

Yemek Sipariřim Kaç Dakikada Gelecek?

Makine öğrenmesiyle teslimat süresi tahmini: bir bitirme projesinin hikâyesi

Şevval OK · Ağustos 2026 · 8 dakikalık okuma

Bir akşam yemek sipariři verdiğinizi düşünün. Uygulama size “tahmini teslimat 30 dakika” diyor. Kırk beşinci dakikada hâlâ kapıda kimse yok ve siz uygulamayı beşinci kez açmış durumdasınız. Herkesin başına gelmiştir.

Bu küçük sinir bozukluğunun arkasında aslında güzel bir veri bilimi problemi var: Bir sipariřin ne kadar sürede teslim edileceğini, sipariř verildiği anda bilebilir miyiz? Bitirme projemde tam olarak bunu yapmaya çalıştım. Bu yazıda projeyi baştan sona, teknik terimlere boğulmadan anlatacağım.

Problem neye benziyor?

Teslimat süresini etkileyen onlarca şey var: mesafe, trafik, hava durumu, kuryenin deneyimi, sipariřin verildiği saat, kuryenin aynı anda kaç sipariř taşıdığı... Bunların hepsini “mesafeyi hıza böl” gibi basit bir formülle çözemezsiniz. Çünkü aradaki ilişkiler düz bir çizgi değil.

Örneğin 3 kilometrelik bir yol sakın bir öğleden sonra 10 dakika sürerken, cuma akşamı yağmurda 25 dakikaya çıkabiliyor. İşte bu yüzden kural yazmak yerine, geçmiş sipariřlerden öğrenen bir model kurmak çok daha mantıklı.

Tahmin etmeye çalıştığım şey bir kategori değil, bir sayı: kaç dakika. Bu da problemi doğrudan bir regresyon problemi yapıyor.

Sınıflandırma “hangisi?” sorusuna cevap verir, regresyon ise “ne kadar?” sorusuna. Biz “ne kadar dakika?” diye soruyoruz.

Elimdeki veri

Gerçek bir yemek dağıtım platformuna ait sipariř kayıtlarıyla çalıştım. Her satır tek bir sipariři temsil ediyor ve içinde şunlar var:

- Kurye bilgileri: yaşı, müşteri puanı, kullandığı araç ve aracın durumu
- Konum bilgileri: restoranın ve teslimat adresinin enlem-boylam koordinatları

- Çevresel koşullar: hava durumu, trafik yoğunluğu, şehir tipi, festival günü olup olmadığı
- Sipariş bilgileri: tarih, sipariş saati, kuryenin siparişi aldığı saat, sipariş türü
- Ve tabii hedefimiz: teslimatın kaç dakika sürdüğü

İlk bakışta “harika, her şey hazır” diye düşünüyor insan. Sonra veriye biraz yaklaşıncaya işin rengi değişiyor.

Verinin hiç de temiz olmayan hâli

Veri bilimiyle ilgili en çok tekrarlanan cümle, işi yaptıktan sonra çok daha anlamlı geliyor: zamanın büyük kısmı veriyi temizlemekle geçiyor.

Karşıma çıkan sorunlardan birkaçı:

- Teslimat süresi sütununda değerler “(min) 24” şeklinde yazılmıştı. Yani sayı değil, metindi.
- Hava durumu değerlerinin başında gereksiz bir “conditions ” eki vardı.
- Eksik değerlerin bir kısmı boş hücre değil, düz metin olarak yazılmış “NaN” ifadesiydi. Pandas bunları eksik saymıyor, gayet mutlu bir şekilde metin kabul ediyordu.
- Bazı koordinatlar sıfırdı. Yani restoran teorik olarak okyanusun ortasındaydı.

Bunları düzeltmek çok karmaşık bir iş değil ama fark etmezseniz modeliniz saçmalıyor. Örneğin hedef değişkeni sayıya çevirmek şu kadar basitti:

```
df["Time_taken(min)"] = (df["Time_taken(min)"]
                          .str.replace("(min)", "", regex=False)
                          .str.strip()
                          .astype(int))
```

Eksik değerlerde şöyle bir yol izledim: sayısal sütunlarda medyan, kategorik sütunlarda ise en sık görülen değer. Medyanı tercih etmemin sebebi, ortalamanın uç değerlerden kolayca etkilenmesi. Tek bir 60 yaşındaki kurye kaydı ortalamayı yukarı çekebilir ama medyanı pek sarsmaz.

İşin en keyifli kısmı: yeni özellikler üretmek

Veri setinde restoranın ve müşterinin koordinatları ayrı ayrı duruyordu. Model bu dört sayıya bakıp “demek ki arada 4 kilometre var” diyemez; enlem ve boylam onun için sadece iki sayıdır.

Bu yüzden Haversine formülüyle iki nokta arasındaki kuş uçuşu mesafeyi hesapladım. Formül, dünyanın küre olduğunu hesaba katarak iki koordinat arasındaki uzaklığı kilometre cinsinden veriyor.

```
def haversine(lat1, lon1, lat2, lon2):
    R = 6371 # Dünya'nın yarıçapı (km)
    lat1, lon1, lat2, lon2 = map(np.radians, [lat1, lon1, lat2, lon2])
    dlat, dlon = lat2 - lat1, lon2 - lon1
    a = (np.sin(dlat / 2) ** 2 +
          np.cos(lat1) * np.cos(lat2) * np.sin(dlon / 2) ** 2)
    return R * 2 * np.arcsin(np.sqrt(a))
```

Bu tek satırlık yeni sütun, modelin en çok işine yarayan bilgilerden biri oldu. Dört anlamsız koordinattan, teslimat süresiyle doğrudan ilişkili tek bir anlamlı sayı çıkardık.

Sonra zaman bilgilerine döndüm. Sipariş saati ile kuryenin siparişi restorandan aldığı saat arasındaki fark, aslında restoranın hazırlık süresi. Toplam sürenin epey büyük bir parçası. Bunun dışında tarihten haftanın günü, hafta sonu olup olmadığı ve saatten “yoğun saat mi” bilgisini türettim.

Özellik mühendisliği bana biraz şuna benziyor: modele cevabı vermiyorsunuz ama doğru soruyu sormasını kolaylaştırıyorsunuz.

Neden XGBoost ve LightGBM?

Bu tür tablo biçimindeki verilerde, karmaşık derin öğrenme modelleri genelde gerekmiyor. Gradient boosting tabanlı ağaç modelleri çoğu zaman hem daha hızlı hem daha başarılı oluyor.

Gradient boosting'in mantığı aslında oldukça sevimli: tek bir mükemmel model kurmaya çalışmıyor. Önce basit bir karar ağacı kuruyor, hatalarına bakıyor, sonra bir sonraki ağacı bu hataları düzeltmek için eğitiyor. Sonra bir sonraki, sonra bir sonraki... Yüzlerce küçük düzeltme üst üste binince ortaya oldukça güçlü bir tahminci çıkıyor.

İki modeli de denedim çünkü aralarında ufak ama pratikte önemli farklar var:

- XGBoost ağacı seviye seviye büyütüyor; biraz daha yavaş ama parametre ayarlarına karşı daha toleranslı.
- LightGBM en çok kazanç sağlayacak yaprağı seçip oradan büyüyor; belirgin şekilde hızlı ve az bellek kullanıyor, ama ayarları yanlış yaparsanız kolayca ezberleyebiliyor.

Kod tarafı sürprizsiz. İkisi de scikit-learn'e alışkın biri için tanıdık geliyor:

```
from xgboost import XGBRegressor
from lightgbm import LGBMRegressor

xgb_model = XGBRegressor(n_estimators=..., learning_rate=...,
                          max_depth=..., random_state=42)
xgb_model.fit(X_train, y_train)

lgbm_model = LGBMRegressor(n_estimators=..., learning_rate=...,
                            max_depth=..., random_state=42)
lgbm_model.fit(X_train, y_train)
```

Model başarılı mı, nereden anlıyoruz?

Veriyi %80 eğitim, %20 test olarak ayırdım. Model test verisini eğitim sırasında hiç görmüyor; böylece “ezberledi mi yoksa gerçekten öğrendi mi” sorusuna dürüst bir cevap alabiliyoruz.

Dört metriğe baktım ama en sevdiğim MAE, çünkü yorumlaması çok kolay:

- MAE: Model ortalama kaç dakika yanılıyor. “Ortalama 4 dakika sapma var” demek gibi.
- RMSE: Büyük hataları daha ağır cezalandırıyor. MAE düşük ama RMSE yüksekse, model genelde iyi tahmin ediyor ama bazı siparişlerde fena hâlde yanılıyor demektir.
- R^2 : Teslimat süresindeki değişimin ne kadarını açıklayabildiğimizi gösteriyor. 1'e yaklaştıkça iyi.

Bu problemde RMSE'ye ayrıca dikkat etmek gerekiyor. Çünkü müşteri açısından 3 dakikalık sapma tolere edilebilir, 25 dakikalık sapma ise doğrudan kötü bir deneyim.

Peki teslimat süresini en çok ne belirliyor?

Modelin güzel yanı, sadece tahmin üretmiyor; hangi bilgiye ne kadar önem verdiğini de söylüyor. Özellik önem analizinde öne çıkanlar hiç şaşırtıcı değildi ve bence bu iyiye işaret:

- Mesafe: Beklenen şekilde en belirleyici faktörlerden biri.
- Trafik yoğunluğu: Aynı mesafe, sıkışık trafikte bambaşka bir süreye dönüşüyor.
- Eş zamanlı teslimat sayısı: Kurye aynı anda üç sipariş taşıyorsa sizinki sırasını bekliyor.
- Hava koşulları ve sipariş saati: Yağmurlu bir cuma akşamı, sakin bir salı öğleninden çok farklı.

Modelin sahadaki gerçeklikle uyumlu şeyler öğrenmiş olması, sonuçlara güvenmemi kolaylaştırdı. Eğer en önemli değişken olarak “sipariş numarası” gibi bir şey çıksaydı, geri dönüp bir yerlerde hata aramam gerekirdi.

Bu gerçek hayatta nasıl kullanılır?

Aslında zaten kullanılıyor. Sipariş verdiğiniz anda uygulama restoran konumunu, adresinizi, saati ve trafik durumunu biliyor. Bunları modele verip “27 dakika” tahminini üretmesi saniyeler sürüyor.

Üstelik bu sadece müşteriye bir sayı göstermekten ibaret değil. Aynı tahmin, yoğun saatlerde kaç kurye gerektiğini planlamak, uzun süreceği öngörülen siparişleri daha deneyimli kuryelere yönlendirmek veya gecikme olacaksa müşteriyi önceden bilgilendirmek için de kullanılabilir.

...

Bu projeden ne öğrendim?

Birkaç madde hâlinde özetlemem gerekirse:

- Model seçimi düşündüğüm kadar kritik değilmiş. Asıl fark, veriyi ne kadar iyi hazırladığınızda ortaya çıkıyor.
- Tek bir metriğe bakmak yanıltıcı. MAE ile RMSE birbirinden farklı hikâyeler anlatabiliyor.
- Özellik önem analizi sadece rapor süsü değil; modelin mantıklı çalışıp çalışmadığını kontrol etmenin en pratik yolu.

- Ve en önemlisi: “neden bu değişkeni ekledim” sorusuna cevap veremiyorsanız, muhtemelen o değişkene ihtiyacınız yoktur.

Bundan sonraki adım olarak Optuna ile hiperparametre optimizasyonu yapmayı, cross validation eklemeyi ve mümkünse gerçek zamanlı trafik verisiyle beslemeyi planlıyorum. Ama şimdilik, sipariş verdiğimde ekranda gördüğüm o küçük sayının arkasında neler döndüğünü çok daha iyi biliyorum.

Buraya kadar okuduğunuz için teşekkürler. Benzer bir proje yapıyorsanız veya takıldığınız bir yer varsa, yorumlarda konuşalım.