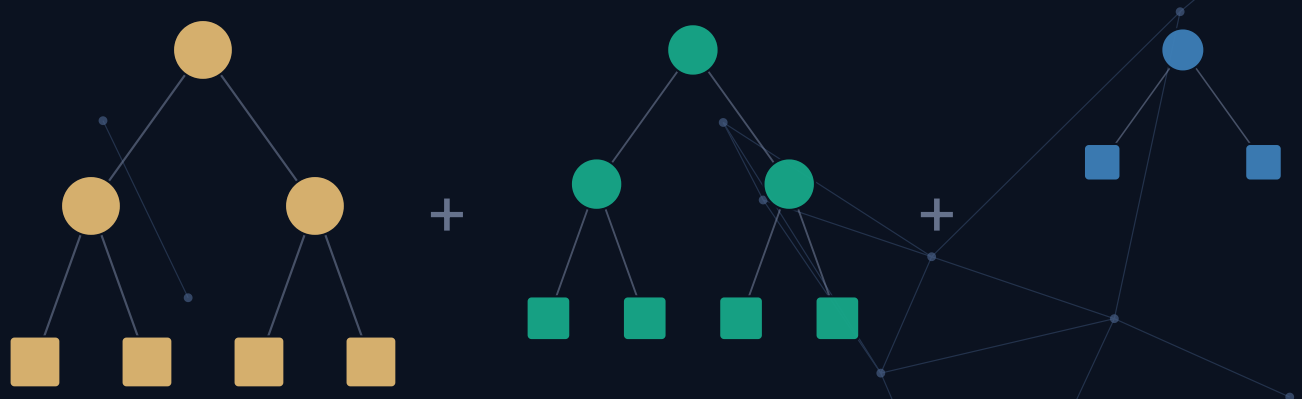
BCDataWorks

GRADİENT BOOSTİNG UYGULAMA REHBERİ

XGBoost Hiperparametreleri

Sezgisel açıklamalar, görsel rehber, kredi riski için başlangıç aralıkları ve adım adım tuning stratejisi.



· Parametre sezgisi · Kredi riski ayarları · Tuning stratejisi

bcdataworks.com · Çalışma Notları № 02

Bu doküman ne anlatıyor?

Amaç, parametrelerin yalnızca ne yaptığını değil; birbirleriyle nasıl etkileştiğini ve hangi problem belirtisinde hangi düğmenin çevrilmesi gerektiğini kurmaktır. Rehber boyunca her bölüm bir soru–belirti–eylem zinciri olarak okunabilir.

Tek cümlelik zihinsel model

max_depth ne öğrenebileceğini, **gamma / min_child_weight** neyin öğrenilmeye değer olduğunu, **learning_rate** ne kadar hızlı öğreneceğini, düzenlileştirme ve örnekleme ise öğrendiğine ne kadar güveneceğini belirler.

İçindekiler

Okuma önerisi

İlk kez öğreniyorsan sırasıyla ilerle. Halihazırda model kuruyorsan önce **2, 12, 13** ve **14.** bölümleri oku; sonra gözlemlediğin performans belirtisine göre ilgili parametre bölümüne dön.

Sürüm notu

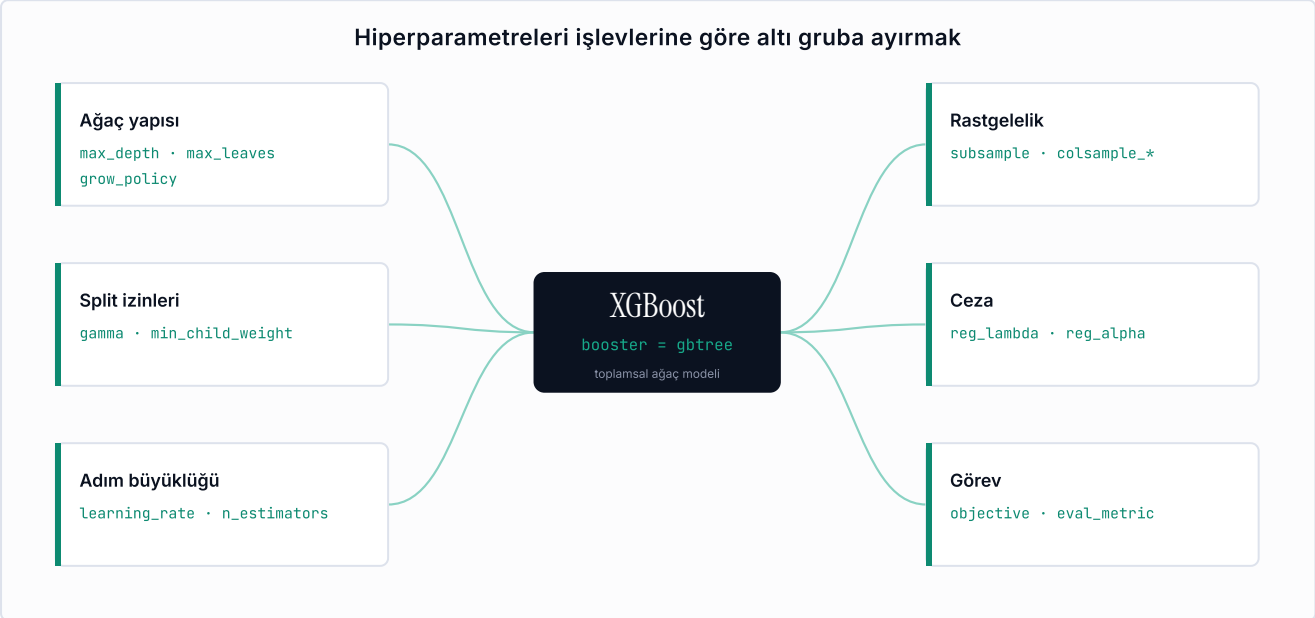
Rehber, 1 Ağustos 2026 tarihinde yayımda olan **XGBoost 3.3.0** kararlı dokümantasyonu temel alınarak hazırlanmıştır. Parametre isimleri scikit-learn arayüzündeki **XGBClassifier** kullanımına göre verilmiştir.

01 Önce büyük resim

XGBoost, her turda önceki modelin hatalarını azaltacak yeni bir karar ağacı ekleyen gradient boosting yöntemidir. Bir XGBoost modeli tek bir büyük ağaç değil, birçok küçük ağacın toplamıdır.

$$\text{Tahmin}(x) = \text{temel skor} + \eta \cdot [\ \text{Ağaç}_1(x) + \text{Ağaç}_2(x) + \ldots + \text{Ağaç}_m(x)\ ]$$

η = learning_rate, M = n_estimators (boosting turu sayısı)



Şekil 1. XGBoost hiperparametrelerinin işlevsel zihinsel haritası.

Grupların işlevi

- **Ağaç yapısı:** modelin ne kadar karmaşık etkileşimler kurabileceğini belirler.
- **Split izinleri:** bir dalın gerçekten açılmaya değer olup olmadığını sınar.
- **Adım büyüklüğü:** her yeni ağacın toplama ne kadar katkı yapacağını ve kaç tur yapılacağını kontrol eder.
- **Rastgelelik:** veri ve değişken örneklemesiyle varyansı azaltır.
- **Ceza:** yaprak skorlarını düzenlileştirir.
- **Görev:** neyin optimize edildiğini ve neyin izlendiğini tanımlar.

En önemli ayrım: objective ile eval_metric

objective eğitim kaybını belirler. **eval_metric** ise doğrulama performansını ölçer ve early stopping kararını yönlendirebilir. AUC ile izlemek, modelin doğrudan AUC kaybını optimize ettiği anlamına **gelmez**.

Hiperparametre ne değildir?

Modelin eğitim sırasında öğrendiği split eşikleri ve yaprak skorları **parametredir**. max_depth veya learning_rate ise eğitimden önce seçilen **hiperparametredir**.

02 En önemli parametreler

Tek sayfalık özet: ne kontrol eder, arttığında ne olur, nereden başlanır.

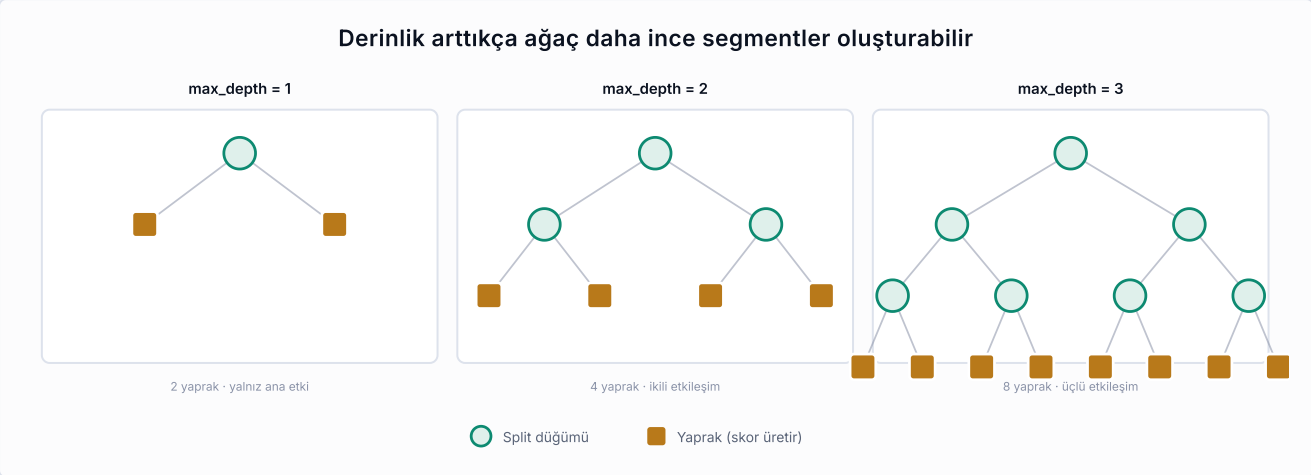
Parametre	Kontrol ettiği şey	Artınca / azalınca	Başlangıç
`learning_rate`	Yeni ağacın katkı payı	Azalırsa daha çok ağaç gerekir; genelleme iyileşebilir	0,01 – 0,10
`n_estimators`	Boosting turu / ağaç sayısı	Artarsa kapasite artar; early stopping ile seç	500 – 5000
`max_depth`	Ağaç derinliği	Artarsa etkileşim kapasitesi ve overfit riski artar	2 – 6
`min_child_weight`	Çocuk düğüm için minimum Hessian	Artarsa küçük ve kararsız yapraklar engellenir	1 – 50+
`gamma`	Split için minimum kayıp azalması	Artarsa düşük kazançlı splitler engellenir	0 – 5
`subsample`	Her turda kullanılan satır oranı	Azalırsa rastgelelik artar; çok düşerse underfit	0,6 – 1,0
`colsample_bytree`	Her ağaçta kullanılan kolon oranı	Değişken bağımlılığını ve korelasyon etkisini azaltabilir	0,5 – 1,0
`reg_lambda`	L2 yaprak cezası	Yaprak skorlarını yumuşatır	0,1 – 100
`reg_alpha`	L1 yaprak cezası	Küçük yaprak skorlarını sıfıra yaklaştırabilir	0 – 10
`scale_pos_weight`	Pozitif sınıfın ağırlığı	Nadir bad sınıfının etkisini yükseltir	negatif / pozitif
`tree_method`	Ağaç kurma algoritması	hist genellikle hızlı ve ölçeklenebilir	hist
`max_bin`	Histogram çözünürlüğü	Artarsa split hassasiyeti ve maliyet artar	128 / 256 / 512

Başlangıç değerleri evrensel değildir

Veri boyutu, bad oranı, değişken sayısı, örnek ağırlıkları ve validasyon tasarımı aynı değerin etkisini tamamen değiştirebilir. Bu aralıklar bir **arama alanı kurmak** içindir; nihai karar OOT performansıyla verilmelidir.

03 Model kapasitesi

max_depth, max_leaves ve grow_policy modelin öğrenebileceği etkileşimlerin karmaşıklığını belirler.



Şekil 2. max_depth arttıkça yaprak sayısı ve yakalanabilen etkileşim derecesi artar.

max_depth

Bir ağacın kökten yaprağa ulaşabileceği maksimum split sayısını sınırlar. Derinlik arttıkça model, örneğin "yüksek limit doluluğu + düşük KKB skoru + son ay nakit çekim artışı" gibi daha yüksek dereceli etkileşimleri yakalayabilir. Buna karşılık yapraklar küçülür; gürültü öğrenme ve bellek tüketimi artar.

> **Kredi riski sezgisi**
> `max_depth=3` yaklaşık olarak üç aşamalı karar etkileşimlerine izin verir. Bu matematiksel olarak yalnızca üç **değişken** demek değildir; aynı değişken bir dal boyunca birden fazla kez split edilebilir.

grow_policy ve max_leaves

Ayar	Davranış	Ne zaman düşünülür?
grow_policy = "depthwise"	Köke yakın düğümleri katman katman büyütür. Varsayılandır.	Kontrollü, dengeli ve kolay yönetilen ağaçlar.
grow_policy = "lossguide"	En yüksek kayıp azalmasını sağlayan düğümü önce büyütür.	Çok sayıda değişken ve seyrek etkileşimde; max_leaves ile birlikte.
max_leaves	Toplam yaprak sayısını sınırlar. 0 → sınır yoktur.	lossguide kullanımında kapasiteyi doğrudan kontrol etmek için.

> **Pratik tercih sırası**
> Klasik kredi riski tabular verisinde önce **depthwise + max_depth = 3–5** ile başlamak; daha sonra **lossguide + max_leaves** alternatifini aynı validasyon tasarımıyla karşılaştırmak daha güvenli bir sıradır.

04 Bölünme kapıları

gamma ve min_child_weight ikisi de ağacı muhafazakârlaştırır; fakat aynı şeyi yapmaz. Bir split hem **yeterli kazanç** üretmeli hem de **yeterli istatistik taşıyan** çocuklar oluşturmalıdır.

Şekil 3. Kazanç kapısı (gamma) ve istatistik kapısı (min_child_weight).

gamma (min_split_loss)

Aday splitin kabul edilmesi için gereken minimum kayıp azalmasıdır. gamma yükseldikçe küçük iyileştirmeler reddedilir. Gürültülü değişkenlerde veya eğitim–validasyon farkı yüksekse faydalıdır.

min_child_weight

Bir çocuk düğümde bulunması gereken minimum Hessian toplamıdır. Sınıflandırmada bunu doğrudan "minimum gözlem sayısı" olarak okumak **doğru değildir**; gözlem ağırlıkları ve tahmin olasılıkları Hessianı etkiler. Değer yükseldikçe küçük ve istatistiksel olarak zayıf yapraklar oluşmaz.

Belirti	Önce denenecek ayar	Neden
Train çok iyi, OOT zayıf	min_child_weight artır; gamma artır	Küçük ve marjinal splitleri sınırlar.
Model belirgin biçimde underfit	min_child_weight azalt; gamma azalt	Daha fazla split ve küçük segmentlere izin verir.
Çok nadir segmentlerde aşırı skor	min_child_weight artır	Az destekli yaprakları engeller.
Ağaçlarda çok sayıda düşük gain split	gamma artır	Düşük katkılı dalları keser.

İkisi arasındaki fark tek cümlede

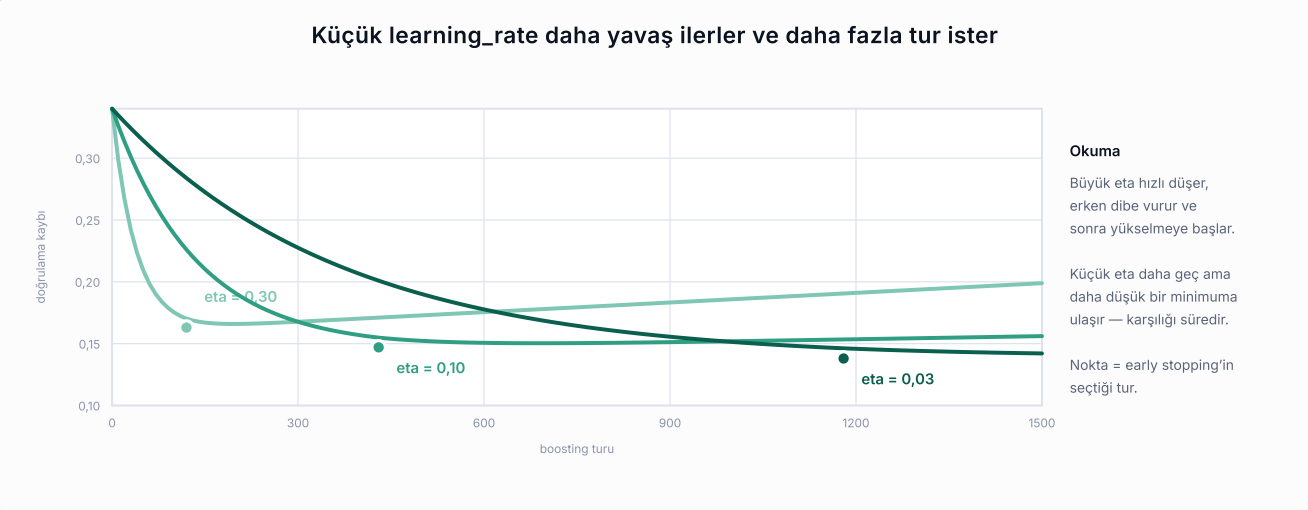
gamma splitin **ne kadar iyi** olduğunu sorgular; **min_child_weight** ise splitin **ne kadar veriye dayandığını**. Yüksek gamma, kazancı zayıf ama kalabalık splitleri de reddeder; yüksek min_child_weight, kazancı yüksek görünen ama yalnızca birkaç gözleme dayanan splitleri reddeder. Overfit belirtisinde ikisini birlikte değil, **sırayla** dene ki hangisinin etkili olduğunu görebilesin.

Ağırlık kullanıyorsan dikkat

`sample_weight` verildiğinde min_child_weight'in ölçeği değişir. Ağırlıkları 10 kat büyütürsen aynı min_child_weight değeri fiilen 10 kat gevşer. Ağırlık şemasını değiştirdiğinde bu parametreyi **yeniden ara**.

05 Öğrenme dinamiği

learning_rate, n_estimators ve early_stopping_rounds ayrı ayrı değil, **tek bir sistem** olarak düşünülmelidir.



Şekil 4. Temsili doğrulama kaybı eğrileri; noktalar early stopping ile seçilen turu gösterir.

learning_rate (eta)

Her yeni ağacın tahmine eklenen katkısını küçültür. Büyük değerler hızlı öğrenir; fakat optimumu aşabilir ve erken overfit olabilir. Küçük değerler daha ince adımlar atar; buna karşılık daha fazla ağaç ve eğitim süresi ister.

n_estimators ve early_stopping_rounds

n_estimators, scikit-learn arayüzündeki boosting turu sayısıdır; native `xgb.train` tarafındaki karşılığı `num_boost_round` kavramıdır. Tek başına yüksek olması sorun değildir; early stopping varsa bir **üst sınır** olarak düşünülebilir. early_stopping_rounds ise doğrulama metriği belirtilen tur boyunca iyileşmezse eğitimi durdurur. XGBClassifier veri bölmeyi kendi yapmaz; `fit` sırasında `eval_set` verilmelidir.

Güvenli reçete

n_estimators değerini yüksek (ör. **3000–5000**), learning_rate değerini düşük/orta (ör. **0,02–0,08**), early_stopping_rounds değerini **50–200** arasında başlat. En iyi turu validation üzerinde bul; sonra OOT üzerinde **bir kez** değerlendir.

CV uyarısı

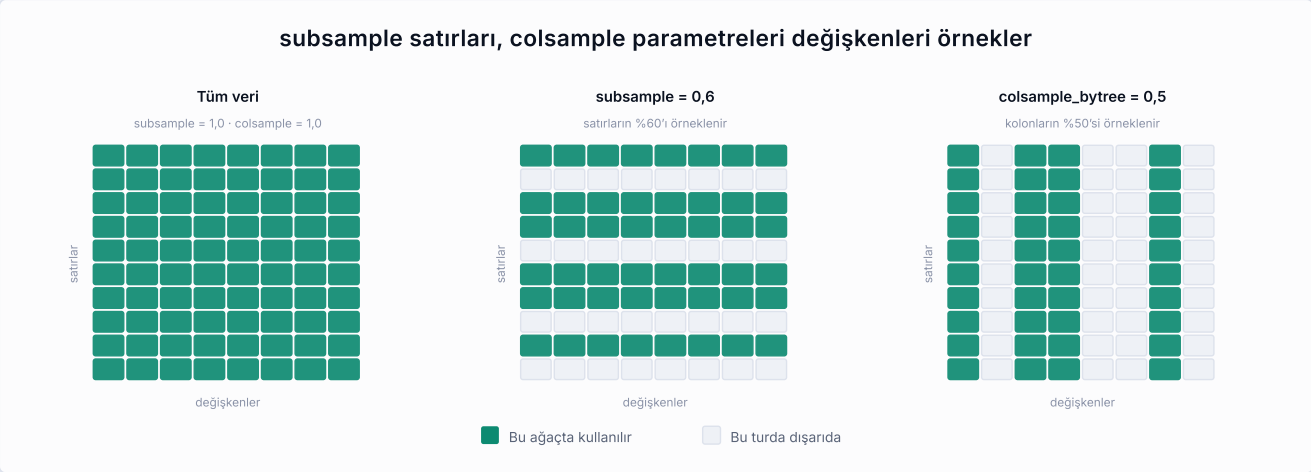
CV ile yapı parametrelerini seç; ardından modeli ayrı bir validation setinde early stopping ile yeniden eğit. Böylece her foldun farklı ağaç sayısı seçmesiyle oluşan karmaşayı azaltırsın. Nihai OOT/test seti early stopping için **kullanılmamalıdır**.

Kabaca değişmeyen çarpım

learning_rate ile en iyi tur sayısı yaklaşık ters orantılıdır: eta'yı yarıya indirirsen gereken tur sayısı kabaca iki katına çıkar. Bu yüzden eta'yı düşürmek "bedava kalite" değildir; eğitim süresi, model dosyası boyutu ve skorlama gecikmesi de artar. Kararı yalnız Gini ile değil, **süre – performans** eğrisiyle ver.

06 Rastgelelik

Satır ve kolon örneklemesi, tek tek ağaçları biraz zayıflatıp topluluğun genellemesini güçlendirebilir.



Şekil 5. Aynı veri matrisinde satır ve kolon örneklemenin etkisi.

subsample

Her boosting turunda eğitim satırlarının ne kadarının kullanılacağını belirtir. 0,8 değeri, her yeni ağaç için satırların yaklaşık yüzde 80'inin örneklenmesi demektir. Çok düşük değer, özellikle küçük veri setinde bilgi kaybı ve kararsızlık yaratabilir.

colsample ailesi

Parametre	Örnekleme zamanı	Sezgisel kullanım
`colsample_bytree`	Her yeni ağaçta bir kez	En yaygın kolon örnekleme parametresi.
`colsample_bylevel`	Her yeni derinlik seviyesinde	Aynı seviyedeki split adaylarını sınırlar.
`colsample_bynode`	Her split düğümünde	En yüksek rastgelelik; exact yönteminde desteklenmez.

Oranlar kümülatif çalışabilir

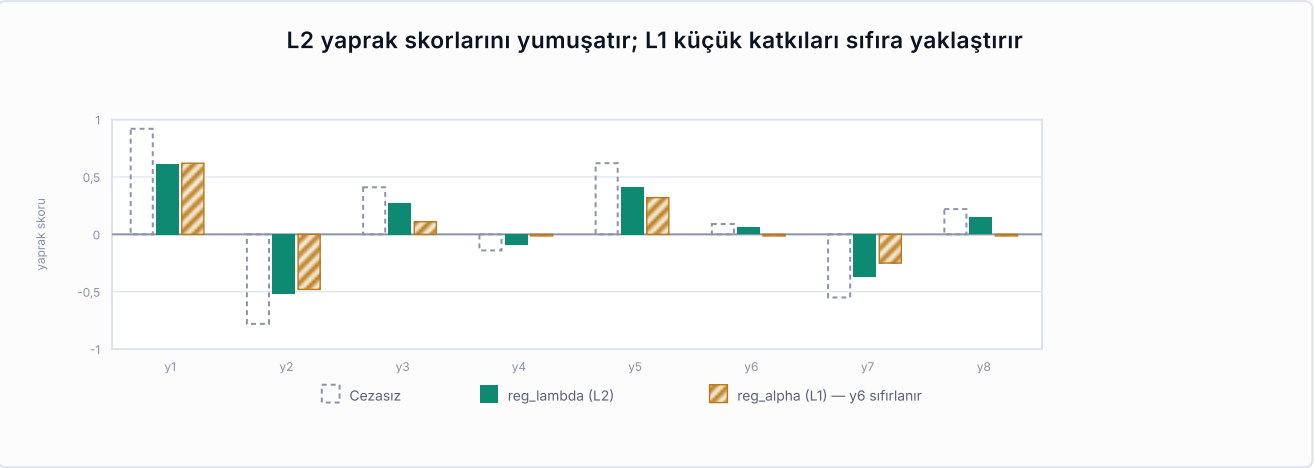
64 değişkenli bir veride üç colsample parametresinin de 0,5 olması, tek bir splitte aday havuzunu yaklaşık **8 değişkene** kadar indirebilir. Üçünü birden düşürmeden önce etkisini ayrı ayrı ölç.

Korelasyonlu bankacılık verisi

Aynı davranışı farklı pencerelerle ölçen çok sayıda değişken varsa `colsample_bytree = 0,6-0,9` aralığı, modelin sürekli aynı güçlü değişken ailesine yaslanmasını azaltabilir.

07 Düzenlileştirme

reg_lambda ve reg_alpha, ağaç **yapısından** çok yaprak **skorlarının büyüklüğünü** kontrol eder.



Şekil 6. Temsili gösterim; gerçek etki veri ve objective'e göre değişir.

reg_lambda (L2)

Yaprak ağırlıklarına karesel ceza uygular. Değer yükseldikçe uç yaprak skorları yumuşar, model daha muhafazakâr olur. Genellikle sıfırdan büyük bir başlangıç değeri uygundur; varsayılan **1**'dir.

reg_alpha (L1)

Mutlak değer temelli ceza uygular. Küçük yaprak katkılarını sıfıra yaklaştırabildiği için daha **seyrek** bir model etkisi yaratabilir. Çok sayıda gürültülü değişken varsa denenebilir.

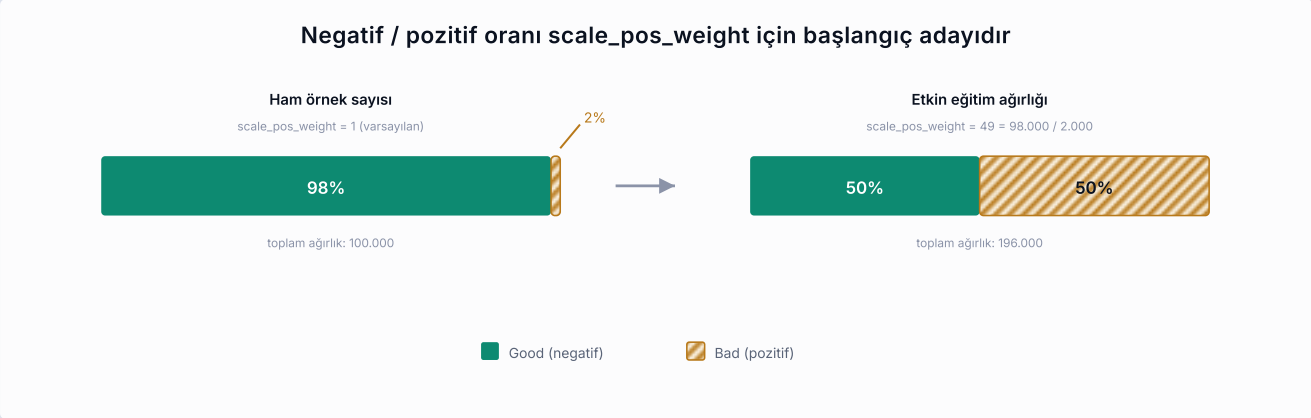
Durum	reg_lambda	reg_alpha
Skorlar aşırı uçlara gidiyor	Önce artır	Gerekirse hafif artır
Çok sayıda küçük katkılı yaprak	Artırılabilir	Özellikle artır
Model underfit	Azalt	Azalt
Değişken sayısı yüksek ve gürültülü	Orta / yüksek	Log ölçekte ara

Arama ölçeği

L1 ve L2 için lineer `0–1–2` araması yerine logaritmik aralıklar daha faydalıdır: `0 · 0,01 · 0,1 · 1 · 10 · 100` gibi.

08 Sınıf dengesizliği

scale_pos_weight modelin pozitif sınıfa verdiği **eğitim ağırlığını** değiştirir; karar eşiğini değiştirmekle aynı şey değildir.



Şekil 7. 98.000 good / 2.000 bad örneğinde başlangıç adayı 49'dur.

scale_pos_weight

Resmî dokümantasyonun tipik başlangıç önerisi, negatif örnek ağırlıkları toplamının pozitif örnek ağırlıkları toplamına oranıdır. Bu değer kesin optimum değildir; **0,5×**, **1×** ve **2×** çevresinde iş metriğiyle test edilebilir.

max_delta_step

Her yaprak güncellemesinin maksimum adımını sınırlar. Varsayılan 0, yani sınır yoktur. Aşırı dengesiz lojistik sınıflandırmada **1–10** aralığı güncellemeleri daha kontrollü hale getirebilir.

> **Sıralama ile olasılık aynı hedef değildir**
>
> scale_pos_weight; AUC, recall veya top capture performansını iyileştirebilir. Fakat `predict_proba` çıktılarının **PD kalibrasyonunu bozabilir**. PD kullanılacaksa calibration curve, Brier score ve dönem/segment kalibrasyonu ayrıca incelenmelidir.

İş hedefi	Önerilen yaklaşım
Bad yakalama / sıralama	scale_pos_weight dene; AUC-PR, top capture, KS ve maliyet metriğiyle seç.
Doğru PD olasılığı	Aşırı yeniden ağırlıklandırmadan kaçın veya sonradan kalibrasyon uygula; logloss / Brier izle.
Operasyonel karar	Model skoru ile karar eşiğini maliyet matrisi üzerinden birlikte optimize et.

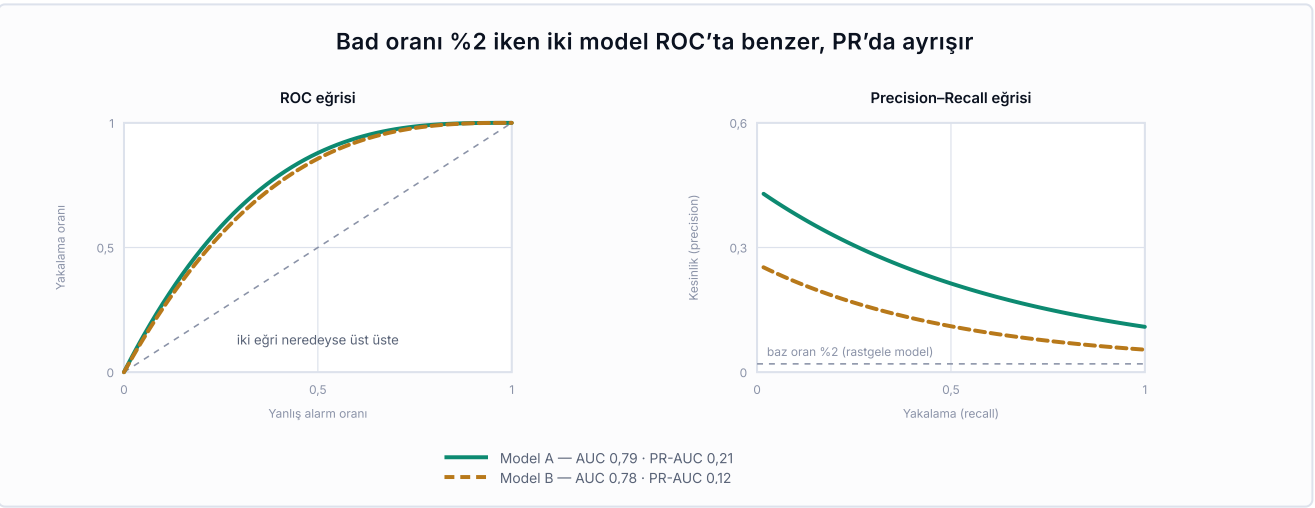
09 Görev tanımı

objective modelin optimize ettiği kaybı; eval_metric ise eğitim sırasında izlenen ölçümü belirler.

Kullanım	objective	Önerilen eval_metric
İkili kredi riski	binary:logistic	auc, aucpr, logloss
Çok sınıflı risk segmenti	multi:softprob	mlogloss, merror
Sürekli hedef / LGD	reg:squarederror	rmse, mae
Sayım / olay adedi	count:poisson	poisson-nloglik
Sağkalım / time-to-default	survival:cox, survival:aft	cox-nloglik, aft-nloglik

Kredi riskinde metrik seçimi

- **AUC / Gini:** Sıralama gücünü ölçer. Gini = 2 × AUC − 1.
- **AUC-PR:** Bad oranı çok düşük olduğunda pozitif sınıf performansına daha duyarlıdır.
- **Logloss:** Tahmin olasılıklarının doğruluğunu cezalandırır; kalibrasyona daha duyarlıdır.
- **KS:** Risk dağılımlarını ayırma gücünü gösterir; XGBoost içinde standart built-in early-stop metriği **değildir**, özel metrik yazılabilir.
- **Top capture / lift:** Operasyonel kapasite ve tahsilat/önlem havuzu için iş açısından çok değerlidir.



Şekil 8. Nadir pozitif sınıfta AUC-PR, AUC-ROC'un gizlediği farkı görünür kılar.

Early stopping ayrıntısı

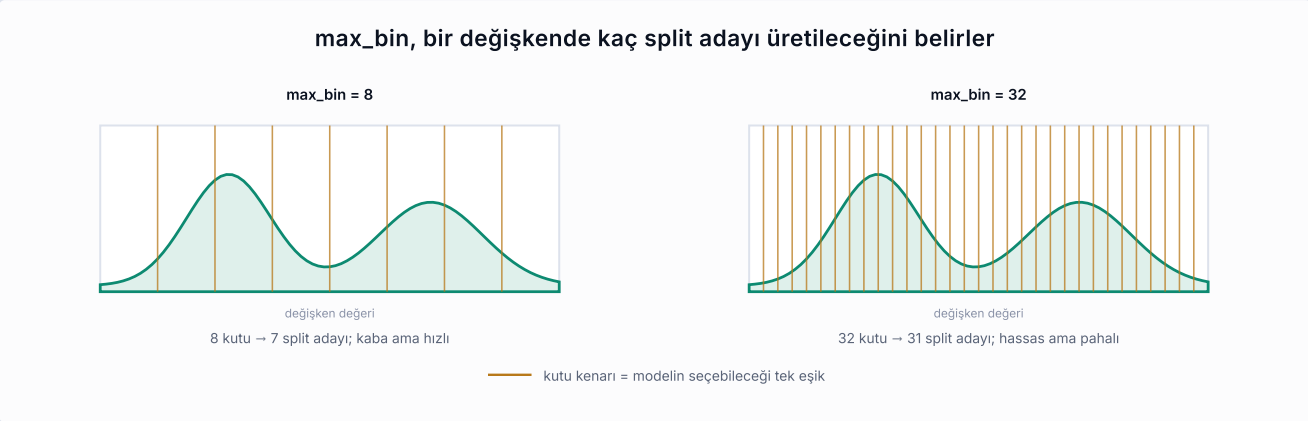
Birden çok eval_metric verirsen early stopping davranışını kullandığın API ve metrik sırasına göre kontrol et. Native API'de **son metrik ve son eval seti** kullanılır; scikit-learn tarafında açık callback ile `metric_name` ve `data_name` belirtmek en şeffaf yöntemdir.

base_score

Başlangıç tahminini / intercepti temsil eder. XGBoost **3.1.0** ve sonrasında seçili objective'lerde otomatik tahmin edilir. Yeterli boosting turu varsa genellikle bir tuning önceliği değildir.

10 Hız ve donanım

Bu parametreler çoğunlukla model mantığından çok, ağaçların **nasıl ve nerede** kurulduğunu belirler.



Şekil 9. hist yöntemi sürekli değişkeni kutulara ayırır; split yalnız kutu kenarlarında olabilir.

Parametre	Öneri	Not
`tree_method = "hist"`	Çoğu tabular iş için varsayılan başlangıç	Histogram tabanlı; hızlı ve ölçeklenebilir.
`tree_method = "exact"`	Küçük veri ve özel karşılaştırma	Tüm split adaylarını tarar; yavaş, özellik desteği sınırlı.
`tree_method = "approx"`	Yaklaşık quantile tabanlı alternatif	Bazı weighted-Hessian senaryolarında farklı davranabilir.
`device = "cpu"`	Standart CPU eğitimi	Kurumsal ortamda en uyumlu seçenek.
`device = "cuda"`	NVIDIA CUDA ile GPU	Güncel kullanım; eski gpu_hist örneklerini kopyalama.
`max_bin`	128–512 arasında karşılaştır	Yüksek değer daha hassas split; daha fazla süre ve bellek.
`n_jobs`	Paralel arama ile birlikte ayarla	GridSearch n_jobs ve model n_jobs aynı anda yüksekse çekişme olur.
`sampling_method`	uniform veya gradient_based	3.2 itibarıyla gradient-based sampling hist ile CPU ve GPU'da desteklenir.

Paralellik tuzağı

`RandomizedSearchCV(n_jobs=8)` içinde `XGBClassifier(n_jobs=8)` çalıştırmak 64 iş parçacığına yakın yük yaratabilir. Genellikle **dış arama paralelse model n_jobs düşük**; tek model eğitiliyorsa model n_jobs yüksek seçilir.

GPU her zaman daha hızlı değildir

Küçük veri, yoğun veri dönüşümü veya düşük ağaç sayısında GPU aktarım maliyeti avantajı silebilir. Süreyi **aynı validasyon ve aynı parametrelerle** ölç; dönüşüm maliyetini de dahil et.

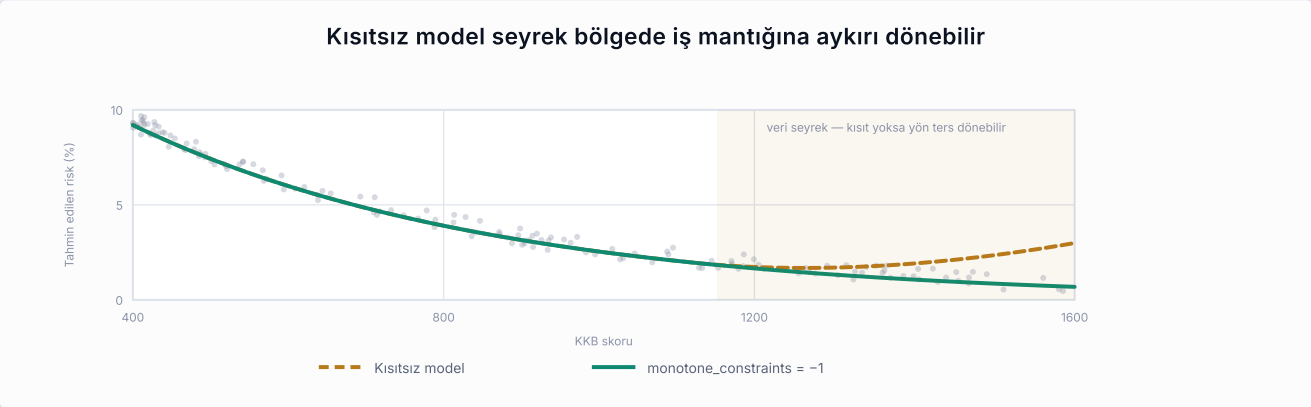
11 İleri parametreler

Temel parametreler oturmadan ileri seçeneklere geçmek genellikle gereksiz karmaşıklık yaratır.

monotone_constraints

Bir değişken arttığında tahminin yalnızca artmasını veya yalnızca azalmasını zorunlu kılar; böylece "KKB skoru arttıkça risk artamaz" gibi bir iş kuralı modelde garanti edilir. Veri ilişkisi gerçekten monoton değilse performansı azaltabilir.

```
model = XGBClassifier(
    monotone_constraints={
        "GGS": 1,              # arttıkça risk skoru azalamaz
        "KKB_SCORE": -1,       # arttıkça risk skoru artamaz
        "LIMIT_UTIL": 1
    }
)
```



Şekil 10. monotone_constraints, veri seyrekleştiğinde yönü iş kuralına sabitler.

interaction_constraints

Hangi değişken gruplarının birbirleriyle etkileşebileceğini sınırlar. Model yönetişimi, leakage şüphesi veya belirli değişken ailelerinin ayrılması için kullanılabilir.

Kategorik değişkenler

- `enable_categorical=True` — Pandas `category` gibi uygun veri tipleriyle native kategorik splitleri açar.
- `max_cat_to_onehot` — Kategori sayısı eşikten düşükse one-hot benzeri split; yüksekse partition tabanlı split seçimine etki eder.
- `max_cat_threshold` — Partition tabanlı splitte değerlendirilen kategori sayısını sınırlandırarak overfittingi kontrol eder.
- **Kısıt:** `exact` tree_method kategorik özelliklerle desteklenmez.

DART booster

DART, boosting sırasında önceki ağaçların bir kısmını drop ederek dropout benzeri regularizasyon uygular. `rate_drop`, `skip_drop`, `sample_type` ve `normalize_type` parametreleri vardır. Tahmin ve tuning davranışı daha karmaşık olduğu için klasik tabular kredi riskinde ilk tercih olarak **gbtree** önerilir.

> **Sürüm uyarısı**
> XGBoost 3.3.0'da `booster="gblinear"` **deprecated** durumundadır ve gelecekte kaldırılması planlanmaktadır. Bu rehber ağaç tabanlı gbtree kullanımına odaklanır.

12 Kredi riski için başlangıç aralıkları

Aşağıdaki alanlar, nadir bad sınıflı orta/büyük tabular veri için pratik bir arama başlangıcıdır; kesin reçete değildir.

Parametre	Arama alanı	Not
`learning_rate`	[0,02 · 0,03 · 0,05 · 0,08 · 0,1]	n_estimators ile birlikte
`n_estimators`	1000 – 5000	early stopping üst sınırı
`max_depth`	[2 · 3 · 4 · 5 · 6]	Önce sığ ağaç
`min_child_weight`	[1 · 3 · 5 · 10 · 20 · 50]	Büyük veride daha yüksek
`gamma`	[0 · 0,05 · 0,1 · 0,5 · 1 · 2 · 5]	Düşük gain split filtresi
`subsample`	[0,6 · 0,7 · 0,8 · 0,9 · 1,0]	Küçük veride çok düşürme
`colsample_bytree`	[0,5 · 0,6 · 0,7 · 0,8 · 0,9 · 1,0]	Korelasyonlu değişkende faydalı
`reg_lambda`	[0,1 · 0,5 · 1 · 2 · 5 · 10 · 50 · 100]	Log ölçek kullan
`reg_alpha`	[0 · 0,001 · 0,01 · 0,1 · 0,5 · 1 · 5 · 10]	Log ölçek kullan
`scale_pos_weight`	1 · oran/2 · oran · oran×2	Kalibrasyonu kontrol et
`max_delta_step`	[0 · 1 · 2 · 5 · 10]	Aşırı dengesizlikte
`max_bin`	[128 · 256 · 512]	Süre – performans dengesi

Önerilen ilk baseline

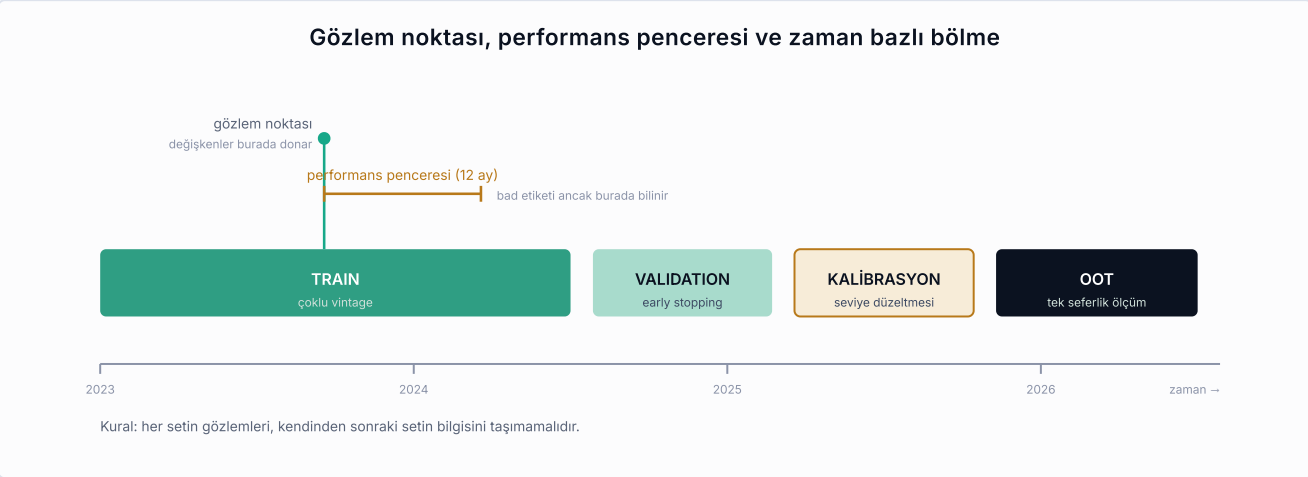
`max_depth=4`, `min_child_weight=10`, `gamma=0.1`, `subsample=0.8`, `colsample_bytree=0.8`, `reg_lambda=5`, `reg_alpha=0.1`, `learning_rate=0.05`, `n_estimators=3000` + early stopping. Bu sadece sağlam bir **başlangıç noktasıdır**.

OOT tasarımı

Kredi riskinde rastgele train–test bölmesi tek başına yeterli değildir. Hiperparametre seçimini **zaman bazlı validation** ile, nihai performansı daha ileri bir **OOT dönemiyle** ölç. Aynı müşterinin farklı tarihli kayıtlarının iki sete dağılmasını engelle.

13 Kredi riskinde validasyon tasarımı

Kredi riskinde hiperparametre seçiminin kalitesini belirleyen şey arama yöntemi değil, **zaman ve müşteri boyutunun doğru kurgulanmasıdır**. Yanlış bir bölme, en iyi tuning'i bile geçersiz kılar.



Şekil 11. Her gözlem, kendi performans penceresi kapandıktan sonra etiketlenebilir.

Önce hedefi tanımla

Gözlem noktası değişkenlerin hesaplandığı andır; **performans penceresi** bad etiketinin arandığı süredir (yaygın olarak 12 ay). Bad tanımı — çoğunlukla **90+ gün gecikme** veya takip/yasal takip statüsü — değişken üretiminden önce netleşmelidir. Belirsiz (indeterminate) segment eğitimden çıkarılıyorsa bu karar da raporlanmalıdır, çünkü modelin gördüğü bad oranını doğrudan değiştirir.

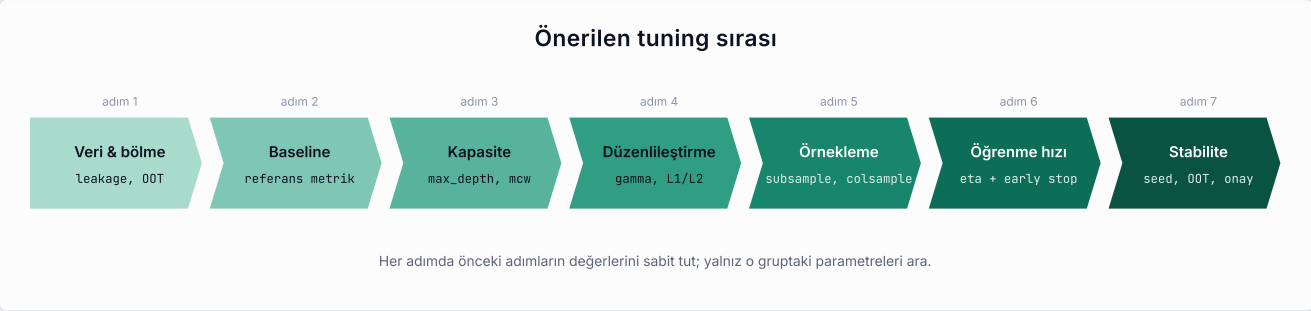
Tasarım kararı	Yanlış yapılırsa	Doğru yaklaşım
Bölme ekseni	Rastgele split, aynı dönemin bilgisini her sete dağıtır; Gini iyimser çıkar.	Zaman bazlı: train → validation → OOT sıralı dönemler.
Müşteri tekrarı	Aynı müşterinin farklı tarihli kayıtları iki sete düşer; model müşteriyi ezberler.	Bölmeyi müşteri kimliği üzerinden yap (grup bazlı split).
Performans penceresi	Penceresi henüz kapanmamış gözlemler "good" sayılır; bad oranı düşük görünür.	Yalnız penceresi dolmuş gözlemleri etiketle; kuyruğu kes.
Değişken zamanlaması	Gözlem noktasından sonraki bilgi değişkene sızar (leakage).	Her değişkenin veri kesim tarihini gözlem noktasına sabitle.
Makro dönem	Tek bir dönemin koşulları modele sabit gibi öğretilir.	Train birden fazla vintage içersin; OOT farklı bir dönem olsun.

Reddedilen başvurular

Başvuru skorkartlarında model yalnız **kabul edilmiş** müşterilerin performansını görür. Bu seçilim yanlılığı, XGBoost dahil her yöntemde vardır ve hiperparametre ayarıyla çözülmez. Reject inference uygulanacaksa yöntemi ve varsayımı ayrıca dokümante et.

14 Adım adım tuning süreci

Her parametreyi aynı anda dev bir grid içinde aramak hem pahalı hem de yorumlanması zordur.



Şekil 12. Kapasiteden öğrenme hızına doğru daralan bir arama sırası.

1. Önce veri sızıntısını, tarih bölmesini, müşteri tekrarlarını ve hedef tanımını doğrula.
2. Basit bir baseline kur; train, validation ve OOT metriklerini **birlikte** kaydet.
3. `max_depth` / `max_leaves` ve `min_child_weight` ile kapasiteyi ayarla.
4. `gamma`, `reg_lambda` ve `reg_alpha` ile gereksiz karmaşıklığı azalt.
5. `subsample` ve `colsample_bytree` ile varyansı düşür.
6. `learning_rate` değerini düşür; `n_estimators` üst sınırını yükselt ve early stopping ile en iyi turu seç.
7. `scale_pos_weight` / `max_delta_step` seçeneklerini iş hedefi ve kalibrasyonla birlikte değerlendir.
8. En iyi birkaç modeli farklı seed ve OOT pencerelerinde **stabilite testine** sok.
9. Son modeli yalnızca AUC/Gini ile değil; PR-AUC, KS, lift, kalibrasyon, PSI ve maliyet etkisiyle onayla.

Arama yöntemi

GridSearch küçük ve ayrık alanlarda iyidir. **RandomizedSearch** daha geniş alanlarda daha verimlidir. **Optuna / Bayesian optimization** sürekli ve log ölçekli parametrelerde avantajlıdır. Yöntemden bağımsız olarak **doğru validasyon tasarımı** daha önemlidir.

Sıranın mantığı

Kapasite önce gelir çünkü diğer parametrelerin etkisi kapasiteye bağlıdır: derinliği 8 olan bir ağaçta gamma'nın etkisi ile derinliği 3 olan bir ağaçtakinden farklıdır. learning_rate en sona bırakılır çünkü onu düşürmek neredeyse her zaman performansı bir miktar iyileştirir ve bu iyileşme diğer parametrelerin gerçek etkisini **maskeler**.

Her adımda kaydet

Her turda yalnız en iyi skoru değil; denenen alanı, seçilen değeri, train–validation–OOT üçlüsünü ve en iyi tur sayısını yaz. Tuning'in çıktısı bir parametre sözlüğü değil, **hangi parametrenin ne kadar önemli olduğunu gösteren bir kayıttır**. Bu kayıt olmadan bir sonraki modelde aynı aramayı baştan yaparsın.

15 Python uygulaması

scikit-learn arayüzüyle güvenli bir XGBClassifier başlangıcı.

```python
from xgboost import XGBClassifier
from sklearn.metrics import roc_auc_score, average_precision_score, log_loss

# pozitif sınıf ağırlığı: negatif / pozitif
neg = int((y_train == 0).sum())
pos = int((y_train == 1).sum())
spw = neg / max(pos, 1)

model = XGBClassifier(
    objective="binary:logistic",
    eval_metric=["auc", "logloss"],
    tree_method="hist",
    device="cpu",              # NVIDIA CUDA varsa: "cuda"
    n_estimators=4000,
    learning_rate=0.03,
    max_depth=4,
    min_child_weight=10,
    gamma=0.10,
    subsample=0.80,
    colsample_bytree=0.80,
    reg_lambda=5.0,
    reg_alpha=0.10,
    scale_pos_weight=spw,
    max_delta_step=1,
    early_stopping_rounds=100,
    random_state=42,
    n_jobs=8
)

model.fit(
    X_train, y_train,
    eval_set=[(X_valid, y_valid)],   # buraya OOT/test KOYMA
    verbose=False
)
```

Önemli

`eval_set` içine test/OOT setini koyma. OOT seti model ve hiperparametre seçiminden **tamamen bağımsız** kalmalıdır; aksi halde raporladığın performans iyimser olur.

Bu bloktaki üç bilinçli tercih

- `eval_metric=["auc","logloss"]` — sıralamayı ve kalibrasyonu aynı anda izler; hangisinin bozulduğunu ancak ikisini birlikte bakarak görürsün.
- `max_delta_step=1` — spw yüksekken lojistik güncellemelerin patlamasını sınırlar; dengesizlik düşükse 0'a çekilebilir.
- `random_state=42` — subsample/colsample aktifken sonucun tekrarlanabilir olması için zorunludur.

Doğrulama metriklerini birlikte raporla

```
p_valid = model.predict_proba(X_valid)[:, 1]

auc    = roc_auc_score(y_valid, p_valid)
pr_auc = average_precision_score(y_valid, p_valid)
gini   = 2 * auc - 1
ll     = log_loss(y_valid, p_valid)

print("Best iteration:", model.best_iteration)
print(f"AUC={auc:.4f} | Gini={gini:.4f} | PR-AUC={pr_auc:.4f} | LogLoss={ll:.4f}")
```

RandomizedSearch sonrası final eğitim mantığı

```
# 1) CV veya zaman bazlı aramayla YAPI parametrelerini seç.
best_params = {
    "max_depth": 4,
    "min_child_weight": 20,
    "gamma": 0.1,
    "subsample": 0.8,
    "colsample_bytree": 0.7,
    "reg_lambda": 10.0,
    "reg_alpha": 0.05
}

# 2) Sonra AYRI validation setinde early stopping uygula.
final_model = XGBClassifier(
    **best_params,
    objective="binary:logistic",
    eval_metric="aucpr",
    tree_method="hist",
    n_estimators=5000,
    learning_rate=0.025,
    early_stopping_rounds=150,
    random_state=42
)
```

Neden iki aşama?

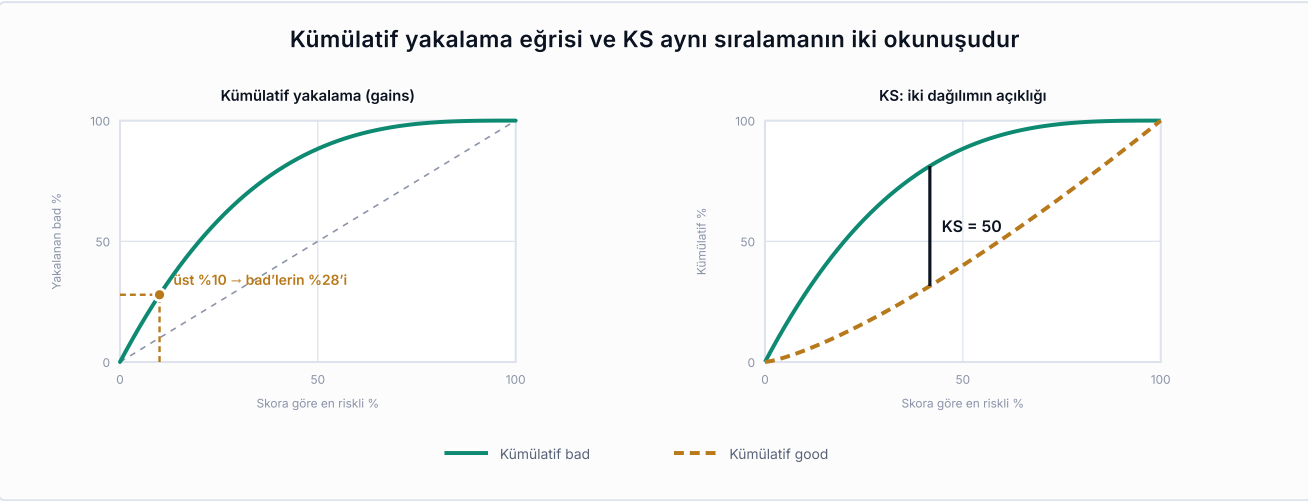
Yapı parametreleri (derinlik, kapılar, ceza, örnekleme) ile ağaç sayısı aynı aramada birlikte seçilirse, her fold farklı bir en iyi tur bulur ve sonuç yorumlanamaz hale gelir. Önce yapıyı sabitle, ardından tek bir validation setinde **yalnızca tur sayısını** seç.

Üretime çıkmadan önce

Seçilen parametreleri, best_iteration değerini, seed'i, veri sürümünü ve validasyon pencerelerini bir model dokümanında kayıt altına al. Aynı sonucu yeniden üretemiyorsan model henüz hazır değildir.

16 Ayrım gücü: KS, gains ve lift

AUC tek bir özet sayıdır ve operasyonun nerede çalışacağını söylemez. Kredi riskinde kararı asıl taşıyan şey, **skorun üst dilimlerinde ne kadar bad yakalandığıdır**.



Şekil 13. Solda gains eğrisi, sağda kümülatif dağılımlar arasındaki maksimum fark (KS).

Metrik	Ne söyler?	Kredi riskinde kullanımı
Gini / AUC	Rastgele bir bad'in rastgele bir good'dan daha riskli skorlanma olasılığı.	Modeller arası genel karşılaştırma ve onay eşiği.
KS	İyi ve kötü kümülatif dağılımları arasındaki en büyük fark.	Bankacılık pratiğinde en yaygın raporlanan ayrım ölçüsü.
Top capture	En riskli %N dilimde yakalanan bad oranı.	Kapasite N ile sınırlıysa asıl karar metriği budur.
Lift	Dilimdeki bad oranının portföy ortalamasına oranı.	Limit, teminat ve izleme kararlarının iş gerekçesi.

Hangi dilim gerçekten önemli?

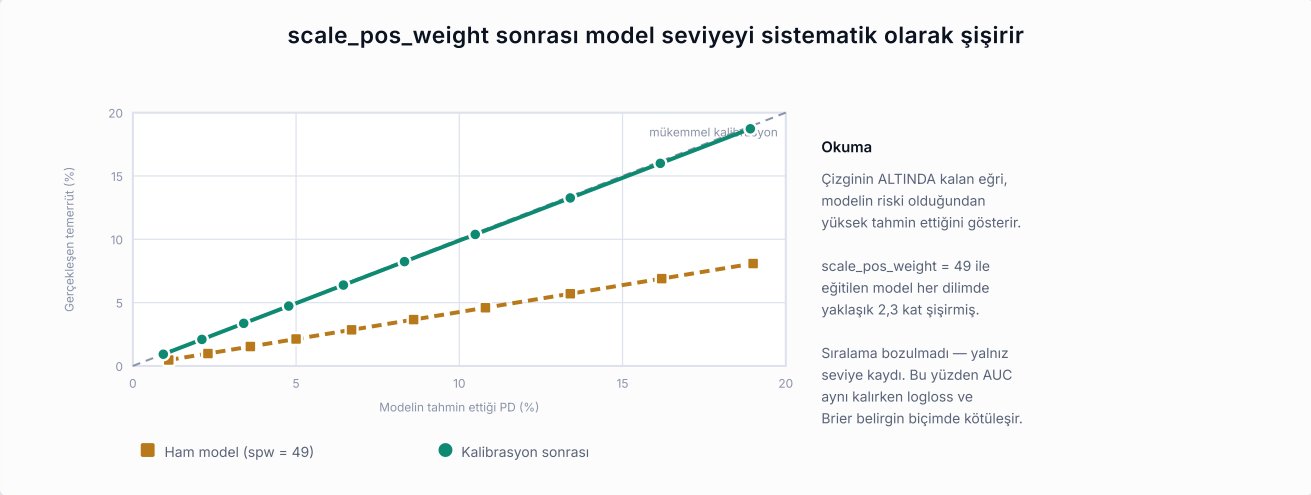
Aylık 5.000 dosya inceleyebilen bir izleme ekibi için 400.000 kişilik portföyde operasyonel kapasite yaklaşık **%1,25**'tir. Bu durumda iki modeli AUC ile değil, **en riskli %1,25 dilimdeki yakalama farkıyla** karşılaştırmak gerekir. 0,004'lük bir Gini farkı önemsizken, üst dilimde 6 puanlık yakalama farkı doğrudan önlenen zarardır.

Segment bazlı ayrımı ayrıca ölç

Toplamda güçlü bir model; yeni müşteri / mevcut müşteri, ürün tipi veya kanal kırılımında zayıf olabilir. Ayrım gücünü en az **vintage, ürün ve müşteri kıdemi** kırılımlarında raporla — tek bir toplam Gini bunu gizler.

17 Kalibrasyon ve PD

Sıralama doğru olabilir ama seviye yanlış olabilir. `predict_proba` çıktısı otomatik olarak **PD değildir**; kalibre edilmemiş bir skor karşılık, fiyatlama veya beklenen zarar hesabında doğrudan kullanılamaz.



Şekil 14. Kalibrasyon eğrisi; 45° çizgiden sapma seviye hatasını gösterir.

Neden bozulur?

`scale_pos_weight` pozitif sınıfı yapay olarak çoğaltır; model artık gerçek bad oranını değil, **yeniden ağırlıklandırılmış** bir evreni öğrenir. Undersampling, aşırı `max_depth` ve düşük `reg_lambda` de skorları uçlara iterek kalibrasyonu bozar.

Yöntem	Ne yapar?	Ne zaman
Platt / lojistik	Skorun logitine düşük parametreli bir düzeltme uygular.	Sapma düzgün ve tek yönlüyse; küçük veride güvenli.
Isotonic	Monoton, parametresiz bir eşleme öğrenir.	Sapma düzensizse; bol veri ister, aşırı uyum riski var.
Offset düzeltmesi	Yalnız kesişimi kaydırıp merkezî eğilimi hedefe oturtur.	Sıralama iyi, yalnız seviye kaymışsa.
Ağırlıksız yeniden eğitim	scale_pos_weight = 1 ile modeli yeniden kurar.	PD asıl çıktıysa; ayrım kaybı ölçülerek.

Kalibrasyon ayrı veride yapılır

Kalibrasyon eğrisini early stopping'in kullandığı validation setinde öğrenirsen aynı veriye iki kez uymuş olursun. Kalibrasyonu **ayrı bir dilimde** öğren, OOT'de doğrula. Sağlaması: gözlenen ve beklenen temerrüt oranlarının dilim bazında binom testi.

Nokta-zaman mı, çevrim-boyu mu?

Beklenen zarar hesapları makro koşullara duyarlı **nokta-zaman (PIT)** PD ister; sermaye tarafında ise **çevrim-boyu (TTC)** yaklaşımı beklenir. Aynı XGBoost skoru her iki amaca da hizmet edebilir — fark modelde değil, **kalibrasyon katmanındadır**. Hangisini ürettiğini model dokümanında açıkça yaz.

18 Stabilite ve izleme

Kredi riski modeli bir kez ölçülüp bitmez; portföy, ürün ve makro koşullar değiştikçe hem **girdi dağılımı** hem de **performans** kayar.

Şekil 15. Dilim payları arasındaki fark büyüdükçe PSI yükselir.

```
PSI = Σ (güncel% - referans%) · ln(güncel% ÷ referans%)
```

Gösterge	Ne izler?	Yaygın eşik
PSI (skor)	Model çıktısının dağılımındaki kayma.	< 0,10 stabil · 0,10–0,25 izle · > 0,25 incele
CSI (değişken)	Tek tek girdilerin kayması; PSI'ın kaynağını bulur.	Aynı ölçek; en çok katkı yapan 5 değişkeni raporla
Gini takibi	Ayrım gücünün dönemler arası seyri.	Geliştirmeye göre bağıl düşüş, ör. > %20
Vintage analizi	Her açılış kohortunun olgunlaşma eğrisi.	Kohortlar arası eğri kayması
Override oranı	Kullanıcının model kararını ezme sıklığı.	Artış, güven kaybının erken işareti

Yüksek PSI tek başına "model bozuldu" demek değildir

Portföy bilinçli olarak değiştiyse — yeni ürün, farklı kanal, sıkılaşan kabul politikası — skor dağılımının kayması **beklenen** sonuçtur. Önce PSI'ı CSI ile ayrıştır: kayma politikadan mı, veri kalitesinden mi, yoksa gerçek risk profilinden mi geliyor? Yeniden eğitim kararı bu ayrım yapılmadan verilmemelidir.

Tuning ile stabilite ilişkisi

Derin ağaçlar ve düşük `min_child_weight`, küçük segmentlere aşırı duyarlı skorlar üretir; bu skorlar dağılım kaydığında hızla bozulur. Stabilite kritikse **sığ ağaç + yüksek min_child_weight + güçlü L2** kombinasyonu, birkaç puan Gini pahasına çok daha uzun ömürlü bir model verir.

19 Yönetişim ve açıklanabilirlik

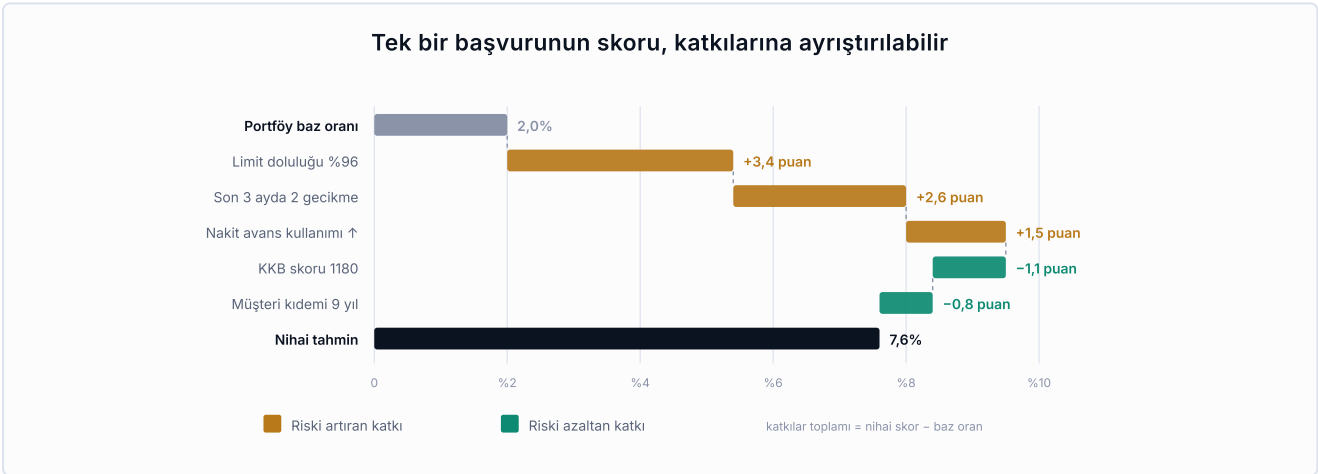
Kredi riskinde bir model yalnız iyi çalışmak zorunda değildir; **gerekçelendirilebilir, denetlenebilir ve savunulabilir** olmak zorundadır. Bu, bazı hiperparametre tercihlerini doğrudan kısıtlar.

Hiperparametrelerin yönetişim karşılığı

Tercih	Model riski tarafındaki karşılığı
max_depth ≤ 4	Etkileşimler anlatılabilir kalır; açıklamalar daha kararlı çıkar.
monotone_constraints	"KKB skoru arttıkça risk artamaz" gibi iş kurallarını modelde garanti eder.
interaction_constraints	Değişken ailelerinin birbirine karışmasını engeller; leakage şüphesini sınırlar.
Yüksek min_child_weight	Az gözleme dayanan uç skorları engeller; bireysel itirazlarda savunulabilirlik artar.
Sabit random_state	Yeniden üretilebilirlik; bağımsız validasyon biriminin ön koşulu.

Açıklanabilirlik katmanı

- **Global:** SHAP özeti ve permutation importance birlikte okunmalı; gain tabanlı importance korelasyonlu değişkenlerde yanıltıcıdır.
- **Yerel:** Reddedilen başvuruya verilecek gerekçe, tek gözlem için SHAP katkılarının en büyük birkaç kalemine dayandırılabilir.
- **Tutarlılık:** Küçük bir girdi değişikliği skoru büyük ölçüde oynatıyorsa model açıklanabilir değildir; kapasiteyi düşür.



Şekil 16. Portföy baz oranından başlayıp değişken katkılarıyla nihai skora ulaşan ayrışım.

Ayrımcı değişkenler ve vekilleri

Kullanılamayacak bir özniteliği modelden çıkarmak yetmez; onun **vekili** (proxy) olan değişkenler aynı bilgiyi taşıyabilir. Değişken listesini yalnız istatistiksel katkıya göre değil, **anlamına göre** de gözden geçir. XGBoost bu ayrımı kendiliğinden yapmaz — güçlü bir vekil bulursa memnuniyetle kullanır.

Şampiyon – meydan okuyan

XGBoost'u üretimdeki lojistik regresyon skorkartının yerine değil, **önce yanında** çalıştır. Aynı OOT üzerinde ayrım, kalibrasyon, stabilite ve açıklanabilirlik başlıklarında karşılaştır. Karar yalnız performansa değil, **farkın operasyonel değerine** dayanmalıdır.

20 Yaygın hatalar ve hızlı teşhis

Belirtiden nedene, nedenden eyleme giden bir okuma tablosu.

Belirti	Muhtemel neden	Eylem
Train AUC çok yüksek, OOT düşüyor	Aşırı kapasite / leakage	max_depth düşür; min_child_weight, gamma, L1/L2 artır; veri sızıntısını kontrol et
Train ve validation ikisi de zayıf	Underfit veya zayıf değişken	max_depth artır; kısıtları gevşet; feature engineering ve hedef tanımını incele
Model 20–30 turda duruyor	learning_rate yüksek veya validation gürültülü	eta düşür; early stopping patience artır; tarih penceresini kontrol et
Tahminler 0 ve 1'e yığılıyor	Aşırı yaprak skorları / dengesizlik	reg_lambda artır; max_delta_step dene; max_depth düşür
AUC iyi, logloss kötü	Sıralama iyi, kalibrasyon kötü	scale_pos_weight etkisini incele; kalibre et; base rate driftini kontrol et
Her çalışmada sonuç değişiyor	Rastgele örnekleme / paralellik	random_state sabitle; seed stabilitesini raporla; veri sırasını kontrol et
GridSearch bilgisayarı kilitliyor	İç ve dış paralellik çakışıyor	CV n_jobs veya XGBoost n_jobs değerlerinden birini düşür
GPU, CPU'dan yavaş	Veri küçük veya aktarım maliyeti yüksek	hist + CPU ile benchmark yap; dönüşüm maliyetini dahil et
Feature importance tek değişkende yoğun	Korelasyon / dominant proxy	colsample düşür; leakage ve proxy kontrolü yap; SHAP ve permutation ile doğrula
Bad recall düşük	Dengesizlik / eşik	scale_pos_weight ve objective sonrası karar eşiğini maliyetle optimize et

Son kontrol listesi

En iyi model yalnızca en yüksek Gini değildir. **OOT stabilitesi, kalibrasyon, iş maliyeti, açıklanabilirlik, segment adaleti, veri bulunabilirliği ve üretim süresi** birlikte değerlendirilmelidir.

Hızlı karar kartı

Ne istiyorsun → hangi parametre → hangi yön.

Ne istiyorsun?	Hangi parametre?	Hangi yön?
Daha basit ağaç	max_depth / max_leaves	Azalt
Küçük yaprakları engelle	min_child_weight	Artır
Düşük kazançlı splitleri engelle	gamma	Artır
Daha yavaş ve kontrollü öğren	learning_rate	Azalt; n_estimators artır
Satır bazlı rastgelelik	subsample	1'den aşağı indir
Kolon bazlı rastgelelik	colsample_bytree	1'den aşağı indir
Yaprak skorlarını yumuşat	reg_lambda	Artır
Küçük yaprak katkılarını sıfırla	reg_alpha	Artır
Nadir bad sınıfını güçlendir	scale_pos_weight	Artır; kalibrasyonu izle
Aşırı lojistik güncellemeyi sınırla	max_delta_step	0'dan 1–10'a dene
Daha hızlı ağaç kur	tree_method	hist
GPU kullan	device	cuda
Histogram hassasiyetini artır	max_bin	Artır; süre ve belleği izle

Tek cümlelik zihinsel model

max_depth ne öğrenebileceğini, **gamma / min_child_weight** neyin öğrenilmeye değer olduğunu, **learning_rate** ne kadar hızlı öğreneceğini, düzenlileştirme ve örnekleme ise öğrendiğine ne kadar güveneceğini belirler.

21 Kaynaklar

Tüm bağlantılar XGBoost 3.3.0 kararlı dokümantasyonuna aittir.

1. XGBoost Parameters — XGBoost 3.3.0 Documentation
   `xgboost.readthedocs.io/en/stable/parameter.html`
2. Notes on Parameter Tuning — XGBoost 3.3.0 Documentation
   `xgboost.readthedocs.io/en/stable/tutorials/param_tuning.html`
3. Using the Scikit-Learn Estimator Interface — XGBoost 3.3.0 Documentation
   `xgboost.readthedocs.io/en/stable/python/sklearn_estimator.html`
4. Tree Methods — XGBoost 3.3.0 Documentation
   `xgboost.readthedocs.io/en/stable/treemethod.html`
5. Introduction to Boosted Trees — XGBoost 3.3.0 Documentation
   `xgboost.readthedocs.io/en/stable/tutorials/model.html`
6. Python Package Introduction — XGBoost 3.3.0 Documentation
   `xgboost.readthedocs.io/en/stable/python/python_intro.html`

Kapsam notu

XGBoost çok sayıda özel objective, ranking, survival, distributed training ve callback parametresi içerir. Bu rehber, **XGBClassifier ile tabular ikili sınıflandırmada** en yüksek pratik etkiye sahip parametrelere odaklanır.

Özet

Hiperparametre tuning bir arama probleminden çok bir **tasarım** problemidir. Doğru validasyon kurgusu, doğru metrik ve doğru iş hedefi tanımlanmadan yapılan en geniş grid araması bile yanıltıcı sonuç üretir. Önce neyi ölçtüğünü, sonra neyi ayarladığını netleştir.

Doküman tarihi: 1 Ağustos 2026 · BCDataWorks Çalışma Notları № 02

— ÇALIŞMA SONU —