

5. Előadás

Együttes tanulás

Adaptív turbózás

Gradiens turbózás

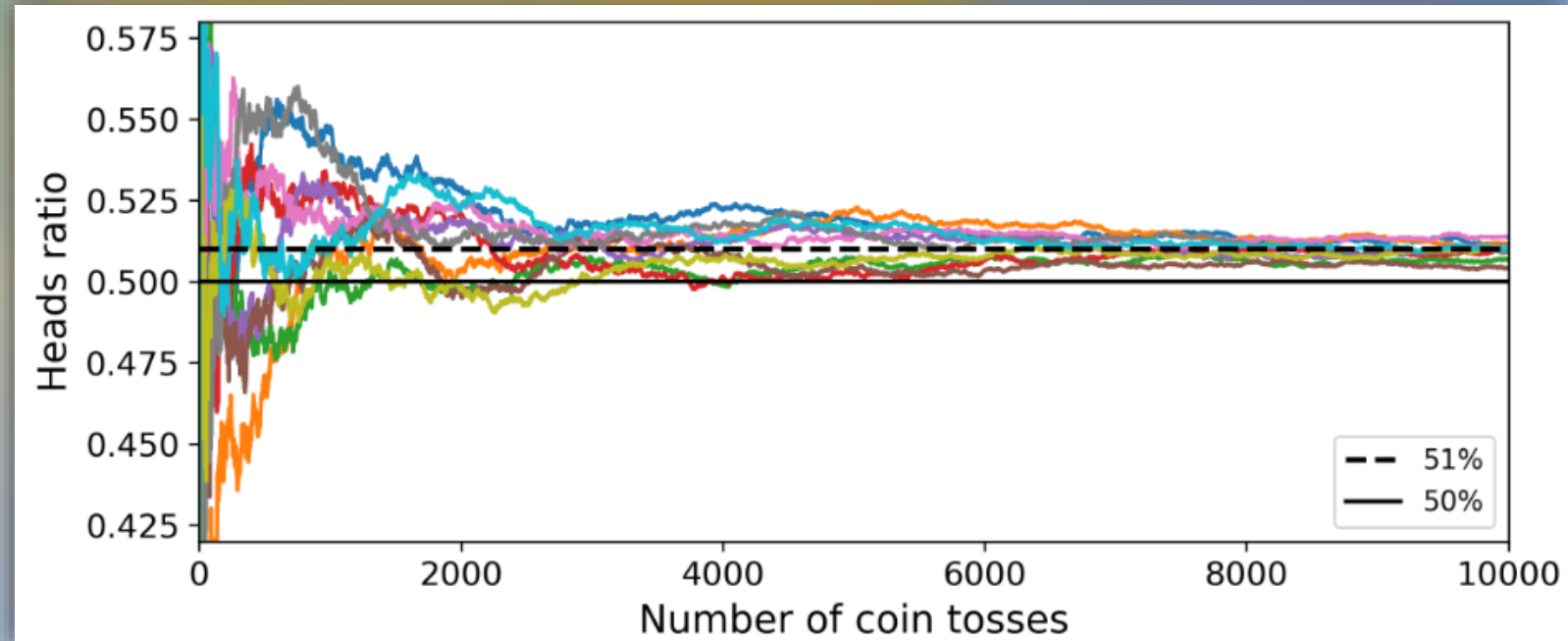
Az együttes tanulás mögötti megfontolás

- 🐍 Hogyan mérjük le, hány fok van a kertben ahhoz, hogy minél jobban tudjuk gondozni, aratni a szőlőt?
- 🐍 A kert egy hegyoldalon fekszik, eltérő hatások érik a felső és alsó tőkéket.



Sok kicsi sokra megy

- 🐍 Meglepő módon, gyenge tanulók gyakran nagyobb pontosságot érnek el közösen, mint az együttes legerősebb modellje.
- 🐍 Még meglepőbb módon, sok gyenge tanuló is lehet erős osztályozó, ha elég sok van belőle, és megfelelő módon választjuk ki az aggregációt.
- 🐍 A nagy számok törvénye: vegyünk egy torzított pénzérmét: 51% fej, 49% írás, valamint 10 gyenge osztályozót, tanítsuk őket 10000 iteráción keresztül.
- 🐍 1000 dobás: kb. 510 fej, 490 írás. Ezután 75%, hogy a modellek többsége fejt fog szavazni.
- 🐍 10000 dobás után ugyanez a valószínűség 97%.



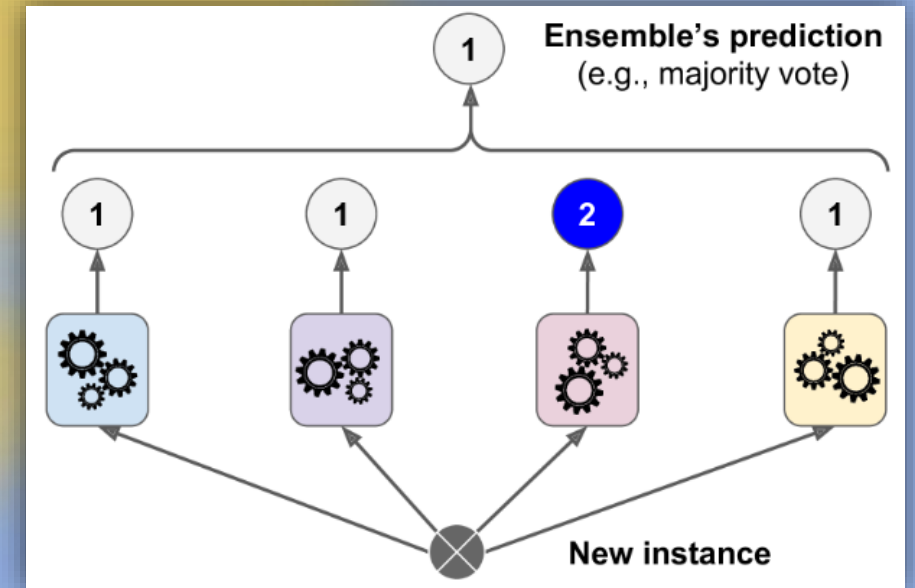
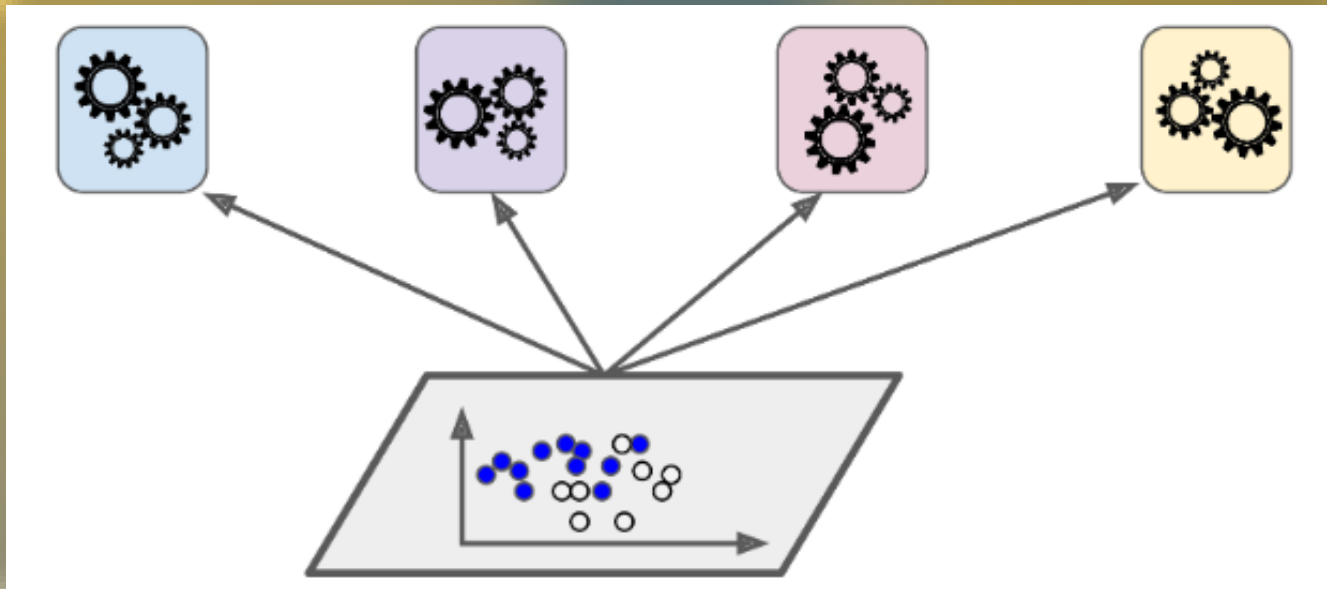
Együttes tanulás: szavazó osztályozók

🐍 **A tömeg bölcsessége:** tegyük fel egy kérdést emberek tömegének, majd a válaszaikat aggregáljuk.

Ez a válasz jobb lesz, mintha egy profi kérdeztünk volna meg?

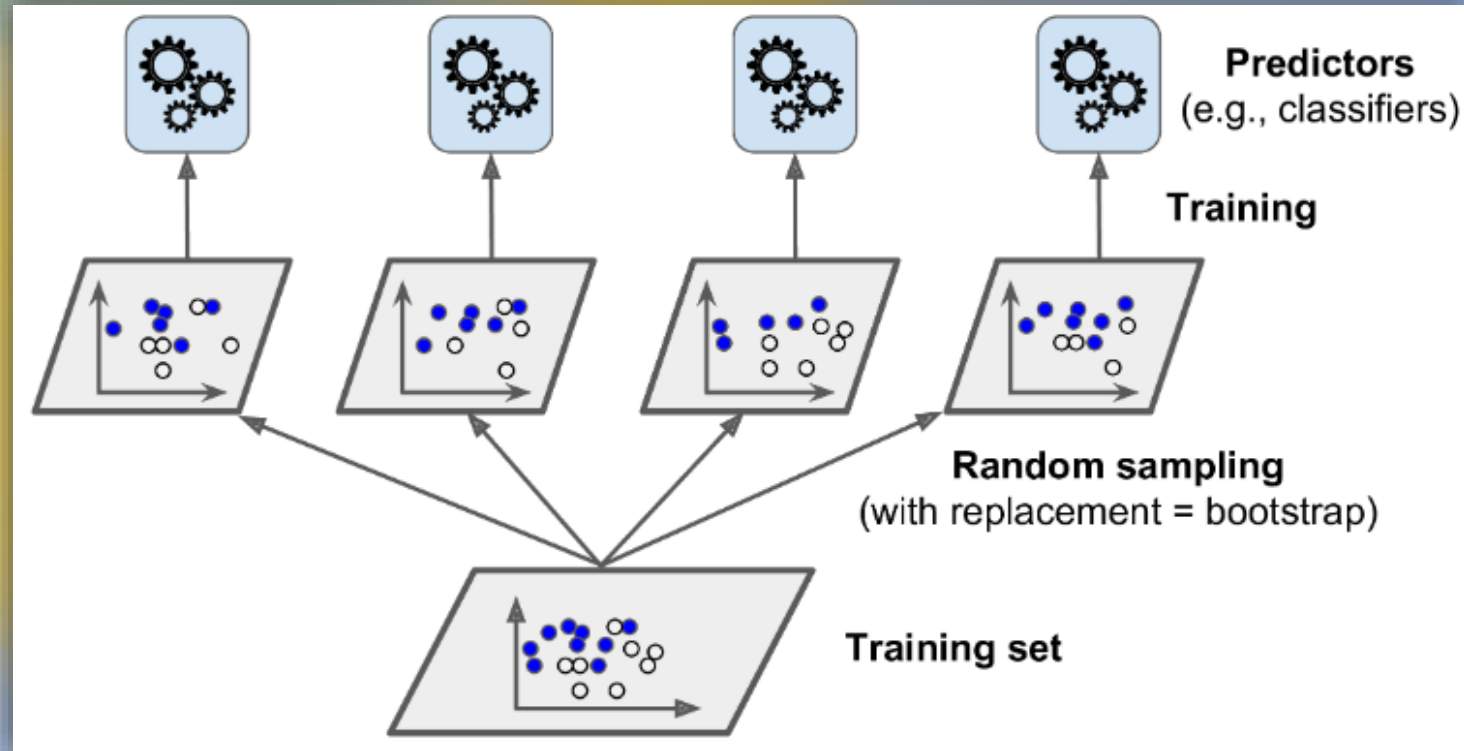
🐍 Pl. vegyünk 4 különböző prediktort, mindegyik kb. 80% pontosságot ért el.

🐍 *Hard voting:* egy egyszerű módja egy még jobb prediktor létrehozásának, ha a válaszokat aggregáljuk, és amelyik a legtöbb szavazatot kapja, azé lesz a predikció.



Bagging & Pasting

- 🐍 Egy másik hozzáállás az együttes tanuláshoz, ha több ugyanolyan modellt tanítatunk, de a tanító adathalmaz különböző véletlen részhalmazain.
- 🐍 Amikor a tanító halmazból való mintavétel visszatevéses, nevezzük az eljárást **pasting**-nek, amikor pedig nem visszatevéses, akkor pedig **bagging**-nek (bootstrap aggregating).
- 🐍 Mi lehet az előnye és hátránya ezeknek az eljárásoknak?
- 🐍 **OOB** (Out-of-Bag) kiértékelés: Bagging esetén azokon a mintaegyedeken történik a validáció, amik nem kerültek bele egyik prediktor tanító pontjai közé sem.



AdaBoost technológia (Adaptive Boosting)

- 🐍 Az együttes tanulás súlyosított változata.
- 🐍 Sok, gyenge fa kombinációjából áll elő a modell.
- 🐍 A prediktorok különböző mértékben járulnak hozzá a modellhez.
- 🐍 Az új fa *tanul* az elődei hibájából. Nézzük lépésről lépésre:
 - 1: Az algoritmus minden egyedhez $1/n$ kezdeti súlyt rendel, ami azt adja meg, hogy az egyedet mennyire fontos helyesen beosztályozni.

Magyarázó változó

Célváltozó

Mintaegyed

x_1	x_2	x_3	y	Súly
				$1/n$
				$1/n$
				$1/n$
\vdots	\vdots	\vdots	\vdots	\vdots

2: Tönkök állítása a mintára

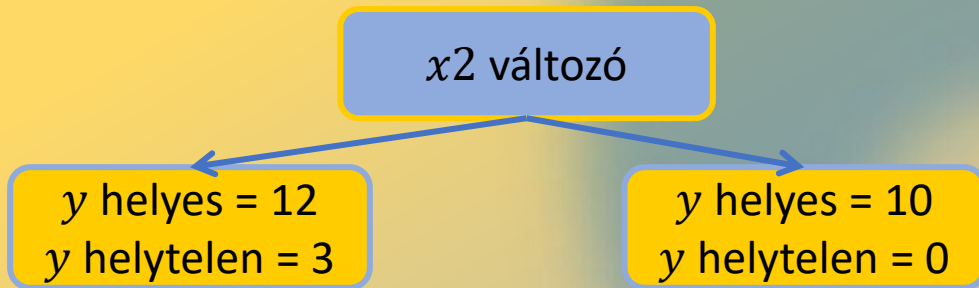
🐍 Azt a változót tartjuk meg, amelyik a legtöbb mintaegyedet osztályozta be helyesen.

🐍 Attól függően kapja meg egy döntési fa a súlyát, hogy mennyire jól osztályozta be a mintaegyedeket.



3: Tönk súlyozása a modellben

🐍 Egy létrehozott modell teljes hibája azon mintaegyedek súlyainak összege, amiket helytelenül osztályozott. Mikor a legrosszabb a prediktor?



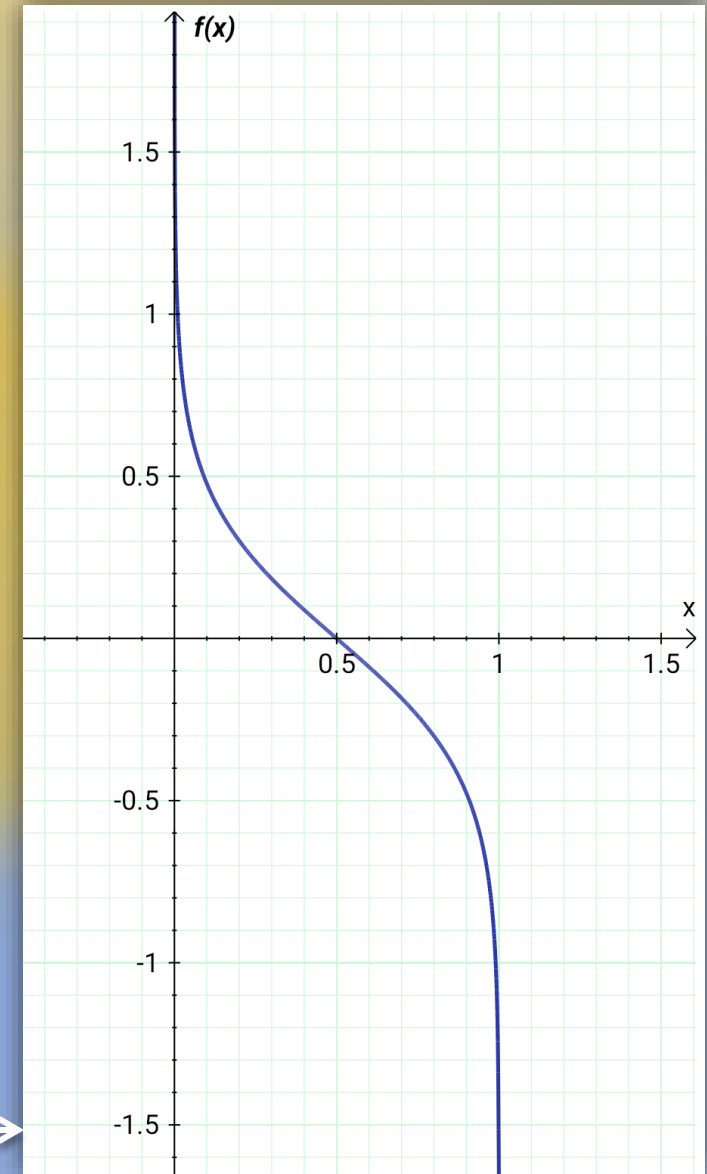
$$(\text{Teljes hiba}) r_j = 3 * \frac{1}{n}$$

Tanulási
sebesség

$$\text{Súly a modellben}(\alpha_j) = \eta * \log\left(\frac{1 - \text{teljes hiba}}{\text{teljes hiba}}\right)$$

$$r_j = \frac{\sum_{i=1}^m w^{(i)} \hat{y}_j^{(i)} \neq y^{(i)}}{\sum_{i=1}^m w^{(i)}}$$

$$\alpha_j = \eta \log \frac{1 - r_j}{r_j}$$



4: Egyedsúlyok frissítése

🐍 Minden **helytelenül** beosztályozott mintaegyedre növelni:

$$\text{Új egyedsúly} = \text{jelenlegi súly} * e^{\text{súly a modellben}}$$

🐍 Minden **helyesen** beosztályozott mintaegyedre csökkenteni:

$$\text{Új egyedsúly} = \text{egyedsúly} * e^{-\text{súly a modellben}}$$

🐍 Miután ez el van végezve minden egyedre, az új súlyokat normalizálni kell, hogy az összegük 1 legyen, mint a kezdőállapotban.

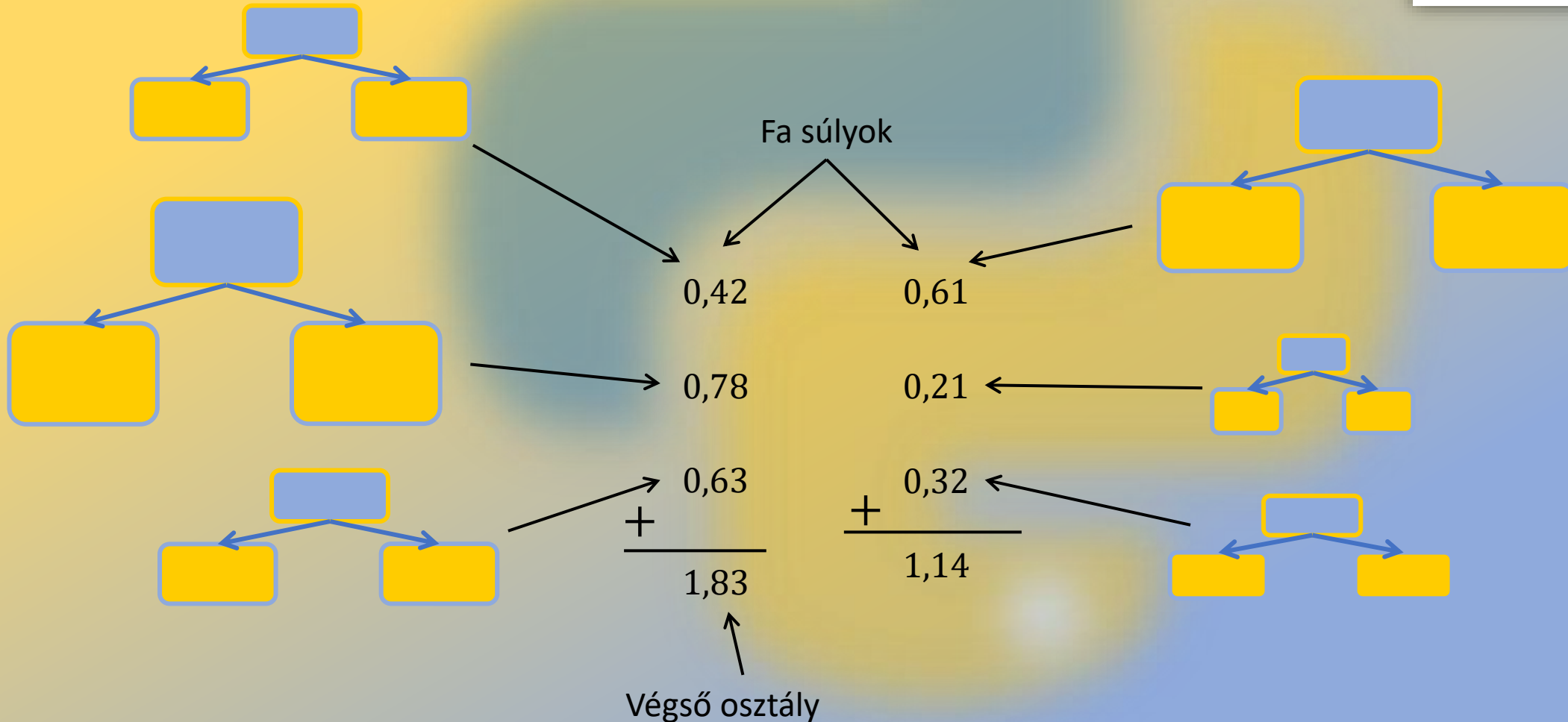
🐍 Ezeket a normalizált súlyokat felhasználva fogja a rákövetkező fa megkapni az ő saját tanuló mintáját, amiben az előző súlyok eloszlásként szerepelnek.

🐍 Az algoritmus ezt a folyamatot addig ismétli, ameddig a fák számossága el nem érte a hiperparaméterül megkapott értéket, vagy nem talált egy tökéletes prediktort.

A predikció eljárása

🐍 Osztályozáskor az egyed abba az osztályba lesz besorolva, amelyikhez tartozó tönkök súlyának összege nagyobb:

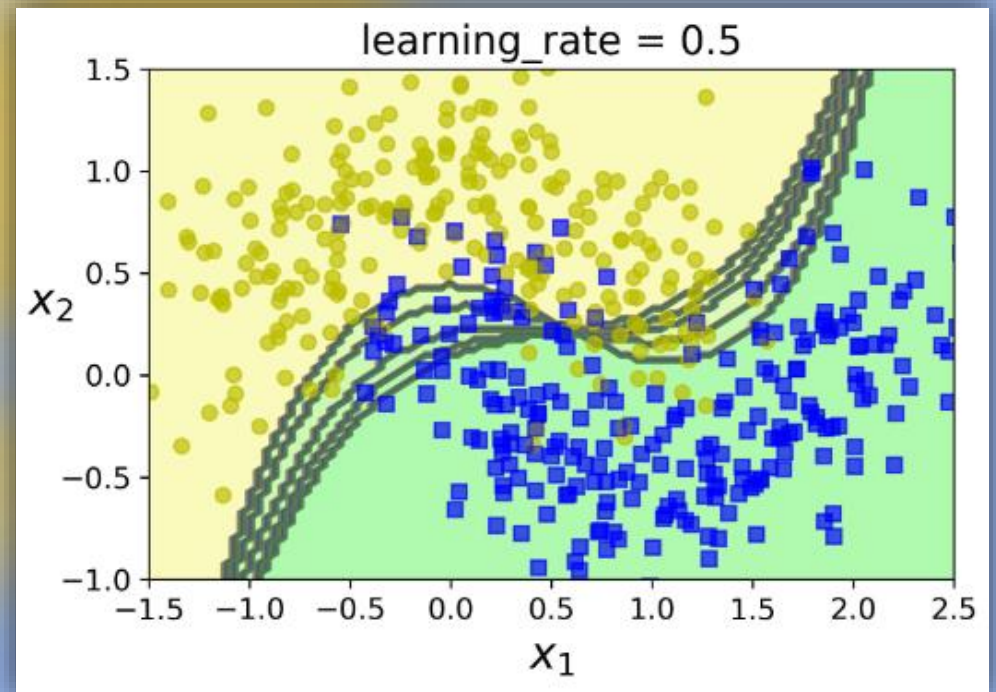
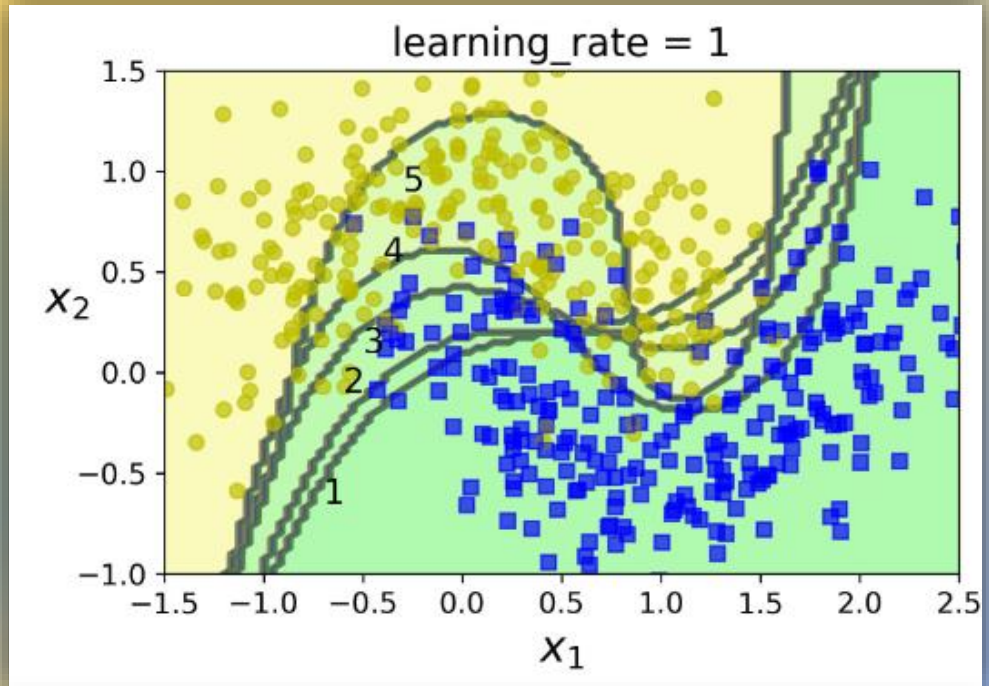
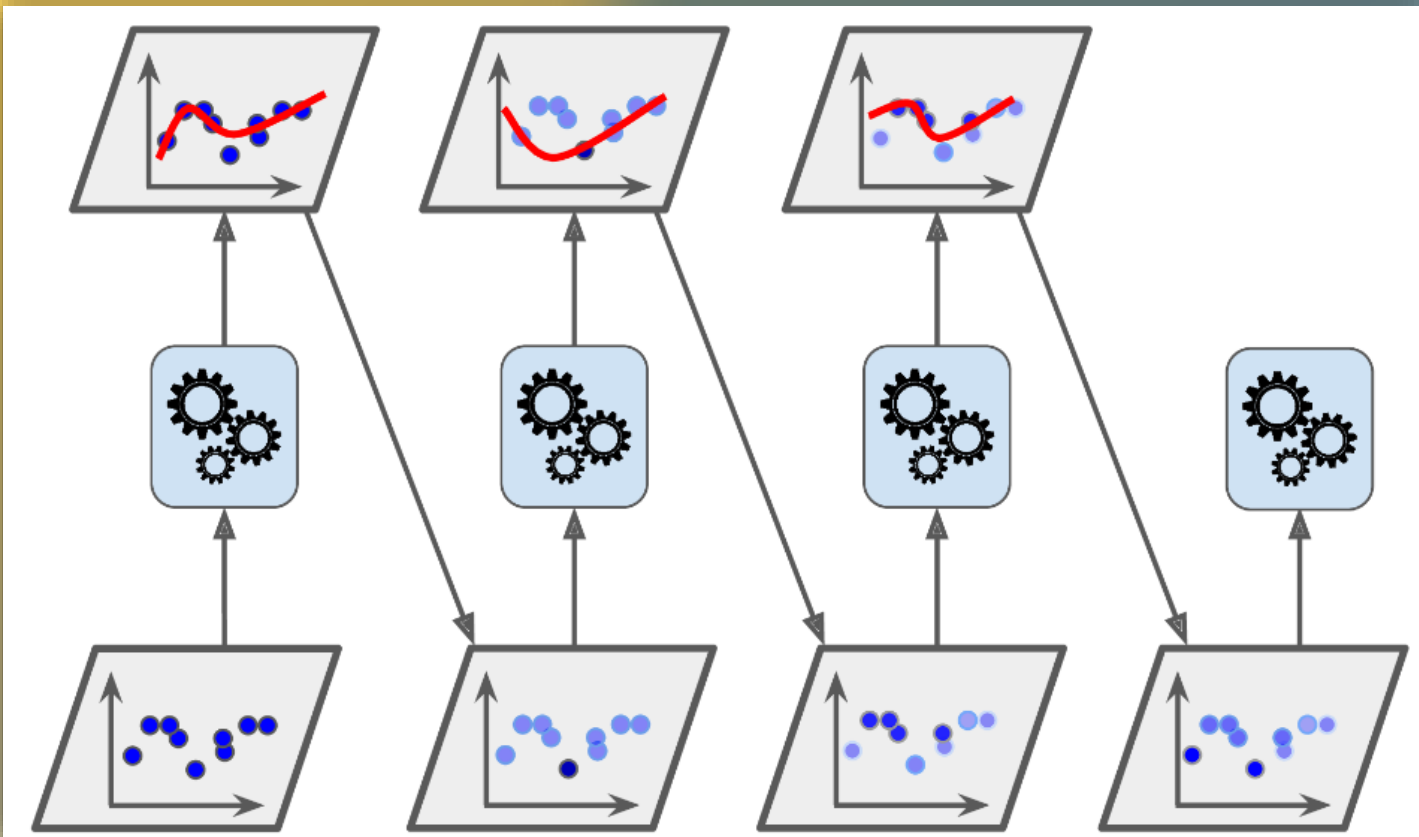
$$\hat{y}(\mathbf{x}) = \underset{k}{\operatorname{argmax}} \sum_{j=1}^N \alpha_j \mathbf{1}_{\hat{y}_j(\mathbf{x}) = k}$$



AdaBoost a Moons halmazon

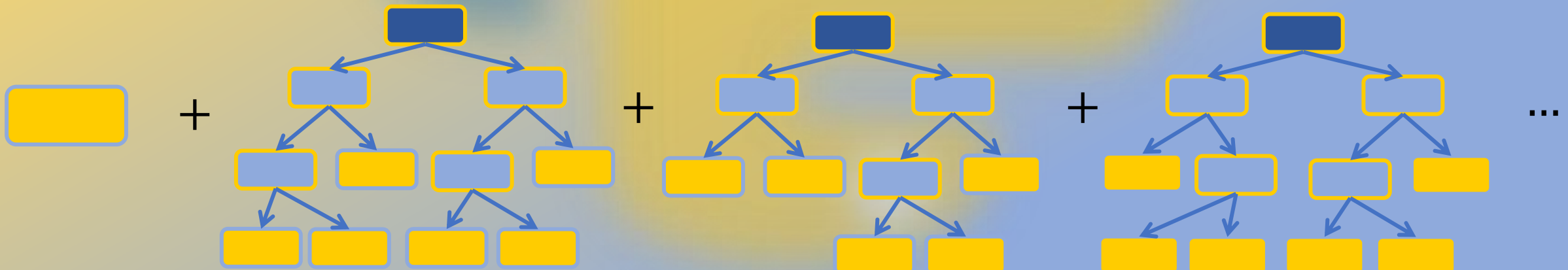
🐍 A képeken egymás utáni prediktorok döntési határait látjuk.

🐍 Mi a hátránya az adaptív turbózásnak?



Gradiens turbózás: az alap elképzelés

- 🐍 Machine learning technológia regresszió- és osztályozás beli problémák megoldására gyenge prediktorok együttesével. Minden prediktor javít az elődje hibáján, és ezt a folyamatot a kilépésig rekurzívan iterálja.
- 🐍 Az egyedek helyett a GBDT a rezidumokat turbózza (javítja szekvenciálisan).
- 🐍 A becsült érték előállítása:
$$\text{becsült érték} = \text{előző predikció} + \text{tanulási sebesség} * \text{rezidum}$$

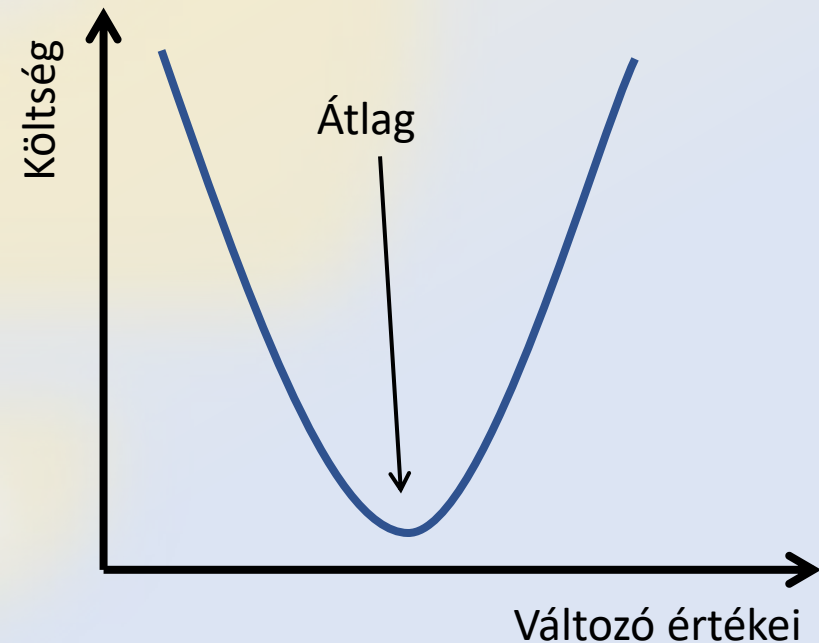
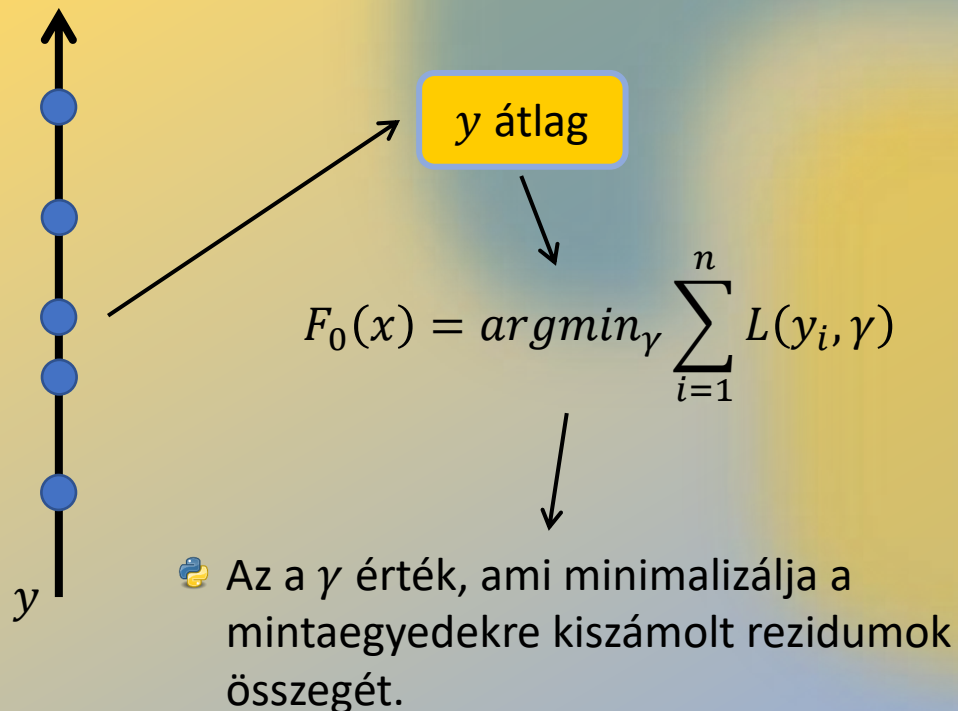


1: A kezdeti modell


🐍 Az input adathalmaz: $\{(x_i, y_i)\}_{i=1}^n$

🐍 Deriválható költségfüggvény: $L(y_i, F(x))$. Eredménye a **rezidum**.

🐍 A GBDT célváltozóra vonatkozó első predikciója annak a várható értéke, mert egy dimenzióban az minimalizálja az eltérés négyzetösszeget.



2: Rezidumok kiszámítása


 A következő lépésben az algoritmus kiszámolja, mennyiben tér el a predikció a valós értékektől. Ez a **rezidum**: a veszteségfüggvénybe behelyettesített valós (y_i) és becsült ($F(x)$) érték.

Magasság	Szemszín	Nem	Súly	Rezidum $r_{i,1}$
1,6	Kék	Férfi	88	16,8
1,6	Barna	Nő	76	4,8
1,5	Kék	Nő	56	-15,2
1,8	Zöld	Férfi	73	1,8
1,5	Barna	Férfi	77	5,8
1,4	Kék	Nő	57	-14,2

$$r_{1,1} = 88 - 71,2$$

$$r_{2,1} = 76 - 71,2$$

:

 r : a pszeudo rezidum, i : a minta indexe és m : a döntési fa, ami épül. A becsült érték szerinti deriválás eredménye az az $F(x)$ becsült érték, ami minimalizálja a költségfüggvényt. $F_{m-1}(x)$ a modell előző fája, ami a második iteráció alatt egy levél, utána pedig teljes döntési fa, ami magában foglalja az előzőek hibáit.

Előző predikció: 71,2

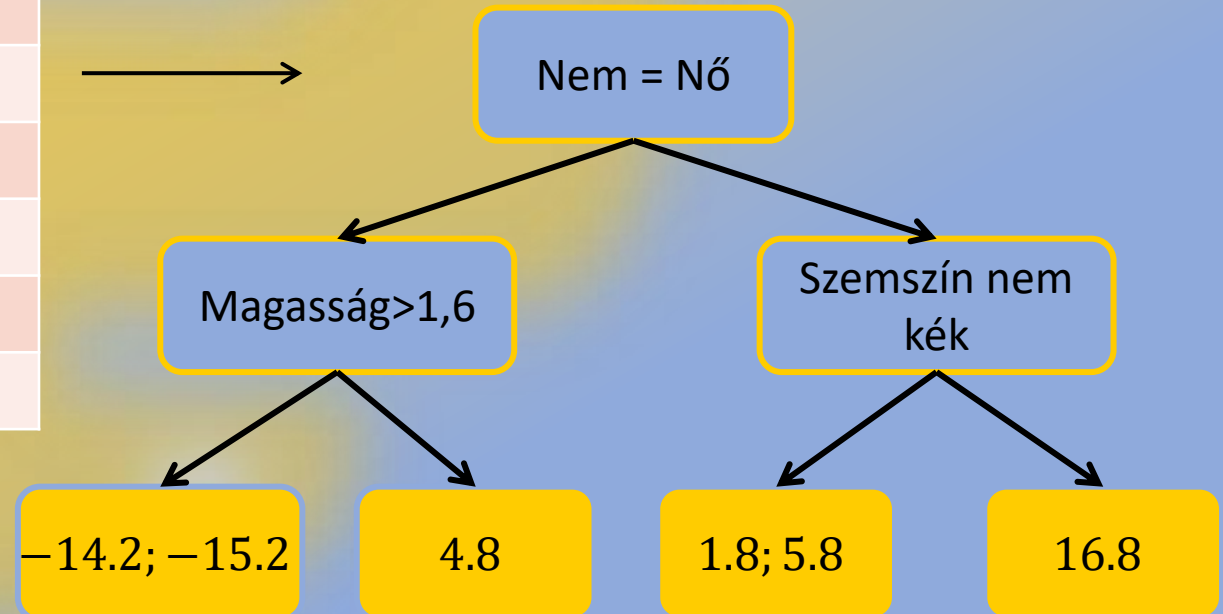
$$r_{i,m} = - \left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} \right]_{F(x)=F_{m-1}(x)}$$

3: Fa illesztése a rezidumokra

🐍 Ebben a lépésben az egyedekhez az algoritmus tartozó rezidumokra illeszt egy döntési fát: besorolja a rezidumokat levelekbe.

🐍 Az illesztett fát fontos és kell is **regularizálni!**

Magasság	Szemszín	Nem	Súly	Rezidum
1,6	Kék	Férfi	88	16,8
1,6	Barna	Nő	76	4,8
1,5	Kék	Nő	56	-15,2
1,8	Zöld	Férfi	73	1,8
1,5	Barna	Férfi	77	5,8
1,4	Kék	Nő	57	-14,2



4: Levelek outputjának kiszámítása

🐍 A levelek outputja az az érték, amelyik minimalizálja a terminális régiókba került mintaegyedekre kiszámított költségfüggvényt.

🐍 Ez az esetek többségében a levélbe bekerült rezidumok átlaga.

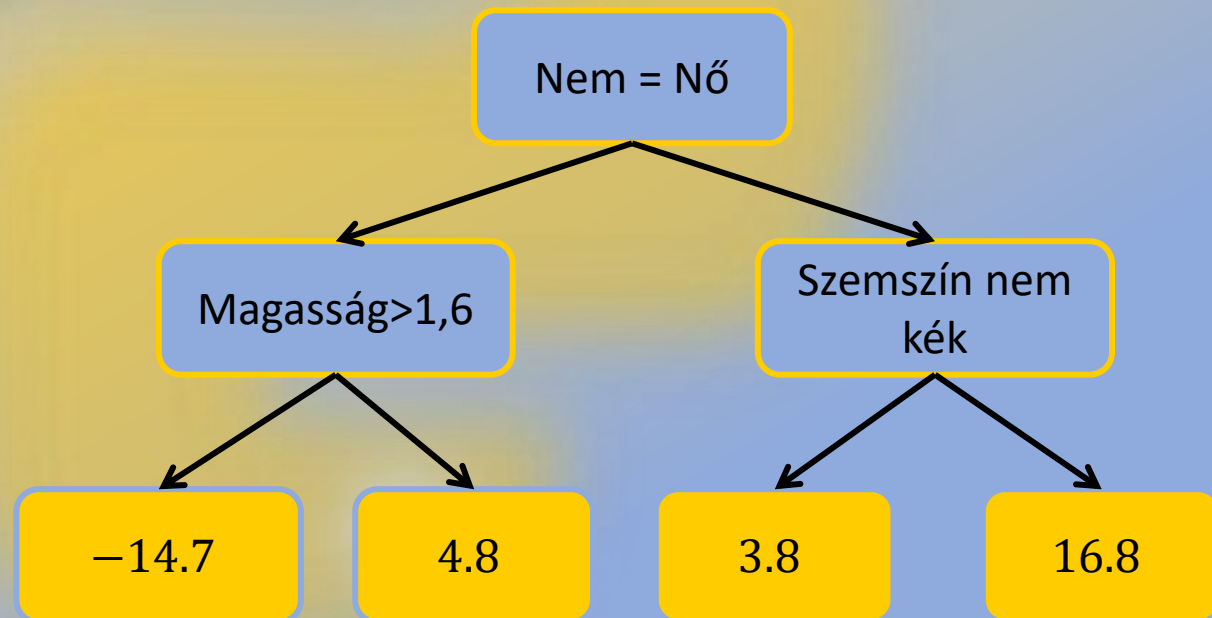
🐍 Matematikailag kifejezve:

$$\gamma_{j,m} = \operatorname{argmin}_{\gamma} \sum_{x_i \in R_{i,j}} L(y_i, F_{m-1}(x_i) + \gamma)$$

🐍 Ahol γ a levél outputja: az a szám, ahol a fenti összegzésnek minimuma van, tehát a terminális régiókba került rezidumok várható értéke.

🐍 $F_{m-1}(x_i)$ az előző fa által adott becsült érték a valós x_i értékek függvényében.

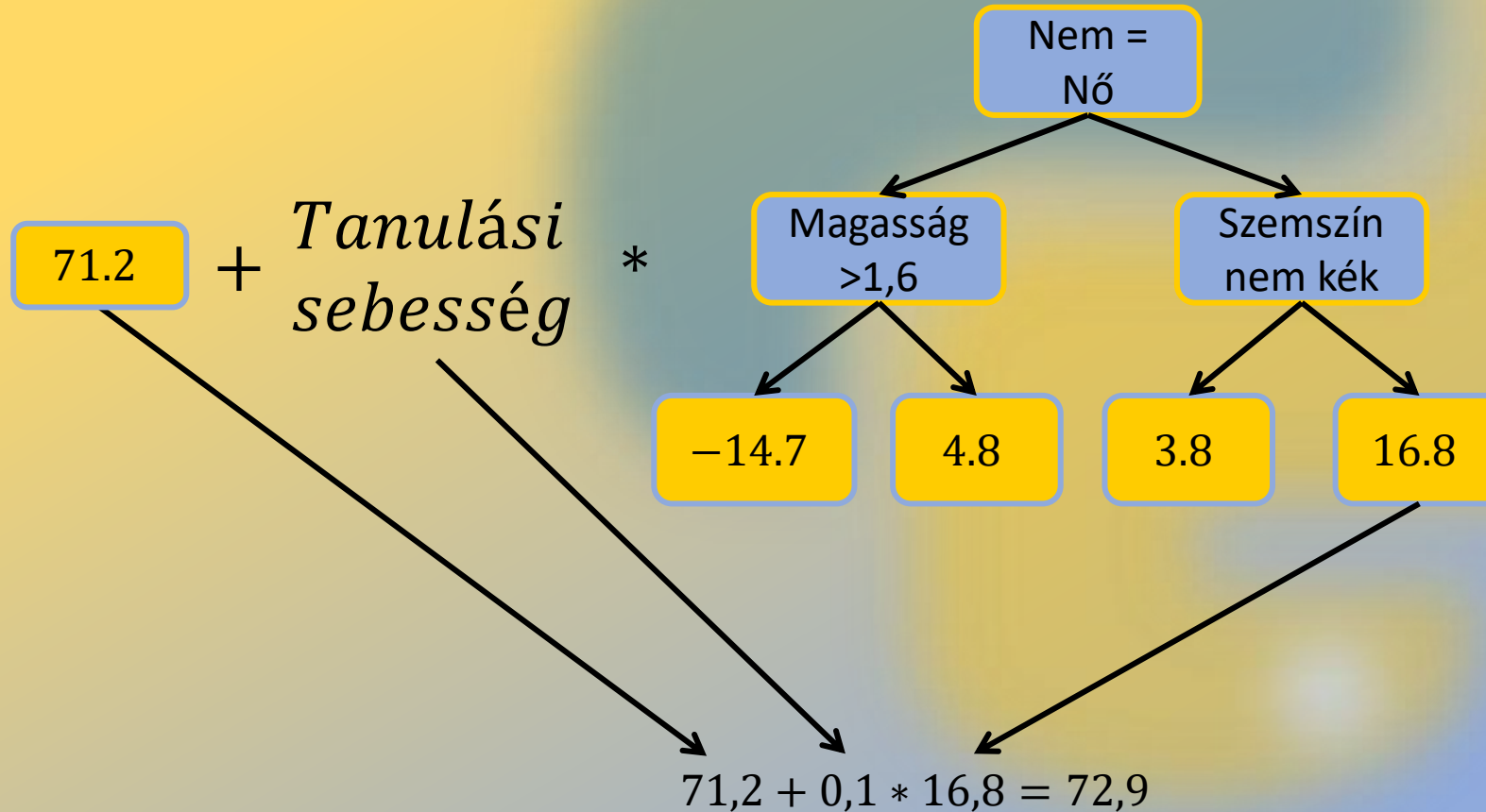
🐍 Az y_i pedig a célváltozó valós értéke, amit a modell összehasonlít a becsült értékkel, a költségfüggvény (L) segítségével.



5: Predikciók készítése a tanító adatokra

🐍 Vegyük az egyik mintaegyedet:

Magasság	Szemszín	Nem	Súly
1,6	Kék	Férfi	88



$$F_m(x) = F_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_{j,m} I(x \in R_{j,m})$$

- 🐍 Ahol $F_{m-1}(x)$ az x mintaegyedre előző fa által adott becsült érték.
- 🐍 ν tanulási sebesség paraméter, ami csökkenti minden fa hozzájárulását a végleges modellhez.
- 🐍 A terminális régiók száma J , az aktuálisan vizsgált levél jele pedig j .

Az első iterációnak vége. És kezdődik előről.

$$88 - (71,2 + 0,1 * 16,8)$$

Magasság	Szemszín	Nem	Súly	Rezidum $r_{i,2}$
1,6	Kék	Férfi	88	15,1
1,6	Barna	Nő	76	4,3
1,5	Kék	Nő	56	-13,7
1,8	Zöld	Férfi	73	1,4
1,5	Barna	Férfi	77	5,4
1,4	Kék	Nő	57	-12,7

Rezidum $r_{i,1}$
16,8
4,8
-15,2
1,8
5,8
-14,2

🐍 Új rezidumok kiszámítása a valós és becsült értékek veszteségfüggvénybe való helyettesítésével.

🐍 A teljes iteráció algoritmus:

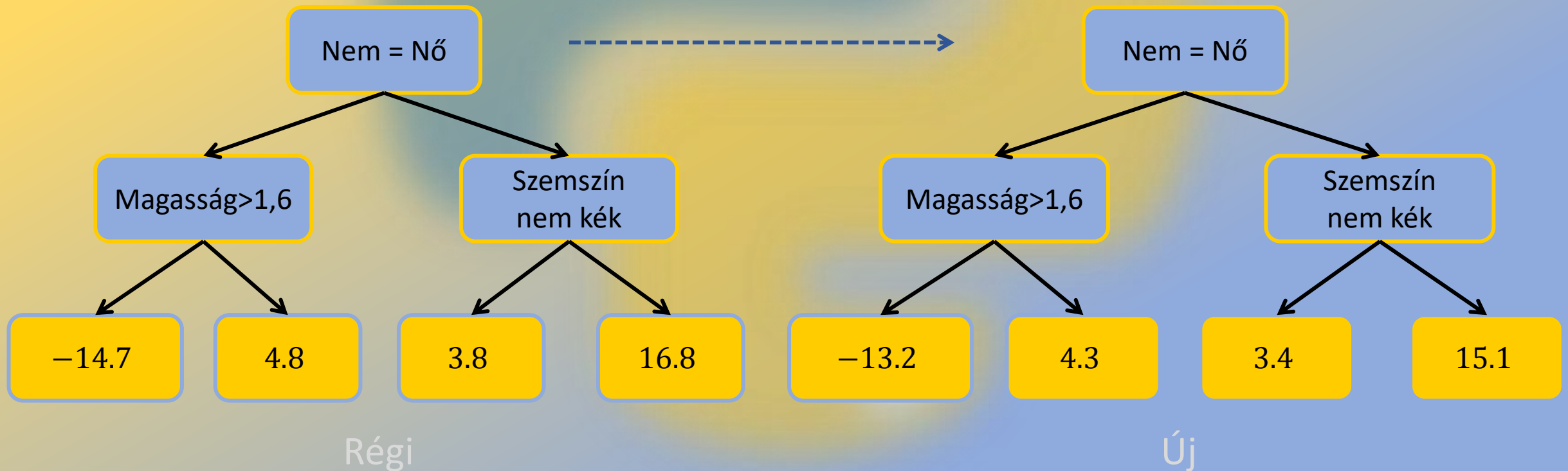
Algorithm 1: Gradient_TreeBoost

```
1  $F_0(\mathbf{x}) = \arg \min_{\gamma} \sum_{i=1}^N \Psi(y_i, \gamma).$ 
2 For  $m = 1$  to  $M$  do:
3    $\tilde{y}_{im} = - \left[ \frac{\partial \Psi(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)} \right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})}, i = 1, N$ 
4    $\{R_{lm}\}_1^L = L - \text{terminal node } tree(\{\tilde{y}_{im}, \mathbf{x}_i\}_1^N)$ 
5    $\gamma_{lm} = \arg \min_{\gamma} \sum_{\mathbf{x}_i \in R_{lm}} \Psi(y_i, F_{m-1}(\mathbf{x}_i) + \gamma)$ 
6    $F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + v \cdot \gamma_{lm} 1(\mathbf{x} \in R_{lm})$ 
7 endFor.
```

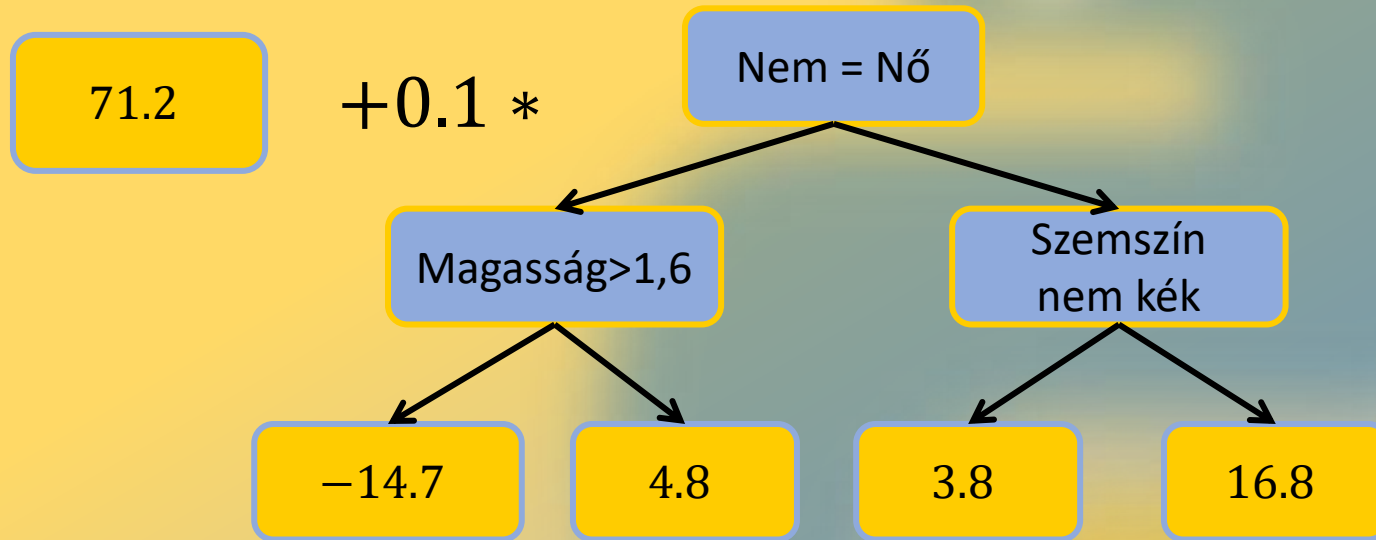
Az új döntési fa

🐍 A rezidumok megbecslésére kiélezve.


🐍 Az új rezidumok kevesebbek lesznek, mint az előző fáé.

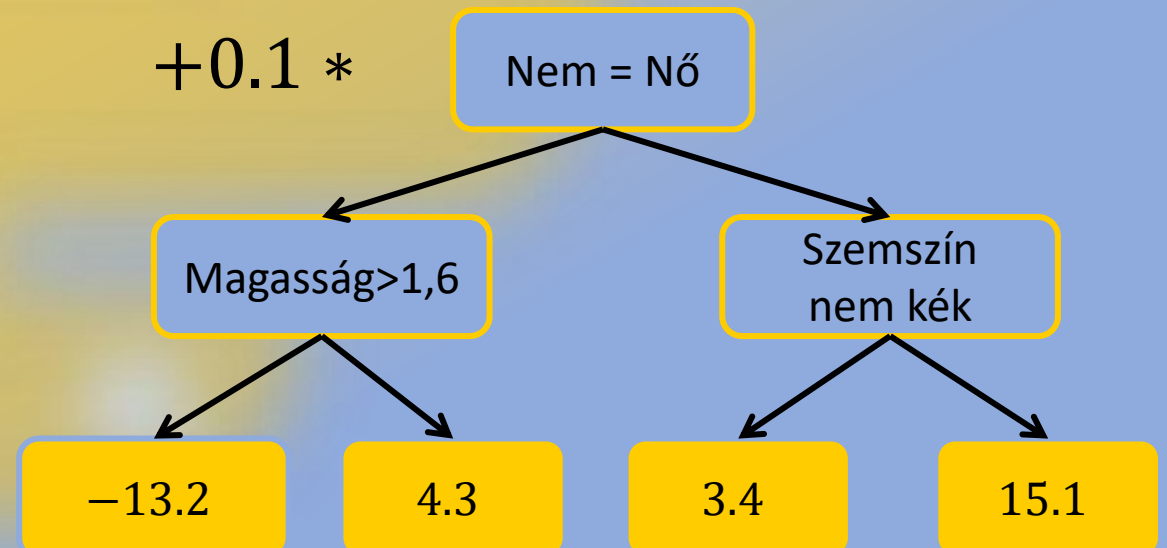


A modell három iteráció alatt



 A döntési fák együttese az erdő.

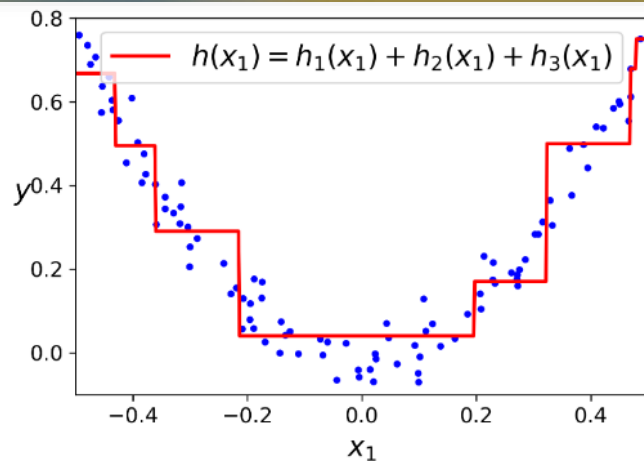
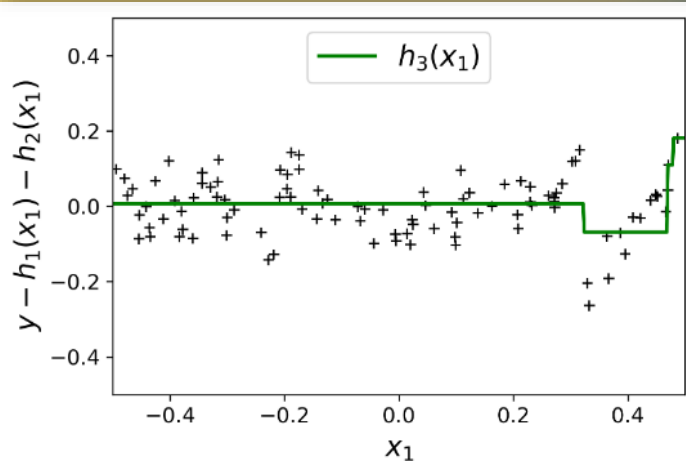
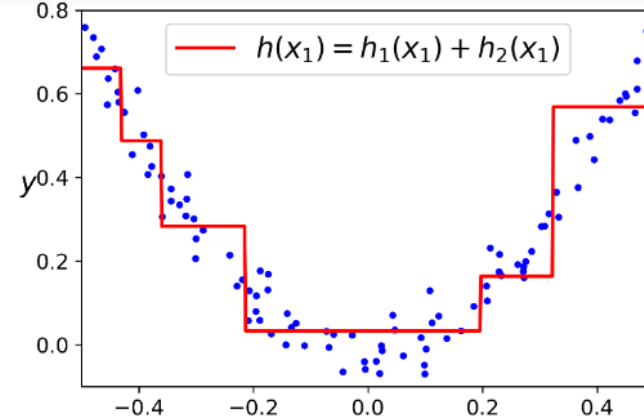
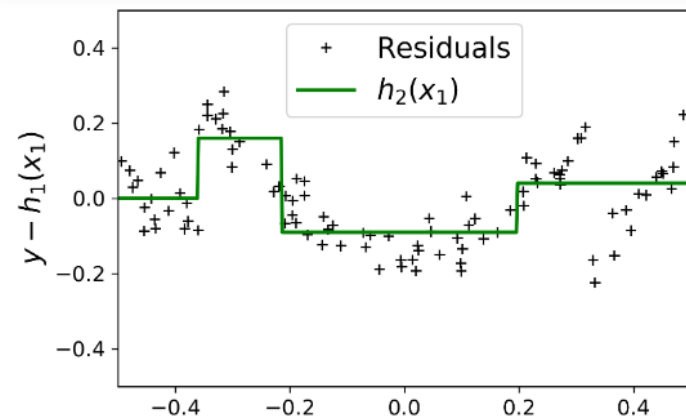
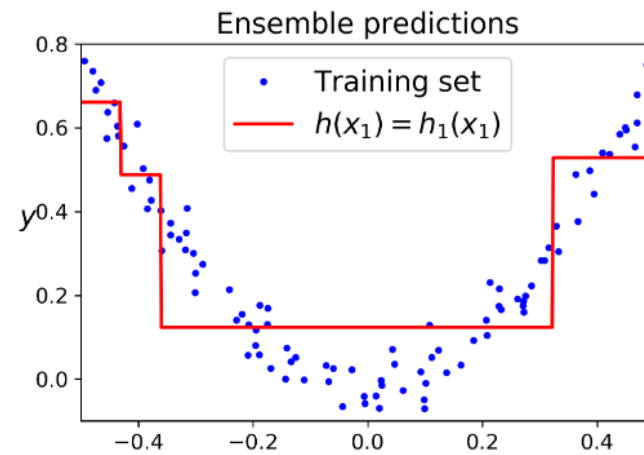
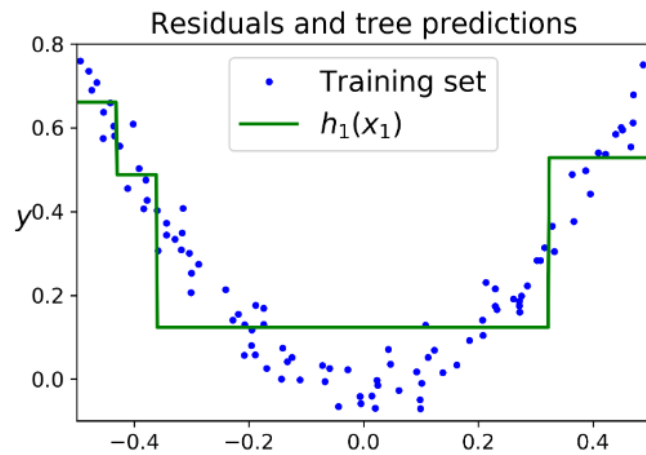
 Az iteráció addig folytatódik, ameddig a fák el nem érik a kívánt számosságot, vagy a fa nem tud *érdebben* hozzátenni a modellhez.



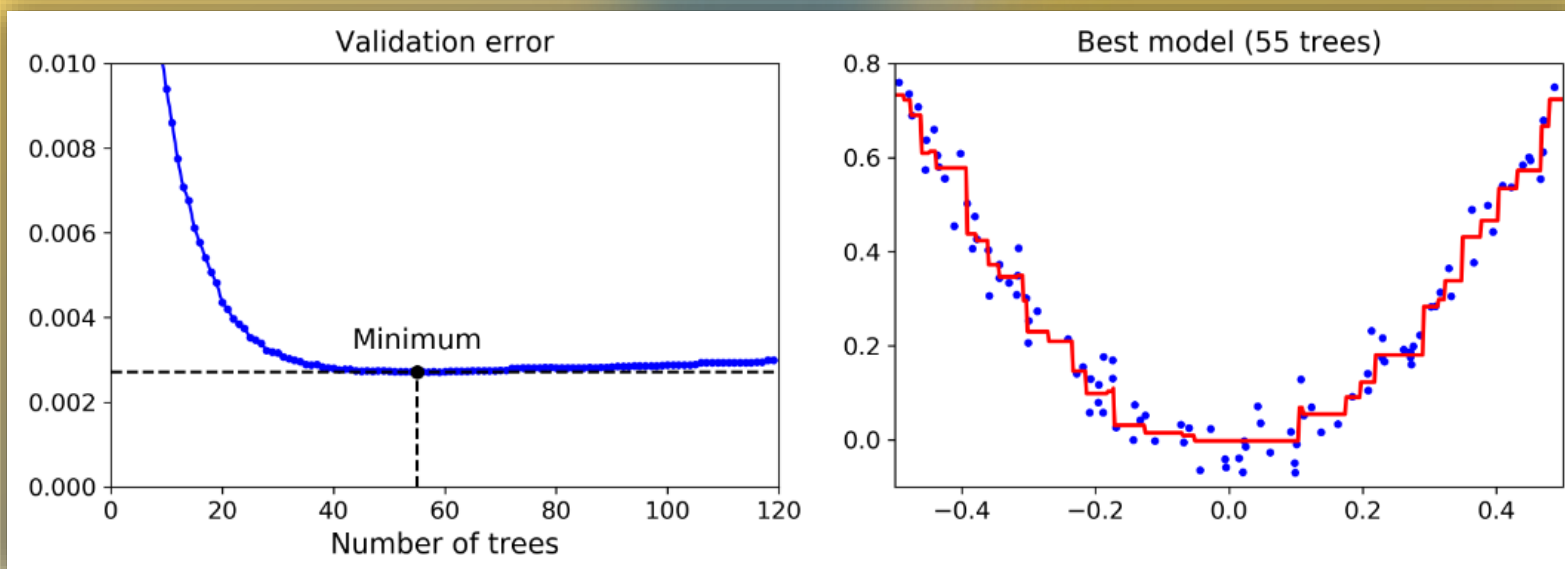
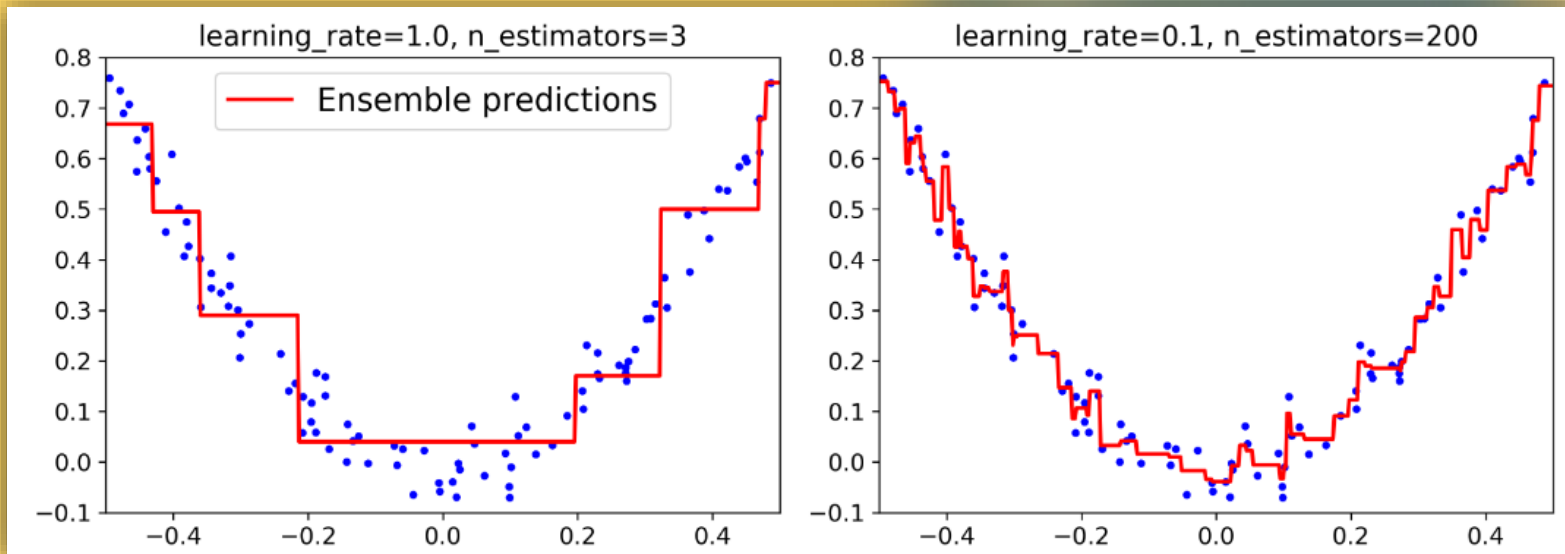
Turbózás lépésenként

🐍 Három döntési fa
predikciója külön-külön
(zöld), illetve turbózással
(piros).

🐍 Figyeljük meg, ahogy
egyetlen létrehozott fa
hozzáad az előzőek
összessége által alkotott
modellhez!



Alultanulás és túltanulás



🐍 A turbózott regresszor fák is hajlamosak a túltanulásra. A feladatunk felismerni és kezelni a jelenséget: **hiperparaméter hangolás.**

🐍 Early stopping implementálása segíthet elkerülni a túltanulást.
←