Compararea modelelor
Cursul 9

Programare şi modelare probabilistă - anul III

Facultatea de Informatică, UAIC

e-mail: adrian.zalinescu@uaic.ro

web: https://sites.google.com/view/fiicoursepmp/home

11 Decembrie 2023

Pâna acum, am abordat problema inferenţei, adică cum să deducem parametrii din date.

În cele ce urmează ne vom concentra asupra comparaţiei a două sau mai multe modele care sunt folosite pentru a explica aceleaşi date.
Aceasta nu este o problemă simplă, dar este în acelaşi timp este esenţială în analiza datelor.

Plan

1. Verificări predictive a posteriori
2. Briciul lui Occam
   - Simplitatea şi acurateţea
   - Overfitting şi underfitting
3. Criterii bazate pe informaţie
4. Regularizarea distribuţiilor a priori

Verificări predictive a posteriori

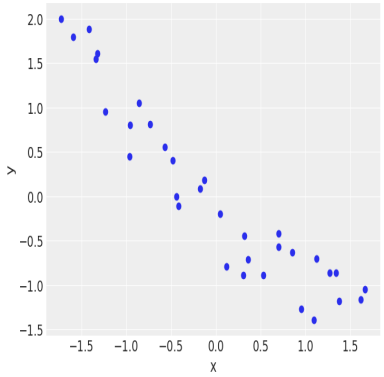
Distribuţia predictivă a posteriori a fost folosită pentru a verifica cât de bine explică modelele datele folosite.

- Prin aceste verificări, sperăm să observăm limitările unui model, fie ca modalitate de a-l înţelege, fie ca o încercare de a-l îmbunătăţi.
- În mod implicit, se înţelege că modelele nu vor reproduce în general toate aspectele unei probleme (la fel de bine).
  - Acest lucru nu pune probleme, deoarece modelele sunt construite cu un scop în minte.
  - O verificare predictivă a posteriori este un mod de a evalua modelul în contextul acestui scop.
  - Astfel, dacă avem mai multe modele, putem efectua acest tip de verificare pentru a le compara.

Exemplu

Generarea datelor

```
import pymc3 as pm
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt
import arviz as az
az.style.use('arviz-darkgrid')

dummy_data = np.loadtxt('./data/dummy.csv')
x_1 = dummy_data[:, 0]
y_1 = dummy_data[:, 1]
order = 2
x_1p = np.vstack([x_1**i for i
                  in range(1, order+1)])
x_1s = (x_1p - x_1p.mean(axis=1, keepdims=True))/
    x_1p.std(axis=1, keepdims=True)
y_1s = (y_1 - y_1.mean()) / y_1.std()
plt.scatter(x_1s[0], y_1s)
plt.xlabel('x')
plt.ylabel('y')
```



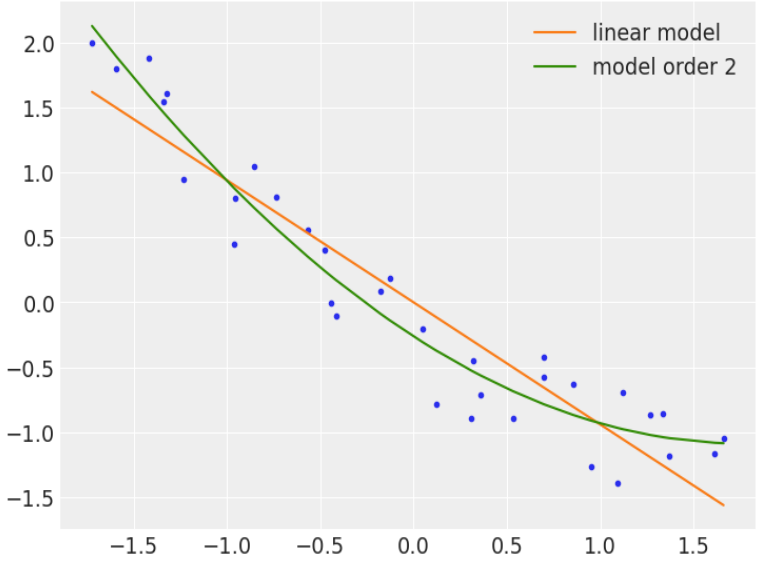
Acum, vom încerca să calibrăm aceste date cu două modele diferite (nu foarte tare), unul liniar şi unul polinomial de ordinul 2 (cunoscut drept *modelul parabolic* sau *modelul pătratic*):

```
with pm.Model() as model_l:
    α = pm.Normal('α', mu=0, sigma=1)
    β = pm.Normal('β', mu=0, sigma=10)
    ϵ = pm.HalfNormal('ϵ', 5)
    μ = α + β * x_1s[0]
    y_pred = pm.Normal('y_pred', mu=μ, sigma=ϵ, observed=y_1s)
    idata_l = pm.sample(2000, return_inferencedata=True)

with pm.Model() as model_p:
    α = pm.Normal('α', mu=0, sigma=1)
    β = pm.Normal('β', mu=0, sigma=10, shape=order)
    ϵ = pm.HalfNormal('ϵ', 5)
    μ = α + pm.math.dot(β, x_1s)
    y_pred = pm.Normal('y_pred', mu=μ, sigma=ϵ, observed=y_1s)
    idata_p = pm.sample(2000, return_inferencedata=True)
```

Acum, vom reprezenta grafic mediile pentru cele două modele:

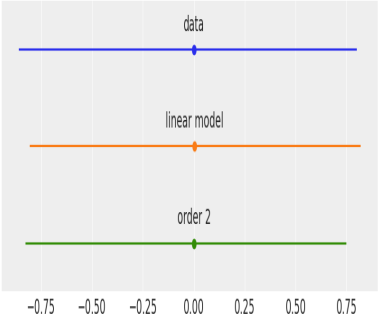
```
x_new = np.linspace(x_1s[0].min(), x_1s[0].max(), 100)

α_l_post = idata_l.posterior['α'].mean(("chain", "draw")).values
β_l_post = idata_l.posterior['β'].mean(("chain", "draw")).values
y_l_post = α_l_post + β_l_post *  x_new

plt.plot(x_new, y_l_post, 'C1', label='linear model')

α_p_post = idata_p.posterior['α'].mean(("chain", "draw")).values
β_p_post = idata_p.posterior['β'].mean(("chain", "draw")).values
idx = np.argsort(x_1s[0])
y_p_post = α_p_post + np.dot(β_p_post, x_1s)

plt.plot(x_1s[0][idx], y_p_post[idx], 'C2', label=f'model order {order}')

plt.scatter(x_1s[0], y_1s, c='C0', marker='.')
plt.legend()
```

Modelul pătratic pare mai bun, dar cel liniar se comportă şi el bine:



Verificările predictive a posteriori:

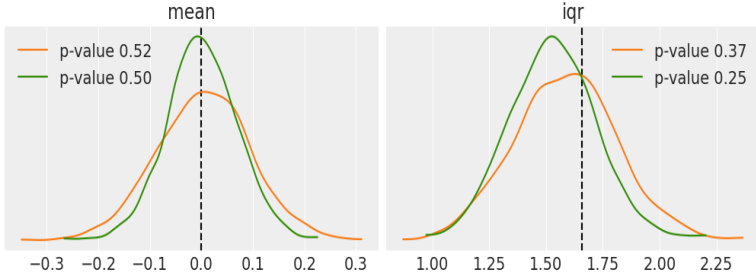
```
pm.sample_posterior_predictive(idata_1, model=model_1, extend_inferencedata=True)
pm.sample_posterior_predictive(idata_p, model=model_p, extend_inferencedata=True)

y_l = idata_1.posterior_predictive['y_pred']
y_p = idata_p.posterior_predictive['y_pred']

plt.figure(figsize=(8, 3))
data = [y_1s, y_l, y_p]
labels = ['data', 'linear model', 'order 2']
for i, d in enumerate(data):
mean = d.mean()
err = np.percentile(d, [25, 75])
plt.errorbar(mean,-i,xerr=[[-err[0]],[err[1]]],
                                        fmt='o')
plt.text(mean, -i+0.2, labels[i], ha='center',
                                fontsize=14)
plt.ylim([-i-0.5, 0.5])
plt.yticks([])
```



- Această diagramă arată media şi intervalul intercvartilic (intervalul $[Q_1, Q_3]$, unde Q_i este a i-a cvartilă) pentru date şi fiecare model.
- Media şi cvartilele se determină după eşantioane provenite din distribuţia predictivă a posteriori pentru fiecare model.

Putem de asemenea reprezenta distribuţiile pentru medii şi lungimile intervalelor intercvartilice, pentru a le compara:

```
fig, ax = plt.subplots(1, 2, figsize=(10, 3), constrained_layout=True)
def iqr(x, a=0):
    return np.subtract(*np.percentile(x, [75, 25], axis=a))
y_l_ = y_l.stack(samples=('chain', 'draw')).values.T
y_p_ = y_p.stack(samples=('chain', 'draw')).values.T
for idx, func in enumerate([np.mean, iqr]):
    T_obs = func(y_1s)
    ax[idx].axvline(T_obs, 0, 1, color='k', ls='--')
    for d_sim, c in zip([y_l_, y_p_], ['C1', 'C2']):
        T_sim = func(d_sim, 1)
        p_value = np.mean(T_sim >= T_obs)
        az.plot_kde(T_sim, plot_kwargs={'color': c},
                    label=f'p-value {p_value:.2f}', ax=ax[idx])
    ax[idx].set_title(func.__name__)
    ax[idx].set_yticks([])
    ax[idx].legend()
```



- Linia punctată reprezintă statistica provenită din date (pentru medie şi $IQR = Q_3 - Q_1$).
- Curbele (cu acelaşi cod de culoare ca pentru figura precedentă) reprezintă distribuţiile mediilor (stânga) şi ale IQR (dreapta), care au fost deduse din eşantioane provenite din distribuţia predictivă a posteriori.
- Valoarea p reprezintă proporţia datelor simulate din statistica dată (media sau IQR) care sunt mai mari sau egale cu cea provenită din date.
- Dacă datele şi simulările sunt în concordanţă, ar trebui să ne aşteptăm la valori p apropiate de 0.5, altfel suntem în prezenţa unei distribuţii predictive a posteriori deplasate.

Briciul lui Occam

Simplitatea şi acurateţea

Atunci când alegem dintre alternative, există un principiu de ghidare cunoscut drept *briciul lui Occam*:

- dacă există două sau mai multe explicaţii echivalente pentru acelaşi fenomen, o vom alege pe cea mai simplă, deoarece
- modelele simple au avantajul de a fi mai uşor de înţeles decât cele complexe.

Un alt factor pe care vrem să-l luăm în evidenţă este acurateţea modelului, adică cât de bine acesta se potriveşte datelor. Verificarea predictivă a posteriori este bazată pe această idee de acurateţe.

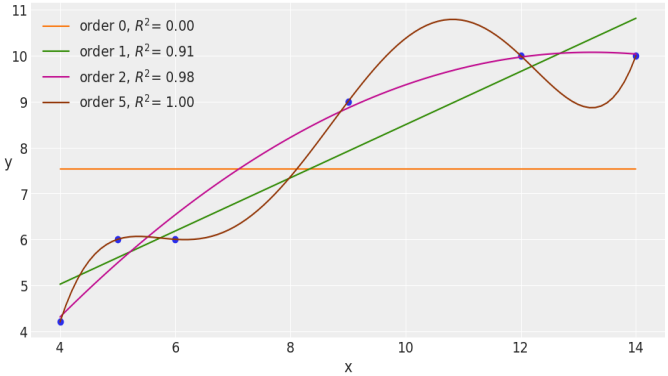
- Astfel, intuitiv, vom tinde să alegem modelele care au o acurateţe mai mare şi sunt în acelaşi timp simple.
- Dar dacă cel mai simplu model are cea mai mică acurateţe?
- Sau, mai general, cum putem împăca cele două tendinţe: simplitatea şi acurateţea?

Exemplu

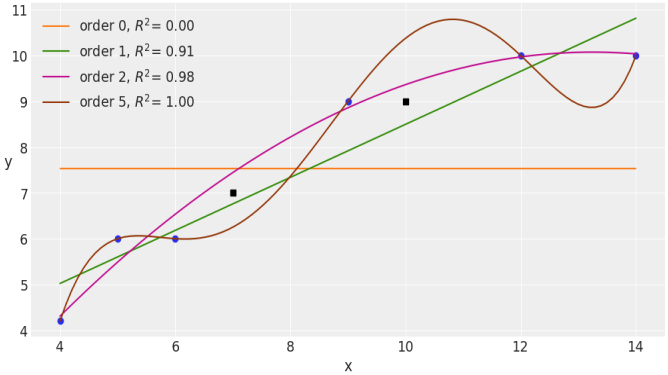
Vom încerca să calibrăm polinoame cu grad din ce în ce mai mare pe un set de date foarte simplu.

- Pentru că exemplul este doar explicativ, nu vom mai face apel la inferenţa Bayesiană, ci la metoda celor mai mici pătrate (care corespunde unui model Bayesian cu distribuţii a priori plate):

```
x = np.array([4., 5., 6., 9., 12, 14.])
y = np.array([4.2, 6., 6., 9., 10, 10.])
plt.figure(figsize=(10, 5))
order = [0, 1, 2, 5]
plt.plot(x, y, 'o')
for i in order:
    x_n = np.linspace(x.min(), x.max(), 100)
    coeffs = np.polyfit(x, y, deg=i)
    ffit = np.polyval(coeffs, x_n)
    p = np.poly1d(coeffs)
    yhat = p(x)
    ybar = np.mean(y)
    ssreg = np.sum((yhat-ybar)**2)
    sstot = np.sum((y - ybar)**2)
    r2 = ssreg / sstot
    plt.plot(x_n,ffit,label=f'order{i},
                $R^2$= {r2:.2f}')
plt.legend(loc=2)
plt.xlabel('x')
plt.ylabel('y', rotation=0)
```



- Observăm că acurateţea modelului, dată de coeficientul de determinare R^2, este acompaniată de complexitatea acestuia.
- se observă, de altfel, că polinomul de grad 5 se potriveşte perfect datelor!



- Observăm că acurateţea modelului, dată de coeficientul de determinare R^2, este acompaniată de complexitatea acestuia.
- se observă, de altfel, că polinomul de grad 5 se potriveşte perfect datelor!
- să presupunem că ulterior, adăugăm două puncte, $(7,7)$ şi $(10,9)$ la setul nostru de date;
- se observă că modelul de ordin 5 nu este bun pentru a generaliza date viitoare, încă neobservate, dar potenţial observabile.

Overfitting şi underfitting

Când un model se potriveşte prea bine cu setul de date care a fost folosit pentru a-i învăţa parametrii, dar se potriveşte rău cu alte seturi de date, spunem că avem *overfitting*.

- Acesta este un fenomen general întâlnit în Machine Learning şi în Statistică.
- O modalitate utilă de a vizualiza overfitting-ul este de a ne imagina că seturile de date au două componente: *semnalul* şi *zgomotul*.
  - semnalul este ceea ce vrem să învăţăm din date;
  - presupunem că datele conţin semnal, altfel demersul ar fi inutil;
  - pe de altă parte, zgomotul este inutil şi este rezultatul erorilor, limitărilor modului în care datele au fost generate sau capturate, datelor corupte, etc.
  - un model realizează overfitting dacă el învaţă din zgomot, ascunzând semnalul.

- Aceasta este o justificare practică pentru briciul lui Occam: prea multă complexitate duce la overfitting;
  - în principiu, putem oricând crea un model atât de complex şi care să justifice totul în cel mai mic detaliu.

Underfitting

- În exemplul de mai sus, avem modelul de grad 0 (o constantă);
- Acest model captează doar media variabilei de output y, fiind complet independent de variabila de input x.
- Spunem că în cadrul acestui model se manifestă fenomenul de *underfitting*.

Balanţa între simplitate şi acurateţe

O variantă a briciului lui Occam:

> "Lucrurile ar trebui să fie cât mai simple cu putinţă, dar nu mai simple" – Einstein

Ideal, ar trebui să avem un model în care nu avem nici overfitting, nici underfitting.
Aşadar, va trebui să facem un compromis între simplitate şi acurateţe.
Acest compromis este discutat de obicei în termeni de *varianţă* şi *tendenţiozitate* (*bias*):

- O *tendenţiozitate mare* este rezultatul unui model ce are o capacitate mică de a se potrivi datelor, şi astfel poate duce la underfitting.
- O *varianţă mare* este rezultatul unui model ce are o sensibilitate mare la detaliile din date. Acest lucru poate cauza captarea zgomotului în date şi duce astfel la overfitting.

În exemplul de mai sus, modelul de grad 0 este cel cu tendenţiozitatea cea mai mare, deoarece are *tendinţa* de a returna o linie orizontală ce nu depinde de valoarea lui x.

Modelul de grad 5 este cel cu cea mai mare varianţă: (aproape) oricum am aranja setul de date de 6 puncte, modelul se va adapta acestuia.

Măsuri pentru acurateţea predictivă

Pentru a ţine cont de acurateţe, pe de o parte, şi de simplitate, pe de alta, introducem mai multe concepte:

- *acurateţea în cadrul eşantionului*: este acurateţea măsurată pe *eşantion*, adică datele folosite pentru calibrarea modelului;
- *acurateţea în afara eşantionului*: este acurateţea modelului măsurată pe date care nu fac parte din eşantion – se mai numeşte şi *acurateţea predictivă*.

- În medie, acurateţea în cadrul eşantionului este mai mare decât cea din afara lui.
- Folosind-o doar pe prima, am putea crede că avem un model mai bun decât este în realitate.
- De aceea, acurateţea predictivă este preferată celei din cadrul eşantionului.

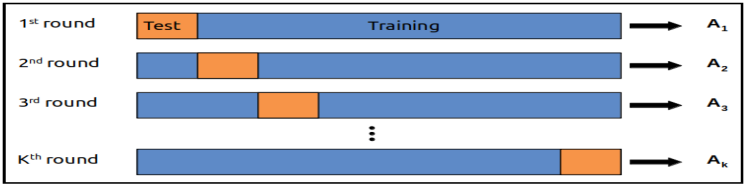
- Pe de altă parte, trebuie să ne permitem să lăsăm pentru testarea modelului, şi nu calibrarea lui, o parte din datele avute la dispoziţie, ceea ce reprezintă un lux pentru mulţi analişti.
- Pentru a trece peste această problemă, s-au creat metode pentru a estima acurateţea predictivă folosind numai datele din eşantion. Două din aceste metode sunt:

- *Cross-validation:* este o strategie empirică bazată pe divizarea datelor disponibile în subgrupuri folosite pentru calibrare şi evaluare într-un mod alternativ.
- *Criteriul informaţiei:* este un termen generic pentru mai multe expresii relativ simple ce pot fi considerate ca moduri de aproximare a rezultatelor pe care le-am fi putut obţine prin cross-validation.

Cross validation

Acest proces este rezumat de următoarea schemă:

- Datele se partiţionează în K porţiuni, ce se încearcă a fi mai mult sau mai puţin egale (în mărime sau alte caracteristici).
- Folosim apoi $K-1$ dintre ele pentru a antrena (calibra) modelul, iar porţiunea rămasă pentru a-l testa.
- Repetăm această procedură în mod sistematic, lăsând deoparte altă porţiune, până completăm K runde:



- Se face apoi media rezultatelor, A_k, $1 \leq k \leq K$; aceasta este cunoscută drept *K-fold cross validation*.

- În cazul în care K este mărimea setului de date, obţinem *leave-one-out cross validation* (*LOO-CV*).
- Cross-validation este o idee simplă şi folositoare, dar pentru unele modele sau pentru date foarte mari, costul computaţional al acesteia poate fi prea mare.
- În cele ce urmează prezentăm alternative pentru estimarea rezultatelor obţinute prin cross-validation.

Criterii bazate pe informaţie

Acestea sunt colecţii de unelte distincte, dar interconectate, ce sunt folosite pentru a compara modelele în funcţie de cum se potrivesc datelor, în timp ce ţin cont de complexitatea lor printr-un factor de penalizare.
Astfel, aceste criterii formalizează intuiţia dezvoltată anterior, aceea de a face un compromis între acurateţe şi simplitate.

Log-verosimilitatea şi devianţa

Un mod intuitiv de a măsura cât de bine se potriveşte datelor este de a calcula *eroarea medie pătratică* între date şi predicţiile făcute de model:

$$\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \mathbb{E}[y_i \mid \theta]\right)^2,$$

unde $\mathbb{E}[y_i \mid \theta]$ este valoarea prezisă, dat fiind parametrul estimat θ.
O măsură mai generală este *log-verosimilitatea*:

$$\sum_{i=1}^{n}\log p(y_i \mid \theta).$$

Atunci când verosimiliatea, $p(\cdot \mid \theta)$, este distribuită normal, aceasta este proporţională cu eroarea medie pătratică.

Din motive istorice, se mai foloseşte o măsură, numită *devianţă*:

$$-2\sum_{i=1}^{n}\log p(y_i \mid \theta).$$

- Cu cât aceasta este mai mică, cu atât este mai mare log-verosimilitatea şi deci acurateţea în cadrul eşantionului;
- Aşadar, modelele complexe vor avea în general o devianţă mai mică decât modelele simple. Prin urmare, trebuie inclus un termen de penalizare pentru complexitate.
- În cele ce urmează, vom utiliza diverse criterii bazate pe informaţie, ce au în comun faptul că folosesc devianţa şi un termen de penalizare.

Criteriul Akaike

Este definit ca:

$$AIC := -2\sum_{i=1}^{n} \log p(y_i \mid \hat{\theta}_{mle}) + 2p_{AIC}, \text{ unde}$$

- p_{AIC} este numărul de parametri, iar
- $\hat{\theta}_{mle}$ este *verosimilitatea maximă* (*MLE*):

$$\hat{\theta}_{mle} = \text{argmax}_{\theta} \prod_{i=1}^{n} p(y_i \mid \theta).$$

  - aceasta este folosită în mod curent de non-Bayesieni şi coincide cu *maximul a posteriori* (*MAP*) dacă se folosesc distribuţii a priori plate.
  - Astfel, acest criteriu este mai util în abordările non-Bayesiene, dar datorită faptului că nu foloseşte distribuţia a posteriori, este problematic altfel.

Criteriul WAIC (widely applicable information criterion)

Este versiunea Bayesiană a criteriului *AIC* şi, la fel, are doi termeni:

$$\begin{aligned} WAIC &= -2\sum_{i=1}^{n}\log\left(\frac{1}{S}\sum_{s=1}^{S}p(y_i \mid \theta^s)\right) + 2\sum_{i=1}^{n}\bigvee_{s=1}^{S}\log p(y_i \mid \theta^s) \\ &= -2lppd + 2p_{WAIC}. \end{aligned}$$

- Termenul *lppd* (*log point-wise predictive density*) calculează verosimilitatea medie pe un eşantion de mărime S din distribuţia a posteriori; este ceea ce numim *devianţă* în cadrul Bayesian.
- Al doilea termen, p_{WAIC}, calculează varianţa log-versomilităţii pe eşantionul de mărime S; este un factor ce penalizează complexitatea: cu cât mai mare e numărul de parametri efectivi, cu atât mai mare deviaţia distribuţiei a posteriori va fi.

Pareto smoothed importance sampling LOO-CV

Este o metodă care aproximează LOO-CV, dar fără a efectua cele K iteraţii.

- Ideea de bază este că este posibil să aproximăm LOO-CV prin re-ponderarea potrivită a verosimilităţilor.
- Acest lucru se face folosind o tehnică foarte cunoscută în statistică, numită *importance sampling*.
- Problema acesteia este că este foarte instabilă, iar
- pentru a remedia acest lucru, o nouă metodă este folosită, anume *Pareto smoothed importance sampling*, ce poate fi utilizată pentru estimări mai stabile ale LOO.
- Interpretarea este similară *AIC* şi *WAIC*: cu cât este mai mică valoarea, cu atât mai bine.

Comparaţia modelelor cu PyMC

Aceasta se face simplu:

```
pm.compute_log_likelihood(idata_l,model=model_l)
waic_l = az.waic(idata_l, scale="deviance")
waic_l
```

```
Computed from 8000 posterior samples and 33 observations log-likelihood matrix.

               Estimate       SE
deviance_waic    28.85     5.36
p_waic            2.51        -
```

Dacă dorim să calculăm PSIS-LOO-CV în loc de WAIC, trebuie să folosim `az.loo`
PyMC prezintă 3 valori:

- o estimare punctuală a WAIC;
- eroarea standard (`SE`) a estimării punctuale;
- numărul efectiv de parametri, `p_waic`.

Deoarece valorile WAIC/PSIS-LOO-CV sunt interpretate în mod relativ (fiecare în modelul propriu), ArviZ oferă două funcţii auxiliare pentru uşurarea comparaţiei:

- prima este `az.compare`:

```
pm.compute_log_likelihood(idata_p,model=model_p)
cmp_df = az.compare({'model_l':idata_l, 'model_p':idata_p},
                    method='BB-pseudo-BMA', ic="waic", scale="deviance")
cmp_df
```

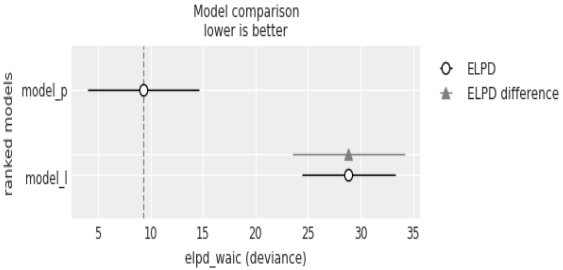
	rank	elpd_waic	p_waic	elpd_diff	weight	se	dse	warning	scale
model_p	0	9.340557	2.708534	0.000000	0.998944	5.395026	0.000000	False	deviance
model_l	1	28.853447	2.506985	19.512889	0.001056	4.421055	5.343268	False	deviance

  - primele două coloane conţin estimările punctuale ale WAIC, respectiv numărul efectiv de parametri, `p_waic`;
  - a treia coloană reprezintă diferenţa între valoarea WAIC a celui mai bun model şi restul;
  - câteodată, nu vrem doar să selectăm cel mai bun model, ci să facem predicţii, incluzându-le pe toate conform unor ponderi, ce sunt date în a patra coloană (şi reprezintă probabilitatea unui model, având în vedere datele);

- a cincea şi a şasea coloană reprezintă erorile standard ale estimărilor punctuale ale WAIC, respectiv erorile standard ale diferenţelor între valorile WAIC;
- a şaptea coloană arată dacă valoarea WAIC este fiabilă, caz în care `False` este indicat.

Putem de asemenea vizualiza grafic aceste informaţii, folosind:

az.plot_compare(cmp_df)



- cercurile goale reprezintă valorile WAIC, cea mai mică fiind de asemenea indicată cu o linie punctată verticală;
- liniile negre orizontale reprezintă valorile erorilor standard ale WAIC;
- pentru toate modelele cu excepţia celui mai bine plasat, avem un triunghi şi o linie gri orizontală care indică diferenţa între valoarea WAIC a celui mai bun model şi restul, respectiv eroarea standard a acestei diferenţe.

Regularizarea distribuţiilor a priori

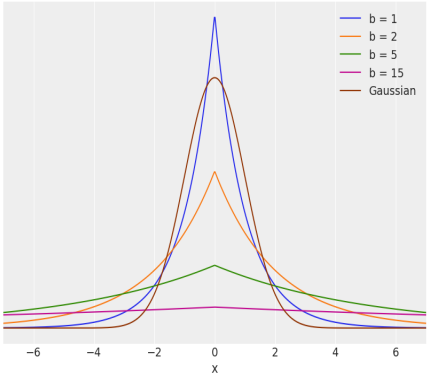
Folosirea distribuţiilor a priori informative sau slab informative este o modalitate de a introduce tendenţiozitate în model.

- Dacă este făcută riguros, acest lucru este benefic, deoarece previne fenomenul de overfitting şi astfel modelele vor fi capabile să facă predicţii bune.
- Această idee de a adăuga tendenţiozitate fără a afecta acurateţea modelului, se numeşte *regularizare*.
- Ea ia de obicei forma penalizării valorilor mari ale parametrilor din model;
- Astfel, se reduce informaţia pe care un model este capabil să o reprezinte, şi în felul acesta reduc şansele ca modelul să capteze zgomotul în locul semnalului.

- În statistica non-Bayesiană, această idee ia forma a două modificări a metodei celor mai mici pătrate, numite *regresia de creastă* (*ridge regression*) şi *regresia Lasso*.
- Dintr-un punct de vedere Bayesian, regresia de creastă poate fi interpretată ca folosind distribuţii normale pentru coeficienţii β (a unui model liniar), cu o deviaţie standard mică ce-i împinge pe aceştia spre 0.
- Acesta este modelul pe care l-am folosit în regresia liniară.
- Pe de altă parte, regresia Lasso poate fi interpretată ca *probabilitatea maximă a posteriori* (*MAP*) calculată dintr-un model care foloseşte distribuţii a priori de tip *Laplace* pentru coeficienţii β.

- Distribuţia Laplace arată similar cu cea Gaussiană, dar derivata densităţii ei în 0 este nedefinită (arată ca un vârf ascuţit):

```
plt.figure(figsize=(8, 6))
x_values = np.linspace(-10, 10, 1000)
for df in [1, 2, 5, 15]:
    distri = stats.laplace(scale=df)
    x_pdf = distri.pdf(x_values)
    plt.plot(x_values, x_pdf,
                label=f'b = {df}')
x_pdf = stats.norm.pdf(x_values)
plt.plot(x_values, x_pdf, label='Gaussian')
plt.xlabel('x')
plt.yticks([])
plt.legend()
plt.xlim(-7, 7)
```



- Distribuţia Laplace concentrează mai multă masă de probabilitate în jurul lui 0 decât distribuţia normală, contribuind astfel la regularizare.