İLERİ DÜZEY UYGULAMALI

Makine Öğrenmesi

& Derin Öğrenme

HAFTA 1 · CUMARTESİ | Scikit-Learn Pipeline → Advanced RF → SHAP → Mini Proje

09:00 — 17:00 · 7 Saat Net Eğitim · 4 Uygulama Bloğu

Günün Planı

Hafta 1 · Cumartesi

09:00–09:45

Giriş & Kurulum

10:00–10:45

Veri Ön İşleme Teorisi

11:00–11:45

Pipeline & ColumnTransf.

12:00–13:00

🍽 Öğle Arası

13:00–13:45

Pipeline Lab

14:00–14:45

Random Forest + Tuning

15:00–15:45

SHAP Görselleştirme

16:00–16:45

Özet & Uygulama & Soru–Cevap

GÜN PLANI — Hafta 1 Cumartesi

Gün Sonu Hedef: Çalışan Pipeline + Eğitimli RF + SHAP Görselleştirmeleri

GÜN SONU ÇIKTILAR:

- ✓ Çalışan preprocessing pipeline
- ✓ SHAP görselleştirilmiş RF modeli
- ✓ Uçtan uca proje notebook'u

4	7	500+	3	6
Uygulama Bloğu	Saat Net Eğitim	Satır Python Kodu	Model Karşılaştırması	SHAP Grafik Türü

Gereksinimler: Python 3.10+ · scikit-learn 1.4+ · XGBoost · SHAP · pandas · matplotlib · seaborn

GELİŞMİŞ VERİ ÖN İŞLEME

Ham Veri Sorunları

Eksik Değerler (NaN)
Modeli eğitemez veya hatalı tahmin yapar

Farklı Ölçekler
Feature 1: [0–1] vs Feature 2: [0–100.000]

Kategorik Sütunlar
Model sayısal girdi bekler, string değil

Aykırı Değerler
Model eğitimini bozar, ağırlıkları çarpıtır

Data Leakage
Test verisinin eğitime sızması, yapay yüksek skor

Feature Engineering Eksikliği
Ham değişkenler yerine türetilmiş özellikler

Scikit-Learn Çözümleri

→ SimpleImputer / IterativeImputer
mean / median / most_frequent / constant stratejisi

→ StandardScaler / RobustScaler
z-score; aykırı değere dayanıklı IQR tabanlı

→ OneHotEncoder / OrdinalEncoder
nominal vs ordinal kategorik dönüşüm

→ RobustScaler + Clip
Aykırı değerleri sınırla, ölçekle

→ Pipeline içinde fit/transform
Eğitim ve test tutarlı dönüştürülür, leak yok

→ FunctionTransformer / Custom
Kendi dönüşüm mantığını pipeline'a entegre et

🛠 CUSTOM TRANSFORMER & FEATURE ENGINEERING

custom_transformer.py

```python
from sklearn.base import BaseEstimator, TransformerMixin
import numpy as np

class OutlierClipper(BaseEstimator, TransformerMixin):
    def __init__(self, lower=0.01, upper=0.99):
        self.lower = lower; self.upper = upper

    def fit(self, X, y=None):
        self.lower_ = np.quantile(X, self.lower, axis=0)
        self.upper_ = np.quantile(X, self.upper, axis=0)
        return self

    def transform(self, X):
        return np.clip(X, self.lower_, self.upper_)
```

⍰ BaseEstimator

get_params() / set_params() otomatik gelir; GridSearchCV ile uyumlu

🔄 TransformerMixin

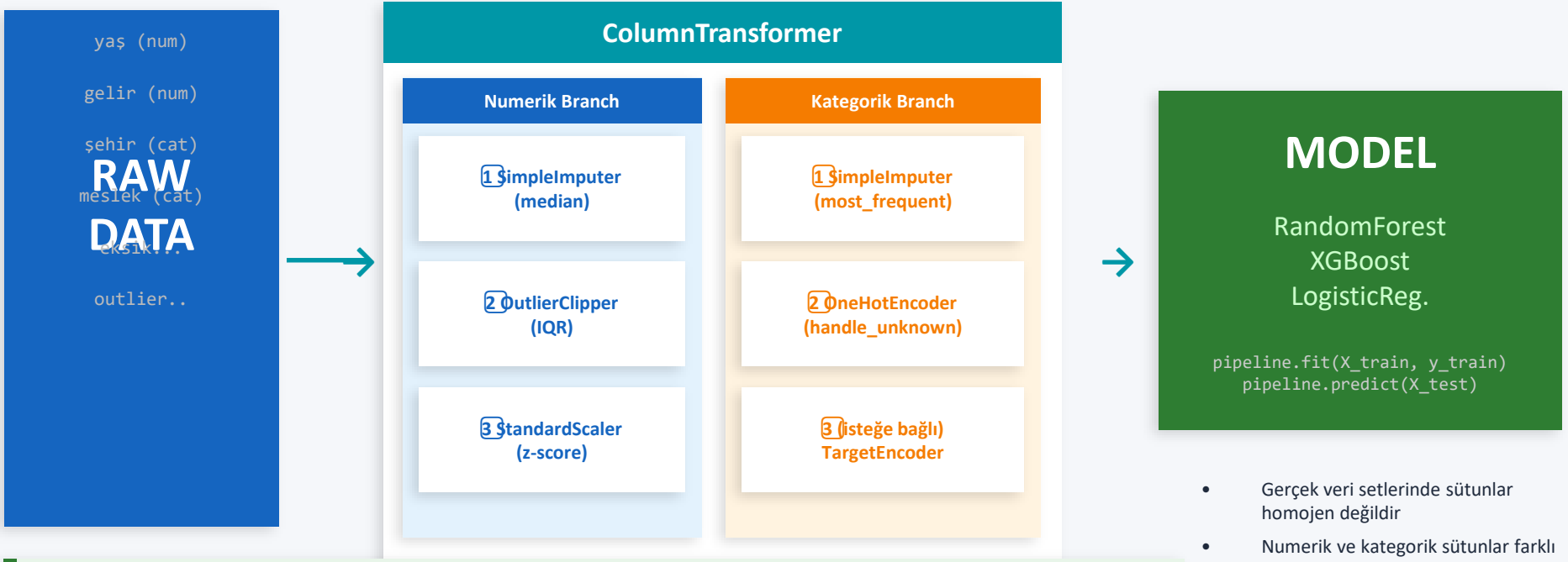
fit_transform() ücretsiz elde edilir; Pipeline ile sorunsuz çalışır

⚡ FunctionTransformer

💡 Pipeline Entegrasyonu

`OutlierClipper() → StandardScaler() → RF()`

SCIKIT-LEARN PIPELINE & COLUMNTRANSFORMER MİMARİSİ



Pipeline Avantajı:

fit() sadece X_train üzerinde çağrılır → transform() X_test üzerinde learnt parametrelerle çalışır → Data Leakage SIFIR!

- Gerçek veri setlerinde sütunlar homojen değildir
- Numerik ve kategorik sütunlar farklı işleme gerektirir
- Pipeline ile birleştirilince fit/transform otomatik yönetilir
- get_feature_names_out() ile feature adları takip edilir

full_pipeline.py — Scikit-Learn Pipeline + ColumnTransformer

```
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler, OneHotEncoder, RobustScaler
from sklearn.impute import SimpleImputer, IterativeImputer
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier

# --- 1. Kolon türlerini tanımla ---
num_cols = ['age', 'balance', 'duration', 'campaign', 'pdays', 'previous']
cat_cols = ['job', 'marital', 'education', 'contact', 'poutcome']

# --- 2. Numerik pipeline (gelişmiş) ---
numeric_pipeline = Pipeline([
    ('imputer', IterativeImputer(max_iter=10, random_state=42)),
    ('clipper', OutlierClipper(lower=0.01, upper=0.99)),
    ('scaler',  RobustScaler())
])

# --- 3. Kategorik pipeline ---
categorical_pipeline = Pipeline([
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('encoder', OneHotEncoder(handle_unknown='ignore', sparse_output=False))
])
```

RANDOM FOREST: İLERİ DÜZEY ANLAMA & HİPERPARAMETRE OPTİMİZASYONU

RF Anatomisi

Bootstrap Sampling
Her ağaç farklı veri örneği alır (%63.2)

Feature Subsampling
Her split'te max_features kadar özellik

Bagging → Paralel
Ağaçlar bağımsız; Boosting'den farklı

OOB Score
Out-of-bag, ücretsiz validation tahmincisi

Gini vs Entropy
Criterion: bölünme kalitesi metrikleri

min_samples_leaf
Overfitting kontrolü için kritik parametre

GridSearchCV + Cross-Validation Stratejisi

Parametre	Denenen Değerler	Ne Anlama Gelir?
`n_estimators`	`100, 200, 500`	Fazla ağaç = daha az variance, ama yavaş
`max_depth`	`None, 10, 20, 30`	None = tam büyüme; overfitting riski
`min_samples_split`	`2, 5, 10`	Bölünme için min örnek; yüksek → underfitting
`min_samples_leaf`	`1, 2, 4`	Yaprak min örneği; regularizasyon etkisi
`max_features`	`sqrt, log2, 0.5`	sqrt = √p tercih edilir sınıflandırmada
`class_weight`	`None, balanced`	Dengesiz veri için balanced kullan

Model Değerlendirme & GridSearchCV

Cross-Validation ile Güvenilir Hiperparametre Seçimi

StratifiedKFold(n_splits=5)

Her fold'da bir bölüm validation, kalanlar eğitim. 5 farklı skor alınır, ortalaması raporlanır.

GridSearchCV Akışı

- param_grid tanımla: denenecek değerler sözlüğü
- Her kombinasyon için CV skoru hesapla
- En yüksek CV skorlu parametreleri seç
- Tüm X_train üzerinde en iyi modeli yeniden eğit

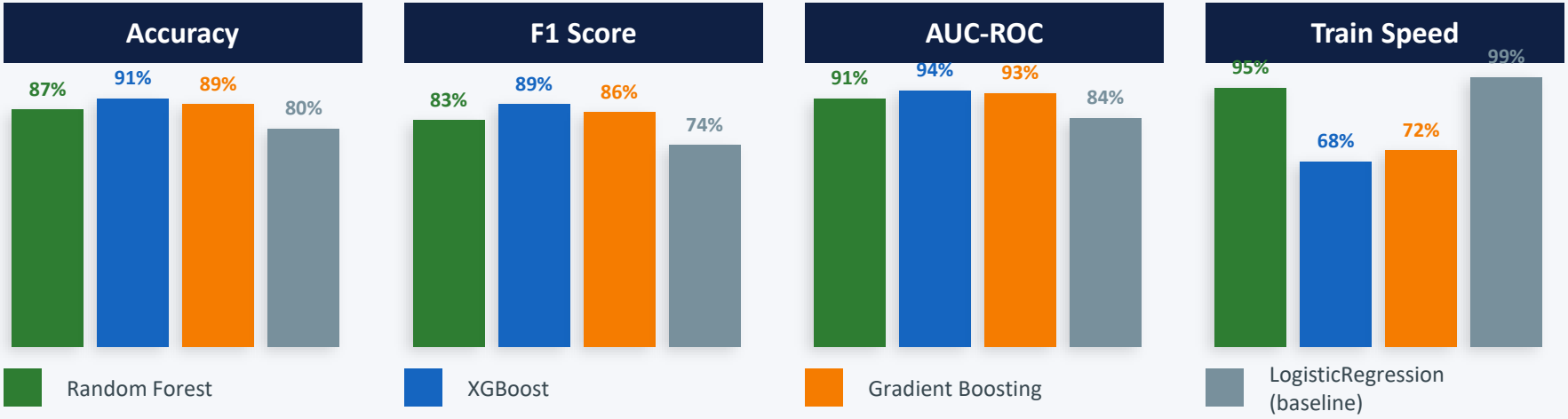
gridsearch.py

```
from sklearn.model_selection import GridSearchCV, StratifiedKFold

param_grid = {
    'model__n_estimators': [50, 100, 200],
    'model__max_depth':    [None, 10, 20],
    'model__max_features': ['sqrt', 'log2']
}

cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
gs = GridSearchCV(full_pipeline, param_grid, cv=cv, scoring='roc_auc', n_jobs=-1)
gs.fit(X_train, y_train)
print('En iyi params:', gs.best_params_)
print('CV AUC:', gs.best_score_.round(4))
```

⚔ MODEL KARŞILAŞTIRMA: Random Forest vs XGBoost vs GradientBoosting



🏆 XGBoost En Yüksek Skor — AUC 0.94 ile lider; ancak tuning hassasiyet ister

⚡ RF Hız+Kalite Dengesi — Varsayılan parametrelerle bile güçlü baseline sağlar

📊 CV ile Güvenilir Karşılaştırma — StratifiedKFold(n_splits=5) ile leak-free değerlendirme

SHAP: SHapley Additive exPlanations — Model Kararlarını Anlamak

Oyun Teorisi Temelli

Her özelliğin modelin tahminindeki katkısını adil bir şekilde paylaştırır. Shapley değerleri garanti özellikler: Etkililik, Simetri, Kukla, Additivity.

$$\varphi(x_i) = \phi_0 + \Sigma \text{ SHAP(feature j)} = f(x)$$

TreeExplainer
RF, XGBoost için; $O(TL^2D)$ hız

KernelExplainer
Model-agnostik; herhangi ML modeli

LinearExplainer
Linear modeller için optimize

GradientExplainer
Deep learning için

6 Temel SHAP Görselleştirmesi

1 summary_plot — *Global feature importance*
→ Tüm örnekler için feature sıralaması; bee-swarm

2 bar_plot — *Ortalama |SHAP|*
→ En etkili özellikleri sıralar

3 force_plot — *Bireysel tahmin*
→ Tek örnek için pozitif/negatif katkı

4 waterfall_plot — *Kademeli katkı*
→ Temel değerden tahmine yolculuk

5 dependence_plot — *Etkileşim analizi*
→ Feature değeri vs SHAP + renk=interaction

6 heatmap — *Toplu karşılaştırma*
→ Birden fazla özellik / örneği aynı anda

shap_analysis.py — Tam SHAP Analizi Kodu

```python
import shap
import matplotlib.pyplot as plt

# 1. Explainer oluştur (TreeExplainer = hızlı)
explainer = shap.TreeExplainer(best_rf_model)
shap_values = explainer.shap_values(X_test)

# 2. Global: Summary Plot (bee-swarm)
shap.summary_plot(shap_values[1], X_test,
                  feature_names=feature_names,
                  max_display=15)

# 3. Global: Bar Plot (ortalama |SHAP|)
shap.summary_plot(shap_values[1], X_test,
                  plot_type='bar', max_display=15)

# 4. Interaction: Dependence Plot
shap.dependence_plot('duration', shap_values[1],
                     X_test, interaction_index='age')

# 5. Local: Force Plot (tek örnek)
idx = 42  # analiz edilecek örnek
shap.force_plot(
    explainer.expected_value[1],
    shap_values[1][idx],
    X_test.iloc[idx],
    matplotlib=True
)

# 6. Local: Waterfall Plot
shap.waterfall_plot(shap.Explanation(
    values=shap_values[1][idx],
    base_values=explainer.expected_value[1],
    data=X_test.iloc[idx],
    feature_names=feature_names
))

# 7. SHAP Interaction Values (gelişmiş)
shap_interaction = explainer.shap_interaction_values(X_test[:100])
```

MİNİ PROJE: Bank Marketing Churn Prediction — Uçtan Uca

ADIM 1 — Veri Yükleme & EDA

- UCI Bank Marketing veri seti (45.211 satır)
- Eksik değer & dağılım analizi
- Class imbalance kontrolü (%88 No / %12 Yes)

ADIM 2 — Preprocessing Pipeline

- IterativeImputer + OutlierClipper + RobustScaler
- OneHotEncoder kategorik sütunlar için
- ColumnTransformer ile birleştirme

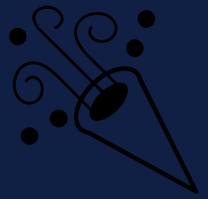
ADIM 3 — Model Eğitimi & Karşılaştırma

- RF, XGBoost, GradientBoosting pipeline'a ekle
- StratifiedKFold(5) cross-validation
- class_weight='balanced' ile dengesiz veri çözümü

ADIM 4 — SHAP Analizi

- summary_plot ile en kritik 15 feature
- force_plot ile 3 farklı müşteri profili
- dependence_plot: 'duration' vs 'age' etkileşimi

Çıktılar: best_pipeline.pkl · shap_summary.png · shap_force_plot.html · model_comparison.csv



HAFTA 1 CUMARTESİ TAMAMLANDI!

Bugün Öğrendikleriniz

- ✓ Custom Transformer & Advanced Pipeline kurma
- ✓ IterativeImputer + RobustScaler + OutlierClipper
- ✓ RF Hiperparametre optimizasyonu + CV
- ✓ XGBoost karşılaştırma & model seçimi
- ✓ SHAP ile 6 farklı görselleştirme
- ✓ Uçtan uca Bank Marketing projesi

Yarın: Keras Sequential API + EarlyStopping + Dropout