Analysis of california\_housing dataset

Matteo Lisi

2025-03-13

## Dataset

Load the training data into R

d\_train <- read.csv("california\_housing\_train.csv")  
str(d\_train)

## 'data.frame': 100 obs. of 15 variables:  
## $ lon : num -71.1 -71.1 -71 -71.1 -71.1 ...  
## $ lat : num 42.2 42.2 42.2 42.2 42.2 ...  
## $ medv : num 22.5 46.7 11.3 25 33.2 36 17.5 22.5 21.8 36.1 ...  
## $ crim : num 0.252 0.298 9.187 0.198 0.105 ...  
## $ zn : num 0 0 0 0 40 20 0 0 0 33 ...  
## $ indus : num 10.59 6.2 18.1 10.59 6.41 ...  
## $ chas : int 0 0 0 0 1 0 0 0 0 0 ...  
## $ nox : num 0.489 0.504 0.7 0.489 0.447 0.647 0.442 0.449 0.532 0.472 ...  
## $ rm : num 5.78 7.69 5.54 6.18 7.27 ...  
## $ age : num 72.7 17 100 42.4 49 100 48.5 45.1 40.3 41.1 ...  
## $ dis : num 4.35 3.38 1.58 3.95 4.79 ...  
## $ rad : int 4 8 24 4 4 5 3 3 24 7 ...  
## $ tax : int 277 307 666 277 254 264 352 247 666 222 ...  
## $ ptratio: num 18.6 17.4 20.2 18.6 17.6 13 18.8 18.5 20.2 18.4 ...  
## $ lstat : num 18.06 3.92 23.6 9.47 6.05 ...

## Model specification

Rather than committing to a single model here I want to consider all possible models, that is including all possible subsets and combinations of the variables. I can do this automatically in R as follow:

var\_list <- colnames(d\_train[,-which(colnames(d\_train)=="medv")])  
dependent\_var <- "medv"  
  
# empty list to hold all formulae  
all\_formulas <- list()  
  
# generate combinations of variables  
for(i in 1:length(var\_list)){  
 combinations <- combn(var\_list, i)  
 num\_combinations <- ncol(combinations)  
   
 # loop over combinations and create formulas  
 for(j in 1:num\_combinations){  
 formula <- paste(dependent\_var, "~",   
 paste(combinations[,j], collapse = " + "))  
   
 all\_formulas <- c(all\_formulas, formula)  
 }  
}

This creates 16383 possible models

length(all\_formulas)

## [1] 16383

As an example, here are 5 randomly selected formulas out of all possible ones:

all\_formulas[sample(length(all\_formulas), 5)]

## [[1]]  
## [1] "medv ~ lat + crim + zn + indus + chas + rad + tax + ptratio"  
##   
## [[2]]  
## [1] "medv ~ crim + zn + indus + rm + age + lstat"  
##   
## [[3]]  
## [1] "medv ~ lon + zn + chas + rm + dis + rad + tax + ptratio + lstat"  
##   
## [[4]]  
## [1] "medv ~ lon + zn + chas + nox + age + dis + rad + tax + ptratio + lstat"  
##   
## [[5]]  
## [1] "medv ~ lon + lat + indus + chas + nox + dis + rad + tax + ptratio"

## Model evaluation and selection via cross-validation

### Custom function for LOO cross-validation

First I prepare a custom function that takes the dataset and a formula as input, run a leave-one-out cross-validation. The function return the cross-validated estimates of the mean squared error and the , that is the proportion of variance explained.

loocrossval <- function(mydata, formula){  
 n <- nrow(mydata) # Total number of observations  
 preds <- numeric(n) # Placeholder for predictions  
   
 # Extract the outcome name from the formula  
 outcome\_var <- all.vars(as.formula(formula))[1]  
   
 # Leave-One-Out Cross-Validation  
 for (i in 1:n) {  
   
 # Define training and test sets  
 train\_data <- mydata[-i, ] # All except the i-th observation  
 test\_data <- mydata[i, ] # The i-th observation  
   
 # Fit the model  
 model <- lm(formula, data = train\_data)  
   
 # Predict for the left-out observation  
 preds[i] <- predict(model, newdata = test\_data)  
 }  
   
 # Compute Mean Squared Error  
 mse\_loo <- mean((mydata[[outcome\_var]] - preds)^2)  
 rsquared\_loo <- 1 - (mse\_loo/mean((mydata[[outcome\_var]] - mean(mydata[[outcome\_var]]))^2))  
   
 # return results as named vector  
 res <- c(mse\_loo, rsquared\_loo)  
 names(res) <- c("MSE","r-squared")  
   
 # result  
 return(res)  
}

### Test all models

Now I can fit all the 16383 possible models using a loop, and estimate their out-of-sample predictive performance using LOO cross-validation (the formula that we have defined above).

**Warning: this may take a long time to complete!**

m <- list() # an empty list  
results <- data.frame()  
  
# warning: this step may take some time to complete!  
for (i in 1:length(all\_formulas)) {  
 m[[i]] <- lm(all\_formulas[[i]], data = d\_train)  
   
 crossval\_res <- loocrossval(d\_train, all\_formulas[[i]])  
   
 # print some output to show progress  
 cat("model ",i, "out of ", length(all\_formulas), "completed:\n")  
 print(crossval\_res)  
 cat("\n\n")  
   
 results <- rbind(results,   
 data.frame(MSE= crossval\_res["MSE"],  
 r\_squared = crossval\_res["r-squared"],  
 formula = all\_formulas[[i]]))  
}

The results will look like this (using the head() function to display the first few rows)

head(results)

## MSE r\_squared formula  
## MSE 87.76874 -0.001004347 medv ~ lon  
## MSE1 90.06561 -0.027200198 medv ~ lat  
## MSE2 79.97468 0.087887053 medv ~ crim  
## MSE3 81.65028 0.068776832 medv ~ zn  
## MSE4 62.46558 0.287578712 medv ~ indus  
## MSE5 90.96837 -0.037496226 medv ~ chas

### Find best model

The best model is the one that achieve the smallest MSE error in the LOO cross-validation. WE can find it by indexing the MSE column in the results

best\_formula <- results$formula[results$MSE==min(results$MSE)]  
print(best\_formula)

## [1] "medv ~ lon + lat + crim + rm + age + dis + tax + lstat"

In terms of performance, the best model has the following cross-validated MSE error and r-squared:

# LOO MSE error of best model  
results$MSE[results$MSE==min(results$MSE)]

## [1] 13.27313

# LOO r-squared of best model  
results$r\_squared[results$MSE==min(results$MSE)]

## [1] 0.8486196

The cross-validation tells us that this model is likely to generalise well to new, unseen data.

**However because we have used the cross-validation to select the model feature we should not take the MSE and r-squared at face value, as they are likely to overestimate the true predictive performance of the model. Since the same data was used for both feature selection and evaluation, these performance metrics do not provide a truly unbiased estimate of the model’s ability to generalize to new data.**

**In the study on suicidal ideation (see Moodle) the authors made a similar mistake, and claimed to have over 90% accuracy in discriminating suicidal ideators from control. This estimate was inflated, which led to the retraction of the study.**

## Out-of-sample test

When we test our best model in the real test set, we see that the performance is less impressive. Firstly, load the test set

# load test set  
d\_test <- read.csv("california\_housing\_test.csv")

Next, let’s estimate the parameters of the best model — *importantly this need to be done using the training data!*

# estimate the model parameters using training set data  
best\_model <- lm(best\_formula, data = d\_train)

Compute the predicted values:

# Predict values of test data  
preds <- predict(best\_model, newdata = d\_test)

Calculate performance in test set:

test\_mse <- mean((d\_test$medv - preds)^2)  
test\_rsquared <- 1 - (test\_mse/mean((d\_test$medv- mean(d\_test$medv))^2))

We can see that the MSE is larger (the MSE cross-validated in the training set was 13.27), indicating larger errors:

test\_mse

## [1] 38.94542

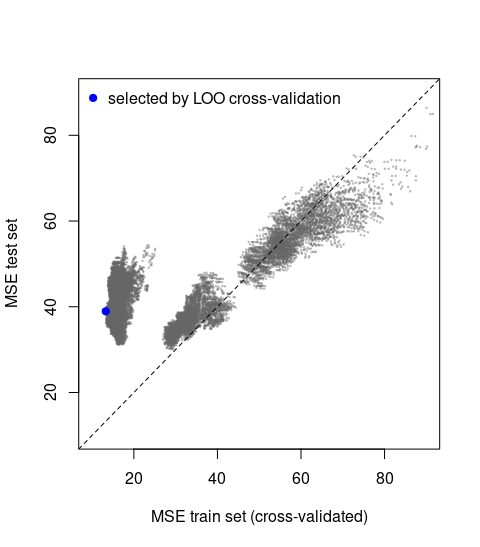
And the r-squared (proportion of variance explained) is lower (the r-squared cross-validated in the training set was 0.85)

test\_rsquared

## [1] 0.5304341

## Take-home

* Approaches like cross-validation allows us to estimate how a model may perform on new, unseen data.
* If cross-validation is used not only for model evaluation but also for feature selection, the reported performance will be optimistic. Since the same data is used for selecting variables and evaluating the model, we lack an unbiased estimate of its true generalization ability. This is evident from the test set results, where the proportion of explained variance drops from ~85% to ~53%.
* Choosing the best model via cross-validation is a reasonable approach, but it does not guarantee that the selected model is truly the best. In our case, evaluating all possible models on the test set shows that the one chosen via LOO cross-validation is not the best-performing model on the test set (marked by the red dot in the figure below).



Each grey dot is a distinct model (16383 in total), characterised by a different combinations of predictors. On the horizontla axis is the cross-validated mean squared error (MSE) in the training data, whereas on the vertical axis is the MSE on the test data. The blue dot indicates the best model according to the LOO cross-validation procedure run on the training data.