Statističko učenje 23./24.
Prva domaća zadaća

Rok za predaju: **3. studenog, 2023.**

Broj bodova: 10

Teorijski dio

Napomena: *Dovoljno je riješiti jedan teorijski zadatak po izboru. U tom slučaju preostali zadatak služi kao vježba za završni ispit.*

Zadatak 1 (Odnos između pristranosti i varijance)

Pretpostavimo da je model $Y = f(X) + \epsilon$ pri čemu je $\mathbb{E}[\epsilon] = 0$, $\mathrm{Var}(\epsilon) = \sigma^2$ te ϵ nezavisna od $X \in \mathbb{R}^p$, a $f : \mathbb{R}^p \to \mathbb{R}$ regresijska funkcija. Nadalje, pretpostavite da su $X^{(i)} = x^{(i)}$, $i = 1, \ldots, n$ neslučajni, tj. da u skupu za učenje $T = \{(x^{(i)}, Y_i) : i = 1, \ldots, n\}$ jedina slučajnost dolazi od Y_i-eva (odnosno od ϵ_i-eva) koji su njd.

Ako je $\hat{f}_k = \hat{f}_k(T)$ procjenitelj za f dobiven kNN regresijom, pokažite da je

$$\mathbb{E}_T[L_x(\hat{f}_k)] = \sigma^2 + \frac{\sigma^2}{k} + \left(f(x) - \frac{1}{k} \sum_{x^{(i)} \in N_k(x)} f(x^{(i)}) \right)^2 ,$$

pri čemu je za proizvoljnu funkciju $g : \mathbb{R}^p \to \mathbb{R}$, $L_x(g) := \mathbb{E}[(Y - g(x))^2 \mid X = x]$, a $N_k(x)$ skup od k najbližih susjeda od x u $\{x^1, \ldots, x^{(n)}\}$, za sve $x \in \mathbb{R}^p$. (*Napomena:* I kada su $X^{(i)}$-evi neslučajni vrijedi dekompozicija greške iz Propozicije 1.1 s predavanja).

Objasnite kako hiperparametar $k \in \{1, \ldots, n\}$ kontrolira odnos između pristranosti i varijance u ovom slučaju (uz pretpostavku da je f dovoljno glatka).

Zadatak 2 (LOOCV)

(i) Pretpostavite da dani algoritam, koji skupu za učenje τ pridružuje $\hat{f}$, zadovoljava

$$\hat{\boldsymbol{y}} := (\hat{f}(x^{(1)}), \ldots, \hat{f}(x^{(n)}))^\tau = S\boldsymbol{y} \,, \tag{1}$$

pri čemu je $S = (S_{ij} : i, j = 1, \ldots, n) \in \mathbb{R}^{n \times n}$ matrica koja ne ovisi o $\boldsymbol{y}$ (ali može of $x^{(i)}$-evima). Neka je za svaki $i = 1, \ldots, n$, $\hat{f}^{-i}$ funkcija dobivena kada se algoritam primijeni na $\tau \setminus \{(x^{(i)}, y_i)\}$.

Ako algoritam za svaki $i = 1, \ldots, n$, zadovoljava

$$\hat{f}^{-i}(x^{(i)}) = \sum_{j \neq i} \frac{S_{ij}}{1 - S_{ii}} y_j \,, \tag{2}$$

pokažite da LOOCV greška zadovoljava

$$L_{CV}^{(n)}(\hat{f}) := \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}^{-i}(x^{(i)}))^2 = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{f}(x^{(i)})}{1 - S_{ii}}\right)^2,$$

tj. za računanje greške $L_{CV}^{(n)}(\hat{f})$ nije potrebno računati funkcije $\hat{f}^{-i}$, $i = 1, \ldots, n$.

(ii) Pokažite da linearna regresija (uz metodu najmanjih kvadrata i pretpostavku da je $\boldsymbol{X}$ punog ranga) zadovoljava (2).

Uputa: Kao međukorak, pokažite da $\hat{f}^{-i}$ minimizira i sumu kvadrata reziduala kada za uzorak uzmemo skup $\{(x^{(j)}, y'_j) : j = 1, 2, \ldots, n\}$, pri čemu je

$$y'_j = \begin{cases} y_j, & j \neq i \\ \hat{f}^{-i}(x^{(i)}), & j = i, \end{cases}$$

te izrazite $\hat{f}^{-i}(x^{(i)})$ u terminima S_{ij} i y'_j, $j = 1, \ldots, n$.

Praktični dio

Zadatak 1 (Procjena testne greške)

Promatramo podatke `PimaIndiansDiabetes` iz paketa `mlbench`, a cilj je predvidjeti ima li ili ne pacijent dijabetes, na temelju nekoliko dijagnostičkih mjerenja; dakle, imamo problem **klasifikacije**. U dokumentaciji paketa pročitajte malo detaljnije o samim podacima.

Koristeći sljedeći kod učitajte podatke te ih na slučajan način podijelite u skup za trening i testni skup, i to na dva jednaka dijela.

```
# install.packages("mlbench")
library(mlbench)
data(PimaIndiansDiabetes)

set.seed(2)  ## koristite ovaj seed
trainid = sample(1:nrow(PimaIndiansDiabetes), nrow(PimaIndiansDiabetes)/2)
Diab.train = PimaIndiansDiabetes[trainid, ]
Diab.test = PimaIndiansDiabetes[-trainid, ]
```

Koristit ćemo kNN metodu za klasifikaciju; pretpostavimo da, kao u ovom primjeru, imamo samo dvije klase 0 i 1. Dakle, za dani $k \in \{1, 2, \ldots, n\}$ te $x \in \mathbb{R}^p$, nađemo k najbližih susjeda od x u $\{x^{(i)} : i = 1, 2, \ldots, n\}$, pogledamo njihove klase (tj. y_i-eve), te stavimo da je $\hat{f}_k(x)$ klasa koja se najčešće pojavlju među tih k podataka (ako postoji jednak broj 0la i 1ca, slučajno odaberemo klasu).

U nastavku koristimo $k \in \{1, 2, 3, \ldots, 30\}$.

(i) Prilagodite kNN modele na `Diab.train`, izračunajte grešku na skupu za trening, te procijenite testnu grešku koristeći `Diab.test`; koristimo $0-1$ funkciju gubitka. Prikažite dobivene greške s obzirom na vrijednosti hiperparametra k – pazite da ih prikladno obojate i dodate legendu. Jesu li dobiveni rezultati u skladu s onim što očekujemo s obzirom na odnos između pristranosti i varijance? Koji se k čini optimalan? Kolika je procjena testne greške za taj optimalni model? *Uputa*: Pripazite da odziv zadate kao faktor (npr. koristeći funkciju `as.factor`).

(ii) Iako je metoda iz dijela (i) dobra za odabir "najboljeg" modela, rezultirajuća procjena greške tipično nije neće biti dobra procjena. Kako bismo to razumjeli ponovit ćemo postupak iz dijela (i) više puta. Točnije, $M = 30$ puta na slučajan način podijelite skup podataka na dva jednaka dijela kao u početku, za svaki k prilagodite kNN model na skupu za trening te procijenite njegovu testnu grešku na testnom skupu; koristite `set.seed` tako da se vaši rezultati mogu reproducirati. Na taj način za svaki k dobivamo niz od M realizacija procjenitelja testne

greške za $\hat{f}_k$ (ovo nisu nezavisne realizacije, ali nisu ni potpuno zavisne). Prikažite *boxplotove* svih ovih uzoraka u ovisnosti o hiperparametru k (sve zajedno ona istom grafu). Naznačite "optimalni" k i procjenu testne greške za taj k iz dijela (i). Objasnite dobivene rezultate.

(iii) Kako bismo riješili ovaj problem, prilagodbu i odabir modela radimo na skupu za trening, a procjenu testne greške odabranog modela radimo na testnom skupu kojeg do tada nismo koristili. Jedan način za odabir modela u tom slučaju je "k-fold" unakrsna validacija (CV). Za provedbu CV metode koristan je paket `caret` kojeg možete (ali ne morate) koristiti u nastavku; u tom slučaju pročitajte kako se koriste funkcije `train` i `trainControl`, a više detalja možete naći na sljedećem linku

`https://topepo.github.io/caret/model-training-and-tuning.html`.

Vaš zadatak je prilagoditi kNN modele na `Diab.train` (isti kao u početku) te 3-CV metodom odabrati "optimalan" hiperparametar k; podjelu skupa `Diab.train` na dijelove treba napraviti na slučajan način (što funkcija `train` i radi).

- To da hoćemo koristiti CV metodu zadajemo koristeći funkciju `trainControl`.
- Kako bi zadali niz vrijednosti za hiperparametar, koristite naredbu `expand.grid(k = c(1:30))`.

Procjenu greške odabranog modela napravite na `Diab.test`, te tu procjenu kao i odabrani k naznačite na slici iz dijela (ii). Komentirajte rezultate – čini li se ovaj pristup bolji nego onaj iz (i)?

(iv) Ponovite postupak iz dijela (ii) pri čemu ćete uzorak umjesto na dva jednaka dijela, dijeliti tako da 2/3 bude skup za trening, a preostala 1/3 testni skup, te nacrtajte *boxplotove* kao u dijelu (ii) (nije potrebno ništa dodatno označavati). Usporedite sa slučajem iz dijela (ii). Kako se promijenila distribucija procjenitelja testne greške? Očekujemo li da ćemo ako sada odaberemo model s najmanjom procjenom testne greške dobiti bolji model?

Zadatak 2 (kNN metoda i prokletstvo dimenzionalnosti)

Promatramo model

$$Y = 2X_1^2 + 3X_2^2 - X_3^2 + \epsilon\,, \tag{3}$$

pri čemu su X_1, X_2, X_3 njd $\mathrm{N}(0,1)$ slučajne varijable, a $\epsilon \sim \mathrm{N}(0,1)$ nezavisna od X-eva, i to u tri slučaja:

(i) $p = 3$, tj. X_1, X_2, X_3 su jedine kovarijate.

(ii) $p = 100$, a $X_4, \ldots, X_{100}$ su njd $\mathrm{N}(0,1)$ **nezavisne** od X_1, X_2, X_3.

(iii) $p = 100$, a $X_4, \ldots, X_{100}$ su linearne kombinacija kovarijata X_1, X_2, X_3, tj.

$$(X_4, \ldots, X_{100})^\tau := A(X_1, \ldots, X_t)^\tau \in \mathbb{R}^{97\times 1}\,,$$

pri čemu je $A \in \mathbb{R}^{97\times 3}$ matrica čiji su elementi njd uniformne slučajne varijable na $[0, 1]$ (A generirajte samo jednom).

Dakle, regresijska funkcija u sva tri slučaja je $f(x) = 2x_1^2+3x_2^2-x_3^3$ za $x = (x_1, \ldots, x_p) \in \mathbb{R}^p$.

Za $M = 200$[1], M puta ponovite sljedeći algoritam:

1. Generirajte uzorak $T = \tau$ duljine $n = 100$ za vektor $(X, Y) \in \mathbb{R}^{p+1}$ u sva tri slučaja, pri čemu u druga dva slučaja koristite kovarijate iz slučaja (i). Neka $\hat{f}_k^{(i)}$ označava procijenjenu regresijsku funkciju iz kNN regresije za i-ti slučaj, pri čemu je $i = 1, 2, 3$, $k = 1, \ldots, n-1$.
2. Izračunajte $\hat{f}_k^{(i)}(0)$ za sve i, k pri čemu, ovisno o p, 0 označava nul-vektor u $\mathbb{R}^p$.
3. Generirajte testni skup duljine $N = 200$ kao u prethodnom koraku, ali uvjetno na $X = 0$ (dakle $Y = f(0) + \epsilon = \epsilon$), te procijenite testnu grešku $L_0(\hat{f}_k^{(i)}) = \mathbb{E}[(Y - \hat{f}_k^{(i)}(0))^2 \mid X = 0]$ u 0, za sve i, k.

Na ovaj način ćete za sve kombinacije i, k dobiti uzorak duljine M za slučajne varijable $\hat{f}_k^{(i)}(0)$ i $L_0(\hat{f}_k^{(i)})$ (pri čemu slučajnost dolazi od skupa za trening T). Koristeći ove uzorke, procijenite očekivanu testnu grešku $\mathbb{E}_T[L_0(\hat{f}_k^{(i)})]$, pristranost $(\mathbb{E}_T[\hat{f}_k^{(i)}(0)] - f(0))^2$ te varijancu $\mathrm{Var}_T(\hat{f}_k^{(i)}(0))$.

Za svaki slučaj i, prikažite dobivene rezultate u ovisnosti o hiperparametru k, te usporedite sva tri slučaja. U kojim slučajevima kNN metoda radi bolje? Zašto? U kojem slučaju imamo "prokletstvo dimenzionalnosti"?

[1]Kod simulacijskih studija, tj. Monte Carlo metoda, preciznije rezultate dobivamo što uzimamo veći broj ponavljanja. Ukoliko je $M = 200$ previše za vaša računalo, uzmite manji M, ali ako je moguće, uzmite i veći M.