[DATA-02] Linear regression

Miguel-Angel Canela, IESE Business School

September 18, 2016

Introduction

In a data science context, the prediction of a numeric variable is called regression. Regression models are not necessarily related to a mathematical equation, as in Statistics, although this is the favourite approach for beginners. When this equation is linear, we have **linear regression**, which is the object of this lecture. Although the predictions of a linear regression model can usually be improved by more sophisticated techniques, most data scientists start there, because it helps them to understand the data.

Two alternatives to linear regression are:

- **Regression trees**. In this course, tree algorithms are discusused in a classification context, but some of them, like the CART algorithm used in the package `rpart`, apply also to regression trees.
- **Neural networks**. A neural network is a programming device whose design is based on the models developed by biologists for the brain neurons. Among the many types of neural networks, the most popular is the **multilayer perceptron** (MLP), which can be regarded as a set of (nonlinear) regression equations. In R, the package `nnet` provides a simple approach to MLP regression models.

Evaluation of a linear regression model

In general, regression models are evaluated through the prediction errors. The basic schema is

$$\text{Error} = \text{Actual value} - \text{Predicted value}.$$

Prediction errors are called **residuals** in linear regression. In that special case, the mean of the prediction errors is zero, but this is no longer true in other models. The usual approach for estimating the regression coefficients is the **least squares method**, which minimizes the sum of the squared residuals.

In Statistics, the predictive power of a linear regression model is evaluated through the **residual sum of squares**. The **R-square statistic** is a standardized measure which operationalizes this. More specifically, it takes advantage of the formula

$$\text{var}\big(\text{Actual values}\big) = \text{var}\big(\text{Predicted values}\big) + \text{var}\big(\text{Error}\big),$$

to evaluate the model through the proportion of **variance explained**,

$$R^2 = \frac{\text{var}\big(\text{Predicted values}\big)}{\text{var}\big(\text{Actual values}\big)}\,.$$

It turns out that the square root of R-squared coincides with the **correlation** between actual values and predicted values. Although this stops being true for other regression methods, this correlation is still the simplest method for a fast and dirty evaluation of a regression model.

Dummies and factors

Categorical variables (also called nominal) enter a regression equation through 1/0 variables, called **dummies**. In R, we can manage this in two ways.

- We can create dummies explicitly and include them in the equation (as many dummies as the number of categories minus 1).
- We can specify the variable as a factor and put it in the formula. Then R creates the dummies itself, without need of changing the data set.

Transformations

Transformations, such the square root or the logarithm, are recommended in Statistics textbooks in many situations. The only transformation that we may use in this course is the **log transformation**, so my comments are restricted to this case. Let me start by refreshing some basic facts.

In R, `log` denotes the **natural logarithm** (it is `ln` in Excel). This function is the inverse of the exponential formula, meaning that

$$y = \log(x) \iff \exp(y) = x.$$

An important property of the logarithm is that it transforms products into sums and quotients into differences,

$$\log\left(x_1 x_2\right) = \log(x_1) + \log(x_2), \qquad \log\left(\frac{x_1}{x_2}\right) = \log(x_1) - \log(x_2).$$

The use of the log transformation can be supported by various arguments:

- Limiting the influence of extreme observations.
- Rendering assumptions about normality more reliable. This applies only to the dependent variable in a testing context.
- When an independent variable has an effect on the dependent variable which is stronger for low values. For instance, the effect on your influence of adding a political contact to your portfolio is strong for the first contacts, but gets weaker and weaker when you already have many contacts.

Is this useful in a data mining context? In data mining the focus is on prediction. So, we are pragmatic. If a transformation improves our prediction, it is welcome. As shown in the example, it is easy to check whether the log transformation of the price improves the model or not.

Example: Windsor housing prices

In this example, I develop a pricing model for residential houses in the area of Windsor, Ontario. The data set, provided by the Windsor and Essex County Real Estate Board, covers the sales of residential houses in Windsor during July, August and September of the previous year through the Multiple Listing Service. The sample size is 546.

Besides the sale price, the data set contains information describing the key features of each house:

- The lot size, which should play a strong role in the model.
- The number of bedrooms.
- The number of full bathrooms (including at least the toilet, the sink and the bathtub).
- The number of stories, excluding the basement.
- The number of garage places.
- A dummy for having a recreational room.
- A dummy for having a full and finished basement.

- A dummy for using gas for hot water heating.
- A dummy for having central air conditioning.
- A dummy for having a driveway.
- A dummy for being located in a preferred neighborhood of the city (Riverside or South Windsor).

Most of the data sets used in this course come in csv files. Theese are text files that use the comma as the column separator. This format is very popular, although it can lead to errors with text data. The names of the attributes are in the first row, and every other row corresponds to an instance.

```
windsor <- read.csv(file="windsor.csv")
```

Please, note that, where I have written `"windsor.csv"`, you have to write the complete path of this file in your computer, for the file to be found by R. As I am writing it, my code will work only if the file is in the **working directory**. You can change the working directory with the function `setwd`.

Actually, `windsor` is a data frame, with 546 and 11 columns.

```
str(windsor)
```

```
## 'data.frame':    546 obs. of  12 variables:
##  $ price  : int  76500 70100 90200 110200 111100 120200 120200 125700 152600 161200 ...
##  $ lotsize: int  5850 4000 3060 6650 6360 4160 3880 4160 4800 5500 ...
##  $ nbdrm  : int  3 2 3 3 2 3 3 3 3 3 ...
##  $ nbhrm  : int  1 1 1 1 1 1 2 1 1 2 ...
##  $ nstor  : int  2 1 1 2 1 1 2 3 1 4 ...
##  $ ngar   : int  1 0 0 0 0 0 2 0 0 1 ...
##  $ drive  : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ recrm  : int  0 0 0 1 0 1 0 0 1 1 ...
##  $ base   : int  1 0 0 0 0 1 1 0 1 0 ...
##  $ gas    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ air    : int  0 0 0 0 0 1 0 0 0 1 ...
##  $ neigh  : int  0 0 0 0 0 0 0 0 0 0 ...
```

A linear regression model can be obtained with the function `lm`. The key arguments are `formula` and `data`, of obvious meaning. The syntax of the formula is `y ~ x1 + x2 + ...`

```
fm1 <- price ~ lotsize + nbdrm + nbhrm + nstor + ngar + drive + recrm + base +
  gas + air + neigh
mod1 <- lm(formula=fm1, data=windsor)
```

`mod1` is an object of class `list`. A **list** is like a vector, but it can contain objects of different type. The same will true for other models used in this course, but I will not discuss what the content is for other models discussed in this course, because that would be too technical. If you are interested in the structure of an R object, you can explore it with the function `names`, or (at your own risk) with `str`, which provides a lot of information but can give you too much output for complex objects.

```
names(mod1)
```

```
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
```

In most models, the function `summary` provides useful information. for a linear regression model, it includes the regression coefficients, with the corresponding **p-values**, and the R-squared statistic, whose square root, R = 0.820, is the correlation between actual and predicted prices.

The function `summary` also provides a summary of the residuals. The maximum and minimum have less interest, but the median (50% percentile), and the first (25% percentile) and third quartiles (75% percentile) can be useful.

```
summary(mod1)
```

```
##
## Call:
## lm(formula = fm1, data = windsor)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -75356 -16944  -1089  13377 136321
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -7358.522   6210.112  -1.185 0.236574
## lotsize         6.460      0.638  10.124  < 2e-16 ***
## nbdrm        3339.897   1907.037   1.751 0.080459 .
## nbhrm       26105.578   2713.786   9.620  < 2e-16 ***
## nstor       11943.252   1685.351   7.087 4.37e-12 ***
## ngar         7732.392   1530.992   5.051 6.06e-07 ***
## drive       12174.532   3725.271   3.268 0.001152 **
## recrm        8215.582   3460.639   2.374 0.017949 *
## base         9931.793   2892.474   3.434 0.000642 ***
## gas         23368.073   5860.627   3.987 7.61e-05 ***
## air         23003.105   2832.361   8.122 3.21e-15 ***
## neigh       17064.219   3040.131   5.613 3.20e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 28090 on 534 degrees of freedom
## Multiple R-squared:  0.6731, Adjusted R-squared:  0.6664
## F-statistic: 99.95 on 11 and 534 DF,  p-value: < 2.2e-16
```

The predicted values of a linear regression model are actually contained in the model (the element `fitted.values`), but I use here the function `predict` in order to apply the same steps as with other models. `predict` can be applied to a new data set, as far as it contains all the variables included on the right side of the formula of the model.

```
pred1 <- predict(object=mod1, newdata=windsor)
```

We can also examine directly the prediction errors, that is, the regression residuals,

```
res1 <- windsor$price - pred1
```

The standard deviation of the residuals is 27,807.30. Although the correlation achieved with this regression equation may look quite strong, the residual standard deviation is still the 57.2% of that of the price (48,364.55).

```
sd(res1)
```

```
## [1] 27807.3
```

```
sd(windsor$price)
```

```
## [1] 48634.55
```

```
sd(res1)/sd(windsor$price)
```

```
## [1] 0.5717602
```

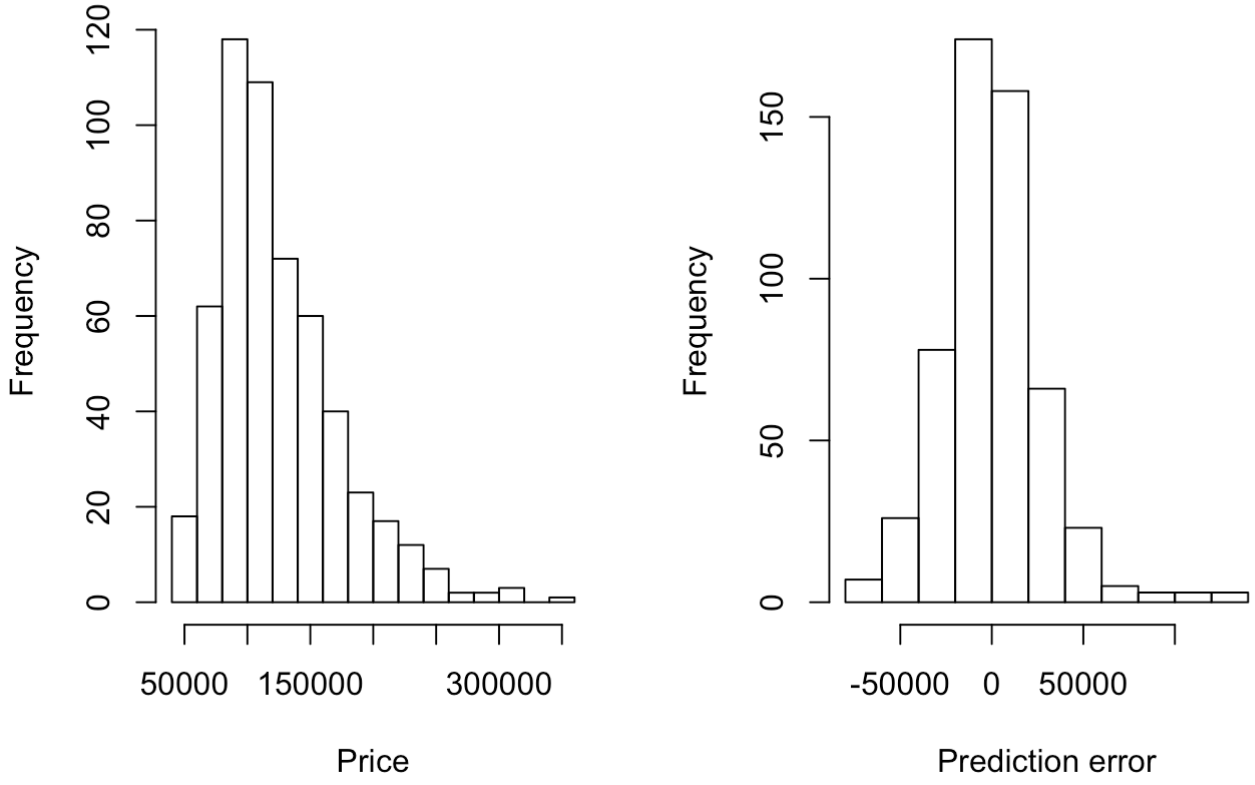
Another way of assessing the performance of the model is through the percentage of cases in which the error exceeds a given threshold. For instance, in this case only 12.8% of the prediction errors exceed 40,000 dollars (in absolute value), while 22.5% exceed 30,000. This allows for an assessment in dollar terms which is always helpful and may bridge the gap between the analyst and a non-trained audience. To get this proportion, I write the expression that I want to evaluate, `abs(res1)>40000` . This creates a logical vector (`TRUE/FALSE`). When I apply `mean` , R transforms this vector into a dummy, so the mean gives the proportion of `TRUE` values.

```
mean(abs(res1)>40000)
```

```
## [1] 0.1282051
```

Although the price has a **skewed distribution**, that of the prediction error is reasonably symmetric, as shown in the exhibit below, obtained by means of

```
par(mfrow=c(1,2))
hist(windsor$price, main="", xlab="Price")
hist(res1, main="", xlab="Prediction error")
```



The command `par(mfrow=c(1,2))` splits the graphic device in two parts, so the next two plots will fill the two slots. With the argument `mfrow` we control the partition. The `mfrow=c(1,2)` specification means "one row, two columns". You may resize the window in your screen top take advantage of this. Note that this split looks great for the R console, but can produce a poor result in RStudio, unless youn resize the windows.

Log transformation

The distribution of the price is skewed, but not severely. So, I don't expect a big change from using log scale for prices. Nevertheless, I try it as an illustration. Note that logarithms can be introduced directly in the formula.

```
fm2 <- log(price) ~ lotsize + nbdrm + nbhrm + nstor + ngar + drive + recrm + base +
  gas + air + neigh
mod2 <- lm(formula=fm2, data=windsor)
summary(mod2)
```

```
##
## Call:
## lm(formula = fm2, data = windsor)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.67936 -0.12226  0.01633  0.12873  0.67753
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.063e+01  4.725e-02 224.869  < 2e-16 ***
## lotsize     5.058e-05  4.855e-06  10.418  < 2e-16 ***
## nbdrm       3.404e-02  1.451e-02   2.346  0.01933 *
## nbhrm       1.677e-01  2.065e-02   8.124 3.15e-15 ***
## nstor       9.229e-02  1.282e-02   7.197 2.10e-12 ***
## ngar        5.078e-02  1.165e-02   4.359 1.57e-05 ***
## drive       1.306e-01  2.834e-02   4.606 5.14e-06 ***
## recrm       7.350e-02  2.633e-02   2.791  0.00544 **
## base        9.942e-02  2.201e-02   4.517 7.71e-06 ***
## gas         1.783e-01  4.459e-02   3.999 7.26e-05 ***
## air         1.780e-01  2.155e-02   8.259 1.16e-15 ***
## neigh       1.271e-01  2.313e-02   5.495 6.04e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2137 on 534 degrees of freedom
## Multiple R-squared:  0.6765, Adjusted R-squared:  0.6699
## F-statistic: 101.5 on 11 and 534 DF,  p-value: < 2.2e-16
```

Now, the summary should be taken with care, since it refers to prices in log scale. For instance, the summary of the residuals is not useful, unless we transform these values into prices with an exponential transformation. Indeed, from the relationship

$$\text{Residual} = \log\big(\text{Actual price}\big) - \log\big(\text{Predicted price}\big)$$

we can derive

$$\exp\big(\text{Residual}\big) = \frac{\text{Actual price}}{\text{Predicted price}}\,.$$

Applying the exponential to the percentiles given in the summary, I get that: 25% of the predicted prices are below of 88.5% of the actual price, 50% are below the 101.6% of the actual price and 75% are below the 113.7%.

```
exp(-0.12226)
```

```
## [1] 0.8849183
```

```
exp(0.01633)
```

```
## [1] 1.016464
```

```
exp(0.12873)
```

```
## [1] 1.137383
```

The R-squared value reported is practically the same in the two models. But, if we want the right comparison, we have to calculate the correlation between predicted prices and actual prices for the second model.

```
pred2 <- exp(predict(mod2, newdata=windsor))
cor(windsor$price, pred2)
```

```
## [1] 0.8284695
```

The calculations and graphics prepared for the linear model can be easily reproduced for this alternative model. We have now 11% of the instances with an error less than 40,000 dollars. So far, the improvement observed does not support the log transformation (it can support it in other examples). The left part of the exhibit shows how the skewness is corrected by the log transformation.

```
res2 <- windsor$price - pred2
sd(res2)
```

```
## [1] 27238.2
```

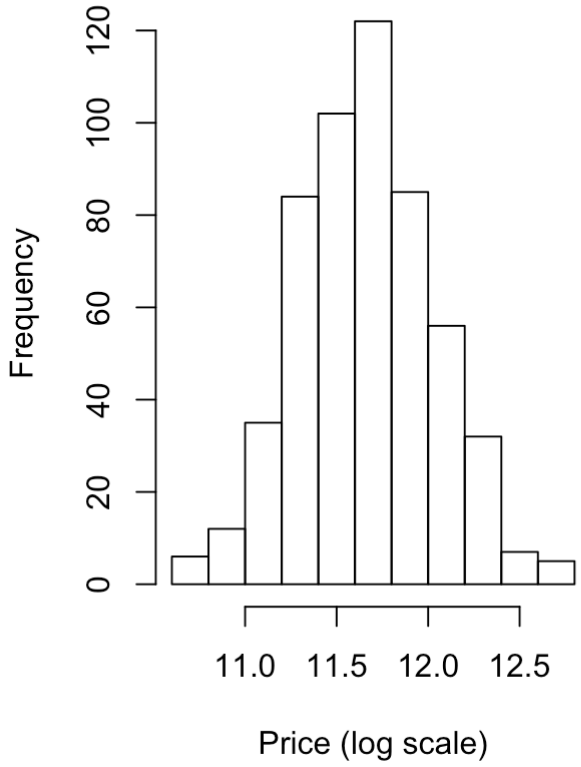
```
sd(res2)/sd(windsor$price)
```

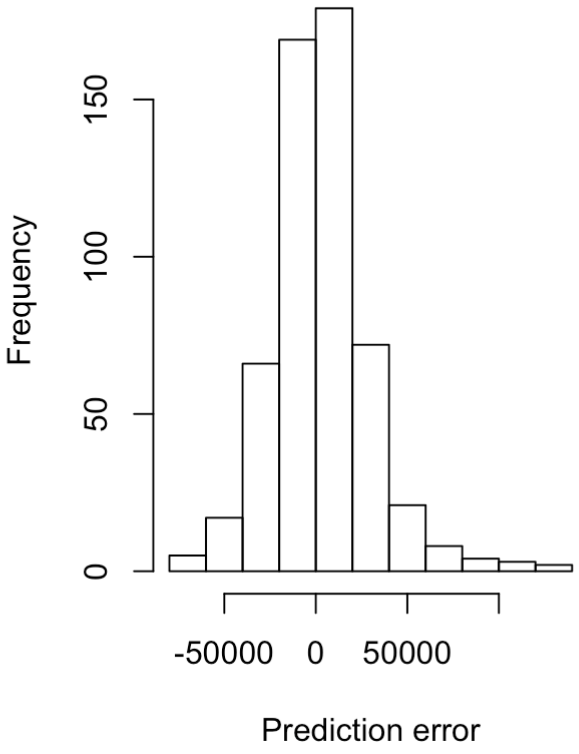
```
## [1] 0.5600587
```

```
mean(abs(res2)>40000)
```

```
## [1] 0.1098901
```

```
par(mfrow=c(1,2))
hist(log(windsor$price), main="", xlab="Price (log scale)")
hist(res2, main="", xlab="Prediction error")
```


Frequency
0
20
40
60
80
100
120
11.0
11.5
12.0
12.5
Price (log scale)


Frequency
0
50
100
150
-50000
0
50000
Prediction error