Tulillesztes

Zoltan Kekecs

November 13, 2019

Contents

1 Automatizalt modellvalasztas es tulillesztes

1.1 Absztrakt

Ez a gyakorlat azt demonstralja, mik a kovetkezmenyei ha elmeleti megalapozottsag nelkul tul sok prediktort foglalunk a modellunkbe. Ebben az esetben a modellunk tul flexibilis lesz, es nagyon jol fog illeszkedni a sajat adatainkhoz, de az illeszkedese a valos populaciora rossz lesz. Az automatizalt modellszelekcios eljarasok ezert gyakran rossz statisztikai dontesekhez vezetnek. Az alabbi kodban erre lathatunk egy demonstraciot.

1.2 Adatmenedzsment es leiro statisztikak

1.2.1 Package-ek betoltese

```
library(tidyverse)
```

1.2.2 A King County lakaseladas adattabla betoltese

Ebben a gyakorlatban lakasok es hazak arait fogjuk megbecsulni.

Egy Kaggle-rol szarmazo adatbazist hasznalunk, melyben olyan adatok szerepelnek, melyeket valoszinusithetoen alkalmasak lakasok aranak bejoslasara. Az adatbazisban az USA Kings County-bol szarmaznak az adatok (Seattle es kornyeke).

Az adatbazisnak csak egy kis reszet hasznaljuk (N = 200).

```
data_house = read.csv("https://bit.ly/2DpwK0r")
```

1.2.3 Adatellenorzes

Mindig nezd at az altalad hasznalt adattablat. Ezt mar megtettuk az elozo gyakorlatban, igy ezt most itt mellozzuk, de a korabbi tapasztalatok alapjan atalakitjuk az arat (price) millio forintra, es a negyzetlabban szereplo terulet ertekeket negyzetmeterre.

```
data_house %>%
  summary()
```

```
##        id                           date          price
##  Min.   :1.600e+07   20140623T000000:  5   Min.   : 153503
##  1st Qu.:1.885e+09   20141107T000000:  5   1st Qu.: 299250
```

```
## Median :3.521e+09   20150317T000000:  4   Median : 425000
## Mean   :4.113e+09   20140627T000000:  3   Mean   : 453611
## 3rd Qu.:6.424e+09   20140717T000000:  3   3rd Qu.: 550000
## Max.   :9.819e+09   20140902T000000:  3   Max.   :1770000
##                     (Other)        :177
##    bedrooms      bathrooms      sqft_living      sqft_lot
## Min.   :1.00   Min.   :0.75   Min.   : 590   Min.   :   914
## 1st Qu.:3.00   1st Qu.:1.00   1st Qu.:1240   1st Qu.:  4709
## Median :3.00   Median :1.75   Median :1620   Median :  7270
## Mean   :2.76   Mean   :1.85   Mean   :1728   Mean   : 12985
## 3rd Qu.:3.00   3rd Qu.:2.50   3rd Qu.:1985   3rd Qu.: 10187
## Max.   :3.00   Max.   :3.50   Max.   :4380   Max.   :217800
##
##     floors        waterfront         view         condition
## Min.   :1.000   Min.   :0.000   Min.   :0.000   Min.   :3.00
## 1st Qu.:1.000   1st Qu.:0.000   1st Qu.:0.000   1st Qu.:3.00
## Median :1.000   Median :0.000   Median :0.000   Median :3.00
## Mean   :1.472   Mean   :0.005   Mean   :0.145   Mean   :3.42
## 3rd Qu.:2.000   3rd Qu.:0.000   3rd Qu.:0.000   3rd Qu.:4.00
## Max.   :3.000   Max.   :1.000   Max.   :4.000   Max.   :5.00
##
##     grade         sqft_above    sqft_basement      yr_built
## Min.   : 5.00   Min.   : 590   Min.   :   0.0   Min.   :1900
## 1st Qu.: 7.00   1st Qu.:1090   1st Qu.:   0.0   1st Qu.:1946
## Median : 7.00   Median :1375   Median :   0.0   Median :1968
## Mean   : 7.36   Mean   :1544   Mean   : 184.1   Mean   :1968
## 3rd Qu.: 8.00   3rd Qu.:1862   3rd Qu.: 315.0   3rd Qu.:1993
## Max.   :11.00   Max.   :4190   Max.   :1600.0   Max.   :2015
##
##  yr_renovated        zipcode           lat             long
## Min.   :   0.00   Min.   :98001   Min.   :47.18   Min.   :-122.5
## 1st Qu.:   0.00   1st Qu.:98033   1st Qu.:47.49   1st Qu.:-122.3
## Median :   0.00   Median :98065   Median :47.58   Median :-122.2
## Mean   :  79.98   Mean   :98078   Mean   :47.57   Mean   :-122.2
## 3rd Qu.:   0.00   3rd Qu.:98117   3rd Qu.:47.68   3rd Qu.:-122.1
## Max.   :2014.00   Max.   :98199   Max.   :47.78   Max.   :-121.7
##
## sqft_living15    sqft_lot15          has_basement
## Min.   : 740   Min.   :   914   has basement: 65
## 1st Qu.:1438   1st Qu.:  5000   no basement :135
## Median :1715   Median :  7222
## Mean   :1793   Mean   : 11225
## 3rd Qu.:2072   3rd Qu.: 10028
## Max.   :3650   Max.   :208652
##
```

```r
data_house = data_house %>%
  mutate(price_mill_HUF = (price * 293.77)/1000000,
         sqm_living = sqft_living * 0.09290304,
         sqm_lot = sqft_lot * 0.09290304,
         sqm_above = sqft_above * 0.09290304,
         sqm_basement = sqft_basement * 0.09290304,
         sqm_living15 = sqft_living15 * 0.09290304,
         sqm_lot15 = sqft_lot15 * 0.09290304
```

```
        )
```

A modellvalasztas legfontosabb szabalya:

Mindig azt a modellt valasztjuk, ami elmeletileg alatamasztott es/vagy korabbi kutatasi eredmenyek tamogatjak, mert az automatikus modellvalasztas rossz modellekhez vezet a tulillesztes (overfitting) miatt.

A jegyzet tovabbi resze angolul elerheto:

"Predicting" variability of the outcome **in your original data** is easy If you fit a model that is too flexible, you will get perfect fit on your intitial data.

For example you can fit a line that would cover your data perfectly, reaching 100% model fit… to a dataset where you already knew the outcome.

However, when you try to apply the same model to new data, it will produce bad model fit. In most cases, worse, than a simple regression.

In this context, data on which the model was built is called the training set, and the new data where we test the true prediction efficiency of a model is called the test set. The test set can be truely newly collected data, or it can be a set aside portion of our old data which was not used to fit the model.

Linear regression is very inflexible, so it is less prone to overfitting. This is one of its advantages compared to more flexible prediction approaches.

1.3 Comparing model performance on the training set and the test set

In the next part of the exercise we will demonstrate that the more predictors you have, the higher your R^2 will be, even if the predictors have nothing to do with your outcome variable.

First, we will generate some random variables for demonstration purposes. These will be used as predictors in some of our models in this exercise. It is important to realize that these variables are randomly generated, and have no true relationship to the sales price of the apartments. Using these random numbers we can demonstrate well how people can be mislead by good prediction performance of models containing many predictors.

```
rand_vars = as.data.frame(matrix(rnorm(mean = 0, sd = 1, n = 50*nrow(data_house)), ncol = 50))
data_house_withrandomvars = cbind(data_house, rand_vars)
```

We create a new data object from the first half of the data (N = 100). We will use this to fit our models on. This is our training set. We set aside the other half of the dataset so that we will be able to test prediction performance on it later. This is called the test set.

```
training_set = data_house_withrandomvars[1:100,] # training set, using half of the data
test_set = data_house_withrandomvars[101:200,] # test set, the other half of the dataset
```

Now we will perform a hierarchical regression where first we fit our usual model predicting price with sqm_living and grade on the training set. Next, we fit a model containing sqm_living and grade and the 50 randomly generated variables that we just created.

(the names of the random variables are V1, V2, V3, …)

```
mod_house_train <- lm(price_mill_HUF ~ sqm_living + grade, data = training_set)
mod_house_rand_train  <- lm(price_mill_HUF ~ sqm_living + grade+ V1 + V2 + V3 + V4 + V5 + V6 + V7 +
                            V8 + V9 + V10 + V11 + V12 + V13 + V14 + V15 + V16 + V17 +
                            V18 + V19 + V20 + V21 + V22 + V23 + V24 + V25 + V26 + V27 +
                            V28 + V29 + V30 + V31 + V32 + V33 + V34 + V35 + V36 + V37 +
                            V38 + V39 + V40 + V41 + V42 + V43 + V44 + V45 + V46 + V47 +
```

```
                              V48 + V49 + V50,
                            data = training_set)
```

Now we can compare the model performance. First, if we look at the normal R^2 indexes of the models or the RSS, we will find that the model using the random variables (mod_house_rand_train) was much better at predicting the training data. The error was smaller in this model, and the overall variance explained is bigger. You can even notice that some of the random predictors were identified as having significant added prediction value in this model, even though they are not supposed to be related to price at all, since we just created them randomly. This is because some of these variables are alligned with the outcome to some extend by random chance.

```
summary(mod_house_train)
```

```
##
## Call:
## lm(formula = price_mill_HUF ~ sqm_living + grade, data = training_set)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -123.594  -24.837   -4.598   23.841  138.479
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -93.7083    36.2391  -2.586 0.011201 *
## sqm_living    0.3659     0.1038   3.524 0.000650 ***
## grade        22.8527     6.3178   3.617 0.000475 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 43.57 on 97 degrees of freedom
## Multiple R-squared:  0.4765, Adjusted R-squared:  0.4657
## F-statistic: 44.15 on 2 and 97 DF,  p-value: 2.327e-14
```

```
summary(mod_house_rand_train)
```

```
##
## Call:
## lm(formula = price_mill_HUF ~ sqm_living + grade + V1 + V2 +
##     V3 + V4 + V5 + V6 + V7 + V8 + V9 + V10 + V11 + V12 + V13 +
##     V14 + V15 + V16 + V17 + V18 + V19 + V20 + V21 + V22 + V23 +
##     V24 + V25 + V26 + V27 + V28 + V29 + V30 + V31 + V32 + V33 +
##     V34 + V35 + V36 + V37 + V38 + V39 + V40 + V41 + V42 + V43 +
##     V44 + V45 + V46 + V47 + V48 + V49 + V50, data = training_set)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -68.763 -17.288   1.377  17.776  98.598
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -87.2727    49.2466  -1.772   0.0829 .
## sqm_living    0.2124     0.1550   1.370   0.1771
## grade        25.7541     8.6996   2.960   0.0048 **
## V1           -1.6709     6.3589  -0.263   0.7939
## V2           -7.4011     6.5022  -1.138   0.2608
```

```
## V3           1.6273     6.4865   0.251   0.8030
## V4          -9.9508     7.1281  -1.396   0.1693
## V5           4.7208     5.8270   0.810   0.4219
## V6          -2.0772     6.9128  -0.300   0.7651
## V7          -5.8652     6.6965  -0.876   0.3856
## V8          -7.0098     6.2898  -1.114   0.2707
## V9           2.5468     6.5709   0.388   0.7001
## V10          5.6209     7.4053   0.759   0.4516
## V11          3.3603     6.7008   0.501   0.6184
## V12         -2.4469     6.3867  -0.383   0.7034
## V13         -7.9675     5.9296  -1.344   0.1855
## V14         -0.9398     6.2445  -0.151   0.8810
## V15         -4.4446     5.6724  -0.784   0.4372
## V16         -0.7827     7.4576  -0.105   0.9169
## V17          0.1011     6.6920   0.015   0.9880
## V18         13.5045     6.0788   2.222   0.0312 *
## V19          6.2938     6.7415   0.934   0.3553
## V20          2.2145     6.0562   0.366   0.7163
## V21          4.4526     5.5723   0.799   0.4283
## V22          1.4972     7.4326   0.201   0.8412
## V23         -1.0509     6.3849  -0.165   0.8700
## V24        -14.0406     6.4571  -2.174   0.0347 *
## V25         -9.8755     6.3435  -1.557   0.1262
## V26          6.6577     6.1845   1.077   0.2872
## V27         -6.8472     6.2097  -1.103   0.2758
## V28         -5.4054     7.0240  -0.770   0.4454
## V29         -5.3591     6.8481  -0.783   0.4378
## V30         -2.0831     5.7878  -0.360   0.7205
## V31         -4.7605     6.4296  -0.740   0.4627
## V32          4.9896     7.4736   0.668   0.5076
## V33         -3.7361     6.0277  -0.620   0.5384
## V34         -2.6411     5.2476  -0.503   0.6171
## V35         -5.8065     5.5339  -1.049   0.2994
## V36          0.1887     6.5629   0.029   0.9772
## V37          3.9660     7.6658   0.517   0.6073
## V38         -9.6516     6.4356  -1.500   0.1404
## V39         -1.2897     6.0524  -0.213   0.8322
## V40         -9.4188     6.4216  -1.467   0.1491
## V41         -1.3324     7.0893  -0.188   0.8517
## V42         -5.2295     6.7707  -0.772   0.4438
## V43          3.1279     6.6817   0.468   0.6419
## V44         -9.0666     6.7567  -1.342   0.1861
## V45         -6.9195     6.2557  -1.106   0.2743
## V46         -0.6677     5.8846  -0.113   0.9102
## V47         -6.1868     6.0440  -1.024   0.3112
## V48          9.3427     6.2234   1.501   0.1400
## V49          5.3021     5.9611   0.889   0.3783
## V50          7.5109     7.2104   1.042   0.3029
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 43.82 on 47 degrees of freedom
## Multiple R-squared:  0.7435, Adjusted R-squared:  0.4597
## F-statistic:  2.62 on 52 and 47 DF,  p-value: 0.0005281
```

```r
pred_train <- predict(mod_house_train)
pred_train_rand <- predict(mod_house_rand_train)
RSS_train = sum((training_set[,"price_mill_HUF"] - pred_train)^2)
RSS_train_rand = sum((training_set[,"price_mill_HUF"] - pred_train_rand)^2)
RSS_train
```

```
## [1] 184140.6
```

```r
RSS_train_rand
```

```
## [1] 90231.25
```

That is why we need to use model fit indexes that are more sensitive to the number of variables we included as redictors, to account for the likelyhood that some variables will show a correlation by chance. Such as adjusted R^2, or the AIC. The anova() test is also sensitive to the number of predictors in the models, so it is not easy to fool by adding a bunch of random data as predictors.

```r
summary(mod_house_train)$adj.r.squared
```

```
## [1] 0.4657206
```

```r
summary(mod_house_rand_train)$adj.r.squared
```

```
## [1] 0.4596818
```

```r
AIC(mod_house_train)
```

```
## [1] 1043.616
```

```r
AIC(mod_house_rand_train)
```

```
## [1] 1072.284
```

```r
anova(mod_house_train, mod_house_rand_train)
```

```
## Analysis of Variance Table
##
## Model 1: price_mill_HUF ~ sqm_living + grade
## Model 2: price_mill_HUF ~ sqm_living + grade + V1 + V2 + V3 + V4 + V5 +
##     V6 + V7 + V8 + V9 + V10 + V11 + V12 + V13 + V14 + V15 + V16 +
##     V17 + V18 + V19 + V20 + V21 + V22 + V23 + V24 + V25 + V26 +
##     V27 + V28 + V29 + V30 + V31 + V32 + V33 + V34 + V35 + V36 +
##     V37 + V38 + V39 + V40 + V41 + V42 + V43 + V44 + V45 + V46 +
##     V47 + V48 + V49 + V50
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     97 184141
## 2     47  90231 50     93909 0.9783 0.5314
```

1.4 Result-based models selection

(Result-based models selection is only shown here with demonstration purposes, to show how it can mislead researchers. Whenever possible, stay away from using such approaches, and rely on theoretical considerations and previous data when building models.)

After seeing the performance of mod_house_rand_train, and not knowing that it contains random variables, one might be tempted to build a model with only the predictors that were identified as having a significant added predictive value, to improve the model fit indices (e.g. adjusted R^2 or AIC). And that would acieve exactly that: it would result in the indcrease of the indexes, but not the actual prediction efficiency, so the better indexes would be just an illusion resulting from the fact that we have “hidden” from the statistical tests, that we have tried to use a lot of predictors in a previous model.

Excluding variables that seem "useless" based on the results will blind the otherwise sensitive measures of model fit. This is what happens when using automated model selection procedures, such as backward regression.

In the example below we use backward regression. This method first fits a complete model with all of the specified predictors, and then determins which predictor has the smallest amount of unique added explanatory value to the model, and excludes it from the list of predictors, refitting the model without this predictor. This procedure is iterated until until there is no more predictor that can be excluded without significantly reducing model fit, at which point the process stops.

```
mod_back_train = step(mod_house_rand_train, direction = "backward")
```

The final model with the reduced number of predictors will have much better model fit indexes than the original compex model, because the less useful variables were excluded, and only the most influential ones were retained, resulting in a small and powerful model. Or at least this is what the numbers would suggest us on the training set.

Lets compare the prediction performance of the final model returned by backward regression (mod_back_train) with the model only containing our good old predictors, sqm_living and grade (mod_house_train) on the training set.

```
anova(mod_house_train, mod_back_train)
```

```
## Analysis of Variance Table
##
## Model 1: price_mill_HUF ~ sqm_living + grade
## Model 2: price_mill_HUF ~ sqm_living + grade + V2 + V4 + V13 + V24 + V25 +
##     V28 + V32 + V40 + V45 + V47 + V49
##   Res.Df    RSS Df Sum of Sq     F   Pr(>F)
## 1     97 184141
## 2     86 124718 11     59423 3.725 0.000231 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(mod_house_train)$adj.r.squared
```

```
## [1] 0.4657206
```

```
summary(mod_back_train)$adj.r.squared
```

```
## [1] 0.5918488
```

```
AIC(mod_house_train)
```

```
## [1] 1043.616
```

```
AIC(mod_back_train)
```

```
## [1] 1026.652
```

All of the above model comparison methods indicate that the backward regression model (mod_back_train) performs better. We know that this model can't be too much better than the smaller model, since it only contains a number of randomly generated variables in addition to the two predictors in the smaller model. So if we would only rely on these numbers, we would be fooled to think that the backward regression model is better.

1.5 Testing performance on the test set

A surefire way of determining actual model performance is to test it on new data, data that was not used in the "training" of the model. Here, we use the set aside test set to do this.

Note that we did not re-fit the models on the test set, we use the models fitted on the training set to make our predictions using the predict() function on the test_set!!!

```
# calculate predicted values
pred_test <- predict(mod_house_train, test_set)
pred_test_back <- predict(mod_back_train, test_set)

# now we calculate the sum of squared residuals
RSS_test = sum((test_set[,"price_mill_HUF"] - pred_test)^2)
RSS_test_back = sum((test_set[,"price_mill_HUF"] - pred_test_back)^2)
RSS_test
```

```
## [1] 314081.5
```

```
RSS_test_back
```

```
## [1] 388365.5
```

This test reveals that the backward regression model has more error than the model only using sqm_living and grade.

1.6 BOTTOM LINE

1. Model selection should be done pre-analysis, based on theory, previous results from the literature, or conventions on the field. Post-hoc result-driven predictor selection can lead to overfitting.
2. The only good test of a model's true prediction performance is to test the accuracy of its predictions on new data (or a set-asid test set)