Prostate Cancer: Ridge and Lasso

Carpineta M., Ciarelli R., D'Onofrio M., Marusco L.

Contents

Il dataset in questione riporta alcune informazioni circa il tumore alla prostata analizzato in 96 pazienti.

Per ogni argomento trattato nelle analisi a seguire vengono riportati gli obbiettivi, i principali metodi utilizzati, i motivi e le conclusioni tratte.

Quindi carichiamo il dataset.

```
prostate.data <- read.delim("prostate.csv")
prostate <- prostate.data[-c(1,11)]
#eliminazione della prima e undicesima variabile

attach(prostate)
```

Con il comando `str` andiamo a visionare la natura del nostro dataset.

```
str(prostate)
```

```
## 'data.frame':    97 obs. of  9 variables:
##  $ lcavol : num  -0.58 -0.994 -0.511 -1.204 0.751 ...
##  $ lweight: num  2.77 3.32 2.69 3.28 3.43 ...
##  $ age    : int  50 58 74 58 62 50 64 58 47 63 ...
##  $ lbph   : num  -1.39 -1.39 -1.39 -1.39 -1.39 ...
##  $ svi    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ lcp    : num  -1.39 -1.39 -1.39 -1.39 -1.39 ...
```

```
##  $ gleason: int  6 6 7 6 6 6 6 6 6 6 ...
##  $ pgg45  : int  0 0 20 0 0 0 0 0 0 0 ...
##  $ lpsa   : num  -0.431 -0.163 -0.163 -0.163 0.372 ...
```

Si riscontrano 97 unità statistiche e 9 variabili numeriche.

Di seguito una spiegazione dettagliata delle nostre variabili:

- **lcavol:** Volume del cancro, espresso in logaritmi;
- **lweight:** Peso del cancro, espresso in logaritmi;
- **age:** Età del paziente;
- **lbph:** Iperplasia prostatica benigna, espressa in logaritmi;
- **svi:** Invasione della vescicola seminale;
- **lcp:** Penetrazione capsulare, espresso in logaritmi;
- **gleason:** Il sistema di classificazione di Gleason è utilizzato per aiutare a valutare la prognosi degli uomini con cancro alla prostata utilizzando campioni da una biopsia prostatica. Un punteggio di Gleason viene assegnato al cancro della prostata in base al suo aspetto microscopico. I tumori con un punteggio di Gleason più alto sono più aggressivi e hanno una prognosi peggiore.
- **pgg45:** Percentuale di Gleason; questa variabile misura la percentuale di punteggi di Gleason 4 o 5 che sono stati registrati nelle loro visite prima del loro attuale punteggio finale di Gleason, memorizzato in *gleason*; un punteggio di Gleason più alto è peggiore, quindi pgg45 ci dice qualcosa sulla gravità del loro cancro nel passato.
- **lpsa:** Antigene prostatico specifico, espresso in logaritmi;

Notiamo subito che la variabile **svi** presenta solo valori pari a 0 e 1, quindi si è deciso di convertirla in una dummy.

```
prostate$svi <- as.factor(prostate$svi)
```

Nell'analisi esplorativa a seguire verrà omessa questa variabile in quanto non più di tipo quantitativa, e quindi non è possibile rappresentarle tramite grafici come *boxplot*, *istogramma*, ecc...

Analisi Esplorativa

Come prima cosa si procede ad un'analisi esplorativa delle variabili, così da comprendere meglio la natura del nostro dataset.

La funzione `summary()` restituisce alcune statistiche descrittive, tra cui:

- il valore minimo e massimo tra tutte le unità statistiche per ciascuna variabile;
- la media calcolata tra le varie unità statistiche su ciascuna variabile;
- il primo, secondo (mediana) e terzo quantile per ogni variabile.

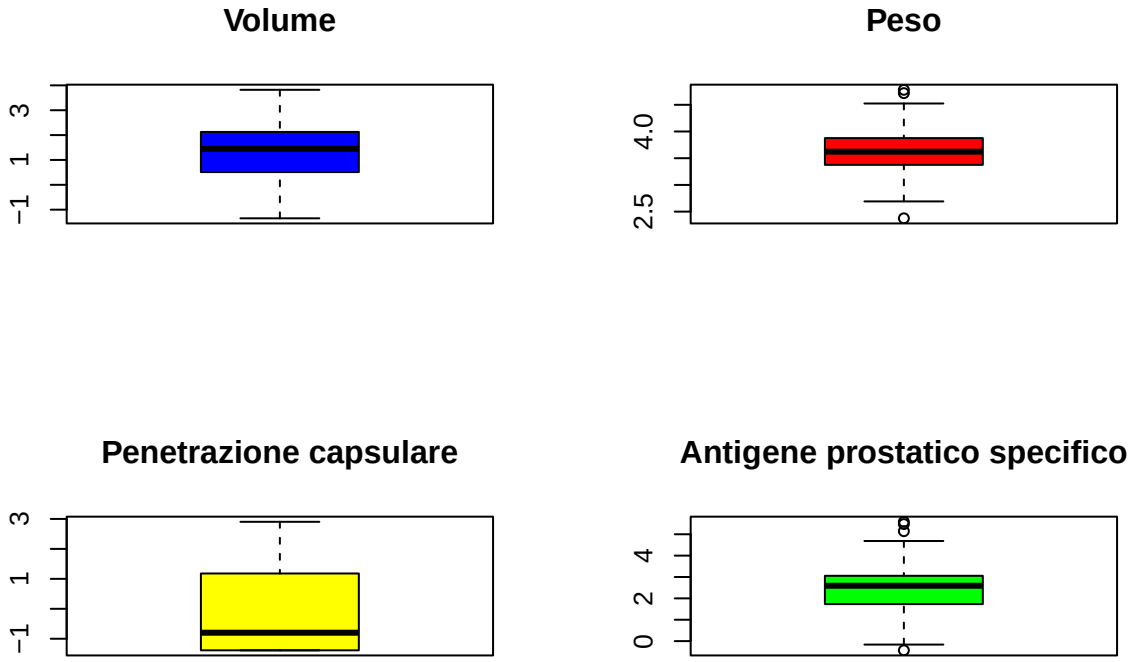
```
summary(prostate[-c(5)])
```

```
##      lcavol           lweight           age              lbph        
##  Min.   :-1.3471   Min.   :2.375   Min.   :41.00   Min.   :-1.3863  
##  1st Qu.: 0.5128   1st Qu.:3.376   1st Qu.:60.00   1st Qu.:-1.3863  
##  Median : 1.4469   Median :3.623   Median :65.00   Median : 0.3001  
##  Mean   : 1.3500   Mean   :3.629   Mean   :63.87   Mean   : 0.1004  
##  3rd Qu.: 2.1270   3rd Qu.:3.876   3rd Qu.:68.00   3rd Qu.: 1.5581  
##  Max.   : 3.8210   Max.   :4.780   Max.   :79.00   Max.   : 2.3263  
##       lcp              gleason          pgg45            lpsa        
##  Min.   :-1.3863   Min.   :6.000   Min.   :  0.00   Min.   :-0.4308  
##  1st Qu.:-1.3863   1st Qu.:6.000   1st Qu.:  0.00   1st Qu.: 1.7317  
##  Median :-0.7985   Median :7.000   Median : 15.00   Median : 2.5915  
##  Mean   :-0.1794   Mean   :6.753   Mean   : 24.38   Mean   : 2.4784  
##  3rd Qu.: 1.1787   3rd Qu.:7.000   3rd Qu.: 40.00   3rd Qu.: 3.0564  
##  Max.   : 2.9042   Max.   :9.000   Max.   :100.00   Max.   : 5.5829
```

Boxplot

Si possono vedere queste informazioni (esclusa la media) sotto forma di grafico grazie al `boxplot`.

```
par(mfrow=c(2,2))
boxplot(prostate$lcavol, col = "blue", main = "Volume")
boxplot(prostate$lweight, col = "red", main = "Peso")
boxplot(prostate$lcp, col = "yellow", main = "Penetrazione capsulare")
boxplot(prostate$lpsa, col = "green", main = "Antigene prostatico specifico")
```

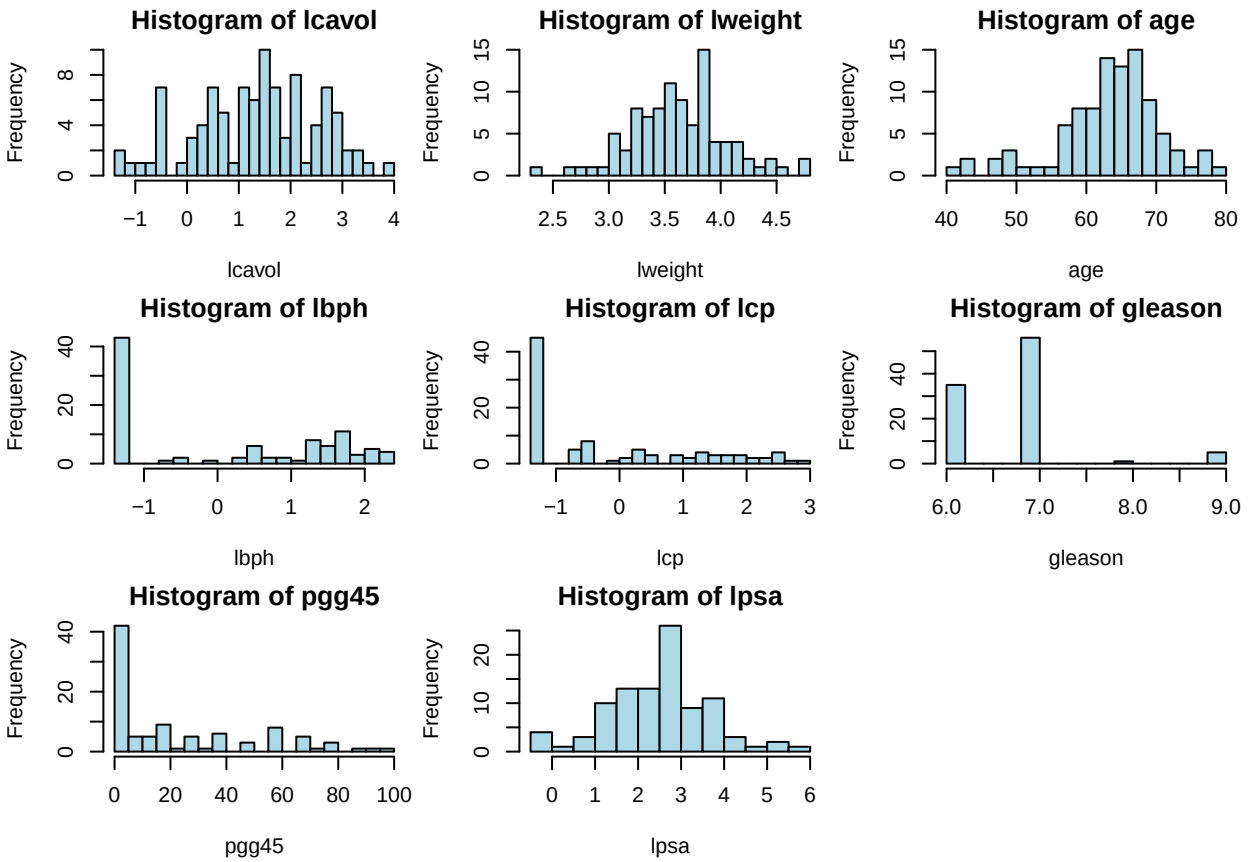


Istogrammi

Di seguito si sono voluti rappresentare degli istogrammi sulle variabili di nostro interesse.

Nel singoli grafici verranno riportate il numero di unità statistiche (i pazienti) sull'asse delle ordinate, e i livelli di ciascuna variabile sull'asse delle ascisse.

```
par(mfrow=c(3,3), mar=c(4,4,2,0.5))
for (j in 1:ncol(prostate[-c(5)])) {
  hist(prostate[-c(5)][,j], xlab=colnames(prostate[-c(5)])[j],
       main=paste("Histogram of", colnames(prostate[-c(5)])[j]),
       col="lightblue", breaks=20)
}
```



Correlazioni

Cona la funzione `cor()` si crea una matrice di correlazione, passando come argomento la matrice dei dati di interesse; quindi la si salva in una variabile così da richiamarla per studi futuri.

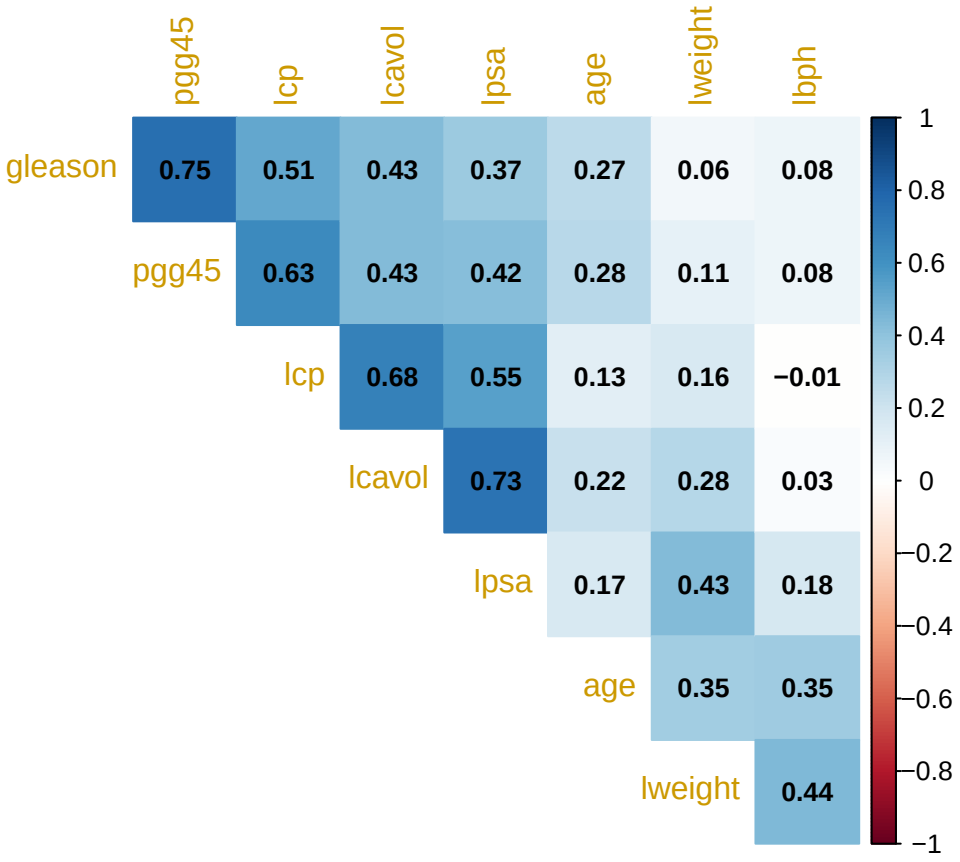
Come già detto in precedenza si escluderanno dalla matrice di correlazione la variabile dummy *svi*. Ne faremo uno studio a parte in seguito.

```
mat.cor <- cor(prostate[-c(5)])
```

Ne si può vedere una migliore rappresentazione tramite grafico, sfruttando la funzione `corrplot()` della medesima libreria:

```
library(corrplot)

corrplot(mat.cor, method = "color", type = "upper",number.cex = .8, order="hclust", addCoef.col = "blac
```



NOTA: Si è stampata solo la diagonale superiore della matrice di correlazione in quanto simmetrica a quella inferiore. Inoltre, si è omessa la diagonale principale dato che la correlazione di ogni variabile con se stessa è sempre pari a 1.

Notiamo delle importanti correlazioni positive tra:

- *gleason* e *pgg45*: 0.75
- *pgg45* e *lcp*: 0.63
- *lcp* e *lcavol*: 0.68
- *lcavol* e *lspa*: 0.73

Per non incorrere a problemi di multicollinearità è buona norma escludere da una coppia di variabili indipendenti fortemente correlate almeno una di loro.

Per questo si è deciso di eliminare dal nostro dataset le variabili *pgg45* e *lcp*. Questa operazione la svolgeremo tra poco.

Variabile SVI

Per quanto riguarda lo studio della correlazione tra le variabili continue e quella dummy si deve applicare un procedimento differente rispetto al calcolo dell'indice di correlazione visto in precedenza.

Difatti, se abbiamo una variabile continua e una binaria, è impossibile calcolare la correlazione tra loro. Tuttavia, si può usare la regressione per ottenere un valore numerico che può essere trattato in modo simile alla correlazione.

Per questo, dobbiamo creare un modello di regressione lineare semplice prendendo le variabili continue come variabile dipendente e la variabile factor come variabile indipendente.

La radice quadrata dell'R^2 prodotto dal modello sarà il nostro coefficiente di correlazione.

```
lcavol.svi <- sqrt(summary(lm(lcavol ~ svi,data = prostate))$r.squared)
lweight.svi <- sqrt(summary(lm(lweight ~ svi,data = prostate))$r.squared)
age.svi <- sqrt(summary(lm(age ~ svi,data = prostate))$r.squared)
lbph.svi <- sqrt(summary(lm(lbph ~ svi,data = prostate))$r.squared)
lcp.svi <- sqrt(summary(lm(lcp ~ svi,data = prostate))$r.squared)
pgg45.svi <- sqrt(summary(lm(pgg45 ~ svi,data = prostate))$r.squared)
lpsa.svi <- sqrt(summary(lm(lpsa ~ svi,data = prostate))$r.squared)

mat.cor.factor <- data.frame(lcavol.svi,lweight.svi,age.svi,lbph.svi,lbph.svi,lcp.svi,pgg45.svi,lpsa.svi
```

```
knitr::kable(mat.cor.factor, "simple")
```

lcavol.svi	lweight.svi	age.svi	lbph.svi	lbph.svi.1	lcp.svi	pgg45.svi	lpsa.svi
0.538845	0.1553849	0.117658	0.0858432	0.0858432	0.6731112	0.4576476	0.5662182

Notiamo che la variabile *svi* ha un'importante correlazione con *lcp*.

Quindi ora andremo a modificare il nostro dataset:

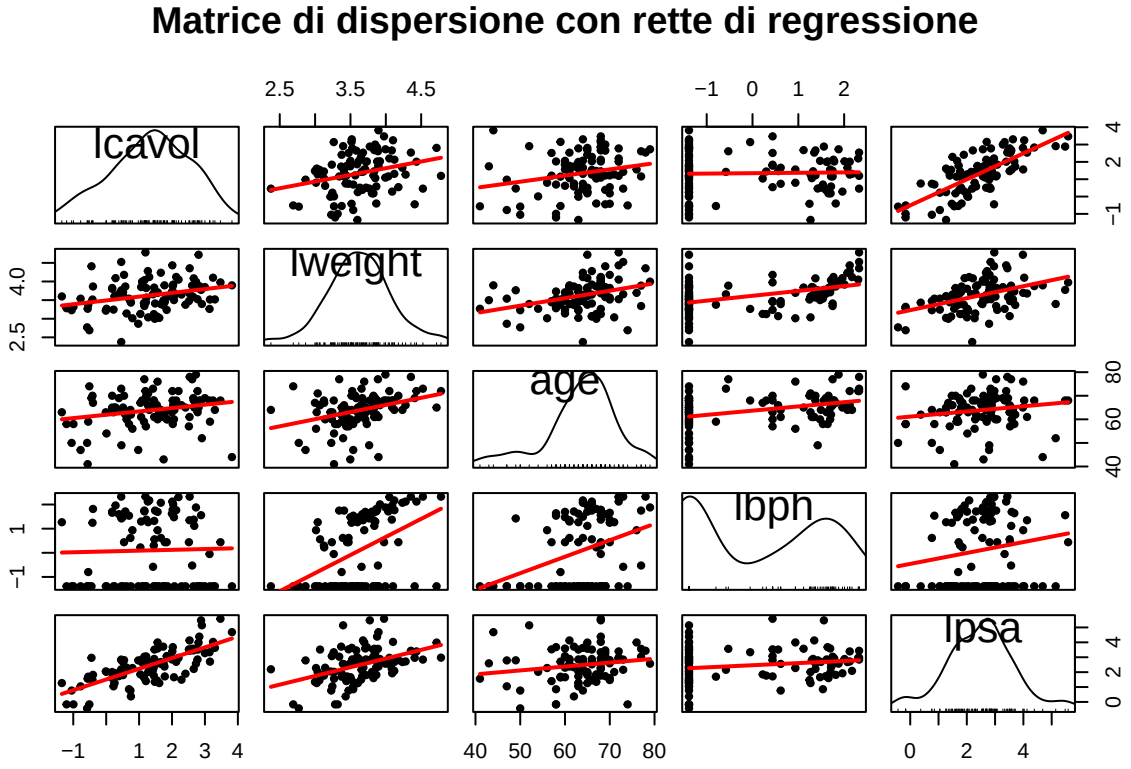
```
prostate.fit <- prostate[-c(6,8)]
detach(prostate)
attach(prostate.fit)
```

Scatter Plot

Di seguito si sono volute rappresentare tutte le nostre unità statistiche in singoli *grafici di dispersione* mettendo sugli assi le varie coppie di variabile, collocando all'interno di esse anche una retta di regressione che andrà a fungere da sintesi.

```
library(car)

scatterplotMatrix( ~ lcavol+lweight+age+lbph+lpsa, col="black", pch=20, regLine = list(method=lm, lty=1
```



Si può notare qualche comportamento anomalo nelle variabili *lbph*.

Regressione

In questa parte andremo ad applicare varie tecniche di regressione.

Prima di entrare nell'analisi della LASSO e della RIDGE, abbiamo effettuato una prima osservazione sulla significatività complessiva del dataset, in quanto potrebbe essere utile per un confronto successivo.

```
summary(lm(lpsa~.,data=prostate.fit))
```

```
##
## Call:
## lm(formula = lpsa ~ ., data = prostate.fit)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.84115 -0.31909 -0.01449  0.42734  1.48310
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.20502    1.12701  -0.182  0.85606
## lcavol       0.51919    0.07876   6.592 2.87e-09 ***
## lweight      0.62118    0.20072   3.095  0.00262 **
```

```
## age         -0.01866    0.01091  -1.710  0.09067 .
## lbph         0.09642    0.05780   1.668  0.09877 .
## svi1         0.68789    0.20836   3.301  0.00138 **
## gleason      0.11272    0.11410   0.988  0.32583
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6989 on 90 degrees of freedom
## Multiple R-squared:  0.6563, Adjusted R-squared:  0.6334
## F-statistic: 28.65 on 6 and 90 DF,  p-value: < 2.2e-16
```

Notiamo che con una prima regressione lineare, includendo tutti i parametri, solo 3 variabili sono risultate significative sull'impatto della nostra variabile dipendente. Inoltre il nostro indice di adattamento ai dati R^2 risulta pari a 0.66 circa, il quale ci indica un discreto adattamento ai dati da parte del modello completo.

Split Dataset

Ora andremo a dividere il nostro dataset in **train** e **test**: il primo verrà usato per allenare il nostro modello, il secondo per effettuare test e previsioni.

Il dataset già conteneva una suddivisione in train e test set attraverso una colonna (*train*) contenente valori logici (True & False).

```
prostate.train <- subset(prostate.fit,prostate.data$train==T)
prostate.train <- prostate.train[-c(10)]

prostate.test <-  subset(prostate.fit,prostate.data$train==F)
prostate.test <- prostate.test[-c(10)]
```

Successivamente andiamo ad escludere la nostra variabile dipendente *lpsa*.

```
train <- prostate.train[-c(9)]
test <- prostate.test[-c(9)]
```

```
library(dplyr)

x.train <- model.matrix(prostate.train$lpsa~.,train)
x.test <- model.matrix(prostate.test$lpsa~., test)

y.train <- prostate.train %>%
  select(lpsa) %>%
  unlist() %>%
  as.numeric()

y.test <- prostate.test %>%
  select(lpsa) %>%
  unlist() %>%
  as.numeric()
```

Ora che i nostri dataset sono pronti possiamo proseguire con le nostre tre tecniche di regressione: OLS, Ridge e Lasso.

OLS

Di seguito verranno calcolate più rette di regressione, utilizzando il metodo **OLS** con una **forward stepwise**: si partirà da un modello con sola intercetta fino ad arrivare a quello completo, aggiungendo di volta in volta una nuova variabile.

```
ols.null <- lm(lpsa ~ 1,data=prostate.train)

ols.1 <- lm(lpsa ~ lcavol, data=prostate.train)

ols.2 <- lm(lpsa ~ lcavol + lweight, data=prostate.train)

ols.3 <- lm(lpsa ~ lcavol + lweight + age, data=prostate.train)

ols.4 <-lm(lpsa ~ lcavol + lweight + age + lbph, data=prostate.train)

ols.5 <- lm(lpsa ~ lcavol + lweight + age + lbph + svi, data=prostate.train)

ols.all <- lm(lpsa ~.,data=prostate.train)
```

In tutto abbiamo ottenuto **7 modelli**: per andare a verificare quale di questi sia quello preferibile si è deciso di utilizzare e calcolare il **BIC** (Criterio di informazione Bayesiano). Un valore del BIC basso corrisponderà ad un modello migliore.

BIC OLS

```
bic.ols.null <- BIC(ols.null)

bic.ols.1 <- BIC(ols.1)

bic.ols.2 <- BIC(ols.2)

bic.ols.3 <- BIC(ols.3)

bic.ols.4 <- BIC(ols.4)

bic.ols.5 <- BIC(ols.5)

bic.ols.all <- BIC(ols.all)
```

```
bics <- cbind (bic.ols.null,bic.ols.1,bic.ols.2,bic.ols.3,bic.ols.4,bic.ols.5,bic.ols.all)

colnames(bics) <- c("OLS NULL","OLS 1","OLS 2","OLS 3","OLS 4","OLS 5","OLS ALL")

knitr::kable(t(bics), "simple")
```

OLS NULL	222.8402
OLS 1	175.3782
OLS 2	167.3397
OLS 3	171.0465

OLS 4	172.7427
OLS 5	170.6534
OLS ALL	173.8563

Notiamo che il BIC inferiore lo troviamo in corrispondenza del **secondo modello**, con le sole variabili *lcavol* e *lweight*. Su questo modello andremo ad applicare la nostra previsione.

Prima di procedere stampiamo i risultati ottenuti dalla regressione su tale modello.

```
summary(ols.2)
```

```
##
## Call:
## lm(formula = lpsa ~ lcavol + lweight, data = prostate.train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.58852 -0.44174  0.01304  0.52613  1.93127
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.04944    0.72904  -1.439 0.154885
## lcavol       0.62761    0.07906   7.938 4.14e-11 ***
## lweight      0.73838    0.20613   3.582 0.000658 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7613 on 64 degrees of freedom
## Multiple R-squared:  0.6148, Adjusted R-squared:  0.6027
## F-statistic: 51.06 on 2 and 64 DF,  p-value: 5.54e-14
```

Notiamo che entrambe le variabili risultano statisticamente significative sull'impatto della variabile *lpsa*. Inoltre, presenta un R^2 pari a 0.61 circa.

Per curiosità andremo a stampare anche i risultati del modello completo.

```
summary(ols.all)
```

```
##
## Call:
## lm(formula = lpsa ~ ., data = prostate.train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.86920 -0.30668 -0.03586  0.44790  1.44824
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.42234    1.34857  -0.313  0.75523
## lcavol       0.49698    0.09855   5.043 4.52e-06 ***
## lweight      0.61620    0.22813   2.701  0.00897 **
## age         -0.01738    0.01387  -1.252  0.21526
```

```
## lbph         0.15162    0.07178   2.112  0.03884 *
## svi1         0.64288    0.27490   2.339  0.02271 *
## gleason      0.14229    0.14967   0.951  0.34557
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7281 on 60 degrees of freedom
## Multiple R-squared:  0.6697, Adjusted R-squared:  0.6366
## F-statistic: 20.27 on 6 and 60 DF,  p-value: 8.506e-13
```

In questo caso risultano significative 4 variabili, ma solo con una di queste possiamo rifiutare l'ipotesi nulla ad un livello di significatività pari a 0.

Test Anova

Il successivo test che si andrà ad applicare è il **test Anova** (Analysis of Variance), che permette di confrontare due modelli alternativi.

In questo caso il test si andrà ad applicare tra il modello *ols.2* e quello completo.

H_0 : il modello completo non fornisce un pieno adattamento ai dati, e quindi non incide sulle variazioni della nostra variabile dipendente *lpsa*.

```
anova(ols.2,ols.all)
```

```
## Analysis of Variance Table
##
## Model 1: lpsa ~ lcavol + lweight
## Model 2: lpsa ~ lcavol + lweight + age + lbph + svi + gleason
##   Res.Df    RSS Df Sum of Sq      F  Pr(>F)
## 1     64 37.092
## 2     60 31.805  4    5.2865 2.4932 0.05235 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Il risultato ottenuto dal test ci permette di conlcudere che il modello *ols.2* è quello preferibile dato un valore del p-value abbastanza alto, quindi andremo ad accettare l'ipotesi nulla.

Previsione

Per proseguire andiamo ad applicare una previsione mettendo in input il nostro modello preferibile (*ols.2*). Tali risultati verrano confrontati con i valori osservati del test set, al fine di calcolare poi il Mean Square Error.

```
predict.ols.2 <- predict(ols.2, newx = x.test)

mean((predict.ols.2 - y.test)^2) #MSE
```

```
## [1] 1.900494
```

RIDGE

In questa sezione andremo ad utilizzare una diversa tecnica di regressione: la **Ridge**. Questo metodo alternativo ci permette di risolvere problemi inerenti all'overfitting e multicollinearità.

```
library(glmnet)

grid <- 10^seq(10, -2, length = 100) #griglia di valori
ridge.mod <- glmnet(x.train, y.train, alpha = 0, lambda = grid, thresh = 1e-12)
```

La nostra griglia di valori utilizzata per la regressione ridge prende come input un insieme di valori che variano da 10^{-2} fino a 10^{10}.
Attraverso la funzione `glmnet` andremo a stimare i nostri β utili per la previsione.

Di seguito si sono voluti riportare due grafici:

- il primo a sinistra mostra i coefficienti ottenuti con la regressione ridge come funzione di λ;
- il secondo a destra mostra i coefficienti ottenuti con la regressione ridge come funzione di $\|\hat{\beta}^R_\lambda\|_2/\|\hat{\beta}\|_2$, dove:
  - $\hat{\beta}$ indica il vettore delle stime dei coefficienti dei minimi quadrati;
  - $\|\beta\|_2$ indica la norma l_2 di un vettore e misura la distanza di β da zero;

Quindi al crescere di λ, la norma l_2 di $\hat{\beta}^R_\lambda$ decresce sempre.

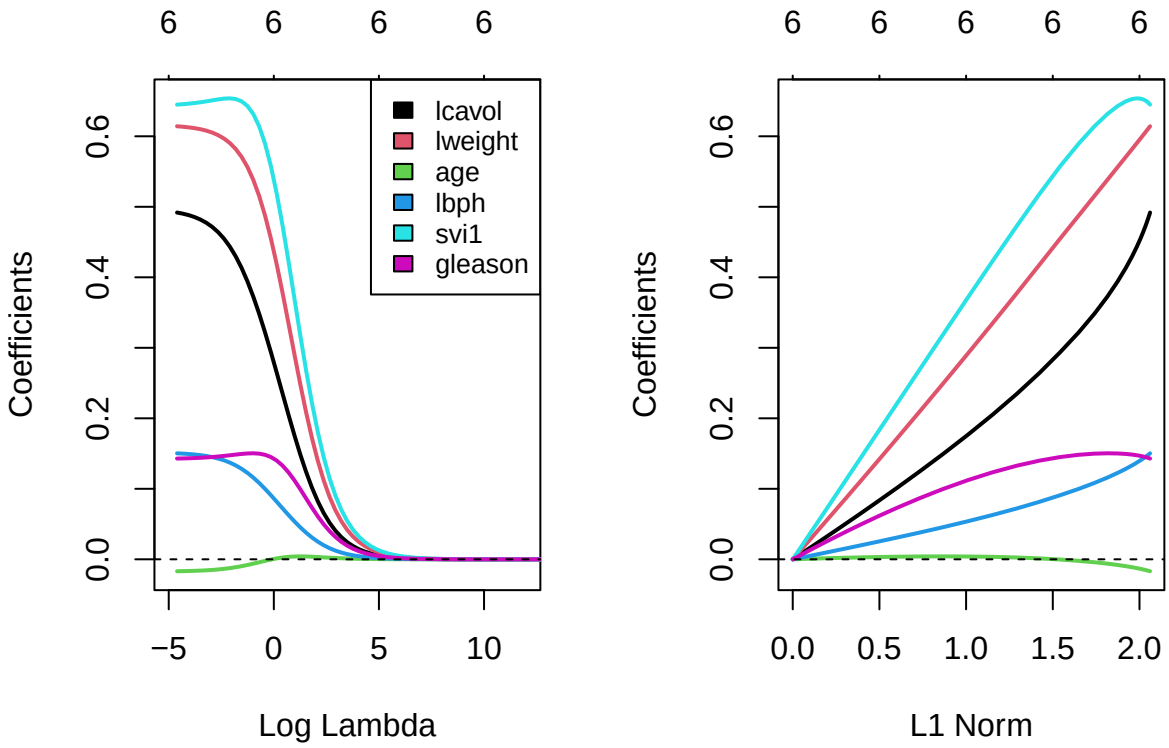
NOTA: quando $\lambda = 0$ i coefficienti della regressione ridge sono gli stessi dei minimi quadrati, così come le loro norme l_2; Quando invece $\lambda = \infty$ allora i coefficienti della regressione ridge sono un vettore di zeri, con norma l_2 pari a zero.

Possiamo pensare all'asse delle x nel grafico di destra come alla quantificazione di quanto le stime dei coefficienti della regressione ridge sono state ridotte verso lo zero: un valore piccolo indica che le stime sono state ridotte molto vicino allo zero.

```
par(mfrow=c(1,2))

plot(ridge.mod, xvar = "lambda", lwd = 2, col=c(1:7), xlim = c(-5,12))
abline(h=0,col="black", lty=2)
legend("topright", legend=names(ols.all$coefficients[-c(1)]), fill = c(1:7), cex=0.8)

plot(ridge.mod, lwd = 2, col=c(1:7))
abline(h=0,col="black", lty=2)
```



Nella parte sinistra ogni curva corrisponde a coefficiente della regressione ridge per ognuna delle variabili come funzione di λ. Nella parte iniziale del grafico λ prende valori prossimi allo zero, quindi le corrispondenti stime dei coefficienti ridge sono le stesse dei minimi quadrati. Però, al crescere di λ queste stime si riducono verso lo zero, e quando λ è molto grande allora tutte le stime dei coefficienti sono praticamente zero, e questo corrisponde ad un *modello nullo* senza predittori.

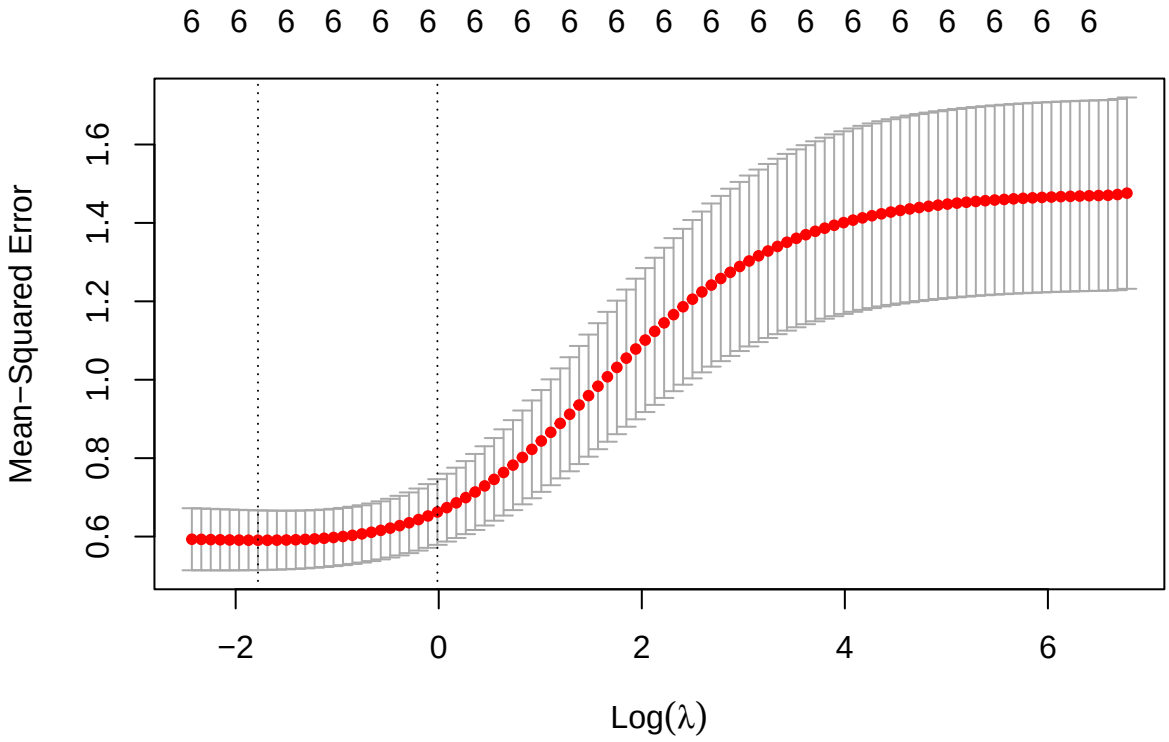
In questo caso le stime dei coefficienti maggiori si trovano in corrispondenza delle variabili **lcavol**, **lweight** e **svi**.
Inoltre, notiamo che per un breve tratto le stime dei coefficienti, in corrispondenza delle variabili **gleason** e **age**, tendono a crescere all'aumentare del λ.

Possiamo dire inoltre che alcune variabili tendono a raggiungere lo zero più velocemente rispetto ad altre (vedi *age*); tali variabili sono quelle che saranno meno importanti ai fini della previsione.

Cross Validazione

In questa sezione andremo ad applicare una tecnica di **K-Fold Cross Validation**, la quale è utilizzata per risolvere problemi di overfitting nel training set. In particolare questo approccio prevede la suddivisione casuale dell'insieme di osservazioni in k *gruppi* (di default pari a 10).
Il primo gruppo viene considerato come validation set e il fitting del metodo viene eseguito sui $k-1$ gruppi. L'MSE viene quindi calcolato sulle osservazioni dei gruppi rimanenti.
Questa procedura viene ripetuta k volte; ogni volta un diverso gruppo di osservazioni viene trattato come validation set.

```
set.seed(1)
#Fittiamo la regressione ridge sul training set
cv.out.ridge <- cv.glmnet(x.train, y.train, alpha = 0)
plot(cv.out.ridge)
```



Il garfico riporta sull'asse delle ascisse i valori di λ, espresso in logaritmi; mentre sull'asse verticale avremo i valori dei vari MSE calcolati.

La linea formata dai punti rossi sta ad indicare la curva di Cross Validazione, mentre le linee al di sopra e al di sotto di essa rappresentano le curve di deviazione standard.

Per quanto riguarda le linee verticali tratteggiate si evidenziano due valori di λ: - quello più a sinistra corrisponde al λ con il minor valore dell'MSE risultante dalla Cross Validazione;

- il secondo, invece, è il valore di λ associato al modello più regolarizzato (tale che l'errore della CV rientra nel minor range di errore standard).

Di seguito andremo a selezionare il λ che ci restituisce il minor MSE.

```
bestlam.ridge <- cv.out.ridge$lambda.min
bestlam.ridge
```

```
## [1] 0.1685614
```

Quindi in corrispondenza del minor MSE si riscontra un λ pari a 0.17 circa.

Previsione

Una volta ottenute le $\hat{\beta}$ dal modello stimato sul training set le andremo a moltiplicare per le x.test in modo tale da ottenere le $\hat{y}$. Quest'ultime saranno utilizzate per ottenere una previsione dell'y.test, cioè una previsione di tutti i valori della variabile dipendente **lpsa** corrispondente a quelle unità statistiche appartenenti al test set.

Tutto ciò avverrà in funzione del miglior λ, calcolato in precedenza.

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

```
## [1] 0.442676
```

Best BIC

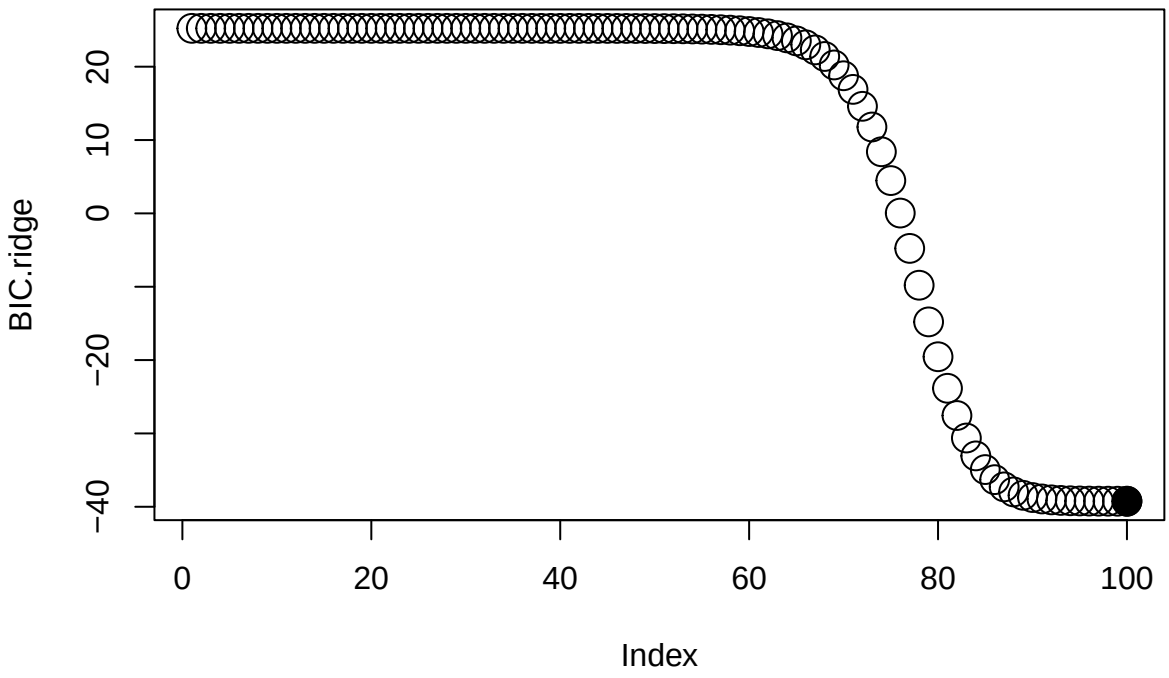
Per avere un confronto con la Lasso dobbiamo ricorrere al calcolo del BIC sul nostro modello stimato.

```
tLL <- ridge.mod$nulldev - deviance(ridge.mod) #devianza residuale
k <- ridge.mod$df #gradi di libertà
n <- ridge.mod$nobs #numero di osservazioni estratte

BIC.ridge <- log(n)*k - tLL
best.bic.ridge <- which.min(BIC.ridge)
best.bic.ridge #Posizione miglior BIC
```

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

```
plot(BIC.ridge,cex=2)
points(best.bic.ridge,BIC.ridge[best.bic.ridge],pch=19,cex=2)
```




```
BIC.ridge[best.bic.ridge] #Valore miglior BIC
```

```
## [1] -39.24535
```

Effettuati i vari calcoli sulla Ridge, abbiamo trovato il miglior BIC nella **posizione 100** con valore pari a **-39.25** circa.

LASSO

Ora andiamo ad applicare una diversa tecnica di regressione, chiamata **LASSO**.
La Ridge includeva tutti i p predittori del modello finale, e questo potrebbe portare ad un problema di interpretazione quando p è elevato.

Il nostro obiettivo quindi è costruire un modello che non comprenda tutti questi predittori, ma soltanto un sottoinsieme di essi.

La Lasso usa un termine di penalità l_1 che ha come efetto quello di portare e forzare verso lo zero alcune delle stime dei coefficienti quando il parametro di tuning λ è abbastanza grande.

I modelli ottenuti dalla Lasso infatti risultano essere più facili da interpretare rispetto a quelli prodotti dalla Ridge.

Come prima operazione adiamo a fittare il modello LASSO sul training set, utilizzando la medesima griglia di valori (*grid*).

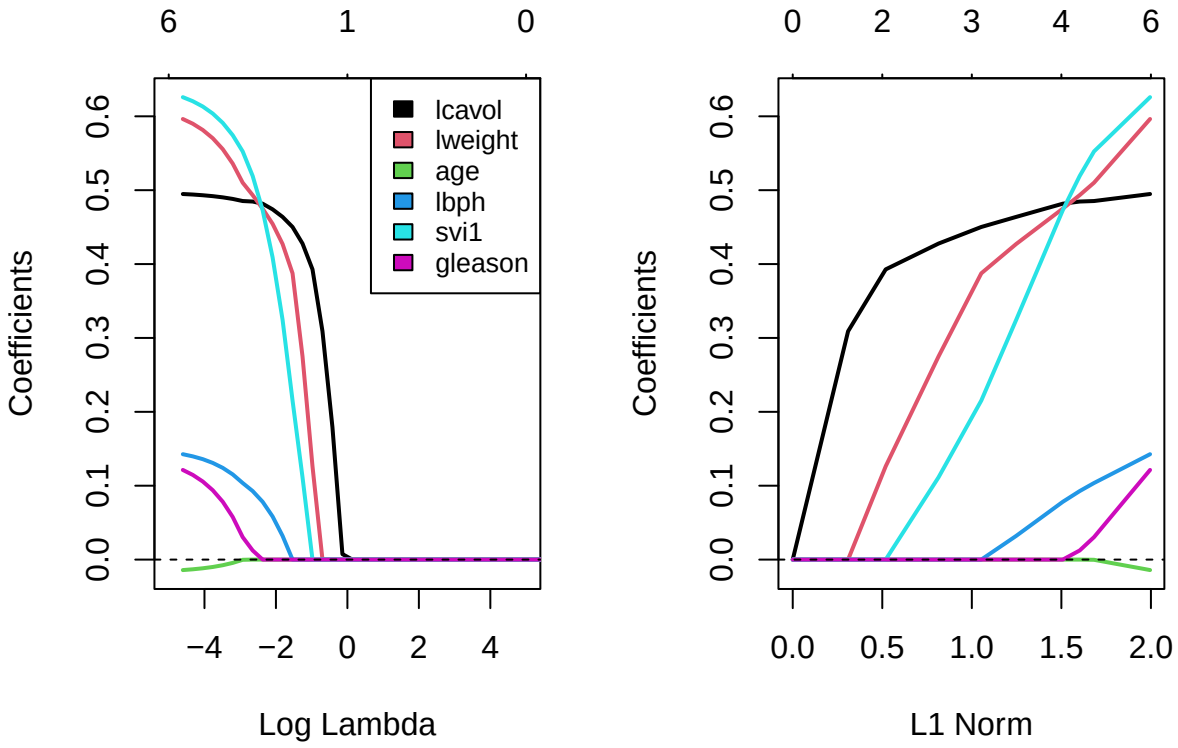
```
lasso.mod <- glmnet(x.train, y.train, alpha = 1, lambda = grid)
```

Quindi andiamo a rappresentare tramite grafico i risultati ottenuti dalla Lasso.

```
par(mfrow=c(1,2))

plot(lasso.mod, lwd = 2, col=c(1:7), xvar = "lambda", xlim = c(-5,5))
abline(h=0,col="black", lty=2)
legend("topright", legend=names(ols.all$coefficients[-c(1)]), fill = c(1:7), cex=0.8)

plot(lasso.mod, lwd = 2, col=c(1:7))
abline(h=0,col="black", lty=2)
```



In particolare, si sono voluti riportare due grafici:

- il primo a sinistra mostra i coefficienti ottenuti con la regressione Lasso come funzione di λ;
- il secondo a destra mostra i coefficienti ottenuti con la regressione Lasso come funzione di $\|\hat{\beta}_\lambda^L\|_1/\|\hat{\beta}\|_1$

Quando $\lambda = 0$ allora la Lasso fornisce semplicemente le stime dei minimi quadrati, mentre quando λ diventa sufficientemente grande la lasso fornisce il modello nullo in cui tutte le stime dei coefficienti sono pari a zero.

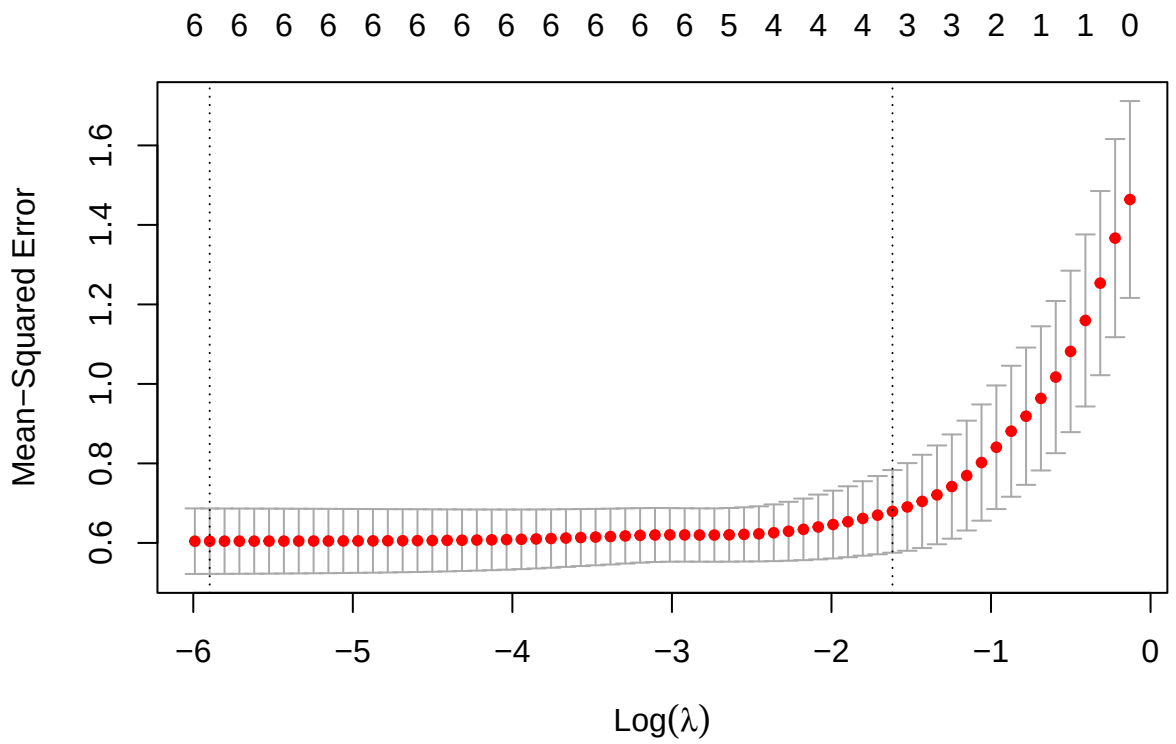
Interpretando la parte destra del grafico si può dire che inizialmente la lasso produce un modello che contiene solo il predittore **lcavol**. Poi entrano anche **lweight** e **svi** quasi contemporaneamente, seguiti infine dagli ultimi tre: **lbph**, **gleason** e **age**.

NOTA: Possiamo affermare che la regressione Lasso, a seconda del valore di λ, produce un modello che coinvolge un numero indefinito di variabili. Al contrario, la Ridge includerà sempre tutte le variabili nel modello, sebbene le stime dei coefficienti dipenderanno anch'esse dal λ.

Cross Validazione

Continuando, si è voluta ripetere la tecnica del **K-Fold Cross Validation**.

```
set.seed(1)
cv.out.lasso <- cv.glmnet(x.train, y.train, alpha = 1)
plot(cv.out.lasso)
```



Quindi selezionamo il λ che minimizza l'MSE del training.

```
bestlam.lasso <- cv.out.lasso$lambda.min
bestlam.lasso
```

```
## [1] 0.00274713
```

Previsione

Ora andremo a selezionare il miglior λ per la nostra previsione e di seguito riportiamo il valore dell'MSE corrispondente.

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

```
## [1] 0.4308338
```

Best BIC

Come detto in precedenza utilizzeremo il BIC come criterio di confronto tra i due modelli.

```
tLL <- lasso.mod$nulldev - deviance(lasso.mod) #devianza residuale
k <- lasso.mod$df #gradi di libertà
n <- lasso.mod$nobs #numero di osservazioni estratte
BIC.lasso <- log(n)*k - tLL
```

```
best.bic.lasso <- which.min(BIC.lasso)
best.bic.lasso #Posizione miglior BIC
```

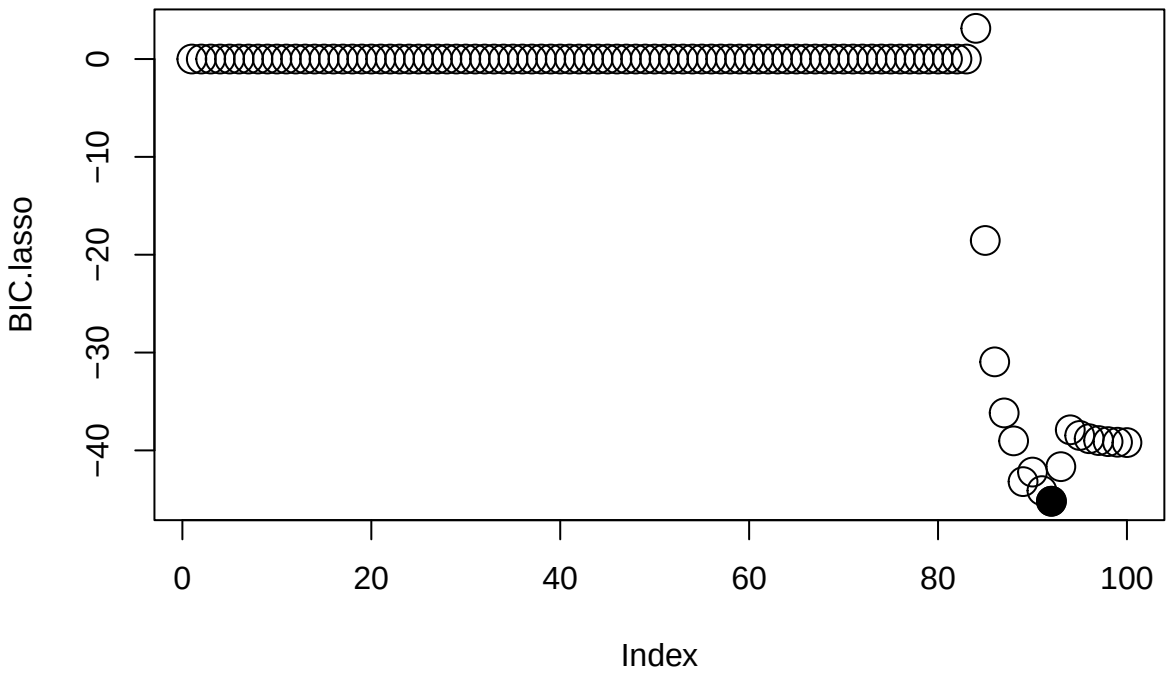
```
## [1] 92
```

```
BIC.lasso[best.bic.lasso] #Valore miglior BIC
```

```
## [1] -45.19699
```

Dopo aver effettuato i calcoli ci risulta che il miglior BIC si trova in posizione 92 con valore pari a -45.2. Possiamo vederlo anche graficamente:

```
plot(BIC.lasso,cex=2)
points(best.bic.lasso,BIC.lasso[best.bic.lasso],pch=19,cex=2)
```



A seguito di questa analisi possiamo concludere che il modello risultante dalla LASSO mostra un miglior adattamento rispetto la RIDGE, in quanto presenta un BIC inferiore.

Analisi Coefficienti

Per concludre, andiamo a verificare se i nostri modelli Ridge e Lasso hanno effettivamente effettuato un operazione di killing dei parametri sul modello completo.

```
x <- model.matrix(lpsa~., prostate.fit)

y <- prostate.data %>%
  select(lpsa) %>%
  unlist() %>%
  as.numeric()
```

RIDGE

```
out.ridge <- glmnet(x, y, alpha = 0, lambda = grid)
#Fittiamo la Ridge sul dataset completo
ridge.coef <- predict(out.ridge, type = "coefficients", s = bestlam.ridge)
#Mostriamo i soli coefficienti non nulli usando il lambda risultante dalla CV
ridge.coef[ridge.coef != 0]
```

```
## [1] -0.47639785  0.44642533  0.58066109 -0.01274399  0.07990824  0.67735359
## [7]  0.13389923
```

LASSO

```
out.lasso <- glmnet(x, y, alpha = 1, lambda = grid)
#Fittiamo la Lasso sul dataset completo
lasso.coef <- predict(out.lasso, type = "coefficients", s = bestlam.lasso)
#Mostriamo i soli coefficienti non nulli usando il lambda risultante dalla CV
lasso.coef[lasso.coef != 0]
```

```
## [1] -0.21230264  0.51594689  0.59986664 -0.01563225  0.08706754  0.67105598
## [7]  0.09798117
```

Confrontando i coefficienti stimati rispettivamente dalla Ridge e dalla Lasso possiamo notare che entrambi i modelli considerano tutti i parametri, senza effettuare alcuna operazione di "killing" (che risulterebbe preferibile quando vogliamo utilizzare dei modelli più parsimoniosi).
Nonostante ciò, possiamo concludere che i risultati migliori li otteniamo in corrispondenza del modello Lasso.