VERİ BİLİMİ DERSİ

Makine Öğrenmesi I: Regresyon

Hafta 5 · Modül 5

6
Saat

4
Notebook

3
Regresyon Türü

Dr. Murat Altun

Veri Bilimi ve Yapay Zekâ Eğitimi · 2026

İçindekiler

Makine Öğrenmesi Nedir?

BÖLÜM 1

Denetimli Öğrenme

Etiketli veri ile eğitim. Model, giriş-çıkış ilişkisini öğrenir. Regresyon ve sınıflandırma bu kategoridedir.

Denetimsiz Öğrenme

Etiketsiz veride örüntü keşfi. Kümeleme (K-Means), boyut indirgeme (PCA) gibi yöntemler kullanılır.

Pekiştirmeli Öğrenme

Ödül-ceza mekanizması ile öğrenme. Ajan, ortamla etkileşerek en iyi stratejiyi keşfeder (oyunlar, robotik).

Bu derste odağımız: Denetimli Öğrenme → Regresyon. Sürekli bir sayısal değeri (fiyat, sıcaklık, satış) tahmin etmek.

Denetimli Öğrenme: Regresyon vs Sınıflandırma

BÖLÜM 1

Regresyon (Sürekli Çıktı)

1. Çıktı: Sürekli sayısal değer (ör: 345.000 TL)
2. Ev fiyat tahmini
3. Araç değer tahmini
4. Sıcaklık tahmini
5. Satış miktarı öngörüsü

Sınıflandırma (Kategorik Çıktı)

1. Çıktı: Kategori (ör: spam / spam değil)
2. E-posta sınıflandırma
3. Hastalık teşhisi
4. Müşteri kaybı tahmini
5. Görüntü tanıma (kedi/köpek)

Basit Lineer Regresyon

$$y = mx + b$$

y: tahmin · m: eğim (katsayı) · x: bağımsız değişken · b: kesişim (intercept)

Görsel Analoji

Noktalar arasından en iyi geçen doğruyu çizmek — tüm noktaların hatalarının karelerinin toplamını minimize eden çizgi.

Amaç

Bağımsız değişken (X) ile bağımlı değişken (Y) arasındaki doğrusal ilişkiyi modellemek.

OLS (En Küçük Kareler)

Hata karelerinin toplamını minimize eden m ve b değerlerini bulur. sklearn bu hesaplamayı otomatik yapar.

Varsayımlar

Doğrusallık, bağımsızlık, homojenlik (sabit varyans), normal dağılım. Gerçek dünyada nadiren tam karşılanır.

Lineer Regresyon — Python Kodu


```python
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

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

# Model olustur ve egit
model = LinearRegression()
model.fit(X_train, y_train)

# Tahmin
y_pred = model.predict(X_test)

print(f"R2 Skoru: {model.score(X_test, y_test):.4f}")
```

1. **Import**
   LinearRegression ve train_test_split kütüphanelerini yükle
2. **Veri Ayır**
   %80 eğitim, %20 test olarak böl
3. **Eğit**
   fit() metodu ile modeli eğit
4. **Tahmin**
   predict() ile test verisi üzerinde tahmin yap
5. **Değerlendir**
   R2 skoru ile modelin başarısını ölç

Çoklu Lineer Regresyon

$$y = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_n x_n$$

Birden fazla bağımsız değişken ile tahmin — her xi bir özellik (feature), her bi o özelliğin katsayısı

Ev Fiyat Tahmini Örneği

Özellik (Feature)	Katsayı (bi)	Yorum
m2 (Alan)	+2.500 TL/m2	Alan arttıkça fiyat artar
Oda Sayısı	+15.000 TL/oda	Her oda fiyatı yükseltir
Şehir Merkezi	-1.200 TL/km	Merkeze yakınlık değerli
Bina Yaşı	-800 TL/yıl	Eski bina fiyatı düşürür

Tek vs Çoklu Regresyon

1. Tek değişken -> sınırlı bilgi
2. Çoklu değişken -> gerçekçi model
3. Multicollinearity riski kontrol et
4. Feature selection önemli
5. Overfitting'e dikkat!

Polinom Regresyon

Doğrusal Olmayan İlişkiler

Veriler doğrusal bir ilişki göstermediğinde, polinom terimler (x2, x3, ...) ekleyerek eğrisel bir model oluşturabiliriz. Dikkat: Yüksek dereceler overfitting'e yol açabilir!

$$y = b0 + b1*x + b2*x2$$

Derece 2 (quadratic) örneği
Derece arttıkça model karmaşıklaşır

```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression

# Polinom ozellikleri olustur (derece=2)
poly = PolynomialFeatures(degree=2)
X_poly = poly.fit_transform(X_train)

# Lineer regresyon uygula (polinom ozelliklerle)
model = LinearRegression()
model.fit(X_poly, y_train)

# Test verisini de donustur
X_test_poly = poly.transform(X_test)
y_pred = model.predict(X_test_poly)
```

Train/Test Split

Neden Veriyi Ayırıyoruz?

1. Modelin görmediği veri üzerinde test etmek
2. Overfitting'i (aşırı öğrenme) tespit etmek
3. Gerçek dünya performansını ölçmek
4. Genelleme yeteneğini doğrulamak

Yaygın Bölme Oranları


```python
from sklearn.model_selection import train_test_split

# Veriyi %80 egitim, %20 test olarak bol
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,        # %20 test
    random_state=42     # Tekrarlanabilirlik
)
```

Regresyon Başarı Metrikleri

Metrik	Formül	Açıklama	İdeal Değer
MAE	`SUM|yi - yi_hat| / n`	Ortalama Mutlak Hata — Tahmin hatalarının ortalama büyüklüğü	0'a yakın
MSE	`SUM(yi - yi_hat)^2 / n`	Ortalama Kare Hata — Büyük hataları daha çok cezalandırır	0'a yakın
RMSE	`SQRT(MSE)`	MSE'nin karekökü — Y birimi ile aynı ölçekte, yorumlaması kolay	0'a yakın
R2	`1 - (SS_res / SS_tot)`	Belirleme katsayısı — Modelin varyansı ne kadar açıkladığı	1'e yakın

Tek bir metriğe güvenmeyin! R2 yüksek olsa bile residual (artık) analizi yapın. RMSE, tahmin hatanızın ortalama büyüklüğünü verir.

Feature Engineering

Label Encoding

Kategorik değişkeni sayıya çevir.
Örnek: Kırmızı=0, Mavi=1, Yeşil=2
Dikkat: Sıralı olmayan verilerde yanıltıcı olabilir.

One-Hot Encoding

Her kategori için yeni sütun.
Örnek: Renk_Kırmızı, Renk_Mavi, Renk_Yeşil
pd.get_dummies() ile kolayca uygulanır.

Ölçeklendirme

Farklı ölçekleri normalize et.
StandardScaler: ortalama=0, std=1
MinMaxScaler: [0, 1] aralığına sıkıştır.

Eksik Değerler

Null değerleri doldur veya sil.
SimpleImputer: ortalama, medyan, mod
KNN Imputer: komşulara bakarak doldur.

Scikit-learn Pipeline

Neden Pipeline?

Ön işleme adımlarını ve modeli tek bir nesnede birleştir. Tekrarlanabilir, temiz, hata riski az. Cross-validation ile doğrudan kullanılabilir.

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler, PolynomialFeatures
from sklearn.linear_model import LinearRegression

# Pipeline olustur: Olcekle -> Polinom -> Regresyon
pipe = Pipeline([
    ("scaler",   StandardScaler()),
    ("poly",     PolynomialFeatures(degree=2)),
    ("model",    LinearRegression()),
])

pipe.fit(X_train, y_train)
score = pipe.score(X_test, y_test)
print(f"Pipeline R2: {score:.4f}")
```

Araç Fiyat Tahmini Örneği

Veri Seti: cars.xls

- Kaynak: Gerçek araç satış verileri
- Özellikler: marka, model, yıl, km, motor hacmi
- Hedef: Araç satış fiyatı (TL)
- Satır sayısı: ~2000+ kayıt

Model Sonuçları

0.87	12.400 TL	18.200 TL
R2	MAE	RMSE

Proje İş Akışı

1 -> 2 -> 3 -> 4 -> 5

Veri Yükleme	EDA	Ön İşleme	Model Eğitimi	Değerlendirme
`pd.read_excel('cars.xls')`	`df.describe(), korelasyon`	`Encoding, scaling, split`	`LinearRegression().fit()`	`R2, MAE, RMSE hesapla`

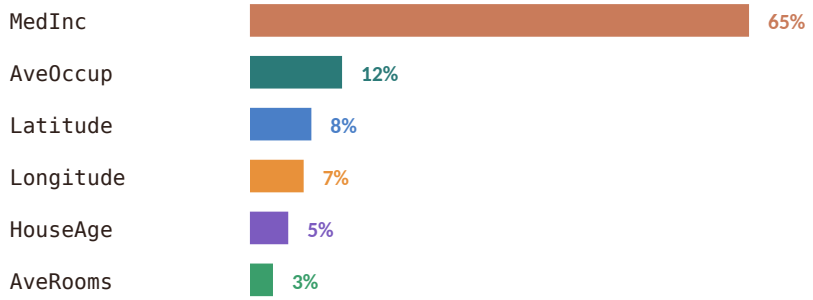
Ev Fiyat Tahmini — California Housing

California Housing Dataset

Scikit-learn ile hazır gelen popüler veri seti

`MedInc`	Medyan gelir
`HouseAge`	Bina yaşı
`AveRooms`	Ortalama oda sayısı
`AveOccup`	Ortalama doluluk
`Latitude`	Enlem
`Longitude`	Boylam

Feature Importance (Önem Sırası)




```python
from sklearn.datasets import fetch_california_housing

data = fetch_california_housing()
X, y = data.data, data.target

# Feature importance: model.coef_ ile katsayilari gor
importance = pd.Series(model.coef_, index=data.feature_names)
importance.abs().sort_values(ascending=False).plot.barh()
```

Model Kaydetme — joblib

Neden Model Kaydederiz?

1. Eğitim saatler sürebilir — tekrar eğitmek istemezsiniz
2. Eğitilmiş modeli farklı ortamlara taşımak (deployment)
3. Model versiyonlama ve karşılaştırma
4. API veya web uygulamasına entegrasyon

joblib vs pickle

Özellik	joblib	pickle
NumPy dizileri	Hızlı	Yavaş
Büyük modeller	Verimli	Yavaş
Kullanım	Kolay	Standart

```
import joblib

# Modeli kaydet
joblib.dump(model, "arac_fiyat_model.joblib")
print("Model kaydedildi!")

# Modeli yukle
loaded_model = joblib.load("arac_fiyat_model.joblib")

# Yuklenen model ile tahmin
prediction = loaded_model.predict([[2020, 15000, 1.6]])
print(f"Tahmin: {prediction[0]:,.0f} TL")
```

Gradio ile Model Arayüzü

Gradio Nedir?

ML modelleriniz için birkaç satır kod ile web arayüzü oluşturun. Paylaşım linki otomatik oluşturulur. Hugging Face Spaces ile deploy edilebilir.

```python
import gradio as gr
import joblib

model = joblib.load("arac_fiyat_model.joblib")

def predict_price(yil, km, motor):
    pred = model.predict([[yil, km, motor]])
    return f"{pred[0]:,.0f} TL"

demo = gr.Interface(
    fn=predict_price,
    inputs=[
        gr.Slider(2005, 2024, label="Yil"),
        gr.Number(label="Kilometre"),
        gr.Slider(1.0, 3.0, label="Motor"),
    ],
    outputs="text",
    title="Arac Fiyat Tahmini"
)
demo.launch()
```

Gradio Bileşenleri

`gr.Slider`	Kaydırıcı (min-max aralık)
`gr.Number`	Sayısal giriş kutusu
`gr.Textbox`	Metin girişi/çıkışı
`gr.Dropdown`	Seçim listesi
`gr.Image`	Görüntü giriş/çıkış
`gr.Plot`	Matplotlib/Plotly grafik

Hugging Face Spaces ile Deploy

BÖLÜM 4

1

Hesap Oluştur

huggingface.co adresinde ücretsiz hesap aç. GitHub hesabınla da giriş yapabilirsin.

2

Space Oluştur

New Space -> SDK: Gradio seç -> Public/Private ayarla -> Create Space butonuna bas.

3

Dosyaları Yükle

app.py (Gradio kodu) + requirements.txt + model dosyalarını yükle veya git push.

4

Canlı Demo!

HF otomatik build eder ve çalıştırır. Linki paylaşarak herkesin modeli denemesini sağla.

Ücretsiz plan: CPU 2 core, 16 GB RAM. Küçük modeller için yeterli. GPU gerektiren modeller için Pro plan gerekir.

Haftalık Notebook'lar

lineer_regresyon.ipynb

Basit ve çoklu lineer regresyon. Scikit-learn ile model eğitimi, fit/predict döngüsü, katsayı yorumlama.

LinearRegression - train_test_split - R2

arac_fiyat_tahmini.ipynb

Gerçek araç verisi ile fiyat tahmini. EDA, encoding, scaling ve model eğitimi end-to-end proje.

cars.xls - EDA - Pipeline - MAE/RMSE

ev_fiyat_tahmini.ipynb

California Housing veri seti. Feature importance, polinom regresyon denemeleri, görselleştirme.

fetch_california_housing - Feature Importance

gradio_deployment.ipynb

Eğitilmiş modeli Gradio ile web arayüzüne dönüştürme. joblib kaydetme ve Hugging Face deploy.

Gradio - joblib - Hugging Face Spaces

Ödev ve Kaynaklar

Haftalık Ödev

1. **Araç Fiyat Modeli**

   cars.xls verisini kullanarak en iyi regresyon modelini oluşturun. R2 > 0.85 hedefleyin.

2. **Gradio Arayüzü**

   Eğittiğiniz modeli Gradio ile web arayüzüne dönüştürün.

3. **Hugging Face Deploy**

   Gradio uygulamanızı Hugging Face Spaces'e deploy edin ve linkini paylaşın.

4. **Rapor**

   EDA bulguları, model karşılaştırması, metrik sonuçları ve çıkarımları bir notebook'ta belgeleyin.

Faydalı Kaynaklar

1. **Scikit-learn Docs**

   `scikit-learn.org/stable/`

   Resmi dokümantasyon — tüm modeller ve API

2. **Kaggle Learn**

   `kaggle.com/learn`

   Ücretsiz ML dersleri + pratik notebook'lar

3. **Gradio Docs**

   `gradio.app/docs`

   Gradio bileşenleri ve kullanım örnekleri

4. **HF Spaces**

   `huggingface.co/spaces`

   Binlerce demo — ilham kaynağı

5. **StatQuest (YouTube)**

   `youtube.com/@statquest`

   Regresyon ve ML konularını görsel anlatan kanal

Hafta 5 — Önemli Çıkarımlar

1. Makine öğrenmesi = veriden öğrenen algoritmalar. Regresyon, sürekli değerleri tahmin eder.
2. Lineer regresyon basit ama güçlü bir temeldir. Çoklu ve polinom ile genişletilir.
3. Train/Test split ve metrikler (R2, RMSE) ile modelin gerçek performansını ölçün.
4. Feature engineering model başarısını dramatik şekilde artırabilir — veriyi iyi tanıyın.
5. Gradio + Hugging Face ile modelinizi dünyayla paylaşın — deployment artık kolay!

“Teori olmadan pratik kördür, pratik olmadan teori işe yaramaz.”

Dr. Murat Altun · Veri Bilimi ve Yapay Zekâ Eğitimi · 2026