VERİ BİLİMİ DERSİ

Ensemble Modeller

Hafta 7 · Modül 7
XGBoost, LightGBM ve Hiperparametre Optimizasyonu

Dr. Murat Altun

Veri Bilimi ve Yapay Zekâ Eğitimi · 2026

6
Saat
(Teori + Uygulama)

3
Notebook
(Ensemble, Churn, Tune)

4
Model
(RF, GBM, XGB, LGBM)

İçindekiler

Ensemble Learning Nedir?

BÖLÜM 1

Kalabalığın Bilgeliği

1906'da Francis Galton bir fuarda keşfetti: 787 kişinin öküzn âğırlığı tahminlerinin ortalaması, gerçek değere uzmanlardan daha yakındı! Ensemble learning de aynı prensibi kullanır — birden fazla modelin birleşik tahmini, tek bir modelden daha güçlüdür.

Tek Model vs Ensemble

Kriter	Tek Model	Ensemble
Doğruluk	Orta	Yüksek
Varyans	Yüksek	Düşük
Overfitting	Risk yüksek	Direnç yüksek
Yorumlanabilirlik	Kolay	Zor

1 Bagging

Paralel eğitim
Örn: Random Forest

2 Boosting

Sıralı hata düzeltme
Örn: XGBoost

3 Stacking

Meta-öğrenme
Örn: StackingClassifier

Bagging vs Boosting

Bagging (Bootstrap Aggregating)

1. Orijinal veriden N rastgele alt örneklem çek (bootstrap)
2. Her alt örneklem üzerinde bağımsız model eğit
3. Modeller paralel çalışır — birbirinden habersiz
4. Tahminleri çoğunluk oyu ile birleştir
5. Varyansı düşürür, bias'ı değiştirmez

Boosting (Sıralı Öğrenme)

1. İlk model tüm veriyle eğitilir
2. Yanlış tahmin edilen örneklerin ağırlığı artırılır
3. Sonraki model hatalara odaklanır
4. Her model öncekinin hatasını düzeltmeye çalışır
5. Bias'ı düşürür, dikkatli olmazsan overfitting riski

Random Forest

Nasıl Çalışır?

Birden fazla karar ağacı (decision tree) eğitilir. Her ağaç farklı veri alt örneklemi ve farklı özellik alt kümesi kullanır. Son tahmin, tüm ağaçların çoğunluk oyu (sınıflandırma) veya ortalaması (regresyon) ile belirlenir.

100+

Ağaç Sayısı (varsayılan)

√p

Özellik Alt Kümesi

```
from sklearn.ensemble import RandomForestClassifier

# Model oluştur ve eğit
rf = RandomForestClassifier(
    n_estimators=200, max_depth=10,
    random_state=42, n_jobs=-1
)
rf.fit(X_train, y_train)
print(f"Accuracy: {rf.score(X_test, y_test):.3f}")
```

Random Forest Avantajları

1 Overfitting Direnci

Bagging ve özellik rastgeleliği sayesinde tekil karar ağaçlarına göre çok daha dayanıklı. Gürültülü veride bile güvenilir.

2 Feature Importance

Hangi özelliklerin tahmine en çok katkıda bulunduğunu ölçer. Veri bilimciler için güçlü yorumlama aracı.

3 Paralel Eğitim

Her ağaç bağımsız eğitilir → n_jobs=-1 ile tüm CPU çekirdeklerini kullanabilir. Büyük veride hız avantajı.

4 Çok Yönlülük

Sınıflandırma ve regresyon. Eksik veri toleransı. Hem kategorik hem sayısal özelliklerle çalışır.

Gradient Boosting

Adım Adım Hata Düzeltme

Gradient Boosting, her yeni modelin bir önceki modelin artıklarını (residuals) öğrenmesine dayanır. Bu, gradyan iniş (gradient descent) optimizasyonunun fonksiyon uzayındaki karşılığıdır. Learning rate ile adım büyüklüğü kontrol edilir.

1 — İlk Model

Basit tahmin (ortalama)

→

2 — Hataları Hesapla

Residuals: $y - \hat{y}$

→

3 — Hatalara Odaklan

Yeni ağaç residuals öğrenir

→

4 — Güncelle

$\hat{y} \mathrel{+}= \eta \times h(x)$ (learning rate)

→

5 — Tekrarla

n_estimators kadar devam

Learning Rate (η): Küçük değer → daha fazla ağaç gerekir ama daha iyi genelleme. Tipik aralık: 0.01–0.3

XGBoost

Neden En Popüler?

- Kaggle yarışmalarının %70+'unda kazanan model
- L1/L2 regularization ile overfitting kontrolü
- Eksik veri otomatik yönetimi (sparse aware)
- Paralel ağaç inşası → CPU çekirdeklerini verimli kullanır

%70+

Kaggle Kazananı

2014

İlk Sürüm Tianqi Chen

```python
import xgboost as xgb

# XGBoost modeli oluştur
model = xgb.XGBClassifier(
    n_estimators=300,
    learning_rate=0.1,
    max_depth=6,
    reg_alpha=0.1,    # L1 regularization
    reg_lambda=1.0,   # L2 regularization
    eval_metric='logloss'
)
model.fit(X_train, y_train)
```

LightGBM

Histogram Tabanlı Bölme

LightGBM, sürekli değerleri histogram kutularına (bins) böler. Bu sayede bölme noktası arama işlemi O(n) yerine O(bins) olur. Leaf-wise büyüme stratejisi ile daha derin ve etkili ağaçlar oluşturur. Microsoft Research tarafından 2017'de geliştirildi.

20x

Hız Farkı
vs GBM

1/5

Bellek
Kullanımı

Büyük Veri

Milyonlarca satır ve yüzlerce özellik ile verimli çalışır

Hız

XGBoost'tan 10-20x daha hızlı eğitim süresi

Kategorik Destek

Kategorik özellikleri doğrudan işler, encoding gerektirmez

4 Model Karşılaştırma

Kriter	Random Forest	GBM	XGBoost	LightGBM
Eğitim Hızı	Orta	Yavaş	Hızlı	Çok Hızlı
Doğruluk	İyi	İyi	Çok İyi	Çok İyi
Bellek Kullanımı	Yüksek	Orta	Orta	Düşük
Overfitting Riski	Düşük	Orta	Düşük	**Orta**
Büyük Veri Desteği	Orta	Zayıf	**İyi**	**Çok İyi**
Eksik Veri	Kısmen	Hayır	**Otomatik**	**Otomatik**
Paralel Eğitim	Evet	Hayır	**Evet**	**Evet**
Hiperparametre	Az	Çok	Orta	Orta

Sonuç: XGBoost genel amaçlı en iyi seçim. Büyük veri ve hız öncelikliyse LightGBM tercih edin.

Feature Importance

Neden Önemli?

Feature importance, modelin hangi özelliklere (sütunlara) en çok güvendiğini gösterir. Bu bilgiyle gereksiz özellikleri çıkarabilir, modeli sadeleştirebilir ve iş süreçlerini iyileştirebilirsiniz.

Gain	Özelliğin bölmelerdeki toplam bilgi kazancı
Weight	Özelliğin kaç bölmede kullanıldığı
Cover	Özelliğin etkilediği örnek sayısı

```
import xgboost as xgb
import matplotlib.pyplot as plt

# Feature importance grafiği çiz
xgb.plot_importance(
    model,
    importance_type='gain',
    max_num_features=10,
    title='En Önemli 10 Özellik'
)
plt.tight_layout()
plt.show()
```

Overfitting Problemi

Modelin En Büyük Düşmanı: Aşırı Uyum

Overfitting, modelin eğitim verisini ezberlemesi demektir. Eğitim setinde mükemmel performans gösterirken, yeni (görülmemiş) veride başarısız olur. Bu, modelin genelleme yeteneğini kaybettiği anlamına gelir.

Underfitting

Model çok basit
Hem eğitim hem test
hatası yüksek

Örn: max_depth=1

```
Eğitim: %60
Test: %58
```

Optimal Fit

Doğru karmaşıklık
Eğitim ve test
hatası dengeli

Örn: max_depth=6

```
Eğitim: %92
Test: %89
```

Overfitting

Model çok karmaşık
Eğitim mükemmel
test kötü

Örn: max_depth=50

```
Eğitim: %99
Test: %72
```

Regularization

L1 — Lasso Regularization

Ağırlıkların mutlak değerlerinin toplamını ceza terimi olarak ekler. Bazı ağırlıkları tamamen sıfırlar → otomatik özellik seçimi yapar.

XGBoost: reg_alpha parametresi
Formül: Loss + $\alpha \times \Sigma|w|$

L2 — Ridge Regularization

Ağırlıkların karelerinin toplamını ceza terimi olarak ekler. Büyük ağırlıkları küçültür ama sıfırlamaz → tüm özellikler kalır.

XGBoost: reg_lambda parametresi
Formül: Loss + $\lambda \times \Sigma w^2$

Ensemble Modellerde Neden Gerekli?

1 Karmaşıklık Kontrolü

Yüzlerce ağaç → milyonlarca parametre. Regularization olmadan model ezberlemeye meyillidir.

2 Genelleme Gücü

Ceza terimi, modeli basit ve genellenebilir tutarak test performansını artırır.

3 Özellik Seçimi (L1)

Gereksiz özelliklerin etkisini otomatik olarak sıfırlayarak modeli sadeleştirir.

Hiperparametre Optimizasyonu

GridSearchCV

1. Tüm parametre kombinasyonlarını dener
2. Kapsamlı ama yavaş (brute force)
3. Küçük parametre uzayları için ideal
4. Cross-validation ile değerlendirir
5. En iyi: best_params_, best_score_

RandomizedSearchCV

1. Rastgele parametre örnekleri dener
2. n_iter ile deneme sayısı kontrol edilir
3. Büyük parametre uzayları için ideal
4. Benzer sonuç, çok daha hızlı
5. Dağılım nesneleri kullanılabilir (scipy)

Kriter	GridSearchCV	RandomizedSearchCV
10 parametre × 5 değer	100.000 deneme	n_iter=50 deneme
Süre	Saatler/Günler	Dakikalar

GridSearchCV Uygulaması


```python
from sklearn.model_selection import GridSearchCV
from xgboost import XGBClassifier

# Parametre ızgarası tanımla
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [3, 5, 7],
    'learning_rate': [0.01, 0.1, 0.3],
    'subsample': [0.8, 1.0]
}

# GridSearchCV ile en iyi parametreleri bul
grid = GridSearchCV(
    XGBClassifier(eval_metric='logloss'),
    param_grid, cv=5, scoring='accuracy',
    n_jobs=-1, verbose=1
)
grid.fit(X_train, y_train)

print(f"En iyi parametreler: {grid.best_params_}")
print(f"En iyi skor: {grid.best_score_:.4f}")
```

Müşteri Terk Analizi (Churn)

BÖLÜM 4

Telco Customer Churn Dataset

Bir telekomünikasyon şirketinin 7.043 müşterisi. Amaç: Hangi müşteriler hizmeti bırakacak? XGBoost ile tahmin ve iş kararlarına yön verme.

7K+
Müşteri Kaydı

21
Özellik Sayısı

1
Veri Yükleme
CSV okuma, ilk inceleme

2
Ön İşleme
Encoding, ölçekleme, train/test

3
XGBoost Eğitim
Model fit, hiperparametre ayarı

4
Değerlendirme
Accuracy, Precision, Recall, F1

Accuracy	Precision	Recall	F1-Score	AUC-ROC
%81	%79	%56	%65	0.85

Model Seçim Stratejisi

1

1. Baseline

Basit model ile başlangıç skoru belirle (Logistic Regression / Decision Tree). Bu skor, iyileştirmenin referans noktasıdır.

2

2. Ensemble Modeller

Random Forest, XGBoost ve LightGBM'i varsayılan parametrelerle dene. Hangi aile daha iyi çalışıyor?

3

3. Hiperparametre Tune

En iyi aileyi GridSearchCV veya RandomizedSearchCV ile optimize et. Cross-validation kullan.

4

4. Karşılaştır

Accuracy, F1, AUC-ROC metrikleri ile modelleri değerlendir. Eğitim/test farkını kontrol et (overfitting).

5

5. Deploy

En iyi modeli joblib ile kaydet. Pipeline oluştur. Monitoring sistemi kur.

Haftalık Notebook'lar

1 ensemble_modeller.ipynb

Random Forest, Gradient Boosting, XGBoost ve LightGBM modellerinin temel kullanımı. Iris ve Wine dataset'leri üzerinde karşılaştırmalı analiz.

Random Forest | XGBoost | LightGBM | Karşılaştırma

2 musteri_terk.ipynb

Telco Churn dataset üzerinde uçtan uca proje. Veri temizleme, feature engineering, XGBoost ile tahmin, SHAP ile yorumlama.

Churn | XGBoost | SHAP | Pipeline

3 hiperparametre_avi.ipynb

GridSearchCV ve RandomizedSearchCV ile hiperparametre optimizasyonu. Learning curve analizi ve overfitting tespiti.

GridSearch | RandomSearch | Cross-Val | Tuning

Ödev ve Kaynaklar

Haftalık Ödev

1. XGBoost ile Telco Churn tahmin modeli kur → %85+ accuracy hedefi
2. GridSearchCV ile en az 3 hiperparametre optimize et
3. Feature importance grafiği çiz ve en önemli 5 özelliği yorumla
4. Random Forest vs XGBoost karşılaştırma raporu hazırla
5. Bonus: LightGBM ile aynı veriyi dene, hız farkını ölç

Teslim: Hafta 8 dersi öncesi · Format: Jupyter Notebook (.ipynb)

Kaynaklar

1. XGBoost Documentation
   xgboost.readthedocs.io
2. LightGBM Documentation
   lightgbm.readthedocs.io
3. scikit-learn Ensemble
   sklearn User Guide
4. Kaggle Learn:
   Intro to Machine Learning
5. Hands-On ML (Géron)
   Bölüm 7: Ensemble

Hafta 7 — Özet

1. Ensemble Learning: Birden fazla modelin birleşik gücü, tek modelden her zaman daha güçlüdür
2. Bagging varyansı düşürür (RF), Boosting bias'ı düşürür (XGBoost, LightGBM)
3. XGBoost: Regularization + paralel hesaplama ile Kaggle'ın en popüler modeli
4. Overfitting kontrolü: Regularization (L1/L2) ve cross-validation şart
5. Hiperparametre optimizasyonu model başarısını %5-15 artırabilir

“Tek bir ağaç devrilebilir, ama bir ormanı devirmek çok zordur.”