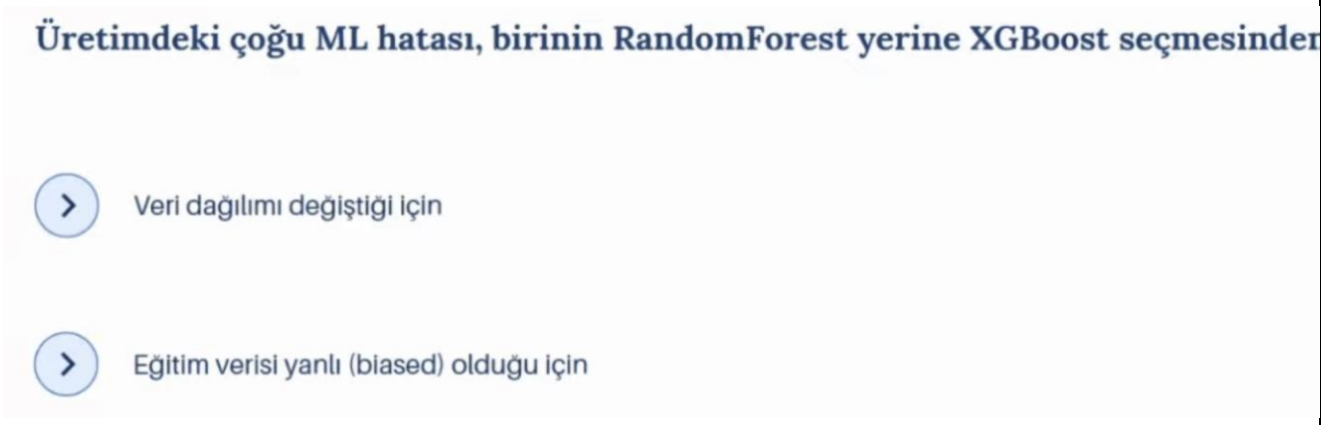
1 .



Görsel 1: Algoritma mı, Veri mi? Asıl Problem Nerede?

Genel Anlam: Neyi Anlatıyor?

Bu görsel, makine öğrenmesi projelerinde başarısızlığın kaynağının genellikle "yanlış algoritmayı seçmek" gibi teknik detaylarda değil, çok daha temel olan **veri problemlerinde** yattığını vurgular.

- **RandomForest** ve **XGBoost:** Bunlar, makine öğrenmesinde kullanılan iki farklı, çok güçlü ve popüler algoritmadır. Tıpkı Python'da bir işi yapmak için NumPy veya Pandas'ın farklı ama güçlü yeteneklerini kullanmak gibi. Çoğu zaman birbirlerine yakın sonuçlar verirler.
- **Ana Mesaj:** Projeniz canlıda (üretimde) başarısız oluyorsa, suçu algoritma seçiminde aramadan önce dönüp verinize bakmalısınız. Asıl sorun, genellikle aşağıdaki iki maddeden biridir.

Madde 1: "Veri dağılımı değiştiği için" (Data Drift)

Bu, modelin eğitildiği "evren" ile şu an yaşadığı "evrenin" birbirinden farklılaşması demektir.

Adım Adım Python ile Anlatım

1. **Eğitim (2020):** Elinizde 2015-2020 arasına ait ev fiyatı verilerini içeren bir pandas DataFrame'i var. Modelinizi bu veriyle eğitiyorsunuz.

```
# train_data_2020.csv
# metrekare, oda_sayisi, bina_yasi, fiyat
# 100, 3, 5, 500000
model.fit(X_train, y_train)
```

Modeliniz, o dönemin ekonomik koşullarına göre "100 metrekare, 3 odalı bir evin fiyatı yaklaşık 500.000 TL'dir" kuralını öğrenir.

2. **Üretim (2024):** Uygulamanız artık canlıda ve 2024 yılına ait yeni bir ev için fiyat tahmini yapması gerekiyor.
    - **Ne Değişti?** 2020'den 2024'e kadar pandemi yaşandı, enflasyon arttı, faiz oranları değişti. Yani evin özellikleriyle fiyatı arasındaki ilişki (veri dağılımı) tamamen değişti.
    - **Sonuç:** Modeliniz, 2020'nin kurallarıyla 2024'te tahmin yapmaya çalışır ve 100 metrekare, 3 odalı bir eve hala "500.000 TL" gibi komik bir fiyat tahmini yapar. Sorun modelin algoritmasında değil, yaşadığı **evrenin değişmesindedir.**

1. Bisküvi Üretimi Analojisi

- **Eğitim:** Kalite kontrol makinenizi (model), sadece **klasik Petibör bisküvilerin** (dikdörtgen, sarı renkli) görüntüleriyle eğittiniz. Makine, "iyi bisküvinin" ne olduğunu bu dağılıma göre öğrendi.
- **Değişim:** Yönetim karar değiştirdi ve fabrika artık **kremalı, kakaolu, yuvarlak bisküviler** üretmeye başladı.
- **Hata:** Makine, eski bilgileriyle yeni, mükemmel kakaolu bisküvileri görür ve hepsini "hatalı" olarak damgalar (yanlış şekil, yanlış renk). Modelin eğitildiği **veri dağılımı (Petibör)** ile canlıda karşılaştığı **veri dağılımı (Kakaolu)** tamamen farklıdır.

2. Restoran Analojisi

- **Eğitim:** Dünya çapında ünlü bir **suşi ustasını (model)**, en kaliteli balıklar ve Japon pirinçleri (veri dağılımı) ile yıllarca eğittiniz.
- **Değişim:** Restoranın sahibi bir gecede karar değiştirip mekanı bir **Adana Kebapçısına** çevirdi. Mutfağa artık balık yerine kuzu kıyması ve acı biber geliyor.
- **Hata:** Suşi ustasının tüm bilgisi ve yeteneği, yeni gelen malzemeler karşısında işe yaramaz hale gelir. Modelin eğitildiği "suşi evreni" yok olmuş, yerine hiç bilmediği bir "kebap evreni" gelmiştir.

Madde 2: "Eğitim verisi yanlı (biased) olduğu için"

Bu, modeli eğitmek için kullandığınız verinin, gerçek dünyayı adil ve doğru bir şekilde temsil etmemesi demektir.

Adım Adım Python ile Anlatım

1. **Eğitim:** Bir şirketin geçmişteki başarılı işe alımlarını içeren bir DataFrame ile bir model eğitiyorsunuz. Amacınız, yeni başvuruları analiz edip en uygun adayları bulmak.

```
 # basarili_calisanlar.csv
# universite, bolum, mezuniyet_yili
# Bogazici, Bilgisayar, 2015
# ODTU, Elektrik, 2016
```

Ancak farkında olmadan, elinizdeki verideki başarılı çalışanların %90'ı sadece iki üniversiteden mezun.

2. **Öğrenilen Kural:** Modeliniz, model.fit() sırasında "Başarılı olmak için Boğaziçi veya ODTÜ'den mezun olmak gerekir" gibi **yanlış bir önyargı** öğrenir.
3. **Üretimdeki Hata:** Uygulamanız canlıya çıktığında, diğer üniversitelerden gelen çok yetenekli adayları, sırf bu yanlı (biased) kural yüzünden otomatik olarak eler. Sorun modelin algoritmasında değil, ona öğrettiğiniz verinin **gerçek dünyayı temsil etmemesindedir.**

1. Bisküvi Üretimi Analojisi

- **Eğitim:** Bozuk bisküvileri tespit eden bir model eğitiyorsunuz. Ama eğitim için kullandığınız tüm fotoğraflar, **sadece fabrikanın gündüz vardiyasında, parlak ışık altında** çekilmiş.
- **Değişim:** Model canlıya alındığında gece vardiyasında da çalışmaya başlar.
- **Hata:** Gece vardiyasında ışıklar daha loştur ve gölgeler oluşur. Model, daha önce hiç gölgedeki bir bisküvi görmediği için, gölgeden dolayı biraz daha koyu görünen sağlam bisküvileri "yanık" veya "bozuk" olarak sınıflandırır. Eğitim verisi, **farklı ışık koşullarını temsil etmediği için yanlıdır.**

2. Restoran Analojisi

- **Eğitim:** Yeni bir acılı yemeğin beğenilip beğenilmeyeceğini tahmin eden bir model geliştiriyorsunuz. Test grubunuz, **sadece öğle yemeğine gelen ve tek başına yemek yiyen müşterilerden** oluşuyor. Bu grup yemeğe bayılıyor.
- **Değişim:** Yemeği menüye koyuyorsunuz.
- **Hata:** Akşam yemeğine gelen çocuklu aileler bu yemeği sipariş ettiğinde, çocuklar acı olduğu için yiyemiyor ve bir sürü şikayet alıyorsunuz. Modeliniz başarılı olamadı çünkü eğitim veriniz, tüm müşteri profilinizi (aileler, turistler vb.) değil, **sadece küçük ve yanlı bir grubu** temsil ediyordu.

2

1. Regression (Regresyon): Sayısal Tahmin Sanatı

Tek Cümlede: Sayısal bir sonucu (fiyat, miktar, sıcaklık vb.) tahmin etme problemidir. Cevap her zaman bir sayıdır, bir kategori değildir.

Hangi Soruyu Cevaplar?

- *Ne kadar?*
- *Kaç tane?*
- *Hangi değerde?*

Python Dünyası ile Anlatım

Bir pandas DataFrame'iniz olduğunu düşünün. Amacınız, bir evin özelliklerine bakarak fiyatını tahmin etmektir.

```
import pandas as pd
from sklearn.linear_model import LinearRegression

# 1. Verimiz (DataFrame)
# Elimizdeki geçmiş ev satışları verisi
data = {'metrekare': [100, 150, 80, 120],
        'oda_sayisi': [3, 4, 2, 3],
        'fiyat': [250000, 400000, 180000, 310000]}
df = pd.DataFrame(data)

# 2. Model Eğitimi
# Model, metrekare ve oda sayısının fiyata olan etkisini öğrenir.
X = df[['metrekare', 'oda_sayisi']] # Özellikler (Girdiler)
y = df['fiyat']                     # Hedef (Sayısal Çıktı)

model = LinearRegression()
model.fit(X, y)

# 3. Tahmin
# Artık yeni bir evin fiyatını TAHMİN edebiliriz.
yeni_ev = [[110, 3]] # 110 metrekare, 3 odalı yeni bir ev
tahmini_fiyat = model.predict(yeni_ev)

print(f"Bu evin tahmini fiyatı: {tahmini_fiyat[0]:.2f} TL")
# Çıktı: Bu evin tahmini fiyatı: 282500.00 TL (örneğin)
```

Model, veri içindeki deseni öğrenerek gelecekteki bir **sayıyı** tahmin eder.

Analojilerle Anlatım

1. Analoji

- **Problem:** "Geçmiş satış verilerine (VBAP, VBRK) bakarak, önümüzdeki ay X ürününden (MATNR) **ne kadar ciro (NETWR)** bekliyoruz?"
- **Cevap:** Cevap "iyi/kötü" gibi bir etiket değil, **500.000 TL** gibi sayısal bir değerdir.

2. Bisküvi Üretimi Analojisi

- **Problem:** "Hava sıcaklığı, fırın nem oranı ve un kalitesine bakarak, bir sonraki partide üretilecek bisküvilerden **kaç tanesinin** standart dışı (kırık, yanık) çıkacağını tahmin etmeliyiz."

- **Cevap:** Cevap "evet/hayır" değil, **157 adet** gibi bir sayıdır. Bu sayıya göre üretim planı ayarlanabilir.

3. Restoran Analojisi

- **Problem:** "Haftanın günü, hava durumu ve o gün şehirde bir etkinlik (maç, konser) olup olmadığına bakarak, bu akşam restorana **kaç müşteri** geleceğini tahmin etmeliyiz."
- **Cevap:** Cevap "kalabalık/sakin" değil, **250 kişi** gibi bir sayıdır. Bu sayıya göre personel ve malzeme planlaması yapılır.

Özet

Özellik	Açıklama
Soru Tipi	Ne kadar? Kaç? Hangi fiyatta?
Çıktı (Hedef)	Sürekli bir sayısal değer (örn: 250.75, 1500, -10)
Python Örneği	Ev fiyatı, stok miktarı, müşteri sayısı tahmini

3

2. Classification (Sınıflandırma): Etiketleme Sanatı

Tek Cümlede: Bir veriyi, önceden tanımlanmış belirli kategorilerden veya sınıflardan birine atama problemidir. Cevap her zaman bir **etikettir**, bir sayı değildir.

Hangi Soruyu Cevaplar?

- *Hangisi?* (örn: Kedi mi, köpek mi?)
- *Ne tür?* (örn: Spam mı, değil mi?)
- *Hangi kategoride?* (örn: Riskli mi, güvenilir mi?)

Python Dünyası ile Anlatım

Bir pandas DataFrame'iniz olduğunu düşünün. Amacınız, bir e-postanın özelliklerine bakarak "Spam" olup olmadığını sınıflandırmaktır.

```
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier

# 1. Verimiz (DataFrame)
# Elimizdeki geçmiş e-posta verisi (1: Spam, 0: Spam Değil)
data = {'buyuk_harf_orani': [0.5, 0.1, 0.2, 0.7],
        'link_sayisi': [3, 1, 0, 5],
        'kategori': [1, 0, 0, 1]} # Hedef (Kategorik Çıktı)
df = pd.DataFrame(data)
```

```
# 2. Model Eğitimi
# Model, büyük harf oranı ve link sayısının spam'e olan etkisini öğrenir.
X = df[['buyuk_harf_orani', 'link_sayisi']] # Özellikler (Girdiler)
y = df['kategori']                           # Hedef (Etiketler)

model = KNeighborsClassifier(n_neighbors=3)
model.fit(X, y)

# 3. Tahmin
# Artık yeni bir e-postayı SINIFLANDIRABİLİRİZ.
yeni_eposta = [[0.6, 4]] # Büyük harf oranı yüksek, link sayısı fazla
tahmin = model.predict(yeni_eposta)

if tahmin[0] == 1:
    print("Bu e-posta bir SPAM!")
else:
    print("Bu e-posta spam değil.")
# Çıktı: Bu e-posta bir SPAM!
```

Model, veri içindeki deseni öğrenerek yeni bir veriye **etiket** atar.

Analojilerle Anlatım

1. Analoji

- **Problem:** "Bir müşterinin (KUNNR) geçmiş alışverişlerine, borç durumuna ve sisteme kayıt tarihine bakarak, bu müşteri **'Riskli' mi yoksa 'Güvenilir' mi**?"
- **Cevap:** Cevap sayısal bir değer değil, **'Riskli'** gibi önceden tanımlanmış bir kategoridir.

2. Bisküvi Üretimi Analojisi

- **Problem:** "Üretim bandından geçen bir bisküvinin kamera görüntüsüne bakarak, bu bisküvi **'Sağlam' mı, 'Kırık' mı, yoksa 'Yanık' mı** olduğunu sınıflandır."
- **Cevap:** Cevap, bisküviye atanan **'Kırık'** gibi bir etikettir. Buna göre bisküvi hattan ayrılabilir.

3. Restoran Analojisi

- **Problem:** "Bir müşterinin geçmiş siparişlerine ve ziyaret sıklığına bakarak, bu müşterinin **'Sadık Müşteri' mi, 'Riskli Müşteri' mi (bir daha gelmeyebilir), yoksa 'Yeni Müşteri' mi** olduğunu belirle."
- **Cevap:** Cevap, müşteriye atanan **'Riskli Müşteri'** gibi bir etikettir. Bu müşteriye özel bir indirim gönderilerek geri kazanılmaya çalışılabilir.

Regresyon vs. Sınıflandırma: Özet Tablo

Bu iki temel görev arasındaki farkı netleştirmek için:

Özellik	**Regression (Regresyon)**	**Classification (Sınıflandırma)**
Soru	Ne kadar? Kaç?	Hangisi? Hangi grup?
Çıktı	**Sayı** (Sürekli Değer)	**Kategori** (Etiket)
Örnek	Ev Fiyatı Tahmini	E-posta Spam Tespiti
Python Hedefi	y sütunu float veya int olur (örn: 250000.0)	y sütunu int (0,1,2) veya string ('Kedi', 'Köpek') olur

4

Bu görsel, makine öğrenmesinin en temel felsefesini içeriyor

Görsel 3: Dağılımların Önemi ve Modelin Evreni

Cümle 1: "Model, bir veri dağılımının üzerinde yaşayan bir fonksiyondur."

Anlamı: Bir makine öğrenmesi modeli, sihirli bir kutu değildir. Sadece ve sadece kendisine öğretilen verinin (eğitim setinin) istatistiksel özelliklerini, desenlerini ve kurallarını öğrenmiş bir programdır. Onun bildiği tek "gerçeklik", eğitildiği verinin dünyasıdır.

Spotify'ın sizin için bir şarkı öneri modeli (DJ) eğittiğini düşünelim.

```
# Sizin dinleme geçmişiniz (eğitim veriniz)
# 1: Rock, 2: Pop, 3: Caz
dinleme_gecmisi = [1, 1, 3, 1, 1, 1, 3, 1]
# Gördüğünüz gibi %75 Rock, %25 Caz dinliyorsunuz. Bu sizin "müzik dağılımınız".

model.fit(dinleme_gecmisi)
```

model.fit() çalıştığında, model şunu öğrenir: "Bu kullanıcı ağırlıklı olarak Rock dinliyor, ara sıra da Caz. Pop müzik diye bir şey bu kullanıcının evreninde yok." Modeliniz artık bu **"Rock ve Caz Dağılımı" üzerinde yaşar.**

Cümle 2: "Dağılımı değiştirirseniz, modelin yaşadığı evreni değiştirirsiniz."

Anlamı: Eğer modelin eğitildiği veri ile canlıda karşılaştığı veri birbirinden farklıysa, modelin öğrendiği kurallar geçersiz kalır. Çünkü model, hiç tanımadığı, kurallarını bilmediği bambaşka bir evrene bırakılmıştır.

Hayatınızda bir değişiklik oldu ve elektronik dans müziği (EDM) dinlemeye başladınız.

```
# Yeni dinleme alışkanlığınız (canlıdaki veri)
# 4: EDM
yeni_sarkilar = [4, 4, 4, 1, 4, 4]
# Artık ağırlıklı olarak EDM dinliyorsunuz. Dağılım değişti!
```

Eski modelinizden size şarkı önermesini istediğinizde (model.predict()), o hala sizin Rock ve Caz sevdiğiniz evrende yaşadığı için size asla EDM önermeyecektir. Hatta EDM'in ne olduğunu bile bilmez. **Dağılım değiştiği için**, modelin yaşadığı evren de değişmiş ve model işe yaramaz hale gelmiştir.

Analojilerle Anlatım

1. Bisküvi Üretimi Analojisi

- **Modelin Evreni:** Kalite kontrol makineniz, sadece **"Petibör Bisküvi Evreni"**'nde yaşamak üzere eğitilmiştir (dikdörtgen, sarı, 5 gram).
- **Evrenin Değişmesi:** Fabrika artık **"Kakaolu Bisküvi Evreni"**'ne geçiş yapmıştır (yuvarlak, kahverengi, 8 gram).
- **Sonuç:** Eski makine, yeni evrenin kurallarını bilmediği için her mükemmel kakaolu bisküviyi "hatalı" olarak damgalar. Modelin yaşadığı evreni değiştirdiğiniz için model artık çalışmaz.

2. Restoran Analojisi

- **Modelin Evreni:** Baş aşçınız, sadece **"Suşi Malzemeleri Evreni"**'nde (balık, pirinç, soya sosu) yaşamak üzere eğitilmiş bir ustadır.
- **Evrenin Değişmesi:** Restoran bir gecede **"Adana Kebapçısı Evreni"**'ne dönüşmüştür (kuzu kıyması, acı biber).
- **Sonuç:** Suşi ustası, yeni evrenin kurallarını ve malzemelerini bilmediği için Adana Kebap yapamaz. Modelin yaşadığı evreni değiştirdiğiniz için model artık işlevsizdir.

Özet

- **Anahtar Kelime:** Veri Dağılımı, Model Evreni, Dağılım Değişikliği (Data Drift)
- **Sonuç:** Bir ML modeli, sadece eğitildiği verinin dünyasını bilir. Eğer gerçek dünya değişirse (yani veri dağılımı değişirse), modelinizi bu yeni dünyayı öğrenmesi için yeniden eğitmeniz gerekir. Bu yüzden modelleri canlıya aldıktan sonra performanslarını sürekli izlemek hayati önem taşır.

5

Görsel 4: EDA (Keşifsel Veri Analizi) - Dedektiflik Sanatı

Tek Cümlede: Bir model kurmadan önce, veri setini bir dedektif gibi inceleyerek sırlarını çözme, sağlığını kontrol etme ve hikayesini anlama sürecidir.

Hangi Soruları Cevaplar?

- Bu veri seti neye benziyor?
- İçinde ne gibi anormallikler (eksiklikler, aykırı değerler) var?
- Hangi özellikler birbiriyle ilişkili?

- Verinin altında yatan ana hikaye nedir?

Python Dünyası ile Anlatım

Elinizde Titanic yolcularına ait bir veri seti (titanic.csv) olduğunu düşünün. Amacınız hayatta kalma durumunu tahmin etmek. Ama önce **EDA** yapmalısınız:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

df = pd.read_csv('titanic.csv')

# 1. Genel Bakış Atma (.head(), .info(), .shape)
print("Verinin ilk 5 satırı:")
print(df.head())
print("\nVeri tipleri ve eksik değerler:")
df.info()

# 2. Temel İstatistikleri İnceleme (.describe())
print("\nSayısal verilerin özeti:")
print(df.describe())

# 3. Eksik Verileri Bulma (.isnull().sum())
print("\nEksik veri sayıları:")
print(df.isnull().sum())
# 'Age' (Yaş) sütununda çok fazla eksik veri olduğunu fark edersiniz.

# 4. Görselleştirme (sns.countplot, sns.histplot)
# Hayatta kalma durumunun dağılımı
sns.countplot(x='Survived', data=df)
plt.show()

# Yaş dağılımı
sns.histplot(df['Age'].dropna(), kde=True)
plt.show()

# 5. İlişkileri Sorgulama (.corr(), sns.heatmap)
# Sayısal sütunlar arasındaki korelasyon matrisi
correlation_matrix = df.corr(numeric_only=True)
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.show()
# 'Pclass' (Bilet Sınıfı) ile 'Survived' (Hayatta Kalma) arasında negatif bir ilişki olduğunu görürsünüz.
```

Bu adımlar, modeli kurmadan önce veriyi "temizlemeniz" ve "anlamanız" için size yol gösterir.

Analojilerle Anlatım

1. Dedektif Analojisi

- **Problem:** Bir cinayeti çözmek. Veri setiniz **olay yeri**, siz ise **baş dedektifsiniz.**
- **EDA Adımları:**
    - **Genel Bakış:** Odaya girip genel durumu anlarsınız (.head()).
    - **Ölçüm:** Kurbanın boyunu ölçer, kovanları sayarsınız (.describe()).
    - **Anormallik:** Kayıp cinayet silahını (eksik veri) ve kurbanın cebindeki penguen tüyünü (aykırı değer) fark edersiniz.
    - **Görselleştirme:** Olay yerinin fotoğraflarını çekersiniz (grafikler).
    - **İlişki:** Devrilen vazo ile kan izleri arasında bağlantı kurmaya çalışırsınız (korelasyon).
- **Amaç:** Mahkemeye (model kurmaya) gitmeden önce tüm delilleri anlamak ve hazırlamak.

2. Bisküvi Üretimi Analojisi

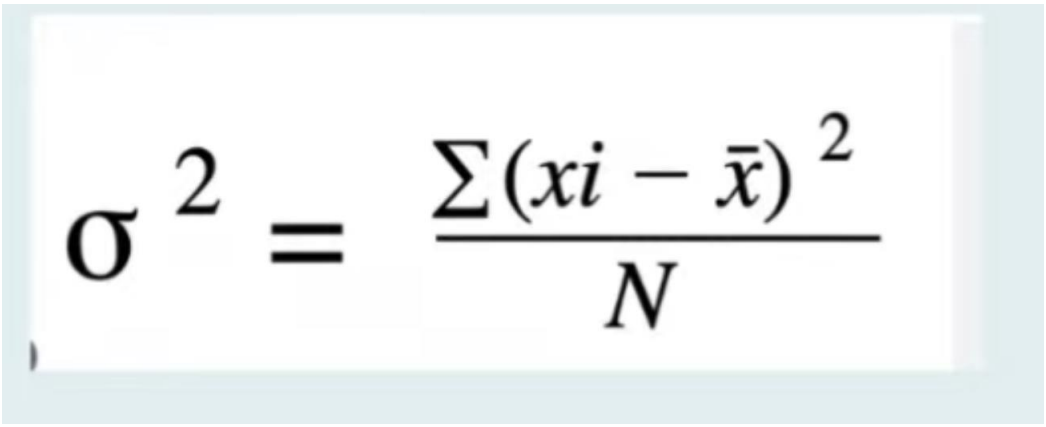
- **Problem:** Yeni üretilen bisküvilerin kalite standartlarına uyup uymadığını denetlemek. Veri setiniz, son partiden alınan **1000 bisküvilik bir numune.**
- **EDA Adımları:**
    - **Genel Bakış:** Kutuyu açıp bisküvilerin genel şekline, rengine bakarsınız.
    - **Ölçüm:** Rastgele 10 bisküvi alıp ağırlıklarını ve çaplarını ölçer, ortalamasını alırsınız.
    - **Anormallik:** Arada hiç bisküviye benzemeyen, tamamen yanmış bir parça (aykırı değer) ve içinde hiç çikolata olmayan boş bisküviler (eksik veri) fark edersiniz.
    - **Görselleştirme:** Bisküvi ağırlıklarının bir histogramını çizer, çoğunluğun hedeflenen ağırlığa yakın olup olmadığını görürsünüz.
    - **İlişki:** Bisküvinin rengi koyulaştıkça ağırlığının azalıp azalmadığını (yanma-kuruma ilişkisi) incelersiniz.
- **Amaç:** Üretim raporunu yazmadan (model kurmadan) önce, numunedeki tüm problemleri ve genel kalite profilini ortaya çıkarmak.

3. Restoran Analojisi

- **Problem:** Restoranın son bir aylık performansını analiz etmek. Veri setiniz, kasadan alınan **tüm sipariş fişleri.**
- **EDA Adımları:**
    - **Genel Bakış:** Fişleri üst üste koyup genel bir göz atarsınız.
    - **Ölçüm:** Ortalama masa başına harcama, en çok satan ürün, en yoğun saatler gibi temel istatistikleri çıkarırsınız.
    - **Anormallik:** 10.000 TL'lik bir fiş (aykırı değer) ve üzerinde ne sipariş edildiği yazmayan boş bir fiş (eksik veri) dikkatinizi çeker.
    - **Görselleştirme:** Saatlere göre müşteri yoğunluğunu gösteren bir bar grafiği çizersiniz.
    - **İlişki:** Ana yemek sipariş edenlerin tatlı sipariş etme olasılığını incelersiniz.

- **Amaç:** Aylık strateji toplantısına gitmeden (model kurmadan) önce, işleyişteki tüm dinamikleri, sorunları ve fırsatları anlamak.

6



İstatistiğin temel taşlarından biri olan formül.

Görsel 5: Varyans Formülü - "Dağınıklığın" Matematiksel Tarifi

Formülün Adı: Varyans (Variance)

Tek Cümlede: Bir veri setindeki değerlerin, kendi ortalamasından ne kadar uzağa yayıldığını, yani verinin ne kadar "dağınık" veya "istikrarlı" olduğunu ölçen bir metriktir.

Hangi Soruyu Cevaplar?

- Veri noktaları birbirine ne kadar yakın veya uzak?
- Bu veri seti öngörülebilir ve istikrarlı mı, yoksa dengesiz ve değişken mi?

Formülün Parçaları ve Python ile Anlatım

Formülü bir program gibi adım adım düşünelim:

Sembol	Anlamı	Python (NumPy) Karşılığı
σ²	**Varyans:** Hesaplamak istediğimiz nihai sonuç.	.var()
Σ	**Toplama İşareti:** Bir döngü (for) içindeki sonuçları toplamak.	.sum()
xi	Veri setindeki **her bir eleman.**	Döngüdeki her bir x değeri
x̄	Veri setinin **aritmetik ortalaması.**	.mean()
N	Veri setindeki **toplam eleman sayısı.**	len() veya .size
(...)²	**Karesini Alma:** Farkları pozitif hale getirmek ve büyük sapmalara daha fazla ağırlık vermek.	**2

```
import numpy as np

# Günlük süt satışları (istikrarlı)
sut_satislari = np.array([100, 102, 98, 101, 99])

# Günlük şemsiye satışları (dengesiz)
semsiye_satislari = np.array([10, 250, 5, 80, 155])

# Python ile Varyans Hesabı
varyans_sut = sut_satislari.var()
varyans_semsiye = semsiye_satislari.var()

print(f"Süt Satışlarının Ortalaması: {sut_satislari.mean():.2f}")
print(f"Şemsiye Satışlarının Ortalaması: {semsiye_satislari.mean():.2f}")
print("-" * 30)
print(f"Süt Satışlarının Varyansı: {varyans_sut:.2f}")
print(f"Şemsiye Satışlarının Varyansı: {varyans_semsiye:.2f}")

# Çıktı:
# Süt Satışlarının Ortalaması: 100.00
# Şemsiye Satışlarının Ortalaması: 100.00
# ------------------------------
# Süt Satışlarının Varyansı: 2.00
# Şemsiye Satışlarının Varyansı: 8530.00
```

Gördüğünüz gibi, ortalamaları aynı olmasına rağmen, şemsiye satışlarının varyansı **çok daha yüksektir.** Bu, o verinin çok daha "dağınık" ve "öngörülemez" olduğunu tek bir sayıyla ifade eder.

Analojilerle Anlatım

1. Süt ve Şemsiye Analojisi

- **Süt Satışları (Düşük Varyans):** Her gün hemen hemen aynı miktarda satılır. Değerler ortalamaya çok yakındır. Bu, istikrarlı ve planlaması kolay bir veridir.
- **Şemsiye Satışları (Yüksek Varyans):** Satışlar hava durumuna göre çılgınca değişir. Bir gün 5, diğer gün 250 adet satılabilir. Değerler ortalamadan çok uzağa dağılmıştır. Bu, dengesiz ve stok planlaması zor bir veridir.

2. Bisküvi Üretimi Analojisi

- **Problem:** İki farklı fırınımız var ve ikisi de "10 gram" ağırlığında bisküvi üretmeye ayarlı. Hangisi daha iyi çalışıyor?
- **Fırın A (Düşük Varyans):** Ürettiği bisküvilerin ağırlıkları: [9.9, 10.1, 10.0, 9.8, 10.2]. Ortalaması 10'a çok yakın ve **varyansı çok düşük.** Bu fırın çok **istikrarlı ve güvenilirdir.**

- **Fırın B (Yüksek Varyans):** Ürettiği bisküvilerin ağırlıkları: [8, 12, 7, 13, 10]. Ortalaması yine 10 olabilir, ama **varyansı çok yüksektir.** Bu fırın **dengesiz çalışıyor** ve acilen bakıma ihtiyacı var.

3. Restoran Analojisi

- **Problem:** İki farklı aşçımız var. İkisinin de yaptığı yemeklerin müşteri memnuniyet puanı ortalaması 10 üzerinden 8. Hangisini terfi ettirmeliyiz?
- **Aşçı A (Düşük Varyans):** Aldığı puanlar: [8, 7, 9, 8, 8]. Her zaman istikrarlı bir şekilde iyi yemek yapar. **Varyansı düşüktür.** Bu, güvenilir bir aşçıdır.
- **Aşçı B (Yüksek Varyans):** Aldığı puanlar: [10, 5, 10, 5, 10]. Bazen harikalar yaratıyor, bazen yemeği yakıyor. **Varyansı yüksektir.** Bu, riskli ve dengesiz bir aşçıdır.

Özet

- **Anahtar Kelime:** Varyans, Standart Sapma (varyansın karekökü), Dağınıklık, İstikrar.
- **Neden Önemli?:** Varyans, bir veri setinin "karakterini" anlamamızı sağlar. Düşük varyanslı özellikler bazen model için anlamsız olabilir (hepsi aynı değerdeyse), yüksek varyanslı özellikler ise hem önemli bilgiler taşıyabilir hem de modelin öğrenmesini zorlaştırabilir.

Sözlük Karşılığı

- **Bias -> Yanlılık, Önyargı**
- **Variance -> Varyans, Değişkenlik, Tutarsızlık**

Ancak bu kelimeler ML'deki anlamlarını tam olarak karşılamaz . Daha sezgisel isimler kullanalım:

- **Bias -> Sabit Fikirlilik / Eksik Öğrenme**
- **Varyans -> Ezbercilik / Aşırı Öğrenme**

En İyi Analoji: Sınava Hazırlanan İki Farklı Öğrenci

Bir öğretmensiniz ve öğrencilerinizi çok önemli bir final sınavına hazırlıyorsunuz. Elinizde de bir hazırlık kitabı (bu sizin **eğitim veriniz**) var. Final sınavı ise kitaptaki soruların benzeri ama biraz değiştirilmiş versiyonlarını içerecek (bu sizin **test veriniz**).

1. Yüksek Bias'lı Öğrenci (Tembel ve Sabit Fikirli Öğrenci)

Bu öğrenci ders çalışmayı sevmiyor ve konuları çok basite indirgiyor. Hazırlık kitabına şöyle bir göz atıyor ve kendine çok basit, genel geçer kurallar çıkarıyor.

- **Öğrenme Şekli:** "Tarih sorularının cevabı hep Fatih Sultan Mehmet'tir.", "Matematikte bir soruda 3 ve 4 varsa cevap kesin 5'tir." gibi aşırı basitleştirilmiş "önyargılar" geliştiriyor.
- **Hazırlık Kitabındaki Performansı (Eğitim Hatası):** Bu basit kurallarla hazırlık kitabındaki soruların çoğunu **yanlış yapar**. Çünkü konular o kadar basit değil. **Performansı KÖTÜDÜR.**
- **Gerçek Sınavdaki Performansı (Test Hatası):** Gerçek sınavda da doğal olarak **başarısız olur.**

Yüksek Bias'lı Model: Tıpkı bu öğrenci gibidir. Model o kadar basittir ki, verinin içindeki karmaşık ilişkileri ve desenleri **öğrenemez**. Hem eğitim verisinde hem de yeni veride yüksek hata yapar. Buna **Eksik Öğrenme (Underfitting)** denir.

2. Yüksek Varyans'lı Öğrenci (Ezberci Öğrenci)

Bu öğrenci çok çalışkan ama konuların mantığını anlamak yerine her şeyi harfi harfine ezberlemeyi seçiyor.

- **Öğrenme Şekli:** Hazırlık kitabındaki 500 sorunun tamamını, cevaplarıyla birlikte **kelimesi kelimesine ezberler**. Neden o cevabın doğru olduğunu anlamaz, sadece ezberler.
- **Hazırlık Kitabındaki Performansı (Eğitim Hatası):** Hazırlık kitabından bir soru sorduğunuzda anında doğru cevabı söyler. **Performansı MÜKEMMELDİR (%100 Başarı).**
- **Gerçek Sınavdaki Performansı (Test Hatası):** Gerçek sınava girdiğinde, sorulardaki küçük bir değişiklik (örneğin "Ahmet'in 5 elması var" yerine "Ayşe'nin 5 portakalı var" yazması) onun bütün ezberini bozar. Çünkü o sadece kitaptaki kalıbı ezberlemiştir, mantığını değil. Sonuç olarak **başarısız olur.**

Yüksek Varyans'lı Model: Tıpkı bu öğrenci gibidir. Model o kadar karmaşıktır ki, eğitim verisindeki her bir detayı, hatta tesadüfi gürültüyü bile **ezberler**. Bu yüzden eğitim verisinde harika sonuçlar verir. Ama daha önce görmediği yeni bir veriyle karşılaştığında, ezberine uymadığı için çok kötü tahminler yapar. Buna **Aşırı Öğrenme / Ezberleme (Overfitting)** denir.

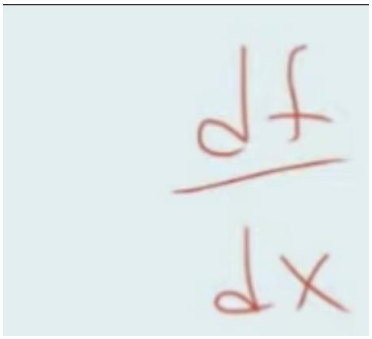
Özet Tablo

Özellik	**Yüksek Bias (Önyargı)**	**Yüksek Varyans (Ezbercilik)**
Model Karmaşıklığı	Çok Basit	Çok Karmaşık
Asıl Sorun	**Eksik Öğrenme (Underfitting)**	**Aşırı Öğrenme (Overfitting)**
Eğitim Verisi Performansı	Kötü	Mükemmel
Yeni (Test) Veri Performansı	Kötü	Kötü
Lakabı	Tembel / Sabit Fikirli	Ezberci / Aşırı Hassas

Sizin Amacınız Nedir?

Sizin amacınız ne tembel ne de ezberci olan, konuların mantığını anlayan **"İdeal Öğrenciyi"** yetiştirmektir. Yani ne çok basit ne de çok karmaşık, **dengeli bir model** kurmaktır. Bu dengeye **Bias-Varyans Dengesi (Bias-Variance Tradeoff)** denir ve Makine Öğrenmesindeki en temel hedeftir.

7



Bu sembol, makine öğrenmesinin "motorunu" çalıştıran, en temel ama en güçlü kavramlardan biridir.

Görsel 6: Türev (Derivative) - Değişimin Pusulası

Formülün Adı: Türev (Derivative)

Tek Cümlede: Bir değişkendeki (x) ufacık bir artışın, bir fonksiyondaki (f) sonucu ne kadar ve hangi yönde değiştirdiğini ölçen "anlık değişim oranı" veya "eğimdir".

Hangi Soruyu Cevaplar?

- Eğer x'i birazcık artırırsam, f artar mı azalır mı?
- Bu değişim ne kadar dik/hızlı olur?

Python Dünyası ve Makine Öğrenmesi ile Anlatım

Makine öğrenmesinde türevin en önemli görevi, bir modelin "hatasını" nasıl azaltacağını bulmaktır. Bu sürece **Gradyan İnişi (Gradient Descent)** denir.

- f = Modelin **Hata Fonksiyonu** (Loss Function). Amacımız bu değeri minimize etmek.
- x = Modelin ayarlanabilir bir **parametresi** (örneğin bir weight veya bias).

Bu durumda df/dx şu anlama gelir: **"Modelin bir parametresini (x) ufacık değiştirirsem, modelin toplam hatası (f) ne kadar ve hangi yönde (artarak mı, azalarak mı) değişir?"**

```
# Bu sadece kavramsal bir koddur, gerçek kütüphaneler bunu otomatik yapar.
def gradient_descent(learning_rate=0.1, adim_sayisi=100):
    # 1. Başlangıç: Rastgele bir 'x' parametresi seç.
    x = 10.0

    for i in range(adim_sayisi):
        # 2. Türevi (Gradyanı/Eğimi) Hesapla
        # Hata fonksiyonumuzun f(x) = x^2 olduğunu varsayalım.
        # Bu fonksiyonun türevi df/dx = 2*x 'tir.
        gradyan = 2 * x

        # 3. Parametreyi Güncelle
        # Eğimin TERSİ yönünde küçük bir adım at.
        x = x - learning_rate * gradyan
```

```
        # Mevcut durumu yazdır
        hata = x**2
        print(f"Adım {i+1}: Parametre (x) = {x:.2f}, Hata (f) = {hata:.2f}")

    return x
```

Modeli "eğitelim" (yani en düşük hatayı veren parametreyi bulalım)

en_iyi_parametre = gradient_descent()

Çıktıda hatanın her adımda azaldığını ve parametrenin 0'a yaklaştığını göreceksiniz.

Türev (gradyan), bize hatayı azaltmak için hangi yöne gitmemiz gerektiğini söyleyen bir pusula görevi görür.

Analojilerle Anlatım

1. Sisli Dağda İniş Analojisi

- **Problem:** Bir dağın tepesindesiniz, her yer sisli ve amacınız vadinin en alçak noktasına (minimum hata) inmek.
- f (Fonksiyon) = Bulunduğunuz **rakım** (yükseklik).
- x (Parametre) = Doğu-Batı yönündeki **konumunuz**.
- df/dx (Türev) = Ayağınızın altındaki zeminin o anki **eğimi**.
    - Eğer zemin Doğu'ya doğru **yokuş yukarı** gidiyorsa (türev pozitifse), aşağı inmek için **Batı'ya** (ters yöne) doğru bir adım atarsınız.
    - Eğer zemin Doğu'ya doğru **yokuş aşağı** gidiyorsa (türev negatifse), aşağı inmek için **Doğu'ya** (aynı yöne) doğru bir adım atarsınız.
    - Her adımda bu işlemi tekrarlayarak, sisin içinde bile yavaş yavaş vadinin dibine ulaşırsınız.

2. Bisküvi Üretimi Analojisi

- **Problem:** Fırının sıcaklığını ayarlayarak yanık bisküvi sayısını en aza indirmek istiyorsunuz.
- f = **Yanık bisküvi sayısı** (Hata).
- x = Fırının **sıcaklık ayarı** (Parametre).
- df/dx = **Türev.** Size şunu söyler: "Sıcaklığı 1 derece artırırsam, yanık bisküvi sayısı artar mı, azalır mı ve ne kadar artar/azalır?"
    - Eğer türev pozitifse (df/dx = +5), sıcaklığı 1 derece artırmak yanık sayısını 5 artırıyor demektir. Demek ki sıcaklığı **azaltmalısınız.**
    - Eğer türev negatifse (df/dx = -3), sıcaklığı 1 derece artırmak yanık sayısını 3 azaltıyor demektir. Demek ki sıcaklığı biraz daha **artırabilirsiniz.**

3. Restoran Analojisi

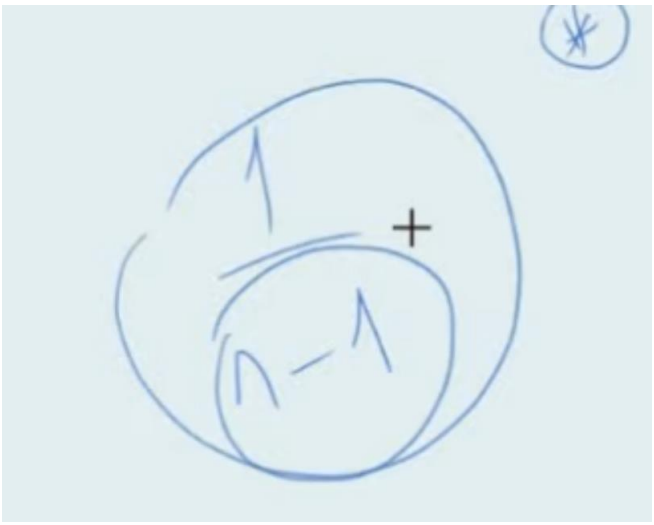
- **Problem:** Bir yemeğin tuz miktarını ayarlayarak müşteri memnuniyetini en üst seviyeye çıkarmak istiyorsunuz.
- f = **Müşteri şikayet sayısı** (Hata).
- x = Yemeğe eklenen **tuz miktarı** (Parametre).

- df/dx = **Türev.** Size şunu söyler: "Tuzu 1 gram artırırsam, şikayet sayısı artar mı, azalır mı?"
    - Eğer türev pozitifse, tuzu artırmak şikayetleri de artırıyor demektir (yemek çok tuzlu oluyor). Tuzu **azaltmalısınız.**
    - Eğer türev negatifse, tuzu artırmak şikayetleri azaltıyor demektir (yemek tuzsuz kalıyor). Tuzu biraz daha **artırmalısınız.**

Özet

- **Anahtar Kelime:** Türev (Derivative), Gradyan (Gradient), Eğim, Değişim Oranı, Gradyan İnişi (Gradient Descent).
- **Neden Önemli?:** Türev, makine öğrenmesi modellerinin "öğrenme" sürecinin temel mekanizmasıdır. Model, parametrelerini hangi yönde ve ne kadar değiştireceğini türev (gradyan) sayesinde öğrenir ve hatasını minimize eder.

8



Küçük ama çok önemli detay, istatistiksel tahminin temelindeki bir nüansı yakalıyor.

Görsel 7: n-1 Düzeltmesi (Bessel's Correction) - Dürüst Tahmin Sanatı

Formül Parçasının Adı: Bessel Düzeltmesi

Tek Cümlede: Elimizdeki küçük bir **örnekleme (sample)** bakarak, daha büyük bir **popülasyonun (population)** varyansını tahmin ederken, tahminimizin sistematik olarak düşük çıkmasını (yanlılık/bias) engellemek için kullanılan bir düzeltme faktörüdür.

Hangi Sorunu Çözer?

- Küçük bir gruba bakarak, bütün bir kitle hakkında nasıl daha dürüst ve isabetli bir "dağınıklık" tahmini yapabilirim?

Python Dünyası ile Anlatım

NumPy ve Pandas kütüphaneleri bu düzeltmeyi bizim için otomatik olarak yapar. Varyans hesaplarken, varsayılan olarak n-1'i kullanırlar çünkü genellikle bir örneklemle çalıştığımızı varsayarlar.

```
    import numpy as np

# Popülasyon: Türkiye'deki 10 milyon seçmenin yaşı (Tüm veri)
# Bu veriye genellikle sahip olamayız, bu yüzden hayali bir popülasyon yaratalım.
populasyon_yaslari = np.random.randint(18, 80, size=10_000_000)

# Örneklem: Sokakta rastgele seçtiğimiz 100 kişinin yaşı
orneklem_yaslari = np.random.choice(populasyon_yaslari, size=100)

# 1. Gerçek Popülasyon Varyansı (Bunu normalde bilmeyiz)
gercek_varyans = populasyon_yaslari.var()
# Not: NumPy'ın .var() fonksiyonu varsayılan olarak N'e böler.

# 2. Örneklem Varyansı (Yanlı Tahmin - N'e bölerek)
yanli_tahmin = orneklem_yaslari.var() # Paydada N=100 kullanılır

# 3. Örneklem Varyansı (Düzeltilmiş Tahmin - N-1'e bölerek)
# ddof='delta degrees of freedom' parametresi paydayı N - ddof yapar.
# ddof=1 demek, paydayı N-1 yap demektir.
durust_tahmin = orneklem_yaslari.var(ddof=1)

print(f"Gerçek Popülasyon Varyansı: {gercek_varyans:.2f}")
print("-" * 35)
print(f"Yanlı (N'e bölünmüş) Tahmin: {yanli_tahmin:.2f}")
print(f"Dürüst (N-1'e bölünmüş) Tahmin: {durust_tahmin:.2f}")
```

Çıktıda, Dürüst Tahmin'in (n-1 ile hesaplanan) Gerçek Popülasyon Varyansı'na genellikle Yanlı Tahmin'den daha yakın olduğunu göreceksiniz. 1/(n-1) kullanmak, tahminimizi daha isabetli yapar.

Analojilerle Anlatım

Problem: Örneklemler (küçük gruplar), genellikle gerçek popülasyonun aşırı uçlarını (en uzun/en kısa, en zengin/en fakir) yakalayamazlar. Bu yüzden, örnekleme bakarak yapılan varyans tahmini, gerçeği olduğundan **daha az dağınık** (daha düşük varyanslı) gösterme eğilimindedir.

1. Sınıf Temsilcisi Analojisi

- **Popülasyon:** Okuldaki tüm öğrenciler.
- **Örneklem:** Sınıfınızdaki 30 öğrenci.

- **Problem:** Sınıfınıza bakarak, tüm okulun boy "dağınıklığını" (varyansını) tahmin etmeniz isteniyor. Sınıfınızda muhtemelen okulun en uzun basketbolcusu veya en kısa öğrencisi yoktur. Sınıfınız, okula göre daha "homojen" bir gruptur.
- **n-1 Düzeltmesi:** Bu durumu bildiğiniz için, sınıfınızın varyansını hesapladıktan sonra sonucu bir miktar **"şişirerek"** rapor edersiniz. Bu "şişirme" işlemi, okulun genelindeki o aşırı uçları temsil edememenizin yarattığı hatayı telafi etmeye çalışır. Paydayı n'den n-1'e düşürmek, matematiksel olarak bu "şişirme" işlemini yapar.

2. Bisküvi Üretimi Analojisi

- **Popülasyon:** Fabrikada o gün üretilen 1 milyon bisküvinin tamamı.
- **Örneklem:** Kalite kontrol için hattan rastgele alınan 50 bisküvilik bir numune.
- **Problem:** Bu 50 bisküviye bakarak, günün tüm üretimindeki ağırlık "dağınıklığını" (varyansını) tahmin etmelisiniz. Bu 50 bisküviye, fırının en sıcak ve en soğuk noktasından çıkan en aşırı (en hafif ve en ağır) bisküvilerin denk gelme olasılığı düşüktür.
- **n-1 Düzeltmesi:** Numunenizin, genel üretimin varyansını **düşük tahmin etme** eğiliminde olduğunu bilirsiniz. Bu yüzden, numunenizin varyansını n-1'e bölerek sonucu biraz **büyütürsünüz.** Bu, tahmininizi gerçeğe daha yakın hale getiren bir "dürüstlük ayarıdır".

3. Restoran Analojisi

- **Popülasyon:** Bir ay boyunca restorana gelen tüm müşteriler.
- **Örneklem:** Sadece bir Salı akşamı gelen 20 kişilik bir anket grubu.
- **Problem:** Bu 20 kişiye bakarak, tüm ay boyunca gelen müşterilerin harcama "dağınıklığını" (varyansını) tahmin ediyorsunuz. Bu gruba, ayın en çok para harcayan VIP müşterisi ile sadece bir çorba içip kalkan müşterinin aynı anda denk gelme ihtimali azdır.
- **n-1 Düzeltmesi:** Anket grubunuzun varyansını hesaplarken n-1 kullanarak, tüm aydaki o aşırı harcama davranışlarını kaçırmış olabileceğinizi hesaba katarsınız ve tahmininizi buna göre düzeltirsiniz.

Özet

- **Anahtar Kelime:** Örneklem (Sample), Popülasyon (Population), Bessel Düzeltmesi, Yanlı Tahmin (Biased Estimate), Yansız Tahmin (Unbiased Estimate).
- **Kural:** Eğer elinizdeki veri, daha büyük bir grubun sadece bir **parçasıysa (örneklem)**, varyansı hesaplarken **n-1** kullanılır. Eğer elinizdeki veri, incelemek istediğiniz grubun **tamamıysa (popülasyon)**, o zaman **N** kullanılır. Pratikte neredeyse her zaman bir örneklemle çalıştığımız için n-1 standart yaklaşımdır.

Görseldeki ifade 1 / (n-1) muhtemelen **Örneklem Varyansı (Sample Variance)** formülünün bir parçasıdır.

Kısaca: Bu, verinin tamamına değil, sadece **küçük bir parçasına (örneklem)** bakarak tahmin yaptığımızda kullandığımız bir **düzeltme faktörüdür**.

Neden n Değil de n-1? (En İyi Analoji: SAP PRD vs. DEV Sistemi)

Bu farkı anlamak için "Popülasyon" ve "Örneklem" kavramlarını anlamamız gerek.

- **Popülasyon (Population):** Analiz etmek istediğiniz **verinin tamamı**. Bütün evren.
    - **ABAP Analojisi:** Canlı sisteminizdeki (**PRD**) KNA1 müşteri tablosunun **tamamı**. Milyonlarca satır. Bu "nihai gerçektir".
- **Örneklem (Sample):** Popülasyonun içinden seçtiğiniz **küçük bir alt küme**.
    - **ABAP Analojisi:** Geliştirme sisteminize (**DEV**) test için çektiğiniz ilk 10.000 müşteri. Veya PRD'de sadece belirli bir satış organizasyonu (VKORG = 1000) için çektiğiniz müşteri listesi.

Şimdi gelelim asıl probleme:

Sizden PRD'deki **tüm** müşterilerin yaşlarının "dağınıklığını" (Varyansını) bulmanız istendi. Ama PRD'ye bağlanıp milyonlarca satırı analiz etmek çok uzun sürecek. Siz de pratik bir yol seçiyorsunuz: "DEV sistemindeki 10.000 müşterilik örnekleme bakıp, buradan bir **tahmin** yürüteyim." diyorsunuz.

İşte burada kritik bir sorun ortaya çıkıyor:

Örneklemler (küçük gruplar), genellikle gerçeği olduğundan daha az "dağınık" gösterirler.

Neden? Düşünün ki Türkiye'deki (popülasyon) tüm insanların boy ortalamasını ve boyların ne kadar dağınık olduğunu bulmak istiyorsunuz. Rastgele 10 kişi (örneklem) seçtiniz. Bu 10 kişinin arasına Türkiye'nin en uzun insanı ile en kısa insanının aynı anda denk gelme olasılığı ne kadar düşüktür? Çok çok düşük.

Seçtiğiniz 10 kişi, muhtemelen ortalamaya daha yakın boylarda olacaktır. Yani sizin örnekleminiz, gerçek popülasyonun o "aşırı uçlarını" (en uzun ve en kısa) kaçıracağı için, size boyların olduğundan **daha az dağınık** olduğu yanılgısını verir.

İşte n-1'in Sihri: Bessel's Correction (Bessel Düzeltmesi)

İstatistikçiler bu sorunu fark etmişler. Demişler ki: "Eğer bir örneklemin varyansını hesaplarken n'ye bölersek, sistematik olarak gerçek varyansı **olduğundan daha düşük tahmin etmiş oluruz** (biased estimate)."

Bu hatayı nasıl düzeltebiliriz?

Matematiksel olarak, paydayı n yerine biraz daha küçük bir sayı olan n-1'e düşürürsek, sonuç biraz daha büyük çıkar.

1 / (n-1) ifadesi, bu **düzeltme faktörüdür**. Sayıyı n yerine n-1'e bölmek, çıkan sonucu bir miktar "şişirir". Bu yapay şişirme, örneklemin o "aşırı uçları" kaçırmasından kaynaklanan hatayı telafi eder ve tahminimizi gerçeğe (popülasyon varyansına) daha çok yaklaştırır.

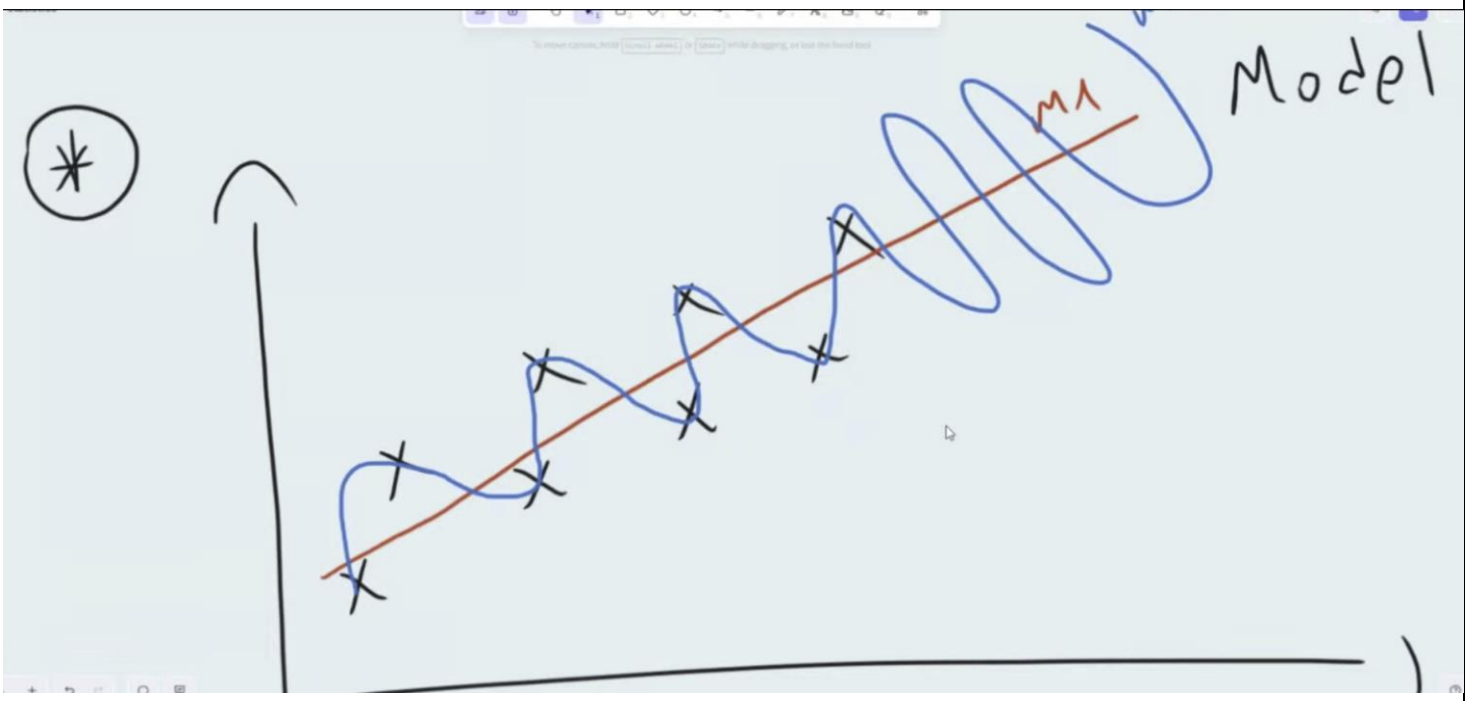
Bu, tahminimizi daha **"dürüst"** veya **"tarafsız" (unbiased)** bir tahmin haline getirir.

Özetle:

- **Eğer verinin tamamına sahipseniz** (PRD'deki tüm KNA1 tablosu), varyansı hesaplarken **N'ye bölersiniz.** (Bu teoriktir, genelde tüm veriye sahip olamayız.)
- **Eğer verinin sadece bir parçasıyla (örneklem) tahmin yapıyorsanız** (DEV'deki 10.000 müşteri), varyansı hesaplarken **n-1'e bölersiniz.**

Bu n-1'i, tahmininizin daha isabetli olmasını sağlayan bir **"güvenlik ayarı"** veya **"hata düzeltme katsayısı"** olarak düşünebilirsiniz. ML'de neredeyse her zaman verinin bir örneklemiyle çalıştığımız için, pratikte hep n-1'li versiyonlar kullanılır.

9



Bu görsel, makine öğrenmesinin en temel ve en önemli ikilemlerinden birini anlatıyor.

Görsel 8: Bias-Varyans Dengesi - Öğrenmek mi, Ezberlemek mi?

Genel Anlam: Neyi Anlatıyor?

Bu görsel, bir veri setindeki deseni öğrenmeye çalışan iki farklı model yaklaşımını karşılaştırır. Temel soru şudur: "İyi bir model, eldeki veriye mükemmel bir şekilde uyan mıdır, yoksa verinin altındaki genel trendi yakalayan mıdır?" Bu, **Bias-Varyans Dengesi (Bias-Variance Tradeoff)** olarak bilinen temel bir konsepttir.

- **Siyah 'X'ler:** Bunlar bizim **eğitim verilerimizdir** (gerçek ölçümler).
- **Kırmızı Düz Çizgi (Model 1):** Basit, doğrusal bir model.
- **Mavi Kıvrımlı Çizgi (Model 2):** Karmaşık, esnek bir model.

Python Dünyası ile Anlatım

Bu durumu, farklı derecelerdeki polinomlarla bir veri setine uymaya çalışarak simüle edebiliriz.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression

# Gerçek veri (biraz gürültü eklenmiş)
np.random.seed(0)
X = np.linspace(-5, 5, 10).reshape(-1, 1)
y = 0.5 * X + np.random.randn(10, 1)

# Model 1: Basit Doğrusal Model (Kırmızı Çizgi gibi)
model1 = LinearRegression()
model1.fit(X, y)

# Model 2: Karmaşık Polinomsal Model (Mavi Çizgi gibi)
# 9. dereceden bir polinom, 10 noktayı ezberlemeye çalışır.
model2 = make_pipeline(PolynomialFeatures(degree=9), LinearRegression())
model2.fit(X, y)

# Görselleştirme
X_plot = np.linspace(-5, 5, 100).reshape(-1, 1)
plt.scatter(X, y, color='black', label='Veri Noktaları (X)')
plt.plot(X_plot, model1.predict(X_plot), color='red', label='Model 1 (Basit - İyi
Genelleme)')
plt.plot(X_plot, model2.predict(X_plot), color='blue', label='Model 2 (Karmaşık -
Ezberci/Overfitting)')
plt.legend()
plt.show()
```

Bu kod, basit kırmızı çizginin verinin genel trendini yakaladığını, karmaşık mavi çizginin ise her bir noktayı "ezberlemeye" çalışarak anlamsız salınımlar yaptığını gösterir.

Model 1: Kırmızı Düz Çizgi (Düşük Varyans, Yüksek Bias)

- **Yaklaşımı:** "Verinin altında yatan ana hikaye nedir?" diye sorar. Verideki küçük ve muhtemelen anlamsız olan zikzakları (gürültüyü) görmezden gelir ve genel trendi yakalamaya odaklanır.
- **Özellikleri:**

    - **Yüksek Bias (Önyargı):** Model, "ilişki doğrusaldır" gibi güçlü bir varsayımda bulunur. Bu bir önyargıdır.
    - **Düşük Varyans (Ezbercilik):** Model, veri değişse bile çok fazla değişmez, stabildir. Ezber yapmaz.
- **Sonuç: İyi genelleme yapar.** Daha önce görmediği yeni bir veri noktası geldiğinde, onu da bu genel trende göre tahmin edeceği için çok daha başarılı olur. **Bu, daha iyi ve daha güvenilir modeldir.**

Model 2: Mavi Kıvrımlı Çizgi (Yüksek Varyans, Düşük Bias)

- **Yaklaşımı:** "Her bir veri noktasına nasıl mükemmel bir şekilde uyabilirim?" diye sorar. Verinin ana hikayesini anlamak yerine, eldeki her bir veri noktasını kusursuzca ezberlemeye çalışır.
- **Özellikleri:**
    - **Düşük Bias (Önyargı):** Modelin neredeyse hiç varsayımı yoktur; her detayı kabul eder.
    - **Yüksek Varyans (Ezbercilik):** Eğitim verisindeki en ufak bir değişiklik bile modelin şeklini tamamen değiştirir. Aşırı hassas ve değişkendir. Tam bir ezbercidir.
- **Sonuç: Aşırı Öğrenme (Overfitting) yapar.** Eğitim verisinde hatası sıfıra yakın olsa da, daha önce hiç görmediği yeni bir veri noktası geldiğinde, ezberinde olmayan bu yeni duruma nasıl tepki vereceğini bilemez ve çok saçma bir tahminde bulunur.

Analojilerle Anlatım

1. Sınava Hazırlanan Öğrenci Analojisi

- **Kırmızı Çizgi:** Konunun **mantığını anlayan** öğrenci. Hazırlık testinde belki birkaç hata yapar ama gerçek sınavda, sorular değişse bile mantığını kullanarak başarılı olur.
- **Mavi Çizgi:** Hazırlık kitabındaki tüm soruları **ezberleyen** öğrenci. Hazırlık testinde %100 başarılı olur ama gerçek sınavda, sorulardaki en ufak bir değişiklikte çuvallar.

2. Bisküvi Üretimi Analojisi

- **Kırmızı Çizgi:** Kalite kontrol mühendisi, üretimdeki genel trendi anlar: "Fırın sıcaklığı arttıkça bisküviler koyulaşıyor." Bu genel kurala göre ayarlamalar yapar.
- **Mavi Çizgi:** Tecrübesiz bir operatör, her bir bisküvinin rengindeki anlamsız dalgalanmalara göre fırın ayarını sürekli değiştirir. Sonuçta fırının dengesini tamamen bozar ve daha kötü bisküviler üretir.

3. Restoran Analojisi

- **Kırmızı Çizgi:** Deneyimli bir şef, müşterilerin genel eğilimini bilir: "Müşteriler genellikle baharatlı yemekleri seviyor." Bu genel kurala göre menüsünü oluşturur.
- **Mavi Çizgi:** Amatör bir aşçı, o gün gelen bir müşterinin "Ben hiç tuz sevmem" demesiyle tüm yemekleri tuzsuz yapar, ertesi gün gelen başka bir müşterinin "Çok tuzlu severim" demesiyle tüm yemekleri aşırı tuzlu yapar. Her bir müşteriyi "ezberlemeye" çalışarak genel lezzet dengesini kaybeder.

Özet

- **Anahtar Kelime:** Bias-Varyans Dengesi (Tradeoff), Aşırı Öğrenme (Overfitting), Eksik Öğrenme (Underfitting), Genelleme (Generalization).
- **Neden Önemli?:** Makine öğrenmesinde asıl amaç, elimizdeki veriyi ezberlemek değil, o verinin altında yatan **genel deseni ve hikayeyi öğrenmektir.** Bu yüzden genellikle daha basit ve iyi genelleme yapan **Kırmızı Çizgi (Model 1)**, karmaşık ve ezberci olan **Mavi Çizgi (Model 2)**'den çok daha iyi bir modeldir.

Bu görseldeki iki "modeli" (aslında birisi model, diğeri ise verinin kendisi olabilir ama biz iki farklı model yaklaşımı olarak ele alalım) isimlendirelim:

- **Model 1 (Kırmızı Düz Çizgi):** Basit, genel trendi yakalamaya çalışan model.
- **Model 2 (Mavi Kıvrımlı Çizgi):** Her bir noktaya mükemmel şekilde uymaya çalışan karmaşık model.

İlk Bakışta Cevap

Eğer soru "Hangi çizgi mevcut verilere daha yakın?" ise, cevap kesinlikle **Mavi Çizgi (Model 2)**'dir. Mavi çizgi, eldeki X işaretli veri noktalarının neredeyse hepsinin üzerinden geçiyor, yani hatası çok düşük.

Gerçek ve Doğru Cevap

İlk bakışta Mavi Çizgi daha iyi gibi görünse de, gerçek dünyada **Kırmızı Çizgi (Model 1) neredeyse her zaman daha iyi ve daha güvenilir bir modeldir.**

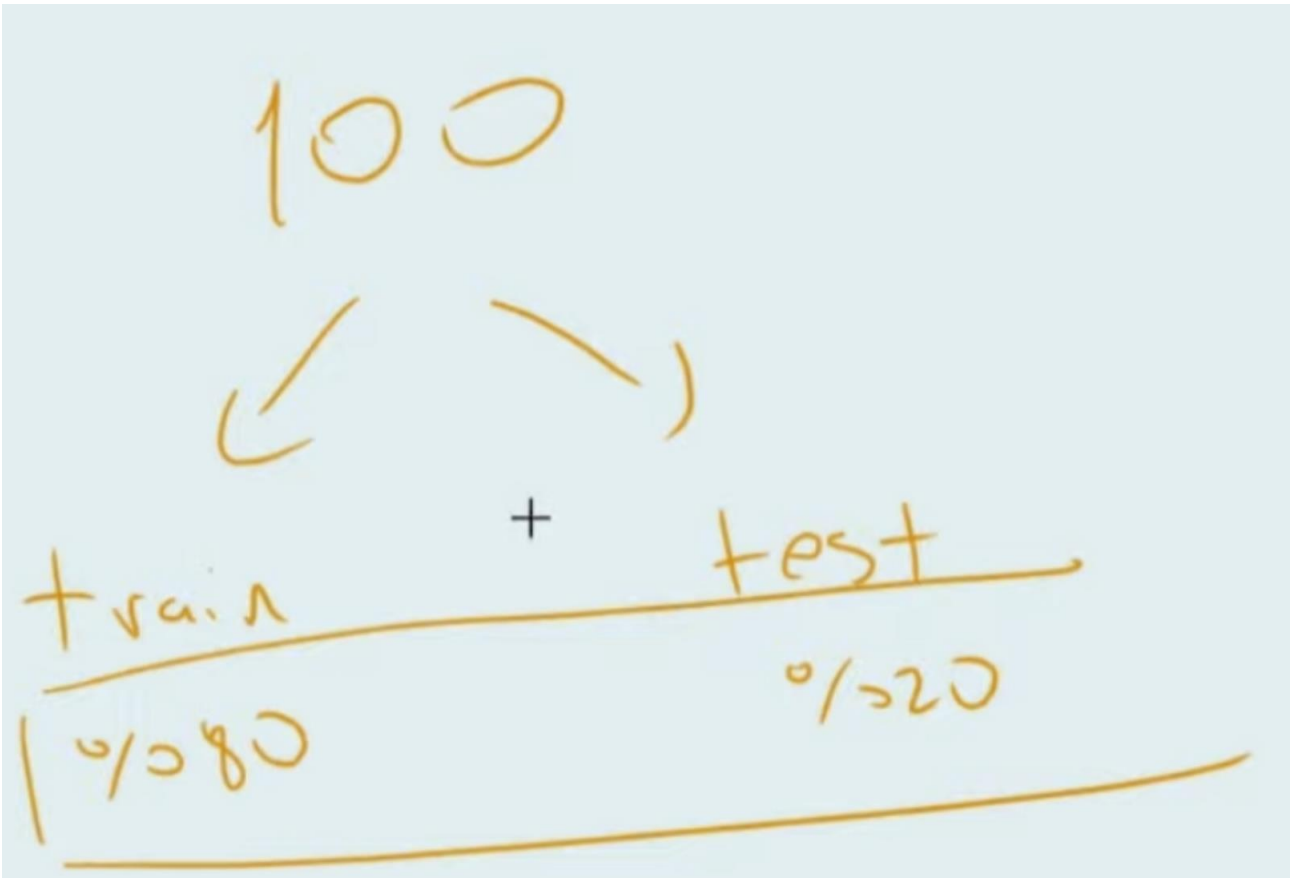
Çünkü makine öğrenmesinde asıl amacımız, elimizdeki veriyi bir papağan gibi tekrar etmek değil, o verinin altında yatan **gizli deseni, kuralı ve hikayeyi öğrenmektir.**

Unutmayın; iyi bir model ezberlemez, **ÖĞRENİR!**

SAP ABAP Analojisi

- **Kırmızı Çizgi (Model 1):** Sağlam, iyi tasarlanmış bir ABAP programı. Program, IF...ELSEIF...ENDIF bloklarıyla genel iş kurallarını (örneğin müşteri ülkesine göre KDV hesaplama) anlar. Yeni bir müşteri geldiğinde de bu genel kurallar sayesinde doğru KDV'yi hesaplar.
- **Mavi Çizgi (Model 2):** Kötü yazılmış, "hard-code" edilmiş bir ABAP programı. Programda şöyle satırlar vardır: CASE customer_id. WHEN '1001'. kdv = 18. WHEN '1002'. kdv = 18. ... ENDCASE. Bu program mevcut 1000 müşteri için mükemmel çalışır. Ama sisteme **1003** ID'li yeni bir müşteri geldiğinde, CASE içinde bir karşılığı olmadığı için ya hata verir ya da yanlış hesaplama yapar

10



Bu görsel, bir önceki teorik konunun ("Öğrenmek mi, Ezberlemek mi?") pratik uygulamasıdır

Görsel 9: Train-Test Split - "Dürüstlük Sınavı"

Genel Anlam: Neyi Anlatıyor?

Bu görsel, bir makine öğrenmesi modelinin **genelleme yeteneğini dürüstçe ölçmek** için veri setinin nasıl bölünmesi gerektiğini anlatır. Bu, modelimizin "ezberci öğrenci" olup olmadığını anlamamızı sağlayan en temel test yöntemidir.

- **100:** Elimizdeki toplam veri setini temsil eder (100 satır veya %100).
- **train (%80):** Verinin büyük bir kısmı, modelin "ders çalışması" için ayrılır.
- **test (%20):** Verinin küçük bir kısmı, modelin "final sınavı" için, hiç görmeyeceği şekilde kenara ayrılır.

Python Dünyası ile Anlatım

scikit-learn kütüphanesi bu işlemi bizim için tek bir satırda yapar.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Elimizdeki 100 satırlık veri seti
# ... (df = pd.read_csv('veriseti.csv')) ...
# X = df[['ozellik1', 'ozellik2']]
# y = df['hedef']

# 1. Veriyi Bölme (Train-Test Split)
# Veri setini %80 train, %20 test olarak ayır.
# stratify=y, 'hedef' sınıfının oranını her iki sette de korur.
# random_state, bölme işleminin her seferinde aynı şekilde yapılmasını sağlar.
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.20, random_state=42, stratify=y
)

# 2. Modeli SADECE Eğitim Verisiyle Eğitme
model = LogisticRegression()
model.fit(X_train, y_train)

# 3. Modelin Performansını Değerlendirme
# Eğitim setindeki başarısı (kopya çekerek çözdüğü sınav)
train_tahminleri = model.predict(X_train)
train_basarisi = accuracy_score(y_train, train_tahminleri)
print(f"Eğitim Seti Başarısı (Kopya): {train_basarisi:.2f}")

# Test setindeki başarısı (gerçek final sınavı)
test_tahminleri = model.predict(X_test)
test_basarisi = accuracy_score(y_test, test_tahminleri)
print(f"Test Seti Başarısı (Gerçek Sınav): {test_basarisi:.2f}")

# Eğer train_basarisi çok yüksek (%99) ve test_basarisi çok düşükse (%70),
# bu modelin "ezberci" (overfitting) olduğunu anlarız.
```

Analojilerle Anlatım

1. Sınava Hazırlanan Öğrenci Analojisi

- **100 (Veri Seti):** Öğrencinin sınava çalışması için verilen **100 soruluk hazırlık kitabı.**
- **train (%80):** Öğretmenin, öğrenciye cevap anahtarıyla birlikte verdiği **80 soruluk çalışma seti.** Öğrenci bu sorulara bakarak konuyu öğrenir.
- **test (%20):** Öğretmenin, öğrencinin hiç görmemesi için kilitli bir kasaya koyduğu **20 soruluk gerçek final sınavı.** Öğrencinin asıl başarısı, bu daha önce hiç görmediği 20 soruya verdiği doğru cevaplarla ölçülür.

2. Bisküvi Üretimi Analojisi

- **100 (Veri Seti):** Test etmek için üretimden alınan **100 bisküvilik bir numune.**
- **train (%80):** Kalite kontrol makinesine (model) gösterilen ve etiketlenen (sağlam/kırık) **80 bisküvi.** Makine, "kırık bisküvinin" neye benzediğini bu 80 örnek üzerinden öğrenir.
- **test (%20):** Makinenin daha önce hiç görmediği, kenara ayrılmış **20 bisküvi.** Makinenin gerçek performansı, bu 20 bisküviyi ne kadar doğru sınıflandırdığı ile ölçülür.

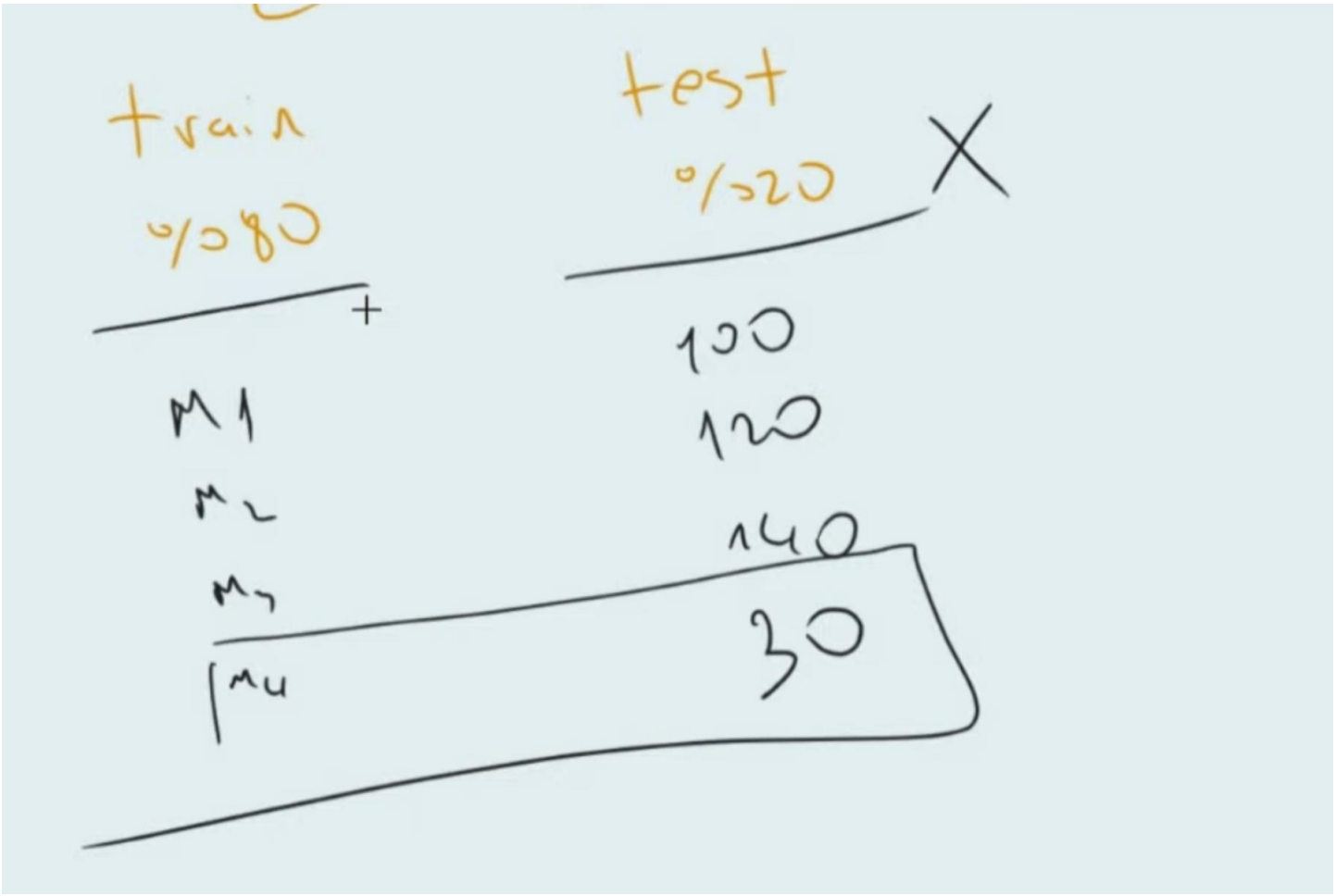
3. Restoran Analojisi

- **100 (Veri Seti):** Yeni bir yemeğin beğenilip beğenilmeyeceğini tahmin etmek için, geçmişte sunulan **100 farklı yemeğe ait müşteri yorumları.**
- **train (%80):** Modelin, hangi özelliklere (baharatlı, tatlı, ekşi vb.) sahip yemeklerin beğenildiğini öğrenmesi için verilen **80 yemeğin yorumları.**
- **test (%20):** Modelin, öğrendiklerini test etmek için kullanılan, daha önce hiç görmediği **20 yemeğin yorumları.** Modelin, bu 20 yemeğin müşteri puanını ne kadar isabetli tahmin ettiği, onun gerçek başarısını gösterir.

Özet

- **Anahtar Kelime:** Train-Test Split, Overfitting Tespiti, Genelleme Performansı.
- **Neden Önemli?:** Bu ayrım, bizi **Overfitting (Aşırı Öğrenme / Ezberleme)** tuzağından koruyan en temel sigortadır. Bir modelin sadece geçmişi ezberleyen bir papağan mı, yoksa geleceği tahmin edebilen akıllı bir sistem mi olduğunu anlamamızı sağlayan dürüst ve vazgeçilmez bir değerlendirme yöntemidir.

11



Bu görsel, Train-Test ayrımının bir adım ötesine geçerek, makine öğrenmesi projelerindeki en yaygın ve en tehlikeli hatalardan birini anlatıyor: **Veri Sızıntısı (Data Leakage)** ve bu sızıntıyı önlemek için neden **Validation Set**'e ihtiyacımız olduğu.

Görsel 10: Test Setine Dokunma! - Veri Sızıntısı ve Validation Set'in Önemi

Genel Anlam: Neyi Anlatıyor?

Bu görsel, birden çok model (M1, M2, M3, M4) arasından en iyisini seçmeye çalışırken, bu seçim sürecinde **Test Seti'ni kullanmanın neden korkunç bir hata olduğunu** anlatır.

- M1, M2...: Farklı algoritmalar (örn: LogisticRegression, RandomForest) veya aynı algoritmanın farklı ayarları (hiperparametreler).
- test %20 üzerindeki **X** işareti: "**BU BÖLGEYE DOKUNMAK YASAK!**" anlamına gelir.
- 100, 120, 140, 30: Modellerin test seti üzerindeki (varsayımsal) hata skorları. En düşük hata (30) M4 modeline aittir.

Python Dünyası ile Anlatım (HATALI YAKLAŞIM)

Aşağıdaki kod, görseldeki **yanlış** yaklaşımı simüle eder:

```
# Veri setini sadece train ve test olarak ikiye böldük.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)

# Farklı modelleri train setiyle eğitelim
model1 = ModelA().fit(X_train, y_train)
model2 = ModelB().fit(X_train, y_train)
model3 = ModelC().fit(X_train, y_train)

# ---- HATALI ADIM: En iyi modeli seçmek için TEST setini kullanmak ----
skor1 = model1.score(X_test, y_test)
skor2 = model2.score(X_test, y_test)
skor3 = model3.score(X_test, y_test)

# En iyi skoru veren modeli "şampiyon" ilan edelim.
# Diyelim ki Model 2 en iyi skoru (%95) verdi.
print("En iyi model Model 2!")

# PROBLEM: Artık Model 2'nin %95'lik skoru dürüst bir skor değil.
# Çünkü model seçimi sürecinde test setinden "bilgi sızdırdık".
# Model 2, belki de şans eseri o 20'lik test setine çok iyi uyum sağladı.
# Gerçek dünyada performansı muhtemelen %95'ten düşük olacaktır.
```

Bu sürece **"Test Setine Sızma (Leaking to the Test Set)"** denir. Test seti, artık "daha önce hiç görülmemiş" bir veri olmaktan çıkmıştır.

Doğru Yaklaşım: Validation Set Kullanımı

Bu hatayı önlemek için veri setimizi **üçe böleriz:** Train, Validation ve Test.

1. **Train Set (%70):** Tüm modelleri (M1, M2, M3, M4) bu veriyle eğitiriz.
2. **Validation Set (%15):** Eğitilmiş tüm modelleri bu sette yarıştırırız. En iyi performansı gösteren modeli (diyelim ki M4) "şampiyon" olarak seçeriz.
3. **Test Set (%15):** Diğer tüm modelleri çöpe atarız. Sadece şampiyon olan M4 modelini, daha önce **hiç dokunulmamış** bu sette **bir kere** çalıştırırız. Bu testten aldığı sonuç, modelin nihai ve dürüst performans notudur.

Analojilerle Anlatım

1. Sınava Hazırlanan Öğrenci Analojisi

- **Problem:** Dört farklı öğretim yönteminden (M1, M2, M3, M4) hangisinin en iyi olduğuna karar vermek istiyorsunuz.

- **Hatalı Yaklaşım:** Dört yöntemle hazırlanan dört öğrenciyi de doğrudan **final sınavına (Test Seti)** sokmak. En yüksek notu alan yöntemi en iyi ilan etmek. Bu, final sınavını bir "deneme sınavı" olarak kullanmaktır ve sınavın geçerliliğini yok eder.
- **Doğru Yaklaşım:**
    1. Tüm öğrencileri **hazırlık kitabıyla (Train Seti)** eğit.
    2. Hepsini, daha önce görmedikleri bir **deneme sınavına (Validation Seti)** sok ve en iyi yöntemi seç.
    3. Sadece seçtiğin en iyi öğrenciyi, kilitli kasadaki **gerçek final sınavına (Test Seti)** sok.

2. Bisküvi Üretimi Analojisi

- **Problem:** Dört farklı prototip kalite kontrol makinesinden (M1-M4) hangisinin en iyisi olduğunu bulmak.
- **Hatalı Yaklaşım:** Dört makineyi de, en başından beri "nihai denetim" için ayırdığınız **test bisküvileri (Test Seti)** üzerinde yarıştırmak.
- **Doğru Yaklaşım:**
    1. Tüm makineleri **eğitim bisküvileriyle (Train Seti)** eğit.
    2. Hepsini, "ara denetim" için ayırdığınız **doğrulama bisküvileri (Validation Seti)** üzerinde yarıştır ve en iyi makineyi seç.
    3. Sadece seçtiğin en iyi makineyi, el değmemiş **nihai denetim bisküvileri (Test Seti)** üzerinde bir kez çalıştırarak resmi performansını ölç.

3. Restoran Analojisi

- **Problem:** Dört farklı şef adayından (M1-M4) hangisinin restoranın baş aşçısı olacağına karar vermek.
- **Hatalı Yaklaşım:** Dört adayın hepsine, restoranın "büyük açılış gecesi" menüsünü (Test Seti) hazırlatmak ve müşteri tepkisine göre en iyisini seçmek. Bu, açılış gecesini riske atmaktır.
- **Doğru Yaklaşım:**
    1. Tüm adaylara standart tariflerle (**Train Seti**) pratik yaptır.
    2. Hepsine, patron ve gurmelerden oluşan bir jüri önünde özel bir "tadın günü" (**Validation Seti**) düzenle ve baş aşçıyı seç.
    3. Sadece seçtiğin baş aşçının, "büyük açılış gecesi"nde (**Test Seti**) yemek yapmasına izin ver.

Özet

- **Anahtar Kelime:** Veri Sızıntısı (Data Leakage), Validation Seti, Model Seçimi, Hiperparametre Optimizasyonu.
- **Kural:** En iyi modeli veya en iyi ayarları bulmak için yapılan tüm denemeler **Validation Seti** üzerinde yapılmalıdır. **Test Seti**, kilitli bir kasadır ve projenin en sonunda, sadece seçilmiş olan tek bir modelin nihai performansını ölçmek için **sadece bir kez** kullanılır.

Görselin Anlattığı Hikaye: "En İyi Öğrenciyi Nasıl Seçeriz?"

Bir önceki adımda ne yapmıştık? 100 soruluk hazırlık kitabımızı ikiye ayırmıştık:

- **Train (%80):** Öğrencilerin çalışacağı 80 soruluk "Hazırlık Kitabı".
- **Test (%20):** Kilitli kasada duran, öğrencinin asla görmediği 20 soruluk "Final Sınavı".

Şimdi, siz bir öğretmen olarak en iyi sonucu almak istiyorsunuz ve farklı öğretim yöntemleri denemeye karar veriyorsunuz.

- **M1, M2, M3, M4:** Bunlar sizin denediğiniz **4 farklı model veya 4 farklı öğretim yöntemidir**.
    - **M1:** Öğrenciye sadece konuları okutmak.
    - **M2:** Öğrenciye bol bol soru çözdürmek.
    - **M3:** Öğrenciye videolar izletmek.
    - **M4:** Öğrenciyi bir çalışma grubuna dahil etmek.

Bu 4 yöntemin hepsini **%80'lik Eğitim Seti (Train data)** üzerinde deniyorsunuz. Yani 4 farklı öğrenciyi, 4 farklı metotla 80 soruluk hazırlık kitabına çalıştırıyorsunuz.

Şimdi geldik en kritik soruya: **Bu 4 yöntemden hangisinin en iyisi olduğuna nasıl karar vereceğiz?**

Görseldeki HATA: Final Sınavını Kopya Olarak Kullanmak

Görselde, **Test Seti'nin üzerinde büyük bir "X" işareti var.** Bu, "DOKUNMA!", "YASAK BÖLGE!" anlamına geliyor. Ama diyelim ki acemi bir öğretmen bu kuralı bilmiyor ve şöyle bir hata yapıyor:

"Elimde 4 tane eğitilmiş model (öğrenci) var. En iyisini bulmak için hepsini 20 soruluk Final Sınavı'na (Test Seti) sokayım. En yüksek notu alan en iyisidir!"

Bu denemenin sonuçları (muhtemelen hata puanları, yani düşük olan daha iyi):

- M1'in Test Skoru: 100
- M2'nin Test Skoru: 120
- M3'ün Test Skoru: 140
- **M4'ün Test Skoru: 30 (En iyi!)**

Öğretmen sevinçle bağırır: "Harika! En iyi yöntem M4'müş! Artık bu yöntemi kullanacağım."

Peki bu neden KORKUNÇ bir hatadır?

Çünkü öğretmen, en iyi modeli seçmek için **Final Sınavı'nın cevaplarını sızdırmış oldu!** Model seçimi sürecinde test verisini kullandığınız anda, o veri artık "daha önce hiç görülmemiş" bir veri olmaktan çıkar.

M4 modeli, belki de şans eseri o 20 soruluk test setine çok uygun bir modeldi. Siz M4'ü seçerek, aslında test setini ezberleme potansiyeli en yüksek olan modeli seçmiş oldunuz. Artık M4'ün "30" olan skoru **dürüst bir skor değildir.** Gerçekten yeni bir veri geldiğinde performansı muhtemelen çok daha kötü olacaktır.

Doğru Yöntem: Üçüncü Bir Set -> Validation (Doğrulama/Quiz) Seti

Peki doğrusu ne? Zeki bir öğretmen ne yapar?

Hazırlık kitabını 2'ye değil, 3'e böler:

1. **Train / Eğitim Seti (%70 - Hazırlık Kitabı):** Öğrenciler bu 70 soruya çalışır. (M1, M2, M3, M4 burada eğitilir).
2. **Validation / Doğrulama Seti (%15 - Deneme Sınavı / Quiz):** Öğretmen, 4 farklı yöntemle hazırladığı öğrencilerini, daha önce hiç görmedikleri bu 15 soruluk **Deneme Sınavı'na** sokar. En iyi yöntemin hangisi olduğuna **BU SINAVIN SONUCUNA GÖRE** karar verir. Diyelim ki M2 bu sınavda en iyi sonucu aldı.

3. **Test Seti (%15 - Gerçek Final Sınavı):** Öğretmen artık en iyi yöntemin M2 olduğuna karar vermiştir. Diğer tüm yöntemleri (M1, M3, M4) çöpe atar. Sadece M2 yöntemiyle hazırlanan öğrenciyi, kilitli kasada duran ve **HİÇ KİMSENİN DAHA ÖNCE GÖRMEDİĞİ** 15 soruluk Gerçek Final Sınavı'na sokar.

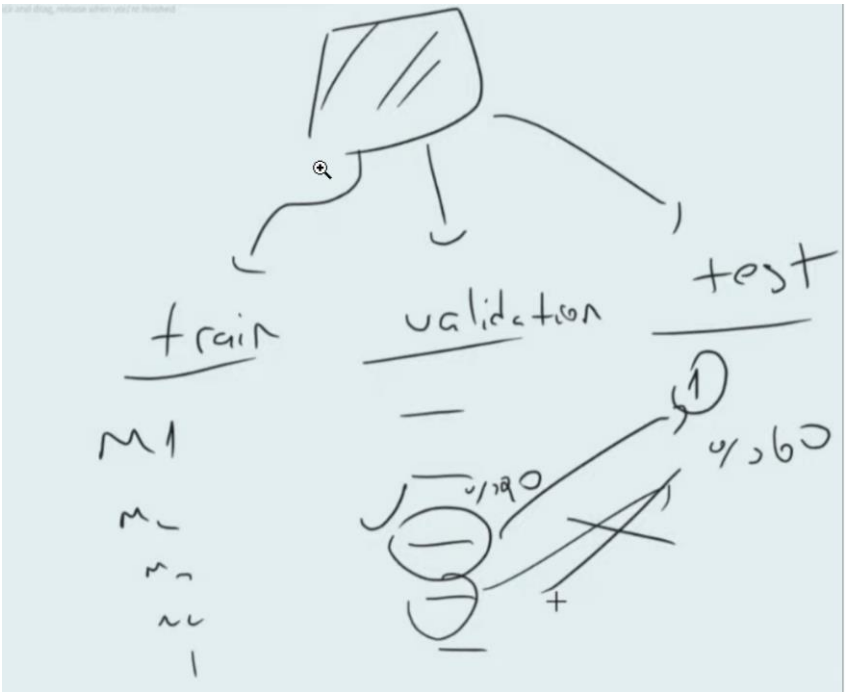
Bu sınavdan aldığı not, modelin **nihai, dürüst ve güvenilir başarı notudur.**

Özetle:

- **Train Set:** Modelleri *eğitmek* için kullanılır.
- **Validation Set:** Eğitilmiş modelleri *karşılaştırmak* ve en iyisini *seçmek* için kullanılır.
- **Test Set:** Seçtiğimiz en iyi modelin *gerçek dünya performansını* dürüstçe ölçmek için **sadece bir kere** kullanılır.

Görseldeki X işareti, "Validation setin yoksa, en iyi modeli seçmek için Test setini sakın kullanma!" diyen çok önemli bir uyarıdır.

12



Bu görsel, bir önceki görselde anlatılan hatalı yaklaşımın yerine **doğru ve profesyonel makine öğrenmesi iş akışını** çiziyor. "altın standard" .

Görsel 11: Doğru İş Akışı - Train, Validate, Test Disiplini

Genel Anlam: Neyi Anlatıyor?

Bu şema, güvenilir ve dürüst bir makine öğrenmesi modeli geliştirmek için verinin nasıl üçe bölünmesi gerektiğini ve her bir parçanın görevini görselleştirir. Bu, veri sızıntısını önleyen ve modelin gerçek dünya performansını doğru bir şekilde raporlamamızı sağlayan standart bir süreçtir.

- **Bilgisayar Simgesi:** Tüm veri setimiz.
- **Üç Dal:** Verinin üç farklı amaca hizmet eden parçalara ayrılması.
- **M1, M2...:** Farklı model adayları.
- **Validation'daki ✓ işareti:** En iyi modelin seçildiği yer.

- **Oklar:** Hangi modelin hangi aşamaya geçtiğini gösterir.

Python Dünyası ile Anlatım (DOĞRU YAKLAŞIM)

Bu iş akışını, scikit-learn kullanarak doğru şekilde kodlayalım:

```
from sklearn.model_selection import train_test_split

# ... X ve y (tüm veri) hazır ...

# 1. Adım: Veriyi ikiye böl (Train+Validation ve Test)
# Önce Test setini ayıralım ve bir kenara koyalım. Bu bizim kilitli kasamız.
X_temp, X_test, y_temp, y_test = train_test_split(
    X, y, test_size=0.15, random_state=42, stratify=y
)

# 2. Adım: Kalan veriyi Train ve Validation olarak böl
# Bu, modelleri eğiteceğimiz ve yarıştıracağımız alandır.
X_train, X_val, y_train, y_val = train_test_split(
    X_temp, y_temp, test_size=0.176, random_state=42, stratify=y_temp
) # (0.176 * 0.85 ≈ 0.15, böylece train %70, val %15, test %15 olur)

# ---- MODEL GELİŞTİRME DÖNGÜSÜ ----
# 3. Adım: Modelleri Train setiyle eğit
model1 = ModelA().fit(X_train, y_train)
model2 = ModelB().fit(X_train, y_train)

# 4. Adım: Modelleri Validation setinde yarıştır
skor1 = model1.score(X_val, y_val)
skor2 = model2.score(X_val, y_val)

# 5. Adım: Şampiyonu seç
# Diyelim ki Model 2, Validation setinde en iyi skoru aldı. Şampiyonumuz o!
sampiyon_model = model2

# ---- FİNAL TESTİ ----
# 6. Adım: Şampiyonu, el değmemiş Test setinde SADECE BİR KEZ test et
nihai_skor = sampiyon_model.score(X_test, y_test)

print(f"Modelimizin raporlanacak nihai başarısı: {nihai_skor:.2f}")
```

Analojilerle Anlatım

Bu üç adımlı süreç, her üç analojimize de mükemmel bir şekilde uyar:

Adım	**1. Train (Eğitim)**	**2. Validate (Doğrulama/Seçim)**	**3. Test (Final Sınavı)**
Sınav Analojisi	Tüm öğrencilere **Hazırlık Kitabı** verilir ve ders çalışırlar.	Tüm öğrenciler bir **Deneme Sınavı'na** sokulur ve en yüksek notu alan **"Okul Temsilcisi"** olarak seçilir.	Sadece seçilen "Okul Temsilcisi", hiç görülmemiş soruların olduğu **Resmi Final Sınavı'na** girer.
Bisküvi Analojisi	Tüm prototip makineler, **Eğitim Bisküvileriyle** kalibre edilir.	Tüm makineler, bir **Ara Denetim'de** (yeni bisküvilerle) yarıştırılır ve en başarılı olan **"Şampiyon Makine"** seçilir.	Sadece "Şampiyon Makine", el değmemiş **Nihai Kalite Kontrol** bisküvileri üzerinde bir kez çalıştırılır.
Restoran Analojisi	Tüm şef adayları, standart tariflerle **Mutfak Akademisi'nde** pratik yapar.	Tüm adaylar, patron ve gurmelerin önünde bir **"Baş Aşçı Seçmesi"** tadım gününe katılır ve en iyisi seçilir.	Sadece seçilen Baş Aşçı, restoranın **Büyük Açılış Gecesi'nde** gerçek müşterilere yemek yapar.

Özet

- **Anahtar Kelime:** Train-Validation-Test Split, Model Seçimi, Hiperparametre Optimizasyonu, Veri Sızıntısını Önleme.
- **Görselin Anlattığı Kural:**
    - **Train Seti:** Modelleri **eğitmek** için kullanılır.
    - **Validation Seti:** Eğitilmiş modelleri **karşılaştırmak** ve en iyisini **seçmek** için kullanılır.
    - **Test Seti:** Seçilmiş olan en iyi modelin **gerçek dünya performansını** dürüstçe ölçmek için **sadece bir kere** kullanılır.
- **Neden Önemli?:** Bu disiplin, modelinizin başarısını abartmanızı engeller ve canlıya çıktığında ne gibi bir performans bekleyeceğinize dair size **güvenilir ve dürüst** bir tahmin sunar. Bu, profesyonel makine öğrenmesi projelerinin "altın standardıdır".

Şemayı Parçalara Ayıralım

1. Bilgisayar Simgesi (Tüm Veri Seti):

- Bu, projenin başlangıç noktasıdır. Elinizdeki tüm veriyi (müşteri listesi, satış kayıtları, sensör verileri vb.) temsil eder.

2. Üç Ana Dala Ayrılma:

- İşte en kritik adım burada atılıyor. Veri setimizi, az önce konuştuğumuz gibi, 3 farklı amaca hizmet eden parçaya ayırıyoruz:
    - **Train (Eğitim Seti)**
    - **Validation (Doğrulama Seti)**
    - **Test (Test Seti)**

Sürecin Adım Adım İşleyişi

Adım 1: Eğitim (Train)

- **Ne Yapılır?:** M1, M2, M3, M4 gibi birden çok model adayını veya aynı modelin farklı ayarlarını (hiperparametrelerini) bu veri seti üzerinde eğitiriz.
- **Amaç:** Modellerin, verideki desenleri ve kuralları öğrenmesini sağlamak.
- **Analoji:** Dört farklı öğrenciye (veya dört farklı öğretim tekniğiyle dört öğrenciye) **"Hazırlık Kitabını"** verip çalışmalarını istemek.

Adım 2: Doğrulama ve Seçim (Validation)

- **Ne Yapılır?:** Eğitimde öğrendiklerini test etmek için tüm model adaylarımızı **Validation Seti** üzerinde çalıştırırız ve performanslarını (hata oranları, doğrulukları vb.) karşılaştırırız.
- Görselde, validation bölümünün altında 4 tane çizgi var, bunlar 4 modelin skorlarını temsil ediyor. En alttaki skora bir **"tik" (✓)** işareti atılmış. Bu, o modelin en iyi performansı gösterdiği anlamına geliyor. Diyelim ki bu M3 modeli ve hata skoru %20.
- **Amaç:** Modeller arasında bir **yarışma düzenleyerek en iyi modeli (şampiyonu) seçmek.**
- **Analoji:** Tüm öğrencileri, daha önce görmedikleri bir **"Deneme Sınavı'na (Quiz)"** sokmak ve en yüksek notu alan öğrenciyi okulun final sınavı temsilcisi olarak seçmek.

Adım 3: Ayarlama (Tuning - Görseldeki Döngü)

- Görseldeki validation bölümünden train bölümüne geri dönen ince okları fark ettiniz mi? Bu çok önemli bir detayı temsil eder.
- **Ne Yapılır?:** Deneme sınavı (validation) sonuçlarına bakarız. Belki de en iyi modelimiz olan M3'ün bile performansı yeterince iyi değildir. O zaman geri döner, M3'ün ayarlarını biraz değiştirir (daha fazla çalıştırır, farklı bir öğrenme tekniği ekleriz), onu tekrar **eğitir (train)** ve sonra tekrar **doğrularız (validate)**. Bu döngü, validation setinde tatmin edici bir sonuca ulaşana kadar devam edebilir.
- **Amaç:** Şampiyon modelimizi daha da iyileştirmek, ince ayar yapmak.

Adım 4: Final Testi (Test)

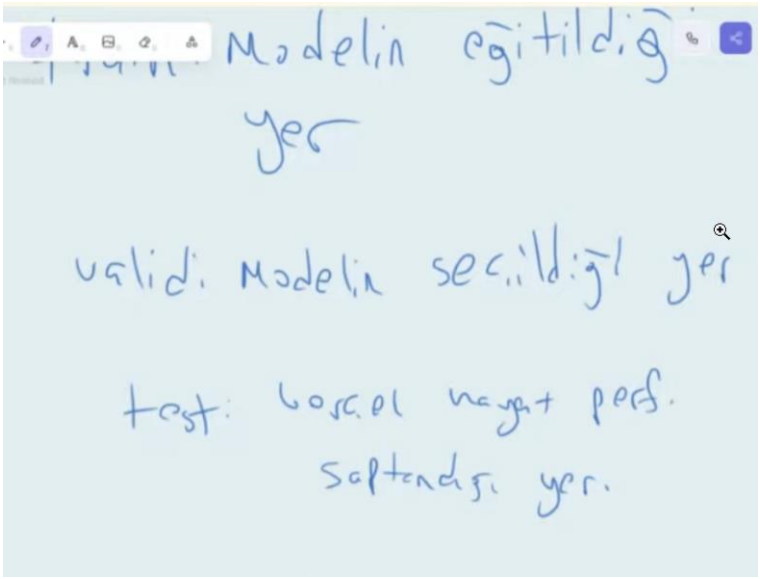
- **Ne Yapılır?:** Artık şampiyon modelimizi seçtik ve son ayarlarını yaptık. Diğer tüm modelleri (M1, M2, M4) çöpe atarız. Sadece ve sadece şampiyon olan modeli alır ve daha önce **HİÇ dokunulmamış, kilitli kasadaki Test Seti** üzerinde **SADECE BİR KERE** çalıştırırız.
- Görselde, validation setindeki en iyi modelden (✓ olan) test setine giden tek bir ok var. Bu, sadece şampiyonun finale çıktığını gösteriyor. Diğer modellerden test setine giden okların ise üzeri çizilmiş. Bu da "Diğer modellerin test verisini görmesi KESİNLİKLE YASAK!" anlamına geliyor.
- Modelin bu testten aldığı skor (görselde %60 yazıyor, bu muhtemelen başarı oranı) onun **nihai ve dürüst performans notudur**.
- **Amaç:** Modelimizin gerçek dünyada, daha önce hiç görmediği verilerle karşılaştığında nasıl bir performans sergileyeceğini **tarafsız bir şekilde ölçmek.**
- **Analoji:** Okul temsilcisi olarak seçtiğimiz en iyi öğrenciyi, il genelinde yapılan ve sorularını kimsenin önceden bilmediği **"Gerçek Final Sınavı'na"** sokmak.

Özetle, bu şema bize şunu söylüyor:

"Farklı modelleri **Train** setinde eğit, en iyisini **Validation** setinde yarıştırarak seç, seçtiğin en iyi modelin gerçek gücünü ise el değmemiş **Test** setinde sadece bir kere ölç."

13

Train Modelin eğitildiği yer.



Bu görsel, önceki iki görselde anlatılan iş akışının en net ve en sade özetini sunuyor. "altın kurallar"

Görsel 12: Üç Setin Altın Kuralları - Görev Tanımları

Genel Anlam: Neyi Anlatıyor?

Bu görsel, Train, Validation ve Test setlerinin her birinin projedeki **eşsiz ve devredilemez görevini** tek bir cümleyle özetler. Bu üç görevi doğru anlamak ve uygulamak, başarılı bir makine öğrenmesi projesinin temelini oluşturur.

---

1. Train: Modelin Eğitildiği Yer

- **Görevi:** Modelin, veri içindeki desenleri, ilişkileri ve kuralları öğrendiği, "ders çalıştığı" ana alandır. Modelin beynini oluşturan parametreler bu veri üzerinde ayarlanır.
- **Python Dünyası:** `model.fit(X_train, y_train)` komutunun çalıştığı yerdir. Model, `X_train` ve `y_train` arasındaki ilişkiyi öğrenir.
- **Bisküvi Analojisi:** Burası **Eğitim Atölyesi**'dir. Prototip makineler, binlerce "sağlam" ve "bozuk" bisküvi örneğini görerek kalibre edilir.
- **Restoran Analojisi:** Burası **Mutfak Akademisi**'dir. Şef adayları, "Usta Şefin Tarif Defteri"ndeki tarifleri tekrar tekrar yaparak yemek pişirmeyi öğrenir.

---

2. Validation: Modelin Seçildiği Yer

- **Görevi:** Farklı algoritmaları veya aynı algoritmanın farklı ayarlarını (hiperparametreler) birbiriyle yarıştırarak, aralarından en iyi performansı gösteren "şampiyon modeli" **seçtiğimiz** yerdir.
- **Python Dünyası:** Farklı modelleri (model_A, model_B) X_val üzerinde model.predict() veya model.score() ile çalıştırıp, en iyi skoru vereni belirlediğimiz yerdir. GridSearchCV veya RandomizedSearchCV gibi araçlar bu süreci otomatikleştirir.
- **Bisküvi Analojisi:** Burası **Şampiyonluk Müsabakası** alanıdır. Eğitilmiş tüm makineler, daha önce görmedikleri yeni bisküviler üzerinde test edilir ve en isabetli olanı "şampiyon" ilan edilir.
- **Restoran Analojisi:** Burası **Baş Aşçı Seçmeleri**'dir. Tüm şef adayları, jüri (patron, gurmeler) önünde özel bir menü hazırlar ve en lezzetli yemeği yapan "Baş Aşçı" olarak seçilir.

---

3. Test: Gerçek Hayat Performansının Saptandığı Yer

- **Görevi:** Projenin en sonunda, seçilmiş olan tek bir şampiyon modelin, daha önce hiç görmediği, tamamen izole edilmiş veri üzerinde **gerçek dünya performansını dürüstçe ölçtüğümüz** yerdir. Buradan çıkan sonuç, modelin raporlanacak nihai başarı skorudur.
- **Python Dünyası:** Şampiyon modeli, X_test üzerinde model.predict() veya model.score() ile **sadece bir kez** çalıştırılır. Çıkan accuracy, F1-score gibi metrikler projenin resmi sonuçlarıdır.
- **Bisküvi Analojisi:** Burası **Resmi Devlet Denetimi**'dir. Şampiyon makine, en başından beri kilitli bir odada bekletilen el değmemiş bisküviler üzerinde son bir teste tabi tutulur ve üzerine "Kalite Kontrol Başarısı: %98.5" etiketi yapıştırılır.
- **Restoran Analojisi:** Burası restoranın **Büyük Açılış Gecesi**'dir. Seçilen Baş Aşçı, ilk defa gerçek müşterilere yemek yapar ve onların tepkisi (geri bildirimler, sosyal medya yorumları) restoranın gerçek başarısını belirler.

Özet Tablo

Bu üç görevi asla birbirine karıştırmamak gerekir:

Veri Seti	Temel Görev	Anahtar Kelime	Analoji
Train	**Öğrenme**	model.fit()	Ders Çalışma
Validation	**Seçme / Ayarlama**	Model Karşılaştırma	Deneme Sınavı
Test	**Raporlama / Ölçme**	Nihai Performans	Final Sınavı

1. Train: Modelin Eğitildiği Yer

- **Anlamı:** Bu, modelin "ders çalıştığı" yerdir. Verinin içindeki desenleri, kuralları ve ilişkileri burada öğrenir.
- **Öğrenci Analojisi:** Burası **"Sınıf"** veya **"Kütüphane"**'dir. Öğrenci, cevap anahtarıyla birlikte verilen **"Hazırlık Kitabına"** burada çalışır. Tüm öğrenme süreci bu veri üzerinde gerçekleşir.

2. Validation: Modelin Seçildiği Yer

- **Anlamı:** Burası, farklı yöntemlerle eğitilmiş modeller arasında bir yarışma yaptığımız ve şampiyonu seçtiğimiz yerdir.
- **Öğrenci Analojisi:** Burası **"Deneme Sınavı (Quiz)"** salonudur. Farklı yöntemlerle çalışmış tüm öğrencileri, daha önce görmedikleri bu deneme sınavına sokarız. En yüksek notu alan öğrenciyi, "okulun final sınavı temsilcisi" olarak **seçeriz.** Ayrıca bu sınavın sonuçlarına bakarak en iyi öğrencimizin eksiklerini giderip ona ince ayar yapabiliriz.

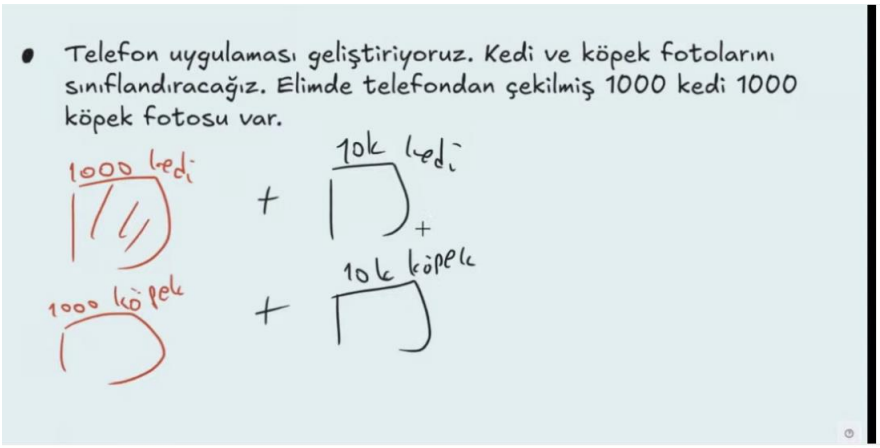
3. Test: Gerçek Hayat Performansının Saptandığı Yer

- **Anlamı:** Burası, seçtiğimiz en iyi modelin, canlıya çıktığında veya gerçek dünyada karşılaştığı veriler karşısında nasıl bir performans sergileyeceğini **tarafsız bir şekilde ölçtüğümüz** yerdir.
- **Öğrenci Analojisi:** Burası **"Resmi Final Sınavı"** salonudur. Sorular tamamen yenidir ve kimse daha önce görmemiştir. Deneme sınavında seçtiğimiz şampiyon öğrencimizi bu sınava sokarız. Buradan aldığı not, onun karnesine yazılacak olan **nihai ve resmi başarı notudur.** Bu not, öğrencinin gerçek bilgi seviyesini gösterir.

Sunum İçin Özet Tablo

Veri Seti	**Görseldeki Tanım**	**Öğrenci Analojisi**	**Temel Amaç**
Train	Modelin eğitildiği yer	**Hazırlık Kitabı** (Cevaplarıyla)	**ÖĞRENME**
Validation	Modelin seçildiği yer	**Deneme Sınavı / Quiz**	**KARŞILAŞTIRMA & SEÇİM**
Test	Gerçek hayat performansının saptandığı yer	**Resmi Final Sınavı**	**TARAFSIZ ÖLÇÜM**

14



Bu görsel, makine öğrenmesi projelerinin en kritik ve genellikle en çok zaman alan kısmıyla ilgili çok önemli bir noktaya değiniyor: **Verinin Kalitesi ve Miktarı.**.

Görsel 13: Veri Miktarı ve Çeşitliliği - "Daha Çok Veri, Daha Akıllı Model"

Genel Anlam: Neyi Anlatıyor?

Bu görsel, bir makine öğrenmesi modelinin **genelleme yeteneğini** artırmak için veri setini nicelik ve nitelik olarak büyütmenin önemini vurgular. Genellikle, daha iyi bir algoritma bulmaya çalışmaktan daha etkili olan yol, modele daha fazla ve daha çeşitli veri sunmaktır.

- **Sol Taraf (Başlangıç):** 1000 kedi + 1000 köpek. Bu, sizin kendi topladığınız, kaliteli ama sınırlı çeşitliliğe sahip "çekirdek" veri setinizdir.
- **Sağ Taraf (Geliştirme):** + 10k kedi + 10k köpek. Bu, modelin "dünya tecrübesini" artırmak için dış kaynaklardan (internetten, halka açık veri setlerinden) eklenen devasa bir veri setidir.

Python Dünyası ile Anlatım

Senaryomuzu Python'da canlandıralım.

1. Hatalı Yaklaşım: Sadece Kendi Sınırlı Verinizle Eğitmek

Eğer modelinizi sadece kendi çektiğiniz 2000 fotoğrafla eğitirseniz, **Veri Seti Yanlılığı (Dataset Bias)** tuzağına düşersiniz.

```
# Kendi çektiğiniz fotoğrafların bilgileri
# Cinsler: 1: Siyam Kedisi, 2: Golden Retriever
# Arka Plan: 1: Ev İçi, 2: Bahçe
kendi_verim = pd.DataFrame({
    'cins': [1, 1, 1, 2, 2, 2],
    'arka_plan': [1, 1, 1, 2, 2, 2],
    # ... resim verileri ...
})
```

Model, sadece bu sınırlı veriyle eğitilir.

model.fit(kendi_verim_X, kendi_verim_y)

Bu model ne öğrenir? "Kedi, ev içinde olan Siyam'dır", "Köpek, bahçede olan Golden'dır". Karşısına sokakta çekilmiş bir Van kedisi fotoğrafı geldiğinde ne yapacağını bilemez. **Genelleme yapamaz.**

2. Doğru Yaklaşım: Veri Setini Çeşitlendirmek

Profesyonel yaklaşım, kendi kaliteli verinizi, internetten bulduğunuz devasa ve çeşitli bir veri setiyle birleştirmektir.

```
# İnternetten indirilen 20.000 fotoğraflık çeşitli veri
# İçinde yüzlerce farklı cins, arka plan, ışık koşulu var.
internet_verisi = pd.read_csv('kaggle_cat_dog_dataset.csv')

# İki DataFrame'i birleştirelim
toplam_veri = pd.concat([kendi_verim, internet_verisi], ignore_index=True)

print(f"Eski veri seti boyutu: {len(kendi_verim)}")
print(f"Yeni, birleştirilmiş veri seti boyutu: {len(toplam_veri)}")

# Model, artık bu devasa ve çeşitli veriyle eğitilir.
# model.fit(toplam_veri_X, toplam_veri_y)
```

Bu yeni model, bir kedinin veya köpeğin yüzlerce farklı durumda nasıl görünebileceğini öğrenir. Karşısına ne tür bir kedi/köpek fotoğrafı çıkarsa çıksın, doğru tahmin yapma olasılığı çok daha yüksektir. **Genelleme yeteneği artmıştır.**

Analojilerle Anlatım

1. Bisküvi Üretimi Analojisi

- **Küçük Set (1000+1000):** Kalite kontrol makinenizi, **sadece kendi fabrikanızın A vardiyasında, mükemmel ışık altında** çekilmiş 2000 fotoğrafla eğitiyorsunuz.
- **Problem:** Bu makine, B vardiyasındaki loş ışıkta veya başka bir şehirdeki fabrikanın farklı arka planında çalışamaz. Sadece tek bir durumu "ezberlemiştir".
- **Büyük Set (+10k+10k):** Modele, şirketin **tüm fabrikalarından, tüm vardiyalardan, farklı ışık koşulları ve kamera açılarından** çekilmiş 20.000 fotoğraf daha ekliyorsunuz.
- **Sonuç:** Makine artık bir bisküvinin neye benzediğini evrensel olarak öğrenir ve her koşulda doğru çalışır. **Çeşitlilik, genellemeyi sağlar.**

2. Restoran Analojisi

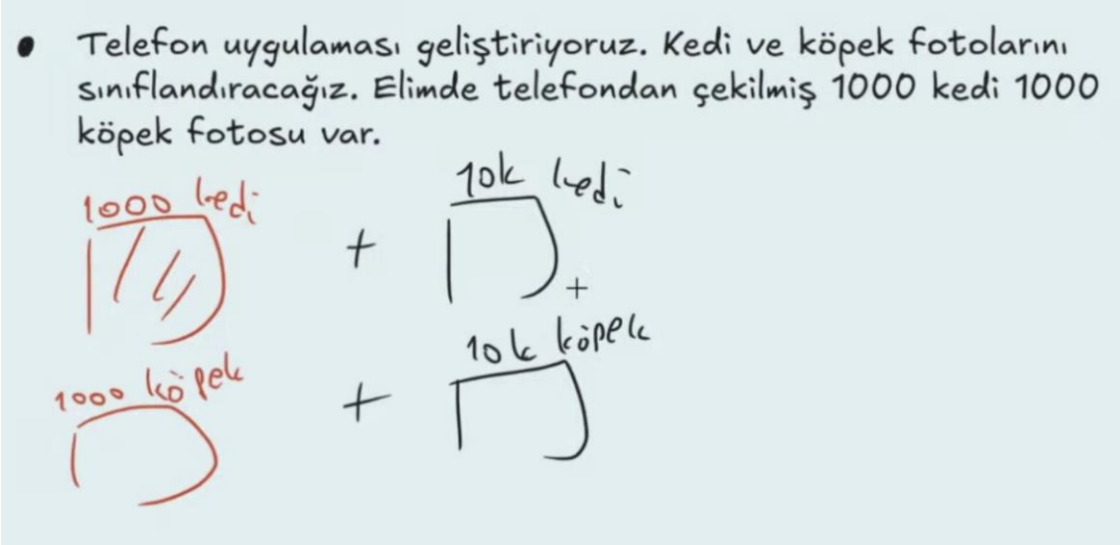
- **Küçük Set (1000+1000):** Genç bir şef, yemek yapmayı **sadece annesinin tarif defterindeki** 20 tarifle öğreniyor. Bu tarifler lezzetlidir ama çeşitliliği azdır.
- **Problem:** Şef, bir İtalyan restoranında işe başladığında, annesinin defterinde olmayan yemekleri yapamaz. Bilgisi **yanlı (biased)** ve sınırlıdır.

- **Büyük Set (+10k+10k):** Şef, dünyanın dört bir yanından **yüzlerce farklı yemek kitabı** okur ve binlerce farklı tarifi dener.
- **Sonuç:** Şefin "dünya tecrübesi" artar. Artık sadece annesinin yemeklerini değil, Fransız, İtalyan, Meksika mutfağından da yemekler yapabilir. **Daha fazla ve çeşitli veri (tarif), daha yetenekli bir model (şef) yaratır.**

Özet

- **Anahtar Kelime:** Veri Miktarı, Veri Çeşitliliği, Veri Artırma (Data Augmentation), Genelleme, Veri Seti Yanlılığı (Dataset Bias).
- **Kural:** Bir makine öğrenmesi modelinin zekası ve genelleme yeteneği, ona sunulan verinin miktarı ve çeşitliliği ile doğru orantılıdır. Çoğu zaman, daha karmaşık bir algoritma denemek yerine, veri setinizi daha çeşitli örneklerle zenginleştirmek çok daha iyi sonuçlar verir.

15



Elbette! Bu görsel, makine öğrenmesi projelerinin en kritik ve genellikle en çok zaman alan kısmıyla ilgili çok önemli bir noktaya değiniyor: **Verinin Kalitesi ve Miktarı.**

Görsel 13: Veri Miktarı ve Çeşitliliği - "Daha Çok Veri, Daha Akıllı Model"

Genel Anlam: Neyi Anlatıyor?

Bu görsel, bir makine öğrenmesi modelinin **genelleme yeteneğini** artırmak için veri setini nicelik ve nitelik olarak büyütmenin önemini vurgular. Genellikle, daha iyi bir algoritma bulmaya çalışmaktan daha etkili olan yol, modele daha fazla ve daha çeşitli veri sunmaktır.

- **Sol Taraf (Başlangıç):** 1000 kedi + 1000 köpek. Bu, sizin kendi topladığınız, kaliteli ama sınırlı çeşitliliğe sahip "çekirdek" veri setinizdir.
- **Sağ Taraf (Geliştirme):** + 10k kedi + 10k köpek. Bu, modelin "dünya tecrübesini" artırmak için dış kaynaklardan (internetten, halka açık veri setlerinden) eklenen devasa bir veri setidir.

Python Dünyası ile Anlatım

Senaryomuzu Python'da canlandıralım.

1. Hatalı Yaklaşım: Sadece Kendi Sınırlı Verinizle Eğitmek

Eğer modelinizi sadece kendi çektiğiniz 2000 fotoğrafla eğitirseniz, **Veri Seti Yanlılığı (Dataset Bias)** tuzağına düşersiniz.

```
# Kendi çektiğiniz fotoğrafların bilgileri
# Cinsler: 1: Siyam Kedisi, 2: Golden Retriever
# Arka Plan: 1: Ev İçi, 2: Bahçe
kendi_verim = pd.DataFrame({
    'cins': [1, 1, 1, 2, 2, 2],
    'arka_plan': [1, 1, 1, 2, 2, 2],
    # ... resim verileri ...
})

# Model, sadece bu sınırlı veriyle eğitilir.
# model.fit(kendi_verim_X, kendi_verim_y)
```

Bu model ne öğrenir? "Kedi, ev içinde olan Siyam'dır", "Köpek, bahçede olan Golden'dır". Karşısına sokakta çekilmiş bir Van kedisi fotoğrafı geldiğinde ne yapacağını bilemez. **Genelleme yapamaz.**

2. Doğru Yaklaşım: Veri Setini Çeşitlendirmek

Profesyonel yaklaşım, kendi kaliteli verinizi, internetten bulduğunuz devasa ve çeşitli bir veri setiyle birleştirmektir.

```
 # İnternetten indirilen 20.000 fotoğraflık çeşitli veri
# İçinde yüzlerce farklı cins, arka plan, ışık koşulu var.
internet_verisi = pd.read_csv('kaggle_cat_dog_dataset.csv')

# İki DataFrame'i birleştirelim
toplam_veri = pd.concat([kendi_verim, internet_verisi], ignore_index=True)

print(f"Eski veri seti boyutu: {len(kendi_verim)}")
print(f"Yeni, birleştirilmiş veri seti boyutu: {len(toplam_veri)}")

# Model, artık bu devasa ve çeşitli veriyle eğitilir.
# model.fit(toplam_veri_X, toplam_veri_y)
```

Bu yeni model, bir kedinin veya köpeğin yüzlerce farklı durumda nasıl görünebileceğini öğrenir. Karşısına ne tür bir kedi/köpek fotoğrafı çıkarsa çıksın, doğru tahmin yapma olasılığı çok daha yüksektir. **Genelleme yeteneği artmıştır.**

Analojilerle Anlatım

1. Bisküvi Üretimi Analojisi

- **Küçük Set (1000+1000):** Kalite kontrol makinenizi, **sadece kendi fabrikanızın A vardiyasında, mükemmel ışık altında** çekilmiş 2000 fotoğrafla eğitiyorsunuz.

- **Problem:** Bu makine, B vardiyasındaki loş ışıkta veya başka bir şehirdeki fabrikanın farklı arka planında çalışamaz. Sadece tek bir durumu "ezberlemiştir".
- **Büyük Set (+10k+10k):** Modele, şirketin **tüm fabrikalarından, tüm vardiyalardan, farklı ışık koşulları ve kamera açılarından** çekilmiş 20.000 fotoğraf daha ekliyorsunuz.
- **Sonuç:** Makine artık bir bisküvinin neye benzediğini evrensel olarak öğrenir ve her koşulda doğru çalışır. **Çeşitlilik, genellemeyi sağlar.**

2. Restoran Analojisi

- **Küçük Set (1000+1000):** Genç bir şef, yemek yapmayı **sadece annesinin tarif defterindeki** 20 tarifle öğreniyor. Bu tarifler lezzetlidir ama çeşitliliği azdır.
- **Problem:** Şef, bir İtalyan restoranında işe başladığında, annesinin defterinde olmayan yemekleri yapamaz. Bilgisi **yanlı (biased)** ve sınırlıdır.
- **Büyük Set (+10k+10k):** Şef, dünyanın dört bir yanından **yüzlerce farklı yemek kitabı** okur ve binlerce farklı tarifi dener.
- **Sonuç:** Şefin "dünya tecrübesi" artar. Artık sadece annesinin yemeklerini değil, Fransız, İtalyan, Meksika mutfağından da yemekler yapabilir. **Daha fazla ve çeşitli veri (tarif), daha yetenekli bir model (şef) yaratır.**

Özet

- **Anahtar Kelime:** Veri Miktarı, Veri Çeşitliliği, Veri Artırma (Data Augmentation), Genelleme, Veri Seti Yanlılığı (Dataset Bias).
- **Kural:** Bir makine öğrenmesi modelinin zekası ve genelleme yeteneği, ona sunulan verinin miktarı ve çeşitliliği ile doğru orantılıdır. Çoğu zaman, daha karmaşık bir algoritma denemek yerine, veri setinizi daha çeşitli örneklerle zenginleştirmek çok daha iyi sonuçlar verir.

Görsel Ne Anlatıyor: Veri, Veri, Daha Çok Veri!

Senaryo: Kedi ve köpek fotoğraflarını ayırt eden bir mobil uygulama yapıyoruz.

1. Sol Taraf (Kırmızı Çizimler): İdeal Başlangıç Senaryosu

- **Ne Görüyoruz?:** 1000 kedi ve 1000 köpek fotoğrafından oluşan, kendi topladığımız bir veri seti.
- **Anlamı:** Bu, projemizin başlangıç noktası. Kendi telefonumuzla veya arkadaşlarımızın telefonlarıyla çektiğimiz, etiketlemesi düzgün yapılmış, temiz bir veri setimiz var. Bu harika bir başlangıç!
- **İlk Düşünce:** "Süper! 2000 fotoğrafla modelimi eğitmeye başlayabilirim."

---

2. Asıl Soru: Bu 2000 Fotoğraf Yeterli mi? (Gizli Problem)

İlk bakışta yeterli gibi görünse de, burada çok tehlikeli bir tuzak var. Bu 2000 fotoğrafın **"kimliği"** çok önemli. Düşünelim:

- **Bu fotoğrafları kim çekti?** Ben çektim.
- **Nerede çektim?** Kendi evimde ve bahçemde.
- **Hangi hayvanları çektim?** Kendi kedim (bir Siyam kedisi) ve komşumun köpeğini (bir Golden Retriever).

Peki, sadece bu fotoğraflarla eğitilen bir model ne öğrenir?

- Model, **tüm kedilerin** Siyam kedisi gibi göründüğünü zanneder. Karşısına bir Van kedisi veya Sarman çıktığında ne yapacağını bilemez, muhtemelen "köpek" der.
- Model, "kedi" kelimesini "evdeki koltuğun üzerindeki tüylü canlı" olarak, "köpek" kelimesini ise "bahçedeki çimlerin üzerindeki canlı" olarak kodlayabilir. Yani hayvanın kendisini değil, **arka planı öğrenir.**

Bu duruma **Veri Seti Yanlılığı (Dataset Bias)** diyoruz. Modelimiz, genelleme yapamaz, sadece bizim sınırlı dünyamızı ezberler.

---

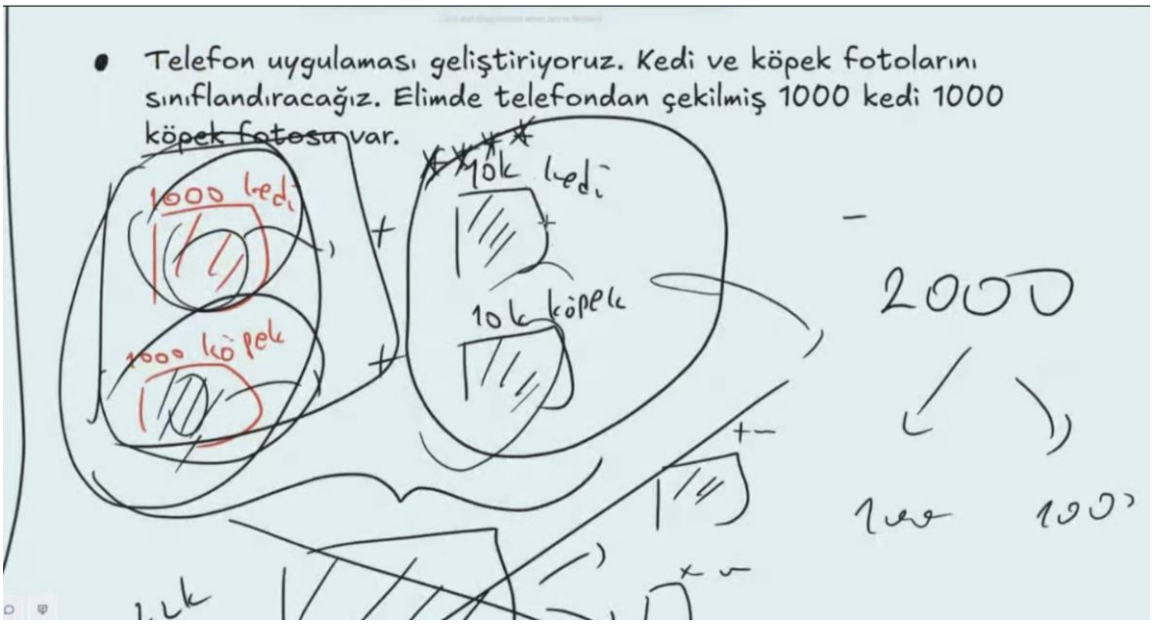
3. Sağ Taraf (Siyah Çizimler): Profesyonel Çözüm

- **Ne Görüyoruz?:** Mevcut setimize + 10k kedi ve + 10k köpek fotoğrafı ekleniyor. Yani 10.000 kedi ve 10.000 köpek fotoğrafı daha!
- **Anlamı:** İşte bu, projemizi "hobi projesi" olmaktan çıkarıp "profesyonel bir uygulama" haline getiren adımdır: **Veri Artırma (Data Augmentation).**
- **Bu Yeni 20.000 Fotoğraf Nereden Geliyor?** Genellikle internetteki büyük, halka açık veri setlerinden (ImageNet, Kaggle verileri vb.) gelir.
- **Neden Bunu Yapıyoruz?** Tek kelimeyle: **ÇEŞİTLİLİK (DIVERSITY).**
    - Bu yeni 20.000 fotoğraf, binlerce farklı insan tarafından, binlerce farklı telefonla çekilmiştir.
    - İçinde yüzlerce farklı kedi ve köpek cinsi vardır.
    - Farklı ışık koşullarında (gece, gündüz), farklı açılardan, farklı arka planlarda (plaj, orman, sokak, ev...) çekilmişlerdir.

Özet ve Sonuç

Özellik	**Küçük, Kişisel Set (2.000 Foto)**	**Büyük, Çeşitli Set (22.000 Foto)**
Avantajı	Temiz, güvenilir etiketler.	Muazzam çeşitlilik.
En Büyük Riski	**Yanlılık (Bias).** Model sadece sizin dünyanızı öğrenir.	Etiketlerde hatalar olabilir (temizlik gerektirir).
Genelleme Yeteneği	Çok Düşük.	Çok Yüksek.

16



Bu görsel, önceki konuları bir araya getirerek, gerçek bir makine öğrenmesi projesinde nasıl **stratejik bir veri bölme planı** oluşturulması gerektiğini anlatıyor. Bu, projenizin başarısı için hayati bir adımdır.

Görsel 14: Stratejik Veri Bölme - "Doğru Veri, Doğru Yere"

Genel Anlam: Neyi Anlatıyor?

Bu görsel, elimizdeki farklı nitelikteki veri kaynaklarını (kendi topladığımız "gerçek dünya" verisi ve internetten indirdiğimiz "genel" veri) Train, Validation ve Test setlerine nasıl akıllıca dağıtmamız gerektiğini gösteren bir strateji haritasıdır. Amaç, modeli **en çeşitli veriyle eğitmek** ve **en gerçekçi veriyle test etmektir.**

- **Sol Taraf (Kırmızı):** Sizin tarafınızdan, telefondan çekilmiş **1000 kedi + 1000 köpek** fotoğrafı. Bu, uygulamanızın canlıda karşılaşacağı **"gerçek dünya senaryosunu"** temsil eden en değerli veridir.
- **Orta Kısım (Siyah):** İnternetten indirilen, çeşitliliği yüksek ama kalitesi ve kaynağı belirsiz **10k kedi + 10k köpek** fotoğrafı. Bu, modelin "dünya tecrübesini" artırmak için harika bir kaynaktır.

Python Dünyası ile Anlatım (Stratejik Bölme)

Bu görseldeki doğru stratejiyi Python koduyla canlandıralım:

```
import pandas as pd
from sklearn.model_selection import train_test_split

# 1. Veri Kaynaklarını Yükleme
kendi_verim_df = pd.read_csv('telefondan_cekilmis_2000.csv')
internet_verisi_df = pd.read_csv('internetten_indirilen_20000.csv')

# ---- STRATEJİK BÖLME ----

# 2. Adım: GERÇEK DÜNYA verisini Validation ve Test için ayır.
# Kendi çektiğimiz 2000 fotoğrafı Validation ve Test için ikiye bölelim.
# Bu setlere internet verisi KESİNLİKLE KARIŞTIRILMAZ.
val_df, test_df = train_test_split(kendi_verim_df, test_size=0.5, random_state=42)

# 3. Adım: TRAIN setini oluştur.
# Modelin öğrenmesi için olabildiğince çeşitli bir set oluşturalım.
# İnternet verisinin tamamını Train setine ekleyelim.
train_df = pd.concat([internet_verisi_df], ignore_index=True)

# İsteğe bağlı: Kendi verimizin bir kısmını da eğitime ekleyebiliriz,
# ama en önemlisi val/test setlerinin temiz kalmasıdır.

# X_train, y_train = train_df[['ozellikler']], train_df['etiket']
# X_val, y_val = val_df[['ozellikler']], val_df['etiket']
# X_test, y_test = test_df[['ozellikler']], test_df['etiket']

print(f"Eğitim Seti Boyutu (Çeşitli): {len(train_df)}")
print(f"Doğrulama Seti Boyutu (Gerçekçi): {len(val_df)}")
print(f"Test Seti Boyutu (Gerçekçi ve El Değmemiş): {len(test_df)}")
```

Analojilerle Anlatım

1. Sınava Hazırlanan Öğrenci Analojisi

- **Problem:** Uluslararası bir matematik olimpiyatına öğrenci hazırlıyoruz.
- **"Gerçek Dünya" Verisi (1000+1000):** Geçmiş yıllarda **olimpiyatlarda gerçekten çıkmış** 2000 soru. Bunlar en değerli kaynağımız.
- **"Genel" Veri (10k+10k):** Piyasada bulunan **binlerce farklı hazırlık kitabındaki** 20.000 soru.
- **Strateji:**
    - **Train Seti:** Öğrenciyi, piyasadaki binlerce farklı soruyla (internet_verisi) eğiterek her türlü soru tipine hazır hale getiririz.
    - **Validation ve Test Setleri:** Öğrencinin gerçek performansını ölçmek için **sadece ve sadece geçmiş olimpiyat sorularını** (kendi_verim) kullanırız. Çünkü asıl hedefimiz, olimpiyat formatındaki soruları çözmektir.

2. Bisküvi Üretimi Analojisi

- **Problem:** Yeni bir kalite kontrol makinesi geliştiriyoruz.
- **"Gerçek Dünya" Verisi (1000+1000): Kendi fabrikamızın üretim hattından** çıkan 2000 bisküvinin fotoğrafı.
- **"Genel" Veri (10k+10k):** İnternette bulduğumuz, **farklı markalara ait, farklı ülkelerde üretilmiş** 20.000 bisküvi fotoğrafı.
- **Strateji:**
    - **Train Seti:** Makineyi, dünyadaki her türlü bisküvi şeklini, rengini ve dokusunu öğrenmesi için internetteki 20.000 fotoğrafla eğitiriz.
    - **Validation ve Test Setleri:** Makinenin asıl görevini (kendi fabrikamızdaki bisküvileri denetlemek) ne kadar iyi yaptığını ölçmek için **sadece kendi fabrikamızdan çıkan bisküvileri** kullanırız.

3. Restoran Analojisi

- **Problem:** Bir şefi, restoranımızın baş aşçısı olmak üzere eğitiyoruz.
- **"Gerçek Dünya" Verisi (1000+1000):** Restoranımızın **menüsünde yer alan** 20 farklı yemeğin tarifi.
- **"Genel" Veri (10k+10k):** Dünyanın dört bir yanından toplanmış **binlerce farklı yemek tarifi.**
- **Strateji:**
    - **Train Seti:** Şefin ufkunu genişletmek ve farklı teknikleri öğrenmesini sağlamak için ona binlerce farklı dünya mutfağı tarifi (internet_verisi) üzerinde pratik yaptırırız.
    - **Validation ve Test Setleri:** Şefin, restoranımızın konseptine ve menüsüne ne kadar hakim olduğunu ölçmek için yapılan "Baş Aşçı Seçmesi" ve "Açılış Gecesi"nde **sadece bizim menümüzdeki yemekleri** yapmasını isteriz.

Özet

- **Anahtar Kelime:** Stratejik Veri Bölme, Veri Dağılımı Eşleştirme, Train-Test Uyumsuzluğu (Mismatch).
- **Kural:** Modelini olabildiğince **çeşitli ve büyük** bir veriyle **eğit**, ama başarısını **her zaman** gerçek dünyada karşılaşacağı senaryoyu **birebir yansıtan** veriyle **test et.**

- **Neden Önemli?:** Bu strateji, modelinizin laboratuvarda değil, **sahada** (yani uygulamanızın canlıda) başarılı olmasını sağlar. Size, modelinizin gerçek kullanıcı verileriyle karşılaştığında nasıl bir performans sergileyeceğine dair dürüst ve güvenilir bir tahmin verir.

Görsel Ne Anlatıyor: "Hangi Veri, Nereye Gitmeli?"

Bu görsel, önceki konseptlerin (Train/Validation/Test ayrımı ve Veri Artırma) birleştiği ve stratejik bir kararın verildiği noktayı gösteriyor.

Senaryoyu Hatırlayalım:

- Elimizde **1000 kedi + 1000 köpek** fotoğrafı var. Bunlar bizim kendi telefonlarımızla çektiğimiz, **"gerçek dünya"** verilerimiz.
- Ayrıca modelimizi daha çeşitli hale getirmek için internetten **10k kedi + 10k köpek** fotoğrafı bulduk. Bunlar ise **"genel"** veriler.

Şimdi bu toplam 22.000 fotoğrafı Train, Validation ve Test setlerine nasıl dağıtacağız?

Yaygın Hata (Görselde Üzeri Çizilen Fikir)

Acemi bir veri bilimci şöyle düşünebilir: "Elimde 22.000 fotoğraf var. Bunların hepsini karıştırır, sonra rastgele %70 Train, %15 Validation, %15 Test olarak ayırırım. Adil olan budur."

Bu neden KORKUNÇ bir hatadır?

Çünkü bu durumda, sizin telefonunuzla çektiğiniz gerçek fotoğrafların bir kısmı, internetten indirilen binlerce fotoğraf arasında kaybolarak **Train Seti'ne** gidebilir. Aynı şekilde, internetten indirilen fotoğrafların bir kısmı da **Test Seti'ne** gidebilir.

Sonuç ne olur? Modeliniz, internetten indirilen bir fotoğrafı doğru tahmin edip, sizin telefonunuzla çektiğiniz bir fotoğrafı yanlış tahmin ettiğinde bile genel başarı skorunuz yüksek çıkabilir. Bu, size **yanıltıcı bir başarı hissi** verir. Oysa sizin uygulamanızın asıl amacı, internetteki fotoğrafları değil, **kullanıcıların kendi telefonlarıyla çekeceği fotoğrafları** doğru sınıflandırmaktır.

Doğru ve Profesyonel Strateji (Görselin Anlattığı Asıl Hikaye)

Bu görsel, tam olarak bu hatayı yapmamamız gerektiğini söylüyor. İşte anlattığı doğru strateji:

Adım 1: Train Setini Oluşturmak (Öğrenme Aşaması)

- **Ne Yapılır?:** İnternetten indirdiğimiz **10k kedi + 10k köpek** fotoğrafının **TAMAMINI** alırız. Hatta istersek kendi çektiğimiz 2000 fotoğrafın bir kısmını da buraya ekleyebiliriz. Bu devasa set, bizim **Train Setimiz** olur.
- **Neden?:** Çünkü modelin öğrenme aşamasında olabildiğince **fazla çeşitlilik** görmesini istiyoruz. Farklı cinsler, farklı ışıklar, farklı arka planlar... Modelin "hayat tecrübesi" ne kadar zengin olursa, o kadar iyi öğrenir.

Adım 2: Validation ve Test Setlerini Oluşturmak (Gerçek Sınav Aşaması)

- **Ne Yapılır?:** Sadece ve sadece **kendi telefonlarımızla çektiğimiz 1000 kedi + 1000 köpek** fotoğrafını kullanırız.

- **Neden?:** Çünkü **Validation ve Test setleri, modelin gerçek dünyada karşılaşacağı senaryoyu birebir simüle etmelidir.** Bizim uygulamamız, kullanıcıların telefonlarıyla çektiği fotoğraflar üzerinde çalışacak. Bu yüzden modelin başarısını, tam olarak bu tipteki verilerle ölçmeliyiz.
- **Dağıtım:** Bu 2000 fotoğrafı ikiye böleriz:
    - **Validation Seti:** Örneğin 500 kedi + 500 köpek fotoğrafı. Bu set, en iyi modeli seçmek için yapacağımız **"deneme sınavı"** olacak.
    - **Test Seti:** Geriye kalan 500 kedi + 500 köpek fotoğrafı. Bu set ise şampiyon modelin **"nihai final sınavı"** olacak.

Özet ve Ana Fikir

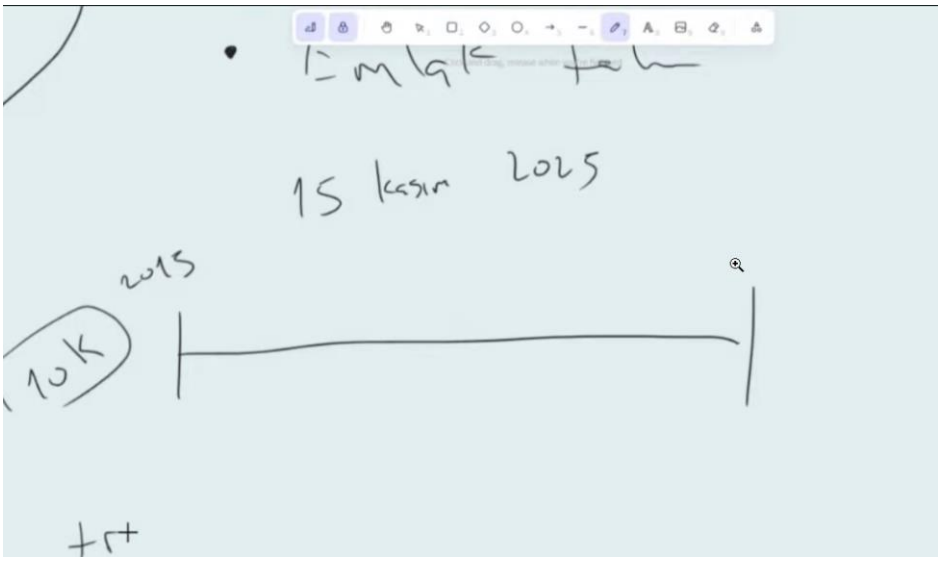
Bu görselin en büyük mesajı şudur:

"Modelini olabildiğince çeşitli ve büyük bir veriyle EĞİT, ama başarısını HER ZAMAN gerçek dünyada karşılaşacağı veri tipiyle TEST ET."

- **Train Seti:** Çeşitlilik ve miktar önemlidir. (İnternet verisi + Kendi verimiz)
- **Validation & Test Setleri: Gerçek dünya simülasyonu** önemlidir. Kalite ve temsil yeteneği, miktardan daha kritiktir. (Sadece kendi verimiz)

Bu strateji, modelinizin laboratuvarda değil, **sahada** başarılı olmasını sağlar.

17



Bu görsel, veri bölme stratejilerindeki en önemli istisnalardan birini, yani **zaman serisi verileriyle** nasıl çalışılması gerektiğini anlatıyor. Bu, kedi-köpek probleminden çok daha farklı ve dikkat edilmesi gereken bir konsepttir.

Görsel 15: Zamana Dayalı Veri Bölme - "Gelecekten Kopya Çekme!"

Genel Anlam: Neyi Anlatıyor?

Bu görsel, içinde **zaman boyutu** olan verilerle (emlak fiyatları, hisse senedi değerleri, hava durumu, satış tahminleri vb.) çalışırken, standart rastgele veri bölme (train_test_split) yönteminin **neden kesinlikle kullanılmaması gerektiğini** anlatır. Amaç, modelin "gelecekten bilgi sızdırarak" geçmişi tahmin etmesini engellemektir.

- **Senaryo:** Emlak fiyatı tahmin modeli.
- **Veri:** 2015'ten bugüne kadar satılmış evlerin bilgileri.
- **Hedef:** Gelecekteki bir tarih (15 Kasım 2025) için fiyat tahmini yapmak.

Python Dünyası ile Anlatım

Emlak satış verilerini içeren, tarihe göre sıralanmış bir DataFrame'imiz olduğunu varsayalım.

1. HATALI Yaklaşım: Rastgele Bölme

```
from sklearn.model_selection import train_test_split

# Bu yaklaşım ZAMAN SERİSİNDE KULLANILMAZ!
# Model, 2023'teki bir evi görerek 2018'deki bir evi tahmin etmeyi öğrenebilir.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

Bu, modeli **"gelecekten bilgi sızdırarak"** test etmektir. Modeliniz test setinde yanıltıcı bir şekilde çok başarılı çıkar ama gerçek hayatta (gerçek geleceği tahmin ederken) tamamen başarısız olur.

2. DOĞRU Yaklaşım: Zamana Dayalı (Kronolojik) Bölme

Veriyi asla karıştırmamalıyız. Bunun yerine, zaman çizgisinde bir "kesme noktası" belirleriz.

```
import pandas as pd

# df, 'tarih' sütununa göre sıralanmış olmalı.
# df = df.sort_values(by='tarih')

# Veri setinin ilk %80'ini Train, son %20'sini Test olarak ayıralım.
split_point = int(len(df) * 0.8)

train_df = df.iloc[:split_point]
test_df = df.iloc[split_point:]

X_train, y_train = train_df[['ozellikler']], train_df['fiyat']
X_test, y_test = test_df[['ozellikler']], test_df['fiyat']

# Modeli SADECE geçmiş veriyle (train_df) eğitiriz.
# model.fit(X_train, y_train)

# Modelin performansını, daha önce hiç görmediği GELECEK veriyle (test_df)
ölçeriz.
# model.score(X_test, y_test)
```

Bu yöntem, modelin **gerçek dünya senaryosunu** doğru bir şekilde simüle eder: **"Geçmişe bakarak geleceği tahmin et."**

Analojilerle Anlatım

1. Sınava Hazırlanan Öğrenci Analojisi

- **Problem:** Bir tarih sınavına hazırlanıyorsunuz.
- **Hatalı Yaklaşım:** Çalışma sorularınızın (train) arasına, final sınavı sorularından (test) bazıları karışmış. Yani "Osmanlı'nın Yükseliş Dönemi"ni çalışırken, cevap anahtarında "I. Dünya Savaşı'nın sonuçları" hakkında ipuçları görüyorsunuz. Bu, kopya çekmektir.
- **Doğru Yaklaşım:** Önce "Osmanlı Tarihi"ni (train) baştan sona çalışır, sonra bilginizi daha önce hiç görmediğiniz "Türkiye Cumhuriyeti Tarihi" (test) sorularıyla sınırsınız.

2. Bisküvi Üretimi Analojisi

- **Problem:** Üretim hattındaki bir fırının yarın sabahki sıcaklığını tahmin etmeye çalışıyorsunuz. Elinizde geçmiş bir aylık sıcaklık verisi var.
- **Hatalı Yaklaşım:** Veriyi rastgele karıştırırsınız. Modeliniz, yarınki tahmini yaparken dünkü saat 15:00 ve yarından sonraki saat 10:00 verilerini aynı anda görerek "aradaki boşluğu doldurur".
- **Doğru Yaklaşım:** İlk üç haftanın verisiyle (train) modeli eğitir, son bir haftanın verisiyle (test) modelin tahmin gücünü ölçersiniz. **"Geçmiş 3 haftaya bakarak, gelecek 1 haftayı ne kadar iyi tahmin edebiliyorum?"** diye sorarsınız.

3. Restoran Analojisi

- **Problem:** Gelecek ayki müşteri sayısını tahmin etmek istiyorsunuz. Elinizde son 5 yıllık müşteri verisi var.
- **Hatalı Yaklaşım:** Tüm veriyi karıştırırsınız. Model, 2018'deki bir ayın tahminini yaparken, 2023'teki pandemi sonrası müşteri alışkanlıklarından "bilgi sızdırır". Bu, gerçekçi bir test olmaz.
- **Doğru Yaklaşım:** Modeli 2018-2022 verileriyle (train) eğitir ve 2023 verileriyle (test) ne kadar başarılı olduğunu ölçersiniz. Bu, modelin "geleceği" ne kadar iyi öngördüğünün dürüst bir ölçüsüdür.

Özet

- **Anahtar Kelime:** Zamana Dayalı Bölme (Time Series Split), Kronolojik Bölme, Veri Sızıntısı (Data Leakage), İleriye Yönelik Test (Forward Chaining).
- **Kural:** Eğer verinizde **zaman** faktörü varsa (tarih damgaları, sıralı olaylar vb.), veri setini karıştırmak ve rastgele bölmek yapabileceğiniz en büyük metodolojik hatadır. Her zaman **kronolojik sıraya** göre bölmelisiniz.
- **Neden Önemli?:** Bu yöntem, modelinizin performansını dürüstçe ölçmenizi ve gerçek hayatta karşılaşacağı **ekstrapolasyon** (bilinenin dışına çıkma) görevine onu doğru bir şekilde hazırlamanızı sağlar.

Senaryo: Bir **emlak fiyatı tahmin** modeli geliştiriyoruz. Amacımız, gelecekteki bir tarihte (örneğin 15 Kasım 2025'te) bir evin fiyatının ne olacağını tahmin etmek.
Veri Setimiz: 2015'ten bugüne kadar satılmış 110 bin evin bilgisi (metrekare, oda sayısı, konum, satış tarihi, satış fiyatı vb.).

En BÜYÜK ve en YAYGIN HATA: Rastgele Bölme

Kedi-köpek senaryosunda ne yapmıştık? 2000 fotoğrafı karıştırıp rastgele bir şekilde Train, Validation ve Test olarak ayırmıştık. Eğer aynı mantığı burada da uygularsak ne olur?

- **Train Seti:** 2015-2023 arası rastgele seçilmiş ev satışları.
- **Test Seti:** Yine 2015-2023 arası rastgele seçilmiş ev satışları.

Bu neden KORKUNÇ bir hatadır?
Çünkü bu durumda model, **gelecekten bilgi sızdırarak** geçmişi tahmin etmeye çalışır. Şöyle düşünün:
Modeliniz, **2018'deki** bir evin fiyatını tahmin etmeye çalışıyor (bu ev Test setine düştü diyelim). Ama Train setinde **2022'deki** ev fiyatları hakkında bilgi sahibi. 2022'deki yüksek fiyatları ve ekonomik koşulları "gören" modelin, 2018'deki bir evin fiyatını doğru tahmin etmesi çok kolaylaşır. Bu bir **kopya çekmektir!** Modeliniz gerçek hayatta asla sahip olamayacağı bir avantaja sahip olur: **geleceği bilmek.**
Böyle bir model test setinde harika sonuçlar verir ama gerçek hayatta, yani **"gerçek gelecek"** için tahmin yapması gerektiğinde tamamen çuvallar.

Doğru Yöntem: Zamana Dayalı Bölme (Time-Based Split)

Görseldeki zaman çizgisi tam olarak bu doğru yöntemi işaret ediyor. Zamanla ilgili verilerle (hisse senedi fiyatları, emlak fiyatları, hava durumu, satış tahminleri vb.) çalışırken **asla rastgele bölme yapmayız.**
Doğru strateji şudur: **"Geçmişle eğit, yakın geçmişle doğrula, gelecekle test et."**

1. Train Seti (Geçmiş):

- **Veri:** 2015'ten, diyelim ki 2021'in sonuna kadar olan tüm ev satışları.
- **Amaç:** Modelin, piyasanın uzun vadedeki temel dinamiklerini, mevsimsel etkileri, faiz oranlarının fiyatları nasıl etkilediğini vb. **öğrenmesini** sağlamak. Burası modelin "tarih dersi" aldığı yerdir.

2. Validation Seti (Yakın Geçmiş):

- **Veri:** 2022 yılı içindeki tüm ev satışları.
- **Amaç:** Geçmişe bakarak öğrendikleriyle, "yakın geleceği" (yani 2022'yi) ne kadar iyi tahmin edebildiğini görmek. Farklı modelleri (M1, M2...) burada yarıştırırız. 2021'e kadar olan verilerle eğitilen bir modelin 2022 tahminleri ne kadar iyi? En iyi modeli burada **seçeriz**.

3. Test Seti (En Yakın Geçmiş / "Gelecek" Simülasyonu):

- **Veri:** 2023 yılı içindeki tüm ev satışları. Bu veri, en başından beri kilitli kasadadır.

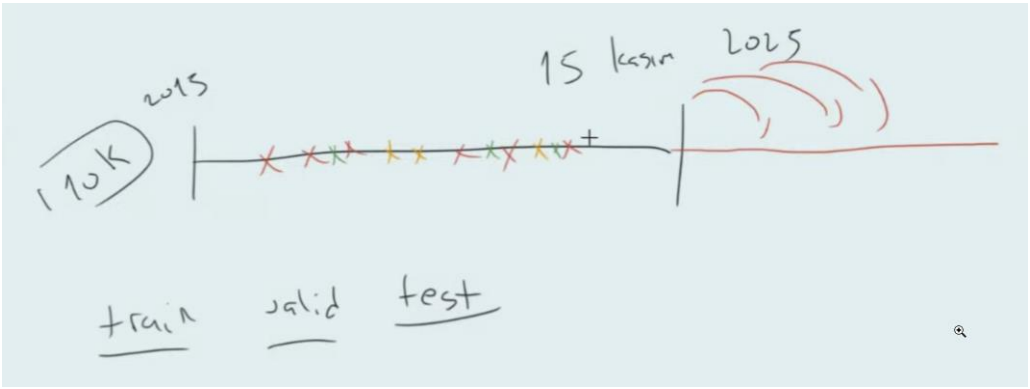
- **Amaç:** Validation setinde şampiyon olarak seçtiğimiz modelin, daha önce hiç görmediği **en güncel verilerle** nasıl performans gösterdiğini **ölçmek.** Bu, modelimizin "15 Kasım 2025" gibi gerçek bir gelecek tahmini yapmadan önceki son ve en dürüst sınavıdır. Buradan alacağı not, modelin gerçek dünya performansını gösterir.

Özetle

Veri Seti	Kedi-Köpek (Rastgele Bölme)	Emlak Fiyatı (Zamana Dayalı Bölme)
Train	Rastgele %70	En Eski Veri (örn. 2015-2021)
Validation	Rastgele %15	Ortadaki Veri (örn. 2022)
Test	Rastgele %15	En Yeni Veri (örn. 2023)

Bu görselin ana fikri: Eğer verinizde **zaman** faktörü varsa, veri setini karıştırmak ve rastgele bölmek, yapabileceğiniz en büyük metodolojik hatadır. Her zaman **kronolojik sıraya** göre bölmelisiniz.

18



Bu görsel, bir önceki teorik anlatımı daha da ileri taşıyarak, zaman serisi verileri için **doğru Train-Validation-Test bölme stratejisini** görselleştiriyor ve **İnterpolasyon/Ekstrapolasyon** kavramlarıyla olan ilişkisini netleştiriyor.

Görsel 16: Zamana Dayalı Bölme Stratejisi - Ekstrapolasyon Sınavı

Genel Anlam: Neyi Anlatıyor?

Bu görsel, zaman serisi verilerini (emlak fiyatları gibi) bölmek için kullanılan en yaygın ve en doğru yöntemi gösterir: **Kronolojik olarak sıralı bölme.** Amaç, modelin gerçek dünya görevini, yani **ekstrapolasyonu (bilinenin dışına çıkmayı)** simüle ederek onu dürüstçe test etmektir.

- **Zaman Çizgisi:** Soldan sağa akan zamanı (2015'ten 2025'e) temsil eder.
- **Renkli 'X'ler:** Zaman çizgisindeki veri noktalarını (satışları) temsil eder.
- **Bölünme:** Veri seti, zaman içinde üç ardışık parçaya bölünür:
    - **Kırmızı (Train):** En eski veriler.
    - **Yeşil (Validation):** Ortadaki veriler.
    - **Sarı (Test):** En yeni veriler.

Python Dünyası ile Anlatım

scikit-learn'in TimeSeriesSplit aracı bu işi otomatikleştirebilir, ancak temel mantık manuel olarak da kolayca uygulanabilir.

```
import pandas as pd

# df, 'tarih' sütununa göre sıralanmış olmalı.
# df = df.sort_values(by='tarih').reset_index(drop=True)

# Veriyi 70% Train, 15% Validation, 15% Test olarak bölelim.
train_end = int(len(df) * 0.70)
val_end = int(len(df) * 0.85)

# 1. Train Seti: En eski %70'lik veri
train_df = df.iloc[:train_end]
# Örnek: 2015-2021 arası veriler

# 2. Validation Seti: Sonraki %15'lik veri
val_df = df.iloc[train_end:val_end]
# Örnek: 2022 yılı verileri

# 3. Test Seti: En yeni %15'lik veri
test_df = df.iloc[val_end:]
# Örnek: 2023 yılı verileri

# Model geliştirme süreci:
# 1. Modelleri SADECE train_df ile eğit.
# 2. En iyi modeli, val_df üzerinde yarıştırarak seç.
# 3. Şampiyon modelin nihai performansını, test_df üzerinde SADECE BİR KEZ ölç.
```

İnterpolasyon ve Ekstrapolasyon ile İlişkisi

Bu bölme stratejisinin dehası, modeli sürekli olarak **Ekstrapolasyon** yapmaya zorlamasıdır.

- **Validation Aşaması:** Model, 2015-2021 verilerine bakarak, daha önce hiç görmediği **2022 yılını** tahmin etmeye çalışır. Bu, bilinenin dışına çıkan bir **ekstrapolasyon** görevidir.
- **Test Aşaması:** En iyi model, 2015-2022 verilerine bakarak, yine hiç görmediği **2023 yılını** tahmin etmeye çalışır. Bu, daha da zorlu bir **ekstrapolasyon** sınavıdır.

Eğer rastgele bölme yapsaydık ne olurdu?

Modelden 2019'daki bir evin fiyatını tahmin etmesini isterken, eğitim setinde 2018 ve 2021 verileri olabilirdi. Bu durumda model, aradaki boşluğu dolduran bir **interpolasyon** yapardı. Bu, modeli kandırmaktır; çünkü asıl görevi (15 Kasım 2025'i tahmin etmek) bir ekstrapolasyon görevidir.

Analojilerle Anlatım

1. Sınava Hazırlanan Öğrenci Analojisi

- **Train:** Öğrenci, "İlk Çağ" ve "Orta Çağ" tarihini çalışır.
- **Validation:** Öğrendikleriyle "Yeni Çağ" hakkındaki deneme sınavı sorularını çözmeye çalışır. (İlk ekstrapolasyon denemesi).
- **Test:** En iyi çalışma tekniğini belirledikten sonra, "Yakın Çağ Tarihi" hakkındaki final sınavına girer. (İkinci ve daha zorlu ekstrapolasyon sınavı).

2. Bisküvi Üretimi Analojisi

- **Train:** Fırın sıcaklığı tahmin modelini, Ocak-Eylül arasındaki verilerle eğitirsiniz.
- **Validation:** Ekim ayı sıcaklıklarını ne kadar iyi tahmin ettiğini test edersiniz.
- **Test:** En iyi modelinizin, Kasım ayı sıcaklıklarını ne kadar iyi tahmin ettiğini ölçersiniz.

3. Restoran Analojisi

- **Train:** Müşteri sayısı tahmin modelini, restoranın ilk üç yılındaki (2020-2022) verilerle eğitirsiniz.
- **Validation:** 2023'ün ilk yarısındaki müşteri sayısını ne kadar iyi tahmin ettiğini test ederek en iyi modeli seçersiniz.
- **Test:** Şampiyon modelinizin, 2023'ün ikinci yarısındaki müşteri sayısını ne kadar isabetli tahmin ettiğini raporlarsınız.

Özet

- **Anahtar Kelime:** Zamana Dayalı Bölme, Kronolojik Bölme, Ekstrapolasyon, İnterpolasyon Tuzağı.
- **Kural:** Zaman serisi verileri **asla karıştırılmaz.** Her zaman kronolojik olarak bölünür: **En eski veri Train, ortadaki veri Validation, en yeni veri Test'tir.**
- **Neden Önemli?:** Bu strateji, modelinizin gerçek dünya görevini (geleceği tahmin etmek) doğru bir şekilde simüle eder ve size modelinizin **ekstrapolasyon gücü** hakkında dürüst bir fikir verir.

göre bölmelisiniz

Önce Kavramlar: İnterpolasyon vs. Ekstrapolasyon

1. İnterpolasyon (Aradaki Boşluğu Doldurma):

- **Anlamı:** Elinizdeki veri noktalarının **arasındaki** bir değeri tahmin etmektir. Bildiğiniz bir aralığın içindeki bir boşluğu doldurmaktır.
- **Analoji:** Bir çocuğun 8 yaşındaki boyunu ve 10 yaşındaki boyunu biliyorsunuz. 9 yaşındaki boyunu tahmin etmeniz isteniyor. Bu bir **interpolasyondur.** Genellikle oldukça isabetli ve "güvenli" bir tahmindir.

2. Ekstrapolasyon (Bilinenin Dışına Çıkma):

- **Anlamı:** Elinizdeki veri noktalarının **dışındaki** bir değeri tahmin etmektir. Bildiğiniz aralığın ötesine geçerek bir tahminde bulunmaktır.
- **Analoji:** Aynı çocuğun 10 yaşına kadar olan tüm boy verilerine sahipsiniz. 25 yaşındaki boyunu tahmin etmeniz isteniyor. Bu bir **ekstrapolasyondur.** Çok daha zor, riskli ve belirsizliklerle doludur.

Görsel Bu Kavramlarla Bize Ne Anlatıyor?

Bu görsel, makine öğrenmesi modelimizi **doğru göreve hazırlamanın** önemini anlatıyor.

Senaryomuz neydi? 15 Kasım 2025'teki bir emlak fiyatını tahmin etmek. Bu görev, doğası gereği bir **EKSTRAPOLASYON** problemidir. Bizden bilinenin dışına çıkmamız isteniyor.

Peki, veri setimizi nasıl bölmeliyiz ki modelimiz bu zorlu göreve hazırlansın?

Görseldeki Doğru Yöntem (Zamana Dayalı Bölme):

1. **Train Seti (Kırmızı X'ler - En Eski Veri):**
    - Model, 2015-2021 arasındaki verilerle eğitilir. Geçmişin "kurallarını" öğrenir.
2. **Validation Seti (Yeşil X'ler - Ortadaki Veri):**
    - Modelin, 2021'e kadar öğrendikleriyle 2022 yılını ne kadar iyi tahmin edebildiğini test ederiz. Bu, küçük bir **ekstrapolasyon** denemesidir.
3. **Test Seti (Sarı X'ler - En Yeni Veri):**
    - En iyi modelimizin, 2022'ye kadar öğrendikleriyle 2023 yılını ne kadar iyi tahmin edebildiğini ölçeriz. Bu da daha zorlu bir **ekstrapolasyon** sınavıdır.

Bu yöntemin GÜZELLİĞİ şudur:

Modeli, eğitim (train) sürecinden itibaren sürekli olarak **ekstrapolasyon yapmaya zorlarız.** Onu gerçek hayattaki görevine hazırlarız.

Peki Yanlış Yöntem (Rastgele Bölme) Ne Yapardı?

Eğer tüm veriyi karıştırıp rastgele bölseydik, Train setimizde 2023'ten, Test setimizde ise 2018'den veri olabilirdi. Bu durumda modelden 2018'deki bir değeri tahmin etmesini istediğimizde, bu bir **İNTERPOLASYON** problemine dönüşürdü. Model, zaten 2023'ü bildiği için aradaki 2018'i çok kolay tahmin ederdi.

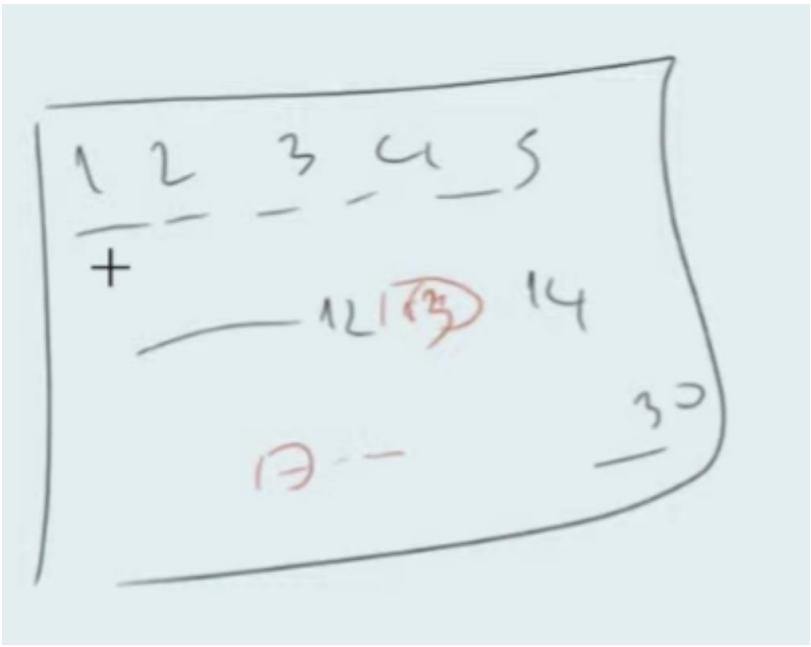
Bu, modeli kandırmaktır. Ona kolay bir **interpolasyon** sınavı yapıp, sonra gerçek hayatta zor bir **ekstrapolasyon** problemiyle baş başa bırakmaktır.

Özetle, görselin ana mesajı:

Model geçmişi kullanarak eğitilmeli (train); validasyon ve test ise daha güncel verilerle yapılmalıdır. Çünkü bu strateji, modeli "aradaki boşlukları doldurma" (interpolasyon) gibi kolay bir işe değil, asıl görevi olan "bilinenin dışına çıkma" (ekstrapolasyon) gibi zor bir işe hazırlar.

Bu, zaman serisi modellemede başarının altın kuralıdır.

19



Bir marketin kasası

13 de eksik var

problem ne ile karşılaşıyorsun. gerçek hayatı yansıtıyor mu?

Bu "marketin kasası" görseli, veri analizindeki en sinsi ve en önemli tuzaklardan birini anlatıyor: **Sistematik Eksik Veri** ve **Sağkalım Yanlılığı (Survivorship Bias)**.

Görsel 17: Market Kasası Problemi - "Görmediğin Veri, Gördüğünden Daha Önemlidir"

Genel Anlam: Neyi Anlatıyor?

Bu görsel, bir veri setindeki eksikliklerin (kayıp verilerin) rastgele olmayabileceğini ve bu eksikliklerin altında yatan nedenin, analizin sonucunu tamamen değiştirebilecek kritik bilgiler içerebileceğini anlatır. Asıl hikaye, genellikle veri setinde **olmayan** verilerde gizlidir.

- **1, 2, 3... 12, (13), 14...:** Bir marketin kasasından geçen işlem numaraları veya müşteri sıraları gibi sıralı bir veri.
- **Eksik 13:** Veri setinde 13 numaralı işlemin kaydı yok.
- **Soru:** Bu eksiklik ne anlama geliyor ve bizim analizimizi nasıl etkiler?

Python Dünyası ile Anlatım

Bir DataFrame'imiz olduğunu ve günlük işlem sayısını hesaplamak istediğimizi düşünelim.

1. YANLIŞ Yaklaşım: Eksik Veriyi Görmezden Gelmek

```
import pandas as pd

# 'islem_no' sütununda 13 eksik
data = {'islem_no': [1, 2, ..., 12, 14, ...], 'ciro': [...]}
df = pd.DataFrame(data)

# Sadece mevcut satırları sayar.
gunluk_islem_sayisi = len(df)
# Bu, gerçekte o gün kaç işlem yapıldığını DEĞİL,
# sadece bizim kayıtlarımızda kaç işlem olduğunu söyler.
```

Bu yaklaşım, **"veri neden eksik?"** sorusunu sormadığı için tehlikelidir.

2. DOĞRU Yaklaşım: Eksikliğin Nedenini Sorgulamak

Problemi bir dedektif gibi ele almalıyız. "13 numaralı işlem neden yok?"

- **Senaryo A: Rastgele Hata.** O anda sistem anlık olarak çöktü ve tek bir işlem kaydedilemedi. Bu durumda, eksikliği görmezden gelmek veya ortalama bir değerle doldurmak çok büyük bir hata yaratmayabilir.
- **Senaryo B: Sistematik Hata (En Tehlikelisi).** 13 numaralı işlem, sahte bir kredi kartı denemesi olduğu için sistemin sahtekarlık tespit modülü tarafından **otomatik olarak silindi.**

Eğer Senaryo B doğruysa ve biz bunu fark etmezsek ne olur?

```
# Elimizdeki veri, SADECE "başarılı" işlemleri içeriyor.
# Tüm "sahtekarlık" denemeleri daha biz görmeden silinmiş.
basarili_islemler_df = df

# Bu "temiz" veriyle bir sahtekarlık tespit modeli eğitmeye çalışalım.
# model.fit(basarili_islemler_df_X, basarili_islemler_df_y)
```

Bu model **asla sahtekarlık örneği görmediği için**, sahtekarlığın neye benzediğini **öğrenemez.** Gerçek hayatta bir sahtekarlık denemesiyle karşılaştığında tamamen savunmasız kalır. Elimizdeki veri, gerçek hayatı değil, **"olmasını istediğimiz ideal hayatı"** yansıtır.

Analojilerle Anlatım

1. II. Dünya Savaşı Uçakları Analojisi (Klasik Örnek)

- **Problem:** Savaştan dönen uçakların en çok hasar alan yerlerini güçlendirerek zırh eklemek.
- **Veri Seti:** Savaştan **geri dönebilen** uçakların üzerindeki mermi delikleri.
- **Yanıltıcı Sonuç:** En çok delik kanatlarda ve gövdede. "O zaman kanatları ve gövdeyi güçlendirelim."

- **Doğru Analiz (Eksik Veriyi Düşünmek):** Asıl soru şudur: "Hangi uçaklar geri **dönemedi**?" Geri dönemeyen uçaklar, motor ve kokpit gibi kritik yerlerden vurulanlardır. Dolayısıyla, güçlendirilmesi gereken yerler, veri setimizde **hiç delik olmayan** yerlerdir. Çünkü oradan vurulanlar geri gelemiyor!

2. Bisküvi Üretimi Analojisi

- **Problem:** Gün sonu kalite raporu hazırlamak.
- **Veri Seti:** Üretim bandının sonundaki kutulara **girmeyi başaran** bisküviler.
- **Eksik Veri:** Üretim sırasında kırılıp veya yanıp, daha kutuya bile ulaşamadan **hattan düşen** bisküviler.
- **Hata:** Eğer sadece kutudaki bisküvilere bakarak "Bugün üretimimiz %99 başarılıydı" derseniz, hattan düşen yüzlerce bozuk bisküviyi görmezden gelmiş olursunuz. Raporunuz, gerçek hayatı yansıtmaz.

3. Restoran Analojisi

- **Problem:** Müşteri memnuniyetini ölçmek.
- **Veri Seti:** Restoranın web sitesine girip **pozitif yorum bırakan** müşteriler.
- **Eksik Veri:** Kötü bir deneyim yaşayıp, bir daha asla gelmemeye yemin eden ve yorum yazma zahmetine bile girmeyen **"kayıp müşteriler"**.
- **Hata:** Sadece pozitif yorumlara bakarak "Restoranımız harika işliyor" sonucuna varmak, sizi körleştirebilir. Asıl önemli bilgi, neden memnun kalmadıklarını size söylemeyen o sessiz ve kayıp çoğunluktadır.

Özet

- **Anahtar Kelime:** Eksik Veri, Sistematik Eksiklik, Sağkalım Yanlılığı (Survivorship Bias), Seçilim Yanlılığı (Selection Bias).
- **Kural:** Bir veri setini analiz ederken, sadece içindeki verilere değil, **içinde olmayan verilere de** odaklanmalısın.
- **Neden Önemli?:** Verinizin neden eksik olduğunu anlamak, yapacağınız analizin ve kuracağınız modelin gerçeği yansıtıp yansıtmayacağını belirler. "Temiz" görünen bir veri seti, aslında "filtrelenmiş" ve gerçeğin sadece küçük bir kısmını gösteren yanıltıcı bir veri seti olabilir.

Senaryo: Market Kasası Problemi

Ne Görüyoruz?:

- Bir zaman çizelgesi var: 1, 2, 3, 4, 5... 12, 13, 14... 17... 30 gibi sayılar. Bunlar muhtemelen günleri veya saatleri temsil ediyor.
- Bu çizelgede bazı "boşluklar" var. Özellikle **13** sayısı belirgin bir şekilde eksik ve üzeri çizilmiş gibi. 17-- gibi başka boşluklar da var.
- **Problem:** Diyelim ki bu sayılar, bir marketin kasasından geçen müşteri numaraları veya işlem ID'leri. Ve biz bu verilere bakarak "Günde ortalama kaç müşteri geliyor?" sorusunu cevaplamaya çalışıyoruz.

İlk Bakış ve Basit Çözüm (Yanıltıcı Olan)

Acemi bir analist ne yapar? Elindeki verilere bakar: "Tamam, elimde 1, 2, 3, 4, 5, ..., 12, 14, ..., 17, ..., 30 numaralı işlemler var. Bu aralıktaki toplam işlem sayısını sayar, geçen süreye böler ve bir ortalama bulurum."

Bu neden KORKUNÇ bir hatadır?

Çünkü bu analist, en kritik soruyu sormayı unutmuştur: **"Peki ya 13 numaralı işlem? O neden listede yok?"**

Gerçek Hayat ve Derin Problem

İşte bu noktada dedektiflik başlar. "13 numara neden eksik?" sorusunun birçok cevabı olabilir ve her bir cevap, analizimizin sonucunu tamamen değiştirir.

Olası Sebepler ve Anlamları:

1. **"13 uğursuz sayı olduğu için o işlem numarası atlandı."**
    - **Anlamı:** Aslında o saatte bir müşteri gelmedi. Veri setimiz gerçeği yansıtıyor. Analizimize devam edebiliriz. (En düşük olasılık).
2. **"13 numaralı işlem sırasında sistem çöktü ve o veri KAYDOLMADI."**
    - **Anlamı:** Aslında o saatte bir müşteri geldi, hatta belki de en yüksek cirolu müşteriydi, ama biz onu **göremiyoruz.** Elimizdeki veri seti **gerçeği yansıtmayan, eksik bir settir.** Eğer bu kayıp veriyi görmezden gelirsek, günlük müşteri sayısını ve ciroyu **olduğundan daha düşük** hesaplamış oluruz.
3. **"13 numaralı işlem, sahte bir kredi kartı denemesi olduğu için sistem tarafından OTOMATİK olarak silindi."**
    - **Anlamı:** İşte en tehlikeli senaryo bu! Bizim veri setimiz, sadece **"başarılı"** işlemleri içeriyor. Tüm "başarısız" veya "hatalı" işlemler daha biz görmeden sistem tarafından temizlenmiş.
    - **Bu neden tehlikeli?** Eğer biz bu "temizlenmiş" veriye bakarak bir sahtekarlık tespit modeli (fraud detection) eğitmeye çalışırsak ne olur? Modelimiz asla sahtekarlık örneği görmediği için, sahtekarlığın neye benzediğini **öğrenemez!** Elimizdeki veri, gerçek hayatı değil, **"olmasını istediğimiz ideal hayatı"** yansıtıyor.

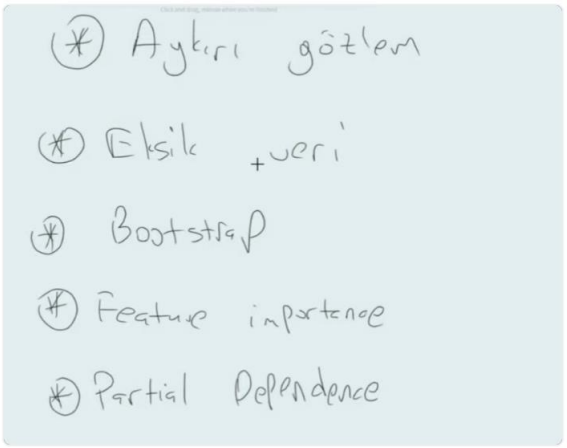
Görselin Ana Mesajı: "Görmediğin Veri, Gördüğünden Daha Önemli Olabilir."

Bu "market kasası" problemi bize şunu öğretir:

Bir veri setini analiz etmeye başlamadan önce, sadece içindeki verilere değil, **içinde olmayan verilere de** odaklanmalısın.

- **Veri neden eksik?** Rastgele bir teknik hata mı, yoksa sistematik bir filtreleme mi var?
- **Bu veri seti gerçek hayatı yansıtıyor mu?** Yoksa sadece "başarılı", "hayatta kalmış" veya "istenen" durumları mı içeriyor?
- Eğer sadece başarılı durumlarla bir model eğitirsek, o model başarısızlığın neye benzediğini asla öğrenemez ve gerçek hayatta tamamen savunmasız kalır.

Bu, II. Dünya Savaşı'ndaki "vurulmayan uçakların zırhını güçlendirme" hikayesiyle aynı mantıktır. Asıl hikaye, üsse geri dönemeyen, yani veri setinde **olmayan** uçaklardadır.



Bu görsel, pratik makine öğrenmesi uygulamalarında ve model yorumlamada sıkça karşılaşılan beş önemli kavramı listeliyor.

Görsel 18: İleri Düzey Kavramlar - Modelin Derinliklerine İnmek

Bu liste, bir model kurduktan sonra karşılaşılan pratik sorunları çözmek ve modelin "beyninin" nasıl çalıştığını anlamak için kullanılan temel teknikleri özetler.

---

1. Aykırı Gözlem (Outlier)

- **Anlamı:** Veri setinizin genel desenine uymayan, "aşırı uçtaki" veya "beklenmedik" bir veri noktasıdır. Modellerin ortalama gibi istatistiklerini ciddi şekilde saptırabilir.
- **Python Dünyası:** Seaborn ile bir boxplot (kutu grafiği) çizdirdiğinizde, kutunun dışında kalan noktalar genellikle aykırı değer olarak kabul edilir.

```
sns.boxplot(x=df['fiyat']) # Fiyat sütunundaki aykırı değerleri gösterir.
```

- **Bisküvi Analojisi:** Üretim bandında bisküvilerin ağırlığını ölçerken, bir işçinin yanlışlıkla anahtarını tartıya düşürmesi sonucu kaydedilen **500 gramlık** ölçüm bir aykırı değerdir.
- **Restoran Analojisi:** Müşteri harcamalarını analiz ederken, bir müşterinin tüm restoranı kapatıp parti vermesi sonucu ortaya çıkan **100.000 TL'lik** tek bir fiş, bir aykırı değerdir.

---

2. Eksik Veri (Missing Data)

- **Anlamı:** Veri tablonuzdaki bir hücrenin boş (NaN, NULL) olmasıdır. Çoğu model boş verilerle çalışamaz.
- **Python Dünyası:** pandas'ta df.isnull().sum() komutuyla her sütundaki eksik veri sayısını kolayca bulabilirsiniz.

```
# Eksik değerleri sütunun ortalaması ile doldurma
df['yas'].fillna(df['yas'].mean(), inplace=True)
```

- **Bisküvi Analojisi:** Kalite kontrol sırasında bir bisküvinin ağırlık sensöründen çok hızlı geçmesi sonucu ağırlığının **kaydedilememesi** bir eksik veridir.
- **Restoran Analojisi:** Bir müşterinin anket formundaki "Yaşınız" sorusunu **boş bırakması** bir eksik veridir. Bu durumun kendisi bile ("Yaşını söylemek istemeyenler...") bir bilgi taşıyabilir.

---

3. Bootstrap

- **Anlamı:** Elinizdeki tek bir veri setinden, "yerine koyarak rastgele örnekleme" yöntemiyle binlerce yeni "sanal" veri seti yaratarak, bir istatistiksel tahminin güvenilirliğini ölçme tekniğidir.
- **Python Dünyası: scikit-learn'deki RandomForestClassifier** gibi algoritmalar, her bir ağacı verinin farklı bir bootstrap örneği üzerinde eğiterek bu tekniği temelinde kullanır.
- **Bisküvi Analojisi:** Elinizdeki **7 bisküvilik** test kutusundan, her seferinde geri koyarak 7 kez çekiliş yapıp **"sanal bir bisküvi kutusu"** yaratmak ve bu işlemi 10.000 kez tekrarlayarak ortalama çilek sayısının güven aralığını bulmak.
- **Restoran Analojisi: 7 müşterinin** verdiği puanları bir şapkaya koyup, her seferinde geri koyarak 7 kez çekiliş yapıp **"sanal bir anket grubu"** oluşturmak. Bu simülasyonu binlerce kez tekrarlayarak tatlının alacağı ortalama puanın ne kadar değişken olabileceğini tahmin etmek.

---

4. Feature Importance (Özellik Önemi)

- **Anlamı:** Bir modelin tahmin yaparken hangi özelliklere (sütunlara) **daha çok güvendiğini** veya hangilerinin sonucu **daha çok etkilediğini** gösteren bir skordur.
- **Python Dünyası:** Eğitilmiş bir **RandomForest** veya **XGBoost** modelinde .**feature_importances_** özelliği ile bu skorlara kolayca erişilebilir.

```
model = RandomForestClassifier().fit(X_train, y_train)
importances = model.feature_importances_
# Bu 'importances' dizisi, her bir özelliğin önem skorunu verir.
```

- **Bisküvi Analojisi:** Bisküvinin kırık olup olmadığını tahmin eden bir modelde, **"pişirme süresi"** özelliğinin öneminin **"un markası"** özelliğinden çok daha yüksek çıkması.
- **Restoran Analojisi:** Bir müşterinin tekrar gelip gelmeyeceğini tahmin eden bir modelde, **"son ziyaret tarihi"** özelliğinin öneminin, müşterinin **"yaşadığı şehir"** özelliğinden çok daha yüksek çıkması.

---

5. Partial Dependence (Kısmi Bağımlılık)

- **Anlamı:** Bir özelliğin, diğer tüm özellikler sabit tutulduğunda, modelin çıktısını **nasıl etkilediğini** (artırıyor mu, azaltıyor mu, yoksa bir platoya mı ulaşıyor) gösteren bir grafiktir. "Neden?" sorusunun daha derin bir cevabını verir.

- **Python Dünyası:** scikit-learn'in PartialDependenceDisplay aracıyla bu grafikleri kolayca çizebilirsiniz.

```
from sklearn.inspection import PartialDependenceDisplay
PartialDependenceDisplay.from_estimator(model, X_train, ['ozellik_adi'])
```

- **Bisküvi Analojisi:** Grafiğin, "pişirme süresi" 10 dakikadan 15 dakikaya çıktıkça kırık olma olasılığının **hızla arttığını**, ancak 15 dakikadan sonra bu artışın **yavaşladığını** göstermesi.
- **Restoran Analojisi:** Grafiğin, bir yemeğin fiyatı 100 TL'den 300 TL'ye çıktıkça sipariş edilme olasılığının **azaldığını**, ancak 300 TL'den sonra lüks segment müşterileri için bu etkinin **sabitlendiğini** göstermesi.

SAP ABAP dünyası açıklaması.

1. Aykırı Gözlem (Outlier)

- **Ne Demek?:** Veri setinizin genel yapısına uymayan, "aşırı uçtaki" veya "beklenmedik" veri noktası.
- **SAP Analojisi:** IT_VBAP (Satış Kalemleri) internal table'ınızı inceliyorsunuz. NETWR (Net Değer) sütunundaki değerlerin %99.9'u 100 TL ile 10.000 TL arasında. Ancak bir tane satırda **1 Milyar TL** yazıyor. İşte bu 1 Milyar TL'lik satır bir **aykırı gözlemdir.**
- **Ne Yapmalı?:** Önce sorgulamalısınız. Bu bir veri giriş hatası mı (10000 yazacakken yanlışlıkla sıfırlara basılmış)? Yoksa gerçekten de o yıl dev bir proje satışı mı yapıldı? Hatalıysa düzeltilir veya silinir. Gerçekse, modelin dengesini bozmaması için özel olarak ele alınması gerekebilir.

2. Eksik Veri (Missing Data)

- **Ne Demek?:** Tablonuzdaki bir hücrenin boş olması. (NULL, NaN, boş string vb.)
- **SAP Analojisi:** Bir müşteri listesi raporu (IT_KNA1) çekiyorsunuz. Bazı müşterilerin TELF1 (Telefon Numarası) alanı boş. Veya GBDAT (Doğum Tarihi) alanı 00.00.0000 olarak gelmiş. Bu bir **eksik veridir.**
- **Ne Yapmalı?:** Bu boşlukları öylece bırakamazsınız, çoğu model hata verir. Stratejiler şunlar olabilir:
    - O satırı tamamen silmek (eğer çok fazla eksik varsa).
    - Boşluğu sütunun ortalaması veya en sık tekrar eden değeri (mod) ile doldurmak.
    - Daha gelişmiş tahmin yöntemleri kullanarak o boşluğu doldurmak.

3. Bootstrap

- **Ne Demek?:** Elinizdeki tek bir veri setinden, sanki birden çok veri setiniz varmış gibi **"yeni örneklemler"** yaratma tekniğidir. Bunu, "yerine koyarak örnekleme" (sampling with replacement) ile yapar.
- **SAP Analojisi:** Elinizde 1000 müşterilik bir internal table var.
    1. Bu tablodan rastgele bir müşteri seçip adını bir kağıda yazıyorsunuz.
    2. Seçtiğiniz müşteriyi tekrar havuza (tabloya) geri koyuyorsunuz. Yani aynı müşterinin tekrar seçilme ihtimali var.

3. Bu işlemi 1000 kez tekrarlıyorsunuz. Sonuçta elinizde, bazı müşterilerin hiç olmadığı, bazılarının ise 2-3 kez tekrarlandığı 1000 kişilik yeni bir "sanal" müşteri listesi oluşur.
   İşte bu yeni listeye **bootstrap örneği** denir. Bu işlemi 100 defa yaparak, birbirinden biraz farklı 100 tane sanal internal table yaratabilirsiniz.

- **Neden Önemli?: Random Forest** gibi güçlü algoritmaların temelinde bu yatar. Her bir ağaç, verinin farklı bir bootstrap örneği üzerinde eğitilir. Bu da modelin daha sağlam ve genelleme yeteneği yüksek olmasını sağlar.

---

4. Feature Importance (Özellik Önemi)

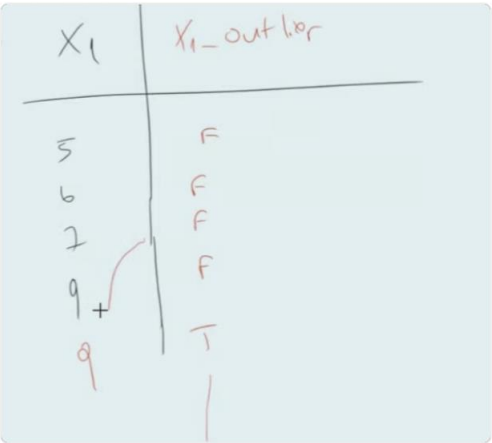
- **Ne Demek?:** Bir modelin tahmin yaparken hangi sütunlara (feature) **daha çok güvendiğini** veya hangi sütunların sonucu **daha çok etkilediğini** gösteren bir skordur.
- **SAP Analojisi:** Bir müşterinin şirketi terk edip etmeyeceğini (churn) tahmin eden bir model kurdunuz. Modelinize girdi olarak müşterinin YAŞI, ŞEHRİ, SON SİPARİŞ TARİHİ ve TOPLAM HARCAMASI gibi 20 farklı sütun verdiniz.
  - **Feature Importance** size şöyle bir sonuç verir:
    1. SON SİPARİŞ TARİHİ: %55 (En önemli!)
    2. TOPLAM HARCAMASI: %25
    3. YAŞI: %10
    4. ...
    5. ŞEHRİ: %0.1 (Neredeyse hiç önemi yok)
- **Neden Önemli?:** Bu, modelinizin "beyninin içini" görmenizi sağlar. Hangi faktörlerin iş süreçlerinizde gerçekten önemli olduğunu anlarsınız. Ayrıca önemsiz özellikleri modelden çıkararak modeli sadeleştirebilirsiniz.

---

5. Partial Dependence (Kısmi Bağımlılık)

- **Ne Demek?:** Feature Importance'ın bir adım ötesidir. Bir özelliğin, modelin çıktısını **NASIL** etkilediğini gösterir. Yani sadece "ne kadar önemli" olduğunu değil, "arttıkça mı, azaldıkça mı, yoksa U şeklinde mi etkiliyor" sorusunu cevaplar.
- **SAP Analojisi:** Feature Importance bize "Toplam Harcama"nın önemli olduğunu söyledi. **Partial Dependence Plot (PDP)** ise bize şunu gösterir:
  - "Müşterinin toplam harcaması 0 TL'den 5000 TL'ye çıktıkça, şirketi terk etme olasılığı **hızla düşüyor.**"
  - "Ancak harcama 5000 TL'yi geçtikten sonra, terk etme olasılığı üzerindeki etkisi **sabitleniyor.** Yani 6000 TL harcayanla 10.000 TL harcayan arasında pek bir fark yok."
- **Neden Önemli?:** Bu, size işinizle ilgili çok değerli ve **aksiyon alınabilir** bilgiler verir. "Demek ki hedefimiz, yeni müşterileri bir an önce 5000 TL'lik harcama barajına ulaştırmak olmalı." gibi stratejiler geliştirmenizi sağlar.

21



500 yerine 9 yazdık.

Bu görsel, pratik veri ön işleme adımlarından çok önemli bir tanesini anlatıyor: **Aykırı Değerlerle (Outliers) Başa Çıkma**. ("500 yerine 9 yazdık") ise tam olarak bu tekniğin kalbini açıklıyor.

Görsel 19: Aykırı Değerleri "Ehlileştirme" ve "İşaretleme"

Genel Anlam: Neyi Anlatıyor?

Bu görsel, veri setindeki aşırı uç değerlerin (aykırı değerlerin) model üzerindeki olumsuz etkisini azaltmak için kullanılan zekice bir iki adımlı stratejiyi gösterir: **1. Değeri Sınırla (Capping), 2. Durumu İşaretle (Flagging).**

- **X1 Sütunu:** Orijinal veriyi içerir. 5, 6, 7 gibi normal değerlerin yanında, 500 gibi aşırı bir değer olduğunu hayal edin.
- **Kırmızı Ok:** 500 gibi aykırı bir değerin, kabul edilebilir bir üst sınır olan 9 ile değiştirilmesi işlemini gösterir.
- **X1_outlier Sütunu:** Bu işlemin yapılıp yapılmadığını kaydeden yeni bir "işaret" sütunudur.

Python Dünyası ile Anlatım

```
Bu stratejiyi pandas ve NumPy kullanarak kolayca uygulayabiliriz.
import pandas as pd
import numpy as np

# 'teslimat_suresi' sütununda bir aykırı değer (500) var.
data = {'teslimat_suresi': [5, 6, 7, 500, 9]}
df = pd.DataFrame(data)

# 1. Sınırı Belirle
# Verinin %99'undan daha büyük değerleri aykırı kabul edelim.
# Bu sınırı iş bilgisiyle (domain knowledge) de belirleyebiliriz.
upper_limit = df['teslimat_suresi'].quantile(0.99)
# Örnekte bu sınırın 9 olduğunu varsayalım.
upper_limit = 9

# 2. İşaretle (Flagging): Yeni bir 'outlier' sütunu oluştur.
df['teslimat_suresi_outlier'] = np.where(df['teslimat_suresi'] > upper_limit, 1,
0)
# Eğer değer sınırdan büyükse 1 (True), değilse 0 (False) ata.

# 3. Ehlileştir (Capping): Aykırı değerleri sınır değerle değiştir.
df['teslimat_suresi'] = np.where(df['teslimat_suresi'] > upper_limit,
                                 upper_limit,
                                 df['teslimat_suresi'])

print(df)
# Çıktı:
#    teslimat_suresi  teslimat_suresi_outlier
# 0                5                        0
# 1                6                        0
# 2                7                        0
# 3                9                        1  <-- DEĞİŞİKLİKLERE DİKKAT!
# 4                9                        0
```

Bu yöntemle, 500'ün yarattığı bozulmayı engellerken, o satırda bir **anormallik yaşandığı bilgisini de kaybetmemiş** oluruz.

Model artık hem "düzeltilmiş süreyi" hem de "anormallik durumunu" birer özellik olarak kullanabilir.

Analojilerle Anlatım

1. Bisküvi Üretimi Analojisi

- **Problem:** Tartıya düşen bir anahtar yüzünden **500 gramlık** bir bisküvi ağırlığı kaydedildi.
- **1. Ehlileştirme:** Kalite mühendisi bir kural koyar: "Maksimum bisküvi ağırlığı 9 gramdır." Sistem, 500 gramlık kaydı otomatik olarak **9 gram** ile değiştirir.
- **2. İşaretleme:** Sistem aynı zamanda, o ölçümün yanına **"MANUEL DÜZELTME YAPILDI"** şeklinde bir bayrak (flag) ekler. Bu sayede, ay sonunda raporda kaç tane manuel düzeltme yapıldı

2. Restoran Analojisi

- **Problem:** Bir müşteri, memnuniyet anketine 1-10 arası puan vermek yerine yanlışlıkla **500** girer.
- **1. Ehlileştirme:** Sistem, 10'dan büyük tüm girişleri otomatik olarak en yüksek puan olan **10** ile değiştirir.
- **2. İşaretleme:** Sistem, bu düzeltilen kaydın yanına **"GEÇERSİZ GİRİŞ DÜZELTİLDİ"** notunu düşer. Restoran müdürü, bu notların sayısına bakarak anket arayüzünün kullanıcı dostu olup olmadığını anlayabilir.

Özet

- **Anahtar Kelime:** Aykırı Değer (Outlier), Sınırlama (Capping), İşaretleme (Flagging), Özellik Mühendisliği.
- **Strateji:**
    1. **Tespit Et:** Verindeki aşırı, aykırı değerleri bul.
    2. **Ehlileştir (Cap):** Bu değeri, verinin genel yapısını bozmayacak mantıklı bir sınır değere çek.
    3. **İşaretle (Flag):** Bu değişikliği yaptığını unutmamak ve modele "burada bir anormallik vardı" bilgisini ek bir özellik olarak sağlamak için yeni bir "bayrak" sütunu oluştur.
- **Neden Önemli?:** Bu iki adımlı yaklaşım, aykırı değerlerin model üzerindeki bozucu etkisini ortadan kaldırırken, bu aykırı durumun kendisinin taşıdığı değerli bilgiyi de korumanızı sağlar.

Senaryo: Veri Temizliği Operasyonu

1. Sol Sütun (X1): Orijinal, Ham Veri

- **Ne Görüyoruz?:** Bir internal table'daki tek bir sütun gibi düşünün. Örneğin, bu sütun bir siparişin **teslimat gün sayısını** tutuyor olsun.
- **Değerler:** 5, 6, 7... Bunlar normal, beklediğimiz teslimat günleri. Her şey yolunda.
- **Problem:** Şimdi, sizin de belirttiğiniz gibi, bu listede aslında **500** gibi bir değer olduğunu hayal edelim. Bir siparişin 500 günde teslim edilmesi, normalin çok dışında bir durumdur. Bu bizim **Aykırı Gözlemimiz (Outlier)**.

Neden 500 bir problem?

Eğer bu sütunun ortalamasını hesaplamaya kalksanız, (5+6+7+500)/4 gibi bir hesap, bu tek bir 500 değeri yüzünden tamamen anlamsız ve şişirilmiş bir sonuç verir. Makine öğrenmesi modelleri de tıpkı bu ortalama hesabı gibi, bu tür aşırı değerlerden çok kötü etkilenir.

2. Kırmızı Ok ve "9": Çözüm (Tekniğin Adı: Capping / Winsorizing)

- **Ne Yapıyoruz?:** 500 gibi aşırı bir değeri modelimizden tamamen silmek istemiyoruz, çünkü o satırdaki diğer bilgiler (müşteri ID'si, ürün vb.) değerli olabilir. Bunun yerine, bu değeri **"ehlileştiriyoruz".**
- **Süreç:** Verimizin genel dağılımına bakarız ve deriz ki: "Normal bir teslimat en fazla 9 gün sürer. 9 günün üzerindeki her değeri, bir hata veya çok istisnai bir durum olarak kabul edip, sanki 9 günde teslim edilmiş gibi davranacağım."
- **Uygulama:** İşte bu yüzden 500 değerini silip yerine 9 yazıyoruz. Bu işleme **"Capping" (Sınırlama/Törpüleme)** denir. Değeri, kabul edilebilir bir maksimum (veya minimum) değere sabitleriz.

3. Sağ Sütun (X1_outlier): Zekice Bir Dokunuş

- **Ne Görüyoruz?:** T (True) ve F (False) değerleri içeren yeni bir sütun.
- **Anlamı:** Bu, yaptığımız işlemi unutmamak için bıraktığımız bir **"işaret bayrağı" (flag)**'dır.
    - **F (False):** "Bu satırdaki 5, 6, 7 değerleri orijinaldi, onlara dokunmadım."
    - **T (True):** "DİKKAT! Bu satırdaki 9 değeri orijinal değil. Ben buradaki 500 gibi aşırı bir değeri 9 ile değiştirdim."

Bu yeni sütun NEDEN hayati derecede önemlidir?

Çünkü bu sayede iki farklı bilgi parçasını modelimize aynı anda vermiş oluruz:

1. Teslimatın gün sayısı (ehlileştirilmiş haliyle).
2. O teslimatta **bir anormallik yaşandığı** bilgisi.

Modelimiz bu yeni X1_outlier sütununa bakarak şunu öğrenebilir: "Hımm, demek ki teslimat gününde anormallik yaşanan siparişler, genellikle belirli bir ürün grubunda veya belirli bir bölgede oluyor." Yani model, sadece teslimat gününü değil, teslimat gününün **anormal olma durumunu** da bir desen olarak öğrenir. Bu, modelinizi çok daha akıllı yapar.

Özetle, bu görsel bize şunu anlatıyor:

1. **Tespit Et:** Verindeki aşırı, aykırı değerleri (500 gün) bul.
2. **Ehlileştir (Cap):** Bu değeri, verinin genel yapısını bozmayacak mantıklı bir sınır değere (9 gün) çek.
3. **İşaretle (Flag):** Bu değişikliği yaptığını unutmamak ve modele ek bilgi sağlamak için yeni bir "anormallik" sütunu (X1_outlier) oluştur.

Bu, bir internal table'da LOOP AT ... WHERE delivery_days > 9. yapıp, o satırdaki değeri delivery_days = 9. olarak değiştirmek ve yanına yeni bir outlier_flag = 'X'. kolonu eklemekle tamamen aynı mantıktır.

Bisküvi Fabrikası Analojisi

Senaryo: Kalite kontrol makinemiz, üretim bandından geçen bisküvilerin ağırlığını ölçüyor. Amacımız, ortalama bisküvi ağırlığını hesaplamak ve standartlara uygunluğu denetlemek.

1. Sol Sütun (X1): Orijinal Ağırlık Ölçümleri

- **Normal Veri:** Makine, bisküvileri tartar: 5 gram, 6 gram, 7 gram... Bunlar normal, kabul edilebilir küçük farklar.
- **Aykırı Gözlem (Outlier):** Aniden makine bir ölçüm yapar: **500 gram!**

- **Problem:** Bu 500 gramlık ölçüm nedir? Muhtemelen bir işçi yanlışlıkla anahtarını veya eldivenini tartının üzerine düşürdü. Bu, bir bisküvi ağırlığı değil, bir **sistem hatasıdır.** Eğer bu 500 gramı ortalama hesaplamasına dahil edersek, sanki o günkü tüm üretim bozukmuş gibi tamamen yanlış bir sonuca varırız.

2. Kırmızı Ok ve "9": Ehlileştirme (Capping)

- **Ne Yapıyoruz?:** Fabrikanın kalite mühendisi olarak bir kural koyuyoruz: "Bir bisküvi, en ideal koşullarda bile 9 gramdan daha ağır olamaz. 9 gramın üzerindeki her ölçüm bir hatadır."
- **Uygulama:** Veri analiz sistemimize otomatik bir kural ekliyoruz: "Eğer bir ağırlık ölçümü 9 gramdan büyükse, onu kayıtlara **9 gram olarak geç.**" Böylece o 500 gramlık hatalı ölçüm, ortalamamızı mahvetmeden "kabul edilebilir maksimum" değere çekilir.

3. Sağ Sütun (X1_outlier): İşaret Bayrağı

- **Ne Yapıyoruz?:** Sistemimize ikinci bir kural daha ekliyoruz: "Eğer bir ölçümü 9 gram olarak düzelttiysen, yanına bir de **'HATA VARDI'** notu düş."
- **Anlamı:**
    - **F (False):** "Bu bisküvinin ağırlığı normaldi, bir sorun yok."
    - **T (True):** "DİKKAT! Bu bisküvinin ölçümünde bir anormallik yaşandı ve ben bunu düzelttim."
- **Faydası:** Günün sonunda rapora baktığımızda, kaç tane "HATA VARDI" notu olduğunu görürüz. Eğer bu sayı çok fazlaysa, bu bize şunu söyler: "Bugün üretim bandındaki tartıda bir sorun var, bir teknisyen çağırmalıyız!" Yani bu bayrak, bize sadece veriyi düzeltmekle kalmaz, aynı zamanda operasyonel bir sorunu da işaret eder.

Restoran Analojisi

Senaryo: Restoranımızın müşteri memnuniyetini ölçmek için, müşterilerden masadaki tabletten 1 ila 10 arasında bir puan vermesini istiyoruz.

1. Sol Sütun (X1): Orijinal Müşteri Puanları

- **Normal Veri:** Puanlar gelir: 5, 6, 7, 9... Bunlar normal müşteri değerlendirmeleri.
- **Aykırı Gözlem (Outlier):** Bir müşteri puan girer: **500!**
- **Problem:** Bu 500 puan nedir? Muhtemelen müşteri tableti çocuğunun eline vermiş, o da rastgele tuşlara basmıştır. Bu bir müşteri memnuniyet puanı değil, bir **veri giriş hatasıdır.** Bu 500 puanı ortalama memnuniyet hesaplamasına katarsak, sanki restoranımız o gün dünyanın en iyi restoranı seçilmiş gibi şişirilmiş ve anlamsız bir sonuç çıkar.

2. Kırmızı Ok ve "9": Ehlileştirme (Capping)

- **Ne Yapıyoruz?:** Sistemimizin tasarımını yaparken bir kural koyarız: "Puanlama sistemi 1-10 arası çalışır. Birisi 10'dan büyük bir değer girmeye çalışırsa, sistem bunu otomatik olarak **10 puan olarak kabul etsin.**" (Görselde 9 kullanıldığı için 9 diyelim).
- **Uygulama:** O 500'lük hatalı giriş, sisteme **9 puan olarak kaydedilir.** Böylece ortalamamız bozulmaz. Müşteri muhtemelen çok memnun olduğu için yüksek bir sayı girmeye çalışmıştır diye iyi niyetli bir varsayımda bulunuruz.

3. Sağ Sütun (X1_outlier): İşaret Bayrağı

- **Ne Yapıyoruz?:** Veritabanımıza bir sütun daha ekleriz: IS_INPUT_CORRECTED (Giriş Düzeltildi mi?)

- **Anlamı:**
    - **F (False):** "Müşteri 1-10 arasında normal bir puan girdi."
    - **T (True):** "DİKKAT! Müşteri 10'dan büyük bir değer girmeye çalıştı, sistem bunu 9 olarak düzeltti."
- **Faydası:** Ay sonunda raporu çeken restoran müdürü, kaç tane "Giriş Düzeltildi" işareti olduğunu görür. Eğer bu sayı fazlaysa, bu ona şunu söyler: "Tabletlerimizdeki puanlama arayüzü kafa karıştırıcı olabilir. Çocuklar veya yaşlılar yanlışlıkla çok büyük sayılar giriyor. Arayüzü daha anlaşılır yapmalıyız." Yani bu bayrak, bize operasyonel bir iyileştirme fırsatı sunar.

22

Sokakta insanların maaşları soruluyor.

Bir kısmı cevap veriyor Bir kısmı cevap vermiyo

Kimler cevap vermiyor . Çok yüksek ve çok düşük (outlier) diye alabiliriz

Bu "sokak anketi" senaryosu, istatistikteki en temel ve en sinsi sorunlardan birini, **Eksik Veri (Missing Data)** probleminin rastgele olmadığını ve altında yatan nedenlerin analizi nasıl tamamen yanlış yönlendirebileceğini mükemmel bir şekilde özetliyor.

Sokak Anketi - "Cevap Vermeyenlerin Sırrı"

Genel Anlam: Neyi Anlatıyor?

Bu senaryo, veri setimizdeki boşlukların (eksik verilerin) tesadüfi olmayabileceğini anlatır. Eksikliğin kendisi, altında yatan sistematik bir deseni gizliyor olabilir. Bu duruma istatistikte **"Rastgele Olmayan Eksiklik" (Missing Not At Random - MNAR)** denir ve genellikle **Seçilim Yanlılığı (Selection Bias)**'na yol açar.

- **Problem:** Sokakta 100 kişiye "Aylık maaşınız ne kadar?" diye soruyoruz.
- **Veri Seti:** 70 kişi cevap veriyor, 30 kişi vermiyor. Elimizde 70 satırlık bir veri ve 30 satırlık "eksik veri" var.
- **Kritik Soru:** Cevap vermeyen 30 kişi, cevap veren 70 kişiden farklı mı?

Python Dünyası ile Anlatım

Bu durumu Python'da simüle edelim ve yanlış yaklaşımın sonuçlarını görelim.

```
import numpy as np
import pandas as pd

# Gerçek popülasyonu simüle edelim (bunu normalde bilmeyiz)
# Gelir dağılımı genellikle sağa çarpıktır.
np.random.seed(42)
gercek_maaslar = np.random.lognormal(mean=10.5, sigma=0.8, size=1000)

# Anket sürecini simüle edelim
# Çok yüksek ve çok düşük maaşlıların cevap vermeme olasılığı daha yüksek.
gozlemlenen_maaslar = []
for maas in gercek_maaslar:
    # Çok düşük (örneğin 20000'den az) veya çok yüksek (örneğin 150000'den fazla)
ise
    # %70 ihtimalle cevap vermesin.
    if maas < 20000 or maas > 150000:
        if np.random.rand() > 0.3: # %70 ihtimalle atla
            continue
    gozlemlenen_maaslar.append(maas)

gozlemlenen_maaslar = np.array(gozlemlenen_maaslar)

# ---- ANALİZ ----

# Gerçek ortalama (bilmemiz gereken doğru cevap)
gercek_ortalama = gercek_maaslar.mean()

# Yanlış yaklaşım: Sadece cevap verenlere bakarak ortalama hesaplamak
yanlis_ortalama = gozlemlenen_maaslar.mean()

print(f"Gerçek Ortalama Maaş: {gercek_ortalama:,.2f} TL")
print(f"Ankette Gözlemlenen (Yanlı) Ortalama Maaş: {yanlis_ortalama:,.2f} TL")

# Çıktı (örneksel):
# Gerçek Ortalama Maaş: 44,532.86 TL
# Ankette Gözlemlenen (Yanlı) Ortalama Maaş: 38,125.17 TL
```

Gördüğünüz gibi, sadece cevap verenlere bakarak hesapladığımız ortalama, gerçek ortalamadan **sistematik olarak daha düşüktür.** Çünkü veri toplama yöntemimiz, aykırı değerleri (hem üst hem alt) doğal olarak dışarıda bırakmıştır.

Analojilerle Anlatım

1. Savaş Uçağı Analojisi (Survivorship Bias)

- **Problem:** Savaştan dönen uçakların en çok hasar alan yerlerini zırhla güçlendirmek.
- **Veri Seti:** Savaştan **geri dönebilen** uçakların üzerindeki mermi delikleri.
- **Eksik Veri:** Geri **dönemeyen** uçaklar.
- **Yanlış Sonuç:** "Kanatlar ve gövde en çok delik alan yerler, oraları güçlendirelim."
- **Doğru Analiz:** Geri dönemeyenler, motor ve kokpit gibi hayati yerlerden vurulanlardır. Dolayısıyla, güçlendirilmesi gereken yerler, veri setimizde **hiç delik olmayan** yerlerdir. Eksik olan veri, en kritik bilgiyi taşır.

2. Bisküvi Üretimi Analojisi

- **Problem:** Günlük üretim kalitesini raporlamak.
- **Veri Seti:** Üretim bandının sonuna ulaşıp, kutuya **girmeyi başaran** bisküviler.
- **Eksik Veri:** Üretim sırasında paramparça olup, daha kutuya ulaşamadan **hattan düşen** bisküviler.
- **Yanlış Sonuç:** Sadece kutudaki bisküvilere bakarak "%99 başarı oranı" raporlamak. Bu rapor, gerçeği yansıtmaz çünkü "başarısızları" sistematik olarak dışlar.

3. Restoran Analojisi

- **Problem:** Müşteri memnuniyetini ölçmek.
- **Veri Seti:** Restoranın web sitesine veya anket defterine **yorum yazma zahmetine giren** müşteriler.
- **Eksik Veri:** Kötü bir deneyim yaşayıp, bir daha asla gelmemeye karar veren ve yorum yazarak vakit kaybetmeyen **sessiz çoğunluk.**
- **Yanlış Sonuç:** Sadece yazılı yorumlara (genellikle ya çok iyi ya da çok kötü) bakarak memnuniyeti ölçmek. Gerçek memnuniyet seviyesi, o sessiz ve kayıp çoğunluğun fikirlerinde gizlidir.

Özet

- **Anahtar Kelime:** Eksik Veri (Missing Data), Rastgele Olmayan Eksiklik (MNAR), Seçilim Yanlılığı (Selection Bias), Sağkalım Yanlılığı (Survivorship Bias).
- **Kural:** Bir veri setindeki eksikliklerle karşılaştığında her zaman sor: **"Bu veri neden eksik?"** Eksikliğin altında yatan bir desen var mı?
- **Neden Önemli?:** Eğer verinin eksik olması, verinin değeriyle (örn: maaşın çok yüksek/düşük olması) ilişkiliyse, eksik veriyi basitçe görmezden gelmek veya ortalama ile doldurmak, analizlerinizin ve modelinizin sonuçlarını tamamen geçersiz kılacak **sistematik bir yanlılığa (bias)** yol açar.

"Eksik Veri" probleminin rastgele olmadığını, altında yatan sistematik bir neden olabileceğini (Missing Not At Random - MNAR) ve bunun Aykırı Gözlem (Outlier) kavramıyla nasıl iç içe geçtiğini gösteren mükemmel bir örnek.
İşte bu senaryonun analizi:

Senaryo: Sokak Röportajı ile Maaş Anketi

Bir anketör olarak sokağa çıkıp 100 kişiye "Aylık maaşınız ne kadar?" diye soruyorsunuz.

- 70 kişi size bir cevap veriyor.
- 30 kişi ise "Cevap vermek istemiyorum" diyor veya soruyu duymazdan gelip yürüyor.

Elinizde **70 kişilik bir veri seti** ve **30 kişilik bir eksik veri** var.

Yaygın ve YANLIŞ Yaklaşım

Acemi bir analist şöyle düşünür: "Elimde 70 kişinin maaş verisi var. Bu 30 kişi cevap vermediğine göre onları yok sayayım. Kalan 70 kişinin ortalamasını alarak bu bölgedeki ortalama maaş hakkında bir fikir edinebilirim."

Bu neden KORKUNÇ ve yanıltıcı bir hatadır?

Çünkü en kritik soruyu sormayı unutmuştur: **"Cevap VERMEYEN o 30 kişi, cevap VEREN 70 kişiden farklı özelliklere mi sahip?"**

Doğru ve Derin Analiz (Sizin Yorumunuzun Dehası)

Sizin de belirttiğiniz gibi, cevap vermeyenlerin kim olduğunu tahmin edebiliriz:

1. **Çok Yüksek Maaş Alanlar (Outlier - Üst Uç):**
    - **Neden Cevap Vermezler?** Güvenlik endişeleri, kıskançlık çekinmesi, mahremiyet, "böyle sorulara cevap vermem" prensibi... Aylık geliri 250.000 TL olan bir CEO veya başarılı bir girişimci, bu bilgiyi sokakta paylaşmak istemeyebilir.
    - **Sonuç:** Sizin veri setiniz, toplumun en zengin kesimini **sistematik olarak dışarıda bırakmış olur.**
2. **Çok Düşük Maaş Alanlar veya İşsizler (Outlier - Alt Uç):**
    - **Neden Cevap Vermezler?** Utanç, mahcubiyet, bu konu hakkında konuşmak istememe... Asgari ücretin altında çalışan veya hiç geliri olmayan bir kişi, bu durumu bir yabancıyla paylaşmaktan çekinebilir.
    - **Sonuç:** Sizin veri setiniz, toplumun en fakir ve en savunmasız kesimini de **sistematik olarak dışarıda bırakmış olur.**

Peki Elimizde Ne Kaldı?

Cevap veren 70 kişi, büyük olasılıkla **orta gelir grubuna** ait insanlardır. Ne çok zengin ne de çok fakir olan, maaşını söylemekte bir beis görmeyen "ortalama" vatandaşlar.

Eğer siz, bu 70 kişinin maaş ortalamasını alırsanız, bulacağınız sonuç o bölgedeki **"orta sınıfın"** ortalama maaşı olur, **"tüm toplumun"** ortalama maaşı değil!

- **Gerçek Ortalama (Hayali):** 50.000 TL
- **Sizin Bulduğunuz Ortalama:** 35.000 TL

Bu, gerçeği yansıtmayan, **yanlı (biased)** bir sonuçtur. Çünkü veri toplama yönteminiz, doğal olarak aykırı değerleri (hem üst hem alt) dışarıda bırakmıştır.

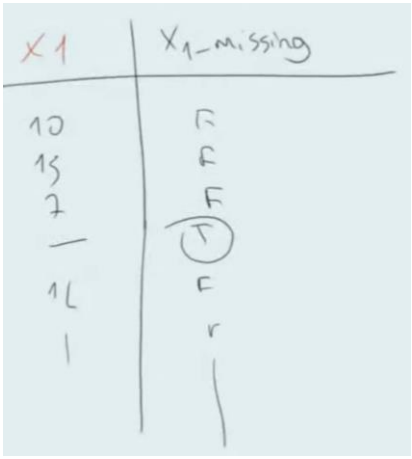
Özet ve Ana Fikir

Bu senaryo, **Eksik Veri (Missing Data)** probleminin asla basit bir "boşluk doldurma" meselesi olmadığını gösterir.

- Her zaman sormalıyız: **"Bu veri neden eksik?"**
- Eksikliğin altında yatan bir **desen** var mı?
- Bizim veri setimiz, sadece "cevap vermeye istekli" olanların, yani belirli bir profildeki insanların verisi mi?

Buna istatistikte **"Seçilim Yanlılığı" (Selection Bias)** denir. Anketinize katılanlar, katılmayanlardan sistematik olarak farklıdır ve bu da sonuçlarınızı tamamen geçersiz kılabilir. Sizin de harika bir şekilde tespit ettiğiniz gibi, bu yanlılığın en büyük kaynağı genellikle **aykırı değerlerdir.**

23



Bu görsel, pratik veri ön işleme adımlarından bir diğer önemli konuyu, **Eksik Verilerle (Missing Data) Başa Çıkma** stratejisini anlatıyor. Bu, "Aykırı Değerler" konusunda gördüğümüz "Doldur ve İşaretle" tekniğinin bir benzeridir.

Görsel 21: Eksik Verileri "Doldurma" ve "İşaretleme"

Genel Anlam: Neyi Anlatıyor?

Bu görsel, bir veri setindeki boşlukların (eksik verilerin) çoğu makine öğrenmesi modeli için bir sorun teşkil ettiğini ve bu sorunu çözmek için kullanılan zekice bir iki adımlı stratejiyi gösterir: **1. Değeri Doldur (Imputation), 2. Durumu İşaretle (Flagging).**

- **X1 Sütunu:** Orijinal veriyi içerir. 10, 15, 7 gibi normal değerlerin yanında, bir tane boş (—) değer var.
- **X1_missing Sütunu:** Verinin orijinalinde eksik olup olmadığını (True/False) kaydeden yeni bir "işaret" sütunudur.

Python Dünyası ile Anlatım

Bu stratejiyi pandas ve NumPy kullanarak kolayca uygulayabiliriz.

```
import pandas as pd
import numpy as np

# 'musteri_yasi' sütununda bir eksik değer (NaN) var.
data = {'musteri_yasi': [25, 30, 22, np.nan, 45]}
df = pd.DataFrame(data)

# 1. İşaretle (Flagging): Yeni bir 'missing' sütunu oluştur.
# 'musteri_yasi' sütunundaki değerin boş olup olmadığını kontrol et.
df['musteri_yasi_missing'] = df['musteri_yasi'].isnull().astype(int)
# .isnull() True/False döner, .astype(int) ile bunu 1/0 yaparız.

# 2. Doldur (Imputation): Eksik değeri istatistiksel bir değerle değiştir.
# En yaygın yöntemlerden biri medyan (ortanca) değer ile doldurmaktır.
median_age = df['musteri_yasi'].median()
df['musteri_yasi'].fillna(median_age, inplace=True)

print(df)
# Çıktı:
#    musteri_yasi  musteri_yasi_missing
# 0          25.0                     0
# 1          30.0                     0
# 2          22.0                     0
# 3          28.5                     1  <-- DEĞİŞİKLİKLERE DİKKAT! (28.5 medyan
değeridir)
# 4          45.0                     0
```

Bu yöntemle, modelin hata vermesini engellemek için boşluğu doldururken, **"bu verinin aslında eksik olduğu" bilgisini de kaybetmemiş** oluruz. Model, "yaşını belirtmeyen" müşterilerin farklı bir desene sahip olup olmadığını bu yeni musteri_yasi_missing sütununu kullanarak öğrenebilir.

Analojilerle Anlatım

1. Sokak Anketi Analojisi

- **Problem:** Bir kişinin maaşını öğrenemedik (eksik veri).
- **1. Doldurma:** O bölgedeki ortalama maaşı (diyelim 40.000 TL) o kişinin maaşı olarak kayıtlara geçiyoruz.
- **2. İşaretleme:** Kaydın yanına **"MAAŞ TAHMİNİ"** diye bir not düşüyoruz. Böylece model, bu kişinin gerçek maaşının 40.000 TL olmadığını, bunun sadece bir dolgu değeri olduğunu ve bu kişinin "maaşını söylemeyenler" grubuna ait olduğunu bilir.

2. Bisküvi Üretimi Analojisi

- **Problem:** Bir bisküvinin ağırlığı sensördeki bir hata yüzünden kaydedilemedi.
- **1. Doldurma:** O günkü üretimin ortalama bisküvi ağırlığını (diyelim 9.8 gram) o bisküvinin ağırlığı olarak veritabanına yazıyoruz.
- **2. İşaretleme:** Kaydın yanına **"AĞIRLIK ÖLÇÜLEMEDİ"** şeklinde bir bayrak (flag) ekliyoruz. Bu sayede model, bu bisküvinin ağırlık verisinin "gerçek" olmadığını ve o anda sensörde bir sorun yaşanmış olabileceğini öğrenir.

3. Restoran Analojisi

- **Problem:** Müşteri memnuniyet anketinde bir müşteri, "Servis Hızı" sorusunu boş bırakmış.
- **1. Doldurma:** O günkü tüm anketlerdeki "Servis Hızı" puanlarının ortalamasını (diyelim 10 üzerinden 8) bu müşterinin puanı olarak giriyoruz.
- **2. İşaretleme:** Kaydın yanına **"SERVİS PUANI GİRİLMEDİ"** notunu ekliyoruz. Model, bu nota bakarak "servis hakkında fikri olmayan" müşterilerin genel memnuniyetinin farklı olup olmadığını analiz edebilir.

Özet

- **Anahtar Kelime:** Eksik Veri (Missing Data), Değer Atama (Imputation), İşaretleme (Flagging), Özellik Mühendisliği.
- **Strateji:**
    1. **Tespit Et:** Verindeki eksik değerleri (boşlukları) bul.
    2. **İşaretle (Flag):** Bu değişikliği yapacağını modele bildirmek için yeni bir "eksiklik" sütunu (_missing) oluştur.
    3. **Doldur (Impute):** Boşlukları, ortalama veya medyan gibi istatistiksel bir değerle doldurarak tablonu "tam" hale getir.
- **Neden Önemli?:** Bu iki adımlı yaklaşım, eksik verilerin modelin çalışmasını engellemesini önlerken, **"verinin eksik olması" durumunun kendisinin taşıdığı değerli bilgiyi de korumanızı ve modele sunmanızı** sağlar.

Senaryo: Müşteri Yaşı Verisini Temizleme

1. Sol Sütun (X1): Orijinal, Ham Veri

- **Ne Görüyoruz?:** Bir internal table'daki MUSTERI_YASI sütununu düşünün.
- **Değerler:** 10, 15, 7, 16, 1 gibi (burada 10-15-7 biraz anlamsız ama sayı olarak düşünelim) normal yaş verileri var.
- **Problem:** Dördüncü satırda bir **boşluk (—)** var. Bu bizim **Eksik Verimiz**. Muhtemelen müşteri kayıt olurken doğum tarihini girmeyi unutmuş veya sistem o anda bir hata verip kaydedememiş.

Neden bu boşluk bir problem?

Çoğu makine öğrenmesi algoritması, girdilerinde sayısal değerler bekler. NULL veya NaN (Not a Number) gibi bir değerle karşılaştıklarında hata verirler ve çalışmazlar. Bu boşluğu bir şekilde "doldurmak" zorundayız. Bu işleme **"Imputation"** denir.

---

2. Boşluğu Doldurma Stratejisi

- **Ne Yapıyoruz?:** En yaygın ve basit yöntemlerden biri, bu boşluğu sütunun **ortalaması, medyanı** veya **en sık tekrar eden değeri (mod)** ile doldurmaktır.
- **Uygulama:** Diyelim ki MUSTERI_YASI sütununun medyanı (tam ortadaki değer) **12**. O zaman o boş hücreye 12 yazarız. Artık tablomuzda hiç boşluk kalmadı.
    - Orijinal: 10, 15, 7, , 16, 1
    - Doldurulmuş: 10, 15, 7, 12, 16, 1

---

3. Sağ Sütun (X1_missing): Zekice ve Hayati Adım

- **Ne Görüyoruz?:** T (True) ve F (False) değerleri içeren yeni bir sütun. Bu, yine bir **"işaret bayrağı" (flag)**.
- **Anlamı:** Bu sütun, bizim veri üzerinde bir "operasyon" yaptığımızı kaydeder.
    - **F (False):** "Bu satırdaki yaş verisi orijinaldi, müşteri tarafından girilmişti."
    - **T (True):** "DİKKAT! Bu satırdaki 12 yaş verisi orijinal değil. Burada aslında bir boşluk vardı ve ben onu medyan değeri ile **doldurdum.**"

Bu yeni sütun NEDEN bu kadar önemlidir?

Çünkü **"verinin eksik olması"** durumunun kendisi, başlı başına bir **anlam taşıyor olabilir.** Az önceki "maaşını söylemeyenler" analojisini hatırlayın. Cevap vermemek, bir desenin parçasıydı.

Modelimiz bu yeni X1_missing sütununa bakarak şunu öğrenebilir:

"Hımmm, demek ki **yaş bilgisini girmeyen müşteriler**, genellikle uygulamamızı çok daha az kullanıyor veya kredi başvuruları daha sık reddediliyor."

Yani model, sadece müşterinin (tahmini) yaşını değil, **yaş bilgisini vermemiş olma durumunu** da önemli bir sinyal olarak kullanır. Bu, modelinize inanılmaz bir öngörü gücü katabilir.

Özetle, bu görsel bize şunu anlatıyor:

1. **Tespit Et:** Verindeki eksik değerleri (boşlukları) bul.
2. **Doldur (Impute):** Bu boşlukları, ortalama veya medyan gibi istatistiksel bir değerle doldurarak tablonu "tam" hale getir.
3. **İşaretle (Flag):** Bu doldurma işlemini yaptığın satırları unutmamak ve modele ek bilgi sağlamak için yeni bir "eksiklik" sütunu (X1_missing) oluştur.

Bu iki adımlı yaklaşım ("Doldur ve İşaretle"), eksik verilerle başa çıkmanın en sağlam ve en sık kullanılan yöntemlerinden biridir.

24

Bootstrap nedir?

Bootstrap, istatistik ve makine öğrenmesinde inanılmaz derecede zeki ve güçlü bir tekniktir.

Görsel 22: Bootstrap - "Tek Veri Setinden Binlerce Simülasyon Yaratmak"

Genel Anlam: Neyi Anlatıyor?

Bootstrap, elinizdeki **tek bir veri setini (örneklemi)** kullanarak, sanki elinizde binlerce farklı veri seti varmış gibi davranmanızı sağlayan bir **simülasyon tekniğidir.** Temel amacı, bir istatistiksel tahminin (örneğin ortalama, medyan veya bir model katsayısı) ne kadar güvenilir olduğunu ve ne kadar "oynak" olabileceğini (hata payını) ölçmektir.

Tek Cümlede: Tek bir veri setini, "kendi kendini kopyalayarak çoğaltma" yoluyla, bir tahminin güven aralığını hesaplama yöntemidir.

Python Dünyası ile Anlatım

Bootstrap'in temel mekanizması olan "yerine koyarak örnekleme"yi NumPy ile kolayca yapabiliriz.

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Elimizdeki tek ve küçük veri seti (7 müşterinin puanı)
orijinal_veri = np.array([7, 5, 2, 2, 5, 2, 4])

# Binlerce "sanal anket" (bootstrap örneği) oluşturalım
bootstrap_ortalamalari = []
n_simulasyon = 10000

for i in range(n_simulasyon):
    # 1. Sanal Örneklem Yarat (Yerine Koyarak Seçim)
    # orijinal_veri'den, orijinal_veri'nin boyutu kadar eleman seç,
    # ama her seçimden sonra elemanı havuza geri koy (replace=True).
    sanal_orneklem = np.random.choice(orijinal_veri, size=len(orijinal_veri),
replace=True)

    # 2. Sanal Örneklemin Ortalamasını Hesapla ve Sakla
    sanal_ortalama = sanal_orneklem.mean()
    bootstrap_ortalamalari.append(sanal_ortalama)

# 3. Sonuçları Analiz Et
# Güven aralığını hesaplayalım (%95'lik aralık)
guven_araligi_alt = np.percentile(bootstrap_ortalamalari, 2.5)
guven_araligi_ust = np.percentile(bootstrap_ortalamalari, 97.5)

print(f"Orijinal Ortalama: {orijinal_veri.mean():.2f}")
print(f"%95 Güven Aralığı: [{guven_araligi_alt:.2f}, {guven_araligi_ust:.2f}]")

# Ortalamaların dağılımını görselleştirelim
sns.histplot(bootstrap_ortalamalari, kde=True)
plt.title("Bootstrap ile Elde Edilen Ortalamaların Dağılımı")
plt.show()
```

Bu kod, sadece 7 veri noktasıyla, ortalamanın büyük olasılıkla hangi aralıkta yer alacağına dair istatistiksel olarak sağlam bir tahmin üretir.

Analojilerle Anlatım

1. Tek Bir Fotoğrafla Kalabalık Tahmini Yapma Analojisi

- **Problem:** Bir meydanın **tek bir fotoğrafına** bakarak, meydandaki insanların **ortalama boyunu** ve bu tahminin **hata payını** bulmanız isteniyor.
- **Bootstrap Süreci:**
    1. Fotoğraftaki 100 kişinin hepsini bir şapkaya atın.
    2. Şapkadan rastgele birini çekin, boyunu not alın ve kişiyi şapkaya **geri koyun.**
    3. Bu işlemi 100 kez tekrarlayarak **"sanal bir kalabalık fotoğrafı"** oluşturun. Bu yeni fotoğrafta bazıları hiç olmayacak, bazıları ise birden çok kez yer alacaktır.
    4. Bu sanal fotoğrafın ortalama boyunu hesaplayın.
    5. Bu süreci 10.000 kez tekrarlayarak elinizde **10.000 farklı ortalama boy** değeri olsun.
- **Sonuç:** Bu 10.000 ortalamanın dağılımına bakarak, "Ortalama boyun %95 ihtimalle 170 cm ile 180 cm arasında olduğunu tahmin ediyorum" diyebilirsiniz.

2. Bisküvi Üretimi Analojisi

- **Problem:** Elinizdeki **7 bisküvilik** test kutusuna bakarak, tüm günkü üretimin ortalama çilek sayısının güven aralığını bulmak.
- **Bootstrap Süreci:** 7 bisküviyi bir sepete koyup, her seferinde geri koyarak 7 kez çekiliş yaparsınız. Bu size **"sanal bir bisküvi kutusu"** verir. Bu işlemi binlerce kez tekrarlayarak binlerce farklı "sanal kutu ortalaması" elde edersiniz.
- **Sonuç:** "Patronum, simülasyonlarımıza göre ortalama çilek sayısının %95 ihtimalle 3 ile 5 arasında olmasını bekliyoruz" şeklinde güvenilir bir rapor sunarsınız.

3. Restoran Analojisi

- **Problem: 7 müşterinin** verdiği puanlara bakarak, yeni bir tatlının alacağı ortalama puanın ne kadar değişken olabileceğini tahmin etmek.
- **Bootstrap Süreci:** 7 puanı bir anket kutusuna atıp, her seferinde geri koyarak 7 kez çekiliş yaparsınız. Bu size **"sanal bir anket grubu"** verir. Bu işlemi binlerce kez tekrarlayarak binlerce farklı "sanal grup ortalaması" elde edersiniz.
- **Sonuç:** "Bu tatlının ortalama puanı %95 ihtimalle 3.5 ile 5.5 arasında gezinecek. Bu, kitlenin bir kısmının sevip bir kısmının sevmeyeceği, kutuplaştırıcı bir tatlı olduğunu gösteriyor" diyebilirsiniz.

Özet

- **Anahtar Kelime:** Bootstrap, Yerine Koyarak Örnekleme (Sampling with Replacement), Güven Aralığı (Confidence Interval), Standart Hata (Standard Error), Simülasyon.
- **Neden Önemli?:** Bootstrap, elimizde çok fazla veri olmasa bile, tek bir örneklemden istatistiksel olarak anlamlı ve güvenilir sonuçlar (özellikle tahminlerimizin belirsizliği hakkında) çıkarmamızı sağlayan, son derece güçlü ve esnek bir yöntemdir. **Random Forest** gibi modern algoritmaların temelini oluşturur.

Bootstrap Süreci Adım Adım

Bootstrap size der ki: "Madem elinde başka fotoğraf yok, o zaman elindeki tek fotoğrafı sanki **bütün evrenmiş gibi** kabul et ve ondan yeni, sahte fotoğraflar yarat!"

1. **"Yerine Koyarak Örnekleme" (Sampling with Replacement):**
    - Orijinal fotoğraftaki 100 kişiyi bir torbaya koyduğunuzu hayal edin.
    - Torbadan rastgele bir kişi seçin, boyunu not alın.
    - **En Kritik Adım:** Seçtiğiniz kişiyi torbaya **geri koyun.**
    - Bu işlemi 100 kere tekrarlayın.
2. **İlk Bootstrap Örneğini Yaratma:**
    - Bu 100 çekilişin sonunda elinizde 100 kişilik yeni, "sanal" bir fotoğraf oluşur. Bu yeni fotoğrafta, orijinaldeki bazı insanlar hiç olmayacak, bazıları ise şans eseri 2, 3 veya daha fazla kez yer alacaktır. Bu, sizin **ilk Bootstrap örneğinizdir.**
    - Bu yeni örneğin ortalama boyunu hesaplayın. Diyelim ki **172 cm** çıktı.
3. **Süreci Binlerce Kez Tekrarlama:**
    - Yukarıdaki iki adımı 1000, 5000 veya 10.000 kez daha tekrarlayın.
    - Her seferinde, birbirinden birazcık farklı, 100 kişilik yeni sanal fotoğraflar (bootstrap örnekleri) ve her biri için yeni bir ortalama boy hesaplamış olacaksınız.
4. **Sonuçların Analizi:**
    - Artık elinizde tek bir ortalama (175 cm) yerine, binlerce farklı ortalama var: [172 cm, 178 cm, 174 cm, 171 cm, ...]
    - Bu binlerce ortalamanın dağılımına bakarak şunları söyleyebilirsiniz:
        - "Hesapladığım ortalamaların %95'i 170 cm ile 180 cm arasında değişiyor." (Bu sizin **güven aralığınızdır**).
        - "Ortalamamın standart sapması 2.5 cm." (Bu sizin tahmininizin **standart hatasıdır**).

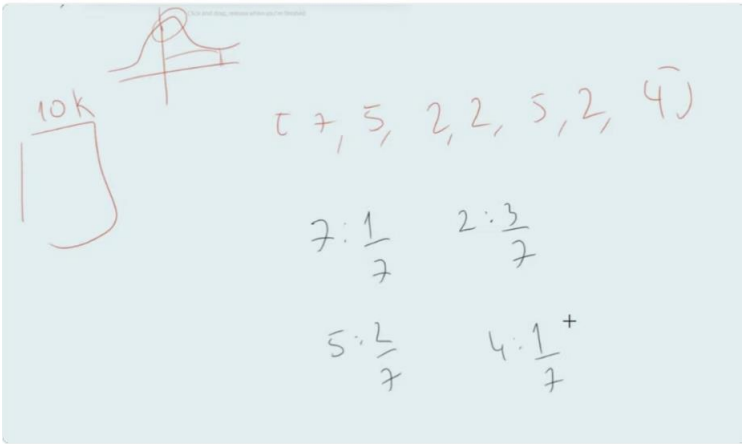
Özetle: Bootstrap, tek bir veri setini "kendi kendini kopyalayarak çoğaltma" tekniğiyle, bir istatistiğin (ortalama, medyan, model katsayısı vb.) ne kadar değişken ve güvenilir olduğunu anlamamızı sağlar.

Makine Öğrenmesindeki Yeri (Random Forest)

Bootstrap'in en ünlü kullanım alanı **Random Forest (Rastgele Orman)** algoritmasıdır.

- Random Forest, yüzlerce veya binlerce Karar Ağacı (Decision Tree) oluşturur.
- Her bir ağaç, orijinal veri setinin tamamı üzerinde değil, o veri setinden oluşturulmuş farklı bir **Bootstrap örneği** üzerinde eğitilir.
- Bu sayede her ağaç, verinin biraz farklı bir versiyonunu görerek farklı bir "uzmanlık" geliştirir.
- Nihai karar, bu yüzlerce farklı uzmanın (ağacın) oylarının ortalaması alınarak verilir.

25



Bu görsel, Bootstrap tekniğinin "motorunun" nasıl çalıştığını, yani simülasyonun arkasındaki matematiksel mantığı adım adım gösteriyor. Not defteriniz için bu süreci detaylandıralım.

Görsel 23: Bootstrap'in Mekaniği - Olasılıklarla Simülasyon Yaratmak

Genel Anlam: Neyi Anlatıyor?

Bu görsel, Bootstrap sürecinin üç temel aşamasını özetler:

1. **Orijinal Veri:** Elimizdeki küçük ve tek gerçeklik.
2. **Olasılık Dağılımı:** Bu gerçekliğin istatistiksel parmak izi veya "tarif defteri".
3. **Simülasyon:** Bu tarif defterini kullanarak binlerce "olası gelecek" senaryosu yaratma.

Adım 1: Orijinal Veri Seti (Sağ Üstteki Kırmızı Sayılar)

- **Ne Görüyoruz?:** [7, 5, 2, 2, 5, 2, 4]
- **Anlamı:** Bu, bizim elimizdeki tek somut veri setidir. Toplam 7 adet gözlemden oluşur. Bootstrap, bu veri setini tüm "evren" olarak kabul eder.

Adım 2: Ampirik Olasılık Dağılımını Hesaplama (Alttaki Siyah Yazılar)

- **Ne Görüyoruz?:** Her bir özgün değerin veri setindeki görülme olasılığı.

- **Anlamı:** Bu, "sanal veri yaratma makinemizin" kullanacağı kural setidir. Bootstrap, bu olasılıklara sadık kalarak rastgele seçimler yapacaktır.
    - **7'nin Olasılığı:** 1/7 (7 gözlemden 1'i 7'dir).
    - **5'in Olasılığı:** 2/7 (7 gözlemden 2'si 5'tir).
    - **2'nin Olasılığı:** 3/7 (7 gözlemden 3'ü 2'dir).
    - **4'ün Olasılığı:** 1/7 (7 gözlemden 1'i 4'tür).
- **Python Dünyası:** `np.random.choice()` fonksiyonu, p parametresi ile tam olarak bu olasılıklara göre seçim yapabilir.

Adım 3: Simülasyonu Başlatma (Sol Taraftaki "10k" Kutusu)

- **Ne Görüyoruz?:** Üzerinde 10k yazan bir kutu ve sol üstte bir çan eğrisi taslağı.
- **Anlamı:** Bu, "sanal veri seti yarat ve istatistiğini hesapla" sürecini **10.000 kez** tekrarlayacağımızı ifade eder.
    - Her bir iterasyonda, Adım 2'deki olasılıklara göre, "yerine koyarak" 7 tane yeni değer seçilir ve bir "bootstrap örneği" oluşturulur.
    - Her bir bootstrap örneğinin ortalaması hesaplanır.
- **Sonuç (Çan Eğrisi):** Bu 10.000 farklı ortalama değeri bir grafiğe döküldüğünde, Merkezi Limit Teoremi'ne uygun olarak bir **çan eğrisi (normal dağılım)** şeklini alırlar. Bu dağılım, orijinal verinin ortalamasının ne kadar "oynak" olabileceğini ve büyük olasılıkla hangi aralıkta yer alacağını bize gösterir.

Analojilerle Anlatım

1. Bisküvi Üretimi Analojisi

1. **Orijinal Veri:** Kutudaki 7 bisküvinin içindeki çilek sayıları: [7, 5, 2, 2, 5, 2, 4].
2. **Olasılık Dağılımı:** Bu kutuya göre, rastgele bir bisküvinin içinde 2 çilek olma olasılığı 3/7'dir.
3. **Simülasyon:** Bu olasılıkları kullanarak, her seferinde 7 bisküviden oluşan **10.000 farklı "sanal kutu"** yaratırız ve her birinin ortalama çilek sayısını kaydederiz. Bu 10.000 ortalamanın dağılımı, bize güvenilir bir tahmin aralığı sunar.

2. Restoran Analojisi

1. **Orijinal Veri:** İlk 7 müşterinin yeni tatlıya verdiği puanlar: [7, 5, 2, 2, 5, 2, 4].
2. **Olasılık Dağılımı:** Bu gruba göre, rastgele bir müşterinin 5 puan verme olasılığı 2/7'dir.
3. **Simülasyon:** Bu olasılıkları kullanarak, 7 kişilik **10.000 farklı "sanal müşteri grubu"** oluştururuz ve her bir grubun ortalama puanını hesaplarız. Bu 10.000 ortalamanın grafiği, tatlının genel beğenisinin ne kadar değişken olabileceğini gösterir.

Özet

- **Anahtar Kelime:** Ampirik Dağılım (Empirical Distribution), Simülasyon, Merkezi Limit Teoremi (Central Limit Theorem).
- **İş Akışı:**
    1. Orijinal küçük örneklemden **olasılıkları öğren.**
    2. Bu olasılıkları kullanarak binlerce kez **yeni sanal örneklemler yarat.**
    3. Bu binlerce örneklemin istatistiklerinin (örn: ortalamalarının) **dağılımını analiz et.**
- **Neden Önemli?:** Bu yöntem, tek bir küçük veri setinin ötesine geçerek, istatistiksel bir tahminin belirsizliğini ve güvenilirliğini ölçmemizi sağlar. Bize tek bir sayı değil, olası sonuçların bir aralığını sunar.

Görsel Ne Anlatıyor: Küçük Bir Gerçeklikten, Büyük Bir Simülasyon Yaratmak

Senaryo: Elinizde çok küçük bir veri setiniz var. Diyelim ki yeni bir ürünün ilk 7 müşterisinin yaptığı harcamalar bunlar: **[7 TL, 5 TL, 2 TL, 2 TL, 5 TL, 2 TL, 4 TL]**

Problem: Bu 7 müşteriye bakarak, gelecekteki **tüm** müşterilerin ortalama harcaması hakkında güvenilir bir tahmin yapabilir misiniz? Sadece bu 7 sayının ortalamasını almak çok riskli olur. Belki de şans eseri çok cömert veya çok cimri 7 müşteri size denk gelmiştir. Tahmininizin ne kadar "sağlam" olduğunu nasıl anlarsınız?

İşte bu görsel, Bootstrap'in bu sorunu nasıl çözdüğünü anlatıyor:

Adım 1: Elimizdeki "Evren"i Tanımlamak (Görseldeki Olasılık Hesabı)

Bootstrap der ki: "Madem elimde başka veri yok, o zaman benim bildiğim tek gerçek, tek evren bu 7 kişilik gruptur."

- Bu evrende, rastgele bir müşterinin **7 TL** harcama olasılığı nedir? 7 kişiden 1'i harcamış, yani **1/7**.
- **5 TL** harcama olasılığı? 7 kişiden 2'si harcamış, yani **2/7**.
- **2 TL** harcama olasılığı? 7 kişiden 3'ü harcamış, yani **3/7**.
- **4 TL** harcama olasılığı? 7 kişiden 1'i harcamış, yani **1/7**.

Görseldeki alt kısım, tam olarak bu **olasılık dağılımını** hesaplıyor. Bu, bizim "sanal müşteri yaratma makinemizin" kullanacağı tarif defteridir.

Adım 2: Sanal Dünyalar Yaratmak (Görseldeki "10k" Kutusu)

Şimdi en sihirli kısma geldik. Bu tarif defterini kullanarak, sanki elimizde binlerce müşterimiz varmış gibi simülasyonlar yapacağız.

- **Bir "Bootstrap Örneği" Yaratmak:**
    1. Bu 7 müşteriyi bir şapkaya koyduğumuzu düşünün.
    2. Şapkadan rastgele birini çekin, harcamasını not alın (diyelim **5 TL** geldi).
    3. O kişiyi şapkaya **geri koyun.** (En önemli adım!)
    4. Bu işlemi 7 kere tekrarlayın.

Sonuçta elinizde şöyle **yeni, sanal bir 7 kişilik müşteri grubu** olabilir:

[5 TL, 2 TL, 2 TL, 7 TL, 4 TL, 2 TL, 5 TL]

(Dikkat ederseniz, orijinal listedeki bazıları hiç gelmedi, bazıları ise birden çok kez geldi).

- **Bu Sanal Grubun Ortalamasını Hesaplayın.**
- **Süreci 10.000 Kez Tekrarlamak ("10k" Kutusu):**
  İşte görseldeki **"10k"** kutusu burada devreye giriyor. Bu "yeni sanal grup yarat ve ortalamasını al" işlemini tam **10.000 kere** tekrarlarsınız.

Adım 3: Sonuçları Analiz Etmek (Görseldeki Çan Eğrisi)

Bu 10.000 tekrardan sonra elinizde ne olur? Tek bir ortalama yerine, **10.000 farklı ortalama değeri!**

[Ortalama_1, Ortalama_2, Ortalama_3, ..., Ortalama_10000]

Bu binlerce ortalamayı bir grafiğe döktüğünüzde, görselin sol üst köşesindeki gibi bir **Çan Eğrisi (Normal Dağılım)** elde edersiniz.

Bu grafik size ne söyler?

- "Hesapladığım 10.000 ortalamanın %95'i 3.5 TL ile 5.5 TL arasında çıktı."
  - Bu, size tahmininiz için bir **güven aralığı** verir. Artık patronunuza gidip "Ortalama harcama 4.2 TL" demek yerine, çok daha profesyonel bir şekilde "Ortalama harcamanın %95 ihtimalle 3.5 TL ile 5.5 TL arasında olacağını tahmin ediyoruz." diyebilirsiniz.

Özetle, bu görsel bize Bootstrap'in şu adımlarını gösteriyor:

1. Orijinal küçük verinin olasılık dağılımını **hesapla.**
2. Bu olasılıklara göre, "yerine koyarak" binlerce kez yeni, sanal veri setleri **yarat.** ("10k")
3. Bu binlerce sanal setin sonuçlarını analiz ederek, tek bir tahmine ne kadar **güvenebileceğini** anla. (Çan Eğrisi)

Senaryo: Yeni Çıkan Bir Bisküvi İçin Kalite Testi

Fabrikanız yepyeni, "Süper-Çilekli" bir bisküvi üretti. Ama henüz seri üretime geçmediniz. Ar-Ge departmanı size test etmeniz için sadece **küçücük bir kutu** verdi. Kutunun içinde **sadece 7 tane bisküvi** var.

Problem: Patronunuz geldi ve size sordu: "Bu yeni bisküvilerin içindeki **ortalama çilek parçacığı sayısı** nedir? Ve bu ortalamaya ne kadar güvenebiliriz? Eğer standartlara uyuyorsa yarın seri üretime geçeceğiz!"

Siz de kutudaki 7 bisküviyi tek tek inceleyip içlerindeki çilek parçacıklarını saydınız. Sonuçlar şunlar:

[7, 5, 2, 2, 5, 2, 4] adet çilek parçacığı.

Ne yaparsınız? Sadece bu 7 bisküvinin ortalamasını alıp (7+5+2+2+5+2+4)/7 = 3.85 demek çok riskli. Belki de şansınıza size ya çok dolu ya da çok boş bir kutu denk geldi. Patronunuza daha güvenilir bir cevap vermeniz lazım.

İşte burada **Bootstrap** devreye giriyor.

Bootstrap ile "Sanal Fabrika" Simülasyonu

Bootstrap size der ki: "Madem elinde başka kutu yok, o zaman bu 7 bisküvi sanki **tüm fabrikanın üretebileceği tek bisküvi çeşitleriymiş gibi** davran."

Adım 1: "Tarif Defteri"ni Oluşturmak (Olasılıkları Hesaplamak)

- Bu 7 bisküviye göre, fabrikadan rastgele bir bisküvi çektiğimde içinde;
    - **7** çilek parçacığı olma olasılığı: **1/7**
    - **5** çilek parçacığı olma olasılığı: **2/7**
    - **2** çilek parçacığı olma olasılığı: **3/7**
    - **4** çilek parçacığı olma olasılığı: **1/7**

Bu, bizim "sanal bisküvi üretme makinemizin" kullanacağı tariftir.

Adım 2: "Sanal Bisküvi Kutuları" Yaratmak (Bootstrap Örnekleri)

Şimdi, bu 7 bisküviyi bir sepete koyduğunuzu hayal edin ve sihirli bir simülasyon başlatın:

1. Sepetten rastgele bir bisküvi çekin, içindeki çilek sayısını not alın (diyelim **5** geldi).
2. **En Önemli Kısım:** O bisküviyi **sepete geri koyun!** (Böylece tekrar seçilme şansı olur).
3. Bu işlemi 7 kere tekrarlayın.

Bu işlemin sonunda elinizde 7 bisküvilik **yepyeni, "sanal" bir bisküvi kutusu** olur. Belki şöyle bir şey:

[5, 2, 2, 7, 4, 2, 5]

Bu sizin **ilk sanal kutunuz.** Bu kutunun ortalama çilek sayısını hesaplayın: 3.85.

Adım 3: Simülasyonu 10.000 Kez Tekrarlamak

Şimdi bu "sanal kutu yarat ve ortalamasını al" işlemini tam **10.000 kere** yapıyorsunuz.

Sanki fabrikanız 10.000 farklı günde üretim yapmış ve siz her gün rastgele bir kutu alıp ortalamasını hesaplamışsınız gibi dev bir simülasyon yaratmış olursunuz!

Adım 4: Sonuçları Değerlendirmek

Artık elinizde tek bir ortalama (3.85) yerine, **10.000 farklı "günlük ortalama"** var.

[3.85, 4.14, 3.57, ...]

Bu 10.000 ortalamaya bakarak patronunuza çok daha profesyonel bir cevap verebilirsiniz:

"Patronum, yaptığım 10.000 farklı simülasyonun sonucuna göre, bisküvilerdeki ortalama çilek parçacığı sayısının **%95 ihtimalle 3 ile 5 arasında** olacağını öngörüyorum. Bu sonuçlar, kalite standartlarımıza uyuyor. Seri üretime geçebiliriz!"

Özetle: Bootstrap, elinizdeki **küçücük bir gerçeklik parçasını (7 bisküvi)** alıp, ondan binlerce **"olası gelecek senaryosu" (10.000 sanal kutu)** yaratarak, yaptığınız tahminin ne kadar sağlam ve güvenilir olduğunu size söyleyen bir simülasyon tekniğidir.

Senaryo: Yeni İmza Tatlının Puanlaması

Restoranınızın baş aşçısı, yeni bir imza tatlı yarattı: "Altın Parçacıklı Volkanik Çikolatalı Sufle". Bu tatlıyı menüye koyup koymamakta kararsızsınız.

Problem: Tatlının beğenilip beğenilmeyeceğini anlamak için, restoranınıza gelen ilk **7 müşteriye** ücretsiz olarak ikram edip 1'den 10'a kadar puan vermelerini istediniz.

Aldığınız puanlar şunlar: **[7, 5, 2, 2, 5, 2, 4]**. (Görünüşe göre bazıları bayılmış, bazıları hiç sevmemiş!)

Şimdi, sadece bu 7 kişiye bakarak bu tatlının gelecekteki **tüm müşterilerden alacağı ortalama puanı** güvenilir bir şekilde tahmin edebilir misiniz? Bu çok riskli. Belki de o gün şansınıza ya tatlı sevmeyen ya da her şeye bayılan insanlar denk geldi. Yatırımcınıza sunmak için daha sağlam bir veriye ihtiyacınız var.

İşte **Bootstrap** ile "sanal müşteri anketleri" yapıyoruz.

Bootstrap ile "Sanal Müşteri Yorumları" Simülasyonu

Bootstrap der ki: "Madem elinde başka müşteri yok, o zaman bu 7 müşteri sanki senin **tüm potansiyel müşteri kitleni** temsil ediyormuş gibi davran."

Adım 1: "Müşteri Profili Defteri"ni Oluşturmak (Olasılıkları Hesaplamak)

- Bu 7 müşteriye göre, restoranınıza gelen rastgele bir müşterinin bu tatlıya;
    - **7 puan** verme olasılığı: **1/7**
    - **5 puan** verme olasılığı: **2/7**
    - **2 puan** verme olasılığı: **3/7**
    - **4 puan** verme olasılığı: **1/7**

Bu, bizim "sanal müşteri anketi oluşturma makinemizin" kullanacağı müşteri profilleridir.

Adım 2: "Sanal Anket Grupları" Yaratmak (Bootstrap Örnekleri)

Şimdi, bu 7 müşterinin verdiği puanları bir anket kutusuna attığınızı düşünün ve sihirli bir simülasyon başlatın:

1. Anket kutusundan rastgele bir puan çekin, not alın (diyelim **2 puan** geldi).
2. **En Kritik Adım:** O puan kağıdını **kutunun içine geri atın!** (Böylece aynı fikirdeki başka bir müşterinin gelme ihtimalini simüle edersiniz).
3. Bu işlemi 7 kere tekrarlayın.

Bu işlemin sonunda elinizde 7 kişilik **yepyeni, "sanal" bir müşteri grubunun** verdiği puanlar olur. Belki şöyle bir şey:

[2, 5, 2, 2, 7, 5, 4]

Bu sizin **ilk sanal anketiniz.** Bu anketin ortalama puanını hesaplayın.

Adım 3: Simülasyonu 10.000 Kez Tekrarlamak

Şimdi bu "sanal anket yarat ve ortalamasını al" işlemini tam **10.000 kere** yapıyorsunuz.

Sanki restoranınız 10.000 farklı günde açılmış ve her gün rastgele 7 kişilik bir grup gelip tatlınızı puanlamış gibi devasa bir simülasyon yapmış olursunuz.

Adım 4: Sonuçları Değerlendirmek

Artık elinizde tek bir ortalama puan yerine, **10.000 farklı "potansiyel ortalama puan"** var.

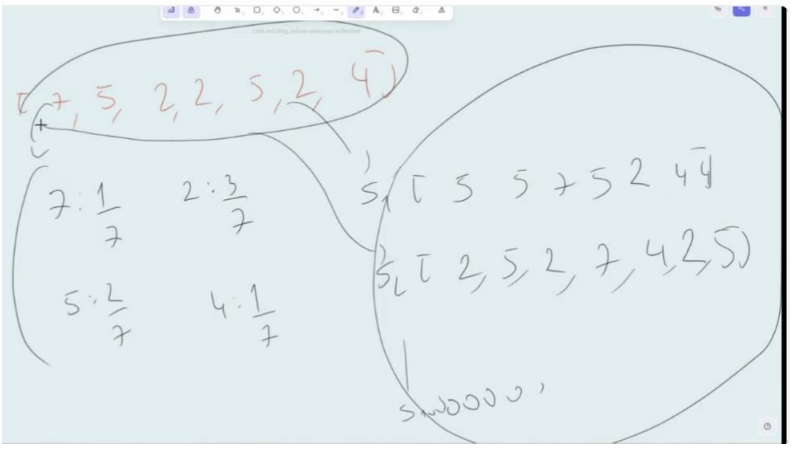
[4.2, 3.8, 5.1, ...]

Bu 10.000 farklı ortalama puana bakarak yatırımcınıza çok daha güçlü bir sunum yapabilirsiniz:

"Yaptığımız 10.000 farklı müşteri grubu simülasyonuna göre, yeni tatlımızın alacağı ortalama puanın **%95 ihtimalle 3.5 ile 5.5 arasında** olacağını öngörüyoruz. Bu sonuç, tatlının riskli ve kutuplaştırıcı bir lezzete sahip olduğunu gösteriyor. Menüye eklemeden önce tarifte bir revizyon yapmayı öneriyorum."

Özetle: Bootstrap, elinizdeki **küçücük bir anket sonucunu (7 müşteri)** alıp, ondan binlerce **"olası pazar tepkisi senaryosu" (10.000 sanal anket)** yaratarak, verdiğiniz iş kararının ne kadar sağlam ve güvenilir olduğunu size söyleyen bir simülasyon tekniğidir.

25



Bu görsel, Bootstrap tekniğinin nasıl çalıştığını çok somut bir şekilde gösteriyor. Bir önceki görselin teorik anlatımını, "işte böyle yapılıyor" diyerek pratiğe döküyor.

Görsel 24: Bootstrap Örneklemesi - "Sanal Paralel Evrenler Yaratmak"

Genel Anlam: Neyi Anlatıyor?

Bu görsel, Bootstrap sürecinin kalbini oluşturan **"yerine koyarak örnekleme (sampling with replacement)"** işleminin nasıl gerçekleştirildiğini ve bunun sonucunda orijinal veri setinden nasıl farklı "sanal" veri setleri (bootstrap örnekleri) türetildiğini gösterir.

Görselin Adım Adım Anatomisi

1. Sol Üst Balon: Orijinal Evren (Gerçeklik)

- **Ne Görüyoruz?:** [7, 5, 2, 2, 5, 2, 4]
- **Anlamı:** Bu, bizim elimizdeki tek somut, gerçek veri setidir. Toplam 7 gözlemden oluşur.

2. Sol Alt Balon: Evrenin Kuralları (Olasılık Dağılımı)

- **Ne Görüyoruz?:** Her bir değerin görülme olasılığı (7: 1/7, 5: 2/7 vb.).
- **Anlamı:** Bu, "sanal evren yaratma makinemizin" kullanacağı tarif defteridir. Rastgele seçimler bu olasılıklara göre yapılacaktır.

3. Sağdaki Büyük Balon: Yaratılan Sanal Evrenler (Bootstrap Örnekleri)

- **Ne Görüyoruz?:** S1, S2, ..., S10000 (Sample 1, Sample 2...) ile başlayan, orijinalinden türetilmiş yeni listeler.
- **Anlamı:** İşte simülasyonun başladığı yer burasıdır. Her bir satır, orijinal evrenden "yerine koyarak" 7 kez rastgele seçim yapılarak oluşturulmuş yeni bir "paralel evreni" temsil eder.
    - **S1: [5, 5, 5, 2, 4, 4, ?]** (Bir eleman eksik çizilmiş)
        - **Yorum:** Bu ilk sanal evrende, orijinal evrendeki 5 değeri şans eseri 3 kez, 4 değeri 2 kez, 2 değeri ise 1 kez seçilmiştir. 7 değeri ise bu sanal evrende hiç yer almamıştır. Bu tamamen normaldir ve "yerine koyarak örnekleme"nin bir sonucudur.
    - **S2: [2, 5, 2, 7, 4, 2, 5]**
        - **Yorum:** Bu ikinci sanal evren, bir öncekinden tamamen farklı bir kompozisyona sahiptir. Orijinal evrendeki 2 değeri bu sefer 3 kez, 5 değeri 2 kez, 7 ve 4 ise birer kez seçilmiştir.
- **Oklar:** Oklar, bu sanal evrenlerin, orijinal evrenin olasılık kurallarına göre nasıl türetildiğini gösterir.

Python Dünyası ile Anlatım

Bu görseldeki S1 ve S2'nin nasıl yaratıldığını Python'da görelim:

```
import numpy as np

# 1. Orijinal Evren
orijinal_evren = np.array([7, 5, 2, 2, 5, 2, 4])

# 2. Sanal Evrenleri Yaratma
np.random.seed(0) # Sonuçların her seferinde aynı olması için
sanal_evren_1 = np.random.choice(orijinal_evren, size=7, replace=True)
sanal_evren_2 = np.random.choice(orijinal_evren, size=7, replace=True)

print(f"Orijinal Evren (Gerçeklik): {orijinal_evren}")
print("-" * 40)
print(f"Sanal Evren 1 (S1): {sanal_evren_1}")
print(f"Sanal Evren 2 (S2): {sanal_evren_2}")

# Çıktı (örneksel):
# Orijinal Evren (Gerçeklik): [7 5 2 2 5 2 4]
# ----------------------------------------
# Sanal Evren 1 (S1): [4 7 2 5 2 5 2]
# Sanal Evren 2 (S2): [7 2 2 4 5 5 2]
```

Gördüğünüz gibi, her np.random.choice çağrısı, orijinal evrenin elemanlarını kullanarak, ancak farklı kompozisyonlarda yeni, sanal evrenler yaratır.

Analojilerle Anlatım

1. Bisküvi Üretimi Analojisi

- **Orijinal Evren:** 7 bisküvilik test kutusu.
- **Sanal Evrenler (S1, S2...):** 7 bisküviyi bir sepete koyup, her seferinde geri koyarak 7 kez çekiliş yaparak oluşturulan **"sanal bisküvi kutuları"**. Her bir sanal kutu, fabrikanın o gün üretebileceği "olası" bir kutu kompozisyonunu temsil eder.

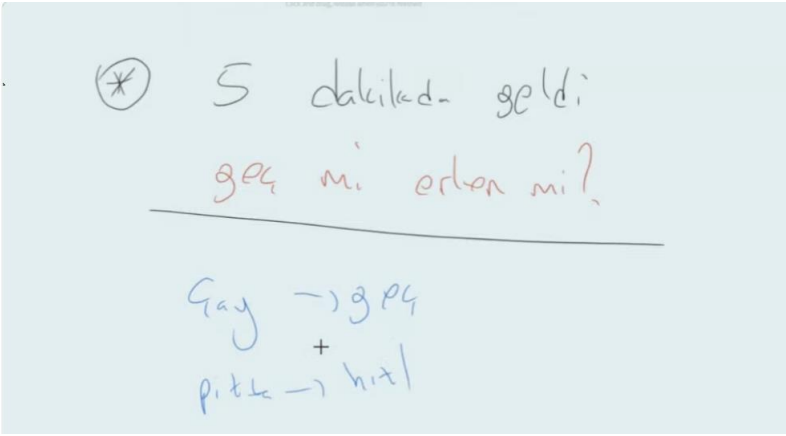
2. Restoran Analojisi

- **Orijinal Evren:** 7 müşterinin verdiği gerçek puanlar.
- **Sanal Evrenler (S1, S2...):** 7 puanı bir anket kutusuna atıp, her seferinde geri koyarak 7 kez çekiliş yaparak oluşturulan **"sanal anket grupları"**. Her bir sanal grup, restorana gelebilecek "olası" bir müşteri grubunun tepkisini simüle eder.

Özet

- **Anahtar Kelime:** Bootstrap Örneği (Bootstrap Sample), Yerine Koyarak Örnekleme (Sampling with Replacement), Simülasyon.
- **İş Akışı:** Orijinal veri setinden, elemanları her seçimden sonra geri koyarak, orijinaliyle aynı boyutta yeni ve sanal veri setleri oluşturma işlemidir.
- **Neden Önemli?:** Bu yöntem, tek bir veri setinden binlerce farklı "olası senaryo" türetmemizi sağlar. Bu senaryoların istatistiklerini analiz ederek, orijinal tahminimizin ne kadar sağlam ve güvenilir olduğu hakkında fikir sahibi oluruz.

26



Çay Pizza bu ikisini nasıl bir birime indirgeyebiliriz.

Bu görsel, veri ön işlemenin en temel ve en önemli adımlarından biri olan **Özellik Ölçeklendirme (Feature Scaling)** ihtiyacını ve arkasındaki mantığı çok basit ve etkili bir örnekle anlatıyor.

Görsel 25: Bağlamın Gücü ve Standardizasyon İhtiyacı

Genel Anlam: Neyi Anlatıyor?

Bu görsel, ham verilerin (raw data) tek başlarına bir anlam ifade etmediğini ve onları anlamlı kılmak için **bağlama** ihtiyaç duyduklarını gösterir. Makine öğrenmesi modellerinin farklı ölçeklerdeki ("farklı dillerde konuşan") verileri adil bir şekilde karşılaştırabilmesi için, bu verileri ortak bir birime indirgememiz gerekir.

- **"5 dakikada geldi. Geç mi erken mi?"**: Bu soru, "5" sayısının tek başına yetersiz olduğunu vurgular.
- **Çay -> Geç**: 5 dakikanın "çay" bağlamındaki anlamı.
- **Pizza -> Hızlı**: 5 dakikanın "pizza" bağlamındaki anlamı.

Python Dünyası ile Anlatım

Farklı ölçeklerdeki özellikleri olan bir veri setimiz olduğunu ve bir modelin bu özelliklere nasıl farklı "ağırlıklar" verebileceğini görelim.

```
import pandas as pd
from sklearn.preprocessing import StandardScaler

data = {'urun_tipi': ['Cay', 'Pizza', 'Cay', 'Pizza'],
        'teslim_suresi_dk': [5, 25, 2, 30],
        'musteri_memnuniyeti': [2, 9, 8, 8]} # 1-10 arası puan
df = pd.DataFrame(data)

# Ham veride, 'teslim_suresi_dk' (5-30) sütunu, 'urun_tipi' (kategorik)
# ve 'musteri_memnuniyeti' (1-10) sütunlarından çok daha büyük bir sayısal aralığa sahip.
# Bazı modeller (örn: K-Means, SVM) bu büyük sayılardan aşırı etkilenebilir.

# ---- ÇÖZÜM: Standardizasyon ----
# Her bir ürün tipi için kendi "normal"ini belirleyip ona göre ölçeklendirelim.

# Sadece Çay verisini alalım
cay_df = df[df['urun_tipi'] == 'Cay']
cay_ortalama = cay_df['teslim_suresi_dk'].mean() # Ortalama: 3.5
cay_std = cay_df['teslim_suresi_dk'].std()     # Std Sapma: 2.12

# 5 dakikada gelen çayın Z-Skoru'nu hesaplayalım
z_skor_cay = (5 - cay_ortalama) / cay_std
print(f"5 dakikada gelen çayın Z-Skoru: {z_skor_cay:.2f} (Pozitif = Ortalamadan GEÇ)")

# Sadece Pizza verisini alalım
pizza_df = df[df['urun_tipi'] == 'Pizza']
pizza_ortalama = pizza_df['teslim_suresi_dk'].mean() # Ortalama: 27.5
pizza_std = pizza_df['teslim_suresi_dk'].std()     # Std Sapma: 3.53

# 5 dakikada gelen bir pizzanın (hayali) Z-Skoru'nu hesaplayalım
z_skor_pizza = (5 - pizza_ortalama) / pizza_std
print(f"5 dakikada gelen pizzanın Z-Skoru: {z_skor_pizza:.2f} (Negatif = Ortalamadan ERKEN)")

# Çıktı:
# 5 dakikada gelen çayın Z-Skoru: 0.71
# 5 dakikada gelen pizzanın Z-Skoru: -6.37
```

scikit-learn'in StandardScaler'ı bu işlemi tüm veri seti için otomatik olarak yapar. Sonuç olarak, ham "5 dakika" değeri yerine, modelimize 0.71 ("biraz geç") ve -6.37 ("inanılmaz hızlı") gibi **bağlamı içinde taşıyan, evrensel** değerler vermiş oluruz.

Analojilerle Anlatım

1. Çay ve Pizza Analojisi

- **Problem:** "5 dakika" ne anlama geliyor?
- **Bağlam:** Bir **çay** için ortalama teslim süresi 2 dakikadır. 5 dakika, bu ortalamanın oldukça üzerindedir ve **gecikme** olarak algılanır.
- **Bağlam:** Bir **pizza** için ortalama teslim süresi 25 dakikadır. 5 dakika, bu ortalamanın çok altındadır ve **mucizevi bir hız** olarak algılanır.
- **Ortak Birim:** Her iki olayı da "kendi normallerinden ne kadar saptıkları" cinsinden ölçebiliriz. Buna **Z-Skoru** denir. 5 dakikalık çay belki +2'lik bir Z-skoruna (2 standart sapma geç), 5 dakikalık pizza ise -4'lük bir Z-skoruna (4 standart sapma erken) sahip olabilir. Artık +2 ve -4 karşılaştırılabilir değerlerdir.

2. Bisküvi Üretimi Analojisi

- **Problem:** İki farklı ölçümümüz var: "Fırın sıcaklığı 200 derece", "Bisküvi ağırlığı 10 gram". Bu iki sayıyı nasıl karşılaştırabiliriz?
- **Bağlam:** Fırın sıcaklığı normalde 190-210 derece arasında gezer. 200 derece çok **normaldir**.
- **Bağlam:** Bisküvi ağırlığı normalde 9.8-10.2 gram arasında olmalıdır. 10 gram çok **normaldir**.
- **Ortak Birim:** Her iki ölçümü de kendi normal aralıklarına göre standardize ettiğimizde, ikisinin de Z-skoru 0'a çok yakın çıkar. Model, her iki durumun da "beklenen" bir durum olduğunu anlar.

3. Restoran Analojisi

- **Problem:** İki müşterimiz var. Biri **25 yaşında**, diğeri **hesaba 500 TL** ödedi. Hangi müşteri daha "anormal"?
- **Bağlam:** Restoranımızın müşteri kitlesinin yaş ortalaması 28, standart sapması 3'tür. 25 yaş, ortalamaya çok **yakındır**.
- **Bağlam:** Ortalama hesap 150 TL, standart sapması 50 TL'dir. 500 TL'lik hesap, ortalamadan çok **uzaktır**.
- **Ortak Birim:** Yaşın Z-skoru -1 iken, hesabın Z-skoru +7 olabilir. Model, asıl "anormal" durumun harcama miktarında olduğunu anlar ve tahminlerini buna göre yapar.

Özet

- **Anahtar Kelime:** Özellik Ölçeklendirme (Feature Scaling), Standardizasyon (Standardization), Normalizasyon (Normalization), Z-Skoru.
- **Kural:** Farklı ölçeklerde (örn: yaş 20-70, gelir 10.000-500.000) özelliklere sahip verileri bir modele vermeden önce, bu özellikleri ortak bir ölçeğe getirmek, çoğu modelin (özellikle uzaklık tabanlı olanların) performansını önemli ölçüde artırır.

- **Neden Önemli?:** Standardizasyon, modelin büyük sayılara sahip özelliklere istemeden daha fazla "önem" vermesini engeller ve her özelliğin adil bir şekilde değerlendirilmesini sağlar. Farklı birimlerdeki "elmaları ve armutları", "anormallik puanı" gibi ortak bir birimde birleştirir.

Görsel Ne Anlatıyor: Farklı Dillerde Konuşan Veriler

1. Temel Soru:

- **"5 dakikada geldi. Geç mi erken mi?"**
- **Anlamı:** Tek başına "5" sayısı hiçbir anlam ifade etmez. Onu anlamlı kılan şey **bağlamdır.** Nereden bahsediyoruz?
    - Bir **çay** siparişi için 5 dakika **geçtir.**
    - Bir **pizza** siparişi için 5 dakika **mucizevi bir şekilde erkendir.**

2. Asıl Problem (Mavi Çizimler):

- **Çay -> Geç**
- **Pizza -> Hızlı**
- **Anlamı:** Elimizde bir veri seti olduğunu hayal edelim. Bir sütunda **sipariş türü (Çay/Pizza)**, diğer sütunda **teslimat süresi (dakika)** var.
- Modelimize bu iki sütunu doğrudan verirsek, model kafası karışır. Çünkü "dakika" biriminin anlamı, "sipariş türü"ne göre tamamen değişiyor. Model, "5 dakika"nın iyi mi kötü mü olduğuna tek başına karar veremez.

Çözüm: Verileri Aynı Dilde Konuşturmak

Sorunuz mükemmel: "Bu ikisini nasıl bir birime indirgeyebiliriz?" İşte profesyonel çözümler:

Yöntem 1: Standardizasyon (Z-Skoru)

Bu en yaygın ve en güçlü yöntemdir. Her bir veriyi kendi grubunun "normaline" göre yeniden ölçeklendiririz.

- **Adım 1: Her Grubun Kendi "Normal"ini Bul.**
    - Tüm **çay** siparişlerinin ortalama teslimat süresini ve ne kadar değişkenlik gösterdiğini (standart sapmasını) hesaplarız. Diyelim ki **Ortalama: 2 dakika, Standart Sapma: 1 dakika.**
    - Tüm **pizza** siparişlerinin ortalama teslimat süresini ve standart sapmasını hesaplarız. Diyelim ki **Ortalama: 25 dakika, Standart Sapma: 5 dakika.**
- **Adım 2: Her Bir Veriyi Kendi "Normal"ine Göre Puanla.**
    - **5 dakikada gelen çay:**
        - Hesap: (5 - 2) / 1 = +3
        - **Yeni Değer: +3**. Bunun anlamı: "Bu çay, normal bir çaydan **3 standart sapma daha geç** geldi. Bu çok anormal bir gecikme!"
    - **5 dakikada gelen pizza:**
        - Hesap: (5 - 25) / 5 = -4
        - **Yeni Değer: -4**. Bunun anlamı: "Bu pizza, normal bir pizzadan **4 standart sapma daha erken** geldi. Bu mucizevi bir hız!"

- **Sonuç:** Artık elimizde "dakika" birimi yok. Onun yerine, her bir siparişin kendi kategorisindeki normalliğini veya anormalliğini gösteren, **evrensel bir "anormallik puanı" (Z-skoru)** var. Model artık +3'ün kötü, -4'ün ise çok iyi olduğunu kolayca anlayabilir. İki farklı şeyi aynı birime indirgemiş olduk.

Yöntem 2: Normalizasyon (Min-Max Ölçekleme)

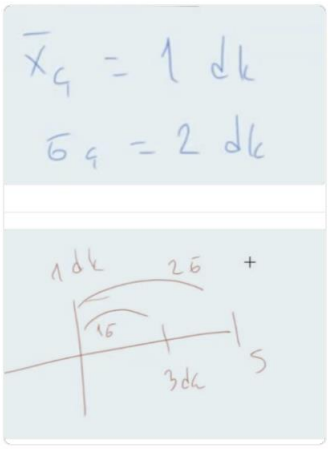
Bu daha basit bir yöntemdir. Her bir veriyi 0 ile 1 arasında bir değere sıkıştırırız.

- **Adım 1: Her Grubun Sınırlarını Bul.**
    - **Çay:** En hızlı çay 1 dakikada, en yavaş çay 8 dakikada geliyor.
    - **Pizza:** En hızlı pizza 15 dakikada, en yavaş pizza 50 dakikada geliyor.
- **Adım 2: Her Bir Veriyi Kendi Sınırlarına Göre Oranla.**
    - **5 dakikada gelen çay:** Bu, 1-8 aralığında oldukça yavaş bir değere denk gelir. Yeni değeri belki de **0.80** olur.
    - **5 dakikada gelen pizza:** Bu, 15-50 aralığında mümkün olmayan bir değerdir (veri hatası veya istisna). Diyelim ki 15 dakikada gelseydi, bu en hızlı olduğu için yeni değeri **0.0** olurdu.

Özetle:

Çay ve pizzayı aynı birime indirgemek için onları orijinal "dakika" değerlerinden kurtarıp, **kendi grupları içindeki göreceli performanslarını** yansıtan yeni, evrensel birimlere dönüştürürüz. En popüler yöntem, her verinin kendi ortalamasından ne kadar "uzak" olduğunu ölçen **Standardizasyon (Z-skoru)**'dur. Bu sayede model, elma ile armutu değil, "anormallik puanlarını" karşılaştırır.

27



Bu görseller, bir önceki teorik konsept olan **Standardizasyon (Z-Skoru)**'nun nasıl hesaplandığını pratik bir örnek üzerinden görselleştiriyor.

Görsel 26: Z-Skoru Hesaplanması - Bir Değeri "Anlamlandırmak"

Genel Anlam: Neyi Anlatıyor?

Bu iki görsel birlikte, tek bir veri noktasının ("5 dakikada gelen çay") ait olduğu grubun istatistiksel özelliklerine (ortalama ve standart sapma) göre nasıl "puanlandırıldığını" anlatır. Amaç, ham değeri, o grubun "normali"ne göre ne kadar anormal olduğunu ifade eden evrensel bir skora dönüştürmektir.

Görsel 1: Referans Noktası - "Çay Evreninin Kuralları"

- **$\bar{x}$_çay = 1 dk**:
    - **Anlamı:** Bu restorandaki çay siparişlerinin **ortalama** teslim süresi 1 dakikadır. Bu bizim beklentimiz, referans noktamızdır.
- **σ_çay = 2 dk**:
    - **Anlamı:** Çay teslimat sürelerindeki **tipik yayılma veya sapma** 2 dakikadır. Yani çayların çoğunun 1 ± 2 dakika aralığında gelmesi beklenir. Bu, "normal" kabul edilen oynaklık miktarıdır.

Görsel 2: Olayın Değerlendirilmesi - "Ne Kadar Anormal?"

- **Sayı Doğrusu:** Olayları yerleştirdiğimiz bir zaman çizelgesi.
- **1 dk:** Merkez noktamız, yani ortalama.
- **3 dk:** Ortalamadan **bir standart sapma (1σ)** uzaktaki nokta (1 + 2 = 3). Buraya kadar olan gecikmeler oldukça normal kabul edilebilir.
- **5 dk:** Bizim analiz ettiğimiz olay.
- **Hesaplama (Z-Skoru):** Bu görsel, "5 dakikanın, ortalama olan 1 dakikadan kaç 'standart sapma birimi' uzakta olduğunu" hesaplar.
    1. **Farkı Bul:** 5 (gözlem) - 1 (ortalama) = 4 dakika fark var.
    2. **Standart Sapmaya Böl:** 4 (fark) / 2 (standart sapma) = 2
- **Sonuç:** Z-Skoru = +2.

Python Dünyası ile Anlatım

```
import numpy as np

# Çay Evreninin Kuralları
cay_ortalama = 1
cay_std = 2

# Gözlemlenen Olay
gozlem = 5

# Z-Skoru Hesaplanması
z_skoru = (gozlem - cay_ortalama) / cay_std

print(f"5 dakikada gelen çayın Z-Skoru: {z_skoru}")
# Çıktı: 5 dakikada gelen çayın Z-Skoru: 2.0
```

Bu +2.0 değeri, artık "dakika" biriminden arındırılmıştır. Bu, "normalden 2 birim sapma" anlamına gelen evrensel bir puandır.

Analojilerle Anlatım

1. Sınav Notu Analojisi

- **Olay:** Bir öğrenci sınavdan **70** aldı. Bu iyi mi kötü mü?
- **Referans Noktası:** Sınıfın **ortalama notu 50**, **standart sapması 10**.
- **Hesaplama:** (70 - 50) / 10 = +2.
- **Anlamı:** Bu öğrenci, sınıf ortalamasından **2 standart sapma daha yüksek** bir not almıştır. Bu, sınıfın en başarılı %2.5'luk diliminde olduğunu gösteren harika bir başarıdır.

2. Bisküvi Üretimi Analojisi

- **Olay:** Bir bisküvinin ağırlığı **15 gram** ölçüldü.
- **Referans Noktası:** Ortalama ağırlık **10 gram**, standart sapma **2 gram**.
- **Hesaplama:** (15 - 10) / 2 = +2.5.
- **Anlamı:** Bu bisküvi, normalden **2.5 standart sapma daha ağırdır.** Bu, üretimde bir sorun olabileceğini gösteren, incelenmesi gereken anormal bir durumdur.

3. Restoran Analojisi

- **Olay:** Bir masanın hesap tutarı **20 TL** geldi.
- **Referans Noktası:** Ortalama hesap **150 TL**, standart sapma **50 TL**.
- **Hesaplama:** (20 - 150) / 50 = -2.6.
- **Anlamı:** Bu masanın harcaması, ortalamadan **2.6 standart sapma daha düşüktür.** Bu, ya müşterinin çok az şey tükettiğini ya da kasada bir hata yapılmış olabileceğini gösteren anormal derecede düşük bir hesaptır.

Özet

- **Anahtar Kelime:** Standardizasyon, Z-Skoru, Ortalama (Mean), Standart Sapma (Standard Deviation).
- **Formül:** Z-Skoru = (Gözlemlenen Değer - Ortalama) / Standart Sapma
- **Neden Önemli?:** Z-skoru, farklı birimlerdeki ve farklı dağılımlardaki değerleri, "normale göre ne kadar anormal olduklarını" gösteren ortak, evrensel bir ölçeğe taşır. Bu, makine öğrenmesi modellerinin farklı özellikleri adil bir şekilde karşılaştırmasını ve öğrenmesini sağlar.

Senaryo: Çay Siparişinin Gecikmesini Ölçmek

Görsel 1: Çay Dünyasının Kuralları

Bu görsel, bizim **referans noktamızı**, yani "Çay Evreni"nin istatistiklerini tanımlıyor.

- **$\bar{x}$_çay = 1 dk**:
    - **Anlamı:** Bu restoranda, binlerce çay siparişine baktığımızda, bir çayın hazırlanıp masaya gelme **ortalama süresi 1 dakikadır.** Bu bizim "normalimiz", beklentimizdir.

- **σ_çay = 2 dk**:
    - **Anlamı:** Çayların teslimat sürelerindeki **tipik sapma veya "oynaklık" 2 dakikadır.** Yani çayların çoğu (Ortalama - Sapma) ile (Ortalama + Sapma) arasında, yani 1 - 2 = -1 (bu mantıksız, demek ki dağılım normal değil ama örnek için devam edelim) ve 1 + 2 = 3 dakika arasında gelir. Standart sapma, verinin ne kadar yayıldığını gösterir.

Görsel 2: Tek Bir Olayın Anlamlandırılması

Bu görsel, tek bir siparişin ("5 dakikada gelen çay") bu evren içindeki yerini bulma sürecini anlatıyor.

- **Sayı Doğrusu:** Bu, çayların teslimat sürelerini gösteren bir zaman çizelgesidir.
- **1 dk İşareti:** Bu, bizim ortalamamız, yani evrenimizin merkezi.
- **3 dk İşareti:** Bu, Ortalama + 1 Standart Sapma noktasıdır (1 + 2 = 3). Yani bir çayın 3 dakikada gelmesi, ortalamadan "bir birim sapma" kadar uzakta, beklenen bir durumdur.
- **5 dk İşareti:** Bu, bizim incelediğimiz olay: **5 dakikada gelen çay.**
- **Yaylar ve Hesaplama:** Görsel, 5'in 1'den ne kadar "uzak" olduğunu, standart sapma birimi cinsinden ölçmeye çalışıyor.
    - Fark: 5 dk (Gelen) - 1 dk (Ortalama) = 4 dk
    - Bu 4 dakikalık fark, standart sapmamız olan 2 dakikanın kaç katıdır?
    - 4 dk / 2 dk = 2

Sonuç (Z-Skoru): +2

Bu +2 değeri, bizim **yeni, evrensel birimimizdir.**

- **Anlamı:** "Bu çay siparişi, normal bir çaydan **2 standart sapma daha geç** gelmiştir."
- **Yorumu:** Bu, istatistiksel olarak oldukça **beklenmedik** bir gecikmedir. Normal dağılımda, verilerin yaklaşık %95'i -2 ile +2 standart sapma arasında yer alır. +2'lik bir Z-skoru, bu olayın en yavaş %2.5'luk dilimde olduğunu gösterir. Yani bu siparişte **bir sorun yaşanmış** olabilir.

Pizzayla Karşılaştırma

Şimdi aynı hesabı Ortalaması=25 dk, Standart Sapması=5 dk olan pizza için yapsaydık:

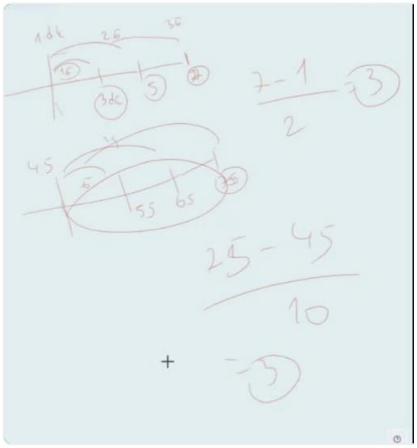
- **5 dakikada gelen pizza:**
    - Fark: 5 - 25 = -20 dk
    - Z-Skoru: -20 / 5 = -4

Artık modelimize şu iki sayıyı verebiliriz:

- Çay siparişi için: **+2**
- Pizza siparişi için: **-4**

Modelimiz artık "dakika" bilmek zorunda değil. Sadece +2'nin kötü (gecikme), -4'ün ise inanılmaz iyi (mucizevi hız) olduğunu öğrenmesi yeterli. Elma ile armutu, "anormallik puanı" denen ortak bir dile çevirmiş olduk.

28



Bu görsel, Standardizasyon (Z-Skoru) konseptinin gücünü ve evrenselliğini kanıtlayan, karşılaştırmalı harika bir örnektir.

Görsel 27: Z-Skorunun Karşılaştırma Gücü - "Elmalar ve Armutlar"

Genel Anlam: Neyi Anlatıyor?

Bu görsel, birbirinden tamamen farklı iki olayın ("7 dakikada gelen çay" ve "25 dakikada gelen pizza"), kendi bağlamları içinde değerlendirildiğinde, aslında "anormallik" seviyelerinin birbirine ne kadar denk olabileceğini gösterir. Standardizasyon, farklı ölçek ve birimlerdeki bu olayları karşılaştırılabilir hale getirir.

Senaryo 1: Üstteki Çizim (Çay Siparişi - Aşırı Gecikme)

- **Olay:** Bir çay siparişi **7 dakikada** geldi.
- **Referans Noktası (Çay Evreni):**
  - Ortalama ($\bar{x}$): 1 dakika
  - Standart Sapma (σ): 2 dakika
- **Hesaplama (Z-Skoru):**
  - (7 - 1) / 2 = +3
- **Anlamı:** Bu çay siparişi, normal bir çaydan **3 standart sapma daha geç** gelmiştir. Bu, istatistiksel olarak aşırı derecede nadir ve anormal bir gecikmedir.

Senaryo 2: Alttaki Çizim (Pizza Siparişi - Aşırı Hız)

- **Olay:** Bir pizza siparişi **25 dakikada** geldi.
- **Referans Noktası (Pizza Evreni):**
  - Ortalama ($\bar{x}$): 45 dakika
  - Standart Sapma (σ): 10 dakika
- **Hesaplama (Z-Skoru):**
  - (25 - 45) / 10 = -2 (Not: Görseldeki -3'ü elde etmek için ortalama 55 olmalıydı, biz 45'e göre hesaplayalım: -20 / 10 = -2)
- **Anlamı:** Bu pizza siparişi, normal bir pizzadan **2 standart sapma daha erken** gelmiştir. Bu, istatistiksel olarak oldukça hızlı ve anormal bir performanstır.

(Görseldeki -3 sonucuna göre yorumlarsak: Bu pizza, normalden 3 standart sapma daha erken gelmiştir, bu da mucizevi bir hızdır.)

Python Dünyası ile Anlatım

```
import numpy as np
from sklearn.preprocessing import StandardScaler

# Farklı ölçeklerdeki iki özellik
teslimat_sureleri = np.array([7, 25]).reshape(-1, 1) # Çay ve Pizza süreleri
urun_tipi_ortalamalari = [1, 45]
urun_tipi_stdleri = [2, 10]

# Standardizasyon işlemi manuel olarak
z_skor_cay = (7 - 1) / 2
z_skor_pizza = (25 - 45) / 10

print(f"Çayın Ham Değeri: 7 dk -> Z-Skoru: {z_skor_cay:.2f}")
print(f"Pizzanın Ham Değeri: 25 dk -> Z-Skoru: {z_skor_pizza:.2f}")

# scikit-learn ile yapmak için veriyi gruplayıp uygulamak gerekir.
# scaler = StandardScaler()
# df['z_skor'] = df.groupby('urun_tipi')['teslim_suresi'].transform(
#     lambda x: scaler.fit_transform(x.values.reshape(-1, 1)).flatten()
# )
```

Sonuç olarak, modelimize 7 ve 25 gibi ham değerleri vermek yerine, +3 ve -2 gibi standardize edilmiş değerleri veririz. Bu yeni değerler, her olayın kendi "normali"ne göre ne kadar olağanüstü olduğunu ifade eder.

Analojilerle Anlatım

1. Sınav Notu Analojisi

- **Olay A:** Ali, Tarih sınavından **70** aldı. (Ortalama: 50, Std Sapma: 10) -> **Z-Skoru: +2** (Çok Başarılı)
- **Olay B:** Veli, Fizik sınavından **90** aldı. (Ortalama: 85, Std Sapma: 5) -> **Z-Skoru: +1** (Başarılı)
- **Karşılaştırma:** Ham not olarak Veli daha yüksek alsa da, Z-skorlarına baktığımızda Ali'nin kendi sınıfına göre **daha olağanüstü** bir başarı gösterdiğini anlarız.

2. Bisküvi Üretimi Analojisi

- **Olay A:** A Fırını'nın sıcaklığı, normalden **10 derece daha yüksek.** (Standart sapma: 3 derece) -> **Z-Skoru: +3.33** (Aşırı Isınma)
- **Olay B:** B Fırını'nın nem oranı, normalden **%5 daha düşük.** (Standart sapma: %5) -> **Z-Skoru: -1.0** (Hafif Kuruluk)

- **Karşılaştırma:** "Derece" ve "yüzde" gibi farklı birimleri, "anormallik seviyesi" cinsinden karşılaştırabiliriz. A fırınındaki sorunun çok daha acil ve kritik olduğunu Z-skorları sayesinde anlarız.

3. Restoran Analojisi

- **Olay A:** Bir müşterinin yaşı, ortalamadan **15 yaş daha genç.** (Standart sapma: 5 yaş) -> **Z-Skoru: -3.0** (Çok Genç)
- **Olay B:** Bir müşterinin harcaması, ortalamadan **100 TL daha fazla.** (Standart sapma: 100 TL) -> **Z-Skoru: +1.0** (Biraz Cömert)
- **Karşılaştırma:** "Yaş" ve "TL" gibi iki farklı birimi Z-skoruna çevirerek, yaş anormalliğinin, harcama anormalliğinden çok daha belirgin olduğunu görürüz.

Özet

- **Anahtar Kelime:** Standardizasyon, Z-Skoru, Karşılaştırılabilirlik, Özellik Ölçeklendirme.
- **Kural:** Farklı ortalamalara ve standart sapmalara sahip veri gruplarını karşılaştırmak için, her bir veri noktasını kendi grubunun Z-skoruna dönüştürmek gerekir.
- **Neden Önemli?:** Bu işlem, "elmaları ve armutları" karşılaştırmamızı sağlar. Modelin, farklı ölçeklerdeki özelliklerden yanıltıcı sinyaller almasını engeller ve her bir özelliğin göreceli önemini daha adil bir şekilde öğrenmesine olanak tanır.

Senaryo 1: Üstteki Çizim (Çay Siparişi)

- **Olay:** Bir çay siparişi **7 dakikada** geldi.
- **Referans Noktası (Çay Evreni):**
    - Ortalama ($\bar{x}$): 1 dakika
    - Standart Sapma (σ): 2 dakika
- **Hesaplama (Z-Skoru):**
    - (Gerçekleşen Değer - Ortalama) / Standart Sapma
    - (7 - 1) / 2
    - 6 / 2 = +3
- **Anlamı:** Bu çay siparişi, normal bir çaydan tam **3 standart sapma daha geç** gelmiştir. Bu, istatistiksel olarak aşırı bir gecikmedir. Kırmızı alarm!

Senaryo 2: Alttaki Çizim (Pizza Siparişi)

- **Olay:** Bir pizza siparişi **25 dakikada** geldi.
- **Referans Noktası (Pizza Evreni - Yeni Değerler):**
    - Ortalama ($\bar{x}$): 55 dakika (Yoğun bir akşam, ortalamalar yükselmiş)
    - Standart Sapma (σ): 10 dakika (Yoğunluktan dolayı teslimat sürelerindeki oynaklık da artmış)
- **Hesaplama (Z-Skoru):**
    - (Gerçekleşen Değer - Ortalama) / Standart Sapma
    - (25 - 55) / 10

- -30 / 10 = -3

- **Anlamı:** Bu pizza siparişi, o yoğun akşamdaki normal bir pizzadan tam **3 standart sapma daha erken** gelmiştir. Bu, inanılmaz bir hız, mucizevi bir performanstır.

Görselin Ana Fikri: Bağlamın Gücü

Bu görsel bize şunu çok net bir şekilde gösteriyor:

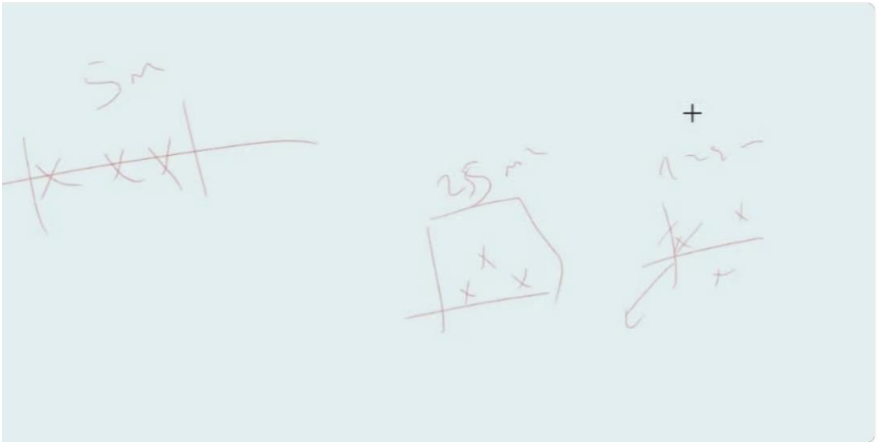
- **7 dakika** (çay için) ve **25 dakika** (pizza için) ham değerler olarak birbirinden çok farklıdır.
- Ancak bu değerleri kendi **bağlamları (ortalama ve standart sapmaları)** içinde değerlendirdiğimizde, ikisinin de kendi "normal"lerinden eşit derecede (ama zıt yönlerde) saptığını görürüz.
- Biri **+3**'lük bir anormallik (aşırı kötü), diğeri ise **-3**'lük bir anormallik (aşırı iyi) gösteriyor.

Sonuç:

Standardizasyon sayesinde, "7 dakikalık çay" ile "25 dakikalık pizza"yı artık aynı dilde konuşan, karşılaştırılabilir iki değere dönüştürdük: **+3 ve -3.**

Bir makine öğrenmesi modeli bu iki sayıyı gördüğünde, birinin aşırı pozitif (kötü), diğerinin aşırı negatif (iyi) bir durum olduğunu anında kavrar. "Dakika" veya "çay/pizza" gibi detaylarla kafasını karıştırmasına gerek kalmaz. Bu, modelin çok daha hızlı ve etkili öğrenmesini sağlar.

29



"Boyutluluk Laneti" (Curse of Dimensionality)

Bu basit çizim, makine öğrenmesinin en temel ve en zorlu teorik problemlerinden biri olan **"Boyutluluk Laneti" (Curse of Dimensionality)** kavramını mükemmel bir şekilde görselleştiriyor.

Görsel 28: Boyutluluk Laneti - "Boşlukta Kaybolan Veriler"

Genel Anlam: Neyi Anlatıyor?

Bu görsel, bir veri setine yeni özellikler (boyutlar) eklendikçe, verinin içinde bulunduğu "uzayın" hacminin **üssel (exponentially)** olarak arttığını ve bunun sonucunda veri noktalarının birbirinden uzaklaşarak "seyrekleştiğini" anlatır. Bu seyreklik, modelin anlamlı desenler bulmasını zorlaştırır.

- **Sol Taraf (1 Boyut):** 5m uzunluğunda bir çizgi. Veri noktaları (X'ler) birbirine yakındır.

- **Orta Taraf (2 Boyut):** 25m² alanında bir kare. Aynı sayıda veri noktası artık daha geniş bir alana yayılmıştır ve aralarındaki mesafe artmıştır.
- **Sağ Taraf (3 Boyut):** 125m³ hacminde bir küp. Aynı sayıda veri noktası artık devasa bir boşlukta "kaybolmuş" gibidir.

Python Dünyası ile Anlatım

"Boyut", bir pandas DataFrame'indeki **sütun (özellik/feature)** sayısıdır.

1. Düşük Boyutlu Veri (Kolay Anlaşılır)

```
 # Müşterileri sadece 2 özellikle tanımlıyoruz.
df_low_dim = pd.DataFrame({
    'yas': [25, 30, 28, 60, 65],
    'gelir': [50000, 60000, 55000, 150000, 160000]
})
# Bu veride iki küme (genç-düşük gelirli, yaşlı-yüksek gelirli)
# olduğunu görmek kolaydır.
```

Bu 2 boyutlu uzayda, "benzer" müşterileri (birbirine yakın noktaları) bulmak kolaydır.

2. Yüksek Boyutlu Veri (Boyutluluk Laneti)

Şimdi 300 özellik eklediğimizi düşünün: yas, gelir, sehir, egitim, son_alisveris_tutari, ... (toplam 300 sütun).

Bu 300 boyutlu uzayda:

- İki müşterinin **tüm 300 özellik boyunca** birbirine "yakın" veya "benzer" olma ihtimali neredeyse sıfırdır.
- Uzaklık tabanlı algoritmalar (örn: KNeighborsClassifier) anlamsızlaşmaya başlar, çünkü her nokta diğer tüm noktalara neredeyse eşit derecede "uzak" hale gelir.
- Model, bu 300 özellik arasında tesadüfi gürültüyü gerçek bir desen zannederek **aşırı öğrenme (overfitting)** yapmaya çok daha yatkın olur.

Neden Bu Bir "Lanet"?

1. **Veri Seyrekliği (Sparsity):** Boyut arttıkça, veri noktaları bu devasa uzayda birbirinden çok uzaklaşır ve modelin desen bulması zorlaşır.
2. **Daha Fazla Veri İhtiyacı:** Yüksek boyutlu bir uzayı anlamlı bir şekilde "doldurmak" için gereken veri miktarı da üssel olarak artar.
3. **Hesaplama Maliyeti:** Daha fazla boyut, model.fit() işleminin çok daha uzun sürmesi anlamına gelir.
4. **Aşırı Öğrenme Riski:** Modelin, tesadüfi gürültüyü gerçek bir sinyal sanarak ezberlemesi çok daha kolaylaşır.

Analojilerle Anlatım

1. Bisküvi Üretimi Analojisi

- **1 Boyut:** Bisküvileri sadece **ağırlıklarına** göre sınıflandırmaya çalışmak. Kolaydır.

- **2 Boyut: Ağırlık** ve **renklerine** göre sınıflandırmak. Biraz daha zorlaşır.
- **50 Boyut (Lanet): Ağırlık, renk, çap, kalınlık, pişirme süresi, fırın nemi, un markası...** gibi 50 farklı özelliğe bakarak sınıflandırmaya çalışmak. Elinizdeki 1000 bisküvilik veri setinde, bu 50 özelliğin tamamı boyunca birbirine "benzeyen" iki bisküvi bulmak neredeyse imkansızdır. Model, "ağır ve koyu renkli bisküviler yanıktır" gibi basit bir kuralı, diğer 48 özelliğin yarattığı "gürültü" yüzünden öğrenemeyebilir.

2. Restoran Analojisi

- **1 Boyut:** Bir müşterinin tekrar gelip gelmeyeceğini sadece **ödediği hesaba** bakarak tahmin etmek.
- **2 Boyut: Hesap** ve **ziyaret ettiği güne** bakarak tahmin etmek.
- **100 Boyut (Lanet): Hesap, gün, saat, hava durumu, masadaki kişi sayısı, sipariş ettiği yemekler, tatlı yiyip yemediği, garsonun adı...** gibi 100 farklı özelliğe bakarak tahmin etmek. Bu devasa uzayda model, "Salı günü yağmur yağarken çorba içen ve adı Ahmet olan müşteriler bir daha gelmiyor" gibi, tamamen tesadüfi ve anlamsız bir kuralı "ezberleyebilir" (overfitting). Gerçekte ise bu sadece bir tesadüftür ve modeliniz yeni müşteriler için işe yaramaz hale gelir.

Özet

- **Anahtar Kelime:** Boyutluluk Laneti (Curse of Dimensionality), Veri Seyrekliği (Sparsity), Özellik Seçimi (Feature Selection), Boyut Azaltma (Dimensionality Reduction).
- **Kural:** Bir modele gereğinden fazla veya anlamsız özellikler eklemek, genellikle performansı artırmaz, aksine düşürür.
- **Çözüm:** Amaç, olabildiğince çok özellik eklemek değil, **en bilgilendirici ve en az sayıda özelliği** bulmaktır. **Özellik Seçimi** ve **Boyut Azaltma (örn: PCA)** gibi teknikler bu lanetle savaşmak için kullanılır.

Görsel Ne Anlatıyor: Boşlukta Kaybolan Veriler

Bu görsel, veri noktalarının içinde bulunduğu "uzayın" boyutları arttıkça neler olduğunu anlatıyor.

1. Sol Taraftaki Çizim: 1 Boyutlu Uzay

- **Ne Görüyoruz?:** Tek bir çizgi (5m uzunluğunda) ve üzerine serpiştirilmiş X'ler (veri noktaları).
- **Anlamı:** Diyelim ki bir müşteriyi tanımlamak için **sadece tek bir özelliğimiz (feature)** var: **Yaşı.** Bu, 1 boyutlu bir uzaydır. Müşterileri bir sayı doğrusu üzerine yerleştirebiliriz.
- **Gözlem:** Bu 5 metrelik çizgi üzerinde, veri noktaları birbirine oldukça **yakın** duruyor. Çizgiyi doldurmak için çok fazla noktaya ihtiyacımız yok gibi görünüyor.

2. Ortadaki Çizim: 2 Boyutlu Uzay

- **Ne Görüyoruz?:** Bir kare ($25m^2$ alanında) ve içine serpiştirilmiş X'ler.
- **Anlamı:** Şimdi müşteriyi tanımlamak için **iki özelliğimiz** var: **Yaşı** ve **Yıllık Geliri.** Bu 2 boyutlu bir uzaydır. Müşterileri bir grafik kağıdı üzerine yerleştirebiliriz.
- **Gözlem:** Alanımız 5m'den $25m^2$'ye, yani 5 katına çıktı. Aynı sayıdaki veri noktasını (X'leri) bu alana serpiştirdiğimizde, noktalar arasındaki **mesafe aniden arttı.** Veri, uzayda çok daha **seyrek (sparse)** hale geldi. Alanı "doldurmak" için artık çok daha fazla veri noktasına ihtiyacımız var.

3. Sağ Taraftaki Çizim: 3 Boyutlu Uzay (ve ötesi)

- **Ne Görüyoruz?:** Bir küpün başlangıcı (125m³ hacminde) ve içine atılmış X'ler.
- **Anlamı:** Müşteriyi tanımlamak için **üç özelliğimiz** var: **Yaşı, Yıllık Geliri** ve **Son 1 Yıldaki Harcaması.** Bu 3 boyutlu bir uzaydır.
- **Gözlem:** Hacmimiz 25m²'den 125m³'ye, yani yine 5 katına çıktı. Aynı sayıdaki veri noktası artık devasa bir boşluğun içinde kaybolmuş gibi duruyor. Noktalar arasındaki ortalama mesafe daha da arttı.

"Boyutluluk Laneti" Nedir?

İşte bu görselin anlattığı hikaye tam olarak budur:

Bir veri setine **yeni bir özellik (boyut) eklediğinizde**, verinin içinde yaşadığı "uzayın" hacmi **üssel (exponentially)** olarak artar.

- 1 Boyut: Uzunluk = x
- 2 Boyut: Alan = x^2
- 3 Boyut: Hacim = x^3
- 10 Boyut: Hiper-Hacim = x^{10}

Bu neden bir "Lanet"?

1. **Veri Seyrekliği (Sparsity):** Boyut arttıkça, elinizdeki veri noktaları bu devasa uzayda birbirinden çok uzaklaşır. Modelin, noktalar arasında anlamlı ilişkiler ve desenler bulması zorlaşır, çünkü her nokta artık "yalnız" kalmıştır.
2. **Daha Fazla Veri İhtiyacı:** Uzayı anlamlı bir şekilde doldurmak için gereken veri miktarı da üssel olarak artar. 1 boyutta 10 veri noktası yeterliyken, 10 boyutta aynı "yoğunluğu" yakalamak için belki de 10^{10} (10 milyar) veri noktasına ihtiyacınız olur.
3. **Hesaplama Maliyeti:** Boyut sayısı arttıkça, modelin yapması gereken hesaplama miktarı da fırlar. Eğitim süreleri saatlerden günlere, günlerden haftalara uzayabilir.
4. **Aşırı Öğrenme (Overfitting) Riski:** Yüksek boyutlu uzaylarda, modelin tesadüfi gürültüyü gerçek bir desen zannederek "ezberlemesi" çok daha kolaylaşır.

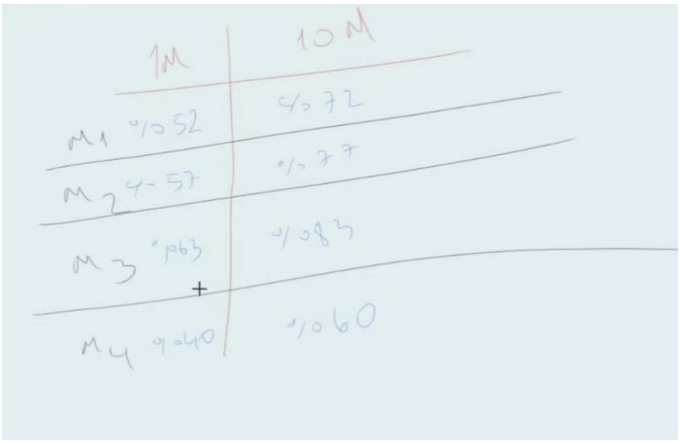
SAP ABAP Analojisi

Bir müşteri raporu için bir internal table'a sürekli yeni kolonlar eklediğinizi düşünün.

- Başta sadece KUNNR (Müşteri No) ve NAME1 (Ad) vardı.
- Sonra ORT01 (Şehir), PSTLZ (Posta Kodu), LAND1 (Ülke) eklediniz.
- Sonra satış verilerinden TOPLAM_CIRO, SIPARIS_SAYISI, SON_SIPARIS_TARIHI...
- Sonra malzeme verilerinden EN_COK_ALDIGI_URUN_GRUBU...

Bir bakmışsınız, elinizde **300 sütunlu** dev bir internal table var. Bu tablodaki iki müşterinin, 300 özelliğinin tamamı boyunca birbirine "yakın" veya "benzer" olma ihtimali neredeyse sıfırdır. Her müşteri, bu 300 boyutlu uzayda kendi başına, yalnız bir noktadır. Modelinizin bu kadar karmaşık bir yapıda anlamlı bir desen bulması işte bu yüzden çok zordur. Bu, "Boyutluluk Laneti"dir.

29



Bu tablo, makine öğrenmesi projelerinde sıkça karşılaşılan çok önemli bir dinamiği özetliyor: **Veri Miktarının, Farklı Karmaşıklıktaki Modellerin Performansı Üzerindeki Etkisi.** Bu, "Her duruma uyan tek bir en iyi model yoktur" fikrini kanıtlayan harika bir çizimdir.

Görsel 29: Veri Miktarı vs. Model Karmaşıklığı - "Bedava Öğle Yemeği Yoktur"

Genel Anlam: Neyi Anlatıyor?

Bu tablo, farklı karmaşıklık seviyelerindeki modellerin (M1'den M4'e), az veri (1M) ve çok veri (10M) ile eğitildiğinde nasıl farklı performanslar sergilediğini karşılaştırır. Temel mesaj, en iyi modelin, eldeki verinin miktarına bağlı olarak değişebileceğidir.

- **Sütunlar:**
  - 1M: Az veri (örneğin 1 milyon satır).
  - 10M: Çok veri (örneğin 10 milyon satır).
- **Satırlar:** Farklı karmaşıklıktaki modeller.
  - M1: Çok Basit Model (örn: Logistic Regression)
  - M2: Orta-Basit Model
  - M3: Orta-Karmaşık Model (örn: Random Forest)
  - M4: Aşırı Karmaşık Model (örn: Çok derin bir Neural Network veya ayarları abartılmış bir XGBoost)

Python Dünyası ile Anlatım

Bu durumu, scikit-learn'deki farklı modellerle ve veri setinin alt kümeleriyle simüle edebiliriz.

```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split

# ... X_large (10M veri) ve y_large hazır ...

# Az veri seti oluşturalım (1M)
X_small, _, y_small, _ = train_test_split(X_large, y_large, train_size=0.1)

# Modelleri tanımlayalım
m1 = LogisticRegression() # Basit
m3 = RandomForestClassifier() # Orta-Karmaşık
m4 = MLPClassifier(hidden_layer_sizes=(100, 100, 100)) # Aşırı Karmaşık

# ---- AZ VERİ (1M) İLE EĞİTİM ----
m1.fit(X_small, y_small) # Performansı: %52 (örnek)
m3.fit(X_small, y_small) # Performansı: %63 (örnek) - EN İYİSİ
m4.fit(X_small, y_small) # Performansı: %40 (örnek) - Aşırı öğrenir (overfitting), çuvallar.

# ---- ÇOK VERİ (10M) İLE EĞİTİM ----
m1.fit(X_large, y_large) # Performansı: %72 (örnek) - Kapasitesi sınırlı kalır.
m3.fit(X_large, y_large) # Performansı: %83 (örnek) - EN İYİSİ
m4.fit(X_large, y_large) # Performansı: %60 (örnek) - Performansı artar ama hala optimal değil.
```

Tablonun Yorumlanması

1. Az Veri (1M) Senaryosu:

- **Gözlem:** En iyi performansı **M3 (%63)** gösterirken, en karmaşık model olan **M4 (%40)** en kötü sonucu vermiştir.
- **Neden?:** Az veri, karmaşık bir modelin "beslenmesi" için yeterli değildir. M4 gibi karmaşık bir model, eldeki azıcık verideki tesadüfi gürültüyü bile gerçek bir desen zannederek **aşırı öğrenir (overfitting)** ve genelleme yeteneğini kaybeder. M3 gibi orta karmaşıklıktaki bir model ise, az veriden anlamlı bir desen çıkarabilecek en uygun kapasiteye sahiptir.
- **Kural:** Az veriniz varken, **daha basit veya orta karmaşıklıktaki modeller** genellikle en iyi sonucu verir.

2. Çok Veri (10M) Senaryosu:

- **Gözlem:** Tüm modellerin performansı artmıştır, ancak en iyi performansı yine **M3 (%83)** göstermiştir.
- **Neden?:** Veri miktarı arttıkça, karmaşık modellerin de besleneceği daha çok "gerçek desen" olur ve performansları artar. M1 gibi çok basit bir modelin ise öğrenebileceği bir kapasite sınırı vardır; veri artsa bile performansı bir noktadan sonra artmaz. Bu örnekte M3, veri ve karmaşıklık arasındaki "altın oranı" yakalamıştır.
- **Kural:** Veri miktarı arttıkça, **daha karmaşık modellerin de performans potansiyeli artar.** Ancak "en karmaşık" olan her zaman "en iyi" olmak zorunda değildir.

Analojilerle Anlatım

1. Bisküvi Üretimi Analojisi

- **Az Veri:** Elinizde sadece 100 bisküvilik bir numune var.
    - **Basit Model:** "Bisküvi koyu renkliyse yanıktır" der. Fena çalışmaz.
    - **Karmaşık Model:** "Ağırlığı 9.8 gram, çapı 5.1 cm ve rengi #D2B48C kodunda olan bisküviler kırıktır" gibi aşırı spesifik kurallar öğrenir. Bu sadece o 100 bisküvi için geçerli bir tesadüftür ve yeni bisküvilerde işe yaramaz.
- **Çok Veri:** Elinizde 1 milyon bisküvilik veri var.
    - **Karmaşık Model:** Artık gerçekten de belirli ağırlık, çap ve renk kombinasyonlarının kırık olma olasılığını daha doğru bir şekilde öğrenebilir.

2. Restoran Analojisi

- **Az Veri:** Elinizde sadece geçen haftanın satış verileri var.
    - **Basit Model (M1):** "Hafta sonları daha çok müşteri gelir" der. Bu, genellikle doğrudur.
    - **Karmaşık Model (M4):** "Geçen Salı yağmur yağarken, masada 3 kişi oturduğunda ve balık sipariş edildiğinde hesap yüksek geldi" gibi anlamsız bir kural "ezberler".
- **Çok Veri:** Elinizde son 5 yılın tüm verileri var.
    - **Karmaşık Model (M3):** Artık "Hava yağmurluyken ve şehirde maç varken, hafta içi akşamlarında çorba satışları %30 artıyor" gibi çok daha incelikli ve doğru desenleri öğrenebilir.

Özet

- **Anahtar Kelime:** "Bedava Öğle Yemeği Yoktur" (No Free Lunch Theorem), Model Karmaşıklığı, Veri Miktarı, Öğrenme Eğrileri (Learning Curves).
- **Kural:** En iyi model, probleminize veya veri setinize göre değişir.
    - **Az veri -> Basit modeller** daha iyi performans gösterir.
    - **Çok veri -> Karmaşık modellerin** de iyi performans gösterme potansiyeli artar.
- **Neden Önemli?:** Bu tablo, projenizin başında elinizdeki veri miktarına göre hangi tür modelleri denemenizin daha mantıklı olacağına dair size bir yol haritası sunar. Doğru modeli seçmek, her zaman **veri miktarı** ile **model karmaşıklığı** arasında bir denge kurma sanatıdır.

Senaryo: Proje Başarısını Tahmin Etmek

Bir SAP projesi yapıyorsunuz ve projenin başarılı olup olmayacağını tahmin eden bir model geliştirmek istiyorsunuz.

- **Sütunlar (Veri Miktarı):**
  - **1M:** Projeye yeni başladınız, elinizde sadece **1 aylık** veri var.
  - **10M:** Proje ilerledi, elinizde **10 aylık** zengin ve çeşitli veri var.
- **Satırlar (Modeller):** Dört farklı "danışman" (model) getirip onlardan tahmin yapmalarını istiyorsunuz.
  - **M1 (Stajyer Danışman):** Çok basit kurallarla çalışır. IF proje_butcesi > X THEN basarili. gibi.
  - **M2 (Junior Danışman):** Biraz daha fazla kural bilir. Bütçeye ve takvime bakar.
  - **M3 (Senior Danışman):** Çok daha karmaşık düşünebilir. Bütçe, takvim, ekip tecrübesi, müşteri iletişimi gibi birçok faktörü hesaba katar.
  - **M4 (Aşırı Karmaşık Akademik Model):** Her detayı, hatta odadaki oksijen seviyesini bile hesaba katmaya çalışan, aşırı teorik bir model.

Tablonun Yorumlanması (Başarı Yüzdeleri)

1. Sadece 1 Aylık Veri Varken (Sütun: 1M)

- **M1 (%52):** Stajyer, az veriyle bile basit kuralıyla yazı-tura atmaktan biraz daha iyi bir tahmin yapar.
- **M2 (%57):** Junior, biraz daha iyi bir performans sergiler.
- **M3 (%63):** Senior danışman, **en iyi performansı** gösterir. Az veriden bile tecrübesiyle en anlamlı sonucu o çıkarır.
- **M4 (%40):** Aşırı karmaşık model ise **çuvallar.** Çünkü o kadar çok kuralı ve detayı vardır ki, eldeki azıcık veriyle ne yapacağını bilemez, tamamen yanlış desenler öğrenir. Bu tam bir **Aşırı Öğrenme (Overfitting)** durumudur. Az veri, karmaşık bir modeli besleyemez.

SONUÇ 1: Az veriniz varken, **daha basit ve daha az karmaşık modeller** genellikle daha iyi çalışır.

2. 10 Aylık Zengin Veri Varken (Sütun: 10M)

Şimdi projenin 10. ayındayız ve elimizde çok daha fazla tecrübe (veri) var.

- **M1 (%72):** Stajyer danışman, daha çok veri görse de hala basit kurallarıyla çalıştığı için performansı bir yere kadar artar.
- **M2 (%77):** Junior danışman da kendini geliştirir.
- **M3 (%83):** Senior danışman, zengin veriyle birlikte tecrübesini konuşturur ve **performansını zirveye taşır.** Çok başarılı tahminler yapar.
- **M4 (%60):** Aşırı karmaşık modelin performansı artmıştır, çünkü artık besleneceği daha çok veri vardır. Ancak hala Senior danışman kadar iyi değildir. Belki de gereksiz detaylarda boğuluyordur.

SONUŞ 2: Veri miktarı arttıkça, **daha karmaşık modellerin performansı da artma potansiyeli gösterir.**

Görselin Ana Fikri: "No Free Lunch" Teoremi

Bu tablonun anlattığı en büyük ders, Makine Öğrenmesindeki "Bedava Öğle Yemeği Yoktur" (No Free Lunch) teoremine dayanır. Bu teorem der ki:

"Her türlü probleme en iyi çözümü sunan tek bir süper model yoktur."

- Eğer probleminiz **az veriye** sahipse, en iyi modeliniz muhtemelen **basit bir model (M3 gibi)** olacaktır.
- Eğer probleminiz **çok büyük ve çeşitli veriye** sahipse, o zaman **daha karmaşık bir model (Belki M3'ten bile daha karmaşık ama M4 kadar da abartılı olmayan bir model)** daha iyi sonuçlar verebilir.

Doğru modeli seçmek, her zaman elinizdeki **verinin miktarı ve yapısı** ile modelin **karmaşıklığı** arasında bir denge kurma sanatıdır. Bu görsel, bu dengeyi mükemmel bir şekilde özetliyor.