Regresie liniară
Cursul 6

Programare şi modelare probabilistă - anul III

Facultatea de Informatică, UAIC

e-mail: adrian.zalinescu@uaic.ro

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

13 Noiembrie 2023

Universul statisticii şi învăţării automate se bazează pe modele recurente, adică tipare ce apar în mod frecvent.
În acest curs, ne vom concentra pe unul dintre cele mai populare şi utile dintre ele, *modelul liniar*.

Plan

1. Regresie liniară simplă
2. Regresie polinomială
3. Regresie liniară multiplă

Regresia liniară simplă

Multe probleme pe care le regăsim în ştiinţă, inginerie şi afaceri sunt de următoarea formă:

- avem o variabilă x şi vrem să modelăm/să prezicem o variabilă y.
- Aceste variabile sunt observate ca

$$\{(x_1, y_1),\ (x_2, y_2),\ \ldots,\ (x_n, y_n)\}\,.$$

- În cel mai simplu scenariu, cunoscut ca *regresie liniară simplă,* atât x, cât şi y sunt variabile aleatoare continue unidimensionale.
- Variabila y se numeşte variabila *dependentă, prezisă* sau *rezultat,* iar variabila x se numeşte variabila *independentă*, *prezicătoare* sau *de intrare.*
- Dacă x este multidimensională, atunci avem de a face cu *regresie liniară multiplă.*

Câteva situaţii tipice în care pot fi utilizate modele de regresie liniară sunt:

- Modelaţi relaţia dintre factori precum ploaia, salinitatea solului şi prezenţa/absenţa îngrăşământului în productivitatea culturilor. Apoi, răspundeţi la astfel de întrebări:
  - Este relaţia liniară?
  - Cât de puternică este această relaţie?
  - Ce factori au cel mai puternic efect?
- Găsiţi o relaţie între consumul mediu de ciocolată pe ţară şi numărul de laureaţi Nobel din ţara respectivă şi apoi înţelegeţi de ce acest lucru nu ar fi relevant.
- Preziceţi consumul de gaz (folosit pentru încălzire şi gătit) al casei dvs folosind cantitatea de radiaţii solare din raportul meteo local. Cât de precisă este această predicţie?

Legătura cu învăţarea automată

Parafrazându-l pe Kevin P. Murphy[1], *învăţarea automată* (*ML*) este un termen larg folosit pentru o colecţie de metode de a învăţa automat tipare în date şi apoi de a folosi ceea ce am învăţat pentru a prezice date ulterioare sau pentru a lua decizii în condiţii de incertitudine.

Astfel, ML şi statistica sunt subiecte cu adevărat interconectate, iar legătura devine clară prin adoptarea unei perspective probabilistice.

[1] Kevin P. Murphy, *Machine learning: A probabilistic perspective*, 2012

Câteva traduceri din terminologia ML:

- Un specialist în ML vorbeşte de obicei despre *caracteristici* sau *atribute* în loc de variabile.
- O problemă de regresie este un exemplu de *învăţare supravegheată*. În cadrul învăţării automate, avem o problemă de regresie atunci când vrem să învăţăm o funcţie ce asociază lui x pe y.
- Spunem că procesul de învăţare este supravegheat deoarece cunoaştem valorile perechilor;
- într-un anumit sens, ştim răspunsul corect şi toate întrebările rămase se referă la modul de generalizare al acestor observaţii la orice observaţie viitoare posibilă, adică la o situaţie când îl ştim pe x, dar nu pe y.

Un model de regresie liniară simplă este specificat de următoarea ecuaţie:

$$y_i = \alpha + x_i\beta.$$

Aceasta spune că există o relaţie *liniară* între variabila x şi variabila y.

- parametrul β controlează *panta* relaţiei liniare şi poate fi intrepretată ca schimbarea ce se produce în variabila y la schimbarea cu o unitate a variabilei x.
- parametrul α se numeşte *intercepţia* (punctul unde dreapta intersectează/interceptează axa Oy.

Există mai multe metode de a determina parametrii pentru un model de regresie simplă.

- Una se numeşte *metoda celor mai mici pătrate*: problemă de optimizare/minimizare a erorii medii pătratice între y-ul observat şi cel prezis:

$$\min_{\alpha,\beta} \sum_{i=1}^{n} |y_i - (\alpha x_i + \beta)|^2 .$$

- O alternativă este să generăm un model probabilist, ce are avantajul că pe lângă găsirea valorilor optimale pentru α şi β, oferă şi o estimare a incertitudinii pe care o avem asupra acestor parametri.
- Astfel, un model de *regresie liniară probabilistă* (sau *Bayesiană*) este exprimat prin

$$y \sim \mathcal{N}(\mu = \alpha + x\beta, \varepsilon).$$

Deoarece nu ştim valorile parametrilor α, β şi ε, vom seta distribuţii a priori pentru aceştia:

$$\begin{aligned} \alpha &\sim \mathcal{N}(\mu_\alpha, \sigma_\alpha) \\ \beta &\sim \mathcal{N}(\mu_\beta, \sigma_\beta) \\ \varepsilon &\sim |\mathcal{N}(0, \sigma_\varepsilon)| \end{aligned}$$

- Pentru α, putem folosi o distribuţie Gaussiană "aplatizată", cu valori mari ale lui σ_α relativ la scara de valori a datelor.
- Aceste valori variază de la o problemă la alta, dar de multe ori μ_α se ia egal cu 0 şi σ_α nu mai mare de 10.
- Pentru β, e în general mai simplu să avem o idee despre cum ar arăta panta de regresie; măcar putem şti semnul acestui parametru.
- Pentru ε, putem seta σ_ε la o valoare mare relativ la scara de valori ale lui y (de exemplu, de 10 ori valoarea deviaţiei standard pentru y).

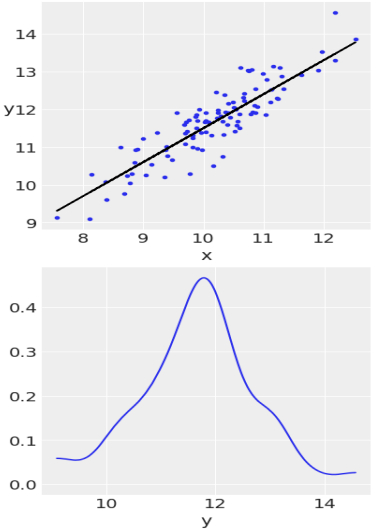
Un prim rezultat este că probabilitatea maximă a posteriori (MAP) dintr-o problemă de regresie liniară simplă Bayesiană cu probabilităţi a priori "plate" coincide cu estimările punctuale obţinute cu metoda celor mai mici pătrate.

Alternative pentru distribuţia jumătate-Gaussiană pentru ε:

- distribuţia jumătate-Cauchy;
- distribuţia uniformă nu este în general o alegere bună, decât dacă parametrul ε este restrâns de "limite robuste";
- dacă dorim distribuţii a priori pentru valori specifice pentru ε, putem folosi distribuţia Gamma.

Exemplu de date modelate de regresia liniară

```
np.random.seed(1)
N = 100
alpha_real = 2.5
beta_real = 0.9
eps_real = np.random.normal(0, 0.5, size=N)

x = np.random.normal(10, 1, N)
y_real = alpha_real + beta_real * x
y = y_real + eps_real

_, ax = plt.subplots(1, 2, figsize=(8, 4))
ax[0].plot(x, y, 'C0.')
ax[0].set_xlabel('x')
ax[0].set_ylabel('y', rotation=0)
ax[0].plot(x, y_real, 'k')
az.plot_kde(y, ax=ax[1])
ax[1].set_xlabel('y')
plt.tight_layout()
```



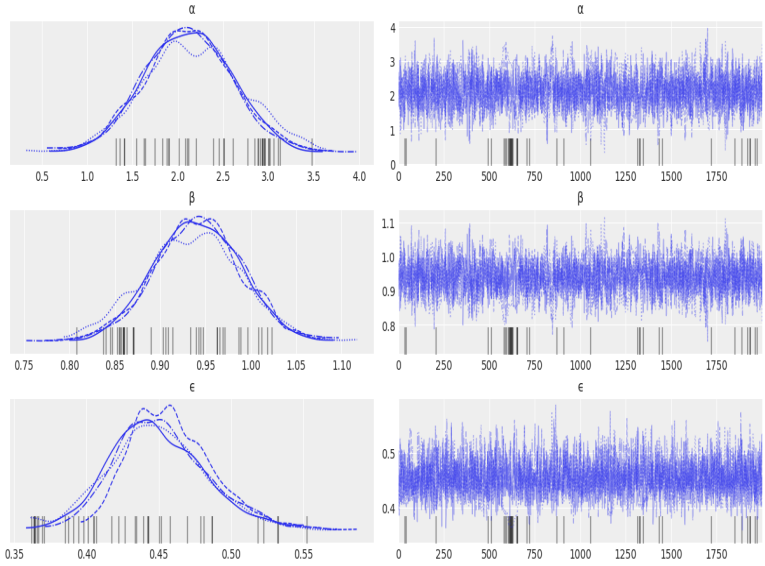
Folosim PyMC pentru a construi şi a calibra (fit) modelul:

```
with pm.Model() as model_g:
    α = pm.Normal('α', mu=0, sigma=10)
    β = pm.Normal('β', mu=0, sigma=1)
    ϵ = pm.HalfCauchy('ϵ', 5)
    μ = pm.Deterministic('μ', α + β * x)
    y_pred = pm.Normal('y_pred', mu=μ, sigma=ϵ, observed=y)

    idata_g = pm.sample(2000, tune=2000, return_inferencedata=True)
```

Observaţie: nu este nevoie să folosim `pm.Deterministic` decât dacă vrem ca variabila μ să fie calculată şi salvată în eşantion.

Rezultatul inferenţei:

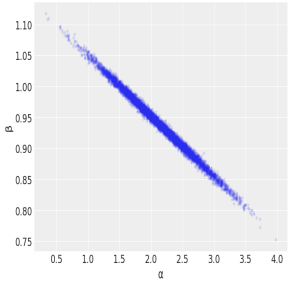
```
az.plot_trace(idata_g, var_names=['α', 'β', 'ε'])
```



Modele liniare şi autocorelaţie ridicată

Modelele liniare conduc la o distribuţie a posteriori unde α şi β sunt puternic (şi liniar) corelate:

```
az.plot_pair(idata_g, var_names=['α', 'β'], scatter_kwargs={'alpha':  0.1})
```



Acest lucru se întâmplă deoarece dreptele de regresie ar trebui să treacă prin punctul de coordonate date de media variabilei x şi media variabilei y (cu o eroare dată de factorul de incertitudine dat de eşantionarea distribuţiei a posteriori).

Modificarea datelor înainte de rulare

O modalitate simplă de a şterge corelaţia între α şi β este de a centra variabila x:

- fie $\bar{x}$ media variabilei x: $\bar{x} := \frac{x_1+\cdots+x_n}{n}$;
- fie $x' := x - \bar{x}$.

Astfel:

- dacă înainte intercepţia reprezenta valoarea lui y când $x = 0$ (ce nu are neapărat o semnificaţie reală),
- acum ea reprezintă valoarea lui y când $x = \bar{x}$.

Dacă vrem să ne întoarcem la parametrii originali, atunci efectuăm transformările

$$\begin{aligned} \alpha &= \alpha' - \beta'\bar{x}; \\ \beta &= \beta'. \end{aligned}$$

Standardizarea

Pe lângă centrare, putem transforma datele prin *standardizarea* lor, adică cerând ca ele să fie centrate, cu deviaţia standard egală cu 1:

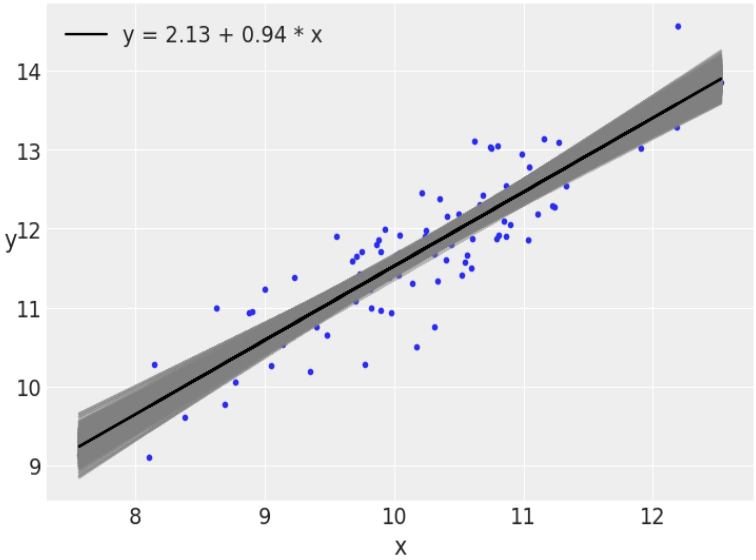
$$\begin{aligned} x' &= \frac{x - \bar{x}}{x_{\text{sd}}}; \\ y' &= \frac{y - \bar{y}}{y_{\text{sd}}}. \end{aligned}$$

- Avantajul acestei proceduri este că vom putea folosi întotdeauna aceleaşi distribuţii a priori slab informative, fără a trebui să ne gândim la scara de valori a datelor.
- Astfel, pentru datele standardizate, intercepţia va fi mereu în jurul lui 0, iar panta va fi restrânsă la intervalul $[-1, 1]$.
- De asemenea, va fi mai uşor de lucrat cu mai multe variabile – a avea toate variabilele la aceeaşi scară simplifică interpretarea datelor.

Interpretarea şi vizualizarea distribuţiei a posteriori

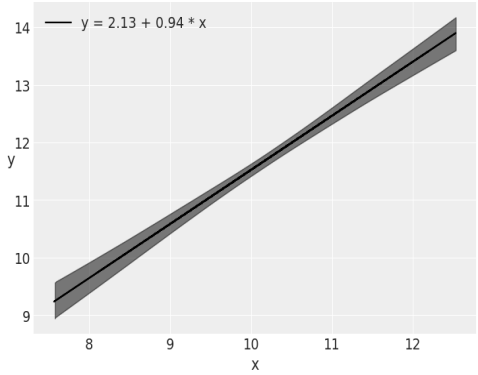
- Putem utiliza funcţii `ArviZ` precum `plot_trace` şi `summary` pentru explorarea distribuţiei a posteriori, dar putem folosi şi propriile noastre funcţii.
- Pentru regresia liniară, ar putea fi util să vizualizăm toate dreptele ce provin din eşantionarea distribuţiei a posteriori, şi mai ales media acestor drepte:

```
plt.plot(x, y, 'C0.')
posterior_g = idata_g.posterior.stack(samples={"chain", "draw"})
alpha_m = posterior_g['α'].mean().item()
beta_m = posterior_g['β'].mean().item()
draws = range(0, posterior_g.samples.size, 10)
plt.plot(x, posterior_g['α'][draws].values
                + posterior_g['β'][draws].values * x[:,None],
                c='gray', alpha=0.5)
plt.plot(x, alpha_m + beta_m * x, c='k',
         label=f'y = {alpha_m:.2f} + {beta_m:.2f} * x')
plt.xlabel('x')
plt.ylabel('y', rotation=0)
plt.legend()
```


y = 2.13 + 0.94 * x
14
13
12
11
10
9
y
8
9
10
11
12
x

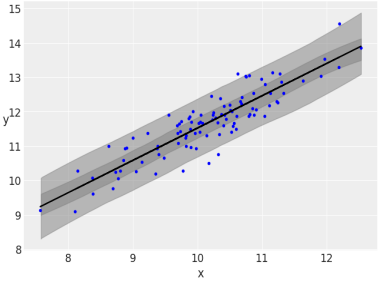
O altă modalitate de a vizualiza distribuţia a posteriori este să folosim funcţia `plot_hdi` din `ArviZ`:

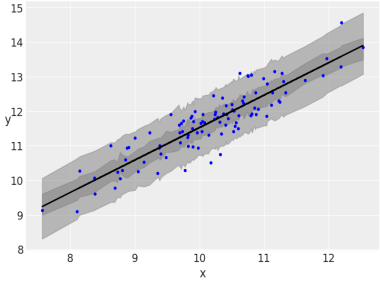
```
plt.plot(x, alpha_m + beta_m * x, c='k',
         label=f'y = {alpha_m:.2f} + {beta_m:.2f} * x')

sig = az.plot_hdi(x, posterior_g['μ'].T, hdi_prob=0.98, color='k')

plt.xlabel('x')
plt.ylabel('y', rotation=0)
plt.legend()
```



De asemenea, putem prezice date conform distribuţiei predictive a posteriori folosind funcţia `sample_posterior_predictive()`, iar apoi să le vizualizăm împreună cu intervalele HPD corespunzătoare:

```
ppc = pm.sample_posterior_predictive(idata_g, samples=2000, model=model_g)
plt.plot(x, y, 'b.')
plt.plot(x, alpha_m + beta_m * x, c='k',
         label=f'y = {alpha_m:.2f} + {beta_m:.2f} * x')
az.plot_hdi(x, ppc.posterior_predictive['y_pred'], hdi_prob=0.5, color='gray', #smooth=False)
az.plot_hdi(x, ppc.posterior_predictive['y_pred'], color='gray', #smooth=False)
plt.xlabel('x')
plt.ylabel('y', rotation=0)
```





Coeficientul de corelaţie al lui Pearson

Câteodată dorim să măsurăm gradul de dependenţă (liniară) între două variabile.
Cea mai întâlnită măsură a acestuia este *coeficientul de corelaţie al lui Pearson*, de multe ori notat cu r:

- $r = \pm 1$ reprezintă o corelaţie perfectă: o creştere cu o unitate a lui x prezice o creştere/descreştere cu β unităţi a lui y;
- $r = 0$ reprezintă absenţa dependenţei liniare.

Wikipedia oferă câteva exemple concludente.

Coeficientul de corelaţie se defineşte ca

$$r := \beta \cdot \frac{x_{\text{sd}}}{y_{\text{sd}}}.$$

- r este în intervalul $[-1, 1]$;
- dacă standardizăm datele, r devine egal cu β.

Coeficientul de determinare, r^2, este cuprins între 0 şi 1, valoarea 1 însemnând o corelaţie perfectă.

- În regresia liniară Bayesiană, acesta nu se calculează ca pătratul coeficientului de corelaţie, ci ia în considerare distribuţia predictivă a posteriori.
- Se apelează în `ArviZ cu r2_score`:

```
az.r2_score(y, az.extract(ppc, group="posterior_predictive", var_names="y_pred").values.T)
```

```
r2        0.787794
r2_std    0.006101
dtype: float64
```

Legătura între corelaţie şi cauzalitate

Să presupunem că vrem să prezicem cât vom plăti pentru gaz pentru a ne încălzi casa în timpul iernii şi să presupunem că ştim cantitatea de radiaţie solară din zona în care trăim.

- În acest exemplu, radiaţia solară va fi variabila independentă x, iar factura este variabilă dependentă, y.
- La fel de bine am putea inversa întrebarea şi întreba despre cantitatea de radiaţie solară, având în vedere factura.
- Numim o variabilă independentă deoarece valoarea ei nu poate fi prezisă de model; în schimb, este *o intrare* a modelului, iar variabila dependentă este *rezultatul*.
- Dar nu stabilim o relaţie cauzală între variabile: *corelaţia nu implică cauzalitate!*
- Nu putem controla cantitatea de radiaţii emise de soare prin schimbarea termostatului casei noastre!

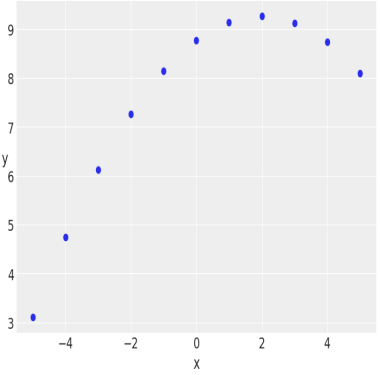
Regresia polinomială

O posibilitate de a parametriza datele este prin construcţia unui polinom:

$$\mu = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_m x^m.$$

Ca set de date, folosim al doilea grup din cvartetul lui Anscombe:
`https://en.wikipedia.org/wiki/Anscombe%27s_quartet`

```
ans = pd.read_csv('./data/anscombe.csv')
x_2 = ans[ans.group == 'II']['x'].values
y_2 = ans[ans.group == 'II']['y'].values
x_2 = x_2 - x_2.mean()

plt.scatter(x_2, y_2)
plt.xlabel('x')
plt.ylabel('y', rotation=0)
```

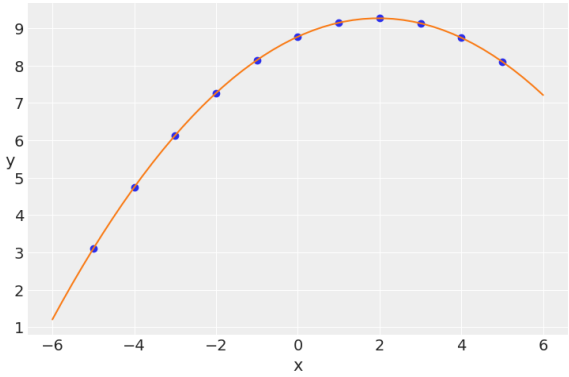


Modelul:

```
with pm.Model() as model_poly:
    α = pm.Normal('α', mu=y_2.mean(), sd=1)
    β1 = pm.Normal('β1', mu=0, sd=1)
    β2 = pm.Normal('β2', mu=0, sd=1)
    ϵ = pm.HalfCauchy('ϵ', 5)

    mu = α + β1 * x_2 + β2 * x_2**2

    y_pred = pm.Normal('y_pred', mu=mu, sd=ϵ, observed=y_2)

    idata_poly = pm.sample(2000, return_inferencedata=True)
```

Analiza rezultatelor:

```
x_p = np.linspace(-6, 6)
y_p = idata_poly.posterior['α'].mean().item()
        + idata_poly.posterior['β1'].mean().item() * x_p
        + idata_poly.posterior['β2'].mean().item() * x_p**2
plt.scatter(x_2, y_2)
plt.xlabel('x')
plt.ylabel('y', rotation=0)
plt.plot(x_p, y_p, c='C1')
```



Interpretarea parametrilor în regresia polinomială

- Valorile lui β_1 şi β_2 nu reprezintă pante, aşa cum era valoarea lui β în regresia liniară.
- Problema interpretării rezultatelor nu este doar o chestiune matematică.
- În cele mai multe cazuri, parametrii nu sunt traduşi în cantităţi ce au sens în domeniul nostru de cunoaştere, adică nu au o semnificaţie fizică.
- În general, se consideră că polinoamele de grad mai mare decât 2 sau 3 nu sunt modele prea folositoare, şi se preferă alternative, ca *procesele Gaussiene* (*mişcarea Browniană*).
- De fapt, mărind suficient de mult gradul polinomului, putem face ca polinomul găsit să se potrivească exact pe datele noastre.
- Un model care se potriveşte prea bine cu datele observate, nu va răspunde bine cu datele neobservate. Acesta este fenomenul de *supraspecializare* (*overfitting*).

Regresie liniară multiplă

Câteodată, vrem să includem în model dependenţa de mai multe variabile, considerând astfel mai multe variabile independente.
Exemple:

- Calitatea (percepută) a vinului, dependentă de aciditate, densitate, nivelul de alcool, zahăr rezidual şi conţinutul de sulfaţi.
- Nota medie a unui student în funcţie de venitul familial, distanţa între casă/şcoală şi nivelul de educaţie al mamei.

Acest model se numeşte *regresie liniară multiplă* (a nu se confunda cu *regresia liniară multivariată*, atunci când avem mai multe variabile dependente).

În regresia liniară multiplă, modelăm media variabilei dependente ca

$$\mu = \alpha + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_m x_m$$

sau, în notaţie prescurtată

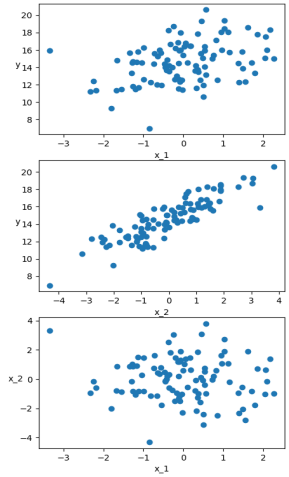
$$\mu = \alpha + X\beta,$$

unde $X = (x_1\ x_2\ \ldots\ x_m)$, iar $\beta = (\beta_1\ \beta_2\ \ldots\ \beta_m)^{\mathrm{T}}$.
Să definim datele:

```
np.random.seed(314)
N = 100
alpha_real = 2.5
beta_real = [0.9, 1.5]
eps_real = np.random.normal(0, 0.5, size=N)

X = np.array([np.random.normal(i, j, N) for i, j in zip([10, 2], [1, 1.5])]).T
X_mean = X.mean(axis=0, keepdims=True)
X_centered = X - X_mean
y = alpha_real + np.dot(X, beta_real) + eps_real
```

Vizualizarea datelor:
trei mulţimi de puncte (`scatter`), două cu dependenţele între variabila independentă şi cele dependente, a treia cu dependenţa între variabilele dependente:

```
def scatter_plot(x, y):
    plt.figure(figsize=(5, 10))
    for idx, x_i in enumerate(x.T):
        plt.subplot(3, 1, idx+1)
        plt.scatter(x_i, y)
        plt.xlabel(f'x_{idx+1}')
        plt.ylabel(f'y', rotation=0)

    plt.subplot(3, 1, idx+2)
    plt.scatter(x[:, 0], x[:, 1])
    plt.xlabel(f'x_{idx}')
    plt.ylabel(f'x_{idx+1}', rotation=0)

scatter_plot(X_centered, y)
```



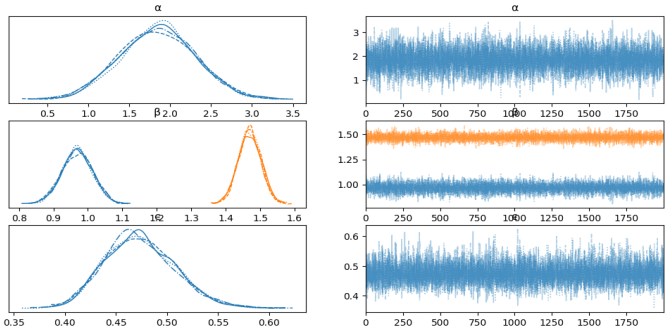
Modelul este similar cu cel din cazul regresiei simple. Diferenţe:

- variabila β este Gaussiană cu shape=2 (o pantă pentru fiecare variabilă);
- variabila μ se defineşte cu ajutorul *produsului scalar* din PyMC, `pm.math.dot()`, deoarece β este un tensor `Pytensor` (şi nu un `array NumPy`).

```
with pm.Model() as model_mlr:
    α_tmp = pm.Normal('α_tmp', mu=0, sigma=10)
    β = pm.Normal('β', mu=0, sigma=1, shape=2)
    ϵ = pm.HalfCauchy('ϵ', 5)

    μ = α_tmp + pm.math.dot(X_centered, β)

    α = pm.Deterministic('α', α_tmp - pm.math.dot(X_mean, β))

    y_pred = pm.Normal('y_pred', mu=μ, sigma=ϵ, observed=y)

    idata_mlr = pm.sample(2000, return_inferencedata=True)
```

Rezumatul distribuţiei a posteriori:

```
az.plot_trace(idata_mlr, var_names=['α', 'β', 'ε'])
```




```
az.summary(idata_mlr, var_names=['α', 'β', 'ε'])
```

	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
α[0]	1.848	0.461	0.936	2.657	0.004	0.003	12074.0	6007.0	1.0
β[0]	0.969	0.044	0.886	1.053	0.000	0.000	12140.0	6440.0	1.0
β[1]	1.469	0.033	1.407	1.530	0.000	0.000	12459.0	6029.0	1.0
ε	0.474	0.035	0.409	0.538	0.000	0.000	11165.0	6351.0	1.0