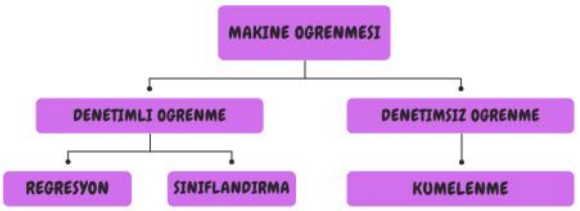
12. Ders - Makine Öğrenmesi

17 Eylül 2025 Çarşamba 09:22



Bu ders kapsamında yanda yer alan başlıkları inceleyeceğiz.

- **Veri Kalitesi > Model Karmaşıklığı :**
  - Temiz, tutarlı ve iyi etiketlenmiş veriler **Yüksek Kaliteli Veri** kapsamında değerlendirilir.
  - Gürültülü, eksik veya yanlış veriler ise **Düşük Kaliteli Veri** kapsamında değerlendirilir.
    - Veri kalitesi düşükse modeli karmaşıklaştırmak problemi çözmez, genellikle daha da kötüleştirir. Öncelik her zaman veriyi iyileştirmek ve temizlemek olmalı, ardından ihtiyaç duyulursa model karmaşıklığı artırılmalıdır.

- **Feature Drift** (Öznitelik Kayması), Modeli eğitmek için kullandığımız eğitim verisi ile modelin canlı (production) ortamda gördüğü giriş verisi (input features) arasındaki dağılımın (istatistiksel özelliklerin) zamanla değişmesidir.
  - Örnek : Bir sensörün (ısı, nem vb.) zamanla kalibrasyonu bozulur ve okuduğu değerler sistematik olarak kayar.

Imbalance Data (Dengesiz Veri)

- Dengesiz Veri (Imbalanced Data), makine öğrenmesinde, özellikle de sınıflandırma problemlerinde karşılaşılan temel bir sorundur. Bu durum, bir veya daha fazla sınıfa (kategoriye) ait örnek sayısının, diğer sınıflara kıyasla çok daha az olduğu veri setlerini tanımlar.
  - Örneğin, 10.000 kredi başvurusundan sadece 100'ünün sahtekarlık olduğu bir veri setinde, sahtekarlık sınıfı (%1) azınlık sınıfıyken, güvenilir başvurular sınıfı (%99) çoğunluk sınıfını oluşturur.
- SMOTE, class weighting, anomaly detection gibi yöntemler bu başlık altında kullanılır.

Feature Importance & Explainability

- Feature Importance (Öznitelik Önemi), bir makine öğrenmesi modelinin tahminlerinde hangi girdi değişkenlerinin (feature'ların) daha etkili ve belirleyici olduğunu ölçen bir kavramdır.
  - Modelin karar verme sürecinde değişkenlere biçtiği nispi önceliği veya ağırlığı nicel olarak gösterir.
  - Örneğin, bir ev fiyat tahmin modelinde "metrekare" özniteliği, "bina yaşı" özniteliğinden daha yüksek bir öneme sahip olabilir.

- Explainability, Sadece hangi özniteliklerin önemli olduğunu değil, her bir tahmin için bu özniteliklerin model kararını nasıl etkilediğini açıklamaktadır.

- Black box model kurgulamak yeterli değil.

Model Drift Takibi

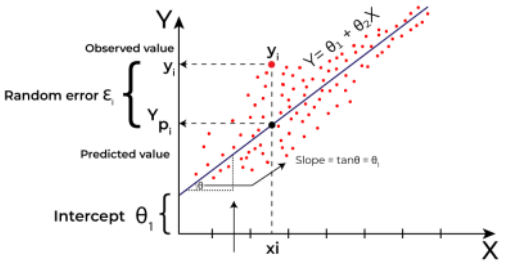
- Model drift (model kayması), zaman içinde gerçek dünya verilerinin dağılımının veya ilişkilerin değişmesi sonucu, bir makine öğrenmesi modelinin performansının ve doğruluğunun sessizce azalmasıdır.

- Drift Detection Teknikleri uygulanmalı. (PSI, KL Divergence)

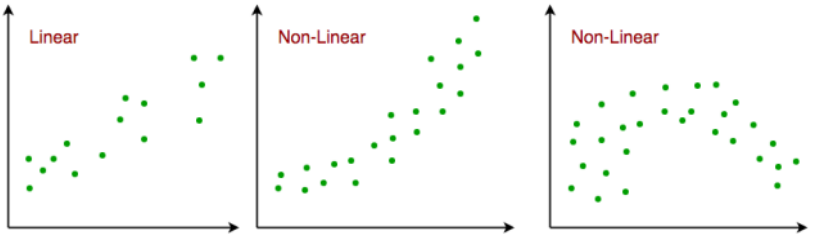
LİNEER REGRESYON

Lineer Regresyon, bir bağımlı değişken (hedef) ile bir veya daha fazla bağımsız değişken (öznitelik) arasındaki doğrusal ilişkiyi modellemek için kullanılan istatistiksel ve makine öğrenmesi yöntemidir.

$$Y = f(x) = x0 + a1.x1 + a2.x2 + \ldots + an.xn$$



Regresyon çeşitlerini görselleştirmek için aşağıdaki görsel daha anlaşılabilir olacaktır.



- Penalty (L1 | L2) -> L1 (Lasso): Feature Selection. **"Hangi feature'lar gereksiz? → Onları at." (Seçim yapar)**
  ->L2 (Ridge): Katsayı Küçültme . **"Tüm feature'ları tut, ama hepsinin etkisini küçült." (Küçültme yapar)**

- Solver -> Bir makine öğrenmesi modelinin (özellikle doğrusal modellerde) eğitimi sırasında en uygun parametreleri (ağırlıkları) bulmak için kullanılan optimizasyon algoritmasıdır. **Kayıp fonksiyonunu (loss function) minimize edecek parametreleri iteratif bir şekilde arar.**
  - **Örnek :** Müşteri churn analiz tahmini.

- C (Inverse Regularization) -> C, bir modelde regularization (düzenlileştirme) gücünü kontrol eden hiper parametredir. "Inverse of regularization strength" olarak tanımlanır, **yani regularization'ın tersidir.**
  - Sabit bir değerdir. **C = 1/ λ**
  - Genellikle çapraz doğrulama (cross-validation) ile seçilir.

OPTUNA TEKNİĞİ

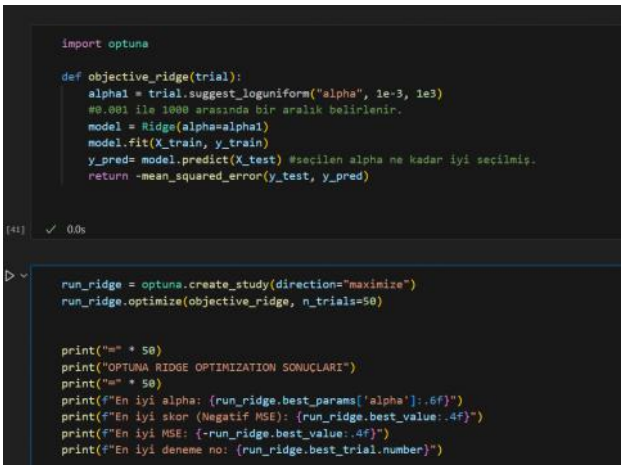
Optuna, makine öğrenmesi modelleri için **hiperparametre optimizasyonu** yapan bir Python kütüphanesidir.

Temel Özellikleri:

- Akıllı Arama: Grid veya Random Search'ten daha verimli çalışan Bayesian Optimizasyon kullanır.
- Esnek Yapı: Çeşitli ML framework'leri (Scikit-learn, TensorFlow, PyTorch) ile uyumludur.
- Görselleştirme: Optimizasyon sürecini görselleştirme imkanı sunar.

Nasıl Çalışır?

Denenecek hiperparametreleri "çalışma" (study) içinde tanımlar. Her denemede (trial), Optuna performansı ölçer ve bir sonraki en umut vadeden parametreleri otomatik seçer.



```python
import optuna

def objective_ridge(trial):
    alpha1 = trial.suggest_loguniform("alpha", 1e-3, 1e3)
    #0.001 ile 1000 arasında bir aralık belirlenir.
    model = Ridge(alpha=alpha1)
    model.fit(X_train, y_train)
    y_pred= model.predict(X_test) #seçilen alpha ne kadar iyi seçilmiş.
    return -mean_squared_error(y_test, y_pred)
```

[41] ✓ 0.0s

```python
run_ridge = optuna.create_study(direction="maximize")
run_ridge.optimize(objective_ridge, n_trials=50)


print("=" * 50)
print("OPTUNA RIDGE OPTIMIZATION SONUÇLARI")
print("=" * 50)
print(f"En iyi alpha: {run_ridge.best_params['alpha']:.6f}")
print(f"En iyi skor (Negatif MSE): {run_ridge.best_value:.4f}")
print(f"En iyi MSE: {-run_ridge.best_value:.4f}")
print(f"En iyi deneme no: {run_ridge.best_trial.number}")
```

->

```
==================================================
OPTUNA RIDGE OPTIMIZATION SONUÇLARI
==================================================
En iyi alpha: 0.001025
En iyi skor (Negatif MSE): -0.2136
En iyi MSE: 0.2136
En iyi deneme no: 45
{'alpha': 0.001025476545830268}
```

Sonuca göre iki çıkarım yapılır. Ya veri seti çok temiz. Ya da elimizde çok fazla özellik (feature) yok.

- Veri setinizde çok az parametre, küçük bir probleminiz varsa (birkaç yüz kombinasyon) GridSearchCV tercih edilebilir.
- Veri seti büyükse, aralık genişse ve çok parametre varsa RandomizedSearchCV tercih edilir.
- Hesap pahalı / verimli denemesi istersen Bayesian Opt (Optuna) tekniği tercih edilir.
- Deep Learning / pahalı deneyler, paralel trials varsa Population-Based Training veya Optuna + Pruner tercih edilir.
- Multi-objective : Bazı durumlarda sadece doğru opt. değil, hız ve hafıza kıllanımı da önemli olursa bu hiperparametre opt. tekniğini tercih ederiz.