_data)
Y_test <- test_data$log_price

cv_fit_treatment <- cv.glmnet(X_treatment, Y_treatment, family = "gaussian", alpha = 1)
cv_fit_test <- cv.glmnet(X_test,Y_test, family = "gaussian", alpha = 1)

# Coefficients for the best lambda value
coefficients_best_lambda_treatment <- coef(cv_fit_treatment, s = "lambda.min")
head(coefficients_best_lambda_treatment, 20)

# Coefficients for the lambda value selected by 1 standard error rule
coefficients_1se_treatment <- coef(cv_fit_treatment, s = "lambda.1se")
head(coefficients_1se_treatment,20)

treatment_pred_train <- predict(cv_fit_treatment, s = "lambda.min", newx = X_treatment)
treatment_pred_test <- predict(cv_fit_treatment, s = "lambda.min", newx = X_test)

in_sample_r2_treatment <- in_sample_r2(Y_treatment, treatment_pred_train)
in_sample_r2_treatment
out_of_sample_r2_treatment <- out_of_sample_r2(Y_test, treatment_pred_test)
out_of_sample_r2_treatment
```

```r
comparison_treatment <- data.frame(
  Model = "Treatment Model",
  In_sample_R2 = in_sample_r2_treatment,
  Out_sample_R2 = out_of_sample_r2_treatment
)
comparison_treatment
```

NAIVE LASSO

```r
set.seed(123)

X_treatment <- model.matrix(log_price ~ . - 1, train_data)
Y_treatment <- train_data$log_price

X_test <- model.matrix(log_price ~ . - 1, test_data)
Y_test <- test_data$log_price

naive_lasso_fit <- glmnet(X_treatment, Y_treatment, family = "gaussian", alpha = 1)
lambda_values <- naive_lasso_fit$lambda
coefficients_best_lambda_treatment <- coef(naive_lasso_fit, s = min(lambda_values))
head(coefficients_best_lambda_treatment, 20)

# Coefficients for the lambda value selected by 1 standard error rule
lambda_value <- cv_fit_treatment$lambda.min
treatment_pred_lambda <- predict(cv_fit_treatment, s = lambda_value, newx = X_treatment)
coefficients_1se_treatment <- coef(naive_lasso_fit, s = lambda_value)
head(coefficients_1se_treatment, 20)

treatment_pred_train <- predict(naive_lasso_fit, s = min(lambda_values), newx = X_treatment)
treatment_pred_test <- predict(naive_lasso_fit, s = min(lambda_values), newx = X_test)

in_sample_r2_treatment <- in_sample_r2(Y_treatment, treatment_pred_train)
out_of_sample_r2_treatment <- out_of_sample_r2(Y_test, treatment_pred_test)

comparison_treatment <- data.frame(
  Model = "Naive LASSO Model",
  In_sample_R2 = in_sample_r2_treatment,
  Out_sample_R2 = out_of_sample_r2_treatment
)
comparison_treatment
```