

Satış Tahmininde En Büyük Düşmanınız Model Değil — Veri Sızıntısı

Rossmann'ın 1 milyon satırlık günlük cirosunu sızıntısız bir hatla tahmin ederken öğrendiklerim: kilitli takvim doğrulaması, log dönüşümü, rekursif lag ve %12,14 RMSPE'ye giden ensemble.

Yasin · softITo Veri Bilimi Bitirme Projesi · ~12 dk okuma

Perakendede yarın kaç kutu satılacağını bilmek neden bu kadar kritik?

Bir mağaza zincirinde stok fazla gelirse raflar şişer, maliyet artar. Az gelirse müşteri boş rafa bakar ve rakibe gider. Satış tahmini bu yüzden “güzel bir Kaggle skoru”ndan önce operasyonel bir karar aracıdır: kaç personel planlanır, hangi mağazaya ne kadar ürün gönderilir, promosyon bütçesi nereye akıtılır.

Ben de softITo bitirme projemde Avrupa'daki **Rossmann** mağaza zincirinin günlük satışlarını tahmin etmeye koyuldum. Eldeki tablo büyüktü: yaklaşık **1 milyon satır** eğitim verisi, **1.115 mağaza**, 6 haftalık bir gelecek penceresi. Yanına hava durumu ve Google Trend sinyallerini de ekledim.

İlk denemede skorum “fena değil” görünüyordu. Sonra doğrulama stratejime baktım ve midem sıkıştı: modeli, gerçek hayatta asla sahip olamayacağım bilgilere bakarak ölçüyordum. Yani skorum iyiye bunun sebebi modelin zekâsı değil, **veri sızıntısı** olabilirdi.

“Zaman serisinde rastgele train_test_split kullanmak, sınavda cevap anahtarını cebinde taşıyıp “ben çok çalıştım” demeye benzer.”

Bu yazıda o hatayı nasıl yakaladığımı, dağılımı nasıl düzelttiğimi, lag özellikleriyle geleceği nasıl (dürüstçe) simüle ettiğimi ve sonunda XGBoost + LightGBM + Optuna ensemble'ıyla doğrulamada **%12,14 RMSPE** skoruna nasıl geldiğimi anlatacağım.

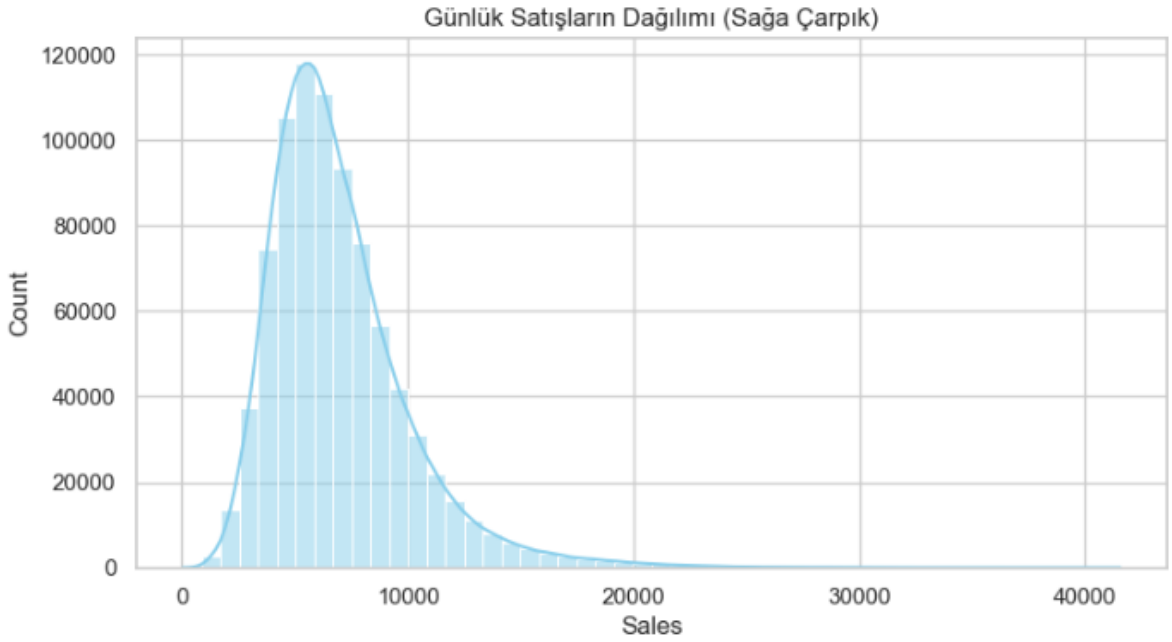
Sahne kurulumu: neyle çalıştım?

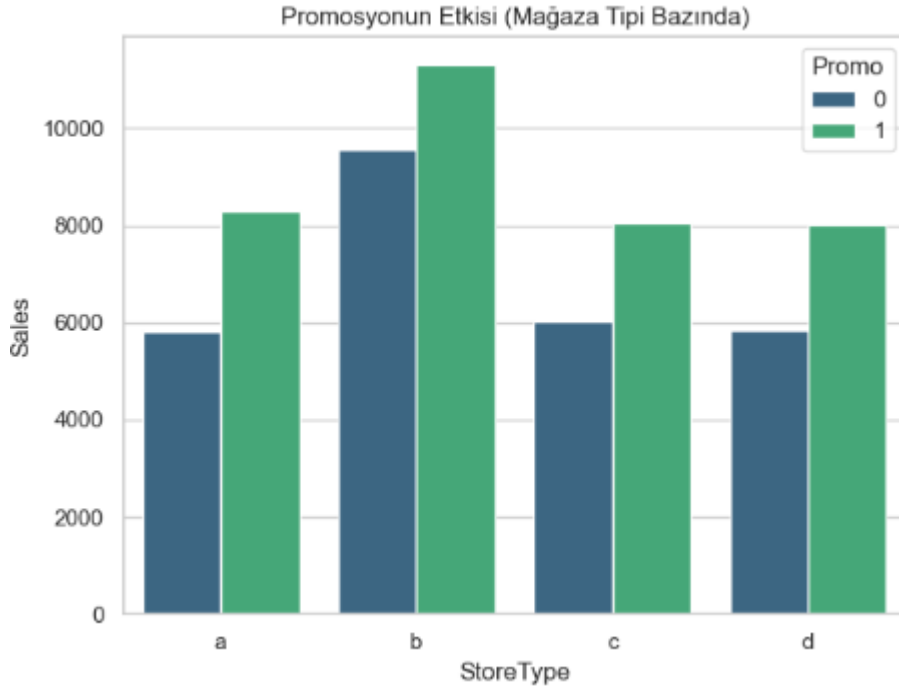
Veri hattı üç Jupyter not defterine bölündü: yükleme/birleştirme, EDA, modelleme. Ham kaynaklar şunlardı:

- `train.csv` — 1.017.209 satır, mağaza-gün bazlı satış
- `test.csv` — 41.088 satır, 2015-08-01 → 2015-09-17 (kilitli dönem)
- `store.csv` — mağaza tipi, rakip mesafe, Promo2 takvimi
- `weather.csv` + `googletrend.csv` — eyalet hava durumu ve arama trendi

Birleştirirken sessiz satır şişmesini önlemek için her merge adımında `validate='many_to_one'` kullandım. Küçük bir parametre; büyük bir güvence. Çünkü yanlış birleştirme de sızıntı kadar tehlikelidir — sadece daha sinsidir.

EDA tarafında birkaç gerçek hızlıca yüzeye çıktı: verinin **%17'si kapalı mağaza günü** (satış tanım gereği 0), kısa vadeli promosyon ortalama ciroyu yaklaşık **%39** artırıyor, Noel dönemi net bir zirve, mağazalar arası ciro farkı 8 kata kadar çıkabiliyor. Modelin “tek bir ortalama mağaza” varsayımıyla işi olmayacağı belliydi.





1) Veri sızıntısı: skorunuzu şişiren görünmez hata

İlk sürümde veriyi Store + Date sırasına dizip satır bazında %85/%15 bölmüştüm. İki problem birden vardı:

- **Doğrulama kümesinde eğitimde hiç görülmemiş mağazalar** çıkabiliyordu — bu, “yeni mağaza genellemesi” testine dönüşüyordu, asıl göreve değil.
- Daha kötüsü: eğitim ve doğrulama **takvimsel olarak çakışıyordu**. Model, aynı haftanın komşu günlerini “gelecek” diye görüp ezberliyordu.

Çözüm: kilitli takvim doğrulaması

Stratejiyi baştan yazdım. Son **6 haftayı** doğrulama penceresi yaptım; önceki tüm geçmiş eğitim oldu. Neden 6 hafta? Çünkü gerçek test dönemi de tam 6 haftaydı. Doğrulama, testi *taklit etmeliydi* — daha kısa veya daha uzun bir pencere, kendini kandırmak olurdu.

Test setine (41.088 satır) gelince kuralım netti: model seçimi, hiperparametre, eşik... hiçbirisi için açılmadı. Yalnızca nihai submission.csv üretiminde bir kez skorlandı. “Kilitli holdout” dediğimiz şey budur.

Tek pencereye fazla güvenmemek için 3 katlı **walk-forward (kayan pencere)** da koşturdum. Ortalama CV RMSPE **%12,60** (std 0,40) çıktı. Düşük sapma, modelin “şanslı bir 6 haftaya” yaslanmadığını gösteriyordu.

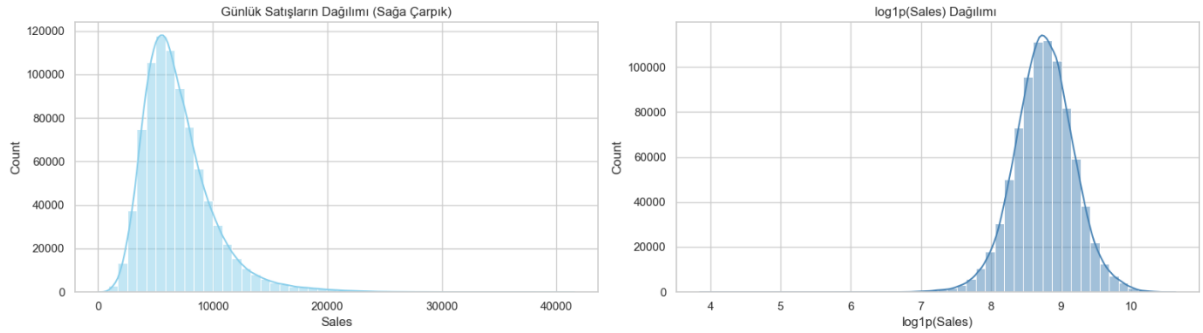
2) Logaritmik dönüşüm: hedefi modelin anlayacağı dile çevirmek

Perakende cirosu nadiren “güzel bir çan eğrisi”dir. Az sayıda çok yüksek satış günü dağılımı sağa çeker. Bizim veride çarpıklık katsayısı **1,59** idi — klasik sağa çarpık tablo.

Yarışmanın resmi metriği **RMSPE** (Root Mean Squared Percentage Error). Bu metrik *görelî* hata ölçer: 10.000 €’luk mağazada 500 € sapmak ile 2.000 €’luk mağazada 500 € sapmak aynı şey değildir. Ham satış üzerinde MSE optimize eden bir model, büyük ciroları memnun etmeye çalışır; yüzdesel hatayı ikinci plana atar.

Bu yüzden hedefe $\log_{1p}(\text{Sales})$ uyguladım. Dönüşüm sonrası çarpıklık **-0,11**’e indi. Tahminleri geri çevirirken $\exp_{m1}$ kullandım.

Etki sayıya dökülünce tartışma bitti: aynı XGBoost mimarisinde ham hedef **%14,91 RMSPE**, log hedef **%12,32 RMSPE**. Yani tek bir dönüşüm, yaklaşık **2,6 puan** kazandırdı. Fancy bir mimari değil — doğru kayıp yüzeyi.



Küçük ama kritik bir operasyonel kural daha: kapalı günleri ($\text{Open} == 0$) modele “öğrenilecek örnek” diye vermedim. Eğitimden izole ettim; tahmin anında doğrudan **0** yazdım. Modelin kapalı bir pazarı “belki 3.000 €” diye tahmin etmesine izin vermek, gereksiz gürültüdür.

3) Rekursif lag: dünün tahminini bugünün verisi yapmak

Bir mağazanın en güçlü sinyali çoğu zaman kendi geçmişidir. Bu yüzden Sales_Lag7 , Sales_Lag14 ve Sales_Roll28 (28 günlük hareketli ortalama) özelliklerini ürettim.

Ama lag’ler iki tuzakla gelir:

- **Sızıntı:** Gecikme yalnızca kesin geçmişten üretilmeli. İlk 28 günde değer yoksa medyanla doldurmak cazip görünür — o da sızıntıdır. Ben o satırları eğitimden düşüm.

- **Gelecekte gerçek satış yok:** Test döneminde “dün ne sattık?” sorusunun cevabı henüz yok. O zaman ne yapılır?

Rekursif simülasyon

Cevap: **gün gün ilerlemek**. Modelin bir önceki gün için ürettiği tahmini, bugünün lag özelliği olarak geri beslersiniz. Kapalı günleri takvimden bildiğimiz için simülasyona doğrudan o yazarız. Aynı simülasyonu doğrulama penceresinde de koşturdum — aksi halde lag modeli haksız yere “kolay puan” toplardı.

Dürüst skor neydi? Rekursif lag-XGBoost **%15,38 RMSPE**. Diğer modellere göre daha kötü — ve bu *beklenen* bir sonuç. Çünkü hata birikir: yanlış tahmin, sonraki günün özelliğini de bozar. Yine de ensemble adayına dahil ettim; metodoloji olarak “denemeden elemek” istemedim.

Model tarafı: XGBoost, LightGBM, Optuna ve ensemble

Ana ailem gradyan artırmalı ağaçlardı: **XGBoost** ve **LightGBM**. Aşırı öğrenmeye karşı L1/L2 cezası, derinlik sınırı, satır/kolon alt-örnekleme ve erken durdurma kullandım. “Parametreleri elle kurcalama” aşamasını da kısa tuttum — asıl işi Optuna’ya verdim.

Optuna ile sistematik arama

Optuna’nın TPE algoritmasıyla `learning_rate`, `max_depth`, `subsample`, `colsample_bytree` ve `reg_lambda` gibi hiperparametreleri taradım. Arama maliyeti için ağaç bütçesini kısık tuttum; bulunan en iyi parametrelerle daha uzun bir final eğitimi koşturdum.

Sonuç: Optuna-XGBoost doğrulamada **%12,26 RMSPE**. Log’lu temel XGBoost’a (%12,32) göre marjinal ama istikrarlı bir iyileşme. Büyük sıçramalar her zaman mimariden gelmez; bazen daha iyi bir arama uzayından gelir.

Ensemble: modellerin birbirinin açığını kapatması

Dört adayı aynı doğrulama penceresinde tarttım: XGBoost + log, LightGBM, Optuna-XGBoost, Lag-XGBoost. Ağırlıkları 0,05 adımlı bir ızgarada, RMSPE’yi minimize edecek şekilde aradım.

Kazanan karışım:

- **XGBoost Optuna** → %50
- **XGBoost + Log** → %31

- **LightGBM** → %19
- **Lag-XGBoost (rekursif)** → %0

Lag modelinin ağırlığının 0 çıkması enteresan bir ders: her “ileri seviye fikir” ensemble’a katkı vermez. Ama aday listesine koymak yanlış değildi; veri karar verdi, ben değil.

Skor tablosu (doğrulama)

- **Ensemble** — RMSE 861,2 | R^2 0,9200 | **%12,14 RMSPE**
- XGBoost Optuna — %12,26
- XGBoost + Log — %12,32
- LightGBM + Log — %12,51
- XGBoost Baseline (ham hedef) — %14,91
- XGBoost Lag (rekursif) — %15,38

Baseline’a göre ensemble, RMSPE’de yaklaşık **2,8 puan (~%18,6 görelî)** iyileşme sağladı. Test döneminde de aynı ensemble mantığıyla `submission.csv` üretildi; kapalı 5.984 güne doğrudan 0 atandı.

	RMSE	MAE	R2	RMSPE
model				
XGBoost Baseline	960.07	681.88	0.90	14.91
XGBoost + Log Hedef	873.92	597.89	0.92	12.32
LightGBM + Log Hedef	895.63	614.29	0.91	12.51

Ne işe yaradı? Kıssadan hisseler

Bu projede “en havalı model” yarışını kazanmadım. Daha sıkıcı ama daha değerli bir şeyi kazandım: **kendini kandırmayan bir değerlendirme disiplini**.

Öne çıkan çıkarımlarım:

- **Önce doğrulamayı kilitleyin.** Zaman serisinde rastgele split, iyimser bir yalandır.
- **Metriğinize uygun hedef kullanın.** RMSPE görelîyse, log dönüşümü çoğu zaman bedava puandır.
- **Lag’i dürüstçe ölçün.** Gelecekte gerçek geçmiş yoksa rekursif simülasyon şarttır.

- **Ensemble sihir deęil, eřitlilik**. Farklı hata rntlerini birleřtirmek, tek modeli zorlamaktan daha ucuz gelebilir.
- **Basit kuralları kmsemeyin**. Kapalı gne o yazmak, fancy bir zellikten daha fazla skor kurtarabilir.

“İyi bir satış tahmin modeli, geleceęi bilen model deęildir. Geleceęi bilmiyormuř gibi drste llen modeldir.”

İleri adımlar olarak CatBoost, stacking, Optuna aramasını CV ortalamasına baęlamak ve maęaza bazlı hedef kodlaması hl masada. Ama bu teslimatın omurgası hazır: sızıntı kontroll hat, kilitli test, gerek bir `submission.csv`.

Eęer siz de zaman serisiyle uęraşıyorsanız, bir sonraki modelinizi eęitmeden nce řunu sorun: **“Bu skoru gerek hayatta da alabilir miyim?”** Cevabınız net deęilse, nce doęrulamayı dzeltin. Model sonra gelir.