Woman in Data Science

Lecture 2

29.11.2025

Veri Ön İşleme (Preprocessing)

Veri ön işleme, ham veriyi makine öğrenmesi ve analiz için uygun hale getirme sürecidir.

Gerçek hayattaki veriler genellikle eksik, düzensiz ve ölçekleri farklıdır.

Bu yüzden veriyi modele vermeden önce düzenlememiz gerekir.

Veri Ön İşleme (Preprocessing)

Veri Ön İşleme (Preprocessing)

Missing Values • Outliers • Encoding • Scaling • Sampling

Amaç: Daha iyi tahmin yapan, daha sağlıklı modeller oluşturmak.

Veri Ön İşleme (Preprocessing)

Dersin Hedefleri

Slayt maddeleri:

- **Eksik değerleri bulma & doldurma**
- **Aykırı değer tespiti & işleme**
- **Kategorik veriyi sayıya çevirme (encoding)**
- **Ölçekleme: normalizasyon / standardizasyon**
- **Sampling: under- & over-sampling**
- **Uygulama: Stroke Prediction veri seti**

Neden Veri Temizliyoruz?

Gerçek veri → mükemmel değildir.

❗ Eksik değerler modelin öğrenmesini bozar

❗ Kategorik veriler modele girmez → sayıya çevirmek gerekir

❗ Değer aralıkları farklıysa büyük sayılar modeli yanıltabilir

❗ Sınıf dağılımı dengesizse model az olan sınıfı görmezden gelebilir

Bu yüzden veri ön işleme başarılı model kurmanın ilk adımıdır.

Ön İşlemenin Mantığı (Neden sıralama önemli?)

Maddeler:

1. **Önce veriyi tanı (info, describe)**
2. **Eksikler → doldur (Missing)**
3. **Aykırı değerleri tespit et & karar ver**
4. **Kategorik → sayısal (Encoding)**
5. **Ölçekle (Scaling)**
6. **Sampling → son adım (Under/Over)**

Her adım diğerini etkiler. Örneğin eksik veriyi doldurmadan encode/scale yaparsak hata alabiliriz. Aykırı değeri scale ettikten sonra düzeltemezsin; çünkü outlier ölçeklemeyi bozar. Sampling ise verinin son haliyle yapılmalı.

Veri İlk Bakışı: info(), head(), describe()

Maddeler:

- df.head() → ilk satırlar
- df.info() → veri tipleri ve boş sayıları
- df.describe() → sayısal özet

Missing Values (Nedir?)

Maddeler:

- NaN, None, boş hücreler
- Neden: ölçüm hatası, veri girilmeme, sistem hatası
- Etkileri: model hataları, bias

Eksik veri modelin öğrenmesini bozabilir. Eğer çoksa satırı düşürürüz; azsa doldururuz. Doldurma yöntemi veri tipine ve dağılıma göre seçilir.

Missing: Doldurma Yöntemleri (Uygulama)

Maddeler:

- dropna() — silme
- fillna(mean/median/mode)
- SimpleImputer / IterativeImputer (ileri)

🔥 Missing Value Handling — Hangi Yöntem Ne Zaman?

Eksik veri yaklaşımını seçerken 3 temel soruyu düşünürüz:

1) Eksiklik oranı ne kadar? → %1 mi? %30 mu?
2) Hangi değişkende eksik? → Sayısal mı kategorik mi?
3) Veri neden eksik? (MNAR/MAR/MCAR) → Sistematik mi rastgele mi?

Bu üçüne göre yöntem değişir.

🔹 A) Eksiklik oranına göre yaklaşım

Eksik oranı	Önerilen yaklaşım	Neden?
< %5	Basit doldurma (mean/median/mode)	Dağılım çok bozulmaz
%5 - %20	İstatistiksel doldurma, KNN/Regression Imputation	Veri önemli, kaybetmemeliyiz
> %20	Silme düşünülür + domain bilgisiyle karar	Tahmin zor → veri güvenilmez
> %40	Genellikle **drop variable**	Sinyalden çok gürültü kalır

📌 Kural: Eksik oran yükseldikçe basit doldurma daha riskli hale gelir.

🔹 B) Değişken türüne göre yöntem seçimi

Veri tipi	En yaygın yöntem	Kullanım alanı
Sayısal (age, bmi, glucose)	Mean / Median / KNN / Regression	Dağılıma göre seçim yapılır
Kategorik (gender, smoking_status)	Mode / "Unknown" / One-hot + NaN	Frekans tabanlı doldurma daha güvenli

🔹 B) Değişken türüne göre yöntem seçimi

Veri tipi	En yaygın yöntem	Kullanım alanı
Sayısal (age, bmi, glucose)	Mean / Median / KNN / Regression	Dağılıma göre seçim yapılır
Kategorik (gender, smoking_status)	Mode / "Unknown" / One-hot + NaN	Frekans tabanlı doldurma daha güvenli

🔥 *Mean mi Median mı? Nasıl Seçilir?*

Durum	Mean (Ortalama)	Median (Ortanca)
Dağılım *normal* ise	✔ Uygun	✔ Uygun
Aykırı değer (outlier) yoksa	✔ Uygun	✔ Uygun
Aykırı değer varsa	❌ Dağılımı bozar	✔ Daha sağlam bir seçim
Dağılım çarpıksa (sağa/sola kaymışsa)	❌ Riskli	✔ Genellikle en doğru seçenek

📌 **Özet:**

Outlier veya çarpık dağılım varsa **Median**,

Temiz ve normal dağılımlı veride **Mean** tercih edilir.

🔥 *Mean mi Median mı? Nasıl Seçilir?*

Durum	Mean (Ortalama)	Median (Ortanca)
Dağılım *normal* ise	✔ Uygun	✔ Uygun
Aykırı değer (outlier) yoksa	✔ Uygun	✔ Uygun
Aykırı değer varsa	❌ Dağılımı bozar	✔ Daha sağlam bir seçim
Dağılım çarpıksa (sağa/sola kaymışsa)	❌ Riskli	✔ Genellikle en doğru seçenek

📌 **Özet:**

Outlier veya çarpık dağılım varsa **Median**,

Temiz ve normal dağılımlı veride **Mean** tercih edilir.

🔥 Kategorikte seçim nasıl?

Durum	Yöntem
Kayıplar azsa	Mode ile doldur (en sık görülen değer)
Kayıp oranı orta ise	"Unknown"/"Missing" sınıfı ekle
Tahmin gücü çok önemliyse	KNN Imputation (komşulara göre doldurma)

📌 Unknown eklemek bazen model performansını artırır → Model “bu kişi eksik bilgiye sahip” diye öğrenir.

🔥 En gelişmiş doldurma yöntemleri (Model tabanlı imputasyon)

Yöntem	Özellik
KNN Imputer	Benzer örneklere bakarak doldurur → daha gerçekçi
Regression Imputation	Eksik değeri başka değişkenlerle tahmin eder
Multivariate Imputation (MICE)	En güçlü yöntem → Tüm değişken ilişkilerini kullanır

📌 Veri çok değerliyse model tabanlı imputasyon performansı belirgin artırabilir.

- 📌 KNN (K-Nearest Neighbors) Nedir?
- 🔷 Tanım:
- KNN, bir verinin sınıfını veya değerini kendisine en yakın K tane komşunun durumuna bakarak tahmin eden basit ama etkili bir makine öğrenmesi algoritmasıdır.
- 🔷 Nasıl çalışır?
- Tahmin edilmek istenen nokta alınır
- Verideki tüm noktalarla olan mesafesi (distance) hesaplanır
- Kendisine en yakın K veri seçilir
- Çoğunluk hangi sınıftaysa → o sınıfa atanır

- Diyelim elimizde bir kişi var ve onun felç geçirme riskini tahmin edeceğiz.
- Bu kişinin özelliklerini en çok benzeyen komşu kişilere bakıyoruz.
- Komşuların çoğu felç geçirmişse → bu kişinin riski de yüksek olabilir.
- Avantaj: Kolay anlaşılır ve non-parametrik.
- Dezavantaj: Büyük veri setlerinde yavaş olabilir.

- Eksik değer çözümü tek bir doğruya sahip değildir.
- Yöntem verinin miktarına, türüne ve dağılımına göre seçilir.
- Az veri kaybı varsa basit imputasyon yeterlidir, çok eksiklikte gelişmiş yöntemler veya değişken atma gerekebilir.

❓ Peki "unknown" ile nasıl başa çıkılır?

Bu durumda 3 farklı yol seçilebilir. Hangisini seçtiğin veri anlamına göre değişir:

Yöntem	Ne yapılır?	Ne zaman uygun?
🔹 1) Unknown'u Ayrı Kategori Olarak Bırakma	`unknown` aynen kalır, one-hot encoding'de ayrı sütun olur	kategori açıklayıcı duruyorsa ✔
🔹 2) Tahmini Doldurma (Model-based imputation)	eksik değerler sigara içme ihtimaline göre tahmin edilir	ileri seviye modellerde ✔
🔹 3) En Yakın Komşu / Mode ile İmpute	en sık görülen kategoriyle doldurulur	dağılım bozulabilir ⚠ genelde tercih edilmez

Outliers (Aykırı Değerler) Nedir?

Maddeler:

- IQR yöntemi
- Z-score yöntemi
- Görselle: boxplot

Outliers (Aykırı Değerler) Nedir?

Maddeler:

- IQR yöntemi
- Z-score yöntemi
- Görselle: boxplot

Aykırı değerler veri dağılımının dışında kalan uç değerlerdir. Her zaman silinmez; bazen doğrular ama model için zararlıysa işlem yapılır (silme, winsorize)

📌 Outlier (Aykırı Değer) Nedir ve Nasıl Bulunur?

Outlier = veri dağılımından çok uzak olan, normalden sapmış veri noktalarıdır.

Örneğin kişinin yaşı 400 görünüyorsa → bu outlier'dır.

🎯 Outlier Kontrollü Yapmak İçin Yöntemler:

YöntemMantık

IQR (Interquartile Range)

Q1 – Q3 aralığı hesaplanır, Q1–1.5IQR altı ve Q3+1.5IQR üstü outlier kabul edilir

Z-Score

Ortalama ± 3 standart sapma dışı outlier'dır

Boxplot

Görsel outlier tespiti kolay

IQR hesap örneği:

python

Kodu kopyala

```python
Q1 = df['bmi'].quantile(0.25)
Q3 = df['bmi'].quantile(0.75)
IQR = Q3 - Q1

alt_sinir = Q1 - 1.5 * IQR
ust_sinir = Q3 + 1.5 * IQR

outliers = df[(df['bmi'] < alt_sinir) | (df['bmi'] > ust_sinir)]
```

Outlier'lar modelin öğrenmesini bozar, dağılımı çarpıtır.

IQR, Z-score ve Boxplot ile tespit edilir. İşlenirken silme, winsorize (kırpma), transformation (log) gibi yöntemler uygulanabilir.

Outlier Handling: Yöntemler

Maddeler:

- Silme (filter)
- Winsorization (uçları sınırla)
- Log transform (dağılım düzeltme)
- Kayıt kontrolü (gerçek mi hatalı mı?)

Önce görselleştir, sonra nedenini araştır: gerçek olay mı yoksa ölçüm hatası mı? Hatalıysa sil; gerçekse dönüşüm ile idare et (log).

Winsorize (Kırpma) Nedir?

Tanım:

Winsorize, outlier (aykırı değer) problemini çözmek için kullanılan bir yöntemdir.

Ama burada silmek yerine uç değerleri sınırlar içine çekiyoruz.

Örnek ile düşünelim

Diyelim ki bir sütunumuz var: bmi

Değerler

18, 22, 25, 27, 30, 120

Normalde 120 çok büyük ve diğer veriyi gölgeleyebilir.

Winsorize ile:

- Sütunun üst sınırını belirleriz (örn. %95 percentile)
- 120 olan değeri bu üst sınır ile değiştiririz (örneğin 35)

Artık veri dağılımı bozulmaz ama aşırı uçlar model üzerinde baskı yapmaz.

Encoding (Neden?)

Maddeler:

- ML sayısal ister
- Label Encoding, One-Hot Encoding
- Ordinal vs Nominal

Basitçe; ‘red’, ‘blue’ yazısını makine anlamaz. One-hot = her kategori için sütun; label = kategoriye sayı verir. Sıralı (small<med<large) ise label uygun.

Label Encoding (LE)

Başlık: Label Encoding (LE)

Tanım:

Kategorik değişkenleri 0,1,2,... gibi sayılara çevirir.

Örnek:	Label Encoding Sonrası:
Gender	Gender
Male	1
Female	0
Female	0

Avantajlar:

- Basit ve hızlı
- Küçük kategorilerde etkili

Dezavantajlar:

- Linear modellerde sıralama hatası yapabilir (model 1>0 algılar)
- Tree tabanlı algoritmalar için sorun yok

One-Hot Encoding (OHE)

Tanım:

Her kategori için ayrı sütun açar, sıralama anlamı yoktur.

Örnek:

color

pembe

mavi

sarı

OHE Sonrası:

Male	Female
1	0
0	1
0	1

Scaling (Neden?)

Maddeler:

- MinMaxScaler (0–1)
- StandardScaler (mean=0, std=1)
- Hangi modele göre? (KNN/SVM uzaklık temelli → StandardScaler)

Farklı aralıklardaki değişkenler modelde yanlış ağırlık yaratabilir.
Özellikle mesafe temelli algoritmalarda şarttır.

Scaling (Neden?)

Maddeler:

- MinMaxScaler (0–1)
- StandardScaler (mean=0, std=1)
- Hangi modele göre? (KNN/SVM uzaklık temelli → StandardScaler)

Farklı aralıklardaki değişkenler modelde yanlış ağırlık yaratabilir.
Özellikle mesafe temelli algoritmalarda şarttır.

Sampling (Under & Over)

Maddeler:

- Dengesizlik problemini tanımla
- RandomUnderSampler (çoğunluğu küçültür)
- RandomOverSampler / SMOTE (azınlığı çoğaltır)

Eğer stroke gibi azınlık sınıf az ise model hep ‘hayır’ demeyi öğrenir. Over-sampling veriyi büyütür, under-sampling veri kaybına yol açar. SMOTE yapay örnek üretir

📌 SMOTE Nedir? (Over-Sampling)

🔹 **Problem: Stroke dataset gibi birçok medikal veri class imbalance içerir.**

Örn: %95 sağlıklı, %5 stroke → model hep "sağlıklı" diyerek yüksek doğruluk alır ama işe yaramaz.

🔹 **SMOTE (Synthetic Minority Oversampling Technique)**

Az görülen sınıfı çoğaltmak için yeni gerçekçi örnekler üretir.

Kopya yapmaz → sentetik veri üretir.

Nasıl çalışır?

1. **Minority class (stroke) örnekleri seçilir**
2. **Her biri için yakın komşuları bulunur**
3. **Aralarındaki noktalardan yeni yapay veri oluşturulur**

📌 Under Sampling Nedir?

🔷 Tam tersi durum: Majority class çok fazla → model dengesiz öğrenir.

Under-sampling:

Çoğunluk sınıfından rastgele veya kurallı olarak veri eksiltir.

YöntemMantık

Random Under Sampling

Fazla olan sınıftan rastgele veri siler

Tomek Links / Edited NN

Sınıfları ayıran ve belirsiz veri noktalarını temizler

Risk: Veri kaybı → bilgiyi azaltabilir.

Ama büyük datasetlerde işlem hızını artırır ve denge sağlar.

Over-sampling = Minority'i çoğalt

Under-sampling = Majority'yi azalt

İkisi de veri dengesini sağlar.

Tam Pipeline (Özet)

Maddeler:

1. İnceleme → 2. Missing → 3. Outlier → 4. Encoding → 5. Scaling → 6. Sampling → 7. Model

Bu pipelinenın mantığını unutmayın; her adımın gerekçesi var.

Uygulama Planı (Stroke Dataset)

Maddeler:

- **Dosya açma, info()**
- **Eksikleri doldurma (bmi örneği)**
- **Outlier kontrol (avg_glucose_level, bmi)**
- **Encoding (get_dummies)**
- **Scaling (MinMax)**
- **Sampling (SMOTE ve Under örnekleri)**
- **Basit Logistic Regression ile karşılaştırma (öncesi & sonrası)**

Konuşma:

"Şimdi adım adım uygulayacağız. Her adımı çalıştırıp sonucu tartışacağız."

Bu ders için kullandığımız veri seti

Kaggle → Stroke Prediction Dataset

ödev

- Aynı pipeline'ı başka veri üzerinde uygula (Titanic / House Prices)

Thanks For Listening