Bölüm 1: Karar Kuramı

Algoritmik Oyun Kuramı

Beklenen fayda | Fayda ölçeği | Risk tutumları | Belirsizlik ölçütleri | Bayesçi karar | Karar ağaçları | Bilginin değeri

Sercan KÜLCÜ · Giresun Üniversitesi · Bilgisayar Mühendisliği

© Sercan Külcü. Tüm hakları saklıdır. · Sunumun etkileşimli HTML sürümü: https://sercankulcu.github.io/files/agt/slides/Bolum_01_Karar_Kurami.html

BEKLENEN FAYDA

Oyunla Başlangıç: Bayesçi Karar Laboratuvarı

Oyunun kuralı

Gizli durumun ön olasılığını bilir, gürültülü bir sinyal görür ve iki eylemden birini seçersiniz; sonra durum açılır ve tablodaki kazancı alırsınız.

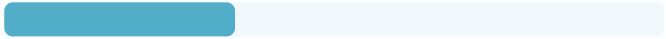
Merak sorusu

- %90 doğru bir tahmin “yağmur” dediyse, sizce yağmur olasılığı kaç?
- Kararınızdan sonra kötü bir sonuç çıktıysa karar da kötü müydü?

Bayesçi Karar Laboratuvarı →

	Yağmur	Kuru
Açık hava	−6	8
Kapalı	4	3

Ön olasılık: P(yağmur) = 0,35



Sinyal: hava tahmini, doğruluk %90 (Seviye 1)

Tahmin: “yağmur”

BEKLENEN FAYDA

Öğrenme Çıktıları

Beklenen fayda ve risk

Bir piyangonun beklenen değerini ve faydasını hesaplamak; kesinlik eşdeğeri ve risk primiyle risk tutumunu yorumlamak; olasılıklar bilinmezken maksimin, Laplace ve Savage ölçütlerini uygulamak.

Fayda ölçeği

vNM aksiyomlarını, Allais ve Ellsberg seçimlerini açıklamak; fayda sayılarının hangi dönüşümlerde kararı koruduğunu göstermek.

Bayesçi karar

Ön olasılığı test sonucuyla güncellemek; taban oranı ihmalini sayısal örnekte göstermek; beklenen kaybı en küçük eylemi seçmek.

Karar ağaçları

Geriye doğru çıkarımla en iyi stratejiyi bulmak; mükemmel ve örnek bilginin değerini hesaplamak.

BEKLENEN FAYDA

Karar Alma: Üç Tür

Karar alma, seçenekler arasından sonuçlarını değerlendirerek **seçim yapma** sürecidir. Kararı etkileyen şeye göre üç tür:

Tür	Sonucu belirleyen	Araç	Örnek
Bireysel, risk altında	Doğa / şans (olasılıklar bilinir ya da tahmin edilir)	Beklenen fayda, Bayes, karar ağacı	Yatırım, sigorta, tıbbi test
Stratejik	Diğer rasyonel aktörlerin seçimleri	En iyi tepki, Nash dengesi	Fiyat rekabeti, pazarlık, spor
Kolektif	Grubun toplu seçimi ve etkileşimi	Oylama, ortak mallar	Politika, şirket stratejisi

Bireysel kararda karar verici yalnızca doğaya (şansa) karşı seçim yapar. Beklenen fayda, Bayes kuralı ve karar ağacı, stratejik ve kolektif kararların da yapı taşlarıdır.

BEKLENEN FAYDA

Rasyonel Karar: Beklenen Değer mi, Beklenen Fayda mı?

- **Piyango** (lottery): olasılıklı sonuçlar kümesi. A = %50 olasılıkla 900 TL, %50 olasılıkla 0 TL.
- **Beklenen değer**: $E[X] = \Sigma\, p_i\, x_i$. $E[A] = ½\cdot900 + ½\cdot0 =$ **450** TL > kesin B = 400 TL.
- **Beklenen fayda** (expected utility; von Neumann–Morgenstern): $E[u] = \Sigma\, p_i\, u(x_i)$. Rasyonel karar verici beklenen *faydayı* maksimize eder.
- $u(x) = \sqrt{x}$ ile: $E[u(A)] = ½\cdot30 + ½\cdot0 =$ **15**; $u(B) = \sqrt{400} =$ **20** ⇒ B seçilir.



Kırmızı: her ölçütün seçtiği. Beklenen değer A'yı, beklenen fayda (√x) B'yi seçer.

Beklenen parasal değer ile beklenen fayda **aynı kararı vermeyebilir**; aradaki farkı fayda fonksiyonunun eğriliği, yani risk tutumu belirler.

St. Petersburg paradoksu: beklenen değeri sonsuz bir oyuna kimse büyük para ödemez; Daniel Bernoulli (1738) bu yüzden faydayı önerdi (benzer fikir: Cramer, 1728).

Beklenen Faydanın Temeli ve Allais Paradoksu

- **von Neumann–Morgenstern**: tercihler dört aksiyomu sağlıyorsa bir u vardır ve karar verici beklenen faydayı maksimize eder. A ≻ B: A, B'ye tercih edilir.
- **Tamlık**: her iki seçenek karşılaştırılabilir. **Geçişlilik**: A ≻ B ve B ≻ C ise A ≻ C. **Süreklilik**: arada kalan bir seçenek, iki ucun bir karışımına denk getirilebilir.
- **Bağımsızlık**: iki piyangoya aynı ortak parçayı (aynı olasılıkla) eklemek sıralamayı değiştirmez.
- **Allais (1953)**: çoğu kişi 1. seçimde A'yı, 2. seçimde D'yi seçer.

	Seçenek	1 milyon	5 milyon	0
1. seçim	A	%100	—	—
	B	%89	%10	%1
2. seçim	C	%11	—	%89
	D	—	%10	%90

A ve B'deki ortak %89 "1 milyon" parçası %89 "0" ile değiştirilince C ve D elde edilir.

Her u için E[u(A)] − E[u(B)] = 0,11·u(1) − 0,10·u(5) − 0,01·u(0) = E[u(C)] − E[u(D)]. Yani A ≻ B ise C ≻ D olmalı; "A ve D" seçimi **hiçbir** fayda fonksiyonuyla açıklanamaz. Kesinliğe verilen fazla ağırlık (kesinlik etkisi), insanların gerçek seçimlerini açıklamaya çalışan beklenti kuramının (prospect theory) çıkış noktalarından biridir.

Fayda Sayıları Neyi Ölçer? Ölçeğin Tekliği

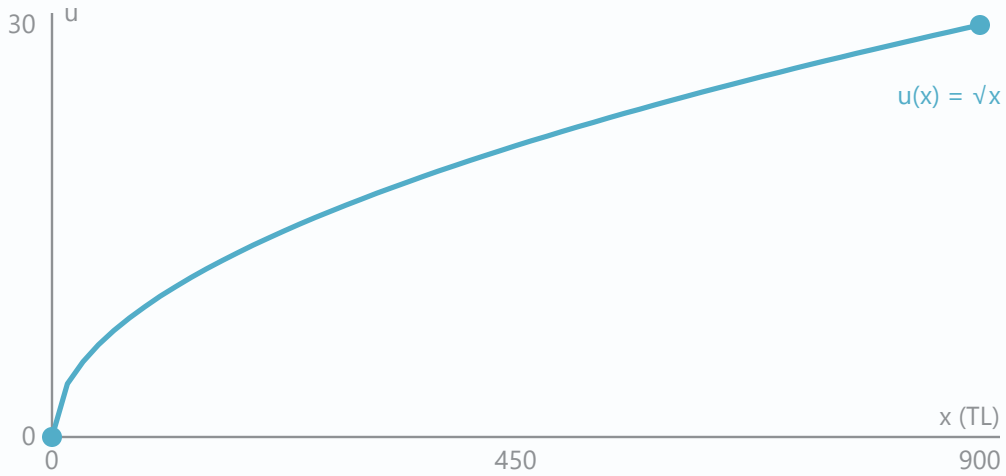
- Fayda bir termometre gibidir; sıfır noktası ve birimi keyfîdir. Celsius ile Fahrenheit aynı sıcaklık sırasını verir.
- **Teklik** (vNM): $v = a \cdot u + b$ $(a > 0)$ biçimindeki dönüşüme *afin dönüşüm* denir: ölçeği büyütür ve kaydırır, ama hiçbir piyango kararını değiştirmez. Başka biçimdeki (afin olmayan) dönüşümler kararı değiştirebilir.
- **Oyun kuramı bağı**: bir oyuncunun bütün kazançlarını 2 ile çarpıp 5 eklemek onun hiçbir kararını değiştirmez.

Fayda	piyango A	kesin B	seçim
$u = \sqrt{x}$	15	20	B
$v = 10\sqrt{x} + 5$	155	205	B
$w = (\sqrt{x})^2 = x$	450	400	A

A: ½ olasılıkla 900 TL, ½ olasılıkla 0 TL; B: kesin 400 TL. Afin v kararı korur; artan ama afin olmayan w kararı değiştirir.

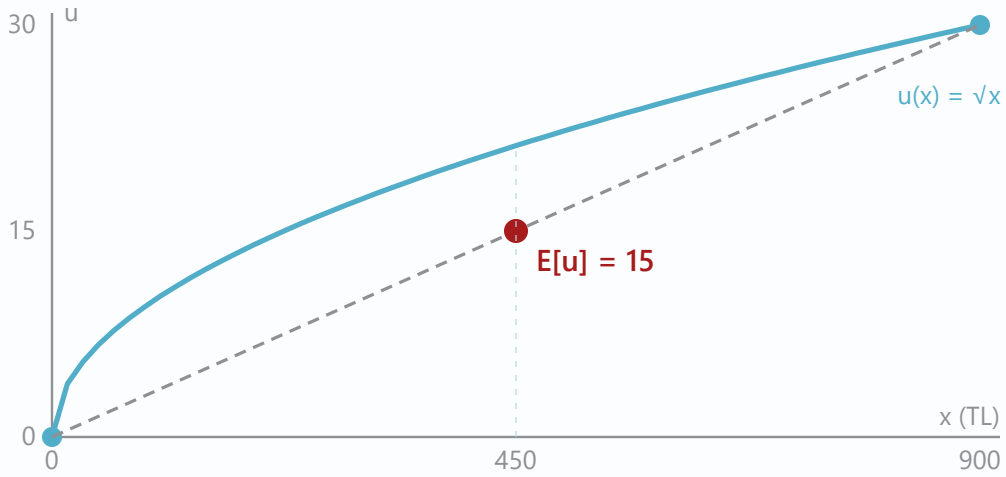
Sık hata: "u(B) − u(A) = 5 birim mutluluk" ya da "Ayşe'nin faydası 20, Bora'nınki 15; Ayşe daha mutlu" demek. Farkın büyüklüğü ve kişiler arası karşılaştırma ölçek seçimine bağlıdır; anlamlı olan, aynı kişinin seçenekleri arasındaki sıralamadır.

Kesinlik Eşdeğeri ve Risk Primi Adım Adım



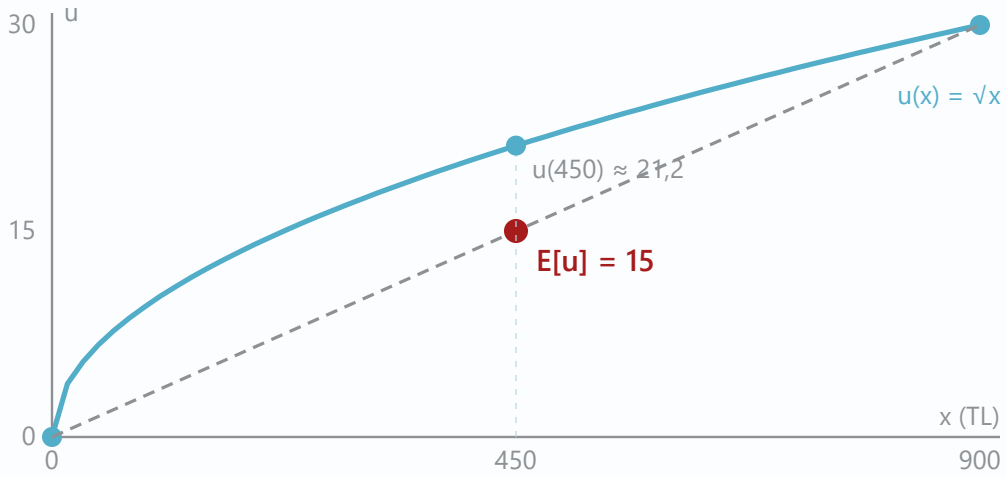
İçbükey fayda u(x) = √x: her ek lira bir öncekinden daha az fayda katar (azalan marjinal fayda). Uçlar: u(0) = 0, u(900) = 30.

Kesinlik Eşdeğeri ve Risk Primi Adım Adım



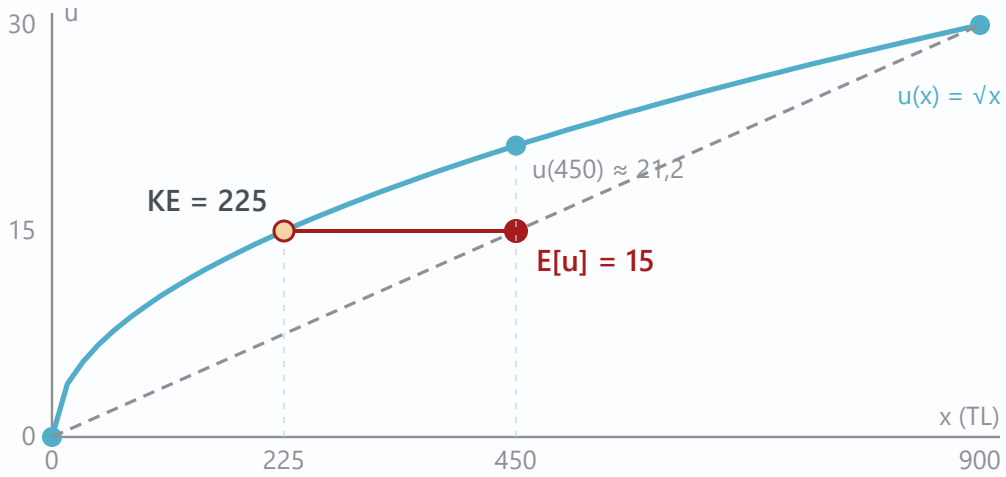
Kiriş: eğrinin iki ucunu birleştiren doğru. Piyango A'nın beklenen faydası bu kirişin ortasıdır: E[u] = ½·0 + ½·30 = **15** (x = 450 hizasında).

Kesinlik Eşdeğeri ve Risk Primi Adım Adım



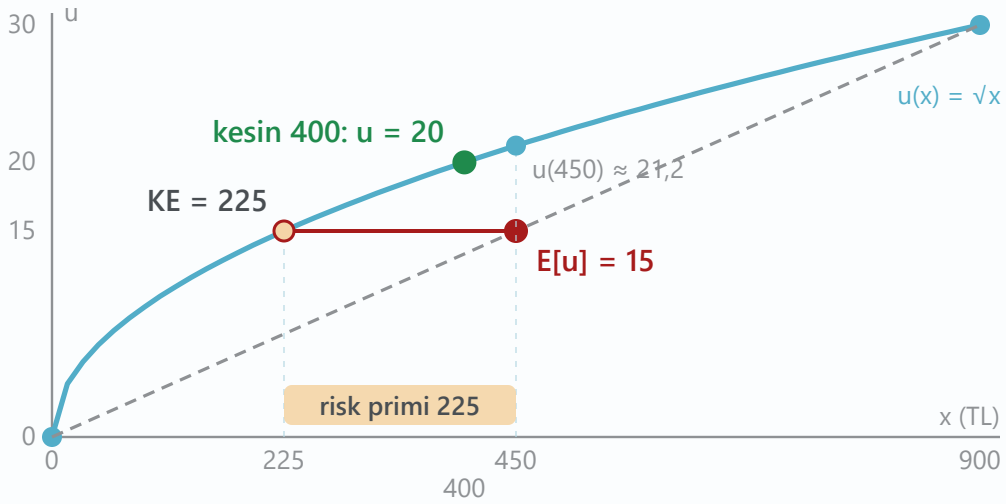
Aynı 450 TL kesin verilse fayda u(450) ≈ 21,2 > 15. İçbükey eğride kiriş eğrinin altında kalır, bu yüzden E[u(X)] < u(E[X]) (Jensen eşitsizliği).

Kesinlik Eşdeğeri ve Risk Primi Adım Adım



Kesinlik eşdeğeri (KE): faydası 15 olan kesin tutar. √x = 15 ⇒ **KE = 225 TL**. Bu kişi piyangoyu 225 TL'ye satmaya razıdır.

Kesinlik Eşdeğeri ve Risk Primi Adım Adım



Risk primi = E[X] − KE = 450 − 225 = **225 TL**. Kesin 400 TL'nin faydası 20 > 15 ⇒ riskten kaçınan kişi **B**'yi seçer.

RİSK TUTUMLARI

Üç Risk Tutumu: Aynı Piyango, Farklı Kararlar

Tutum	Fayda	E[u(A)]	u(400)	Seçim	KE (TL)	Risk primi
Riskten kaçınan	u(x) = √x (içbükey)	15	20	B	225	225
Risk nötr	u(x) = x (doğrusal)	450	400	A	450	0
Risk seven	u(x) = x^2/900 (dışbükey)	450	177,8	A	636,4	−186,4

İçbükey → kaçınan

KE < E[X], prim > 0: riski üstlenmek için ödül ister. İnsanların çoğu, büyük tutarlarda.

Doğrusal → nötr

KE = E[X]: yalnızca beklenen değere bakar. Büyük şirketler, küçük tutarlar.

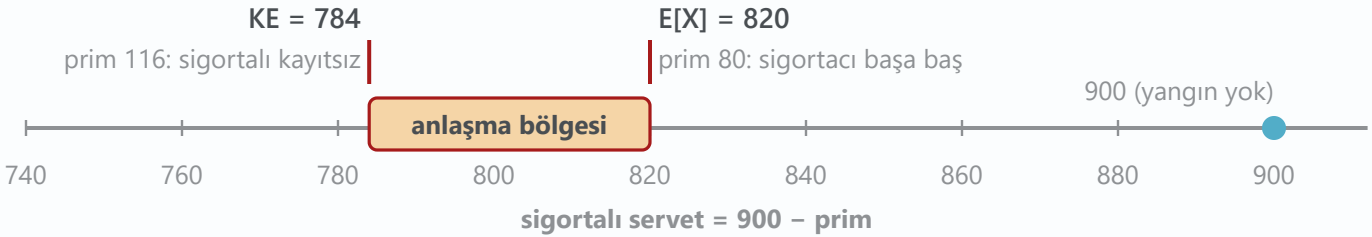
Dışbükey → seven

KE > E[X], prim < 0: risk için para öder. Piyango bileti alan kişi.

Piyango A: %50 olasılıkla 900 TL, %50 olasılıkla 0 TL (E[X] = 450). B: kesin 400 TL. KE = u^{-1}(E[u(A)]): u^{-1}, u'yu geri çözmektir; √x için $u^{-1}(15) = 15^2 = 225$. Risk primi = E[X] − KE.

Uygulama: Sigorta Neden Var?

- Servet 900 (bin TL); %10 olasılıkla 800'lük kayıp (yangın). Fayda $u(x) = \sqrt{x}$.
- Sigortasız: $E[u] = 0{,}9 \cdot \sqrt{900} + 0{,}1 \cdot \sqrt{100} = 27 + 1 = \mathbf{28}$.
- Kesinlik eşdeğeri: $28^2 = \mathbf{784}$. Kişi riskli 900 yerine kesin 784'e razı.
- En fazla ödenecek prim = 900 − 784 = **116**; beklenen kayıp yalnızca 0,1·800 = 80.



Sigortalı: prim ≤ 116 ödeyip riskten kurtulmak faydasını artırır. Sigortacı: çok sayıda bağımsız poliçeyle risk nötre yakındır; 80'den fazla her prim kârdır.

80 ile 116 arasındaki her prim **iki taraf için de kazançlıdır**: risk, ona en kolay katlanabilen tarafa aktarılır.

RİSK TUTUMLARI

Bilinen Risk mi, Bilinmeyen Olasılık mı? Ellsberg Torbası

- Risk altında olasılıklar bilinir (zar, rulet); belirsizlik altında bilinmez. İnsanlar ikisine aynı davranmaz.
- **Torba**: 90 top; 30'u kırmızı, kalan 60'ı siyah ya da sarı, ama kaçının siyah olduğu bilinmiyor. Kazandıran renk çekilirse 100 TL.
- **Tipik tercihler**: 1. bahiste "kırmızı", 2. bahiste "siyah ya da sarı"; ikisinde de kazanma olasılığı *bilinen* seçenek.

Bahis	Seçenek	kazandıran top
1	kırmızı	30 (bilinen)
	siyah	s (0–60, bilinmiyor)
2	kırmızı ya da sarı	30 + (60 − s)
	siyah ya da sarı	60 (bilinen)

s: torbadaki siyah top sayısı. Turuncu: çoğunluğun seçimi.

1. tercih 30 > s, 2. tercih 60 > 90 − s, yani s > 30 demektir: iki tercihi birlikte açıklayan **hiçbir olasılık yoktur** (Ellsberg, 1961). Olasılığı bilinmeyen seçenekten uzak durmaya **belirsizlikten kaçınma** (ambiguity aversion) denir; olasılıklar hiç bilinmediğinde en kötü durumu en iyi yapan seçimi aramak (maksimin) bu tutumun uç biçimidir.

Olasılıklar Bilinmezse: Belirsizlik Altında Karar

eylem \ durum	İyi	Orta	Kötü
Büyük yatırım	200	50	−120
Küçük yatırım	90	40	−20
Yatırım yok	0	0	0

Olasılıklar **bilinmiyor** (belirsizlik). Satırlar eylemler, sütunlar durumlar; her ölçüt farklı bir "karamsarlık" derecesi taşır.

Adım 1 / 9

```
1  def olcutler(U):
▶      A, m = range(len(U)), len(U[0])
3      mm = max(A, key=lambda i: min(U[i]))      # maximin
4      mx = max(A, key=lambda i: max(U[i]))      # maximax
5      lp = max(A, key=lambda i: sum(U[i]) / m)  # Laplace
6      enb = [max(r[j] for r in U) for j in range(m)]
7      pis = [max(enb[j] - r[j] for j in range(m))
8             for r in U]            # en büyük pişmanlık
9      sv = pis.index(min(pis))                  # Savage
10     return mm, mx, lp, sv
```

DEĞİŞKENLER

`U = 3×3 tablo` `m = 3`

Olasılıklar Bilinmezse: Belirsizlik Altında Karar

eylem \ durum	İyi	Orta	Kötü	min
Büyük yatırım	200	50	−120	−120
Küçük yatırım	90	40	−20	−20
Yatırım yok	0	0	0	0

Maksimin (maximin; Wald, en kötüyü düşün): satır minimumları −120, −20, 0 → en büyüğü 0: **Yatırım yok**.

Adım 2 / 9

```
1  def olcutler(U):
2      A, m = range(len(U)), len(U[0])
▶      mm = max(A, key=lambda i: min(U[i]))      # maximin
4      mx = max(A, key=lambda i: max(U[i]))      # maximax
5      lp = max(A, key=lambda i: sum(U[i]) / m)  # Laplace
6      enb = [max(r[j] for r in U) for j in range(m)]
7      pis = [max(enb[j] - r[j] for j in range(m))
8             for r in U]            # en büyük pişmanlık
9      sv = pis.index(min(pis))                  # Savage
10     return mm, mx, lp, sv
```

DEĞİŞKENLER

`U = 3×3 tablo` `mm = 2 (Yok)`

Olasılıklar Bilinmezse: Belirsizlik Altında Karar

eylem \ durum	İyi	Orta	Kötü	min	max
Büyük yatırım	200	50	−120	−120	200
Küçük yatırım	90	40	−20	−20	90
Yatırım yok	0	0	0	0	0

Maksimaks (maximax; iyimser): satır maksimumları 200, 90, 0 → **Büyük yatırım**.

Adım 3 / 9

```
1  def olcutler(U):
2      A, m = range(len(U)), len(U[0])
3      mm = max(A, key=lambda i: min(U[i]))      # maximin
▶      mx = max(A, key=lambda i: max(U[i]))      # maximax
5      lp = max(A, key=lambda i: sum(U[i]) / m)  # Laplace
6      enb = [max(r[j] for r in U) for j in range(m)]
7      pis = [max(enb[j] - r[j] for j in range(m))
8             for r in U]            # en büyük pişmanlık
9      sv = pis.index(min(pis))                  # Savage
10     return mm, mx, lp, sv
```

DEĞİŞKENLER

`U = 3×3 tablo` `mm = 2` `mx = 0 (Büyük)`

Olasılıklar Bilinmezse: Belirsizlik Altında Karar

eylem \ durum	İyi	Orta	Kötü	min	max	ort.
Büyük yatırım	200	50	−120	−120	200	43,3
Küçük yatırım	90	40	−20	−20	90	36,7
Yatırım yok	0	0	0	0	0	0

Laplace (durumlar eşit olası say): ortalamalar 43,3, 36,7, 0,0 → **Büyük yatırım**.

Adım 4 / 9

```
1  def olcutler(U):
2      A, m = range(len(U)), len(U[0])
3      mm = max(A, key=lambda i: min(U[i]))      # maximin
4      mx = max(A, key=lambda i: max(U[i]))      # maximax
▶      lp = max(A, key=lambda i: sum(U[i]) / m)  # Laplace
6      enb = [max(r[j] for r in U) for j in range(m)]
7      pis = [max(enb[j] - r[j] for j in range(m))
8             for r in U]            # en büyük pişmanlık
9      sv = pis.index(min(pis))                  # Savage
10     return mm, mx, lp, sv
```

DEĞİŞKENLER

`U = 3×3 tablo` `mm = 2` `mx = 0` `lp = 0 (Büyük)`

Olasılıklar Bilinmezse: Belirsizlik Altında Karar

eylem \ durum	İyi	Orta	Kötü	min	max	ort.
Büyük yatırım	200	50	−120	−120	200	43,3
Küçük yatırım	90	40	−20	−20	90	36,7
Yatırım yok	0	0	0	0	0	0
sütun en iyisi:	200	50	0			

Her durumda ulaşılabilecek en iyi sonuç: enb = [200, 50, 0]. Pişmanlık = enb − elde edilen.

Adım 5 / 9

```
1  def olcutler(U):
2      A, m = range(len(U)), len(U[0])
3      mm = max(A, key=lambda i: min(U[i]))      # maximin
4      mx = max(A, key=lambda i: max(U[i]))      # maximax
5      lp = max(A, key=lambda i: sum(U[i]) / m)  # Laplace
▶      enb = [max(r[j] for r in U) for j in range(m)]
7      pis = [max(enb[j] - r[j] for j in range(m))
8             for r in U]            # en büyük pişmanlık
9      sv = pis.index(min(pis))                  # Savage
10     return mm, mx, lp, sv
```

DEĞİŞKENLER

`U = 3×3 tablo` `mm = 2` `mx = 0` `lp = 0` `enb = [200, 50, 0]`

Olasılıklar Bilinmezse: Belirsizlik Altında Karar

eylem \ durum	İyi	Orta	Kötü	min	max	ort.	maks. pişm.
Büyük yatırım	200	50	−120	−120	200	43,3	120
Küçük yatırım	90	40	−20	−20	90	36,7	110
Yatırım yok	0	0	0	0	0	0	200
sütun en iyisi:	200	50	0				

Satır başına en büyük pişmanlık: Büyük (kötü durumda 0 − (−120)) = 120, Küçük (iyide 200 − 90) = 110, Yok (iyide 200) = 200.

Adım 6 / 9

```python
def olcutler(U):
    A, m = range(len(U)), len(U[0])
    mm = max(A, key=lambda i: min(U[i]))      # maximin
    mx = max(A, key=lambda i: max(U[i]))      # maximax
    lp = max(A, key=lambda i: sum(U[i]) / m)  # Laplace
    enb = [max(r[j] for r in U) for j in range(m)]
    pis = [max(enb[j] - r[j] for j in range(m))
           for r in U]            # en büyük pişmanlık
    sv = pis.index(min(pis))                  # Savage
    return mm, mx, lp, sv
```

DEĞİŞKENLER

`U = 3×3 tablo` `mm = 2` `mx = 0` `lp = 0` `enb = [200, 50, 0]` `pis = [120, 110, 200]`

Olasılıklar Bilinmezse: Belirsizlik Altında Karar

eylem \ durum	İyi	Orta	Kötü	min	max	ort.	maks. pişm.
Büyük yatırım	200	50	−120	−120	200	43,3	120
Küçük yatırım	90	40	−20	−20	90	36,7	110
Yatırım yok	0	0	0	0	0	0	200
sütun en iyisi:	200	50	0				

Savage (minimaks pişmanlık): en küçük maksimum pişmanlık 110 → **Küçük yatırım**.

Adım 7 / 9

```
1  def olcutler(U):
2      A, m = range(len(U)), len(U[0])
3      mm = max(A, key=lambda i: min(U[i]))      # maximin
4      mx = max(A, key=lambda i: max(U[i]))      # maximax
5      lp = max(A, key=lambda i: sum(U[i]) / m)  # Laplace
6      enb = [max(r[j] for r in U) for j in range(m)]
7      pis = [max(enb[j] - r[j] for j in range(m))
8             for r in U]            # en büyük pişmanlık
▶      sv = pis.index(min(pis))                # Savage
10     return mm, mx, lp, sv
```

DEĞİŞKENLER

`U = 3×3 tablo` `mm = 2` `mx = 0` `lp = 0` `pis = [120, 110, 200]` `sv = 1 (Küçük)`

Olasılıklar Bilinmezse: Belirsizlik Altında Karar

eylem \ durum	İyi	Orta	Kötü	min	max	ort.	maks. pişm.
Büyük yatırım	200	50	−120	−120	200	43,3	120
Küçük yatırım	90	40	−20	−20	90	36,7	110
Yatırım yok	0	0	0	0	0	0	200
sütun en iyisi:	200	50	0				

return (2, 0, 0, 1). Dönen indisler (0: Büyük, 1: Küçük, 2: Yok): maksimin → Yok, maksimaks ve Laplace → Büyük, Savage → Küçük. Aynı tabloda üç farklı karar.

Adım 8 / 9

```
1 def olcutler(U):
2     A, m = range(len(U)), len(U[0])
3     mm = max(A, key=lambda i: min(U[i]))      # maximin
4     mx = max(A, key=lambda i: max(U[i]))      # maximax
5     lp = max(A, key=lambda i: sum(U[i]) / m)  # Laplace
6     enb = [max(r[j] for r in U) for j in range(m)]
7     pis = [max(enb[j] - r[j] for j in range(m))
8            for r in U]            # en büyük pişmanlık
9     sv = pis.index(min(pis))                  # Savage
▶     return mm, mx, lp, sv
```

DEĞİŞKENLER

`sonuç = (2, 0, 0, 1)`

Olasılıklar Bilinmezse: Belirsizlik Altında Karar

eylem \ durum	İyi	Orta	Kötü	min	max	ort.	maks. pişm.
Büyük yatırım	200	50	−120	−120	200	43,3	120
Küçük yatırım	90	40	−20	−20	90	36,7	110
Yatırım yok	0	0	0	0	0	0	200
sütun en iyisi:	200	50	0				

Kodda olmayan bir ölçüt daha: **Hurwicz** iki uç arasında durur: α·(satır maks.) + (1 − α)·(satır min.); iyimserlik katsayısı α = 0,5 ile 40, 35, 0 → Büyük. Olasılıklar tahmin edilebiliyorsa beklenen faydaya dönülür; edilemiyorsa ölçüt seçimi karar vericinin tutumunu yansıtır.

Adım 9 / 9

```
1  def olcutler(U):
2      A, m = range(len(U)), len(U[0])
3      mm = max(A, key=lambda i: min(U[i]))      # maximin
4      mx = max(A, key=lambda i: max(U[i]))      # maximax
5      lp = max(A, key=lambda i: sum(U[i]) / m)  # Laplace
6      enb = [max(r[j] for r in U) for j in range(m)]
7      pis = [max(enb[j] - r[j] for j in range(m))
8             for r in U]            # en büyük pişmanlık
9      sv = pis.index(min(pis))                 # Savage
▶      return mm, mx, lp, sv
```

DEĞİŞKENLER

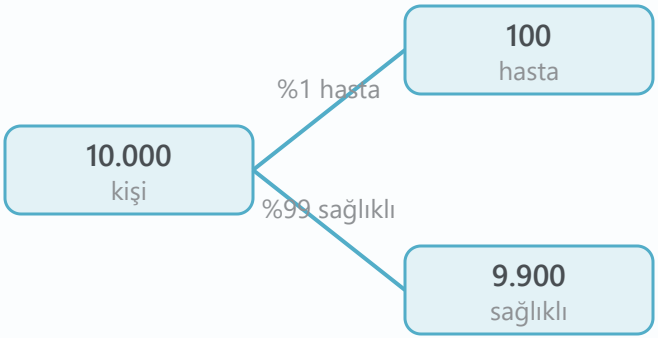
`sonuç = (2, 0, 0, 1)`

Tıbbi Test: Pozitif Sonuç Ne Anlama Gelir?

10.000
kişi

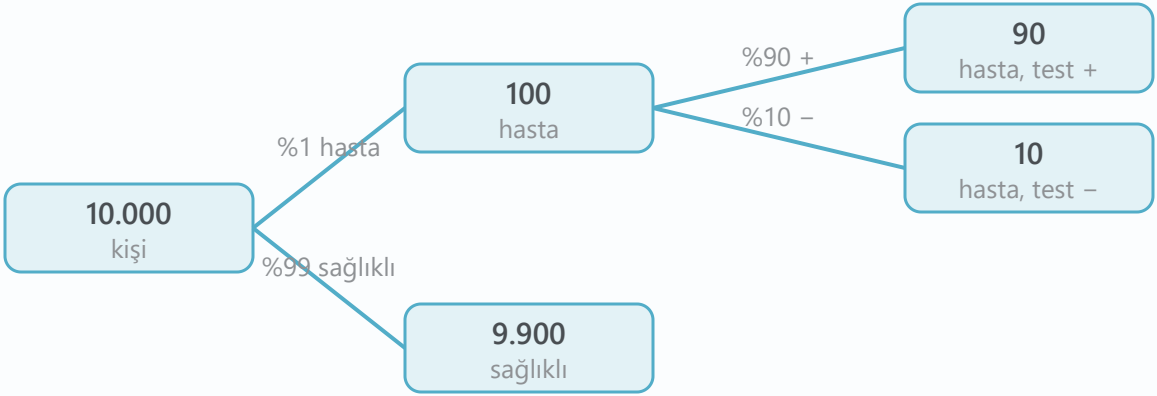
10.000 kişilik bir topluluk düşünelim. Olasılıklar yerine **doğal sıklıklar** (kişi sayıları) kullanmak Bayes hesabını sezgisel kılar.

Tıbbi Test: Pozitif Sonuç Ne Anlama Gelir?



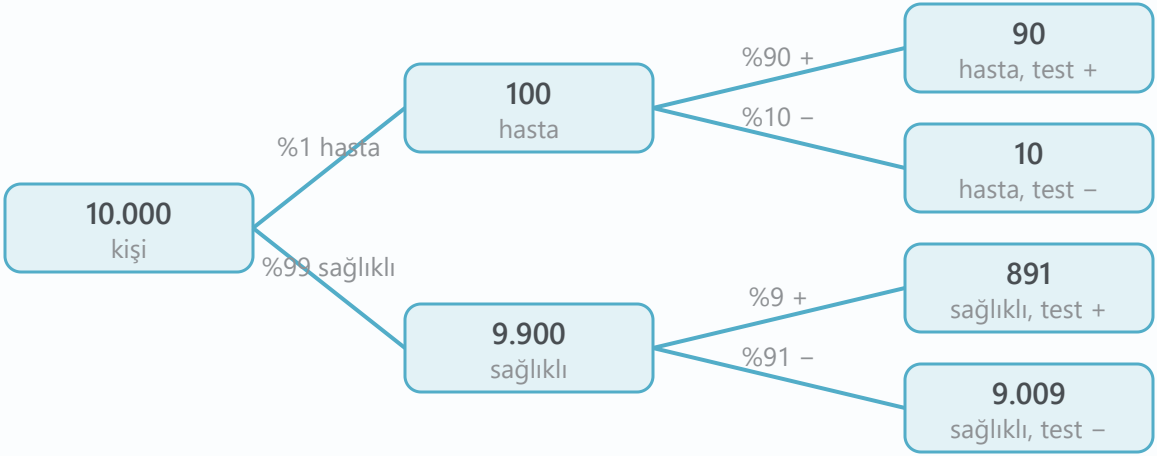
Ön olasılık (prior): yaygınlık %1 ⇒ 100 hasta, 9.900 sağlıklı.

Tıbbi Test: Pozitif Sonuç Ne Anlama Gelir?



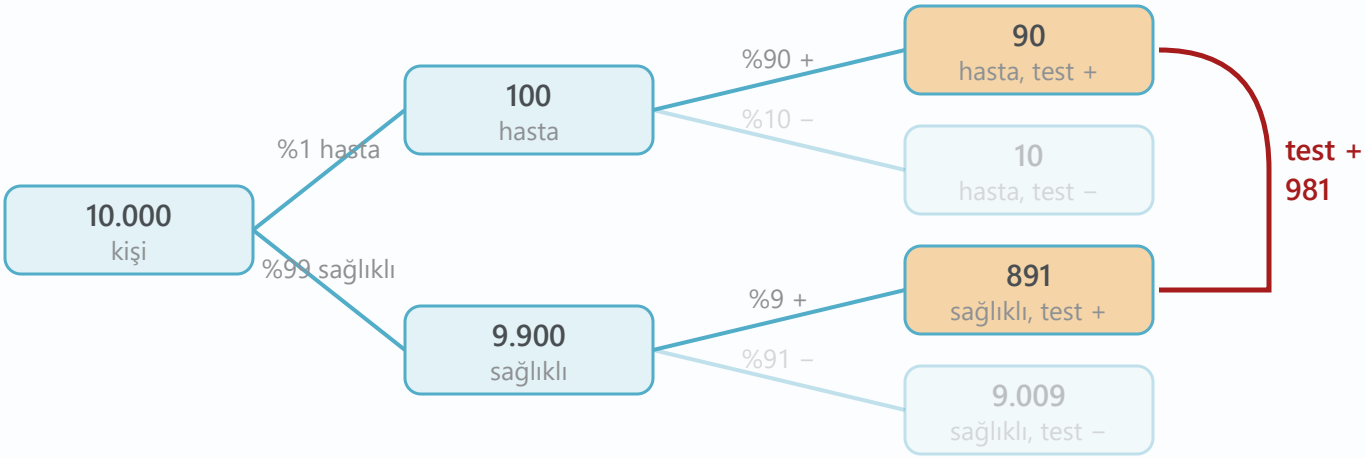
Duyarlılık (sensitivity) %90: 100 hastanın 90'ı pozitif, 10'u negatif çıkar.

Tıbbi Test: Pozitif Sonuç Ne Anlama Gelir?



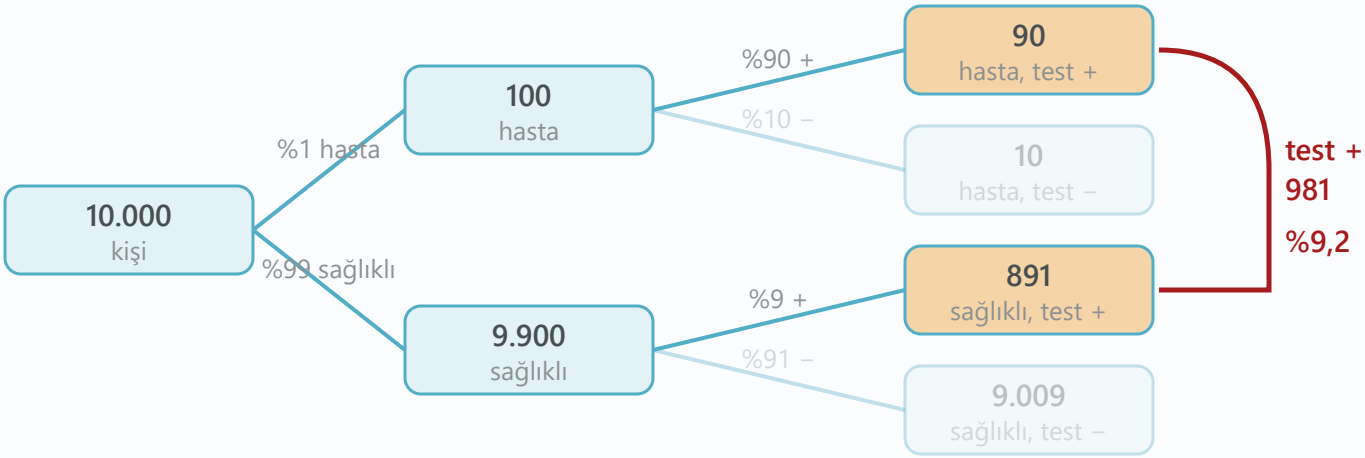
Yanlış pozitif oranı %9: 9.900 sağlıklının 891'i de pozitif çıkar! Sağlıklılar çok kalabalık olduğu için bu sayı büyüktür.

Tıbbi Test: Pozitif Sonuç Ne Anlama Gelir?



Test pozitif çıkan herkes: 90 + 891 = **981** kişi. Bunlardan yalnızca 90'ı gerçekten hasta.

Tıbbi Test: Pozitif Sonuç Ne Anlama Gelir?



Son olasılık: P(hasta | +) = 90 / 981 = **%9,2**. Test %90 duyarlı olsa da pozitif sonuç çoğunlukla yanlış alarmdır (taban oranı ihmali).

Kod İzleme: `bayes(p, duyarlilik, yp, testler)`



Çağrı: `bayes(0.01, 0.9, 0.09, ["+", "+", "−"])`. p ön olasılıkla (%1) başlar; üç test sonucu sırayla işlenecek.

Adım 1 / 14

```
▶ def bayes(p, duyarlilik, yp, testler):
2   for sonuc in testler:
3     if sonuc == "+":
4       pay = duyarlilik * p
5       payda = pay + yp * (1 - p)
6     else:
7       pay = (1 - duyarlilik) * p
8       payda = pay + (1 - yp) * (1 - p)
9     p = pay / payda
10    print(sonuc, round(p, 4))
11  return p
```

DEĞİŞKENLER

`p = 0.0100` `duyarlilik = 0.9` `yp = 0.09` `testler = [+, +, −]`

ÇIKTI

Kod İzleme: `bayes(p, duyarlilik, yp, testler)`



1. test sonucu: **+**. Şu anki inanç p = 0.0100 bu testin ön olasılığı olur.

Adım 2 / 14

```
1  def bayes(p, duyarlilik, yp, testler):
▶    for sonuc in testler:
3      if sonuc == "+":
4        pay = duyarlilik * p
5        payda = pay + yp * (1 - p)
6      else:
7        pay = (1 - duyarlilik) * p
8        payda = pay + (1 - yp) * (1 - p)
9      p = pay / payda
10     print(sonuc, round(p, 4))
11   return p
```

DEĞİŞKENLER

`p = 0.0100` `duyarlilik = 0.9` `yp = 0.09` `sonuc = +`

ÇIKTI

Kod İzleme: `bayes(p, duyarlilik, yp, testler)`



sonuc == "+" ? → **Doğru**: pozitif sonucun olabilirlikleri kullanılır. Olabilirlik: hastalık durumu bilindiğinde bu test sonucunun olasılığı, ör. P(+ | hasta) = 0,9.

Adım 3 / 14

```
1  def bayes(p, duyarlilik, yp, testler):
2    for sonuc in testler:
▶      if sonuc == "+":
4        pay = duyarlilik * p
5        payda = pay + yp * (1 - p)
6      else:
7        pay = (1 - duyarlilik) * p
8        payda = pay + (1 - yp) * (1 - p)
9      p = pay / payda
10     print(sonuc, round(p, 4))
11   return p
```

DEĞİŞKENLER

`p = 0.0100` `duyarlilik = 0.9` `yp = 0.09` `sonuc = +`

ÇIKTI

Kod İzleme: `bayes(p, duyarlilik, yp, testler)`



pay = P(+|hasta)·p = 0.9 × 0.0100 = 0.0090; payda = pay + P(+|sağlıklı)·(1 − p) = 0.0981 (toplam pozitif olasılığı).

Adım 4 / 14

```
1  def bayes(p, duyarlilik, yp, testler):
2    for sonuc in testler:
3      if sonuc == "+":
▶        pay = duyarlilik * p
▶        payda = pay + yp * (1 - p)
6      else:
7        pay = (1 - duyarlilik) * p
8        payda = pay + (1 - yp) * (1 - p)
9      p = pay / payda
10     print(sonuc, round(p, 4))
11   return p
```

DEĞİŞKENLER

`p = 0.0100` `duyarlilik = 0.9` `yp = 0.09` `sonuc = +` `pay = 0.0090` `payda = 0.0981`

ÇIKTI

Kod İzleme: `bayes(p, duyarlilik, yp, testler)`



p = pay / payda = **0.0917**; print ile yazdırılır. Pozitif teste rağmen hastalık olasılığı hâlâ %10'un altında.

Adım 5 / 14

```
1  def bayes(p, duyarlilik, yp, testler):
2    for sonuc in testler:
3      if sonuc == "+":
4        pay = duyarlilik * p
5        payda = pay + yp * (1 - p)
6      else:
7        pay = (1 - duyarlilik) * p
8        payda = pay + (1 - yp) * (1 - p)
▶      p = pay / payda
▶      print(sonuc, round(p, 4))
11   return p
```

DEĞİŞKENLER

`p = 0.0917` `duyarlilik = 0.9` `yp = 0.09` `sonuc = +` `pay = 0.0090` `payda = 0.0981`

ÇIKTI

```
+ 0.0917
```

Kod İzleme: `bayes(p, duyarlilik, yp, testler)`



2. test sonucu: **+**. Şu anki inanç p = 0.0917 bu testin ön olasılığı olur.

Adım 6 / 14

```
1  def bayes(p, duyarlilik, yp, testler):
▶    for sonuc in testler:
3      if sonuc == "+":
4        pay = duyarlilik * p
5        payda = pay + yp * (1 - p)
6      else:
7        pay = (1 - duyarlilik) * p
8        payda = pay + (1 - yp) * (1 - p)
9      p = pay / payda
10     print(sonuc, round(p, 4))
11   return p
```

DEĞİŞKENLER

`p = 0.0917` `duyarlilik = 0.9` `yp = 0.09` `sonuc = +`

ÇIKTI

```
+ 0.0917
```

Kod İzleme: `bayes(p, duyarlilik, yp, testler)`



sonuc == "+" ? → **Doğru**: pozitif sonucun olabilirlikleri kullanılır.

Adım 7 / 14

```
1  def bayes(p, duyarlilik, yp, testler):
2    for sonuc in testler:
▶      if sonuc == "+":
4        pay = duyarlilik * p
5        payda = pay + yp * (1 - p)
6      else:
7        pay = (1 - duyarlilik) * p
8        payda = pay + (1 - yp) * (1 - p)
9      p = pay / payda
10     print(sonuc, round(p, 4))
11   return p
```

DEĞİŞKENLER

`p = 0.0917` `duyarlilik = 0.9` `yp = 0.09` `sonuc = +`

ÇIKTI

```
+ 0.0917
```

Kod İzleme: `bayes(p, duyarlilik, yp, testler)`



pay = P(+|hasta)·p = 0.9 × 0.0917 = 0.0826; payda = pay + P(+|sağlıklı)·(1 − p) = 0.1643 (toplam pozitif olasılığı).

Adım 8 / 14

```
1  def bayes(p, duyarlilik, yp, testler):
2    for sonuc in testler:
3      if sonuc == "+":
▶        pay = duyarlilik * p
▶        payda = pay + yp * (1 - p)
6      else:
7        pay = (1 - duyarlilik) * p
8        payda = pay + (1 - yp) * (1 - p)
9      p = pay / payda
10     print(sonuc, round(p, 4))
11   return p
```

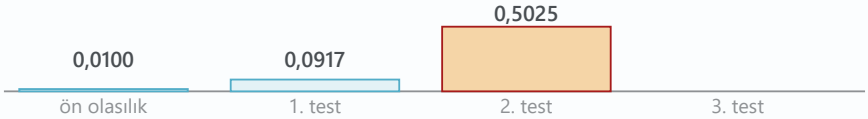
DEĞİŞKENLER

`p = 0.0917` `duyarlilik = 0.9` `yp = 0.09` `sonuc = +` `pay = 0.0826` `payda = 0.1643`

ÇIKTI

```
+ 0.0917
```

Kod İzleme: `bayes(p, duyarlilik, yp, testler)`



p = pay / payda = **0.5025**; print ile yazdırılır. İki bağımsız pozitif, olasılığı ~%50'ye taşıdı.

Adım 9 / 14

```
1  def bayes(p, duyarlilik, yp, testler):
2    for sonuc in testler:
3      if sonuc == "+":
4        pay = duyarlilik * p
5        payda = pay + yp * (1 - p)
6      else:
7        pay = (1 - duyarlilik) * p
8        payda = pay + (1 - yp) * (1 - p)
▶      p = pay / payda
▶      print(sonuc, round(p, 4))
11   return p
```

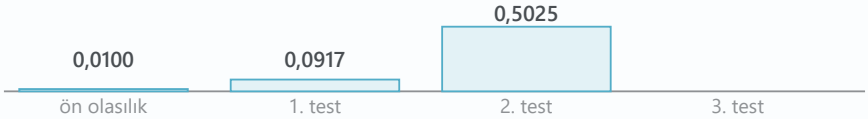
DEĞİŞKENLER

`p = 0.5025` `duyarlilik = 0.9` `yp = 0.09` `sonuc = +` `pay = 0.0826` `payda = 0.1643`

ÇIKTI

```
+ 0.0917
+ 0.5025
```

Kod İzleme: `bayes(p, duyarlilik, yp, testler)`



3. test sonucu: **-**. Şu anki inanç p = 0.5025 bu testin ön olasılığı olur.


```
1  def bayes(p, duyarlilik, yp, testler):
▶    for sonuc in testler:
3      if sonuc == "+":
4        pay = duyarlilik * p
5        payda = pay + yp * (1 - p)
6      else:
7        pay = (1 - duyarlilik) * p
8        payda = pay + (1 - yp) * (1 - p)
9      p = pay / payda
10     print(sonuc, round(p, 4))
11   return p
```

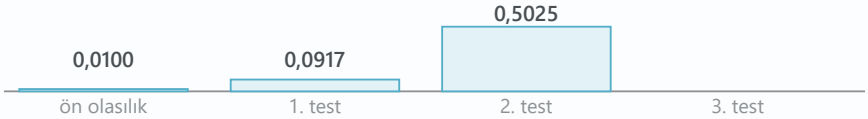
DEĞİŞKENLER

`p = 0.5025` `duyarlilik = 0.9` `yp = 0.09` `sonuc = -`

ÇIKTI

```
+ 0.0917
+ 0.5025
```

Kod İzleme: `bayes(p, duyarlilik, yp, testler)`



sonuc == "+" ? → **Yanlış**: negatif sonucun olabilirlikleri kullanılır.

Adım 11 / 14

```
1  def bayes(p, duyarlilik, yp, testler):
2    for sonuc in testler:
▶      if sonuc == "+":
4        pay = duyarlilik * p
5        payda = pay + yp * (1 - p)
6      else:
7        pay = (1 - duyarlilik) * p
8        payda = pay + (1 - yp) * (1 - p)
9      p = pay / payda
10     print(sonuc, round(p, 4))
11   return p
```

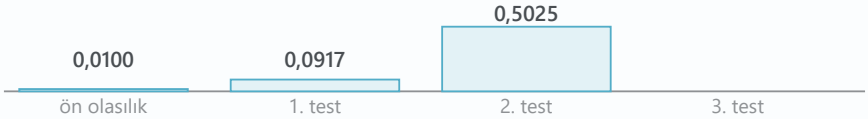
DEĞİŞKENLER

`p = 0.5025` `duyarlilik = 0.9` `yp = 0.09` `sonuc = -`

ÇIKTI

```
+ 0.0917
+ 0.5025
```

Kod İzleme: `bayes(p, duyarlilik, yp, testler)`



pay = P(−|hasta)·p = 0.1 × 0.5025 = 0.0503; payda = pay + P(−|sağlıklı)·(1 − p) = 0.91 × 0.4975 + pay = 0.5030.

Adım 12 / 14

```
1  def bayes(p, duyarlilik, yp, testler):
2    for sonuc in testler:
3      if sonuc == "+":
4        pay = duyarlilik * p
5        payda = pay + yp * (1 - p)
6      else:
▶        pay = (1 - duyarlilik) * p
▶        payda = pay + (1 - yp) * (1 - p)
9      p = pay / payda
10     print(sonuc, round(p, 4))
11   return p
```

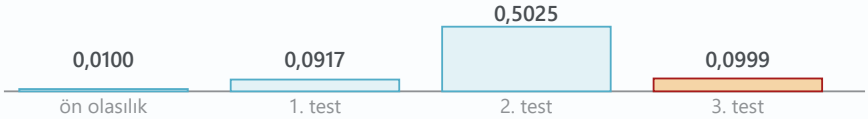
DEĞİŞKENLER

`p = 0.5025` `duyarlilik = 0.9` `yp = 0.09` `sonuc = -` `pay = 0.0503` `payda = 0.5030`

ÇIKTI

```
+ 0.0917
+ 0.5025
```

Kod İzleme: `bayes(p, duyarlilik, yp, testler)`



p = pay / payda = **0.0999**; print ile yazdırılır. Negatif sonuç inancı yeniden %10'a indirdi.

Adım 13 / 14

```
1  def bayes(p, duyarlilik, yp, testler):
2    for sonuc in testler:
3      if sonuc == "+":
4        pay = duyarlilik * p
5        payda = pay + yp * (1 - p)
6      else:
7        pay = (1 - duyarlilik) * p
8        payda = pay + (1 - yp) * (1 - p)
▶      p = pay / payda
▶      print(sonuc, round(p, 4))
11   return p
```

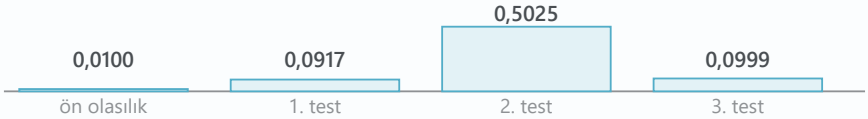
DEĞİŞKENLER

`p = 0.0999` `duyarlilik = 0.9` `yp = 0.09` `sonuc = -` `pay = 0.0503` `payda = 0.5030`

ÇIKTI

```
+ 0.0917
+ 0.5025
- 0.0999
```

Kod İzleme: `bayes(p, duyarlilik, yp, testler)`



Döngü bitti; fonksiyon p = 0.0999 döndürür. Her test, bir öncekinin son olasılığını ön olasılık olarak kullanır (ardışık güncelleme).

Adım 14 / 14

```
1  def bayes(p, duyarlilik, yp, testler):
2    for sonuc in testler:
3      if sonuc == "+":
4        pay = duyarlilik * p
5        payda = pay + yp * (1 - p)
6      else:
7        pay = (1 - duyarlilik) * p
8        payda = pay + (1 - yp) * (1 - p)
9      p = pay / payda
10     print(sonuc, round(p, 4))
▶    return p
```

DEĞİŞKENLER

`p = 0.0999` `sonuç = 0.0999`

ÇIKTI

```
+ 0.0917
+ 0.5025
- 0.0999
```

Bayesçi Karar Kuralı: Tedavi mi, Bekle mi?

- Ön olasılık kanıtla güncellenir; sonra her eylemin **beklenen kaybı** son olasılıkla hesaplanır. En düşük beklenen kayıplı eylem seçilir.
- Kayıplar: sağlıklıya gereksiz tedavi 30, hastayı tedavi etmemek 200.
- E[kayıp | tedavi] = (1 − p)·30; E[kayıp | bekle] = p·200. Tedavi daha az kayıplıysa seçilir: (1 − p)·30 < p·200 ⇒ 30 < 230p ⇒ p > 30/230 = **3/23 ≈ %13,0**.

Durum	p	tedavi	bekle	karar
1 pozitif	0,0917	27,2	18,3	bekle, yeniden test
2 pozitif	0,5025	14,9	100,5	tedavi

Beklenen kayıplar (turuncu = en küçük)

Karar yalnızca olasılığa değil **kayıpların oranına** da bağlıdır: hastalığı kaçırmak ne kadar pahalıysa eşik o kadar düşer.

Oyuna Dönüş: Bayesçi Karar Laboratuvarı

Kuramın cevabı: %82,9

P(yağmur | “yağmur”) = 0,9·0,35 / (0,9·0,35 + 0,1·0,65) = 0,315 / 0,38 ≈ 0,829.

Bilgi	P(yağmur)	BF(açık)	BF(kapalı)	Karar
yalnız ön olasılık	0,35	3,10	3,35	kapalı
“yağmur”	0,829	−3,61	3,83	kapalı
“kuru”	0,056	7,21	3,06	açık

BF: beklenen fayda. Açık hava ancak P(yağmur) < 1/3 iken seçilir. Seviye 3'te (doğruluk %78) sinyalin bilgi değeri 1,765'tir; bu değer sinyalin fiyatından (1) büyük olduğu için sinyali satın almaya değer.

Gerçekte: taban oranı ihmali

Klasik bir çalışmada hekim ve tıp öğrencilerine yaygınlığı 1/1000, yanlış pozitif oranı %5 olan bir test soruldu: 60 kişiden yalnızca 11'i doğru cevaba (yaklaşık %2) ulaştı, en sık verilen cevap %95'ti (Casscells, Schoenberger ve Graboys, 1978).

Oyundaki “EU-optimal” sayacı şanslı sonucu değil, eldeki bilgiyle beklenen faydayı maksimize eden kararı sayar: iyi karar kötü sonuç verebilir.

Bayesçi Oyunlara Köprü: Rakibin Tipini Güncellemek

- Eksik bilgili oyunda bilinmeyen “durum” rakibin *tipidir*. Tipe dair inanç ön olasılık, rakibin hamlesi kanıttır.
- **Örnek** (poker): rakip %20 olasılıkla blöfçü, %80 dürüst. Blöfçü elinden bağımsız %90 olasılıkla yükseltir. Dürüst oyuncu yalnızca güçlü eli varken yükseltir; güçlü el %30 olasılıkla geldiği için dürüst oyuncu %30 olasılıkla yükseltir.
- Rakip yükseltti: P(blöfçü | yükseltme) = 0,2·0,9 / (0,2·0,9 + 0,8·0,3) = 0,18 / 0,42 ≈ **0,429**.
- Görmek (call) blöfçüye karşı +30, dürüste karşı −30 getirsin: E[gör] = 0,429·30 − 0,571·30 ≈ **−4,3** < 0 ⇒ çekil. İkinci yükseltmede 0,429 ön olasılık olur ve son olasılık 0,692'ye çıkar.

Gözlem	P(blöfçü)	E[gör]	karar
yalnız ön olasılık	0,200	−18,0	çekil
1 yükseltme	0,429	−4,3	çekil
2 yükseltme	0,692	+11,5	gör

Görmek ancak P(blöfçü) > 0,5 iken kârlıdır (kazanç ve kayıp eşit büyüklükte).

Sık hata: P(yükseltme | blöfçü) = 0,9'u P(blöfçü | yükseltme) sanmak. Dürüst oyuncular daha kalabalık olduğu için yükseltmelerin çoğu (0,24 / 0,42 ≈ %57) onlardan gelir. Poker ve Hokm'da bahis ve koz seçimleri bu güncellemenin kanıtlarıdır; blöf, rakibin güncellemesini yönlendirme çabasıdır.

Karar Ağacı: Ürün Lansmanı ve Pazar Araştırması

- Bir firma yeni ürünü piyasaya **sürüp sürmemeye** karar verecek. Pazar iyi (olasılık 2/5) ise +500, kötü (3/5) ise −200 bin TL; sürmezse 0.
- Önce 20 bin TL'ye **pazar araştırması** yaptırabilir. Rapor kusurludur: P(olumlu | iyi) = 0,8, P(olumlu | kötü) = 0,1.
- Bayes ile: P(olumlu) = 0,4·0,8 + 0,6·0,1 = 0,38; P(iyi | olumlu) = 0,32/0,38 = 16/19; P(iyi | olumsuz) = 0,08/0,62 = 4/31.

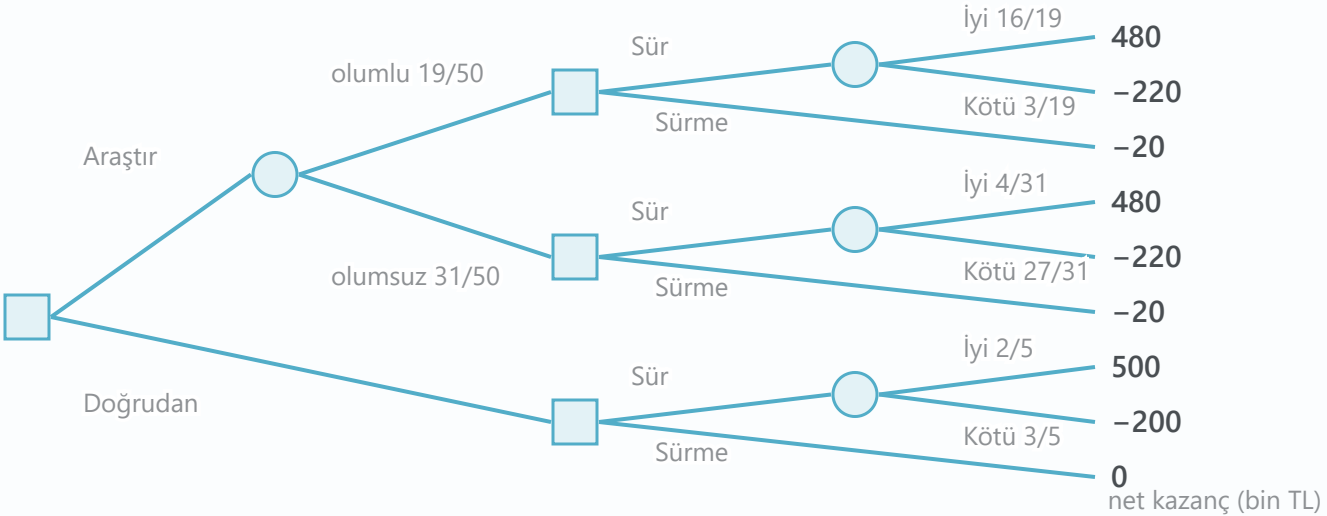
□ Karar düğümü
Karar verici seçer → dallardan en büyük değer.

○ Şans düğümü
Doğa seçer → olasılıklarla ağırlıklı ortalama.

Yaprak
Bir yolun net sonucu.

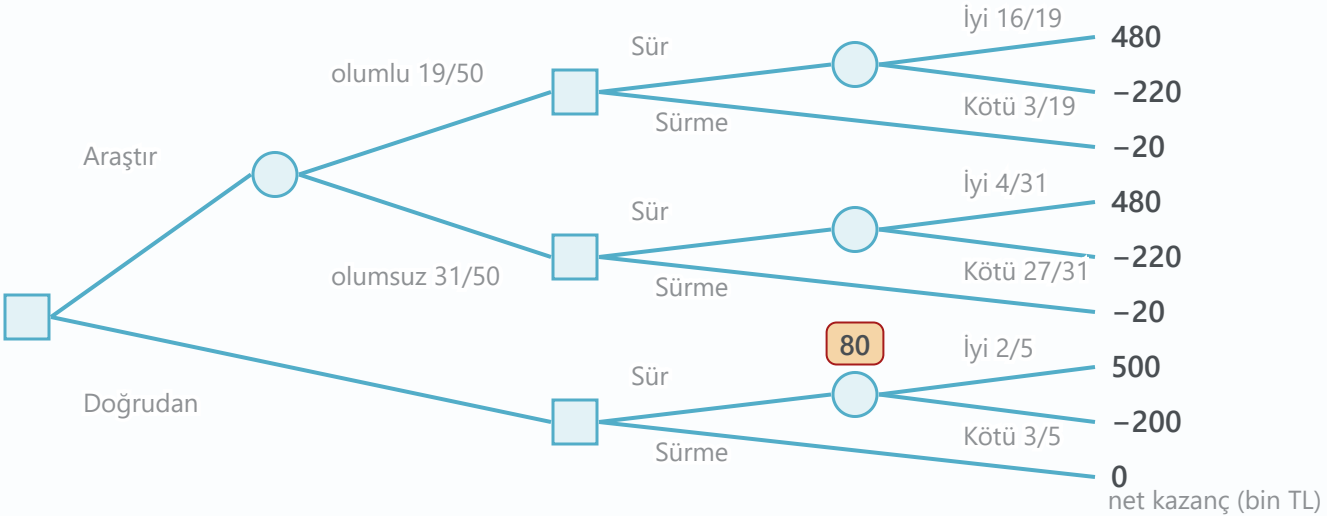
Karar ağacı, doğanın da hamle yaptığı (şans düğümlü) bir oyun ağacıdır; **geriye doğru çıkarımla** yapraktan köke çözülür.

Karar Ağacında Geriye Doğru Çıkarım



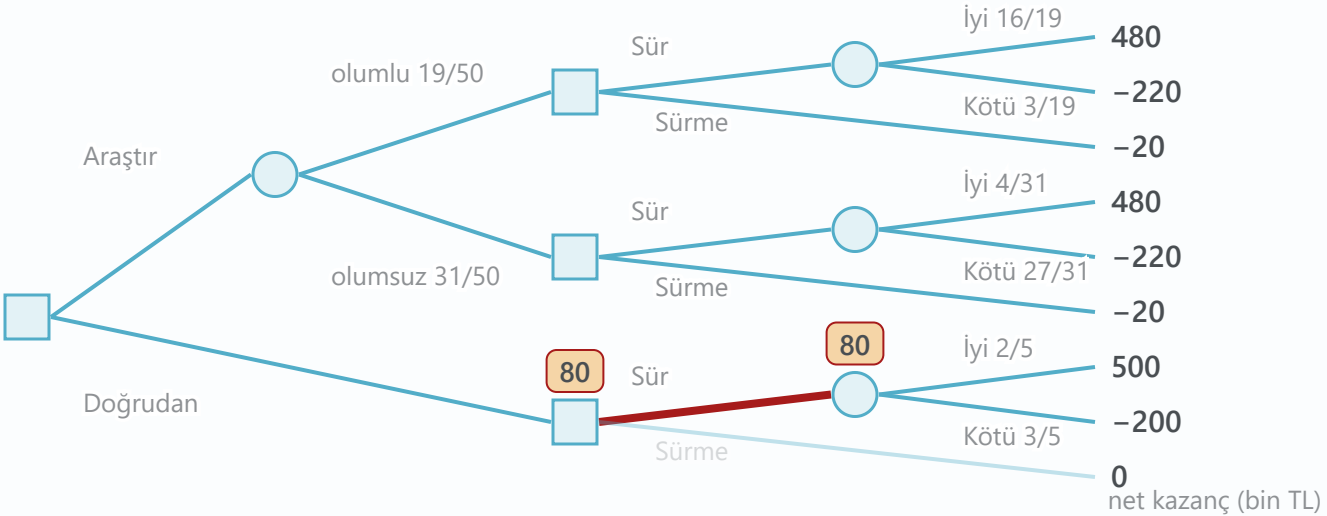
Kare = karar düğümü, daire = şans düğümü. Yapraklarda net kazanç (araştırma yapılırsa 20 düşülmüş). Kökteki iki dal: Araştır ve Doğrudan (araştırmadan karar ver). Geriye doğru çıkarım: şans düğümünde **beklenen değer**, karar düğümünde **en büyük** değer.

Karar Ağacında Geriye Doğru Çıkarım



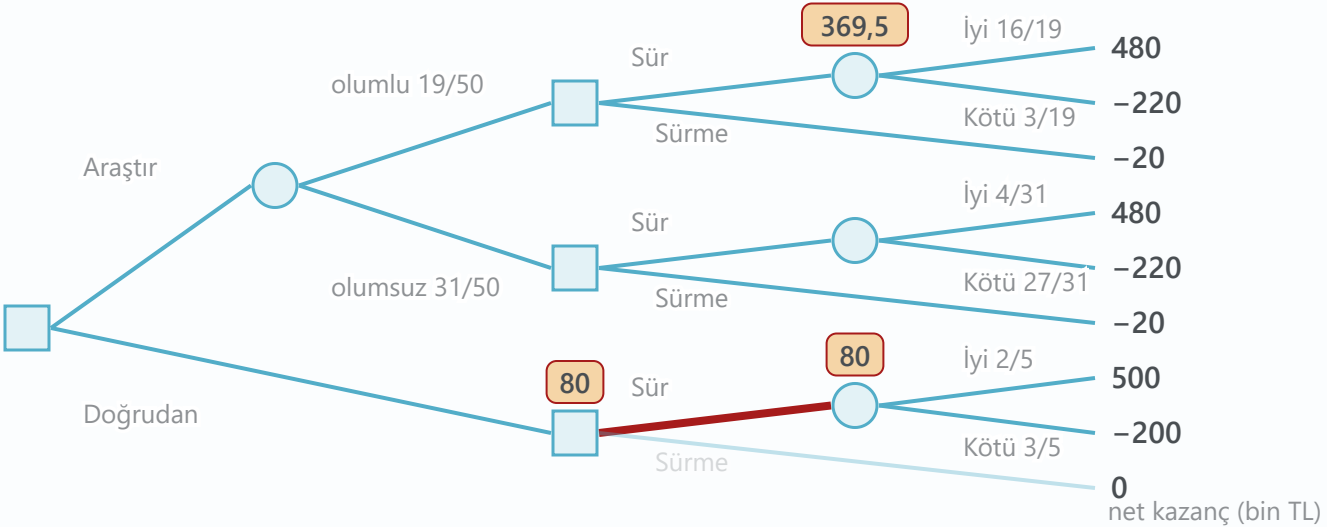
Araştırmasız sürme: 2/5·500 + 3/5·(−200) = 200 − 120 = **80**.

Karar Ağacında Geriye Doğru Çıkarım



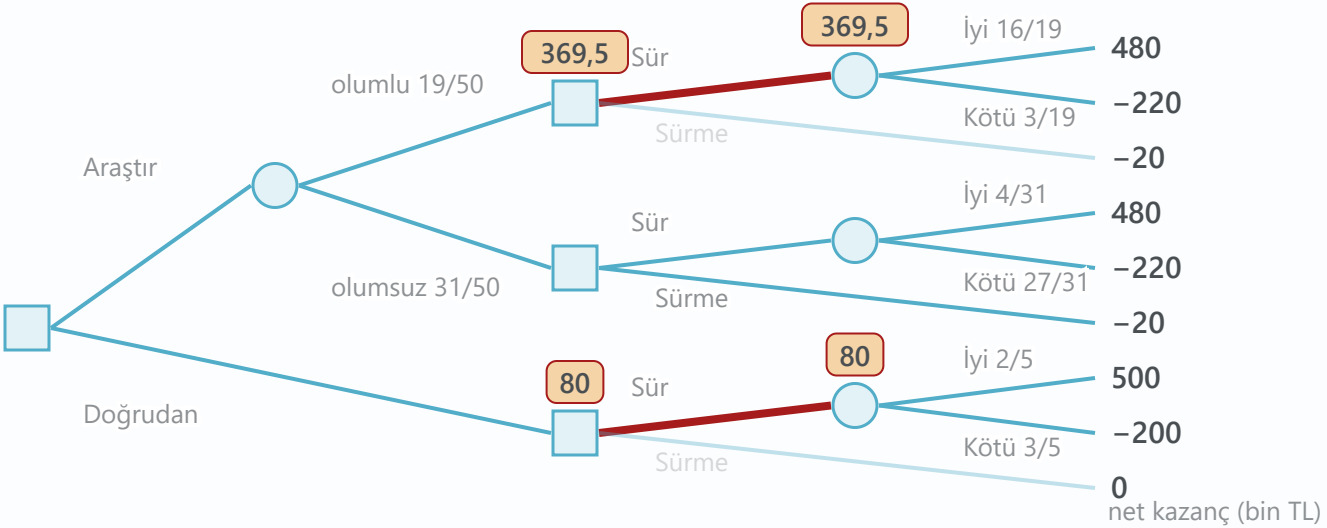
Karar: max(Sür 80, Sürme 0) = 80 ⇒ **Sür**. Sürme dalı elenir.

Karar Ağacında Geriye Doğru Çıkarım



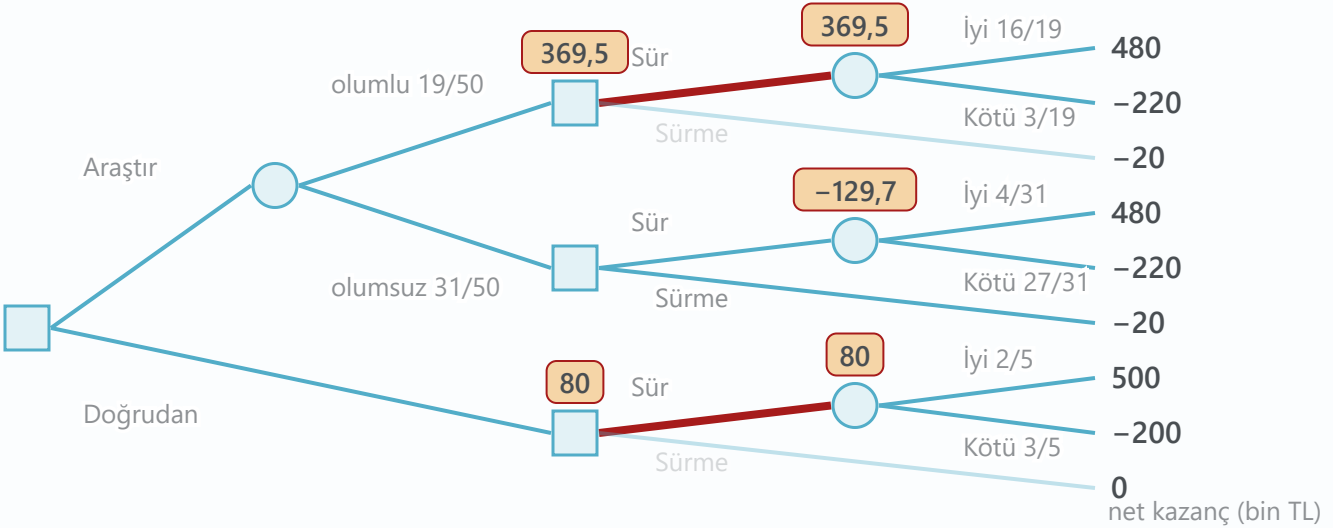
Olumlu raporda P(İyi | olumlu) = 0,32/0,38 = 16/19 (Bayes). Beklenen: 16/19·480 + 3/19·(−220) ≈ **369,5**.

Karar Ağacında Geriye Doğru Çıkarım



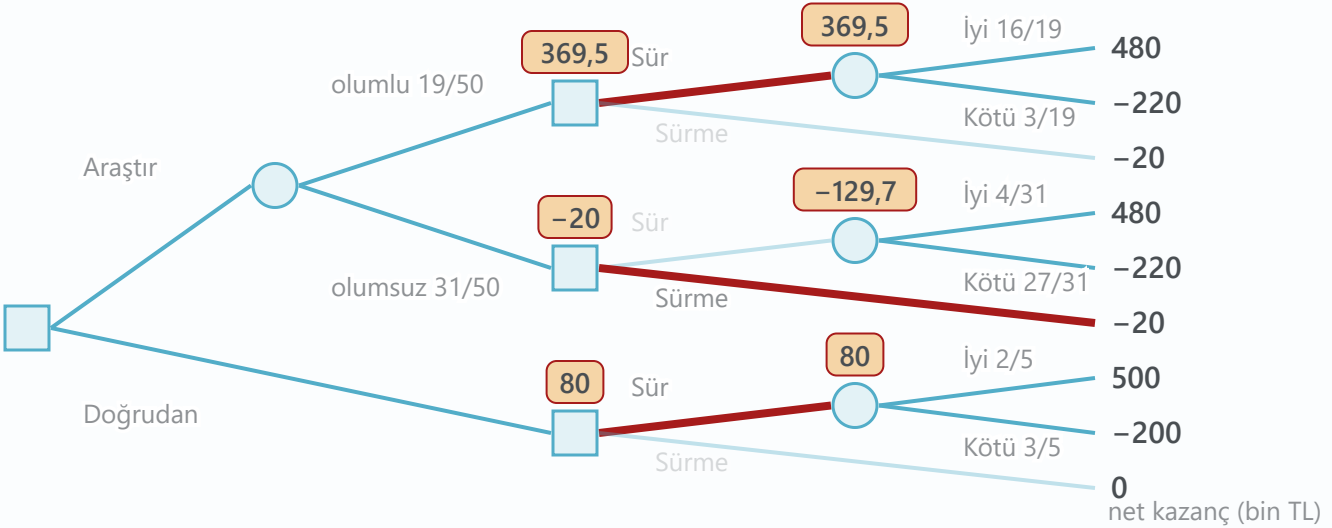
Karar: max(369,5, −20) ⇒ olumlu raporda **Sür**.

Karar Ağacında Geriye Doğru Çıkarım



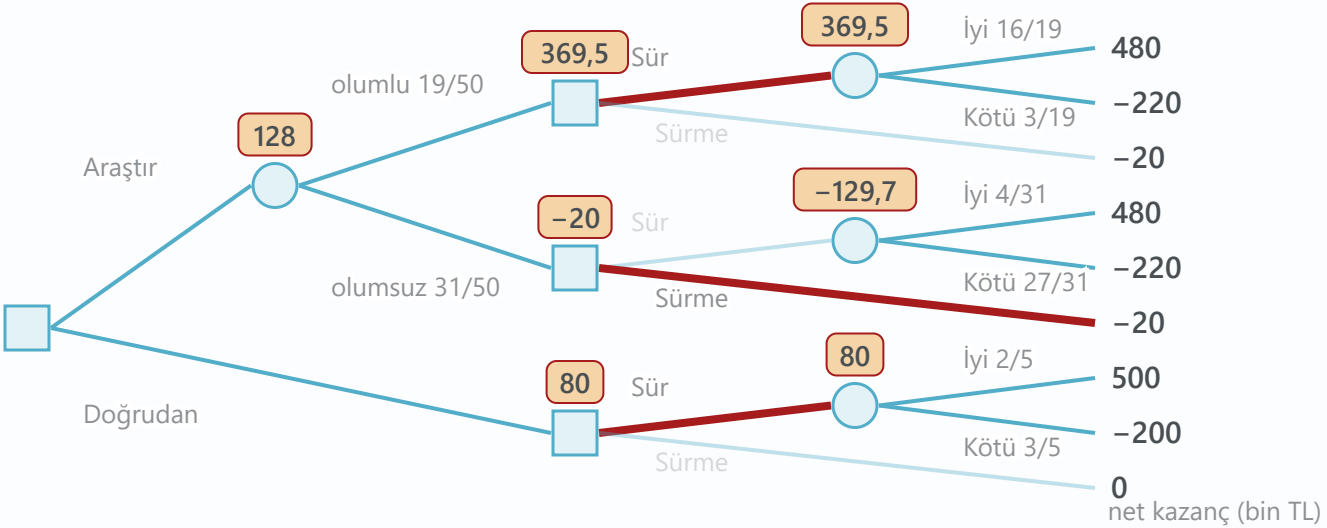
Olumsuz raporda P(İyi | olumsuz) = 0,08/0,62 = 4/31. Beklenen: 4/31·480 + 27/31·(−220) ≈ **−129,7**.

Karar Ağacında Geriye Doğru Çıkarım



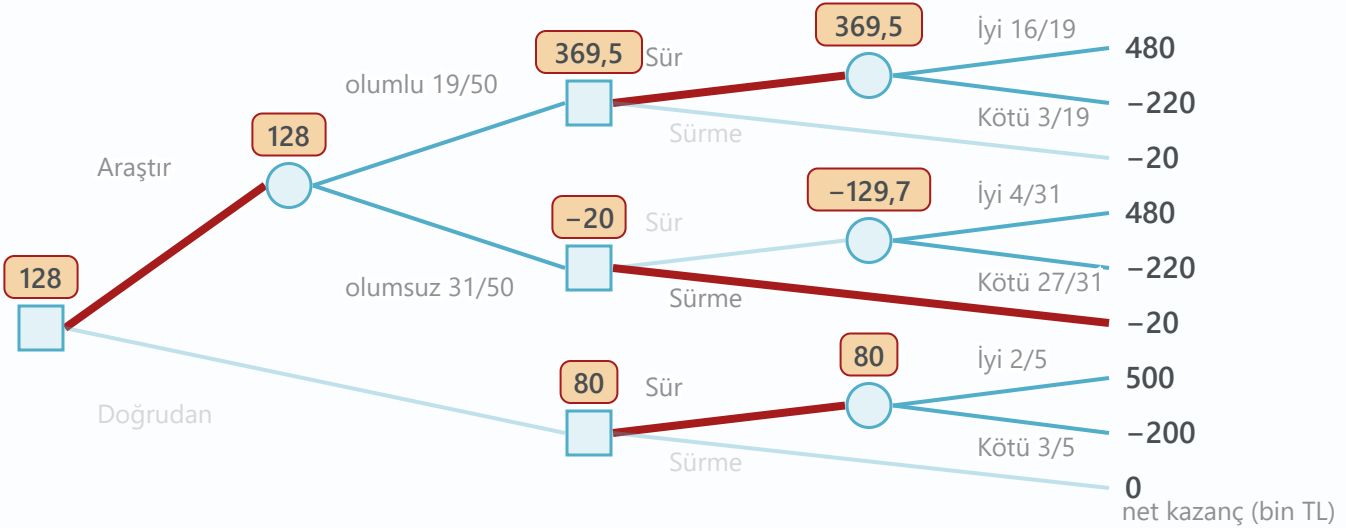
Karar: max(−129,7, −20) = −20 ⇒ olumsuz raporda **Sürme** (yalnızca araştırma maliyeti kaybedilir).

Karar Ağacında Geriye Doğru Çıkarım



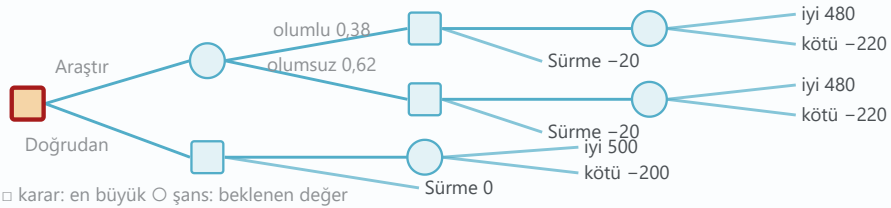
Rapor şans düğümü: 0,38·369,5 + 0,62·(−20) = **128**.

Karar Ağacında Geriye Doğru Çıkarım



Kök: max(Araştır 128, Doğrudan 80) ⇒ **Araştır**. Strateji: araştır; olumluysa sür, olumsuzsa sürme.

Kod İzleme: Özyinelemeli Geriye Doğru Çıkarım



`deger(kök)` çağrıldı. Ağaç iç içe demettir: `kök = ("K", [("Araştır", ("S", [(0.38, …), (0.62, …)])), ("Doğrudan", ("K", […]))])` ; "K" karar, "S" şans düğümü, yaprak bir sayı.

Adım 1 / 16

```
1 def deger(n):
2     if not isinstance(n, tuple):    # yaprak
3         return n
      tur, dallar = n
      if tur == "S":                  # şans düğümü
6         return sum(p * deger(c) for p, c in dallar)
      return max(deger(c) for ad, c in dallar)  # karar
```

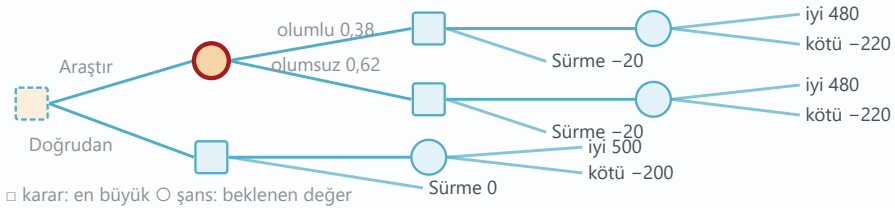
DEĞİŞKENLER

`n = kök`

ÇAĞRI YIĞINI

`deger(kök)`

Kod İzleme: Özyinelemeli Geriye Doğru Çıkarım



`deger(Araştır)` çağrıldı: şans düğümü; çocukları sırayla değerlendirilir.

Adım 2 / 16

```
1 def deger(n):
2     if not isinstance(n, tuple):    # yaprak
3         return n
▶     tur, dallar = n
▶     if tur == "S":                  # şans düğümü
6         return sum(p * deger(c) for p, c in dallar)
7     return max(deger(c) for ad, c in dallar)  # karar
```

DEĞİŞKENLER

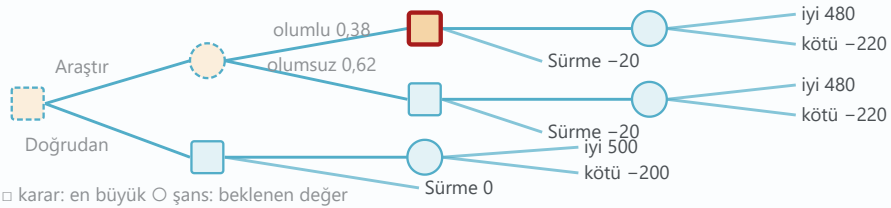
`n = Araştır`

ÇAĞRI YIĞINI

`deger(Araştır)`

`deger(kök)`

Kod İzleme: Özyinelemeli Geriye Doğru Çıkarım



`deger(olumlu)` çağrıldı: karar düğümü; çocukları sırayla değerlendirilir.

Adım 3 / 16

```
1 def deger(n):
2     if not isinstance(n, tuple):    # yaprak
3         return n
▶     tur, dallar = n
▶     if tur == "S":                  # şans düğümü
6         return sum(p * deger(c) for p, c in dallar)
▶     return max(deger(c) for ad, c in dallar)  # karar
```

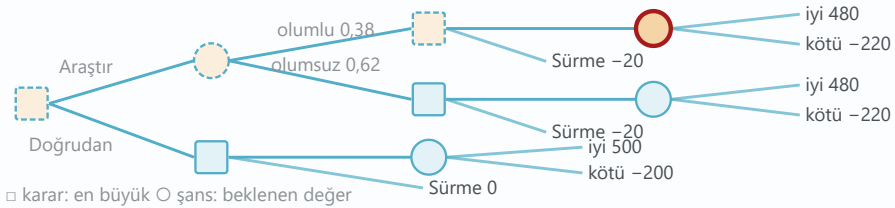
DEĞİŞKENLER

`n = olumlu`

ÇAĞRI YIĞINI

- **`deger(olumlu)`**
- `deger(Araştır)`
- `deger(kök)`

Kod İzleme: Özyinelemeli Geriye Doğru Çıkarım



`deger(olumlu·Sür)` çağrıldı: şans düğümü; çocukları sırayla değerlendirilir.

Adım 4 / 16

```
1 def deger(n):
2     if not isinstance(n, tuple):    # yaprak
3         return n
▶     tur, dallar = n
▶     if tur == "S":                  # şans düğümü
6         return sum(p * deger(c) for p, c in dallar)
7     return max(deger(c) for ad, c in dallar)  # karar
```

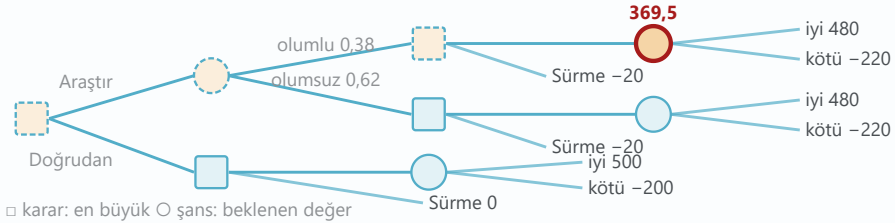
DEĞİŞKENLER

`n = olumlu·Sür`

ÇAĞRI YIĞINI

- `deger(olumlu·Sür)`
- `deger(olumlu)`
- `deger(Araştır)`
- `deger(kök)`

Kod İzleme: Özyinelemeli Geriye Doğru Çıkarım



olumlu·Sür (şans): 0,842·480 + 0,158·−220 = **369,5**.

Adım 5 / 16

```
1 def deger(n):
2     if not isinstance(n, tuple):    # yaprak
3         return n
4     tur, dallar = n
5     if tur == "S":                  # şans düğümü
▶         return sum(p * deger(c) for p, c in dallar)
7     return max(deger(c) for ad, c in dallar)  # karar
```

DEĞİŞKENLER

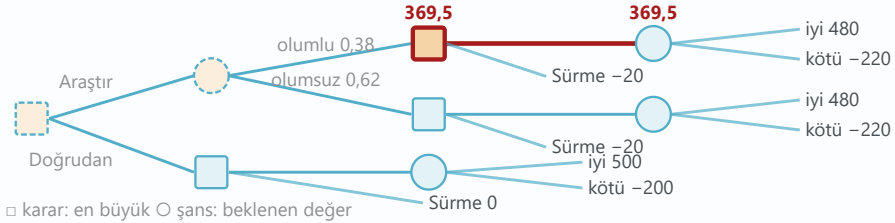
`n = olumlu·Sür`

`return = 369.47`

ÇAĞRI YIĞINI

- `deger(olumlu·Sür)`
- `deger(olumlu)`
- `deger(Araştır)`
- `deger(kök)`

Kod İzleme: Özyinelemeli Geriye Doğru Çıkarım



olumlu (karar): max(Sür 369,5, Sürme −20,0) = **369,5** → **Sür**.

Adım 6 / 16

```
1 def deger(n):
2     if not isinstance(n, tuple):    # yaprak
3         return n
4     tur, dallar = n
5     if tur == "S":                  # şans düğümü
6         return sum(p * deger(c) for p, c in dallar)
▶     return max(deger(c) for ad, c in dallar)  # karar
```

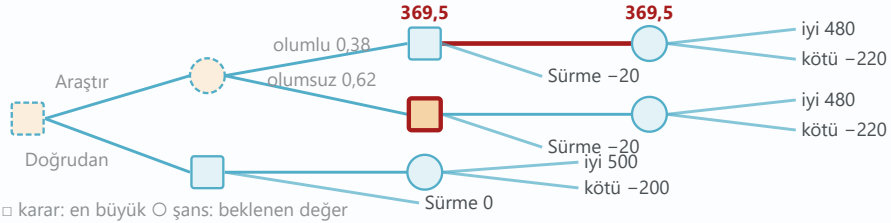
DEĞİŞKENLER

`n = olumlu` `return = 369.47`

ÇAĞRI YIĞINI

- **deger(olumlu)**
- deger(Araştır)
- deger(kök)

Kod İzleme: Özyinelemeli Geriye Doğru Çıkarım



`deger(olumsuz)` çağrıldı: karar düğümü; çocukları sırayla değerlendirilir.

Adım 7 / 16

```
1 def deger(n):
2     if not isinstance(n, tuple):    # yaprak
3         return n
▶     tur, dallar = n
▶     if tur == "S":                  # şans düğümü
6         return sum(p * deger(c) for p, c in dallar)
▶     return max(deger(c) for ad, c in dallar)  # karar
```

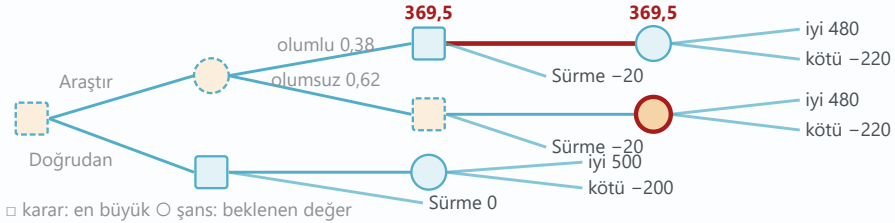
DEĞİŞKENLER

`n = olumsuz`

ÇAĞRI YIĞINI

- `deger(olumsuz)`
- `deger(Araştır)`
- `deger(kök)`

Kod İzleme: Özyinelemeli Geriye Doğru Çıkarım



`deger(olumsuz·Sür)` çağrıldı: şans düğümü; çocukları sırayla değerlendirilir.

Adım 8 / 16

```
1 def deger(n):
2     if not isinstance(n, tuple):    # yaprak
3         return n
▶     tur, dallar = n
▶     if tur == "S":                  # şans düğümü
6         return sum(p * deger(c) for p, c in dallar)
7     return max(deger(c) for ad, c in dallar)  # karar
```

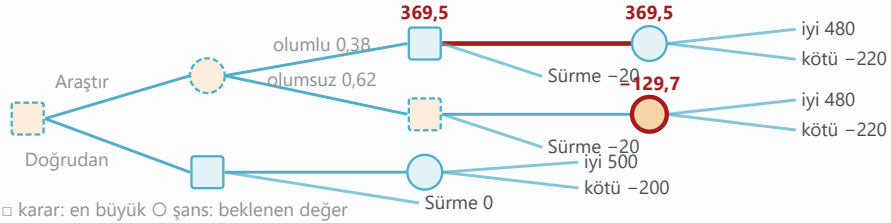
DEĞİŞKENLER

`n = olumsuz·Sür`

ÇAĞRI YIĞINI

- `deger(olumsuz·Sür)`
- `deger(olumsuz)`
- `deger(Araştır)`
- `deger(kök)`

Kod İzleme: Özyinelemeli Geriye Doğru Çıkarım



olumsuz·Sür (şans): 0,129·480 + 0,871·−220 = **−129,7**.

Adım 9 / 16

```
1 def deger(n):
2     if not isinstance(n, tuple):    # yaprak
3         return n
4     tur, dallar = n
5     if tur == "S":                  # şans düğümü
▶         return sum(p * deger(c) for p, c in dallar)
7     return max(deger(c) for ad, c in dallar)  # karar
```

DEĞİŞKENLER

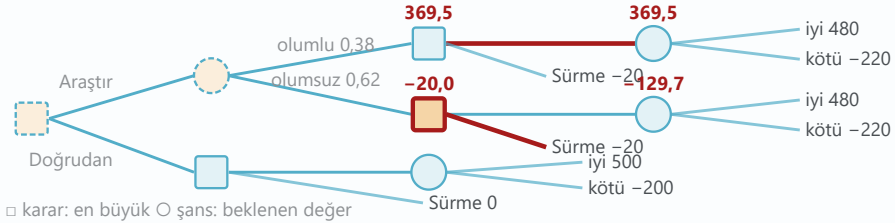
`n = olumsuz·Sür`

`return = -129.68`

ÇAĞRI YIĞINI

- `deger(olumsuz·Sür)`
- `deger(olumsuz)`
- `deger(Araştır)`
- `deger(kök)`

Kod İzleme: Özyinelemeli Geriye Doğru Çıkarım



olumsuz (karar): max(Sür −129,7, Sürme −20,0) = **−20,0** → **Sürme**.

Adım 10 / 16

```
1 def deger(n):
2     if not isinstance(n, tuple):    # yaprak
3         return n
4     tur, dallar = n
5     if tur == "S":                  # şans düğümü
6         return sum(p * deger(c) for p, c in dallar)
▶     return max(deger(c) for ad, c in dallar)  # karar
```

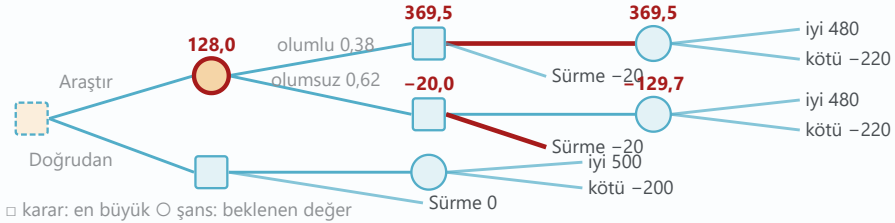
DEĞİŞKENLER

`n = olumsuz` `return = -20`

ÇAĞRI YIĞINI

- `deger(olumsuz)`
- `deger(Araştır)`
- `deger(kök)`

Kod İzleme: Özyinelemeli Geriye Doğru Çıkarım



Araştır (şans): 0,38·369,5 + 0,62·−20 = **128,0**.

Adım 11 / 16

```
1 def deger(n):
2     if not isinstance(n, tuple):    # yaprak
3         return n
4     tur, dallar = n
5     if tur == "S":                  # şans düğümü
▶         return sum(p * deger(c) for p, c in dallar)
7     return max(deger(c) for ad, c in dallar)  # karar
```

DEĞİŞKENLER

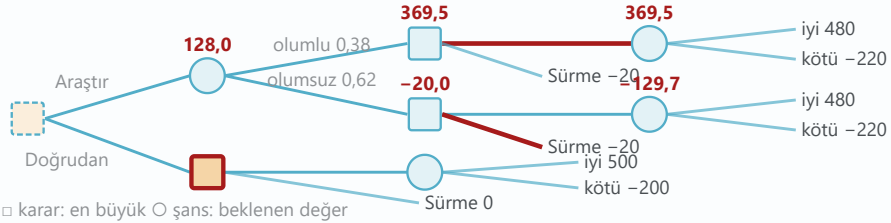
`n = Araştır` `return = 128.0`

ÇAĞRI YIĞINI

`deger(Araştır)`

`deger(kök)`

Kod İzleme: Özyinelemeli Geriye Doğru Çıkarım



`deger(Doğrudan)` çağrıldı: karar düğümü; çocukları sırayla değerlendirilir.

Adım 12 / 16

```
1 def deger(n):
2     if not isinstance(n, tuple):    # yaprak
3         return n
▶     tur, dallar = n
▶     if tur == "S":                  # şans düğümü
6         return sum(p * deger(c) for p, c in dallar)
▶     return max(deger(c) for ad, c in dallar)  # karar
```

DEĞİŞKENLER

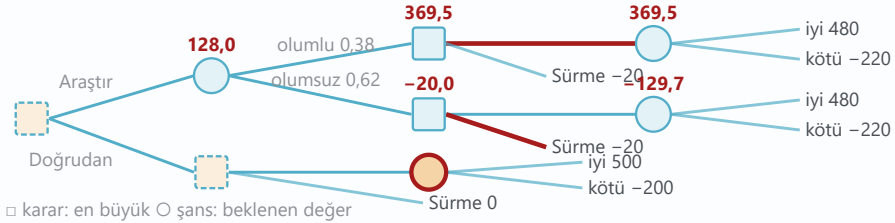
`n = Doğrudan`

ÇAĞRI YIĞINI

`deger(Doğrudan)`

`deger(kök)`

Kod İzleme: Özyinelemeli Geriye Doğru Çıkarım



`deger(Doğrudan·Sür)` çağrıldı: şans düğümü; çocukları sırayla değerlendirilir.

Adım 13 / 16

```
1 def deger(n):
2     if not isinstance(n, tuple):    # yaprak
3         return n
▶     tur, dallar = n
▶     if tur == "S":                  # şans düğümü
6         return sum(p * deger(c) for p, c in dallar)
7     return max(deger(c) for ad, c in dallar)  # karar
```

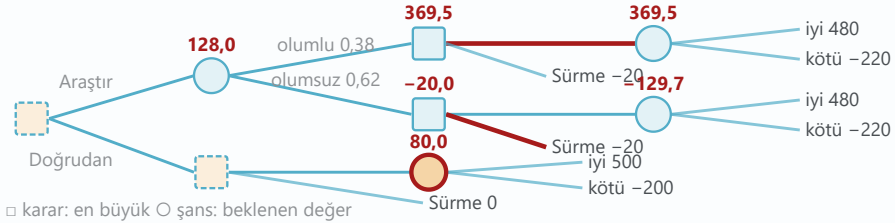
DEĞİŞKENLER

`n = Doğrudan·Sür`

ÇAĞRI YIĞINI

- `deger(Doğrudan·Sür)`
- `deger(Doğrudan)`
- `deger(kök)`

Kod İzleme: Özyinelemeli Geriye Doğru Çıkarım



Doğrudan·Sür (şans): 0,4·500 + 0,6·−200 = **80,0**.

Adım 14 / 16

```
1 def deger(n):
2     if not isinstance(n, tuple):    # yaprak
3         return n
4     tur, dallar = n
5     if tur == "S":                  # şans düğümü
▶         return sum(p * deger(c) for p, c in dallar)
7     return max(deger(c) for ad, c in dallar)  # karar
```

DEĞİŞKENLER

`n = Doğrudan·Sür`

`return = 80.0`

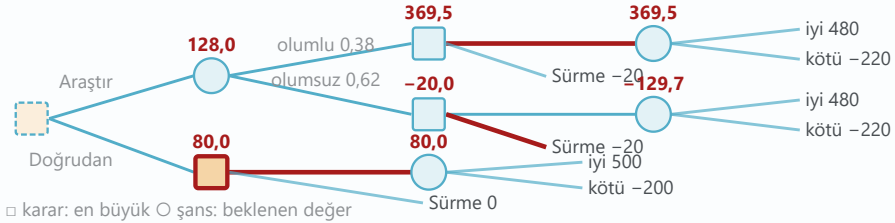
ÇAĞRI YIĞINI

`deger(Doğrudan·Sür)`

`deger(Doğrudan)`

`deger(kök)`

Kod İzleme: Özyinelemeli Geriye Doğru Çıkarım



Doğrudan (karar): max(Sür 80,0, Sürme 0,0) = **80,0 → Sür**.

Adım 15 / 16

```
1 def deger(n):
2     if not isinstance(n, tuple):    # yaprak
3         return n
4     tur, dallar = n
5     if tur == "S":                  # şans düğümü
6         return sum(p * deger(c) for p, c in dallar)
▶     return max(deger(c) for ad, c in dallar)  # karar
```

DEĞİŞKENLER

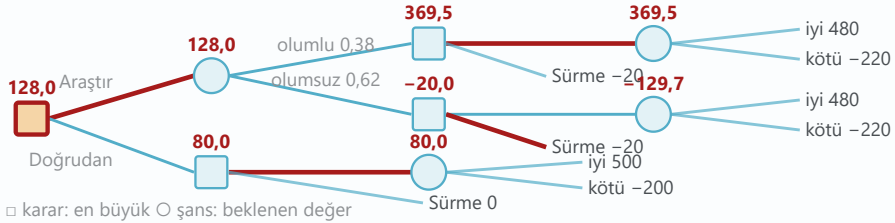
`n = Doğrudan` `return = 80.0`

ÇAĞRI YIĞINI

`deger(Doğrudan)`

`deger(kök)`

Kod İzleme: Özyinelemeli Geriye Doğru Çıkarım



kök (karar): max(Araştır 128,0, Doğrudan 80,0) = **128,0** → **Araştır**. Strateji: araştır; olumluysa sür, olumsuzsa sürme.

Adım 16 / 16

```
1 def deger(n):
2     if not isinstance(n, tuple):    # yaprak
3         return n
4     tur, dallar = n
5     if tur == "S":                  # şans düğümü
6         return sum(p * deger(c) for p, c in dallar)
▶     return max(deger(c) for ad, c in dallar)  # karar
```

DEĞİŞKENLER

`n = kök` `return = 128.0`

ÇAĞRI YIĞINI

`deger(kök)`

Bilginin Beklenen Değeri: EVPI ve EVSI

Durum	Hesap	Değer
Bilgi yok (hemen karar)	Sür: 0,4·500 + 0,6·(−200)	80
.	.	.
.	.	.
.	.	.
.	.	.

Bilgi olmadan en iyi eylem sürmek: beklenen değer 80.

Bilgi ancak kararı değiştirebiliyorsa değerlidir: olumsuz rapor "sürme" kararına yol açtığı için araştırmanın değeri pozitiftir.

Bilginin Beklenen Değeri: EVPI ve EVSI

Durum	Hesap	Değer
Bilgi yok (hemen karar)	Sür: 0,4·500 + 0,6·(−200)	80
Mükemmel bilgi (kâhin)	iyi → sür 500, kötü → sürme 0: 0,4·500 + 0,6·0	200
.	.	.
.	.	.
.	.	.

Bir kâhin pazarın durumunu önceden söyleseydi: iyi pazarda sürer (500), kötüde sürmezdi (0). Beklenen 200.

Bilgi ancak kararı değiştirebiliyorsa değerlidir: olumsuz rapor "sürme" kararına yol açtığı için araştırmanın değeri pozitiftir.

Bilginin Beklenen Değeri: EVPI ve EVSI

Durum	Hesap	Değer
Bilgi yok (hemen karar)	Sür: 0,4·500 + 0,6·(−200)	80
Mükemmel bilgi (kâhin)	iyi → sür 500, kötü → sürme 0: 0,4·500 + 0,6·0	200
EVPI = mükemmel bilginin beklenen değeri	200 − 80	**120**
.	.	.
.	.	.

EVPI = 200 − 80 = **120**: hiçbir bilgi kaynağına bundan fazlası ödenmez (üst sınır).

Bilgi ancak kararı değiştirebiliyorsa değerlidir: olumsuz rapor "sürme" kararına yol açtığı için araştırmanın değeri pozitiftir.

Bilginin Beklenen Değeri: EVPI ve EVSI

Durum	Hesap	Değer
Bilgi yok (hemen karar)	Sür: 0,4·500 + 0,6·(−200)	80
Mükemmel bilgi (kâhin)	iyi → sür 500, kötü → sürme 0: 0,4·500 + 0,6·0	200
EVPI = mükemmel bilginin beklenen değeri	200 − 80	**120**
Örnek bilgi (araştırma, maliyetsiz)	ağaçtaki 128 + araştırma maliyeti 20	148
.	.	.

Kusurlu araştırmanın brüt değeri: ağaçtaki 128'e maliyeti geri ekle ⇒ 148.

Bilgi ancak kararı değiştirebiliyorsa değerlidir: olumsuz rapor "sürme" kararına yol açtığı için araştırmanın değeri pozitiftir.

Bilginin Beklenen Değeri: EVPI ve EVSI

Durum	Hesap	Değer
Bilgi yok (hemen karar)	Sür: 0,4·500 + 0,6·(−200)	80
Mükemmel bilgi (kâhin)	iyi → sür 500, kötü → sürme 0: 0,4·500 + 0,6·0	200
EVPI = mükemmel bilginin beklenen değeri	200 − 80	**120**
Örnek bilgi (araştırma, maliyetsiz)	ağaçtaki 128 + araştırma maliyeti 20	148
EVSI = örnek bilginin beklenen değeri	148 − 80	**68**

EVSI = 148 − 80 = **68** > maliyet 20 ⇒ araştırma yapılmalı (net kazanç 48). Verimlilik EVSI/EVPI = %57.

Bilgi ancak kararı değiştirebiliyorsa değerlidir: olumsuz rapor "sürme" kararına yol açtığı için araştırmanın değeri pozitiftir.

ÖZET

Özet

Beklenen fayda

$E[u] = \Sigma\, p_i\, u(x_i)$. Piyango A: $E[X] = 450$ ama $\sqrt{x}$ ile $E[u] = 15 < u(400) = 20$.

Fayda ölçeği

$a \cdot u + b$ $(a > 0)$ dönüşümü kararı korur; Allais ve Ellsberg seçimleri beklenen faydadan sistematik olarak sapar.

Risk tutumu

İçbükey → kaçınan (KE 225, prim 225); doğrusal → nötr; dışbükey → seven.

Belirsizlik ölçütleri

Aynı tabloda maksimin Yatırım yok, maksimaks ve Laplace Büyük, Savage Küçük yatırımı seçer.

Bayesçi karar

Yaygınlık %1, duyarlılık %90, yanlış pozitif %9 ⇒ $P(\text{hasta} \mid +) \approx$ %9,2; tedavi ancak $p > 3/23$ iken.

Karar ağacı

Şansta ortalama, kararda maksimum. Araştır (128) > araştırmasız (80); EVPI = 120, EVSI = 68.

ÖZET

Risk ve Belirsizlik Altında Daha İyi Karar

Bilginin değeri hesaplanır

Güncel veri kararı iyileştirir; bilgi toplamak ancak EVSI maliyetinden büyükse kârlıdır.

Senaryo analizi ve karar ağacı

Olası durumlar ve olasılıkları açıkça yazılınca ağaç geriye doğru çözülebilir.

Ön olasılık hesaba katılır

Taban oranları Bayes kuralıyla son olasılığa girer; doğal sıklıklarla düşünmek hatayı azaltır.

Rakipler ve çeşitlendirme

Oyun kuramı ve simülasyon rakip davranışını modeller; portföy çeşitlendirmesi riski azaltır.

ÖZET

Etkileşimli Oyunlar

Bayesçi Karar Laboratuvarı
Ön olasılık, olabilirlik ve kayıplara göre son olasılık ve karar.

Blackjack
Kart çek / dur kararında batma olasılığı ve temel strateji ipucu.

Mayın Tarlası
Açılan sayılara göre hücrelerin mayın olasılığını güncelleme.

Pişti
Oynanan kartları hatırlayarak rakibin elindeki kartlar hakkında çıkarım.

Hokm
Eksik bilgili, takım hâlinde oynanan kart oyunu: koz seçimi ve sinyalleşme.

Poker
Eksik bilgi, olasılık ve blöf: rakibin elini olasılıkla tahmin etme.

Tavla
Zarlı oyun: hamleleri sezgisel bir değerlendirme fonksiyonuyla puanlama; üç farklı ağırlık profili.

ÖZET

Örnek Problemler

1. Piyango: %20 olasılıkla 2.500 TL, aksi hâlde 100 TL. Beklenen değer? $u(x) = \sqrt{x}$ için kesinlik eşdeğeri ve risk primi?
2. Sigorta örneğinde kayıp olasılığı %20 olsaydı en fazla ödenecek prim ne olurdu?
3. Tıbbi testte yaygınlık %10 olsaydı (duyarlılık %90, yanlış pozitif %9) $P(\text{hasta} \mid +)$ kaç olurdu?
4. Karar ağacında araştırma 70 bin TL olsaydı en iyi strateji ne olurdu?

▶ **Cevapları göster**

Kaynaklar

- J. von Neumann, O. Morgenstern, *Theory of Games and Economic Behavior*, Princeton University Press, 1944.
- S. Russell, P. Norvig, *Artificial Intelligence: A Modern Approach*, 4. baskı, Pearson, 2020 (Bölüm 12–16: belirsizlik, Bayes, karar kuramı).
- H. Raiffa, *Decision Analysis*, 1968; D. Bernoulli, "Specimen theoriae novae de mensura sortis", 1738.
- M. Allais, "Le comportement de l'homme rationnel devant le risque", *Econometrica* 21, 1953; L. J. Savage, "The theory of statistical decision", *JASA* 46, 1951; D. Ellsberg, *Quarterly J. Economics* 75, 1961.
- G. Gigerenzer, U. Hoffrage, "How to improve Bayesian reasoning without instruction: Frequency formats", *Psychological Review* 102(4), 1995.
- W. Casscells, A. Schoenberger, T. B. Graboys, "Interpretation by physicians of clinical laboratory results", *NEJM* 299, 1978.
- D. Easley, J. Kleinberg, ***Networks, Crowds, and Markets***, Cambridge University Press, 2010 (Bölüm 16: bilgi çağlayanları ve Bayes kuralı). Araç: **Gambit** (şans düğümlü ağaçları çizip geriye doğru çözer).

Son

Bölüm 1: Karar Kuramı

Sonraki bölüm: Bölüm 2: Oyunlar ve Temsiller

Çalışma soruları · Terim sözlüğü