Exercise 2

105.707 Advanced Methods for Regression and Classification, 2024W

11912007 - Yahya Jabary

27.10.2024

Contents

Task 1: Full Model

Analysis

The first task involves working with the full model using the `lm()` function. First we load the data - either by downloading it from the UCI repository or by using the provided `.RData` file. After splitting the data randomly into training (2/3) and test (1/3) sets, we check for missing values and ensure that the response and feature dimensions match. We then fit the linear regression model using all available features and analyze the model's performance through several steps.

(a) We plot fitted values versus response and RMSE for training data. We can see that the points generally follow the red diagonal line $y = x$, indicating reasonable fit. The training RMSE is 0.1910, indicating relatively good model fit on the training data.

(b, c) The model was the evaluated using 5-fold cross-validation with 100 replications in two ways: Using `rmspe` (Regular Root Mean Square Prediction Error) and `rtmspe` (Regular Trimmed Root Mean Square Prediction Error).

We can see the distribution of RMSE values from cross-validation ranges roughly from 0 to 250. There are some extreme values/outliers in the CV results. This wide range and presence of outliers suggests that the model's performance varies significantly across different folds of the data. This indicates the model might not be very stable in its predictions.

For the RTMSPE boxplot we observe a much tighter range of values, approximately between 0.15 and 0.19. This trimmed version of RMSE excludes extreme prediction errors. The much smaller range compared to regular RMSE suggests that the model's poor performance might be driven by a few outliers. The trimmed RMSE values are closer to the training RMSE we saw in part (a).

(d) Finally we look at the test data predictions and RMSE via the right plot. The test RMSE is 2.9346, which is substantially higher than the training RMSE. The scatter plot shows much more deviation from the ideal $y = x$ line. Many points deviate significantly from the diagonal, especially for higher predicted values. There appears to be some systematic bias in the predictions, with several extreme outliers.

Additionally the code warnings revealed high multicollinearity among economic variables (`Econ3`, `Econ7`, `Econ11`, etc.), which might explain some of the model's instability between training and test sets.

So while the model performs well on the training data (RMSE = 0.191), it shows signs of overfitting:

- Much higher test RMSE (2.9346)
- Unstable cross-validation results when using regular RMSE
- Better stability when using trimmed RMSE, suggesting outlier sensitivity
- Presence of multicollinearity among predictors

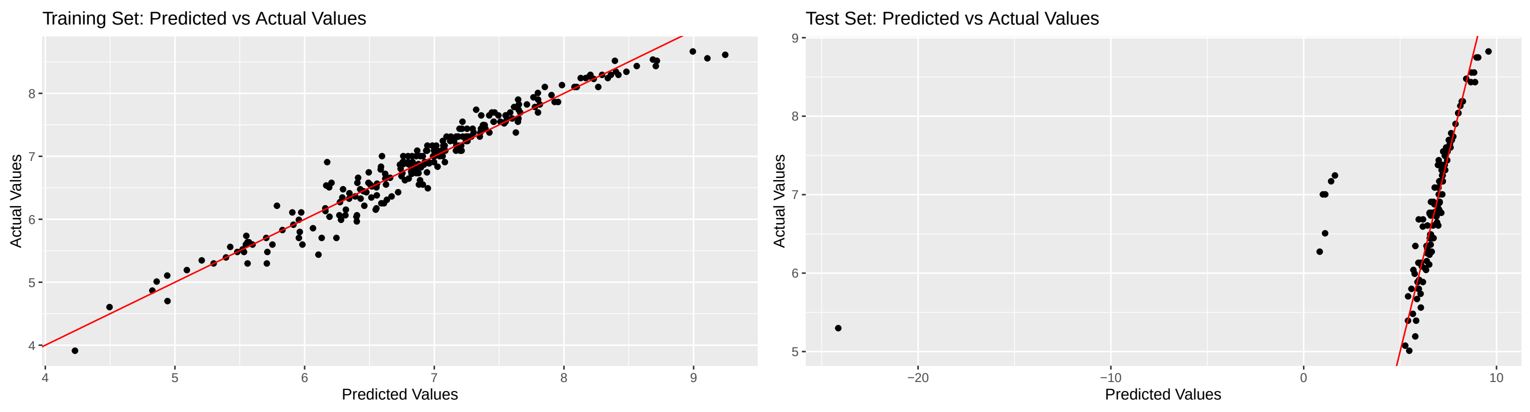
These results suggest that some form of regularization or variable selection might be beneficial, which we will address in the next section of the exercise.

Code

```r
# a) download
url <- "https://archive.ics.uci.edu/ml/machine-learning-databases/00437/Residential-Building-Data-Set.xlsx"
temp <- tempfile(fileext = ".xlsx")
download.file(url, temp, mode = "wb")
df <- read_excel(temp)
unlink(temp)

# b) use provided .RData file
load("building.RData")
```

```r
# random 2/3 train, 1/3 test split
split_and_validate_data <- function(df) {
    n <- nrow(df)
    n_train <- round(2/3 * n)
    n_test <- n - n_train
    train_indices <- sample(1:n, n_train)
    train_data <- df[train_indices, ]
    test_data <- df[-train_indices, ]
    y_train <- train_data$y
    X_train <- train_data[, setdiff(names(train_data), "y")]
    y_test <- test_data$y
    X_test <- test_data[, setdiff(names(test_data), "y")]

    assert_that((n_train + n_test == n) && nrow(train_data) == n_train && nrow(test_data) == n_test, msg="split sizes don't add up")
    assert_that(length(intersect(train_indices, which(!1:n %in% train_indices))) == 0, msg="train and test sets overlap")
    assert_that(all(names(X_train) == names(X_test)), msg="feature names don't match between train and test")
    assert_that(length(y_train) == nrow(X_train) && length(y_test) == nrow(X_test), msg="response and feature dimensions mismatch")
    assert_that(!any(is.na(X_train)) && !any(is.na(X_test)) && !any(is.na(y_train)) && !any(is.na(y_test)), msg="missing values found in data")
    return(list(
        X_train = X_train,
        y_train = y_train,
        X_test = X_test,
        y_test = y_test
    ))
}
split_data <- split_and_validate_data(df)
X_train <- split_data$X_train
y_train <- split_data$y_train # already log transformed
X_test <- split_data$X_test
y_test <- split_data$y_test
```

```r
# fit
lm_model <- lm(y ~ ., data = data.frame(y = y_train, X_train))

# eval
train_predictions <- predict(lm_model, newdata = X_train)
p1 <- ggplot(data.frame(predicted = train_predictions, actual = y_train), aes(x = predicted, y = actual)) +
    geom_point() + geom_abline(intercept = 0, slope = 1, color = "red") +
    labs(x = "Predicted Values", y = "Actual Values", title = "Training Set: Predicted vs Actual Values")
test_predictions <- predict(lm_model, newdata = X_test)
p2 <- ggplot(data.frame(predicted = test_predictions, actual = y_test), aes(x = predicted, y = actual)) +
    geom_point() + geom_abline(intercept = 0, slope = 1, color = "red") +
    labs(x = "Predicted Values", y = "Actual Values", title = "Test Set: Predicted vs Actual Values")
grid.arrange(p1, p2, ncol = 2)
```



```r
cat("training RMSE:", round(rmse(y_train, train_predictions), 4), "\n")
```

```
## training RMSE: 0.191
```

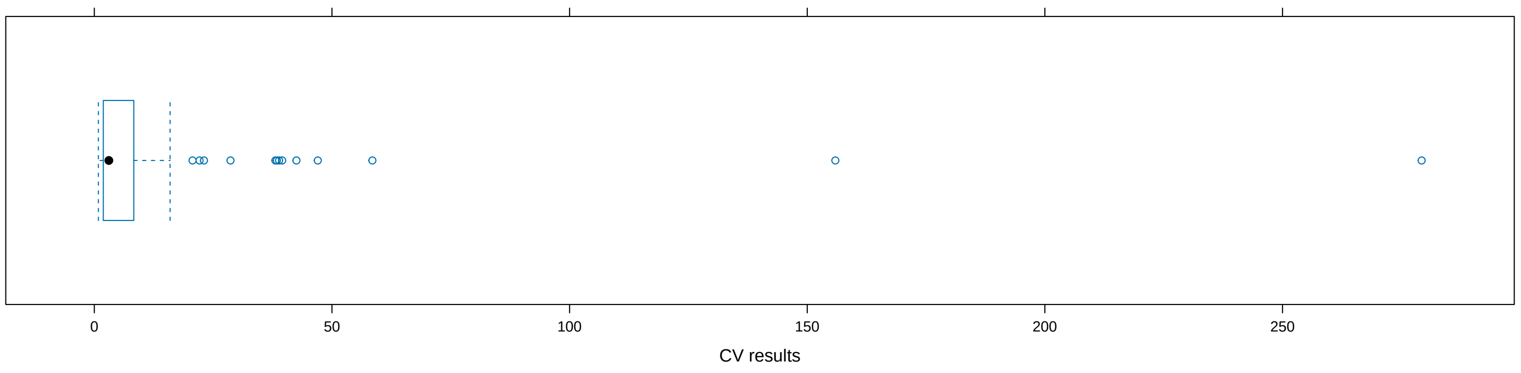
```r
cat("test RMSE:", round(rmse(y_test, test_predictions), 4), "\n")
```

```
## test RMSE: 2.9346
```

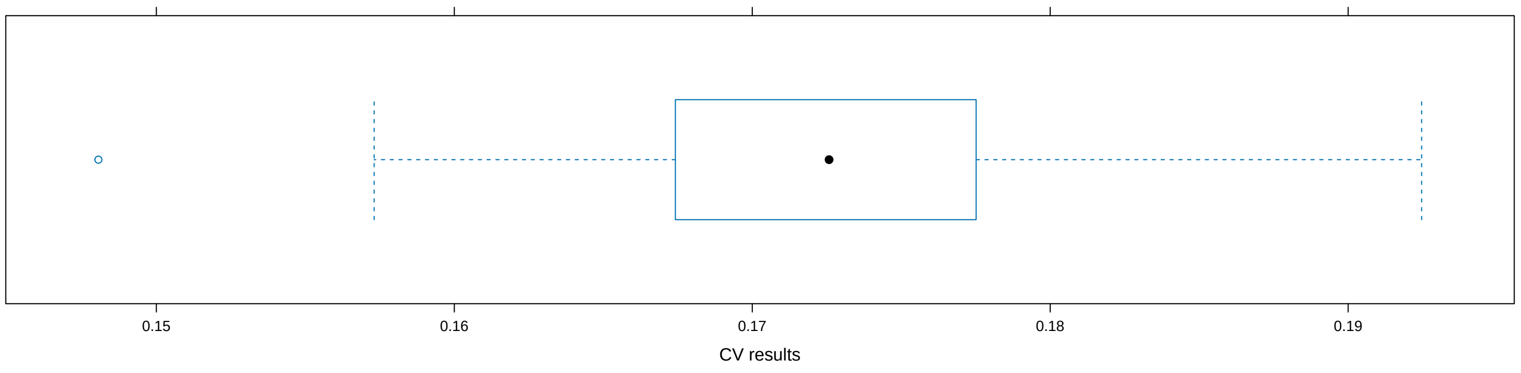
```r
# check for multicollinearity (because of "doubtful cases" warning)
cor_matrix <- cor(X_train)
high_cor <- findCorrelation(cor_matrix, cutoff = 0.95)
cat("highly correlated features: ", paste(names(X_train)[high_cor], collapse = ", "))
```

```
## highly correlated features:  Econ3, Econ7, Econ11, Econ12, Econ15, Econ16, Econ19, Econ2.lag1, Econ3.lag1, Econ5.lag1, Econ7.lag1, Econ11.lag1, Econ12.lag
```

```r
# 5 fold, 100 replications, RSMPE loss (Regular Root Mean Square Prediction Error)
cv_results <- cvFit(lm_model, data = data.frame(y = y_train, X_train), y = y_train, cost = rmspe, K = 5, R = 100)
plot(cv_results)
```



```r
# 5 fold, 100 replications, RTMSPE loss (trimmed version of RMSE, which is more robust to outliers by excluding extreme prediction errors)
cv_results <- cvFit(lm_model, data = data.frame(y = y_train, X_train), y = y_train, cost = rtmspe, K = 5, R = 100)
plot(cv_results)
```



Task 2: Best Subset Regression

Analysis

(a) The task was approached by first reducing the number of predictors since the full `regsubsets()` analysis wasn't computationally feasible.

To handle the large number of predictors, we used a simple correlation-based approach: We calculated the absolute correlation between each predictor and the response variable y and then selected the top 50 predictors with the highest absolute correlations. This method is relatively simple. The analysis was limited to a maximum of 10 regressors in the final model.

(b, c, d) The best model identified using BIC criterion included 5 predictors: `PhysFin8`, `START.YEAR`, `Econ15.lag4`, `Econ2.lag4`, and `Econ2.lag3`. The model was then fit using these predictors and evaluated on the training and test sets.

The best subset model showed significant improvements over the full model. The improvement in test performance is nearly 90%.

Metric	Full Model	Best Subset Model
Training RMSE	0.1910	0.2950
Test RMSE	2.9346	0.3072

The scatter plots show much better alignment with the diagonal line for both training and test sets. The predictions are more stable and don't show the extreme outliers seen in the full model. The range of predictions is more reasonable and consistent between training and test sets.

The Boxplots of the cross-validation results show a much tighter range of errors, both fairly close to the trimmed RMSE values of the full model - indicating better stability and less sensitivity to outliers.

The variable selection approach effectively addressed the overfitting issue present in the full model. Despite using only 5 predictors, the best subset model achieves much better generalization. The large gap between training and test performance has been substantially reduced.

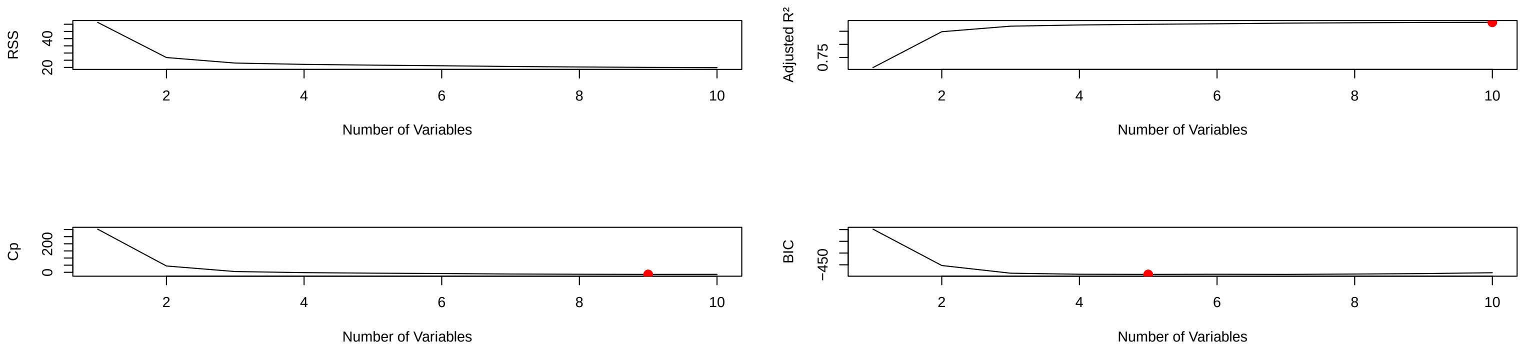
The selected variables represent a mix of physical, financial, and economic features, suggesting these are the most important predictors for building costs. The simple correlation-based feature selection method, although not sophisticated, proved effective as a preprocessing step. This analysis demonstrates the value of model simplification and variable selection in improving predictive performance, especially when dealing with many potential predictors.

Code

```r
# reduce predictors to top 50 based on correlation with response
reduce_predictors <- function(X_train, y_train, max_predictors = 50) {
    correlations <- apply(X_train, 2, function(x) abs(cor(x, y_train)))
    # ensure selected predictors exist in both train and test sets
    common_predictors <- intersect(names(X_train), names(X_test))
    correlations <- correlations[common_predictors]
    top_predictors <- names(sort(correlations, decreasing = TRUE))[1:min(max_predictors, length(correlations))]
    return(list(
        X_train = X_train[, top_predictors],
        predictor_names = top_predictors
    ))
}
reduced_data <- reduce_predictors(X_train, y_train)
X_train_reduced <- reduced_data$X_train
X_test_reduced <- X_test[, reduced_data$predictor_names]
assert_that(all(colnames(X_train_reduced) == colnames(X_test_reduced)))
```

```
## [1] TRUE
```

```r
# select best model with max 10 predictors
best_subset <- regsubsets(y ~ ., data = data.frame(y = y_train, X_train_reduced), nvmax = 10, really.big = TRUE)
summary_best <- summary(best_subset)
par(mfrow = c(2, 2))
plot(summary_best$rss, xlab = "Number of Variables", ylab = "RSS", type = "l") # RSS
plot(summary_best$adjr2, xlab = "Number of Variables", ylab = "Adjusted R²", type = "l") # adjusted R^2
points(which.max(summary_best$adjr2), summary_best$adjr2[which.max(summary_best$adjr2)], col = "red", cex = 2, pch = 20)
plot(summary_best$cp, xlab = "Number of Variables", ylab = "Cp", type = "l") # cp
points(which.min(summary_best$cp), summary_best$cp[which.min(summary_best$cp)], col = "red", cex = 2, pch = 20)
plot(summary_best$bic, xlab = "Number of Variables", ylab = "BIC", type = "l") # bic
points(which.min(summary_best$bic), summary_best$bic[which.min(summary_best$bic)], col = "red", cex = 2, pch = 20)
```



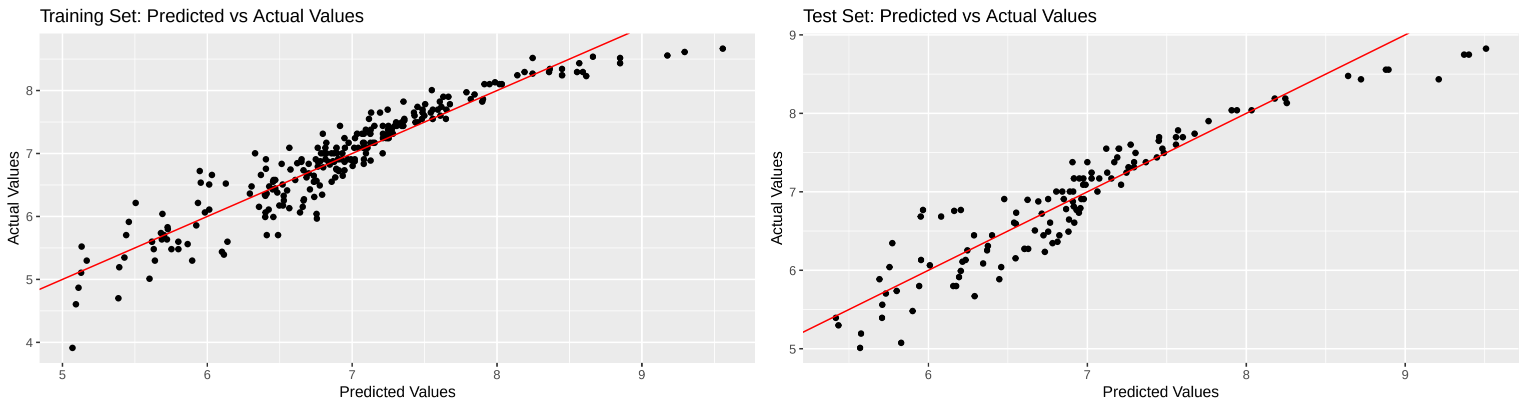
```
par(mfrow = c(1, 1))

best_model_size <- which.min(summary_best$bic)
best_coeffs <- coef(best_subset, id = best_model_size)
cat("best model size (BIC criterion):", best_model_size)
```

```
## best model size (BIC criterion): 5
```

```
cat("selected variables:", paste(names(best_coeffs)[-1], collapse = ", "))
```

```
## selected variables: PhysFin8, START.YEAR, Econ15.lag4, Econ2.lag4, Econ2.lag3
```

```
# fit
selected_vars <- names(best_coeffs)[-1]
X_train_final <- X_train_reduced[, selected_vars, drop = FALSE]
X_test_final <- X_test_reduced[, selected_vars, drop = FALSE]
lm_best <- lm(y_train ~ ., data = data.frame(X_train_final))

# eval
train_predictions <- predict(lm_best, newdata = data.frame(X_train_final))
p1 <- ggplot(data.frame(predicted = train_predictions, actual = y_train), aes(x = predicted, y = actual)) +
    geom_point() + geom_abline(intercept = 0, slope = 1, color = "red") +
    labs(x = "Predicted Values", y = "Actual Values", title = "Training Set: Predicted vs Actual Values")
test_predictions <- predict(lm_best, newdata = data.frame(X_test_final))
p2 <- ggplot(data.frame(predicted = test_predictions, actual = y_test), aes(x = predicted, y = actual)) +
    geom_point() + geom_abline(intercept = 0, slope = 1, color = "red") +
    labs(x = "Predicted Values", y = "Actual Values", title = "Test Set: Predicted vs Actual Values")
grid.arrange(p1, p2, ncol = 2)
```



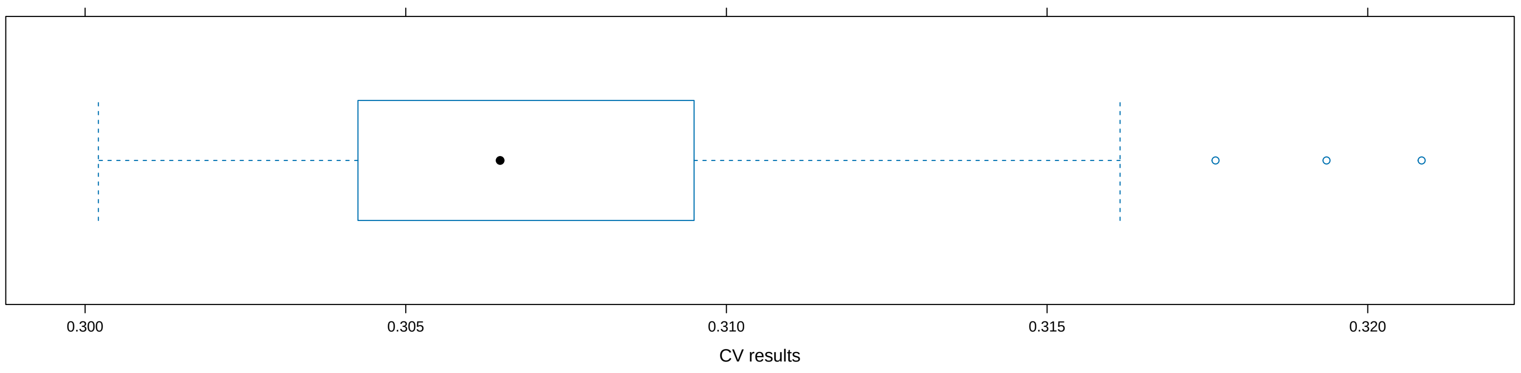
```
train_rmse <- sqrt(mean((y_train - train_predictions)^2))
test_rmse <- sqrt(mean((y_test - test_predictions)^2))
cat("training RMSE:", round(train_rmse, 4), "\n")
```

```
## training RMSE: 0.295
```

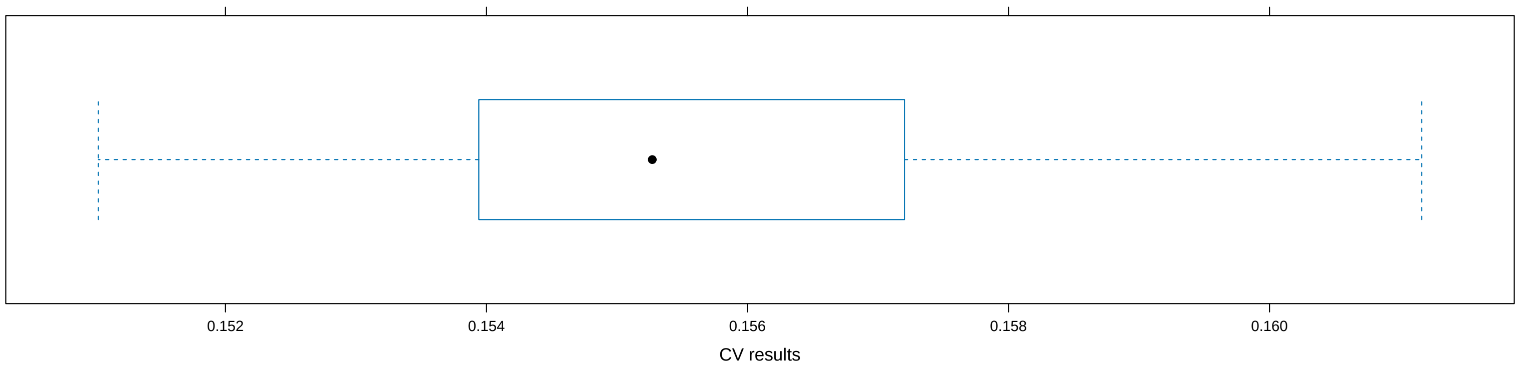
```
cat("test RMSE:", round(test_rmse, 4), "\n")
```

```
## test RMSE: 0.3072
```

```
# cv eval
X_train_matrix <- as.matrix(X_train_final)
model_data <- data.frame(y = y_train, X_train_matrix)
formula_best <- as.formula("y ~ PhysFin8 + START.YEAR + Econ15.lag4 + Econ2.lag4 + Econ2.lag3")
lm_best_cv <- lm(formula_best, data = model_data)
cv_results <- cvFit(lm_best_cv, data = model_data, y = model_data$y, cost = rmspe, K = 5, R = 100)
plot(cv_results)
```



```
cv_results_rt <- cvFit(lm_best_cv, data = model_data, y = model_data$y, cost = rtmspe, K = 5, R = 100)
plot(cv_results_rt)
```


0.152
0.154
0.156
0.158
0.160
CV results