Modell választás

Contents

Bevezetés

Az adatelemzési modellek felállítása során kritikus fogalom az ún. túl- illetve alulillesztés. A fogalmak megismerését egy egyszerű, polinomiális regresszió példáján fogjuk megismerni.

A modell

Tekintsünk egy egyszerű regressziós feladatot, amely során egy polinomot szerenénk illeszteni az adathalmazra. A polinomok általános formája a következő:

$$\beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_M x^M$$

A regresszió során láttuk, hogy a polinomok β paramétereinek meghatározására klasszikus regressziós technikákat használhatunk. Általánosságban, a polinomoknak van egy további, ún. hiperparamétere, jelen esetben a polinom maximális fokszáma, M. A *hiperparaméter* egy általános fogalom, minden parametrizálható modellnek lehet hiperparamétere, ezekre számos példát fogunk látni. A továbbiakban áttekintjük, miként lehetséges a hiperparaméterek megválasztása.

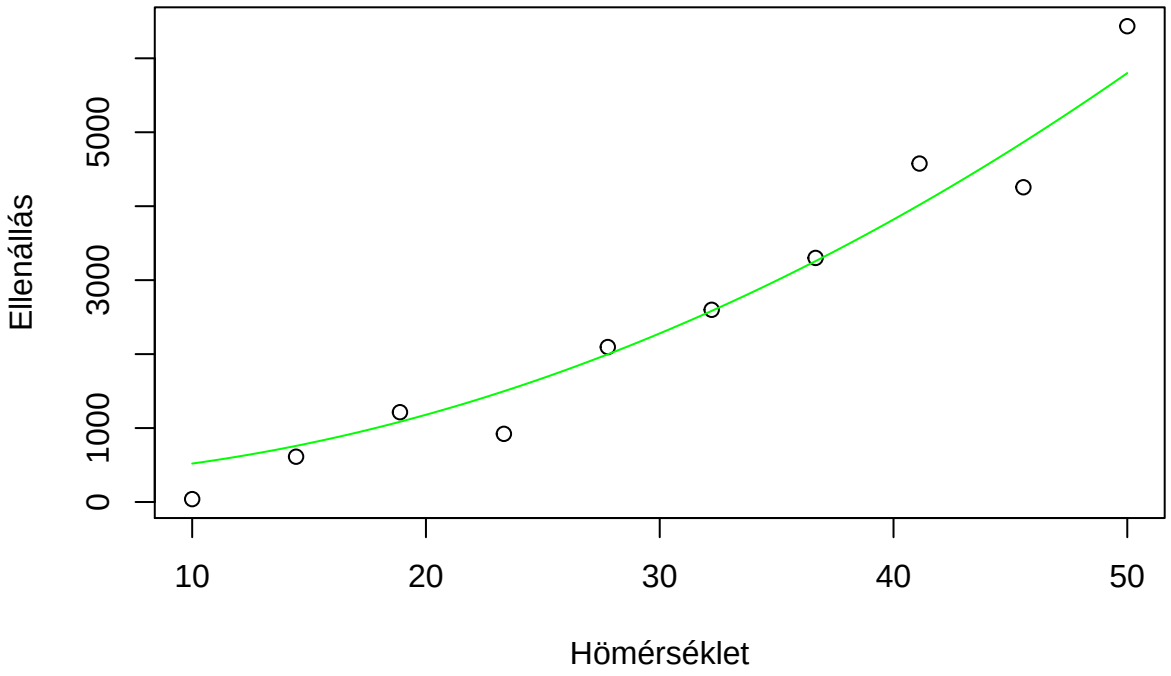
Túl- és alulillesztés

A túl- és alulillesztés az adatelemzés központi fogalmai. Az alulillesztés egyszerűen azt jelenti, hogy az algoritmus az adatsorokat túl nagy hibával tudja előállítani, tehát nem alkalmas az adatok hatékony leírására. A túlillesztés ezzel ellentétes fogalom, és azt jelenti, hogy az algoritmus túlságosan illeszkedik a tanító adatokra, viszont jövőbeli adatokra rosszul működik (nem generalizálható). Habár a túl és alulillesztés

egyszerű fogalmak, felismerésük egy összetett adathalmazon meglehetősen nehéz. A fogalmak bemutatására tekintsünk egy egyszerű polinomillesztés feladatot!

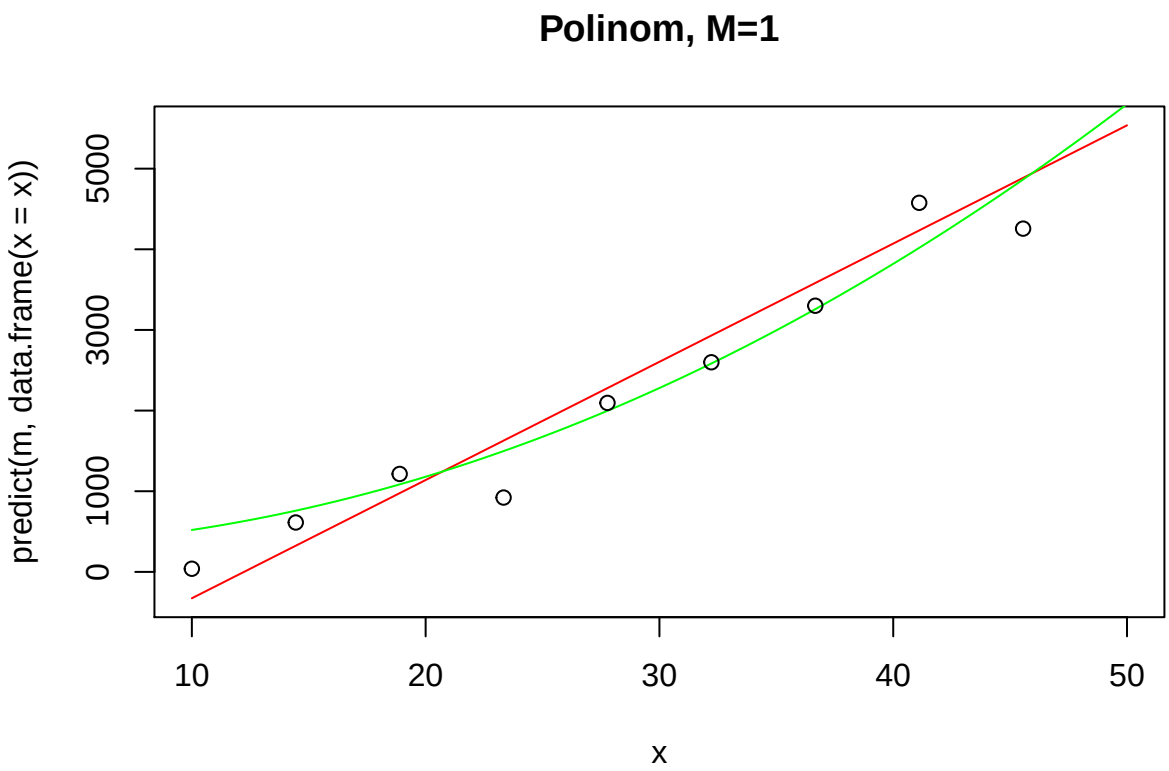
Tételezzük fel, hogy egy gázszenzor ellenállásának hőmérsékletfüggését mérjük! Tételezzük fel azt is demonstrációs célokból, hogy ezt az összefüggést zárt alakban is ismerjük 10 és 50 Celsius-fok között ($f(t) = 2.2t^2 + 300\Omega$)! A függvényt a következő ábrán láthatjuk, és ábrázoltuk a függvényből vett $\sigma = 500$ szórással rendelkező zajos mintákat is.

```
set.seed(3)
fn<-function(t) {
  2.2*t^2 + 300
}

x<-seq(10,50,length.out=10)
y<-fn(x) + rnorm(length(x),sd=500)
df<-data.frame(x=x,y=y)
plot(df,xlab='Hömérséklet',ylab='Ellenállás')

curve(fn,add=T,col='green')
```

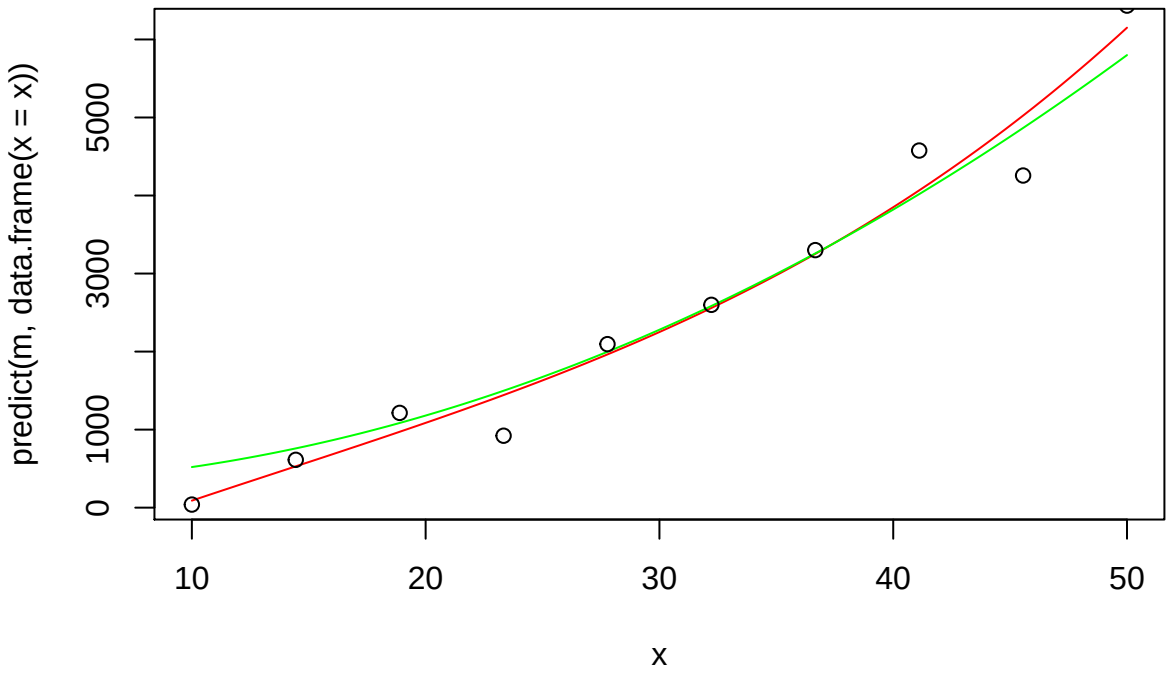


Illesszünk különböző fokszámú polinomokat a mintákra! Nyilvánvaló, hogy 10 pontra maximum 9 fokszámú polinom illeszthető. Érdemes megtekinteni, hogy 1 fokszámú polinom (lineáris függvény) esetén a modell erősen *alulillesztett*, hiszen az eredeti függvényünket nem sikerült jó minőségben modelleznie. Érdekes azonban, hogy a regresszió során használt reziduálok hibája tetszőlegesen csökkenthető, akár nulláig, ahogy azt a 9 fokszámú polinom mutatja.

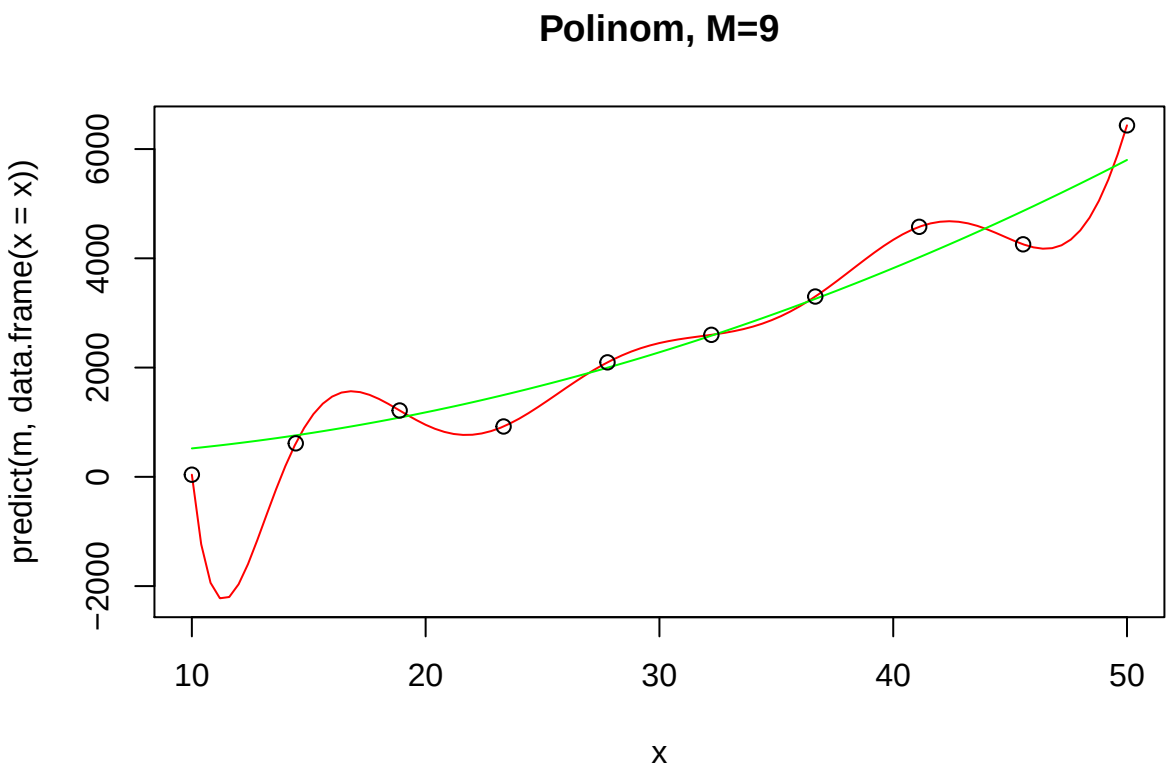
```
m<-lm(y~poly(x,1),df)
curve(predict(m,data.frame(x=x)),col='red',from=10,to=50,main='Polinom, M=1')
curve(fn,col='green',add=T)
points(df)
```




```
m<-lm(y~poly(x,3),df)
curve(predict(m,data.frame(x=x)),col='red',from=10,to=50,main='Polinom, M=3')
curve(fn,col='green',add=T)
points(df)
```

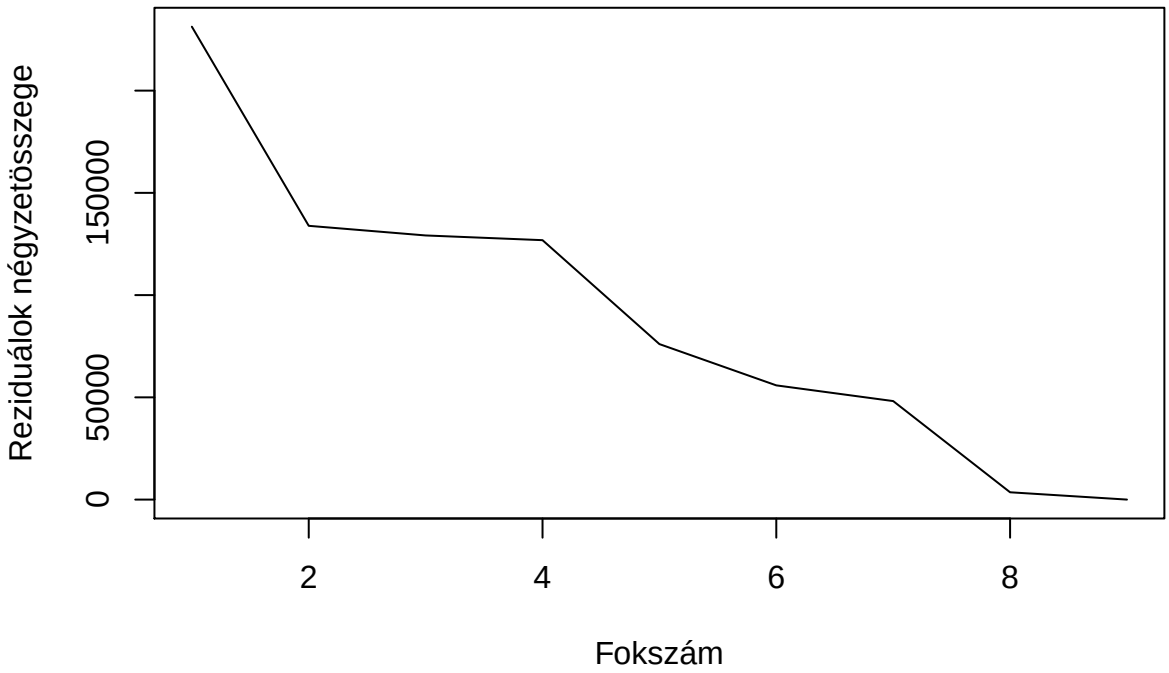


```
m<-lm(y~poly(x,9),df)
curve(predict(m,data.frame(x=x)),col='red',from=10,to=50,main='Polinom, M=9')
curve(fn,col='green',add=T)
points(df)
```



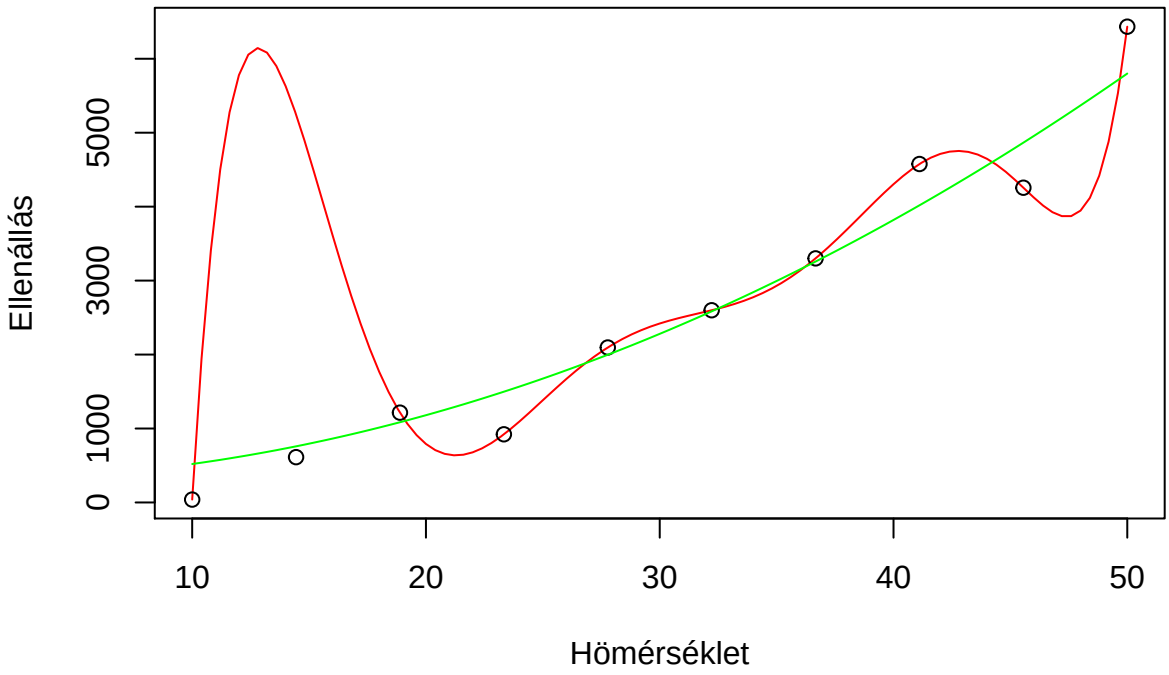
Természetesen, ha a polinomot ábrázoljuk, akkor láthatjuk, hogy ugyan a reziduálok hibája nulla, mégis, a polinom nagyon rosszul közelíti az eredeti függvényt. Ekkor azt mondjuk, hogy a modell *túlillesztett*, és az eredeti adatpontok és eredeti modell helyett elsősorban a mintákban található zajra illeszkedett a modell. Láthatjuk, hogy az eredeti adatpontokon túl, tetszőleges másik adatpont predikciója esetén a hiba nagy lesz. Tekintsük a reziduálok négyzetösszegét különböző fokszámokra:

```
res<-sapply(1:9,function(k) {
  m<-lm(y~poly(x,k),df)
  mean(m$residuals^2)
})
plot(1:9,res,type='l',xlab = 'Fokszám', ylab='Reziduálok négyzetösszege')
```



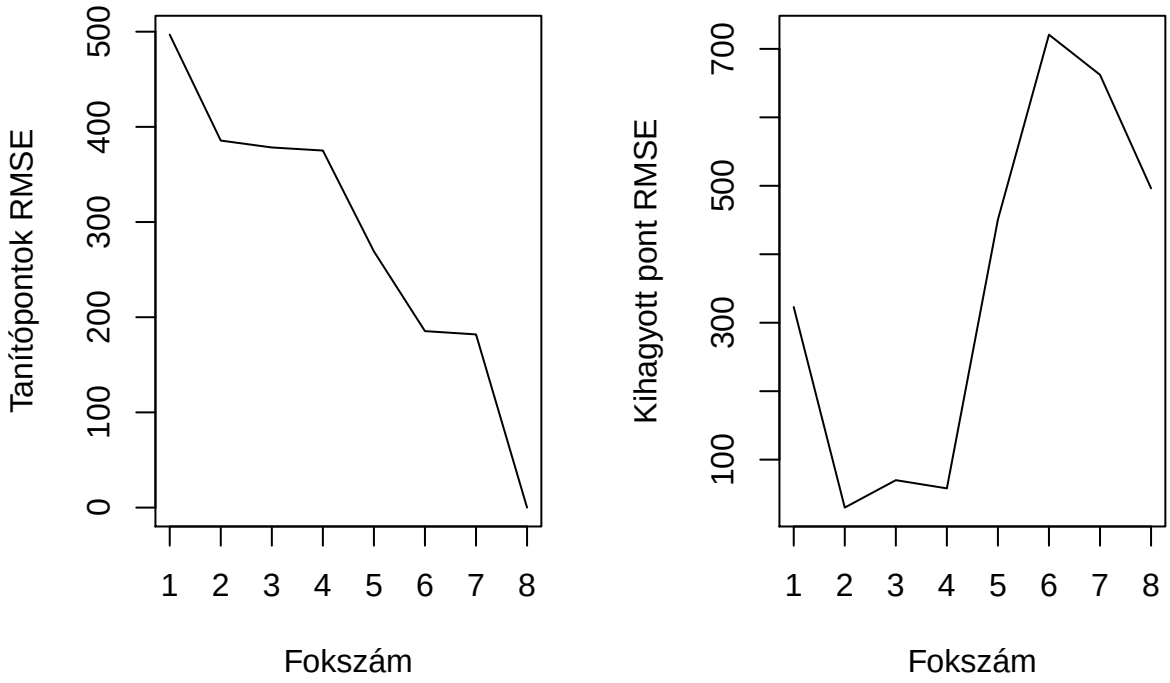
Láthatjuk, hogy a fokszámok növelésével a reziduálok hibája csökken. Legyen az ötletünk az, hogy a 10 pontból egyet elhagyunk, és a maradék 9 pontra elvégezzük a polinom illesztését, majd a 10-dik pontra kiszámítjuk, mekkora a modell hibája. Láthatjuk, hogy ebben az esetben a polinom nem illeszkedik a kihagyott pontra:

```
idx<-2
train<-df[-idx,]
m<-lm(y~poly(x,8),train)

curve(predict(m,data.frame(x=x)),from=10,to=50,col='red',type='l',xlab='Hömérséklet',ylab='Ellenállás')
points(df)
curve(fn,add=T,col='green')
```



Ha ábrázoljuk minden fokszámhoz a reziduálok hibáját, valamint a kihagyott pontunk hibáját, azt vesszük észre, hogy míg a reziduálok hibája folyamatosan csökken, addig a kihagyott pontunk hibája kezdetben csökken, majd emelkedésnek indul.

```
validate<-function(degree,trainIdx,df) {
  train<-df[trainIdx,]
  val<-df[-trainIdx,]

  # ha kevesebb pontot adunk meg, mint a fokszám+1, akkor az lm hibát dob
  tryCatch({
    m<-lm(y~poly(x,degree),train)
    pred<-predict(m,val)

    sqrt(c(mean(m$residuals^2),mean((pred-val$y)^2)))
  },error = function(err) {
    c(NA,NA)
  })
}

res<-sapply(1:8,validate,trainIdx=-7,df=df)

par(mfrow=c(1,2))
plot(1:8,res[1,],type='l',xlab='Fokszám',ylab='Tanítópontok RMSE')
plot(1:8,res[2,],type='l',xlab='Fokszám',ylab='Kihagyott pont RMSE')
```



A túllillesztés jelensége tehát tipikusan a következőképpen írható le: egy tetszőleges modellt tanítunk különböző hiperparaméter értékekre az ún. *tanító adathalmazon*, amelyre a modell hibája - tipikusan - folyamatosan csökken. Ezzel párhuzamosan, a modell hibáját egy független adathalmazon kiértékeljük. Ez a független adathalmaz a *validációs adathalmaz*, és az ezen előálló hibát *validációs hibának* nevezzük. A modell akkor kerül túllillesztett állapotba, ha a validációs adathalmazon a hiba emelkedni kezd, azaz a modell illesztés már csak a tanító adathalmaz pontjaira csökkenti a hibát. A jelenséget, miszerint egy modell a tanító adathalmazokon nyújtott teljesítményét nyújtja általa még nem látott adatokon, *általánosíthatóságnak* (generalizáció) nevezzük.

Validáció

A validációs eljárás célja, hogy egy modell optimális hiperparamétereit (pl. polinomiális regresszió fokszámát) meghatározzuk. Ehhez, ahogy azt az előzőekben láttuk, a modell paramétereinek optimalizálására használt adatoktól független adatsorokkal végezzük, amelyeket validációs adatoknak hívunk. A validációs adatsorok előállítására a legegyszerűbb módszer, hogy az adathalmazt szétvágjuk egy tanító és egy validációs adathalmazra. Ezzel azonban értékes adatot pazarolnánk el, másrészt statisztikai szemszögből sokkal hatékonyabb algoritmusok is léteznek. A validációs adatok kiválasztására számos módszer terjedt el, amelyeket a következőkben mutatunk be.

Bootstrapping

A bootstrapping egy statisztikából származó megoldás, és lényege, hogy az eredeti adathalmazból visszatevéses mintavétellel új adathalmazt generálunk, ugyanakkora méretben, mint az eredeti adathalmaz. Ebből az adathalmazból kimaradó mintákat (*holdout*) használjuk validációra. Például, generáljuk egy bootstrap adathalmazt a mintapontjainkból:

```
trainIdx<-sample(10,replace=T)
print(trainIdx) # tanító indexek
```

```
##  [1] 5 8 5 9 9 8 6 2 9 8
```

```
print((1:10)[-trainIdx]) # validációs indexek
```

```
## [1]  1  3  4  7 10
```

A bootstrapping elvégezhető tetszőleges alkalommal, majd az ebből származó eredmények átlagát vesszük.

```
set.seed(1)
grid<-expand.grid(iter=1:10,degree=1:6)
out<-apply(grid,1, function(row) {
    trainIdx<-sample(10,replace=T)
    validate(row['degree'],trainIdx,df)
})
aggregate(t(out),list(degree=grid$degree),mean,na.rm=T)
```

```
##   degree           V1        V2
## 1      1 3.657256e+02  802.9673
## 2      2 3.105973e+02  663.5035
## 3      3 2.614103e+02  755.2786
## 4      4 1.619196e+02 3122.0488
## 5      5 6.703537e+01 3334.0800
## 6      6 6.272233e-13 6048.2272
```

Bootstrapping elmélete (érdekesség)

A bootstrapping módszer a statisztikából származtatható, és eredeti célja, hogy tetszőleges statisztika eloszlását becsülni tudjuk. Nézzük például az átlagot, mint statisztikát. Tudjuk, hogy a becsült átlag szórásnégyzete, ha a minták függetlenek, az eredeti minták szórásnégyzetének és a minták számának a hányadosa:

$$\mathrm{Var}\left[\frac{1}{N}\sum_i x_i\right] = \frac{1}{N^2}\sum_i \mathrm{Var}\left[x_i\right] = \frac{\sigma_x^2}{N}$$

Persze az átlag, mint statisztika eloszlása egyszerűen számítható, de a valóságban léteznek olyan statisztikák, melyek eloszlása bonyolultabb problémákhoz vezet. Ettől függetlenül itt most példának az átlag szórásnégyzetének a becslése a megértéshez bőven elegendő.

Tehát, például, tételezzük fel, hogy megmérjük 30 ember magasságát, és tételezzük fel továbbá, hogy az emberek magasságának szórása 4 centiméter, várható értéke 175 centiméter. Persze a valóságban ezeket nem tudjuk, de szemléltetésnek vegyük ezeket a paramétereket ismertnek. Nyilván, többször elvégezhetjük a kísérletet, mindig másik 30 ember magasságát mérve, és minden kísérletre némileg eltérő magasságátlagot kapva. Ha nem áll rendelkezésünkre több minta, akkor az eredeti 30 mintán is elvégezhetjük a kísérleteket, szimulált adathalmazokon, *bootstrapping* módszert használva. Itt minden halmazra egy-egy magasságátlagot kapunk, melynek eloszlása jól becsüli a teljes populáción az átlag eloszlását. Tekintsük például a következőt, ahol 40 bootstrap halmazra végezzük el a kiértékelést:

```
set.seed(1)
height<-rnorm(30,175,4)

means<-sapply(1:40,function(i) mean(sample(height,replace=T)))

var(means)
```

```
## [1] 0.5130572
```

Láthatjuk, hogy az eredmény jól közelíti az elméleti szórásnégyzetét az átlagnak, amely $\frac{4^2}{30} = 0.5333$.

Keresztvalidáció

A keresztvalidáció nem véletlenszerű módszert alkalmaz a tanító és validációs adathalmaz létrehozásához, hanem valamilyen determinisztikus logikát. A lényege, hogy az adathalmazt úgy osztják kétfelé, hogy minden

adatpont *legalább egyszer* a tanító adathalmazba és a validációs adathalmazba is bekerüljön.

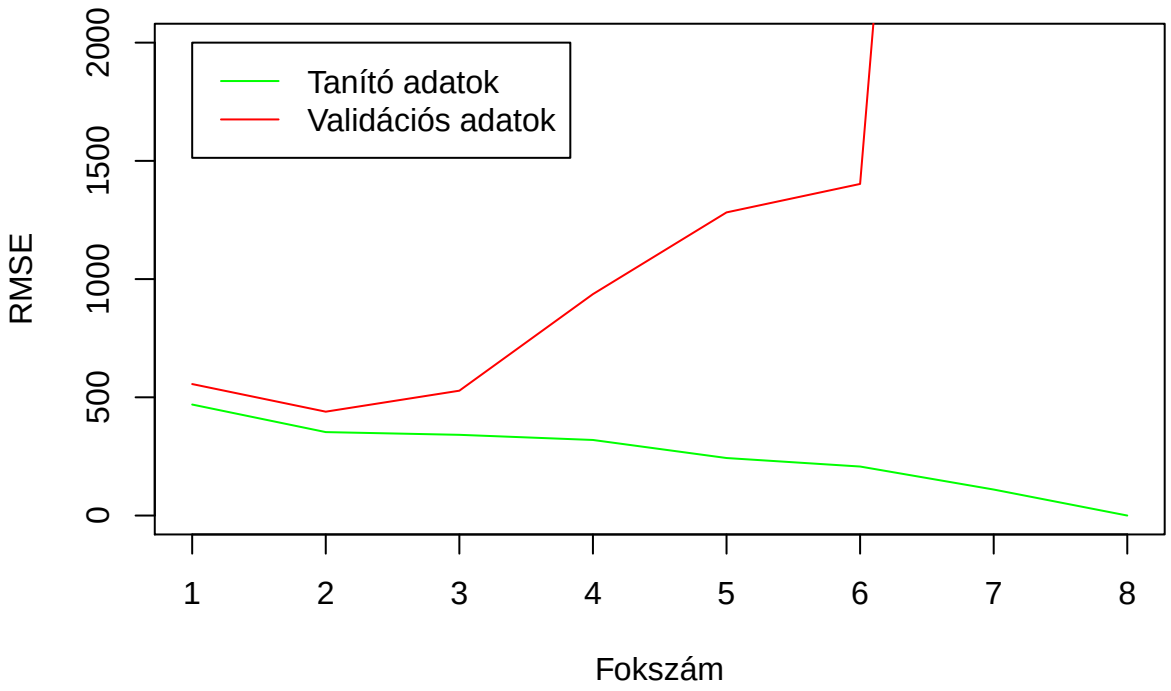
Leave-p-out

A módszer p darab adatpontot kiválaszt a validációs adathalmazba és a maradék adatpontot pedig tanításra használja fel. Ezt a kiválasztást *minden* kombinációra megteszi, tehát $\binom{n}{p}$ adathalmazt generálva (ahol n az adathalmaz mérete). Ritkán használják, hiszen a mintahalmazok mérete hatalmas lehet.

Leave-one-out

Az előző módszer $p = 1$ esetre. Ez a módszer n adatpont esetén n adathalmazt generál. Például, a polinomillesztés esetére:

```
set.seed(1)
grid<-expand.grid(idx=1:10,degree=1:8)
out<-apply(grid,1,function(row) {
  validate(row['degree'],(1:10)[-row['idx']],df)
})
res<-aggregate(t(out),list(degree=grid$degree),mean)

plot(res$V1,col='green',type='l',ylim=c(0,2000),xlab='Fokszám',ylab='RMSE')
lines(res$V2,col='red')
legend(1,2000,legend=c('Tanító adatok','Validációs adatok'),col=c('green','red'),lty=1)
```

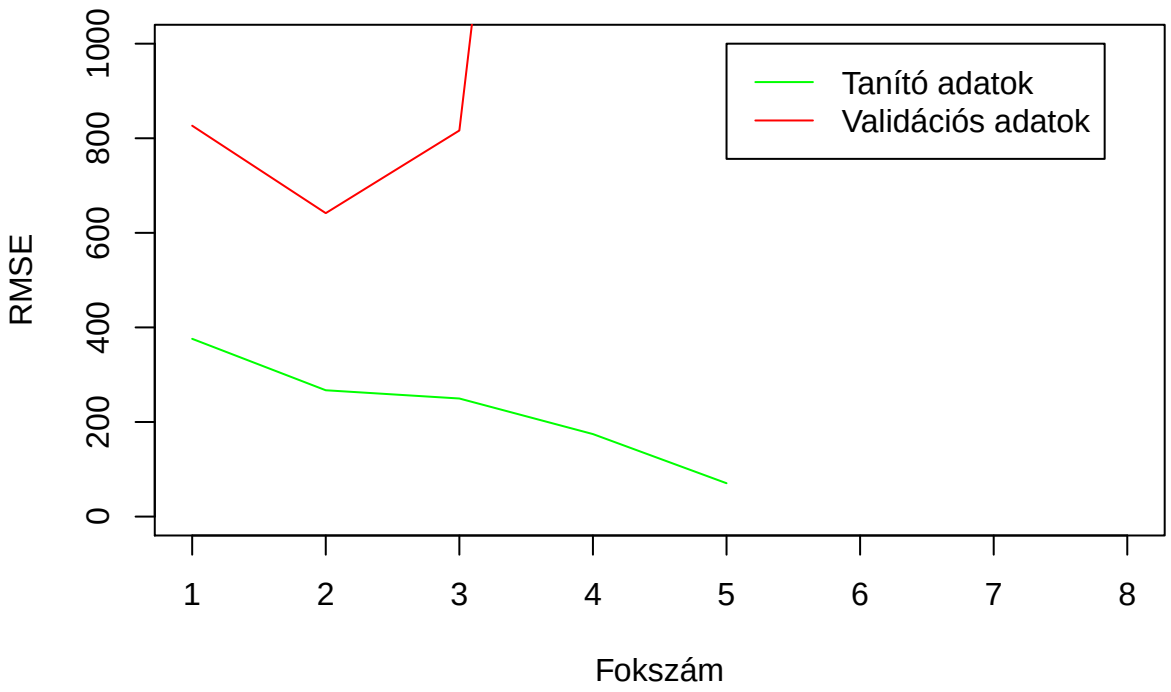


K-fold cross-validation

A K-fold keresztvalidáció esetén az adathalmazt összekevrjük véletlenszerűen, majd az adathalmazt K egyenlő részre osztjuk. Ebből K lépésben minden egyes tanításra $K-1$ részt tanító adatnak, és 1 részt validációs adathalmaznak használunk. Tekintsük például a polinomillesztés esetére:

```
set.seed(1)
idx<-sample(10)
shuffled<-df[idx,]
```

```r
K<-3
grid<-expand.grid(fold=1:K,degree=1:8)
out<-apply(grid,1,function(row) {
  valIdx<-seq(ceiling((row['fold']-1)*10/K), floor(row['fold']*10/K))
  validate(row['degree'],(1:10)[-valIdx],shuffled)
})
res<-aggregate(t(out),list(degree=grid$degree),mean)

plot(res$V1,col='green',type='l',ylim=c(0,1000),xlab='Fokszám',ylab='RMSE')
lines(res$V2,col='red')

legend(5,1000,legend=c('Tanító adatok','Validációs adatok'),col=c('green','red'),lty=1)
```



Teszthalmaz, kiértékelés

Láttuk, hogy az adatelemzés során el kell végeznünk többféle modell illesztését az adatokra. Az egyes modellek illesztése során az adathalmazt tanító és validációs adathalmazra osztottuk, s míg az előzőt a modell paramétereinek becslésére, addig a validációs adatokat a hiperparaméterek becsléséhez használtuk. A valóságban azonban mindig elkülönítünk egy harmadik, úgynevezett **teszt adathalmazt**, amelyen a kiválasztott modell kiértékelését végezzük.

Felmerül a kérdés, hogy erre miért van szükség, miért nem a teljes adathalmazon értékeljük ki a modellt. Ehhez fontos észrevennünk, hogy a paraméterek és hiperparaméterek meghatározása esetében mind a tanító, mint a validációs adathalmazt felhasználtuk, és így a modell paraméterei és hiperparaméterei ezekre az adatokra illeszkednek. Ahhoz, hogy *objektív* véleményt tudjunk formálni a modell minőségéről, célszerű azt egy a *modell által soha sem látott* adathalmazra tesztelni, hogy a modell *általánosíthatóságáról* képet kapjunk.

Hogy a tesztelést hatékonyan el tudjuk végezni, kiemelt fontosságú feladat, hogy a teljes adatelemzési feladat megkezdése előtt az adatokból egy részt elhatároljunk tesztelési célra, melyet tulajdonképpen *félreteszünk*, és az adatelemzés során nem használunk. Ha az általánosíthatóságot biztosítani akarjuk, a tesztadatokat még az adathalmaz áttekintése során sem használjuk fel. Persze, a teszthalmaz kiválasztásához ismernünk kell magát az adathalmazt, hiszen a kiválasztás során biztosítani kell, hogy a tesztadatok és a tanító/validációs

adatok **függetlenek** legyenek. Ettől eltekintve azonban a tesztadatok a modellépítés és kiválasztás során nem vesznek részt.

Egyszerű példa

Tudjuk, hogy egy gépjármű fékútja a sebességgel négyzetesen arányos. Tekintsük a `cars` beépített adathalmazt, amely gépjárművel sebesség és fékút összefüggését tartalmazza:

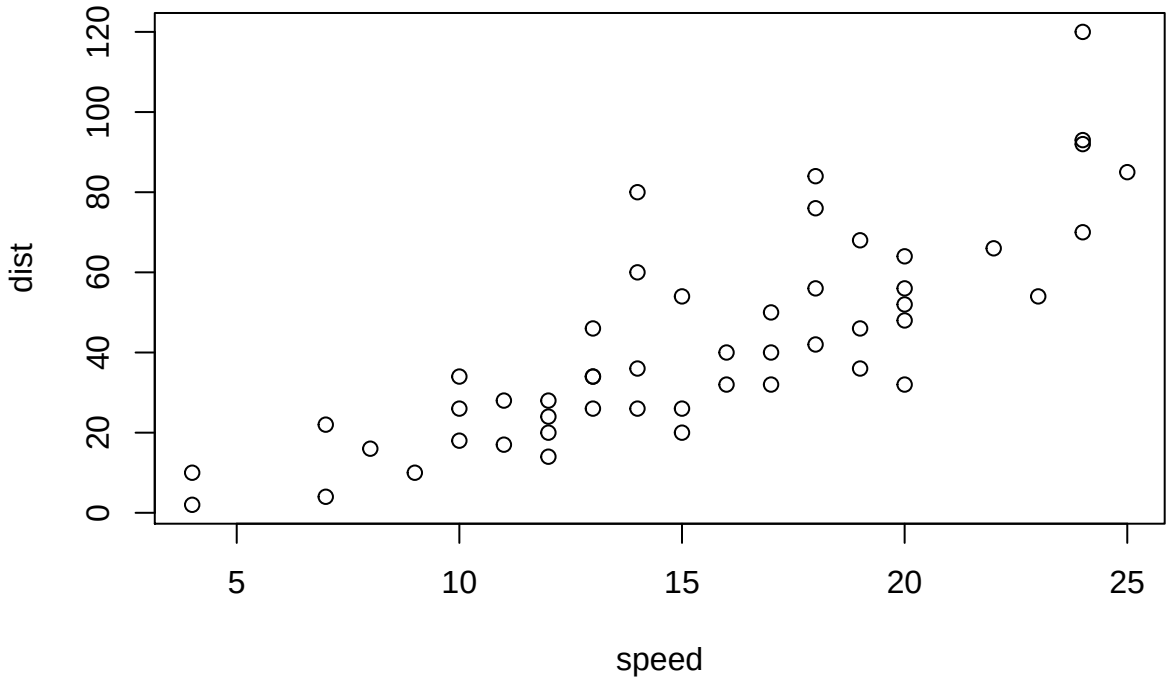
```
summary(cars)
```

```
##      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
```

```
head(cars)
```

```
##   speed dist
## 1     4    2
## 2     4   10
## 3     7    4
## 4     7   22
## 5     8   16
## 6     9   10
```

```
plot(cars)
```



Illesszünk maximum 10-ed fokú polinomot az adatokra, és bootstrapping módszerrel (50 adathalmazzal) válasszuk ki az optimális fokszámot!

```r
set.seed(1)
grid<-expand.grid(bsample=1:50,degree=1:10)
out<-apply(grid,1,function(row) {
    idx<-sample(nrow(cars),replace=T)
    train<-cars[idx,]
    valid<-cars[-idx,]
    m<-lm(dist~poly(speed,row['degree']),train)
    pred<-predict(m,valid)

    sqrt(mean((pred-valid$dist)^2))
})

res<-tapply(out,grid$degree,mean)
print(res)
```

```
##           1           2           3           4           5           6
##    15.95971    15.45169    16.24611    16.79919    19.05681    31.31570
##           7           8           9          10
##    35.86746    77.36374   274.38433 19823.16297
```

Láthatjuk, hogy a másodfokú polinom átlagos négyzetes eltérése a legkisebb, de az adatok alapján maximum 5 fokú polinom illeszthető hatékonyan az adatokra. Érdemes megfigyelni, hogy a hiba a 6-dik foktól kezdve meredeken emelkedni kezd!

Gyakorló feladatok

- Végezzük el a validációs eljárást a bemutatott `cars` adathalmazra, különböző keresztvalidációs módszerekkel!