Analisi Colonial Broadcasting - Tesina

Luca Bajardi e Francesca Collini

15/5/2020

Caricamento dati, esplorazione e preprocessing

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

```
rm(list =ls())
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)
```

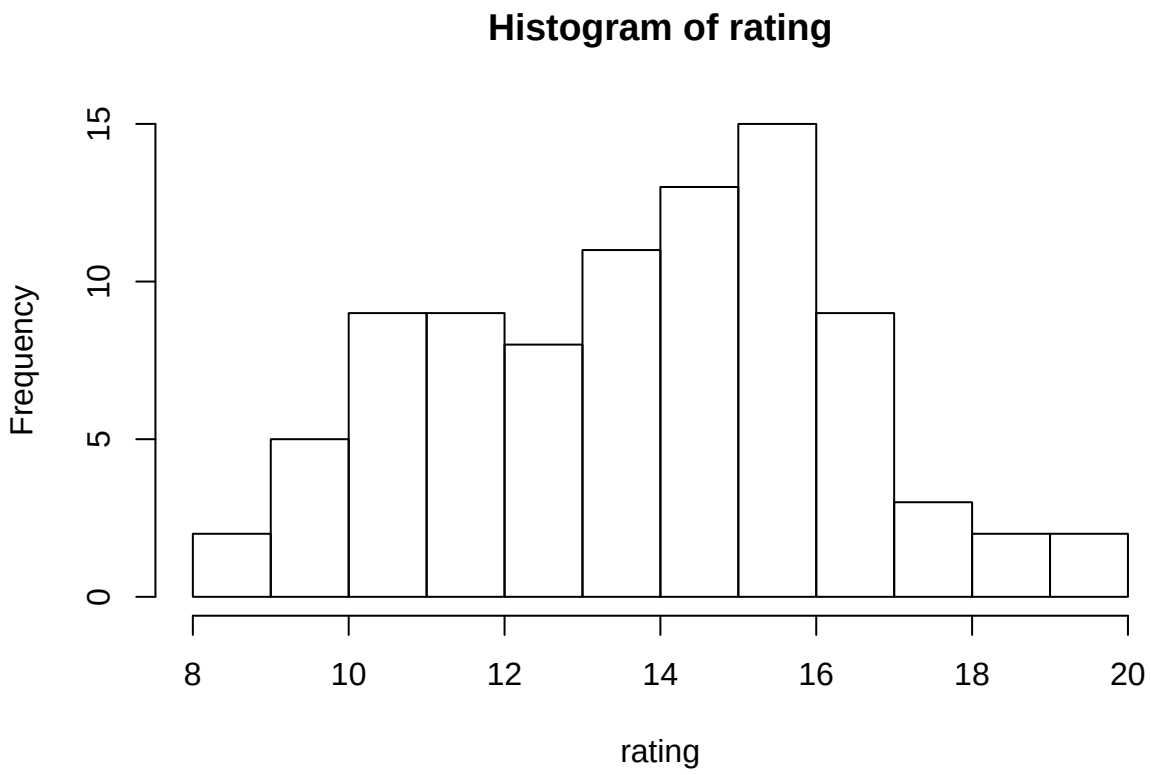
```
##   network fact stars month day rating prevratings competition bbs abn oct
## 1     BBS    0     1     1   1   15.6        14.2        14.5   1   0   0
## 2     BBS    1     0     1   7   10.8        15.3        17.2   1   0   0
## 3     BBS    0     1     1   7   14.1        13.8        14.4   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:

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

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

```r
hist(rating)
```



Modello lineare con una variabile

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

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

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

```r
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. Inoltre la variabile "prevratings", avendo un p-value sufficientemente piccolo, risulta significativo.

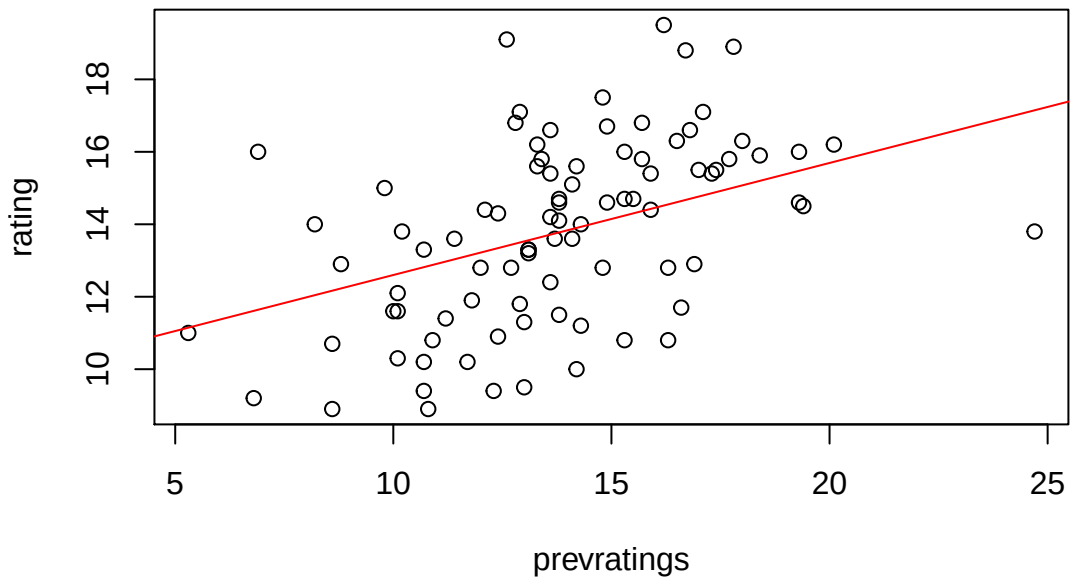
```r
summary(lm.fit)
```

```
##
## 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 il modello non spiega tutta la variabilità, infatti il valore R^2 è solo 0.1717. L'obiettivo è quello di massimizzare questo valore, in modo da poter predire il valore del rating con la maggiore accuratezza possibile. 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) andando a validare questo modello sul sottoinsieme di test:

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

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

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

Il MSE di un solo modello non è una misura molto indicativa perchè non abbiamo termini di paragoni, 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. Questi diversi esperimenti, probabilmente ci suggeriscono che utilizzare solo questa variabile potrebbe non essere sufficiente per prevedere il rating.

LOOCV

Utilizzando la libreria `boot` possiamo fare la Leave-one-out cross-validation così da vedere l'errore di previsione togliendo ogni volta un elemento dal training set.

```
library(boot)
glm.fit=glm(rating~prevratings,data=CBC) #costruisco il modello GLM
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]
```

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

Seguendo i ragionamenti fatti prima con un training set del 70%, possiamo eseguire il LOOCV su 5 modelli, ognuno dei quali con polinomi di grado massimo diverso.

```
cv.error=rep(0,5)
for (i in 1:5){
  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
```

Come abbiamo notato prima, anche eseguendo la LOOCV possiamo vedere che aumentando il grado massimo del polinomio l'errore non diminuisce, anzi aumenta all'aumentare del grado massimo. Per questo dobbiamo provare a prevedere il rating con l'utilizzo di un insieme variabili.

Subset selection

Con la funzione **regsubsets** possiamo, in modo automatico, selezionare quelle variabili che sono le migliori per predire la variabile risposta, quando viene fissato il numero totale di variabili da utilizzare nel modello. Il termine migliore si riferisce alle prestazioni, guardando la RSS. Utilizziamo il dataset eliminando la colonna che contiene la media dei rating di programmi mandati in onda dalle reti concorrenti. Infatti, questo valore è molto difficile da prevedere e potrebbe essere rischioso includerlo nel modello in quanto potrebbe portare a conclusioni sbagliate.

```
library(leaps)
CBC$network=factor(CBC$network,levels = c("CBC","ABN","BBS"))
CBC$day=as.factor(CBC$day)
CBC$month=as.factor(CBC$month)
CBCr=CBC[, 1:7]
regfit.full=regsubsets(rating~.,CBCr,nvmax=15)
summary(regfit.full)
```

```
## Subset selection object
## Call: regsubsets.formula(rating ~ ., CBCr, nvmax = 15)
## 15 Variables  (and intercept)
##             Forced in Forced out
## networkABN      FALSE      FALSE
## networkBBS      FALSE      FALSE
## fact            FALSE      FALSE
## stars           FALSE      FALSE
## month2          FALSE      FALSE
## month3          FALSE      FALSE
## month4          FALSE      FALSE
## month5          FALSE      FALSE
## month9          FALSE      FALSE
## month10         FALSE      FALSE
## month11         FALSE      FALSE
## month12         FALSE      FALSE
## day2            FALSE      FALSE
## day7            FALSE      FALSE
## prevratings     FALSE      FALSE
## 1 subsets of each size up to 15
## Selection Algorithm: exhaustive
##          networkABN networkBBS fact stars month2 month3 month4 month5
## 1  ( 1 ) " "        " "        " "  " "   " "    " "    " "    " "
## 2  ( 1 ) " "        " "        "*"  " "   " "    " "    " "    " "
## 3  ( 1 ) " "        " "        "*"  " "   " "    " "    " "    " "
## 4  ( 1 ) " "        " "        "*"  "*"   " "    " "    " "    " "
## 5  ( 1 ) " "        " "        "*"  "*"   " "    " "    " "    " "
## 6  ( 1 ) "*"        " "        "*"  " "   "*"    " "    " "    " "
## 7  ( 1 ) "*"        " "        "*"  " "   "*"    " "    " "    " "
## 8  ( 1 ) " "        "*"        "*"  "*"   " "    " "    "*"    " "
## 9  ( 1 ) "*"        " "        "*"  " "   "*"    " "    "*"    "*"
## 10  ( 1 ) "*"       " "        "*"  " "   "*"    " "    "*"    "*"
## 11  ( 1 ) "*"       "*"        "*"  "*"   "*"    " "    "*"    "*"
## 12  ( 1 ) "*"       "*"        "*"  "*"   "*"    " "    "*"    "*"
## 13  ( 1 ) "*"       "*"        "*"  "*"   "*"    "*"    "*"    "*"
## 14  ( 1 ) "*"       "*"        "*"  "*"   "*"    "*"    "*"    "*"
## 15  ( 1 ) "*"       "*"        "*"  "*"   "*"    "*"    "*"    "*"
##          month9 month10 month11 month12 day2 day7 prevratings
```

```
## 1  ( 1 )  " "    " "     " "     " "     " "  " "  "*"
## 2  ( 1 )  " "    " "     " "     " "     " "  " "  "*"
## 3  ( 1 )  " "    " "     " "     "*"     " "  " "  "*"
## 4  ( 1 )  " "    " "     " "     "*"     " "  " "  "*"
## 5  ( 1 )  " "    "*"     " "     "*"     " "  " "  "*"
## 6  ( 1 )  " "    " "     " "     "*"     "*"  " "  "*"
## 7  ( 1 )  " "    "*"     " "     "*"     "*"  " "  "*"
## 8  ( 1 )  " "    "*"     " "     "*"     "*"  " "  "*"
## 9  ( 1 )  " "    "*"     " "     "*"     "*"  " "  "*"
## 10  ( 1 ) " "    "*"     " "     "*"     "*"  "*"  "*"
## 11  ( 1 ) " "    "*"     " "     "*"     "*"  " "  "*"
## 12  ( 1 ) " "    "*"     " "     "*"     "*"  "*"  "*"
## 13  ( 1 ) " "    "*"     " "     "*"     "*"  "*"  "*"
## 14  ( 1 ) "*"    "*"     " "     "*"     "*"  "*"  "*"
## 15  ( 1 ) "*"    "*"     "*"     "*"     "*"  "*"  "*"
```

```
reg.summary=summary(regfit.full)
```

Osserviamo che, come ci potremmo aspettare, l'R^2 aumenta in maniera monotona all'aumentare delle variabili, ma questo non necessariamente ci fa concludere che il modello è migliore.

```
reg.summary$rsq
```

```
##  [1] 0.2343602 0.3151354 0.3720341 0.4084705 0.4308890 0.4534652 0.4681348
##  [8] 0.4825366 0.4933392 0.5025046 0.5080425 0.5122618 0.5135400 0.5137033
## [15] 0.5137192
```

Per questo sarebbe opportuno considerare altri indici, al fine di capire qual è il giusto equilibrio tra numero di variabili e proporzione della variabilità spiegata.

Poichè stiamo confrontando modelli con un numero diverso di variabili, guardare solamente all'R^2, potrebbe non fornirci un'informazione utile o quanto meno un'informazione piuttosto distorta. A questo proposito sarebbe utile utilizzare altri indici come R^2_{adj}, il C_p e il BIC:

- R^2_{adj} è definito dal seguente rapporto dove q rappresenta il numero di variabili ed n il numero di osservazioni:

$$R^2_{adj} = 1 - \frac{\frac{SSR}{(n-q-1)}}{\frac{TSS}{(n-1)}}$$

- C_p, definendo $p = q + 1$ e σ^2 come la stima dell'errore della varianza, il Mallow's C_p, si calcola come:

$$C_p = \frac{1}{n}(SSR + 2p\sigma^2)$$

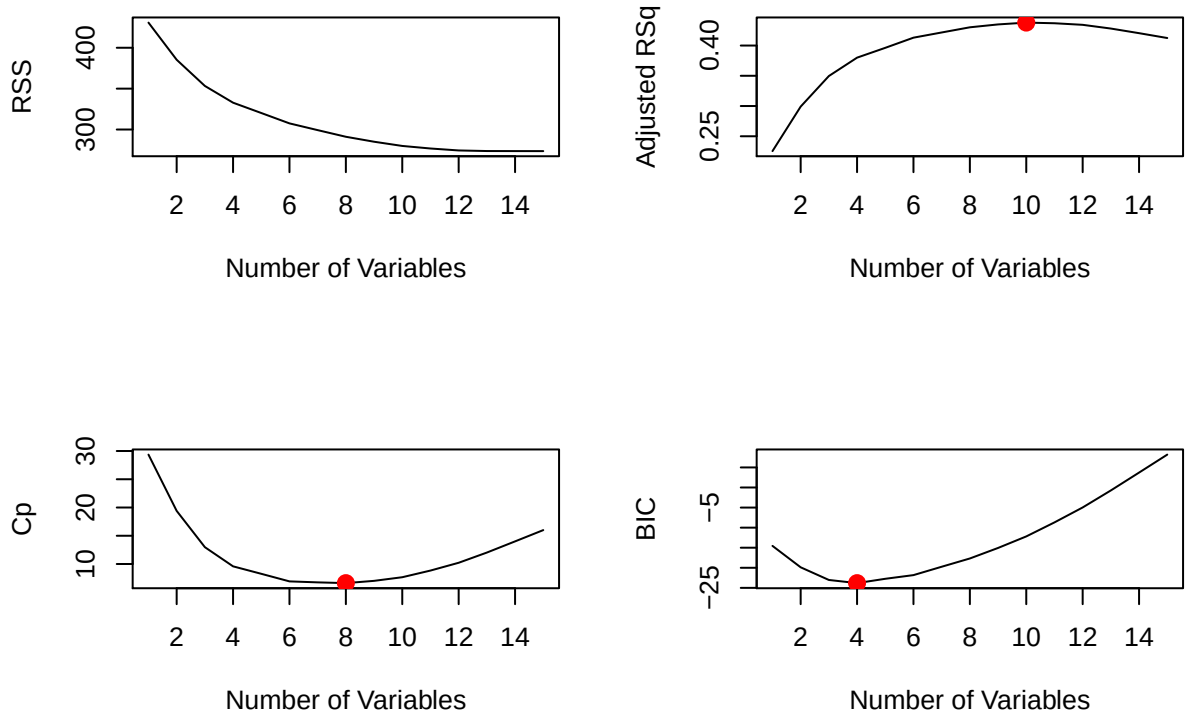
- BIC (Bayesian Information Criterion) è definito da:

$$BIC = \frac{1}{n}(SSR + log(n)p\sigma^2)$$

  In questo caso l'uso di più variabili viene penalizzato maggiormente rispetto all'utilizzao del C_p perchè in genere il logaritmo del numero delle variabili tende ad essere più grande di due. In generale quindi, utilizzando questo parametro, otterremo un modello più conservativo.

```
par(mfrow=c(2,2))
plot(reg.summary$rss,xlab="Number of Variables",ylab="RSS",type="l")
plot(reg.summary$adjr2,xlab="Number of Variables",ylab="Adjusted RSq",type="l")
t=which.max(reg.summary$adjr2)
```

```
points(t,reg.summary$adjr2[t], col="red",cex=2,pch=20)
plot(reg.summary$cp,xlab="Number of Variables",ylab="Cp",type='l')
q=which.min(reg.summary$cp)
points(q,reg.summary$cp[q],col="red",cex=2,pch=20)
u=which.min(reg.summary$bic)
plot(reg.summary$bic,xlab="Number of Variables",ylab="BIC",type='l')
points(u,reg.summary$bic[u],col="red",cex=2,pch=20)
```




```
# Choosing Among Models using the cross-validation approach
set.seed(1)
train=sample(c(TRUE,FALSE), nrow(CBC),rep=TRUE)
test=(!train) #complemento
regfit.best=regsubsets(rating~.,data=CBCr[train,],nvmax=15)
test.mat=model.matrix(rating~.,data=CBCr[test,])
#estrae dal dateset è la matrice X, potrebbe servire per i calcoli
val.errors=rep(NA,12)
for(i in 1:12){
  coefi=coef(regfit.best,id=i)
  pred=test.mat[,names(coefi)]%*%coefi
  #prodotto righe per colonne
  val.errors[i]=mean((CBCr$rating[test]-pred)^2)
}
val.errors
```

```
##  [1] 4.511783 6.196545 5.177542 5.789412 5.791497 6.079886 5.595607
##  [8] 5.082818 5.275389 5.086785 5.134484 6.021948
```

```r
which.min(val.errors)
```

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

```r
coef(regfit.best,1)
```

```
## (Intercept) prevratings
##   8.6498631   0.3609965
```

```r
regfit.fwd=regsubsets(rating~.,data=CBCr,nvmax=15,method="forward")
summary(regfit.fwd)
```

```
## Subset selection object
## Call: regsubsets.formula(rating ~ ., data = CBCr, nvmax = 15, method = "forward")
## 15 Variables  (and intercept)
##             Forced in Forced out
## networkABN      FALSE      FALSE
## networkBBS      FALSE      FALSE
## fact            FALSE      FALSE
## stars           FALSE      FALSE
## month2          FALSE      FALSE
## month3          FALSE      FALSE
## month4          FALSE      FALSE
## month5          FALSE      FALSE
## month9          FALSE      FALSE
## month10         FALSE      FALSE
## month11         FALSE      FALSE
## month12         FALSE      FALSE
## day2            FALSE      FALSE
## day7            FALSE      FALSE
## prevratings     FALSE      FALSE
## 1 subsets of each size up to 15
## Selection Algorithm: forward
##           networkABN networkBBS fact stars month2 month3 month4 month5
## 1  ( 1 )  " "        " "        " "  " "   " "    " "    " "    " "
## 2  ( 1 )  " "        " "        "*"  " "   " "    " "    " "    " "
## 3  ( 1 )  " "        " "        "*"  " "   " "    " "    " "    " "
## 4  ( 1 )  " "        " "        "*"  "*"   " "    " "    " "    " "
## 5  ( 1 )  " "        " "        "*"  "*"   " "    " "    " "    " "
## 6  ( 1 )  " "        "*"        "*"  "*"   " "    " "    " "    " "
## 7  ( 1 )  " "        "*"        "*"  "*"   " "    " "    " "    " "
## 8  ( 1 )  " "        "*"        "*"  "*"   " "    " "    "*"    " "
## 9  ( 1 )  "*"        "*"        "*"  "*"   " "    " "    "*"    " "
## 10  ( 1 ) "*"        "*"        "*"  "*"   "*"    " "    "*"    " "
## 11  ( 1 ) "*"        "*"        "*"  "*"   "*"    " "    "*"    "*"
## 12  ( 1 ) "*"        "*"        "*"  "*"   "*"    " "    "*"    "*"
## 13  ( 1 ) "*"        "*"        "*"  "*"   "*"    "*"    "*"    "*"
## 14  ( 1 ) "*"        "*"        "*"  "*"   "*"    "*"    "*"    "*"
## 15  ( 1 ) "*"        "*"        "*"  "*"   "*"    "*"    "*"    "*"
##           month9 month10 month11 month12 day2 day7 prevratings
## 1  ( 1 )  " "    " "     " "     " "     " "  " "  "*"
## 2  ( 1 )  " "    " "     " "     " "     " "  " "  "*"
## 3  ( 1 )  " "    " "     " "     "*"     " "  " "  "*"
## 4  ( 1 )  " "    " "     " "     "*"     " "  " "  "*"
## 5  ( 1 )  " "    "*"     " "     "*"     " "  " "  "*"
```

```
## 6  ( 1 )  " "    "*"     " "     "*"     " "  " "  "*"
## 7  ( 1 )  " "    "*"     " "     "*"     "*"  " "  "*"
## 8  ( 1 )  " "    "*"     " "     "*"     "*"  " "  "*"
## 9  ( 1 )  " "    "*"     " "     "*"     "*"  " "  "*"
## 10  ( 1 ) " "    "*"     " "     "*"     "*"  " "  "*"
## 11  ( 1 ) " "    "*"     " "     "*"     "*"  " "  "*"
## 12  ( 1 ) " "    "*"     " "     "*"     "*"  "*"  "*"
## 13  ( 1 ) " "    "*"     " "     "*"     "*"  "*"  "*"
## 14  ( 1 ) "*"    "*"     " "     "*"     "*"  "*"  "*"
## 15  ( 1 ) "*"    "*"     "*"     "*"     "*"  "*"  "*"
```

```
regfit.bwd=regsubsets(rating~.,data=CBCr,nvmax=15,method="backward")
summary(regfit.bwd)
```

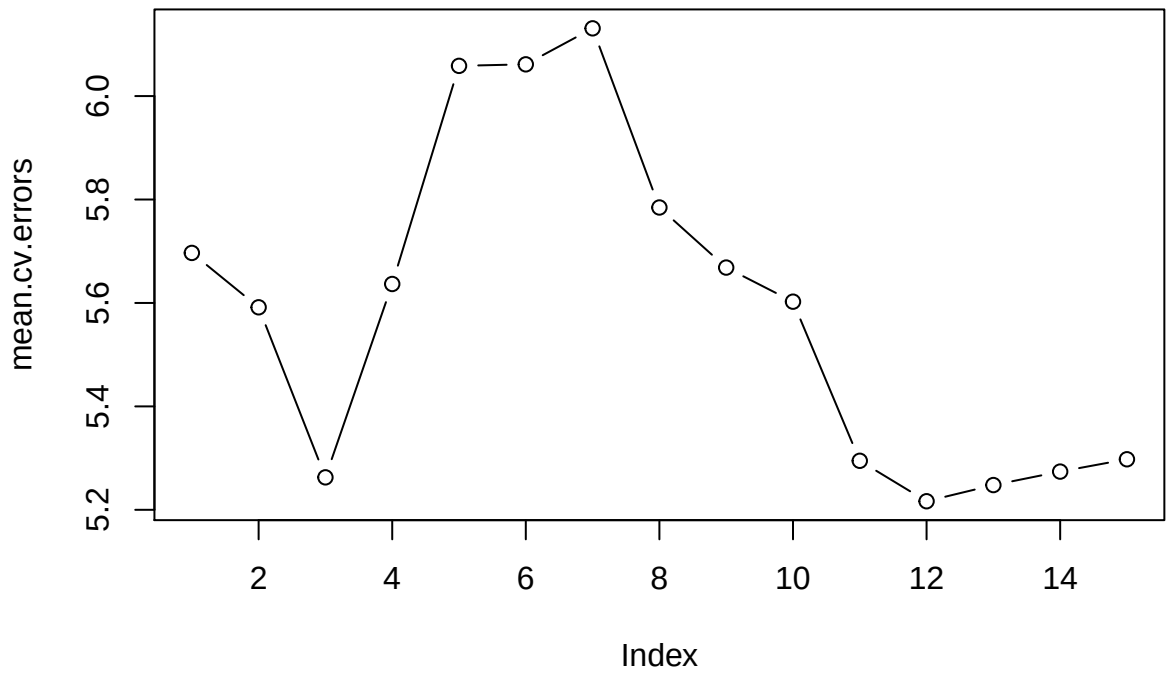
```
## Subset selection object
## Call: regsubsets.formula(rating ~ ., data = CBCr, nvmax = 15, method = "backward")
## 15 Variables  (and intercept)
##             Forced in Forced out
## networkABN      FALSE      FALSE
## networkBBS      FALSE      FALSE
## fact            FALSE      FALSE
## stars           FALSE      FALSE
## month2          FALSE      FALSE
## month3          FALSE      FALSE
## month4          FALSE      FALSE
## month5          FALSE      FALSE
## month9          FALSE      FALSE
## month10         FALSE      FALSE
## month11         FALSE      FALSE
## month12         FALSE      FALSE
## day2            FALSE      FALSE
## day7            FALSE      FALSE
## prevratings     FALSE      FALSE
## 1 subsets of each size up to 15
## Selection Algorithm: backward
##           networkABN networkBBS fact stars month2 month3 month4 month5
## 1  ( 1 )  " "        " "        " "  " "   " "    " "    " "    " "
## 2  ( 1 )  " "        " "        "*"  " "   " "    " "    " "    " "
## 3  ( 1 )  " "        " "        "*"  " "   " "    " "    " "    " "
## 4  ( 1 )  " "        " "        "*"  "*"   " "    " "    " "    " "
## 5  ( 1 )  " "        " "        "*"  "*"   " "    " "    " "    " "
## 6  ( 1 )  " "        "*"        "*"  "*"   " "    " "    " "    " "
## 7  ( 1 )  " "        "*"        "*"  "*"   " "    " "    " "    " "
## 8  ( 1 )  " "        "*"        "*"  "*"   " "    " "    "*"    " "
## 9  ( 1 )  "*"        "*"        "*"  "*"   " "    " "    "*"    " "
## 10  ( 1 ) "*"        "*"        "*"  "*"   "*"    " "    "*"    " "
## 11  ( 1 ) "*"        "*"        "*"  "*"   "*"    " "    "*"    "*"
## 12  ( 1 ) "*"        "*"        "*"  "*"   "*"    " "    "*"    "*"
## 13  ( 1 ) "*"        "*"        "*"  "*"   "*"    "*"    "*"    "*"
## 14  ( 1 ) "*"        "*"        "*"  "*"   "*"    "*"    "*"    "*"
## 15  ( 1 ) "*"        "*"        "*"  "*"   "*"    "*"    "*"    "*"
##           month9 month10 month11 month12 day2 day7 prevratings
## 1  ( 1 )  " "    " "     " "     " "     " "  " "  "*"
## 2  ( 1 )  " "    " "     " "     " "     " "  " "  "*"
## 3  ( 1 )  " "    " "     " "     "*"     " "  " "  "*"
```

```
## 4  ( 1 )  " "    " "     " "     "*"     " "  " "  "*"
## 5  ( 1 )  " "    "*"     " "     "*"     " "  " "  "*"
## 6  ( 1 )  " "    "*"     " "     "*"     " "  " "  "*"
## 7  ( 1 )  " "    "*"     " "     "*"     "*"  " "  "*"
## 8  ( 1 )  " "    "*"     " "     "*"     "*"  " "  "*"
## 9  ( 1 )  " "    "*"     " "     "*"     "*"  " "  "*"
## 10  ( 1 ) " "    "*"     " "     "*"     "*"  " "  "*"
## 11  ( 1 ) " "    "*"     " "     "*"     "*"  " "  "*"
## 12  ( 1 ) " "    "*"     " "     "*"     "*"  "*"  "*"
## 13  ( 1 ) " "    "*"     " "     "*"     "*"  "*"  "*"
## 14  ( 1 ) "*"    "*"     " "     "*"     "*"  "*"  "*"
## 15  ( 1 ) "*"    "*"     "*"     "*"     "*"  "*"  "*"
```

```
k=10
predict.regsubsets=function(object,newdata,id,...){
  form=as.formula(object$call[[2]])
  mat=model.matrix(form,newdata)
  coefi=coef(object,id=id)
  xvars=names(coefi)
  mat[,xvars]%*%coefi
}
set.seed(1)
folds=sample(1:k,nrow(CBC),replace=TRUE)
cv.errors=matrix(NA,k,15, dimnames=list(NULL, paste(1:15)))
for(j in 1:k){
  best.fit=regsubsets(rating~.,data=CBCr[folds!=j,],nvmax=15)
  for(i in 1:15){
    pred=predict(best.fit,CBCr[folds==j,],id=i)
    cv.errors[j,i]=mean( (CBC$rating[folds==j]-pred)^2)
  }
}
mean.cv.errors=apply(cv.errors,2,mean)
mean.cv.errors
```

```
##        1        2        3        4        5        6        7        8
## 5.696734 5.591594 5.262762 5.636649 6.058448 6.061406 6.130980 5.784581
##        9       10       11       12       13       14       15
## 5.668436 5.602479 5.294795 5.216580 5.247858 5.273894 5.297743
```

```
min=which.min(mean.cv.errors)
par(mfrow=c(1,1))
plot(mean.cv.errors,type='b')
```




```
reg.best=regsubsets(rating~.,data=CBC, nvmax=15)
```

```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
## force.in = force.in, : 8 linear dependencies found
```

```
coef(reg.best,min)
```

```
## (Intercept)  networkABN  networkBBS        fact       stars      month2
##  12.7072567   1.2367143  -0.9051037   1.7163747   0.5873807   0.9417679
## prevratings competition         oct         dec      aprmay         mon
##   0.1889170  -0.2970816  -1.3866385   1.5339747  -1.2911335   2.6244930
##         sun
##   1.5842134
```

Regressione Ridge e Lasso

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 3.0-2
```

```
#CBC = read.csv(file = "02 - Dati per caso Colonial Broadcasting.csv", header=T)
x=model.matrix(rating~.,CBC)[,-1] #tolgo l'intercetta
#perchè Beta_0 non va penalizzato
y=CBC$rating
grid=10^seq(10,-2,length=100)
```

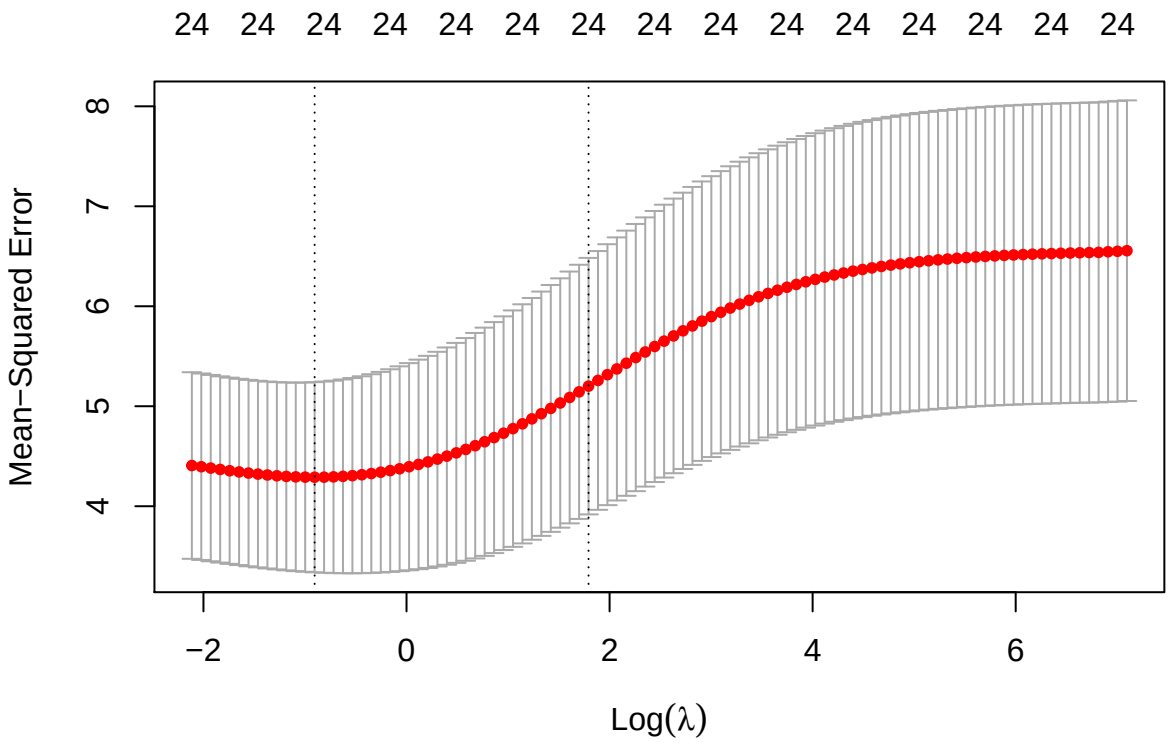
La funzione `glmnet` implementa sia i due casi estremi di regressione ridge e lasso, sia regressioni intermedie a seconda del valore di α che fissiamo.

```
ridge.mod=glmnet(x,y,alpha=0,lambda=grid)
dim(coef(ridge.mod))
```

```
## [1]  25 100
```

In questo modo abbiamo ottenuto 100 diversi modelli di regressione, uno per ogni valore di λ. Invece di utilizzare tutti i dati per la costruzione del modello, potremmo dividerli in un insieme di training e uno di test e utilizzando la cross-validation scegliere il miglior valore di λ

```
set.seed(1)
train=sample(1:nrow(x), nrow(x)/2)
test=(-train)
y.test=y[test]
cv.out=cv.glmnet(x[train,],y[train],alpha=0)
plot(cv.out)
```




```
#MSE in funzione di lamda per trovare quello migliore
#c'è una certa variabilità nella stima
bestlam=cv.out$lambda.min
bestlam
```

```
## [1] 0.404555
```

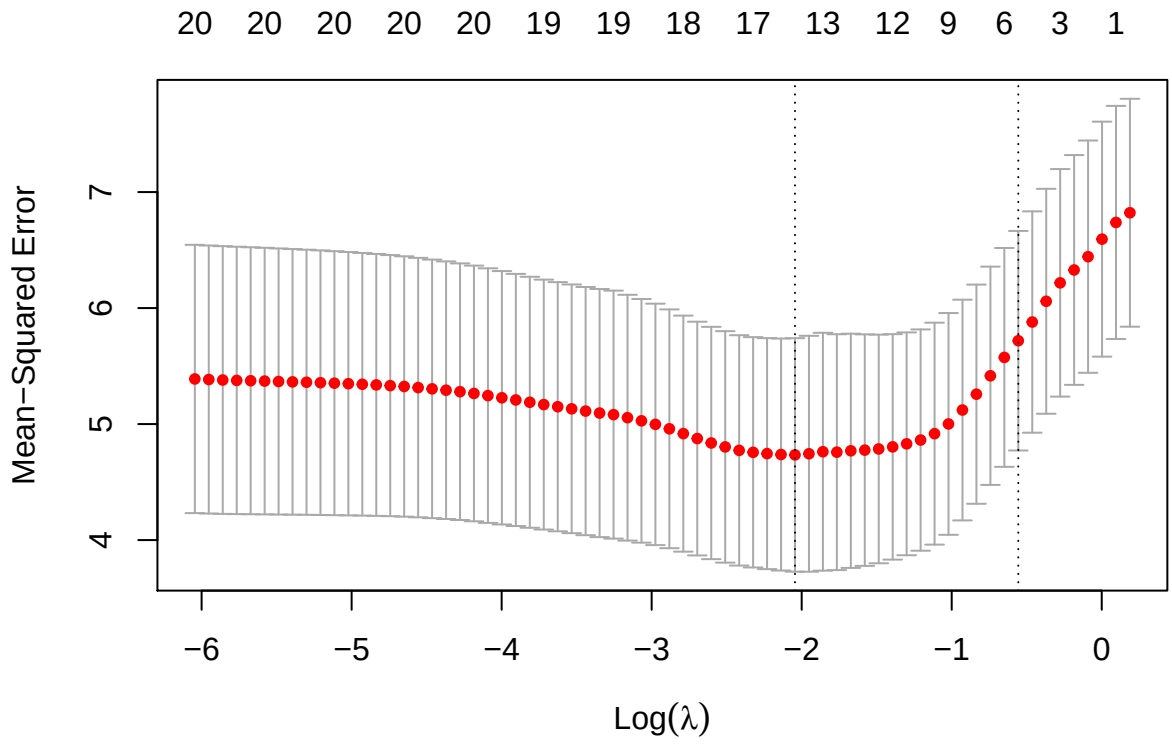
```
ridge.pred=predict(ridge.mod,s=bestlam,newx=x[test,])
mean((ridge.pred-y.test)^2)
```

```
## [1] 3.088482
```

```
#dopo che ho capito che questo è il modello migliore, vado a
#prevederlo con tutti i dati
out=glmnet(x,y,alpha=0)
predict(out,type="coefficients",s=bestlam)[1:17,]
```

```
## (Intercept)  networkABN  networkBBS        fact       stars      month2
## 13.73458085  0.48155147 -0.45167696  1.41260412  0.46550463  0.92237697
##      month3      month4      month5      month9     month10     month11
##  0.06744899 -0.75852848 -0.23789144 -0.19633645 -0.61594226  0.10921119
##     month12        day2        day7 prevratings competition
##  0.70326774 -1.17310659  0.05103956  0.18185751 -0.23440827
```

```
lasso.mod=glmnet(x[train,],y[train],alpha=1,lambda=grid)
set.seed(1)
cv.out=cv.glmnet(x[train,],y[train],alpha=1)
plot(cv.out)
```




```
bestlam=cv.out$lambda.min
bestlam
```

```
## [1] 0.129428
```

```
lasso.pred=predict(lasso.mod,s=bestlam,newx=x[test,])
mean((lasso.pred-y.test)^2)
```

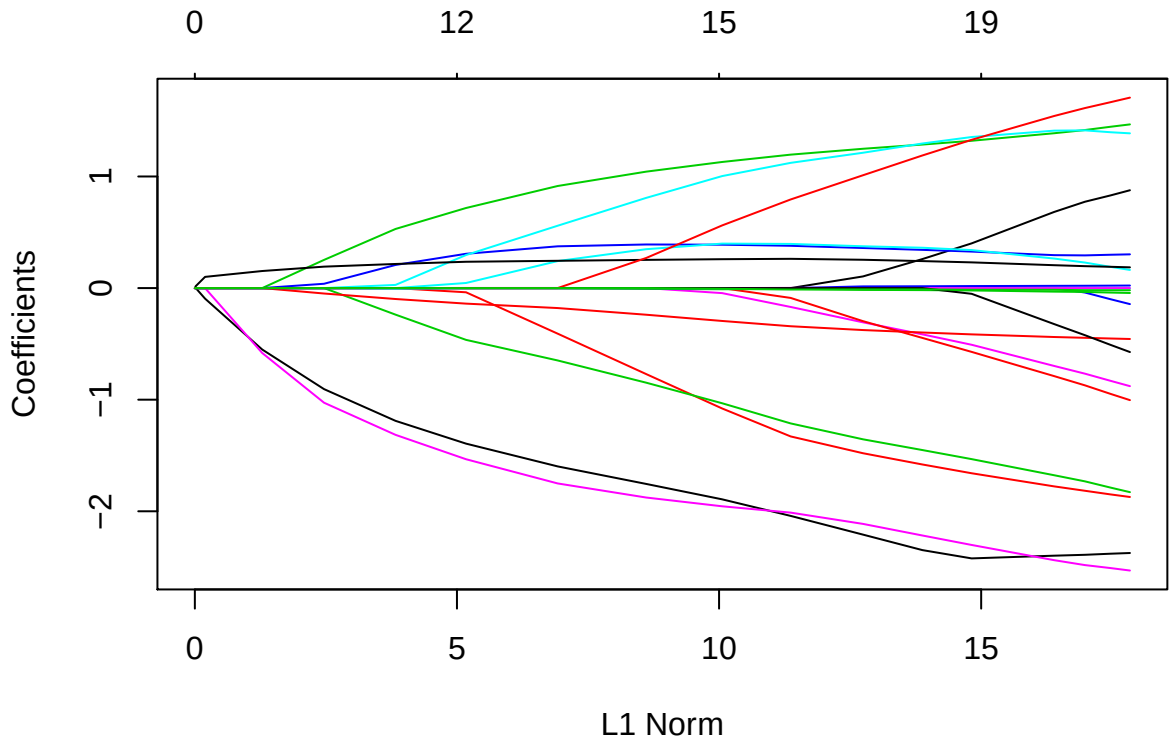
```
## [1] 4.332984
```

```
out=glmnet(x,y,alpha=1,lambda=grid)
lasso.coef=predict(out,type="coefficients",s=bestlam)[1:17,]
lasso.coef
```

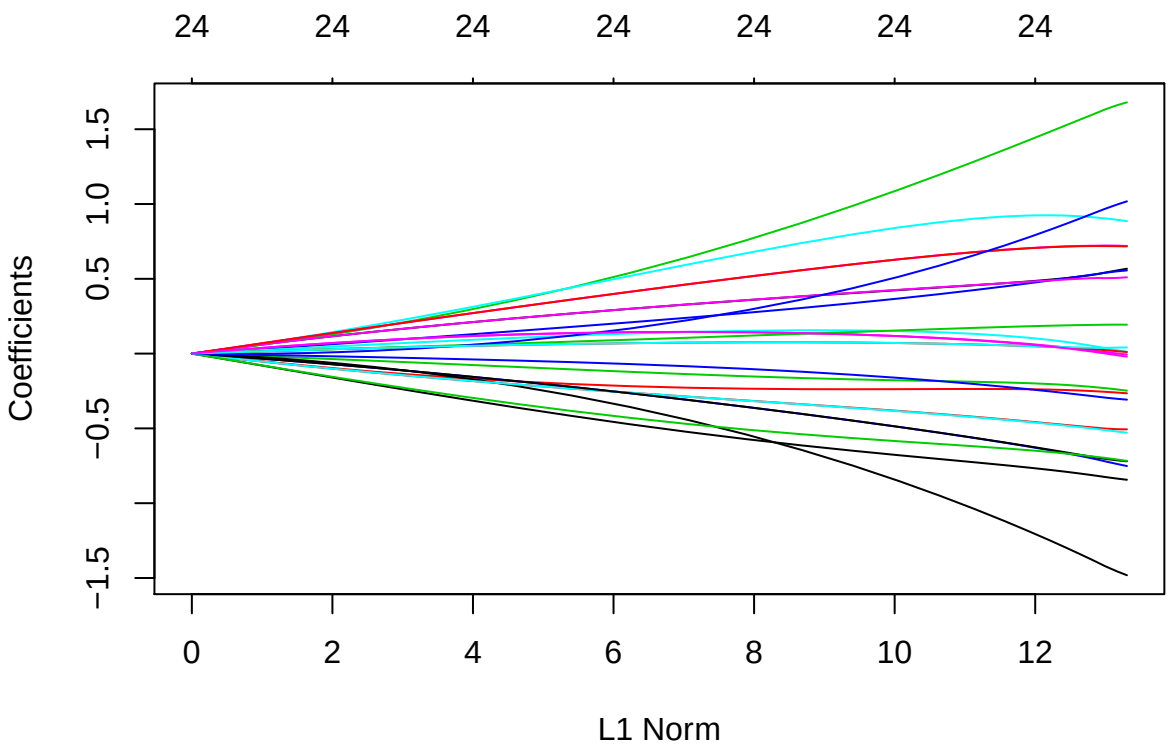
```
## (Intercept)  networkABN  networkBBS        fact       stars      month2
##  12.3943302   0.0000000  -0.6307602   1.4007788   0.4513570   0.6875773
##      month3      month4      month5      month9     month10     month11
##   0.0000000  -0.4506918   0.0000000   0.0000000  -0.9655532   0.0000000
##     month12        day2        day7 prevratings competition
##   1.2076515  -0.6843400   0.0000000   0.2680403  -0.1872562
```

```
plot(lasso.mod)
```

```
## Warning in regularize.values(x, y, ties, missing(ties)): collapsing to
## unique 'x' values
```




```
plot(ridge.mod)
```


24 24 24 24 24 24 24
Coefficients
1.5
1.0
0.5
−0.5
−1.5
0 2 4 6 8 10 12
L1 Norm