Analisi Colonial Broadcasting - Tesina

Luca Bajardi

22/4/2020

Carichiamo i dati leggendo il file csv e settiamo il seme del generatore pseudo-casuale così da avere i risultati sempre uguali

```
set.seed(1)
CBC = read.csv(file = "02 - Dati per caso Colonial Broadcasting.csv", header=T)
attach(CBC)
```

Il dataset che stiamo analizzando ha 88 osservazioni e 16 variabili:

```
dim(CBC)
```

```
## [1] 88 16
```

Le variabili si chiamano:

```
names(CBC)
```

```
##  [1] "network"     "fact"        "stars"       "month"       "day"
##  [6] "rating"      "prevratings" "competition" "bbs"         "abn"
## [11] "oct"         "dec"         "aprmay"      "mon"         "sun"
## [16] "march"
```

Vediamo l'inizio del dataset per vedere com'è fatto:

```
head(CBC)
```

```
##   network fact stars month day rating prevratings competition bbs abn oct
## 1     BBS    0     1     1   1   15.6        14.2        14.5   1   0   0
## 2     BBS    1     0     1   7   10.8        15.3        17.2   1   0   0
## 3     BBS    0     1     1   7   14.1        13.8        14.4   1   0   0
## 4     BBS    1     1     1   1   16.8        12.8        15.3   1   0   0
## 5     BBS    1     1     2   1   14.3        12.4        13.3   1   0   0
## 6     BBS    1     1     2   1   17.1        12.9        15.1   1   0   0
##   dec aprmay mon sun march
## 1   0      0   1   0     0
## 2   0      0   0   1     0
## 3   0      0   0   1     0
## 4   0      0   1   0     0
## 5   0      0   1   0     0
## 6   0      0   1   0     0
```

Vediamo che non ci sono elementi NA (Not Available) quindi posso usare tutte le osservazioni nel dataset:

```
sum(is.na(CBC$rating))
```

```
## [1] 0
```

Visto che il dataset è piccolo prendiamo un test set piccolo (30% del dataset totale)

```
train=sample(88,88*0.7)
```

Considero il dataset CBC, ma per la regressione prendo solo il sottoinsieme train

```
lm.fit=lm(rating~prevratings,data=CBC,subset=train)
```

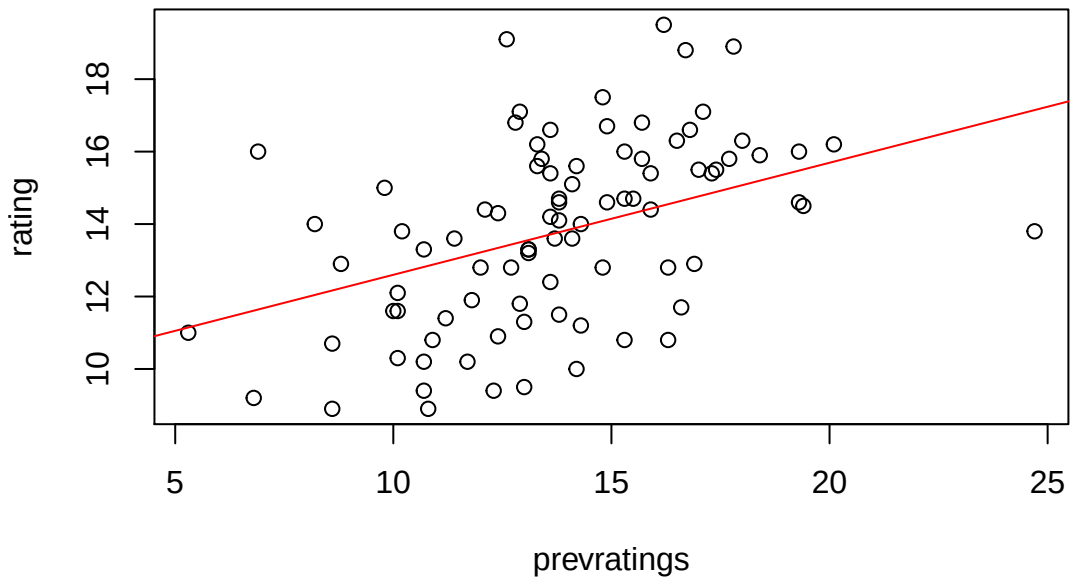
Dal summary vediamo che prevratings ha un coefficiente di regressione positivo. Questo fatto ha un senso logico perché il fatto che il programma precedente ha un rating più alto implica che le persone rimangono sul quel canale e non lo cambiano a fine del programma precedente.

```
summary(lm.fit)
```

```
##
## Call:
## lm(formula = rating ~ prevratings, data = CBC, subset = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.0270 -1.8332 -0.0239  1.7018  5.6967
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.50644    1.26880   7.492 3.96e-10 ***
## prevratings  0.30928    0.08843   3.498 0.000899 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.405 on 59 degrees of freedom
## Multiple R-squared:  0.1717, Adjusted R-squared:  0.1577
## F-statistic: 12.23 on 1 and 59 DF,  p-value: 0.0008991
```

Però questo non spiega tutto il modello infatti il valore R^2 è solo 0.1717. Possiamo vedere anche nel `plot` che non tutta la variabilità è rappresentata.

```
plot(prevratings,rating)
abline(lm.fit, col="red")
```



Possiamo calcolare il MSE (Mean Square Error):

$$MSE = \frac{\sum_{i \in test} (y_i - \widehat{y}_i)^2}{|test|}$$

```
mean((rating-predict(lm.fit,CBC))[-train]^2)
```

```
## [1] 3.489889
```

Il MSE di un solo modello non serve a niente, quindi dobbiamo calcolarlo anche di altri modelli:

```
potenze = c(1,2,3,4,5,6,10,16)
ma<-matrix(,nrow=3,ncol=length(potenze))
ma[1,]=potenze
for (i in 1:length(potenze)){
  pwr = potenze[i]
  lm.fit_n= lm(rating~poly(prevratings,pwr),data=CBC,subset=train)
  ma[2,i] = mean((rating-predict(lm.fit_n,CBC))[-train]^2)
  ma[3,i] = summary(lm.fit_n)$r.squared
}
ma
```

```
##         [,1]   [,2]   [,3]   [,4]   [,5]   [,6]    [,7]    [,8]
## [1,] 1.0000 2.0000 3.0000 4.0000 5.0000 6.0000 10.0000 16.0000
## [2,] 3.4899 3.4080 3.0002 2.9891 3.0434 3.0487  3.2968  2.9483
## [3,] 0.1717 0.1822 0.2216 0.2217 0.2341 0.2436  0.2687  0.3565
```

Possiamo vedere che aumentando il grado dei polinomi in funzione di `prevratings` non diminuisce il MSE, ma aumenta R^2, questo significa che stiamo facendo overfitting.

```
library(boot)
glm.fit=glm(rating~prevratings,data=CBC)
cv.err=cv.glm(CBC,glm.fit) #passa la struttura dati e il modello
names(cv.err)
```

```
## [1] "call"  "K"     "delta" "seed"
```

```
cv.err$K
```

```
## [1] 88
```

```
cv.err$delta
```

```
## [1] 5.169679 5.168069
```

```
#proviamo con polinomi con grado massimo diverso
cv.error=rep(0,5) #numeric(5)
for (i in 1:5){ #LOOCV su 5 modelli
  glm.fit=glm(rating~poly(prevratings,i),data=CBC)
  cv.error[i]=cv.glm(CBC,glm.fit)$delta[1]
}
cv.error
```

```
## [1]  5.169679  5.436229  5.102095  6.600530 15.841702
```

#campionamento senza reinserimento #slip come voglio 50-50,70-30 train #indici del training set ## R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

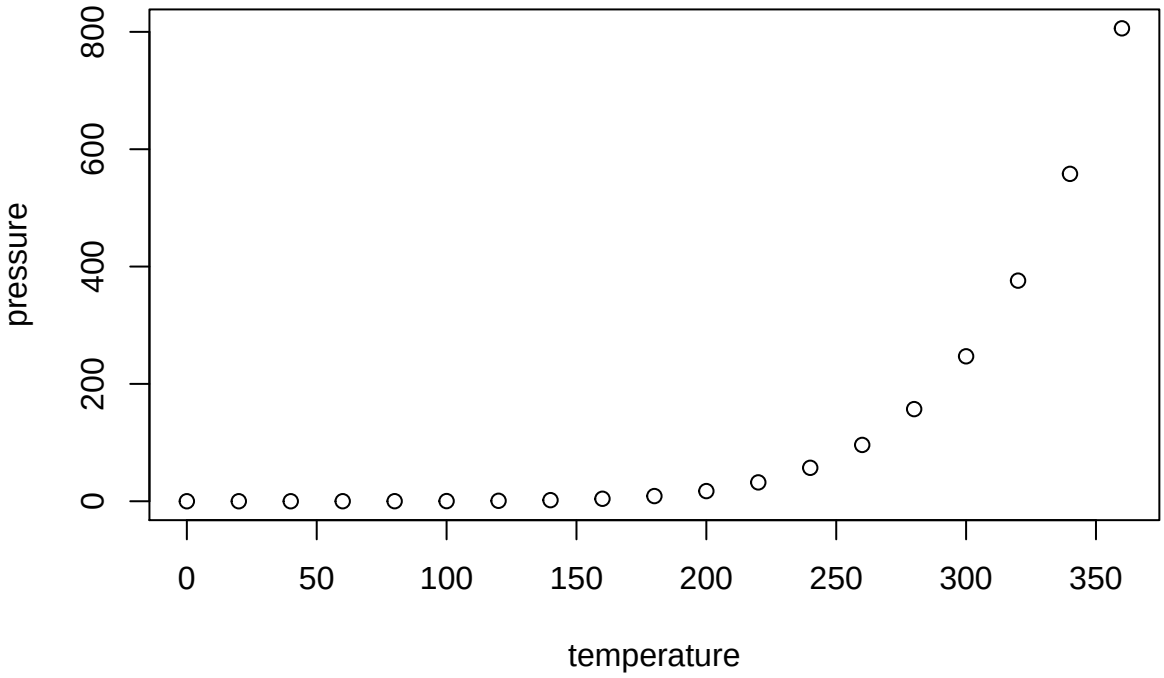
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00
```

Including Plots

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.