

8. Előadás

Generatív modellek

Naive Bayes

Gauss-i keverékek

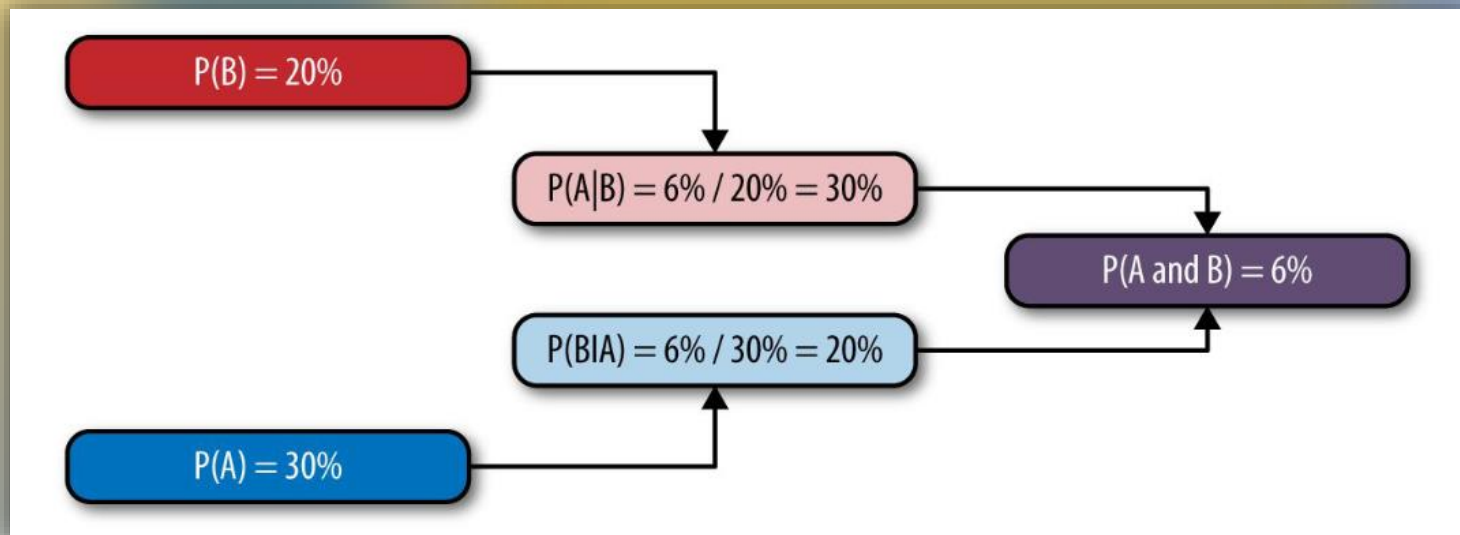
Naive Bayes: Feltételes valószínűségek

🐍 Valamely A esemény feltételes valószínűsége azt jelenti, hogy mekkora az esély A bekövetkezésére feltéve, hogy B esemény már bekövetkezett vagy bekövetkezik. A következőképpen definiálható:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

🐍 Például: mekkora a valószínűsége, hogy a megrendelés csalóktól érkezik, feltéve, ha ajándékutalvánnyal fizettek?

$$P(\text{Fraud} | \text{Giftcard}) = \frac{P(\text{Fraud} \cap \text{Giftcard})}{P(\text{Giftcard})}$$



Inverz feltételes valószínűség: a Bayes-tétel

🐍 Ugyanez a feltételes valószínűség kiszámítható a másik feltételes valószínűség és a nem feltételes valószínűségek segítségével:

$$P(B \mid A) = \frac{P(A \mid B)P(B)}{P(A)}$$

$$P(Fraud \mid Giftcard) = \frac{P(Giftcard \mid Fraud)P(Fraud)}{P(Giftcard)}$$


🐍 A Naive Bayes osztályozók ezt a tételt veszik alapul. A bayes-i osztályozásban ez úgy reprezentálódik, hogy valamely L címke valószínűségét akarjuk meghatározni abban az esetben, ha adott változók halmaza: *features*.


$$P(L \mid \text{features}) = \frac{P(\text{features} \mid L)P(L)}{P(\text{features})}$$

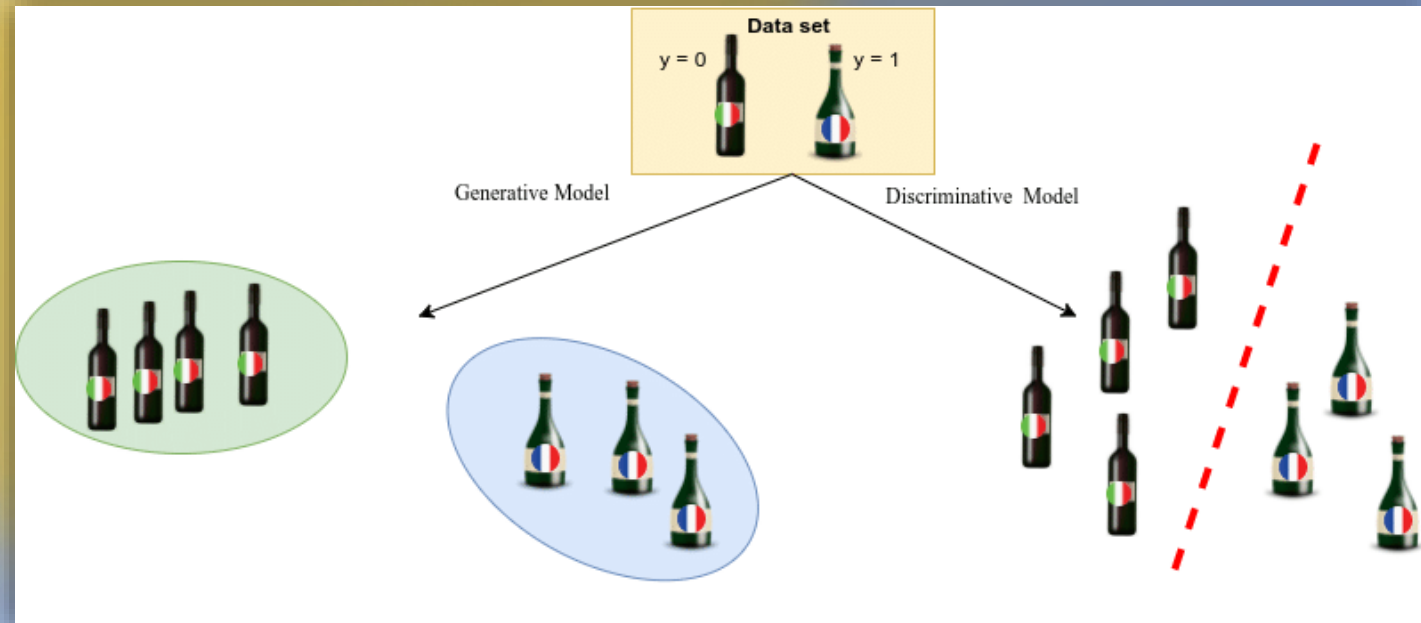
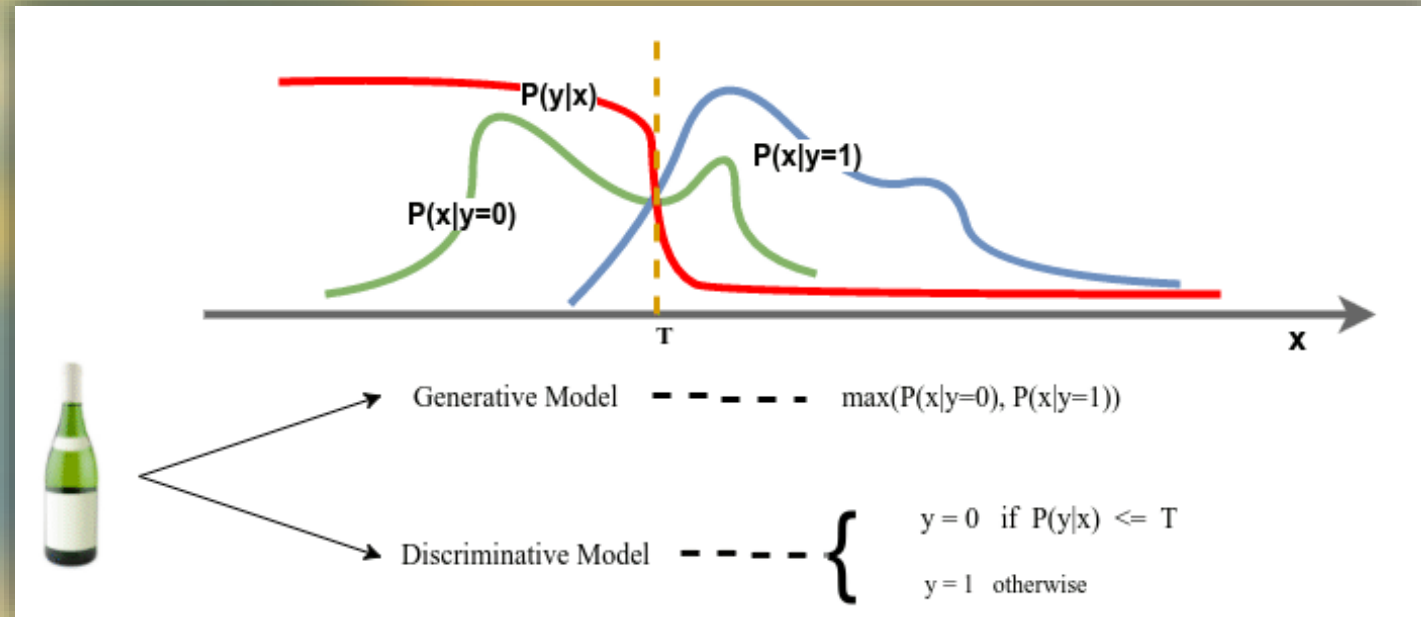
🐍 Ha több címke (L_1, L_2) közül szeretnénk eldönteni, hogy melyik az inkább jellemző egy adott címkehalmazra, a következőképpen kell számolnunk:

$$\frac{P(L_1 \mid \text{features})}{P(L_1 \mid \text{features})} = \frac{P(\text{features} \mid L_1) P(L_1)}{P(\text{features} \mid L_2) P(L_2)}$$


Generatív modellek

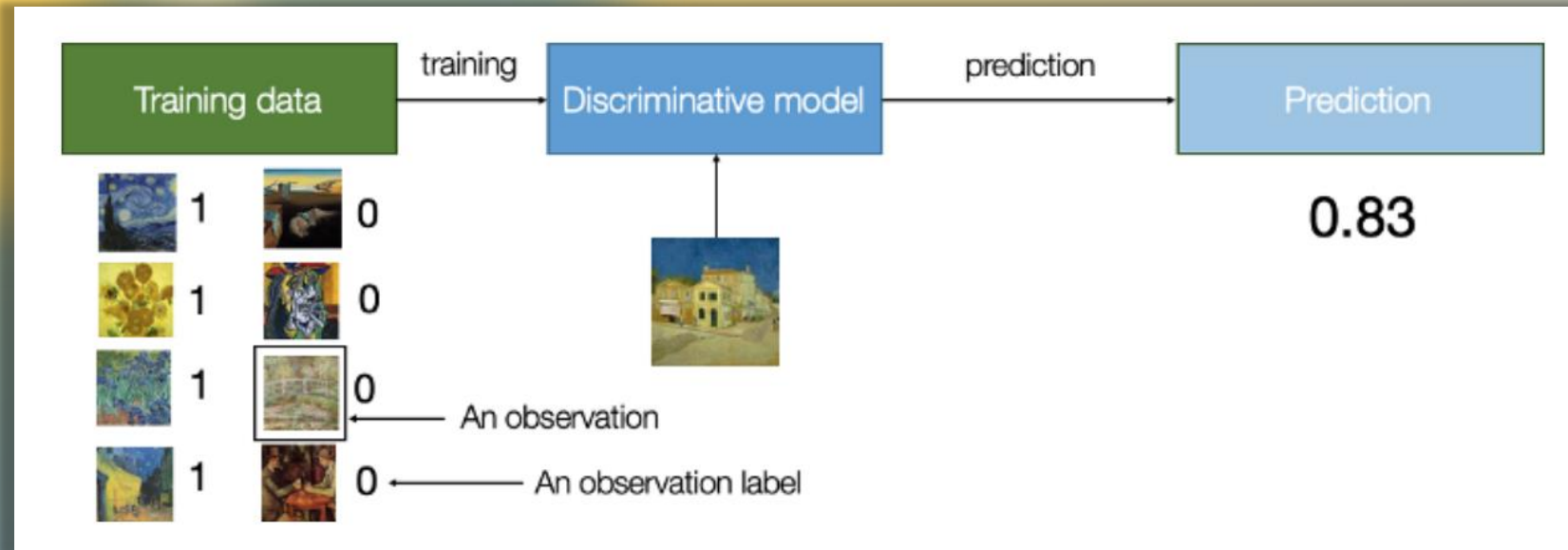
 A Naive Bayes egy **generatív modellezési eljárás**, mert azt a hipotetikus véletlenszerű eljárást modellezi, ami az adatokat generálhatta. Ezeknek az eloszlásoknak a meghatározása a NB feladata.


 Az általános eljárás erre a problémára nagyon bonyolult, de gyorsítható a folyamat a feltevéseink egyszerűsítésével. A NB azzal egyszerűsíti a problémát, hogy kevés előfeltétellel él az adatok irányába.

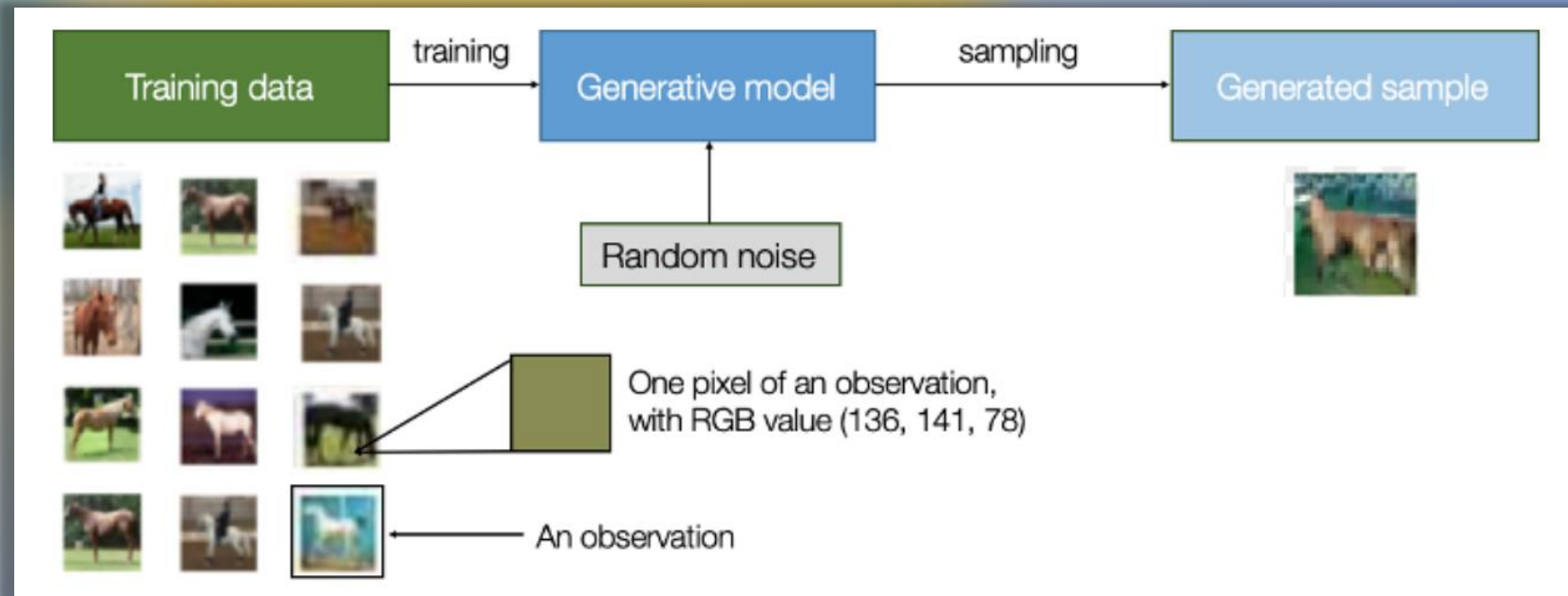


Diszkriminatív vs. Generatív modellezés

 Diszkriminatív módszer esetén minta alapján tanítunk, majd a létrejövő modell segítségével egy új mintaegyedről meg tudjuk állapítani a hozzá tartozó predikciót.



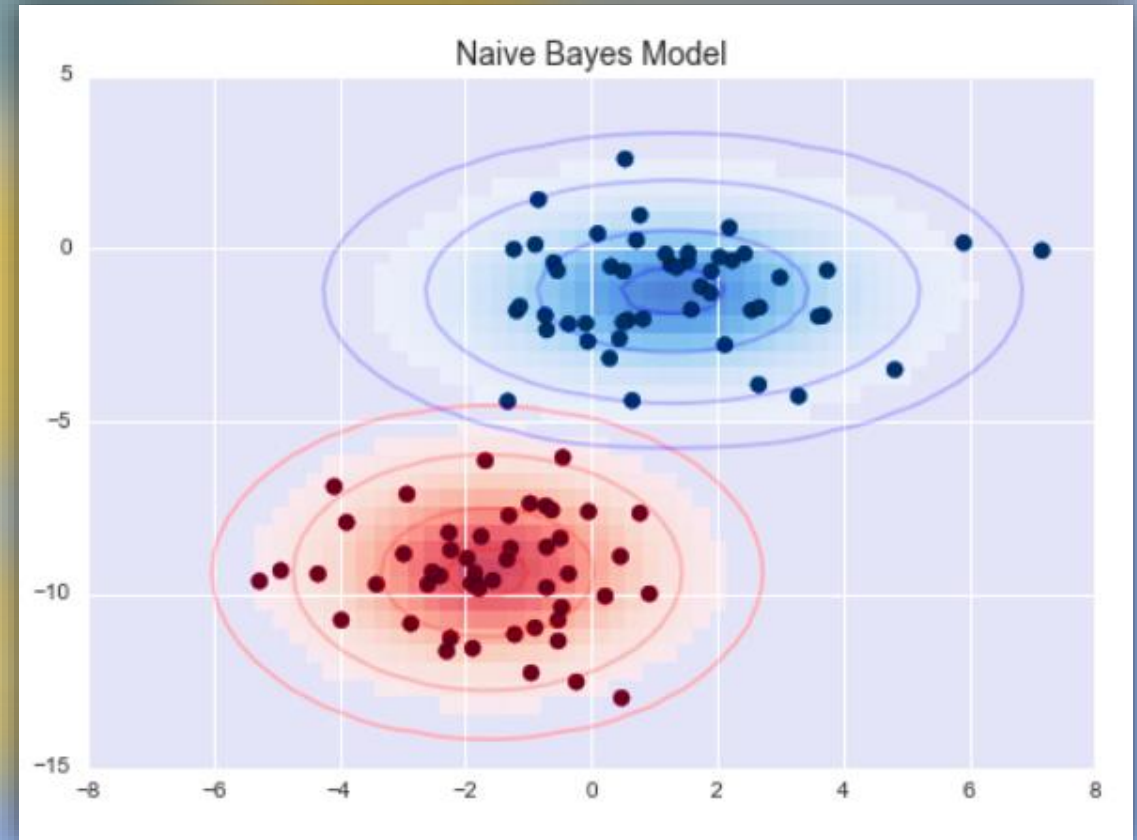
 Generatív módszer esetén minta alapján tanítunk, majd zajt engedünk a rendszerbe. A modell outputja egy új mintaegyed lesz.



Gauss-i Naive Bayes osztályozó

- 🐍 Egy nagyon egyszerű és gyors módja a modellezésnek az, ha azt feltételezzük, hogy az adatokat Gauss-i eloszlások generálták.
- 🐍 Ezt a modellt úgy lehet tanítani, hogy megtaláljuk a Gauss-i függvények átlagát és szórását minden címkeosztályra. Ez a Naive Bayes-i feltételezés a *make_blobs()* által létrehozott adathalmazra:

```
from sklearn.datasets import make_blobs
X, y = make_blobs(100, 2, centers=2, random_state=2, cluster_std=1.5)
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='RdBu');
```

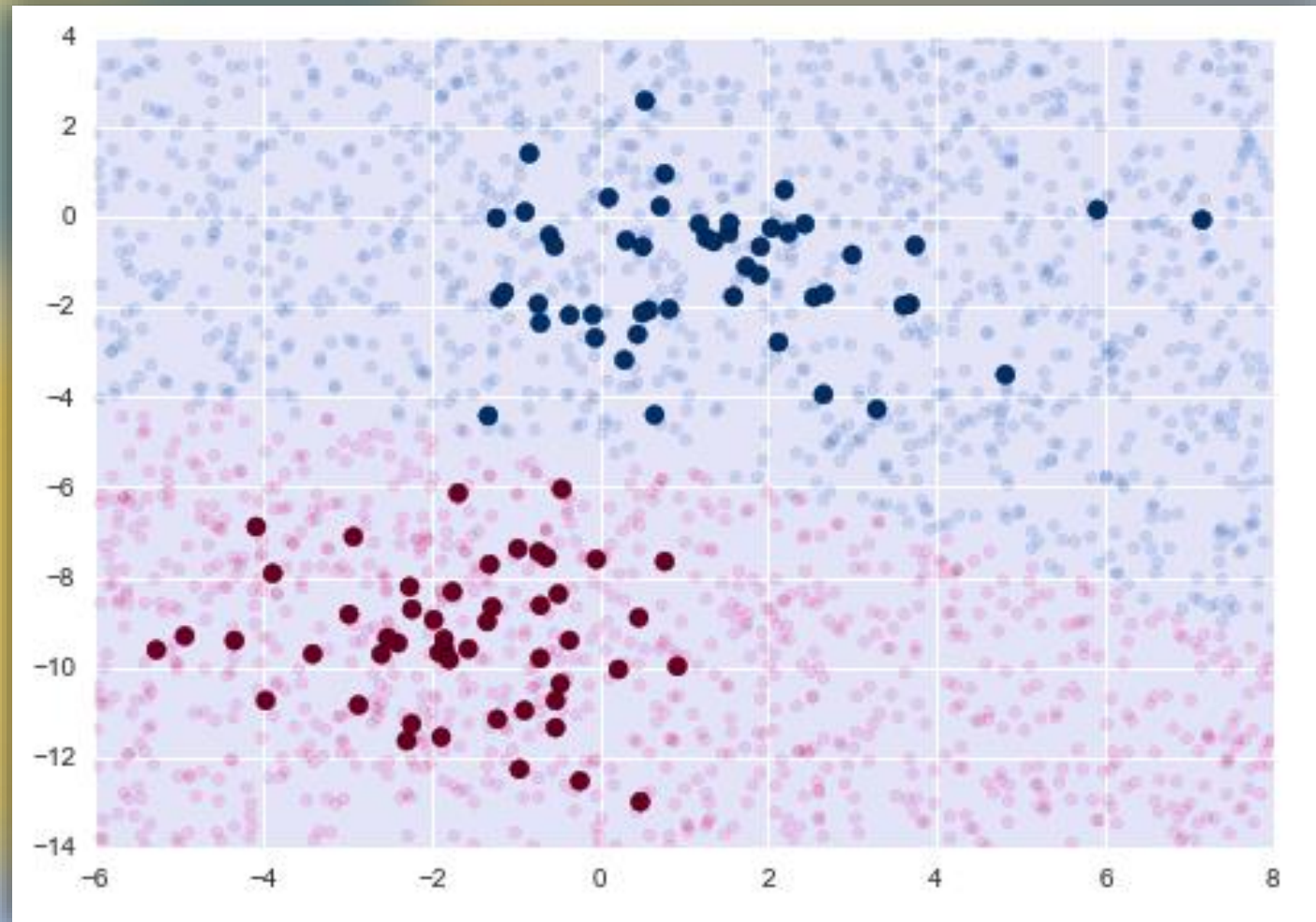


Generatív döntési határok

🐍 A generatív modellek esetén a térben az osztályok területeit az osztályokhoz tartozó valószínűségek határozzák meg.

🐍 Ahol egy adott osztályba esés valószínűsége magasabb, ott az adott osztályba lesznek sorolva a mintaegyedek. Ahol ezek a valószínűségek metszik egymást, jönnek létre a generatív döntési határok.

🐍 Ezek általában kvadratikusak.

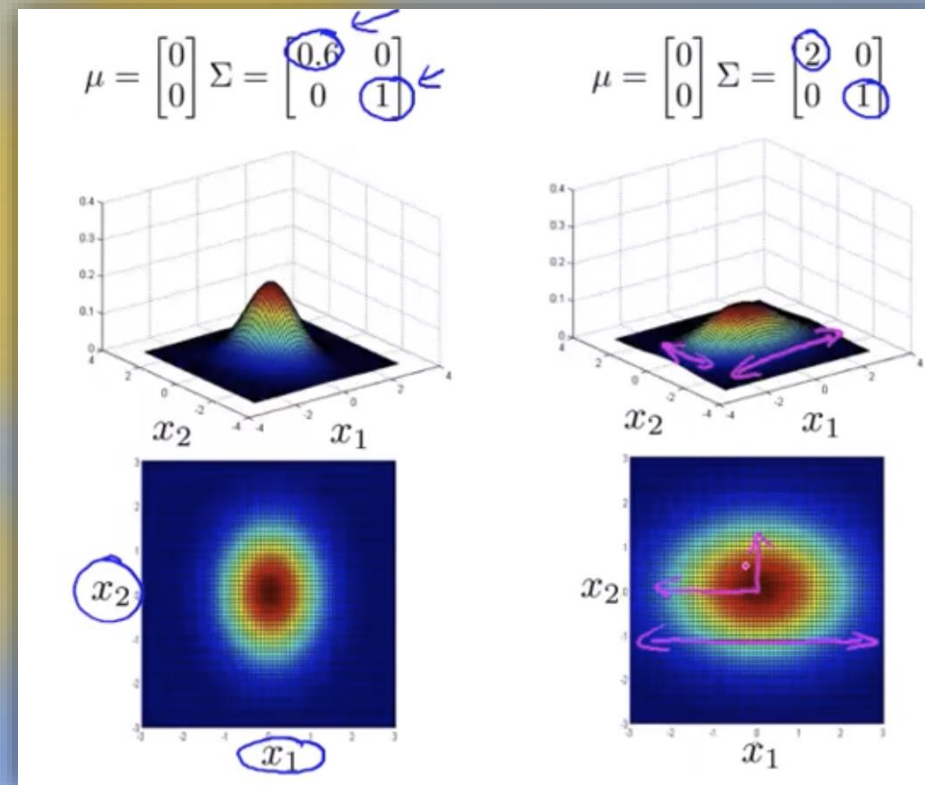
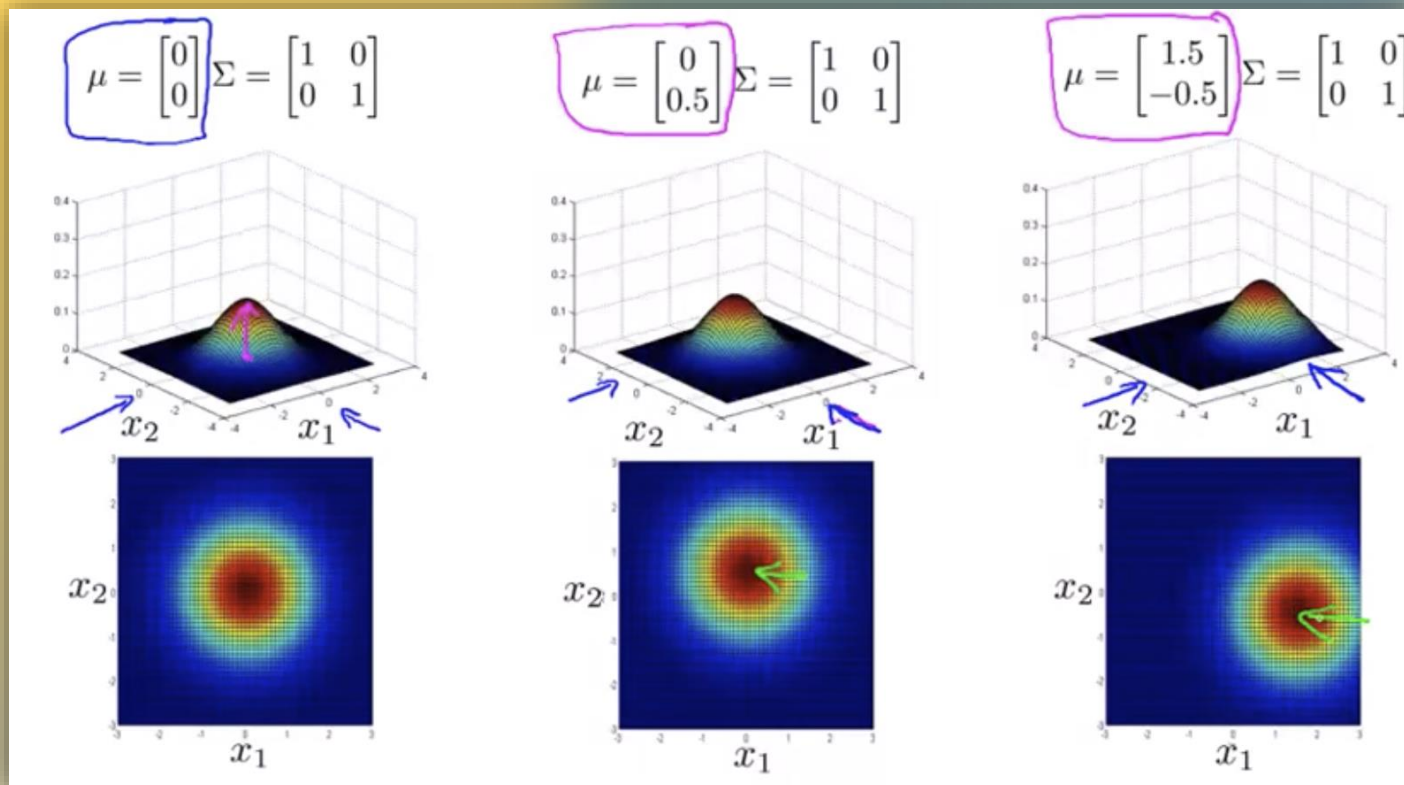


Gauss-i keverékek (GMM)



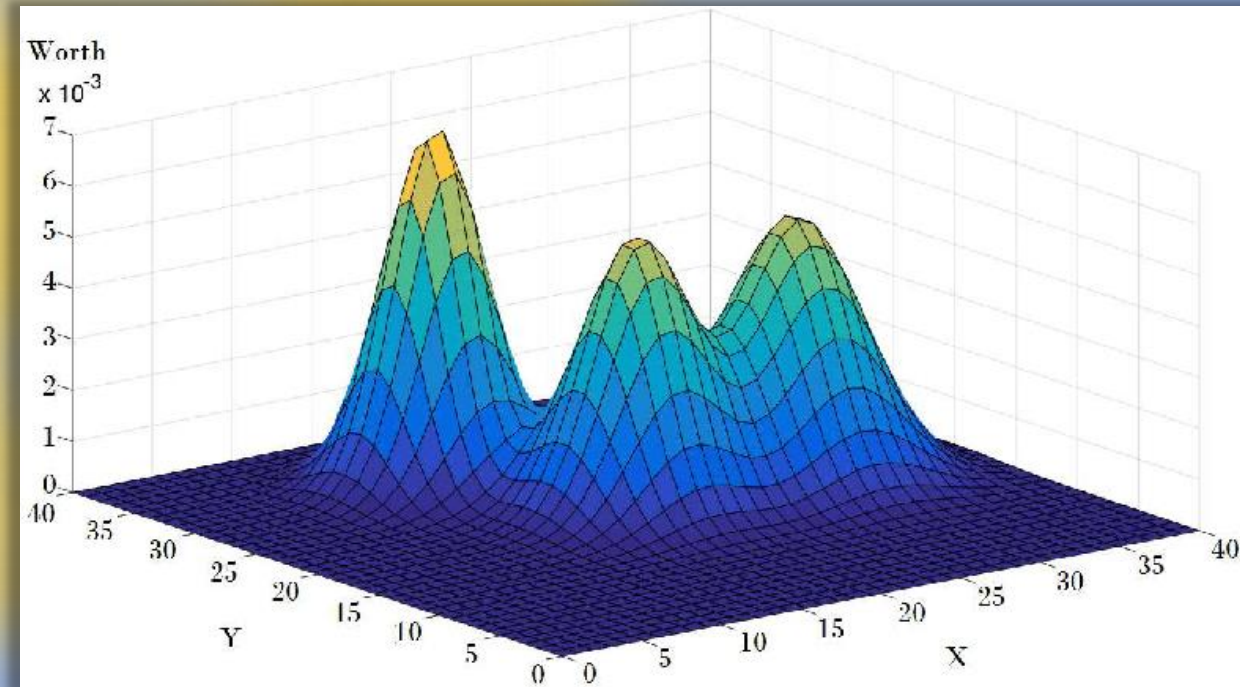
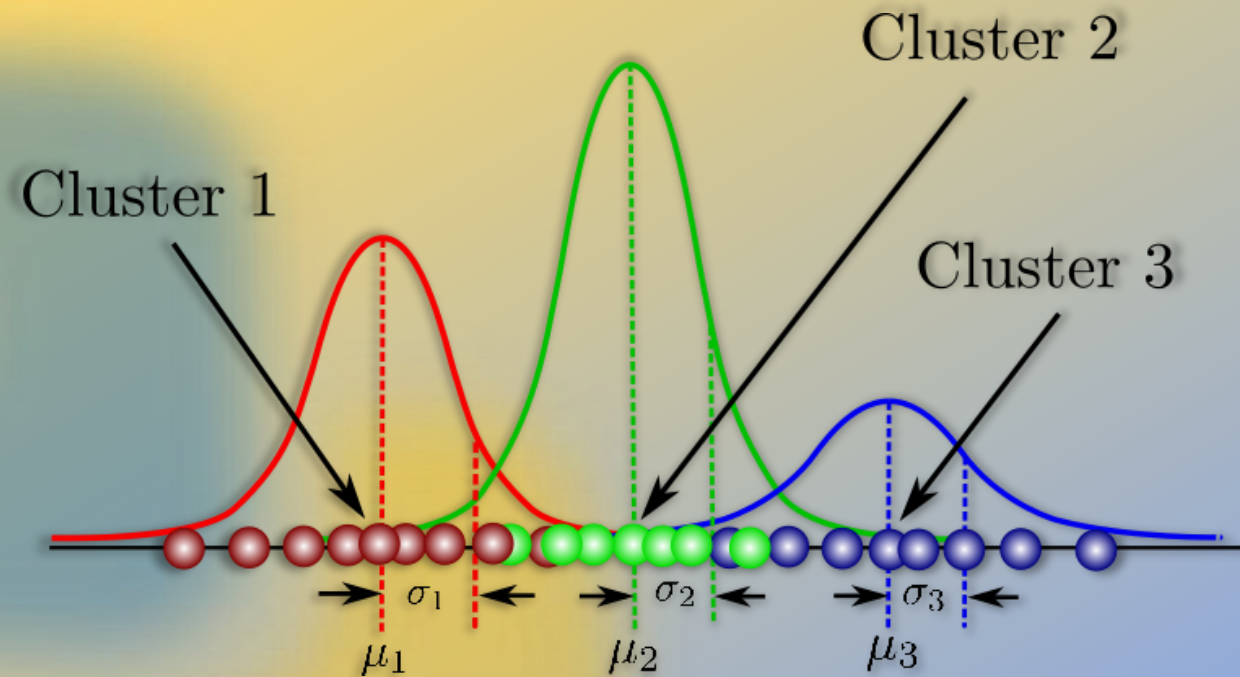
Egy probabilisztikus modell fajta, amelynek alap feltételezése, hogy a mintaegyedek több Gauss-eloszlás keveréke által lettek generálva, amiknek paraméterei nem ismertek. Azt kell meghatározni, hogy melyikből származnak.

A Gauss-i eloszlásoknak két paramétere van: μ a várható értéket jelöli, Σ pedig a szóródást adja meg. Paraméterek változtatására a függvény így reagál:



GMM modellezési eljárás

- A két ábrán két Gauss-i keverék eljárásból létrejövő modellt láthatunk.
- A felsőn jól látszik, hogy a mintaegyedek abba a klaszterbe lesznek besorolva, amelyhez a legnagyobb valószínűség tartozik.
- Az alsó képen pedig megfigyelhetjük azokat a Gauss-i haranggörbéket, amelyek a modellezés által jöttek létre.
- Az eljárás azt mondja, hogy az adathalmaz, ha véletlen minta lenne, ebből a komplex eloszlásból számazhatna!



A generatív folyamat [ábrázolás]

🐍 Ismerni kell előre a k keverékek számát, továbbá X adathalmazt.

🐍 Minden mintaegyedhez véletlenszerűen klasztert rendelünk a k klaszterből. Annak a valószínűsége, hogy a j -edik klaszter választjuk, a klaszter súlya: $\varphi^{(j)}$. A i -edik egyedhez rendelt klaszter $z^{(i)}$.

🐍 Ha $z^{(i)} = j$, azaz az i -edik egyed j -edik klaszterhez rendelődik, az i -edik mintaegyed pozíciója szerint véletlen mintát ($\mathbf{x}^{(i)}$) veszünk abból a Gauss-eloszlásból, aminek a várható értéke $\mu^{(j)}$ és a kovariancia mátrixa $\Sigma^{(j)}$. Ennek jelölése: $\mathbf{x}^{(i)} \sim \mathcal{N}(\mu^{(j)}, \Sigma^{(j)})$, azaz a $p(x_i | z_i = k, \mu_k, \Sigma_k)$ valószínűség.

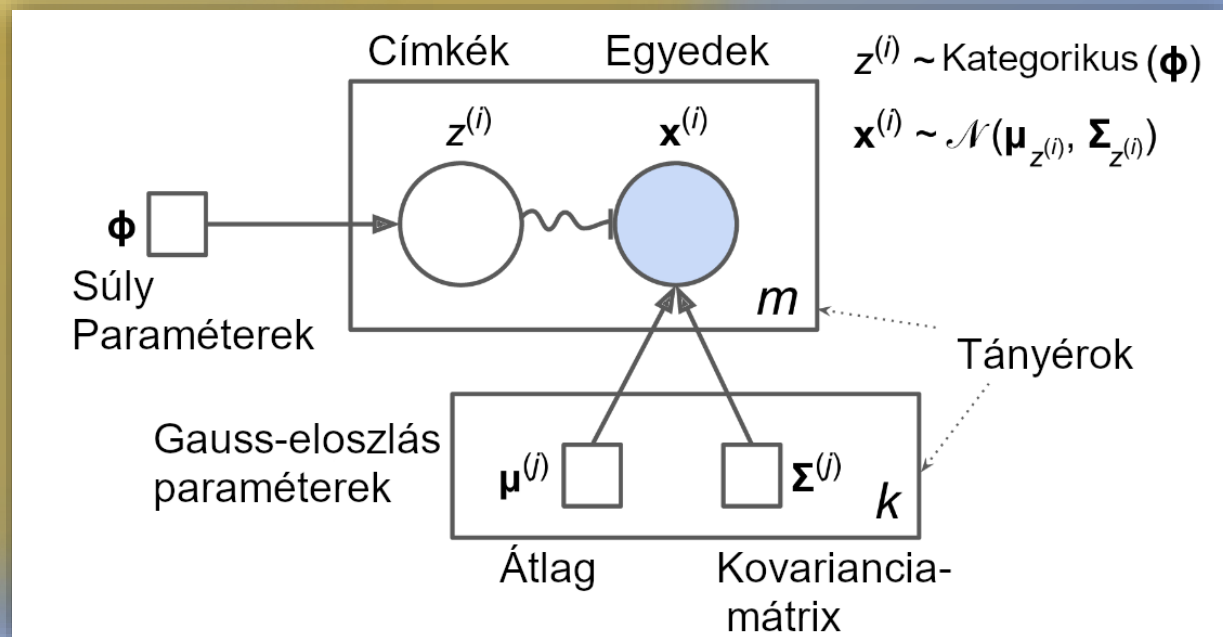
🐍 A körök véletlen változók

🐍 A téglalapok fix értékek

🐍 A nagy téglalapok tényérok: a tartalmuk ismétlődik, m és k a számosságuk

🐍 Az egyenes nyilak feltételes függések:
pl: a véletlen változók függése a súlyoktól

🐍 A hullámos nyilak kapcsolók:
pl: $\mathbf{x}^{(i)}$ különböző Gauss-eloszlásokból származhat a $z^{(i)}$ értékétől függően



Na jó, de mit lehet egy ilyennel kezdeni?

🐍 Legelőször a φ súlyok értékeit, és az eloszlási paramétereket ($\mu^{(k)}$, $\Sigma^{(k)}$) szeretnénk megtalálni. A Scikit *GaussianMixtures* osztálya ezt triviálissá teszi.

🐍 Generáljunk random adatokat

```
X1, y1 = make_blobs(n_samples=1000, centers=((4, -4), (0, 0)), random_state=42)
X1 = X1.dot(np.array([[0.374, 0.95], [0.732, 0.598]]))
X2, y2 = make_blobs(n_samples=250, centers=1, random_state=42)
X2 = X2 + [6, -8]
X = np.r_[X1, X2]
y = np.r_[y1, y2]
```

🐍 A scikit-learn GMM osztálya

```
from sklearn.mixture import GaussianMixture
```

```
gm = GaussianMixture(n_components=3, n_init=10, random_state=42)
gm.fit(X)
```

🐍 A Gauss-eloszlások paraméterei

gm.weights_

```
array([0.39025715, 0.40007391, 0.20966893])
```

gm.means_

```
array([[ 0.05131611,  0.07521837],
       [-1.40763156,  1.42708225],
       [ 3.39893794,  1.05928897]])
```

gm.covariances_

```
array([[[ 0.68799922,  0.79606357],
        [ 0.79606357,  1.21236106]],
       [[ 0.63479409,  0.72970799],
        [ 0.72970799,  1.1610351 ]],
       [[ 1.14833585, -0.03256179],
        [-0.03256179,  0.95490931]]])
```

🐍 Konvergált az algoritmus?

gm.converged_

```
True
```

🐍 Hány iteráció alatt?

gm.n_iter_

```
4
```

🐍 Klaszterek egyedekhez rendelése

gm.predict(X)

```
array([0, 0, 1, ..., 2, 2, 2], dtype=int64)
```

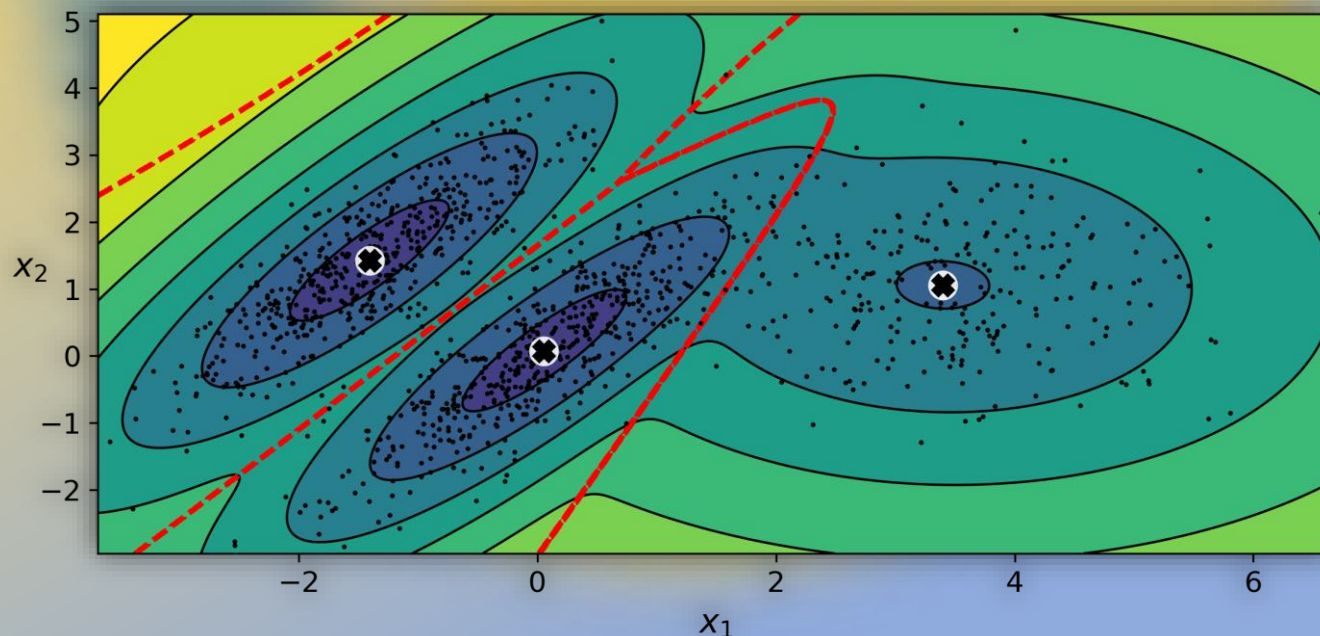
🐍 Valószínűségek becslése

gm.predict_proba(X)

```
array([[9.76741808e-01, 6.78581203e-07, 2.32575136e-02],
       [9.82832955e-01, 6.76173663e-04, 1.64908714e-02],
       [7.46494398e-05, 9.99923327e-01, 2.02398402e-06],
       ...,
       ...])
```


A haranggörbék keveréke

- 🐍 A Gauss-i keverékek alapja az **EM** (Expectation-Maximization) algoritmus.
- 🐍 Az EM kezdete a véletlenszerű centroid-iníciálizáció, majd két lépést ismételt a konvergálásig: egyedek klaszterhez rendelése, majd a centroidok frissítése. Ismerős?
- 🐍 Igen! Az EM a K-közép generalizált változata, ami nem csak a centroidokat találja meg, hanem a méretüket, orientációjukat és a relatív súlyaikat is.

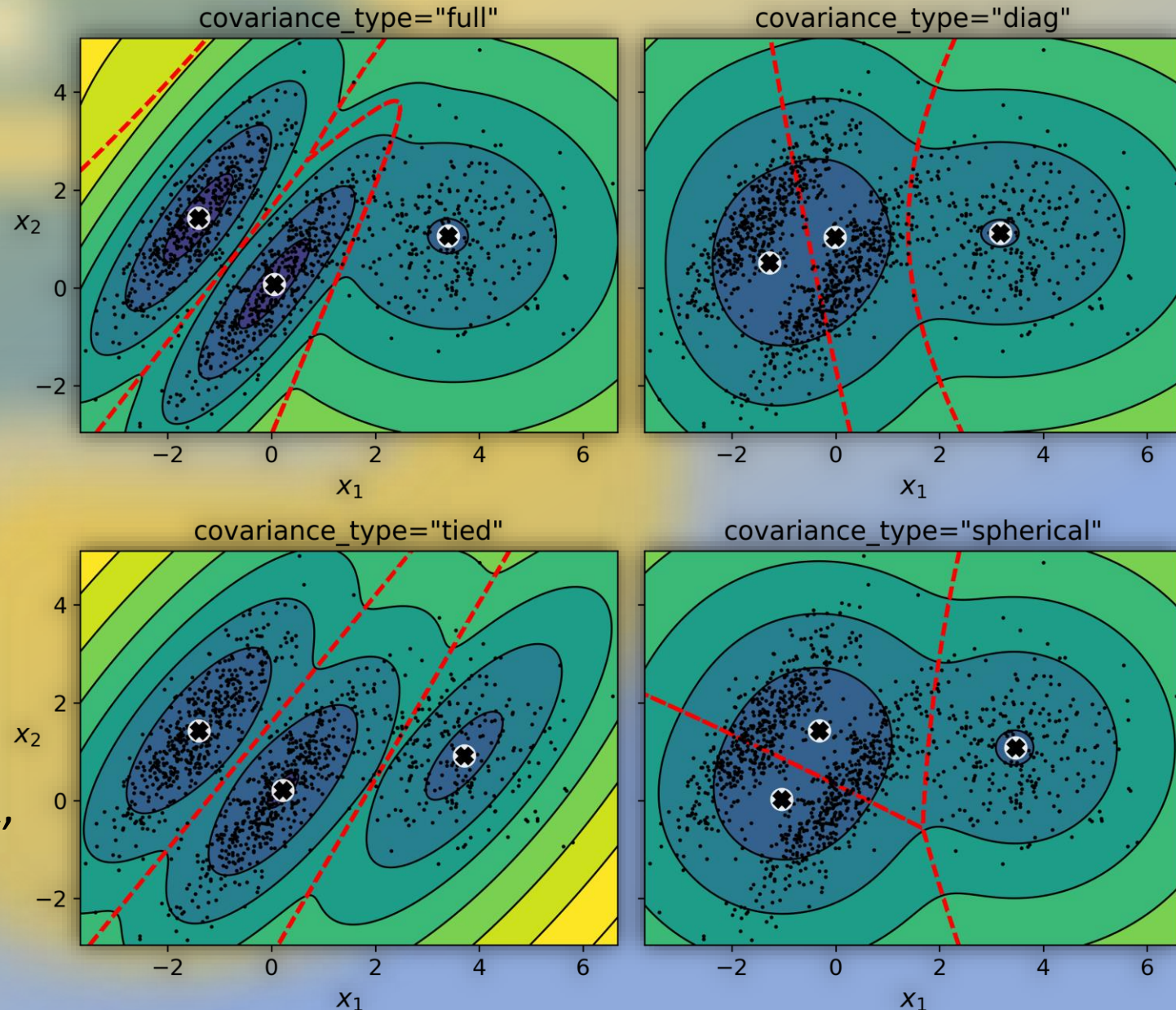


Regularizáció a keverékek esetén

A Gauss-i keverékek esetén a regularizáció a kovariancia mátrixokra tett megkötésekkel érhető el.

Ez a *covariance_type*:

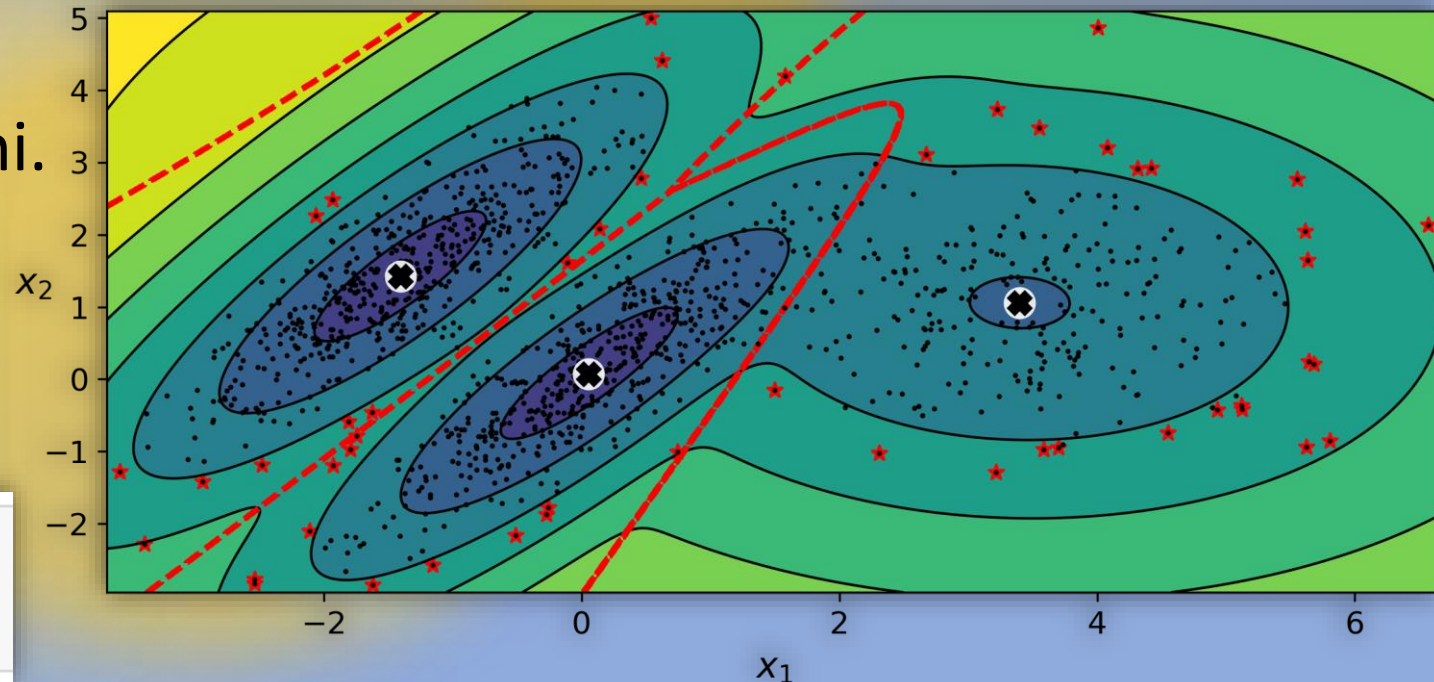
- spherical*: kör alakú klaszterek, különböző átmérőkkel (szórással)
- diag*: csak ellipszoid alakú lehet a klaszter, és a tengelyeinek a koordináta-rendszer tengelyeivel párhuzamosnak kell lennie.
- tied*: minden létrejövő klaszternek ugyanolyan formájúnak, méretűnek, és orientációjúnak kell lennie.
- full*: nincs regularizáció (ez az alapérték)



Anomália-detekció Gauss-i keverékekkel

- 🐍 Az anomália (**outlier**) keresés az az eljárás, amikor olyan egyedeket derítünk fel, amelyek merőben eltérnek a normától, vagy **inlier** egyedektől.
- 🐍 Ez a Gauss-i keverékek esetén meglehetősen egyszerű: minden mintaegyed, amelyik alacsony sűrűségű helyen van, anomáliának számít. Ehhez a sűrűségi küszöbértéket meg kell határozni.
- 🐍 Ha túl sok a False Positive, a küszöbértéket csökkenteni kell, ha túl sok a False Negative, növelni. Ez a precision/recall tradeoff.
 - 🐍 A küszöbérték legyen a negyedik percentilis:

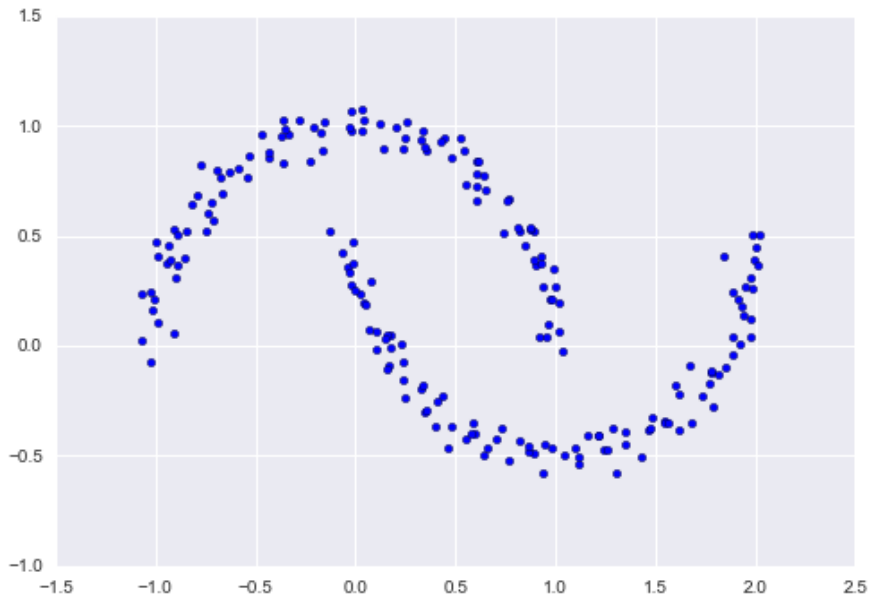
```
densities = gm.score_samples(X)
density_threshold = np.percentile(densities, 4)
anomalies = X[densities < density_threshold]
```



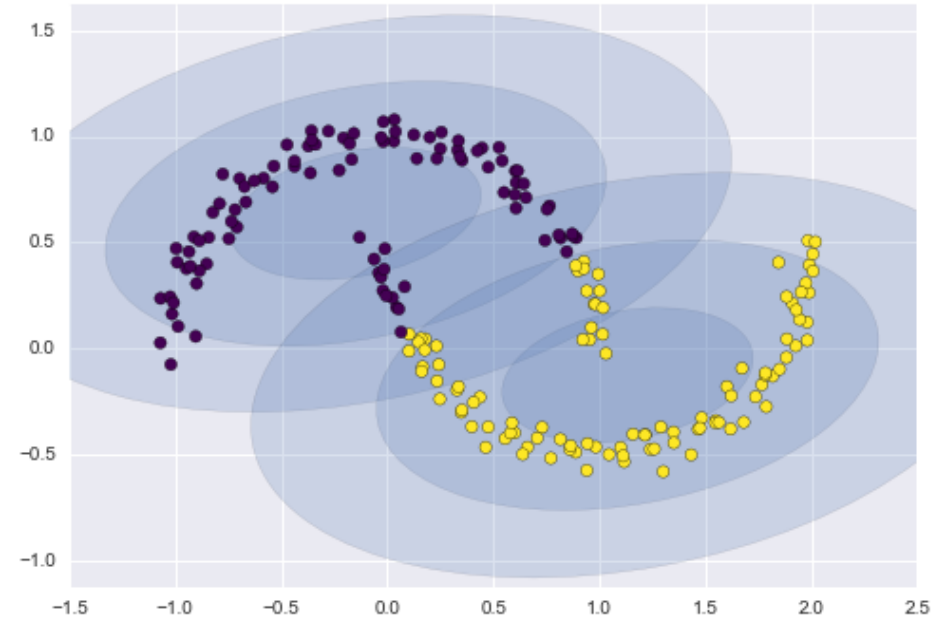
GMM sűrűségek becslésére

- 🐍 A GMM alapjaiban nem egy klaszterező algoritmus, hanem adatpontok sűrűségbecslő algoritmus. Egy adatokra illesztett GMM technikailag nem egy klaszterező, hanem egy probabilisztikus modell, ami a pontok eloszlását írja le.
- 🐍 Próbáljunk meg a *make_moons* által készített holdakra GMM-et illeszteni.

```
from sklearn.datasets import make_moons  
Xmoon, ymoon = make_moons(200, noise=.05, random_state=0)  
plt.scatter(Xmoon[:, 0], Xmoon[:, 1]);
```

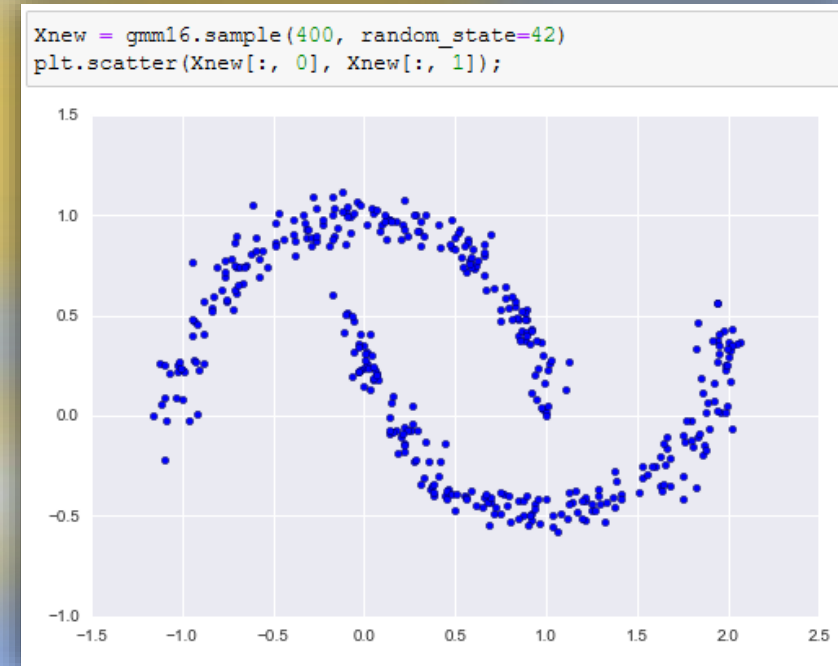
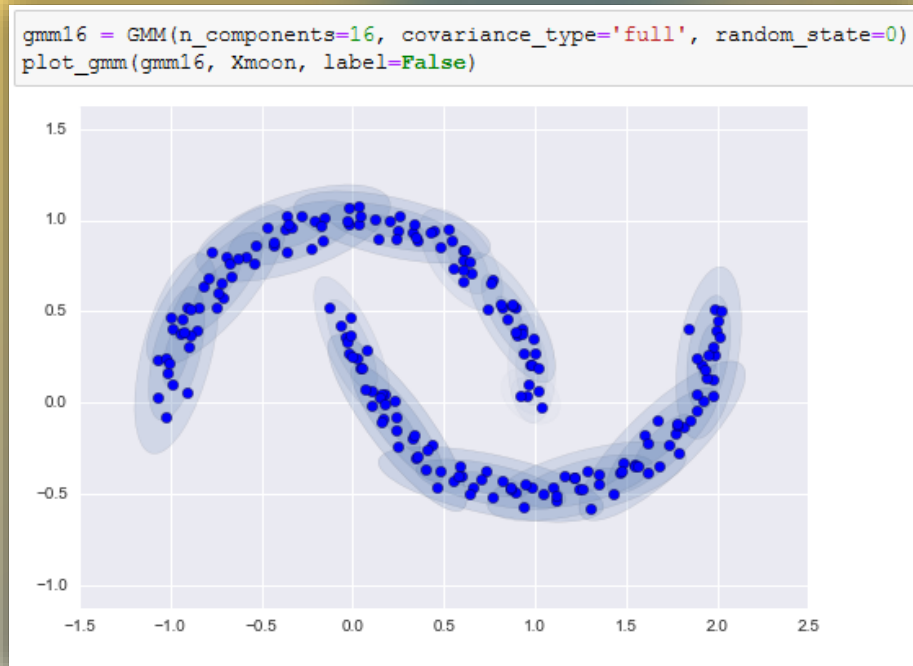


```
gmm2 = GMM(n_components=2, covariance_type='full', random_state=0)  
plot_gmm(gmm2, Xmoon)
```



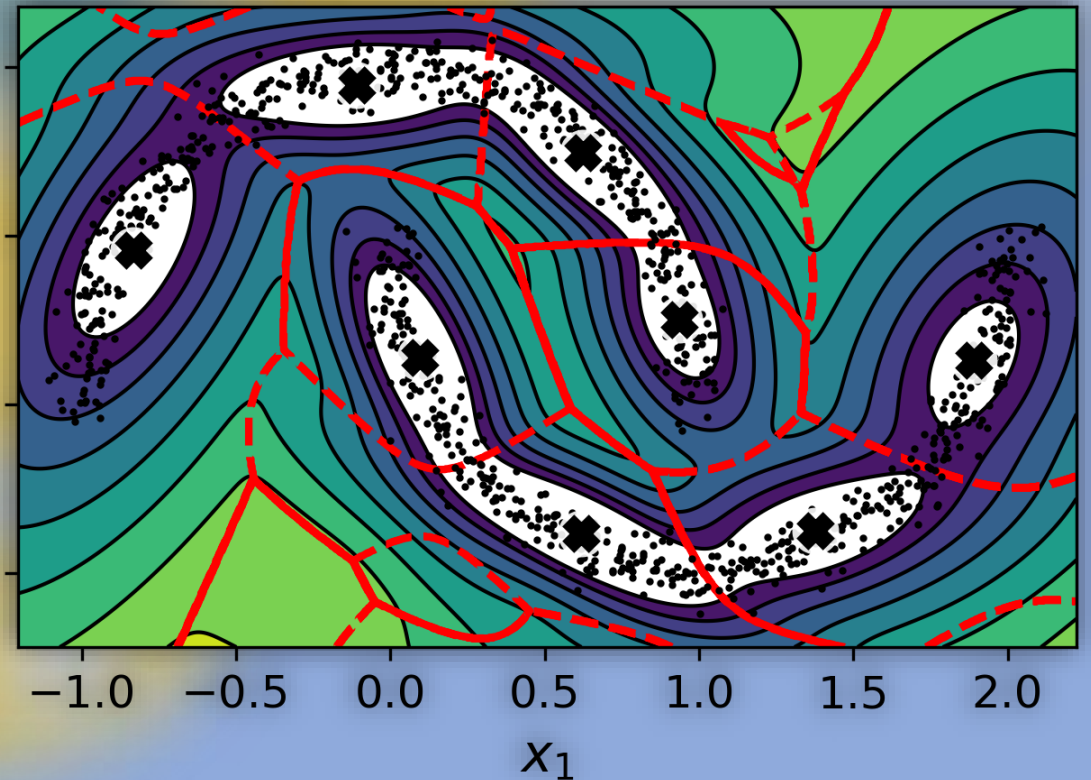
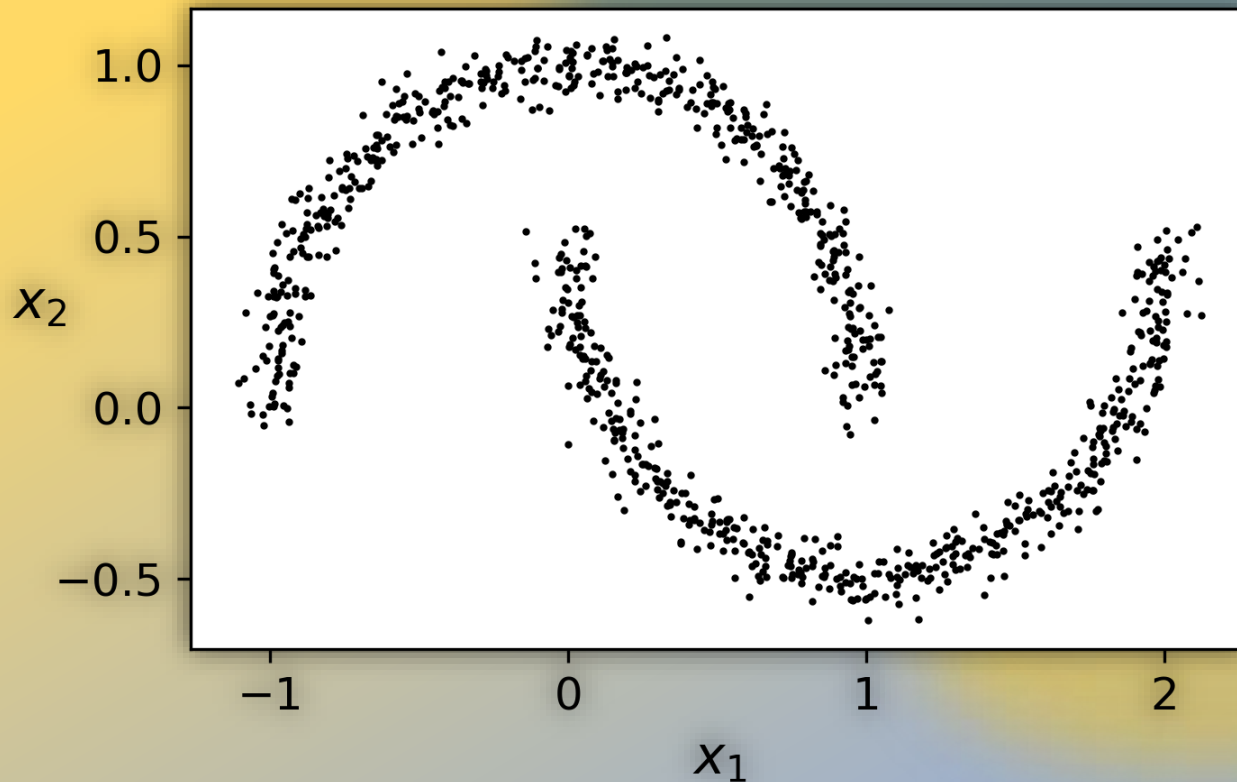
Tehát mi a megoldás?

- Ha megpróbálunk sokkal több komponenst illeszteni az adatokra, látni fogjuk, hogy sokkal közelebb van az input adatokhoz.
- Ebben az esetben a 16 Gauss-eloszlás nem a klasztereket modellezi, sokkal inkább az adatok sűrűségére ad becslést.
- Mivel ez egy generatív modell, lehet őket felhasználva új adatokat generálni, amik közel vannak az eredeti eloszláshoz (bal ábra).



Lehetne a holdakat klaszterezni?

- 🐍 Igen lehetne, de nem Gauss-i keverék modellekkel.
- 🐍 Nézzük meg, mi történne, ha ábrázolnánk a keverék modellek döntési határait.
- 🐍 Talán anomália detekcióra lehetne felhasználni, mert a sűrűségeket jól eltalálta.



Optimális generátorszám megtalálása

🐍 A keverékek esetén nem használható a könyök módszer, és a sziluett sem, mert nem megbízhatóak ellipszis alakú klaszterek esetén.

🐍 Ehelyett egy olyan modellt szeretnénk választani, ami egy teoretikus információs kritériumot minimalizál, mint a **BIC** (Bayesian information criterion) és az **AIC** (Akaike information criterion).

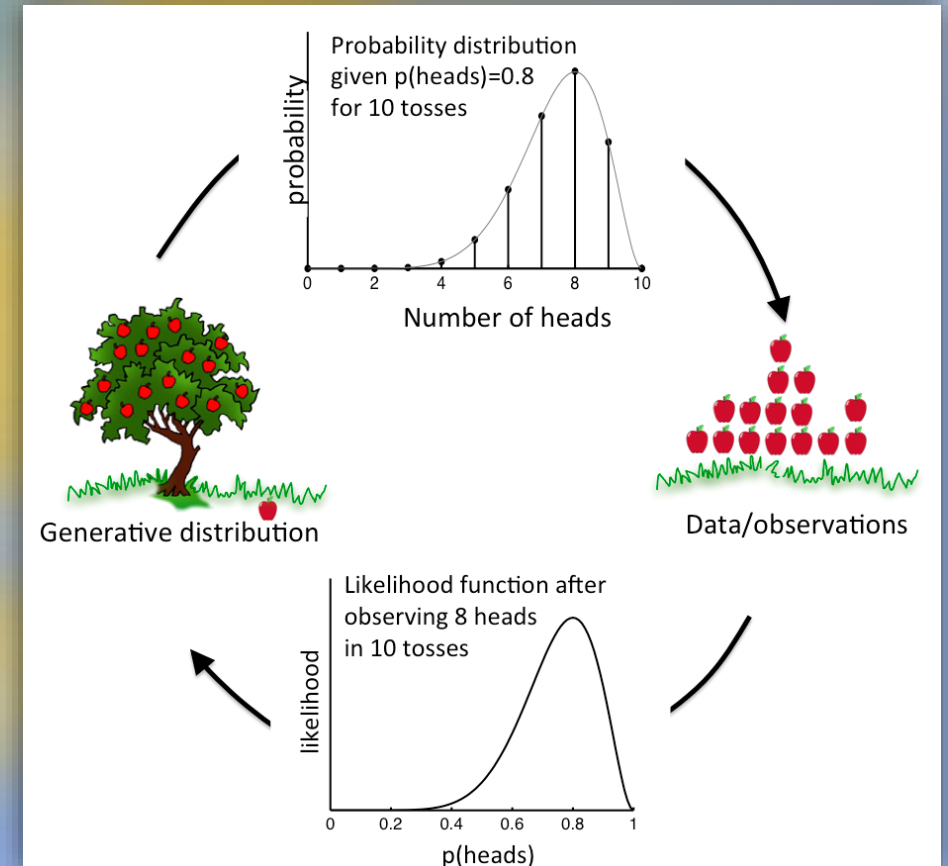
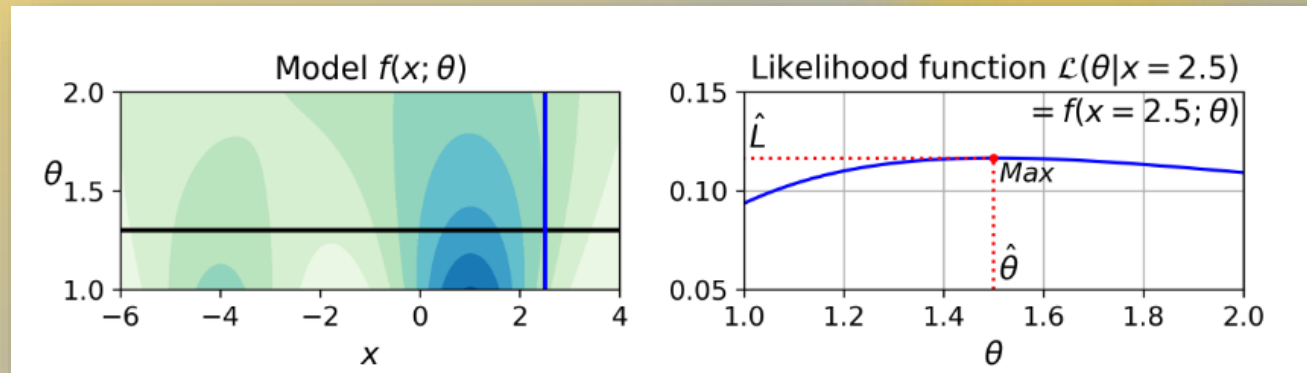
$$\text{BIC} = \log(m) - 2 \log(\hat{L})$$

$$\text{AIC} = 2p - 2 \log(\hat{L})$$

🐍 m : az egyedek száma

🐍 p : a modell paramétereinek száma

🐍 \hat{L} : a modell [Likelihood-függvényének](#) maximuma



A BIC-AIC diagram

- 🐍 A **BIC** és **AIC** kiszámításához hívjuk meg a GM modell BIC és AIC metódusait.
- 🐍 Ezt végezzük el minden k generátorszámra, és ahol minimumuk van, lesz az optimális k érték.

```
gm.bic(X)  
8189.74345832983
```

```
gm.aic(X)  
8102.518178214792
```

```
gms_per_k = [GaussianMixture(n_components=k, n_init=10, random_state=42).fit(X)  
             for k in range(1, 11)]
```

```
bics = [model.bic(X) for model in gms_per_k]  
aics = [model.aic(X) for model in gms_per_k]
```

