



softITo 4. DÖNEM 2. GRUP

**MAKİNE ÖĞRENMESİ İLE TAHMİNLEME
BİTİRME PROJESİ RAPORU**

Şevval OK

Ağustos, 2026

softITo Yazılım Bilişim Akademisi, İstanbul

İÇİNDEKİLER

1. PROJE ÖZETİ
2. PROBLEM TANIMI
 - 2.1. Problemin Regresyon Problemi Olarak Ele Alınması
 - 2.2. Hedef Değişken
3. VERİ SETİNİN TANITILMASI
 - 3.1. Veri Setinin Genel Yapısı
 - 3.2. Değişkenlerin Açıklaması
4. KEŞİFSEL VERİ ANALİZİ (EDA)
 - 4.1. Veri Setinin Genel İncelenmesi
 - 4.2. Eksik Değer Analizi
 - 4.3. Tekrarlayan Kayıtların Kontrolü
 - 4.4. Aykırı Değer Analizi
5. VERİ ÖN İŞLEME
 - 5.1. Veri Temizleme
 - 5.2. Tarih ve Zaman Değişkenleri
 - 5.3. Kategorik Değişkenlerin Dönüştürülmesi
6. ÖZELLİK MÜHENDİSLİĞİ
 - 6.1. Haversine Formülü ile Mesafe Hesaplama
 - 6.2. Zaman Tabanlı Özellikler
 - 6.3. Oluşturulan Özelliklerin Özeti
7. MODELLEME SÜRECİ
8. XGBOOST MODELİ
9. LIGHTGBM MODELİ
10. MODEL PERFORMANS DEĞERLENDİRMESİ
11. XGBOOST VE LIGHTGBM KARŞILAŞTIRMASI
12. ÖZELLİK ÖNEM ANALİZİ
13. TAHMİN SONUÇLARININ GÖRSELLEŞTİRİLMESİ
14. SONUÇ VE DEĞERLENDİRME
15. GELECEKTE YAPILABİLECEK GELİŞTİRMELER
16. GENEL SONUÇ
17. KAYNAKÇA

1. PROJE ÖZETİ

Bu projede, gerçek bir yemek dağıtım platformuna ait sipariş verileri kullanılarak bir siparişin müşteriye kaç dakika içinde teslim edileceğini tahmin eden bir makine öğrenmesi modeli geliştirilmiştir. Teslimat süresi, müşteri memnuniyetini doğrudan etkileyen en kritik operasyonel göstergelerden biridir. Müşteriye verilen tahmini teslimat süresinin gerçeğe yakın olması, hem beklenti yönetimi hem de kurye planlaması açısından büyük önem taşımaktadır.

Teslimat süresi; mesafe, trafik yoğunluğu, hava koşulları, kuryenin deneyimi, sipariş tipi ve şehir yoğunluğu gibi çok sayıda değişkenin birlikte etkilendiği karmaşık bir yapıya sahiptir. Bu değişkenler arasındaki ilişkiler doğrusal olmadığı için sabit kurallara dayalı klasik hesaplama yöntemleri yetersiz kalmaktadır. Bu nedenle projede, geçmiş sipariş verilerinden öğrenerek genelleme yapabilen makine öğrenmesi yaklaşımı tercih edilmiştir.

Çalışma kapsamında veri seti önce keşifsel veri analizi (EDA) ile incelenmiş, eksik ve hatalı kayıtlar temizlenmiş, koordinat bilgilerinden Haversine formülü ile teslimat mesafesi hesaplanmış ve zaman bilgilerinden yeni özellikler türetilmiştir. Hazırlanan veri seti üzerinde iki güçlü gradient boosting algoritması olan XGBoost Regressor ve LightGBM Regressor eğitilmiş; modeller MAE, MSE, RMSE ve R^2 metrikleri ile karşılaştırılmıştır.

Projenin nihai çıktısı, sipariş anındaki bilgiler girildiğinde dakika cinsinden tahmini teslimat süresini üreten, karşılaştırmalı olarak değerlendirilmiş ve en başarılı bulunan regresyon modelidir.

2. PROBLEM TANIMI

Projenin cevap aradığı temel soru şudur:

Bir siparişin müşteriye teslim edilmesi için gereken süre, siparişe ait bilgiler kullanılarak önceden tahmin edilebilir mi?

Yemek teslimat platformlarında müşteriye sipariş anında bir tahmini teslimat süresi gösterilir. Bu sürenin gerçekleşen süreden önemli ölçüde sapması, müşteri memnuniyetsizliğine, çağrı merkezi yüküne ve platforma duyulan güvenin azalmasına yol açar. Dolayısıyla problem, geçmiş teslimatlardan öğrenilerek yeni bir sipariş için teslimat süresinin mümkün olduğunca düşük hata ile tahmin edilmesidir.

2.1. Problemin Regresyon Problemi Olarak Ele Alınması

Makine öğrenmesinde problemler, tahmin edilmek istenen hedef değişkenin yapısına göre sınıflandırılır. Hedef değişken kategorik ise sınıflandırma, sayısal ve sürekli ise regresyon problemi söz konusudur.

Bu projede tahmin edilmek istenen değer teslimat süresidir ve dakika cinsinden ölçülen sürekli bir sayısal değerdir. Örneğin bir teslimat 18 dakika, bir diğeri 26 dakika sürebilir; bu değerler önceden tanımlanmış sınırlı sayıda kategoriye değil, sürekli bir sayı eksenine dağılır. Bu nedenle problem bir regresyon problemidir ve modelin başarısı, tahmin edilen süre ile gerçekleşen süre arasındaki farkın (hata) büyüklüğü üzerinden ölçülür.

2.2. Hedef Değişken

Hedef değişken *Time_taken(min)* sütunudur. Bu sütun, siparişin kurye tarafından alınmasından müşteriye teslim edilmesine kadar geçen süreyi dakika cinsinden ifade eder. Veri setindeki diğer tüm değişkenler bağımsız değişken (özellik) olarak kullanılmıştır.

3. VERİ SETİNİN TANITILMASI

3.1. Veri Setinin Genel Yapısı

Veri seti, gerçek bir yemek dağıtım platformuna ait sipariş kayıtlarından oluşmaktadır. Her satır tek bir siparişi temsil eder ve o siparişe ait kurye bilgileri, konum bilgileri, çevresel koşullar ve teslimat süresini içerir.

Özellik	Değer
Gözlem (satır) sayısı	[Gözlem sayısı]
Değişken (sütun) sayısı	[Değişken sayısı]
Sayısal değişken sayısı	[Sayısal değişken sayısı]
Kategorik değişken sayısı	[Kategorik değişken sayısı]
Hedef değişken	Time_taken(min)
Eksik değer durumu	[Eksik değer sayısı / oranı]
Tekrarlayan kayıt sayısı	[Duplicate sayısı]

Tablo 3.1. Veri setine ait genel bilgiler

3.2. Değişkenlerin Açıklaması

Veri setinde yer alan değişkenler, veri tipleri ve anlamları aşağıdaki tabloda özetlenmiştir.

Değişken	Veri Tipi	Açıklama
ID	object	Her sipariş için benzersiz kayıt numarası
Delivery_person_ID	object	Siparişi teslim eden kuryenin kimlik bilgisi
Delivery_person_Age	float	Kuryenin yaşı
Delivery_person_Ratings	float	Kuryenin müşteri değerlendirme puanı
Restaurant_latitude	float	Restoranın enlem koordinatı
Restaurant_longitude	float	Restoranın boylam koordinatı
Delivery_location_latitude	float	Teslimat adresinin enlem koordinatı
Delivery_location_longitude	float	Teslimat adresinin boylam koordinatı
Order_Date	object	Siparişin verildiği tarih
Time_Orderd	object	Siparişin verildiği saat
Time_Order_picked	object	Siparişin kurye tarafından restorandan alındığı saat
Weatherconditions	object	Teslimat sırasındaki hava koşulu
Road_traffic_density	object	Yol trafik yoğunluğu (Low, Medium, High, Jam)
Vehicle_condition	int	Teslimat aracının durumu (0–3 arası skor)
Type_of_order	object	Sipariş türü (Snack, Meal, Drinks, Buffet)
Type_of_vehicle	object	Kullanılan araç türü (motorcycle, scooter vb.)
multiple_deliveries	float	Aynı anda taşınan ek teslimat sayısı
Festival	object	Teslimatın festival gününe denk gelip gelmediği
City	object	Şehir tipi (Urban, Metropolitan, Semi-Urban)
Time_taken(min)	int	HEDEF DEĞİŞKEN – Teslimat süresi (dakika)

Tablo 3.2. Veri setindeki değişkenlerin listesi ve açıklamaları

Not: Rapor boyunca yalnızca yukarıdaki tabloda yer alan değişkenler kullanılmış; veri setinde bulunmayan hiçbir değişken analize dâhil edilmemiştir.

4. KEŞİFSEL VERİ ANALİZİ (EDA)

Keşifsel veri analizi, modelleme öncesinde verinin yapısını, kalitesini ve değişkenler arasındaki ilişkileri anlamak amacıyla yapılan inceleme aşamasıdır. Bu aşamada tespit edilmeyen bir hata, modelin tüm sonuçlarını yanıltıcı hale getirebilir.

4.1. Veri Setinin Genel İncelenmesi

Veri seti ilk olarak temel pandas fonksiyonları ile incelenmiştir.

```
import pandas as pd

df = pd.read_csv("delivery_data.csv")

df.head()          # İlk 5 kaydı göstererek veri yapısını tanıma
df.shape           # Satır ve sütun sayısını öğrenme
df.info()          # Veri tipleri ve dolu kayıt sayılarını görme
df.describe()      # Sayısal değişkenlerin istatistiksel özeti
```

- *head()*: Verinin ilk satırlarını göstererek sütunların hangi formatta tutulduğunu ortaya koyar.
- *shape*: Veri setinin boyutunu verir; modelin ne kadar veriyle eğitileceğini gösterir.
- *info()*: Veri tiplerini ve eksik değerleri hızlıca görmeyi sağlar. Sayısal olması gereken bir sütunun object olarak görünmesi, veride hatalı karakterler bulunduğuna işaret eder.
- *describe()*: Ortalama, standart sapma, minimum ve maksimum değerleri göstererek mantıksız değerlerin (örneğin negatif süre) fark edilmesini sağlar.

4.2. Eksik Değer Analizi

Eksik değerler, model eğitimini engelleyebildiği gibi yanlış yönlendirmelere de yol açabilir. Bu nedenle her sütundaki eksik kayıt sayısı kontrol edilmiştir.

```
df.isnull().sum()
df.isnull().mean() * 100 # Eksik değerlerin yüzdesel oranı
```

Bu veri setinde eksik değerler yalnızca boş hücre olarak değil, aynı zamanda metin biçiminde "NaN" ifadesi olarak da bulunabilmektedir. Bu nedenle önce bu ifadeler gerçek eksik değere dönüştürülmüş, ardından aşağıdaki strateji uygulanmıştır:

- Eksik oranı çok düşük olan kayıtlar veri setinden çıkarılmıştır.
- *Delivery_person_Age* ve *Delivery_person_Ratings* gibi sayısal değişkenlerdeki eksikler, aykırı değerlerden daha az etkilenen medyan değeri ile doldurulmuştur.
- *Weatherconditions*, *Road_traffic_density*, *multiple_deliveries* ve *Festival* gibi kategorik değişkenlerdeki eksikler, en sık görülen değer (mod) ile tamamlanmıştır.

Eksik değerlerin doğrudan silinmesi yerine doldurulması tercih edilen durumlarda, veri kaybını en aza indirmek amaçlanmıştır.

4.3. Tekrarlayan Kayıtların Kontrolü

Aynı siparişin veri setinde birden fazla kez yer alması, modelin bazı örneklerle gereğinden fazla ağırlık vermesine ve performans metriklerinin olduğundan iyi görünmesine neden olur.

```
df.duplicated().sum()
df = df.drop_duplicates()
```

Yapılan kontrolde tespit edilen [Duplicate sayısı] adet tekrarlayan kayıt veri setinden çıkarılmıştır.

4.4. Aykırı Değer Analizi

Aykırı değerler, diğer gözlemlerden belirgin şekilde uzak olan uç değerlerdir. Bu projede özellikle Delivery_person_Age, Delivery_person_Ratings, hesaplanan teslimat mesafesi ve hedef değişken Time_taken(min) üzerinde aykırı değer incelemesi yapılmıştır.

Aykırı değerlerin tespiti için kutu grafiği (boxplot) ve IQR (Çeyrekler Arası Açıklık) yöntemi kullanılmıştır.

```
Q1 = df["Time_taken(min)"].quantile(0.25)
Q3 = df["Time_taken(min)"].quantile(0.75)
IQR = Q3 - Q1

alt_sinir = Q1 - 1.5 * IQR
ust_sinir = Q3 + 1.5 * IQR

aykiri = df[(df["Time_taken(min)"] < alt_sinir) |
            (df["Time_taken(min)"] > ust_sinir)]
```

Koordinat sütunlarında 0 değeri veya negatif işaretli hatalı kayıtlar tespit edilmiş; bu kayıtlar gerçek bir konumu ifade etmediği için düzeltilmiş ya da veri setinden çıkarılmıştır. Aykırı değerler modelin öğrenmesini şu yönlerden etkiler:

- Model, nadir görülen uç örneklerle aşırı uyum sağlayarak genelleme yeteneğini kaybedebilir.
- RMSE gibi hataların karesini alan metrikler aykırı değerlerden fazlasıyla etkilenir ve model olduğundan başarısız görünebilir.
- Gerçekten var olan uç durumlar (örneğin festival günü yoğunluğu) ise değerli bilgi taşır; bu nedenle her aykırı değer silinmemiş, mantıksal olarak değerlendirilmiştir.

5. VERİ ÖN İŞLEME

Bu aşamada veri seti, makine öğrenmesi algoritmalarının işleyebileceği temiz ve sayısal bir yapıya dönüştürülmüştür.

5.1. Veri Temizleme

Veri setindeki metin tabanlı sütunlarda gereksiz boşluklar ve ön ekler bulunmaktadır. Örneğin hava durumu değerleri "conditions Sunny" biçiminde, teslimat süresi ise "(min) 24" biçiminde tutulmuştur. Bu ifadeler temizlenerek sütunlar kullanılabilir hale getirilmiştir.

```
# Sütun adlarındaki ve değerlerdeki fazla boşlukların temizlenmesi
df.columns = df.columns.str.strip()

# Hedef değişkenin sayısal hale getirilmesi
df["Time_taken(min)"] = (df["Time_taken(min)"]
                        .str.replace("(min)", "", regex=False)
                        .str.strip()
                        .astype(int))

# Hava durumu sütunundaki ön ekin kaldırılması
df["Weatherconditions"] = (df["Weatherconditions"]
                          .str.replace("conditions ", "", regex=False)
                          .str.strip())
```

Ayrıca sayısal olması gereken Delivery_person_Age, Delivery_person_Ratings ve multiple_deliveries sütunları uygun veri tiplerine dönüştürülmüştür.

5.2. Tarih ve Zaman Değişkenleri

Order_Date, Time_Orderd ve Time_Order_picked sütunları ham hâllerıyla metin formatındadır ve model tarafından doğrudan kullanılamaz. Bu sütunlar tarih-saat tipine dönüştürülerek anlamlı yeni özellikler üretilmiştir.

```
df["Order_Date"] = pd.to_datetime(df["Order_Date"], format="%d-%m-%Y")

df["order_day"] = df["Order_Date"].dt.day          # Ayın günü
df["order_month"] = df["Order_Date"].dt.month      # Ay bilgisi
df["order_dayofweek"] = df["Order_Date"].dt.dayofweek # Haftanın günü

# Hafta sonu bilgisi: 5 ve 6 -> Cumartesi ve Pazar
df["is_weekend"] = df["order_dayofweek"].apply(
    lambda x: 1 if x >= 5 else 0)

# Sipariş saati
df["order_hour"] = pd.to_datetime(df["Time_Orderd"],
                                   format="%H:%M:%S",
                                   errors="coerce").dt.hour
```

Bu dönüşümler sayesinde model, teslimat süresinin gün içindeki saate ve haftanın gününe göre nasıl değiştiğini öğrenebilmektedir.

5.3. Kategorik Değişkenlerin Dönüştürülmesi

Veri setinde Weatherconditions, Road_traffic_density, Type_of_order, Type_of_vehicle, Festival ve City sütunları kategoriktir. Makine öğrenmesi algoritmaları metin verisiyle çalışmadığı için bu sütunlar sayısal değerlere dönüştürülmüştür.

Bu projede Label Encoding yöntemi tercih edilmiştir. Tercih nedenleri şunlardır:

- XGBoost ve LightGBM gibi ağaç tabanlı modeller, kategorileri eşik değerlerine göre bölerek işlediklerinden sayısal etiketler arasında yapay bir büyüklük ilişkisi kurmazlar.
- One-Hot Encoding, özellikle Delivery_person_ID gibi çok sayıda benzersiz değer içeren sütunlarda sütun sayısını aşırı artırarak modeli yavaşlatır ve bellek tüketimini yükseltir.
- Road_traffic_density değişkeni (Low < Medium < High < Jam) doğal bir sıralamaya sahiptir ve bu sıralama etiketleme ile korunabilmektedir.

```
from sklearn.preprocessing import LabelEncoder

kategorik_sutunlar = ["Weatherconditions", "Road_traffic_density",
                     "Type_of_order", "Type_of_vehicle",
                     "Festival", "City"]

le = LabelEncoder()
for sutun in kategorik_sutunlar:
    df[sutun] = le.fit_transform(df[sutun].astype(str))
```

6. ÖZELLİK MÜHENDİSLİĞİ

Özellik mühendisliği, mevcut değişkenlerden modelin daha kolay öğrenebileceği yeni ve anlamlı değişkenler türetme sürecidir. Bu projede en kritik katkı, ham koordinat bilgilerinin tek başına anlam taşımaması nedeniyle bunlardan teslimat mesafesinin hesaplanması olmuştur.

6.1. Haversine Formülü ile Mesafe Hesaplama

Veri setinde restoranın (Restaurant_latitude, Restaurant_longitude) ve teslimat adresinin (Delivery_location_latitude, Delivery_location_longitude) enlem-boylam bilgileri bulunmaktadır. Model bu dört sayıyı ayrı ayrı gördüğünde aralarındaki uzaklık ilişkisini kurmakta zorlanır.

Haversine formülü, dünyanın küresel yapısını dikkate alarak iki koordinat arasındaki kuş uçuşu mesafeyi kilometre cinsinden hesaplar. Bu sayede dört ayrı koordinat sütunu, teslimat süresiyle doğrudan ilişkili tek bir anlamlı değişkene dönüştürülmüştür.

```
import numpy as np

def haversine(lat1, lon1, lat2, lon2):
    R = 6371 # Dünya'nın yarıçapı (km)

    # Dereceden radyana çevirme
    lat1, lon1, lat2, lon2 = map(np.radians, [lat1, lon1, lat2, lon2])

    dlat = lat2 - lat1
    dlon = lon2 - lon1

    a = (np.sin(dlat / 2) ** 2 +
         np.cos(lat1) * np.cos(lat2) * np.sin(dlon / 2) ** 2)
    c = 2 * np.arcsin(np.sqrt(a))

    return R * c

df["distance_km"] = haversine(df["Restaurant_latitude"],
                              df["Restaurant_longitude"],
                              df["Delivery_location_latitude"],
                              df["Delivery_location_longitude"])
```

Hesaplama sonrasında mantıksız mesafe değerleri (örneğin 0 km veya olağandışı derecede büyük değerler) hatalı koordinat kaydı olarak değerlendirilmiş ve temizlenmiştir.

6.2. Zaman Tabanlı Özellikler

Sipariş saati ile siparişin restorandan alındığı saat arasındaki fark, restoranın hazırlık süresini yansıtan önemli bir göstergedir. Bu fark ayrı bir özellik olarak hesaplanmıştır.

```
fmt = "%H:%M:%S"
siparis = pd.to_datetime(df["Time_Orderd"], format=fmt, errors="coerce")
alinma = pd.to_datetime(df["Time_Order_picked"], format=fmt, errors="coerce")

# Hazırlık süresi (dakika)
df["prep_time_min"] = (alinma - siparis).dt.total_seconds() / 60

# Gece yarısını geçen siparişler için düzeltme
df.loc[df["prep_time_min"] < 0, "prep_time_min"] += 24 * 60

# Yoğun saat bilgisi (akşam yemeği saatleri)
df["is_peak_hour"] = df["order_hour"].apply(
    lambda x: 1 if x in [12, 13, 19, 20, 21] else 0)
```


6.3. Oluşturulan Özelliklerin Özeti

Yeni Özellik	Kaynak Değişkenler	Teslimat Süresine Katkısı
distance_km	Restoran ve teslimat koordinatları	Mesafe arttıkça yolda geçen süre artar; süreyi doğrudan belirleyen temel faktördür.
prep_time_min	Time_Orderd, Time_Order_picked	Restoranın hazırlık süresini yansıtır; toplam sürenin bir parçasıdır.
order_hour	Time_Orderd	Gün içindeki yoğunluk farkını modele öğretir.
is_peak_hour	order_hour	Öğle ve akşam yoğun saatlerinde artan gecikmeleri temsil eder.
order_dayofweek	Order_Date	Hafta içi ve hafta sonu trafik farkını yakalar.
is_weekend	order_dayofweek	Hafta sonu sipariş yoğunluğunun etkisini gösterir.
order_day, order_month	Order_Date	Dönemsel ve mevsimsel değişimleri temsil eder.

Tablo 6.1. Özellik mühendisliği ile oluşturulan yeni değişkenler

Bunların yanında veri setinde hazır bulunan Road_traffic_density (trafik yoğunluğu), Weatherconditions (hava koşulları), Type_of_vehicle (araç türü), Vehicle_condition (araç durumu), City (teslimat bölgesi), Delivery_person_Age (kurye yaşı), Delivery_person_Ratings (kurye puanı), multiple_deliveries (eş zamanlı teslimat sayısı) ve Festival değişkenleri de modelin girdileri arasında yer almaktadır. ID ve Delivery_person_ID sütunları ise tahmin gücü taşımadıkları ve aşırı öğrenmeye yol açabilecekleri için modelleme öncesinde çıkarılmıştır.

7. MODELLEME SÜRECİ

Modelleme aşamasında veri seti, bağımsız değişkenler (X) ve hedef değişken (y) olarak ikiye ayrılmıştır. Ardından model performansının tarafsız biçimde ölçülebilmesi için veri, eğitim ve test kümelerine bölünmüştür.

```
from sklearn.model_selection import train_test_split

X = df.drop("Time_taken(min)", axis=1)
y = df["Time_taken(min)"]

X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    test_size=0.2,
    random_state=42
)

print(X_train.shape, X_test.shape)
```

- **Eğitim verisi (X_train, y_train):** Modelin örüntüleri öğrendiği veri kümesidir. Verinin %80'ini oluşturur.
- **Test verisi (X_test, y_test):** Modelin daha önce hiç görmediği verilerdir. Modelin gerçek dünyadaki başarısını ölçmek için kullanılır.
- **test_size=0.2:** Verinin %20'sinin test için ayrıldığını belirtir. Bu oran, hem yeterli eğitim verisi bırakmak hem de güvenilir bir test ölçümü yapmak açısından yaygın bir tercihtir.
- **random_state=42:** Veri bölünmesinin rastgeleliğini sabitler. Böylece kod her çalıştırıldığında aynı bölünme elde edilir ve sonuçlar tekrarlanabilir olur.

Modelin test verisi üzerinde ölçülen başarısı, eğitim verisindeki başarısından belirgin şekilde düşükse aşırı öğrenme (overfitting) söz konusudur. Bu nedenle her iki model de yalnızca test verisi üzerinden değerlendirilmiştir.

8. XGBOOST MODELİ

8.1. XGBoost Neden Seçildi?

XGBoost (Extreme Gradient Boosting), yapılandırılmış (tablo biçimli) veri setlerinde en yüksek başarıyı sağlayan algoritmalarından biridir. Bu projede tercih edilmesinin temel nedenleri şunlardır:

- Teslimat süresi ile mesafe, trafik ve hava koşulları arasındaki ilişkiler doğrusal değildir; XGBoost bu tür karmaşık ilişkileri başarıyla modelleyebilir.
- Değişkenler arasındaki etkileşimleri (örneğin yüksek trafik ile uzun mesafenin birlikte oluşturduğu etkiyi) otomatik olarak öğrenir.
- İçerdiği düzenleme (regularization) mekanizmaları sayesinde aşırı öğrenmeye karşı dirençlidir.
- Hangi değişkenin tahmin üzerinde ne kadar etkili olduğunu gösteren özellik önem analizini desteklemesi, sonuçların yorumlanabilmesini sağlar.

8.2. Gradient Boosting Mantığı

Gradient Boosting, tek bir güçlü model kurmak yerine çok sayıda zayıf modeli (karar ağacını) art arda ekleyerek güçlü bir model oluşturma yaklaşımıdır. Süreç şu şekilde işler:

- İlk karar ağacı kurulur ve tahminler üretilir.
- Gerçek değerler ile tahminler arasındaki farklar, yani hatalar (artıklar) hesaplanır.
- Bir sonraki ağaç, bu hataları azaltmaya odaklanacak şekilde eğitilir.
- Bu döngü belirlenen ağaç sayısına ulaşılan kadar tekrarlanır ve tüm ağaçların tahminleri toplanarak nihai sonuç elde edilir.

Böylece model her adımda bir önceki adımın eksiklerini kapatarak kademeli olarak iyileşir. XGBoost, karar ağaçlarını verinin farklı bölgelerine göre dallandırdığı için "mesafe 5 km'den büyükse ve trafik yoğunsa süre belirgin şekilde artar" gibi kural benzeri örüntüleri kendiliğinden keşfedebilir.

8.3. Model Kodu ve Hiperparametreler

```
from xgboost import XGBRegressor

xgb_model = XGBRegressor(
    n_estimators=[değer],          # Kurulacak karar ağacı sayısı
    learning_rate=[değer],         # Her ağacın katkı oranı (öğrenme hızı)
    max_depth=[değer],            # Ağaçların maksimum derinliği
    subsample=[değer],            # Her ağaçta kullanılacak veri oranı
    random_state=42
)

xgb_model.fit(X_train, y_train)

xgb_pred = xgb_model.predict(X_test)
```

Hiperparametre	Açıklama
n_estimators	Modelin kuracağı toplam ağaç sayısıdır. Artması başarıyı yükseltebilir ancak eğitim süresini uzatır ve aşırı öğrenme riskini artırır.
learning_rate	Her ağacın nihai tahmine katkı oranıdır. Küçük değerler daha kararlı öğrenme

Hiperparametre	Açıklama
	sağlar, buna karşılık daha fazla ağaç gerektirir.
max_depth	Ağaçların derinliğini sınırlar. Derin ağaçlar karmaşık ilişkileri öğrenir ancak ezberleme eğilimi gösterir.
subsample	Her ağacın eğitiminde kullanılan veri oranıdır. 1'den küçük değerler modelin genelleme gücünü artırır.
random_state	Rastgeleliği sabitleyerek sonuçların tekrarlanabilir olmasını sağlar.

Tablo 8.1. XGBoost modelinde kullanılan temel hiperparametreler

9. LIGHTGBM MODELİ

9.1. LightGBM Neden Seçildi?

LightGBM (Light Gradient Boosting Machine), XGBoost ile aynı gradient boosting mantığına dayanan ancak hız ve bellek verimliliği için optimize edilmiş bir algoritmadır. Projede tercih edilme nedenleri:

- Yemek teslimat verileri binlerce hatta milyonlarca sipariş kaydına ulaşabilir; LightGBM büyük veri setlerinde belirgin bir hız avantajı sağlar.
- Eğitim süresinin kısa olması, farklı hiperparametre kombinasyonlarının daha çok denenebilmesine imkân tanır.
- Bellek kullanımı XGBoost'a kıyasla daha düşüktür.
- Ağaç tabanlı yapısı sayesinde değişkenler arasındaki doğrusal olmayan ilişkileri ve etkileşimleri öğrenebilir.
- XGBoost ile aynı problem üzerinde karşılaştırılabilir olması, model seçimi için sağlıklı bir kıyas zemini oluşturur.

9.2. Tree-Based Yapısı ve Çalışma Mantığı

LightGBM'i XGBoost'tan ayıran temel fark, ağaçların büyütülme biçimidir. XGBoost ağacı seviye seviye (level-wise) genişletirken, LightGBM en fazla hata azalması sağlayacak yaprağı seçerek büyütür (leaf-wise). Bu yaklaşım aynı ağaç sayısı ile daha düşük hata elde edilmesini sağlar; ancak ağaçlar dengesiz büyüyebildiği için küçük veri setlerinde aşırı öğrenme riski artar. Bu risk num_leaves ve max_depth parametreleri ile kontrol edilmiştir.

Ayrıca LightGBM, sürekli değişkenleri histogram tabanlı olarak gruplara ayırarak işler. Bu sayede olası bölünme noktalarının tamamını tek tek denemek yerine sınırlı sayıda aday üzerinden hızlı karar verir.

9.3. Model Kodu ve Hiperparametreler

```
from lightgbm import LGBMRegressor

lgbm_model = LGBMRegressor(
    n_estimators=[değer],      # Ağaç sayısı
    learning_rate=[değer],    # Öğrenme hızı
    max_depth=[değer],        # Maksimum ağaç derinliği
    num_leaves=[değer],       # Bir ağaçtaki maksimum yaprak sayısı
    random_state=42
)

lgbm_model.fit(X_train, y_train)

lgbm_pred = lgbm_model.predict(X_test)
```

Hiperparametre	Açıklama
n_estimators	Eğitilecek ağaç sayısıdır. Öğrenme hızı ile birlikte dengeli seçilmelidir.
learning_rate	Her ağacın katkı ağırlığıdır. Düşük değerler daha kararlı ancak daha yavaş öğrenme sağlar.
max_depth	Ağaç derinliğini sınırlayarak aşırı öğrenmeyi engeller.
num_leaves	LightGBM'e özgü en kritik parametredir. Modelin karmaşıklığını belirler; çok yüksek değerler ezberlemeye yol açar.
random_state	Sonuçların tekrarlanabilirliğini sağlar.

Tablo 9.1. LightGBM modelinde kullanılan temel hiperparametreler

10. MODEL PERFORMANS DEĞERLENDİRMESİ

Regresyon modellerinin başarısı, tahmin edilen değerler ile gerçek değerler arasındaki farkın büyüklüğü üzerinden ölçülür. Bu projede dört farklı metrik kullanılmıştır.

```
from sklearn.metrics import (mean_absolute_error,
                             mean_squared_error, r2_score)
import numpy as np

def degerlendir(model_adi, y_test, y_pred):
    mae = mean_absolute_error(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_test, y_pred)

    print(f"{model_adi} -> MAE: {mae:.2f} | MSE: {mse:.2f} | "
          f"RMSE: {rmse:.2f} | R2: {r2:.2f}")

degerlendir("XGBoost", y_test, xgb_pred)
degerlendir("LightGBM", y_test, lgbm_pred)
```

10.1. Metriklerin Anlamı

Metrik	Açıklama
MAE (Ortalama Mutlak Hata)	Tahminlerin gerçek değerlerden ortalama kaç dakika saptığını gösterir. Yorumlanması en kolay metriktir; MAE değerinin 4 olması, modelin ortalama 4 dakika hata yaptığı anlamına gelir.
MSE (Ortalama Kare Hata)	Hataların karelerinin ortalamasıdır. Büyük hataları daha ağır cezalandırır, ancak birimi dakikanın karesi olduğu için doğrudan yorumlanamaz.
RMSE (Kök Ortalama Kare Hata)	MSE'nin kareköküdür ve dakika birimine geri döner. Büyük sapmalara duyarlı olduğundan, ciddi gecikmelerin önemli olduğu bu problemde kritik bir metriktir.
R ² (Belirleme Katsayısı)	Modelin, teslimat süresindeki değişkenliğin ne kadarını açıklayabildiğini gösterir. 1'e yakın değerler yüksek açıklama gücünü ifade eder.

Tablo 10.1. Kullanılan performans metrikleri

10.2. Model Sonuçları

Model	MAE	MSE	RMSE	R ²
XGBoost	[MAE değeri]	[MSE değeri]	[RMSE değeri]	[R ² değeri]

Model	MAE	MSE	RMSE	R ²
LightGBM	[MAE değeri]	[MSE değeri]	[RMSE değeri]	[R ² değeri]

Tablo 10.2. XGBoost ve LightGBM modellerinin test verisi üzerindeki performansı

10.3. Sonuçların Yorumlanması

- **Daha düşük MAE:** Modelin ortalama tahmin sapmasının daha az olduğunu gösterir. Kullanıcıya gösterilen tahmini teslimat süresinin gerçeğe daha yakın olması anlamına gelir.
- **Daha düşük RMSE:** Modelin büyük hatalar yapma eğiliminin daha az olduğunu ifade eder. MAE'nin düşük ancak RMSE'nin yüksek olması, modelin genelde iyi tahmin yaptığını fakat bazı siparişlerde ciddi şekilde yanlış olduğunu gösterir.
- **Daha yüksek R²:** Modelin teslimat süresindeki değişimi daha iyi açıkladığını gösterir. Örneğin 0.82'lik bir R², süredeki değişkenliğin %82'sinin model tarafından açıklandığı anlamına gelir.

Değerlendirmede tek bir metriğe bakmak yanıltıcı olabilir; bu nedenle modeller dört metrik birlikte incelenerek karşılaştırılmıştır.

11. XGBOOST VE LIGHTGBM KARŞILAŞTIRMASI

İki algoritma aynı veri seti, aynı eğitim-test bölünmesi ve aynı özellikler kullanılarak eğitilmiş; hem performans hem de pratik kullanım kriterleri açısından karşılaştırılmıştır.

Kriter	XGBoost	LightGBM
Eğitim Hızı	Daha yavaş; seviye bazlı ağaç büyütme daha fazla hesaplama gerektirir.	Daha hızlı; histogram tabanlı ve yaprak bazlı büyütme sayesinde eğitim süresi kısadır.
Tahmin Performansı	Yüksek; iyi ayarlandığında çok başarılı sonuç verir.	Yüksek; çoğu durumda XGBoost ile benzer veya daha iyi sonuç üretir.
Büyük Veri Setleri	Veri büyüdükçe eğitim süresi belirgin şekilde artar.	Büyük veri setlerinde açık üstünlük sağlar.
Bellek Kullanımı	Daha yüksek bellek tüketir.	Daha az bellek kullanır.
Hiperparametre Hassasiyeti	Varsayılan değerlerle bile makul sonuç verir; daha toleranslıdır.	num_leaves ve max_depth ayarlarına duyarlıdır; yanlış ayarda aşırı öğrenme görülebilir.
Özellik Önemi Analizi	Gelişmiş ve yorumlanabilir önem ölçütleri sunar.	Hızlı ve pratik önem analizi sağlar.

Tablo 11.1. XGBoost ve LightGBM algoritmalarının karşılaştırılması

11.1. Bu Proje İçin Hangi Model Daha Başarılıdır?

Test verisi üzerinde elde edilen metriklere göre: *[Model sonuçlarına göre XGBoost / LightGBM daha başarılı bulunmuştur.]*

Model seçimi yapılırken yalnızca hata metrikleri değil, aşağıdaki pratik kriterler de dikkate alınmalıdır:

- İki model arasındaki metrik farkı çok küçükse, eğitim süresi ve bellek verimliliği nedeniyle LightGBM operasyonel olarak daha avantajlı bir tercih olur.
- XGBoost belirgin şekilde düşük RMSE üretiyorsa, büyük gecikmelerin maliyetli olduğu bu problemde XGBoost tercih edilmelidir.
- Gerçek zamanlı çalışacak bir sistemde modelin yeniden eğitilme sıklığı da önemlidir; sık güncellenen bir yapıda hızlı eğitim büyük avantaj sağlar.

12. ÖZELLİK ÖNEM ANALİZİ

Özellik önem analizi, modelin tahmin yaparken hangi değişkenlere ne ölçüde ağırlık verdiğini gösterir. Bu analiz hem modelin mantığını anlamayı hem de gereksiz değişkenleri tespit etmeyi sağlar.

```
import pandas as pd

feature_importance = pd.DataFrame({
    "Feature": X_train.columns,
    "Importance": xgb_model.feature_importances_
}).sort_values(by="Importance", ascending=False)

print(feature_importance.head(10))
```

Analiz sonucunda teslimat süresini en çok etkilediği belirlenen değişkenler aşağıdaki tabloda özetlenmiştir. Sıralama model çıktısına göre değişebileceğinden ilgili alanlar yer tutucu olarak bırakılmıştır.

Sıra	Değişken	Teslimat Süresi Üzerindeki Etkisi
1	[En önemli değişken]	Model tarafından en yüksek ağırlık verilen değişkendir.
2	distance_km	Mesafe arttıkça yolda geçirilen süre doğrudan artar.
3	Road_traffic_density	Trafik sıkışıklığı, aynı mesafenin çok daha uzun sürede kat edilmesine neden olur.
4	Delivery_person_Ratings	Yüksek puanlı kuryelerin genellikle daha hızlı ve düzenli teslimat yaptığını yansıtır.
5	multiple_deliveries	Aynı anda birden fazla sipariş taşınması teslimat süresini uzatır.
6	Weatherconditions	Yağış, sis ve fırtına gibi koşullar sürüş hızını düşürür.
7	order_hour / is_peak_hour	Öğle ve akşam yoğun saatlerinde süreler belirgin şekilde artar.
8	City	Metropol bölgelerde trafik ve adres bulma süresi daha uzundur.

Tablo 12.1. Teslimat süresi üzerinde en etkili değişkenler

Bu sonuçlar, modelin rastgele değil operasyonel gerçeklikle uyumlu bir mantıkla öğrendiğini göstermektedir. Mesafe ve trafik yoğunluğunun ön plana çıkması, sahadaki teslimat dinamikleriyle birebir örtüşmektedir.

13. TAHMİN SONUÇLARININ GÖRSELLEŞTİRİLMESİ

Model performansı yalnızca sayısal metriklerle değil, görsel analizlerle de değerlendirilmiştir. Görselleştirmeler, metriklerin gizlediği hataları ortaya çıkarabilir.

13.1. Gerçek ve Tahmin Edilen Değerlerin Karşılaştırılması

Gerçek değerlerin yatay, tahmin edilen değerlerin dikey ekseninde gösterildiği saçılım grafiği çizilir. Noktaların 45 derecelik doğruya yakın toplanması, modelin başarılı olduğunu gösterir.

```
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 6))
plt.scatter(y_test, xgb_pred, alpha=0.3)
plt.plot([y_test.min(), y_test.max()],
         [y_test.min(), y_test.max()], "r--")
plt.xlabel("Gerçek Teslimat Süresi (dk)")
```

```
plt.ylabel("Tahmin Edilen Süre (dk)")
plt.title("Gerçek ve Tahmin Edilen Değerler")
plt.show()
```

13.2. Residual (Artık) Analizi

Artıklar, gerçek değer ile tahmin arasındaki farklardır. Artıkların sıfır etrafında rastgele dağılması modelin sistematik bir hata yapmadığını gösterir. Belirli bir yönde eğilim görülmesi ise modelin örneğin uzun teslimatları sürekli olduğundan kısa tahmin ettiğine işaret eder.

```
residuals = y_test - xgb_pred

plt.figure(figsize=(8, 6))
plt.scatter(xgb_pred, residuals, alpha=0.3)
plt.axhline(y=0, color="r", linestyle="--")
plt.xlabel("Tahmin Edilen Süre (dk)")
plt.ylabel("Artık (Hata)")
plt.title("Residual Analizi")
plt.show()
```

13.3. Özellik Önemi Grafiği

En etkili değişkenlerin yatay çubuk grafiği ile gösterilmesi, sonuçların hızlı yorumlanmasını sağlar.

```
plt.figure(figsize=(8, 6))
ilk10 = feature_importance.head(10)
plt.barh(ilk10["Feature"], ilk10["Importance"])
plt.gca().invert_yaxis()
plt.title("En Önemli 10 Değişken")
plt.show()
```

13.4. Model Performans Karşılaştırma Grafiği

XGBoost ve LightGBM modellerinin MAE, RMSE ve R^2 değerleri yan yana çubuk grafik olarak gösterilerek karşılaştırma görselleştirilmiştir.

```
sonuclar = pd.DataFrame({
    "Model": ["XGBoost", "LightGBM"],
    "MAE": [xgb_mae, lgbm_mae],
    "RMSE": [xgb_rmse, lgbm_rmse]
})

sonuclar.plot(x="Model", kind="bar", figsize=(8, 5))
plt.title("Model Performans Karşılaştırması")
plt.ylabel("Hata (dakika)")
plt.show()
```

14. SONUÇ VE DEĞERLENDİRME

Bu projenin amacı, bir yemek siparişinin müşteriye kaç dakikada teslim edileceğini geçmiş sipariş verilerinden yararlanarak tahmin eden bir makine öğrenmesi modeli geliştirmektir. Hedef değişkenin sürekli sayısal bir değer olması nedeniyle problem regresyon problemi olarak ele alınmıştır.

Çalışma kapsamında ham veri seti önce keşifsel veri analizi ile incelenmiş; eksik değerler, tekrarlayan kayıtlar ve aykırı değerler tespit edilerek uygun yöntemlerle giderilmiştir. Metin sütunlarındaki gereksiz ifadeler temizlenmiş, sayısal olması gereken sütunlar doğru veri tipine dönüştürülmüş ve kategorik değişkenler Label Encoding ile sayısallaştırılmıştır.

Özellik mühendisliği aşamasında koordinat bilgilerinden Haversine formülü ile teslimat mesafesi hesaplanmış; sipariş ve alınma saatlerinden hazırlık süresi türetilmiş; tarih bilgisinden gün, ay, haftanın günü, hafta sonu ve yoğun saat gibi anlamlı özellikler oluşturulmuştur. Bu adımlar, modelin ham veriden doğrudan çıkaramayacağı ilişkileri öğrenmesini mümkün kılmıştır.

Modelleme aşamasında, tablo biçimli verilerde yüksek başarımlı gösteren iki gradient boosting algoritması olan XGBoost ve LightGBM tercih edilmiştir. Her iki model de aynı eğitim ve test kümeleri üzerinde eğitilmiş; MAE, MSE, RMSE ve R^2 metrikleriyle değerlendirilmiştir. Yapılan karşılaştırma sonucunda [En başarılı model] daha iyi performans göstermiştir.

Özellik önem analizine göre teslimat süresini en çok etkileyen faktörler *teslimat mesafesi*, *trafik yoğunluğu*, *eş zamanlı teslimat sayısı*, *hava koşulları* ve *sipariş saati* olarak belirlenmiştir. Bu bulgular, teslimat süresinin büyük ölçüde yol koşulları ve operasyonel yoğunluk tarafından belirlendiğini göstermektedir.

14.1. Gerçek Uygulama Senaryosu

Geliştirilen model, bir yemek teslimat platformunun sipariş akışına şu şekilde entegre edilebilir:

- Müşteri sipariş oluşturduğunda sistem, restoranın ve teslimat adresinin koordinatlarını alarak Haversine formülü ile mesafeyi anlık olarak hesaplar.
- Sipariş saati, haftanın günü, güncel trafik yoğunluğu ve hava durumu bilgisi model girdisi olarak hazırlanır.
- Siparişi üstlenen kuryenin yaşı, puanı, araç türü, araç durumu ve o an taşıdığı eş zamanlı sipariş sayısı eklenir.
- Model bu bilgileri kullanarak dakika cinsinden tahmini teslimat süresini üretir ve müşteriye "Tahmini teslimat: 27 dakika" biçiminde gösterilir.
- Teslimat tamamlandığında gerçekleşen süre kaydedilir ve bu veri, modelin periyodik olarak yeniden eğitilmesinde kullanılır.

Bu yapı yalnızca müşteri bilgilendirmesi için değil, aynı zamanda operasyonel planlama için de kullanılabilir. Tahmini süresi yüksek çıkan siparişler daha deneyimli kuryelere yönlendirilebilir, yoğun saatlerde kurye sayısı önceden artırılabilir veya beklenen gecikmeler müşteriye proaktif olarak bildirilebilir.

15. GELECEKTE YAPILABİLECEK GELİŞTİRMELER

Proje, aşağıdaki başlıklarla daha ileri bir seviyeye taşınabilir.

Geliştirme Alanı	Açıklama
Hiperparametre Optimizasyonu	Modellerin parametreleri manuel değil sistematik olarak aranarak en iyi kombinasyon bulunabilir.
GridSearchCV	Belirlenen parametre kombinasyonlarının tamamını dener. Kesin sonuç verir ancak yavaştır.
RandomizedSearchCV	Parametre uzayından rastgele seçim yaparak çok daha kısa sürede iyi sonuçlara ulaşır.
Optuna	Önceki denemelerin sonuçlarından öğrenerek akıllı arama yapar; büyük parametre uzaylarında en verimli yöntemdir.
Cross Validation	Veri birden fazla parçaya bölünerek model her parçada test edilir. Böylece başarı tek bir bölünmeye bağlı kalmaz ve daha güvenilir ölçüm yapılır.

Geliştirme Alanı	Açıklama
Daha Fazla Veri	Farklı şehir ve dönemlere ait verilerle model daha genellenebilir hale gelir.
Gerçek Zamanlı Trafik Verisi	Statik trafik kategorileri yerine harita servislerinden alınan anlık trafik verisi kullanılabilir.
Hava Durumu API Verisi	Sipariş anındaki gerçek sıcaklık, yağış ve rüzgâr verileri modele eklenebilir.
Ensemble / Stacking	XGBoost ve LightGBM tahminleri birleştirilerek tek bir modelden daha iyi sonuç elde edilebilir.
Deep Learning	Çok büyük veri hacimlerinde sinir ağı tabanlı modeller denenerek karşılaştırma yapılabilir.

Tablo 15.1. Gelecekte yapılabilecek geliştirmeler

16. GENEL SONUÇ

Bu proje, bir makine öğrenmesi çalışmasının ham veriden başlayarak kullanılabilir bir tahmin modeline dönüşmesine kadar geçen tüm aşamaları kapsamaktadır. Veri setinin tanınması, eksik ve hatalı kayıtların temizlenmesi, keşifsel veri analizi ile verinin yapısının anlaşılması, koordinat ve zaman bilgilerinden anlamlı yeni özellikler türetilmesi, kategorik değişkenlerin sayısallaştırılması ve verinin eğitim-test olarak bölünmesi; modelleme öncesinde atılan ve sonuç kalitesini doğrudan belirleyen adımlar olmuştur. Elde edilen deneyim, model başarısının yalnızca seçilen algoritmaya değil, büyük ölçüde veriye uygulanan hazırlık kalitesine bağlı olduğunu açıkça göstermektedir.

Modelleme aşamasında XGBoost ve LightGBM algoritmalarının aynı koşullar altında eğitilip MAE, MSE, RMSE ve R^2 metrikleriyle karşılaştırılması, model seçiminin sezgisel değil ölçüme dayalı biçimde yapılmasını sağlamıştır. Özellik önem analizi ise modeli bir "kara kutu" olmaktan çıkararak hangi faktörlerin teslimat süresini belirlediğini yorumlanabilir hâle getirmiştir. Sonuç olarak proje; veri temizleme, keşifsel veri analizi, özellik mühendisliği, regresyon modelleme, gradient boosting algoritmaları, model karşılaştırması, performans değerlendirmesi ve özellik önemi analizi becerilerini bir arada barındıran, gerçek bir iş problemine uygulanabilir bütünsel bir veri bilimi çalışmasıdır.

17. KAYNAKÇA

- [1] Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.
- [2] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T. Y. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. Advances in Neural Information Processing Systems.
- [3] Pedregosa, F. ve diğerleri (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825–2830.
- [4] XGBoost Belgelendirmesi. <https://xgboost.readthedocs.io>
- [5] LightGBM Belgelendirmesi. <https://lightgbm.readthedocs.io>
- [6] Scikit-learn Belgelendirmesi. <https://scikit-learn.org>
- [7] Pandas Belgelendirmesi. <https://pandas.pydata.org>