

11. Ders

16 Eylül 2025 Salı 09:15

Özellik Mühendisliği: Özellik mühendisliği, ham veriyi bir makine öğrenimi modeli için daha anlamlı ve etkili özelliklere (features) dönüştürme sürecidir. Modelin performansını ve doğruluğunu artırmayı hedefler.

Gradyan tabanlı optimizasyon, bir fonksiyonun minimum veya maksimum noktasını, fonksiyonun gradyanının (eğim) yönünü takip ederek bulan bir yöntemdir. Gradyan Tabanlı Optimizasyonda (SGD, Adam) dengesiz adımlar büyük problemler yaratır. Bazı parametreler çok hızlı güncellenir. Bazıları da hiç öğrenmez.

- İncele : <https://medium.com/intuition/understanding-l1-and-l2-regularization-with-analytical-and-probabilistic-views-8386285210fc>

General regularization formulation

$$\mathcal{L}(w) + \lambda R(w)$$

λ : Regularization strength
 w : the model parameters or weights
 $\mathcal{L}(w)$: Objective function or loss function
 $R(w)$: Regularization term or regularizer or penalty

Ex) In the previous example case, we can describe the above formula as:

$$\mathcal{L}(\beta) + \lambda R(\beta) = \frac{1}{n} \sum_{i=1}^n \left(y_i - \left(\sum_{p=0}^{15} \beta_p x_i^p \right) \right)^2 + \lambda \sum_{p=0}^{15} \|\beta_p\|^2$$

Least mean square error
※ the error between the original and predicted data

Regularization

- **λ (lambda)**: Düzenleme (regularization) katsayısıdır. Ceza teriminin ($R(w)$) ne kadar güçlü uygulanacağını belirler.
- **w** : Modelin parametreleri/ağırlıklarıdır. Tahmin yapan katsayılardır.

Düzenleştirme adımında scale edilmemiş bir özellik olursa cezası adaletsiz olur.

Mesafe Tabanlı Algoritmalarda (KNN, SVM, K-Means, PCA) uzay geometrisi bozulur. Büyük aralıklar daha baskın hale gelir.

Ölçeklendirme (Scaling), Normalizasyon, Standartlaştırma

- **Mutlak Maksimum Ölçeklendirme (MaxAbs)**

- X_i : Ham verideki i. gözlem/değer.
- X_{max} : Veri kümesindeki en büyük değer.
- $|X_{max}|$: Bu en büyük değerın mutlak değeri.
- X'_i : Normalleştirilmiş yeni değer.

$$X'_i = \frac{X_i}{abs(X_{max})}$$

Her bir değeri (X_i/X_{max}) **en büyük değerin mutlak değerine böleriz**. Böylece tüm veriler $[-1,1]$ aralığına sıkıştırılır.

Özellikleri:

- Eğer tüm değerler pozitifse → sonuç 0 ile 1 arasında olur.
- Eğer negatif değerler varsa → sonuç -1 ile 1 arasında dağılır.
- Outlier (uç değer) varsa, tüm veriler o değere göre ölçeklendiği için etkilenir.

Yani bu yöntem, **maksimum mutlak değer ölçekleme (max-abs scaling)** olarak bilinir.

Sparse veriler (Seyrek veriler) (TD-IDF, Bag of Words) geldiğinde en ideal teknik.

- ♦ Mesela arama motorları verileri sparse verileridir.

Outlier yüzünden çoğu continuous numeric future'da güvenilmez.

Silikon vadisinde RobustScaler + Clipping veya Quantile Transform ile outlier'lar temizlendikten sonra uygulanır. Bu işlemlerin kütüphaneleri var. Fakat biz bu eğitimde pandas üzerinden direkt formülleri yazacağız.

```
from sklearn.preprocessing import MaxAbsScaler, StandardScaler, RobustScaler
```

MaxAbs işlemini manuel olarak kod ile şu şekilde yazdık :

```
import pandas as pd
import numpy as np

data = {
    "RPM" : [800, 1500, 2000, 2200, 8000],
    "Engine_temp" : [70, 85, 90, 95, 300],
    "Oil_pressure" : [30, 35, 40, 38, 150]
}

df = pd.DataFrame(data)

def maxabs_scaler(X):
    X = X.astype(float)
    max_abs = np.abs(X).max(axis=0) #her sütundaki mutlak maksimumu hesaplamış olur axis=0 ile beraber
    return X / max_abs

df_max_abs = maxabs_scaler(df)

print("MaxAbs Scaler Öncesi Veri : \n", df)
print("MaxAbs Scaler : \n", df_max_abs)
```

MaxAbs Scaler Öncesi Veri :			
	RPM	Engine_temp	Oil_pressure
0	800	70	30
1	1500	85	35
2	2000	90	40
3	2200	95	38
4	8000	300	150

MaxAbs Scaler :			
	RPM	Engine_temp	Oil_pressure
0	0.1000	0.233333	0.200000
1	0.1875	0.283333	0.233333
2	0.2500	0.300000	0.266667
3	0.2750	0.316667	0.253333
4	1.0000	1.000000	1.000000

- StandardScaler

$$z = \frac{x - \mu}{\sigma}$$

μ = Mean
 σ = Standard Deviation

StandardScaler, bir veri setindeki sayısal özellikleri ortalama (mean) 0 ve standart sapma (std) 1 olacak şekilde standardize eden bir ölçeklendirme yöntemidir.

Bu başlığın manuel olarak kod ile yazılımı şu şekildedir :

```
def standard_scaler(X: pd.DataFrame) -> pd.DataFrame :
    mean = X.mean(axis=0)
    std = X.std(axis=0)
    scaled = (X - mean) / std
    return scaled, mean, std

df_std, mean_val, std_val = standard_scaler(df)

print("Elimizdeki veri seti :\n", df)
print("\nStandard Scaling :\n", df_std)
print("\nKullanılan mean :\n", mean_val)
print("\nKullanılan std değeri :\n", std_val)
```

Elimizdeki veri seti :			
	RPM	Engine_temp	Oil_pressure
0	800	70	30
1	1500	85	35
2	2000	90	40
3	2200	95	38
4	8000	300	150

Standard Scaling :			
	RPM	Engine_temp	Oil_pressure
0	-0.723708	-0.600384	-0.558236
1	-0.482472	-0.445112	-0.460642
2	-0.310160	-0.393355	-0.363049
3	-0.241236	-0.341598	-0.402086
4	1.757576	1.780449	1.784013

Kullanılan mean :	
RPM	2900.0
Engine_temp	128.0
Oil_pressure	58.6
dtype: float64	

Kullanılan std değeri :	
RPM	2901.723626
Engine_temp	96.604865
Oil_pressure	51.232802
dtype: float64	

Değerleri yorumladığımızda :

- Ortalamaya göre bir dağılım yapıyor. Fakat elimizdeki "8000" değeri ortalamayı epey şişiriyor.
- Aykırı değerler, StandardScaler uygulandığında bile normalleşmedi.
- Bu tür aykırı değerlerle çalışırken StandardScaler yerine RobustScaler (medyan ve IQR kullanır) daha iyi sonuç verebilir, çünkü aykırı değerlere karşı daha dirençlidir.

- RobustScaler

StandardScaler'dan farklı olarak aykırı değerlere (outliers) karşı daha dayanıklıdır.

$$X_{new} = \frac{X - X_{median}}{IQR}$$

1. X: Orijinal veri noktası (ham değer).
2. median(X): Veri setinin medyanı (ortadaki değer). Ortalama (mean) yerine medyan kullanılır çünkü aykırı değerlerden etkilenmez.
3. IQR: Interquartile Range (Çeyrekler Arası Aralık).
 - IQR = Q3 (üçüncü çeyrek) - Q1 (birinci çeyrek)
 - Verinin merkezi %50'sinin yayılımını ölçer. Standart sapmaya kıyasla aykırı değerlere karşı çok daha dirençlidir.

Kodumuz şu şekildedir:

```
def robust_scaler(X):
    median = X.median(axis=0)
    q1 = X.quantile(0.25)
    q3 = X.quantile(0.75)
    iqr = q3 - q1
    scaled = (X - median) / iqr
    return scaled, median, q1, q3, iqr

df_robust, med_val_robust, q1_val, q3_val, iqr_val = robust_scaler(df)

print("Elimizdeki veri seti :\n", df)
print("Robust Scaling:\n", df_robust)
print("\nMedyan değeri:\n", med_val_robust)
print("\nQ1 değeri:\n", q1_val)
print("\nQ3 değeri:\n", q3_val)
print("\nIQR değeri:\n", iqr_val)
```

```
Elimizdeki veri seti :
  RPM  Engine_temp  Oil_pressure
0   800           70           30
1  1500           85           35
2  2000           90           40
3  2200           95           38
4  8000          300          150

Robust Scaling:
  RPM  Engine_temp  Oil_pressure
0 -1.714286       -2.0       -1.6
1 -0.714286       -0.5       -0.6
2  0.000000        0.0        0.4
3  0.285714        0.5        0.0
4  8.571429       21.0       22.4

Medyan değeri:
  RPM  Engine_temp  Oil_pressure
2000.0      90.0      38.0
dtype: float64

Q1 değeri:
  RPM  Engine_temp  Oil_pressure
1500.0      85.0      35.0
...
  RPM  Engine_temp  Oil_pressure
700.0      10.0       5.0
dtype: float64
```

Kaggle'da yer alan <https://www.kaggle.com/datasets/parvmodi/automotive-vehicles-engine-health-dataset> veri seti üzerinden derste işlemimize devam ettik.

```

numeric_cols = ["Engine rpm", "Lub oil pressure", "Fuel pressure", "Coolant pressure", "lub oil temp", "Coolant temp", "Engine Condition"]

for col in numeric_cols:
    df[col] = pd.to_numeric(df[col], errors='coerce')
    #tüm sayısal sütunları numeric şekilde alacak. Eksik değerleri de alacak
    #eksik değerleri dolduralım aşağıda :
    if df[col].isna().any():
        med = df[col].median()
        df[col].fillna(med, inplace=True)

mins = df[numeric_cols].min(axis=0)
maxs = df[numeric_cols].max(axis=0)

def min_max_scale(X: pd.DataFrame, mins: pd.Series, maxs: pd.Series, feature_range = (0,1)) -> pd.DataFrame:
    #min sıfır max bir olacak şekilde ayarlamış olduk
    X = X.copy().astype(float)
    min_range, max_range = feature_range
    denom = (maxs-mins).replace(0, 1.0)
    X_scaled = (X - mins) / denom
    return X_scaled * (max_range - min_range) + min_range

print(min_max_scale(df,mins, maxs))

```

Kodumuzun çıktısı şu şekilde oldu :

	Engine rpm	Lub oil pressure	Fuel pressure	Coolant pressure
0	0.293388	0.342901	0.557732	0.424891
1	0.374197	0.404592	0.766055	0.329322
2	0.210744	0.407365	0.309909	0.142036
3	0.189164	0.510102	0.922965	0.498256
4	0.256198	0.780693	0.744527	0.274179
...
19530	0.386134	0.566484	0.235540	0.581069
19531	0.290634	0.662933	0.514002	0.827205
19532	0.286042	0.367653	0.232986	0.254292
19533	0.291552	0.425599	0.392173	0.163088
19534	0.203398	0.519384	0.187332	0.272359

	lub oil temp	Coolant temp	Engine Condition
0	0.702246	0.149109	1.0
1	0.346077	0.155186	0.0
2	0.352175	0.134268	1.0
3	0.153785	0.075465	1.0
4	0.387485	0.189212	0.0
...
19530	0.253557	0.196122	1.0
19531	0.216852	0.099026	1.0
19532	0.302482	0.184260	1.0
19533	0.320816	0.089284	1.0
19534	0.232345	0.140063	1.0

[19535 rows x 7 columns]