Samsun Üniversitesi
Yazılım Mühendisliği Bölümü

VERİ MADENCİLİĞİ (DATA MİNİNG)

Dr. Öğr. Üyesi Alper Talha Karadeniz

2025-2026

Hafta 8

İlişki Kuralları Madenciliği

Samsun Üniversitesi
Yazılım Mühendisliği Bölümü

01

Destek, güven ve kaldırma

02

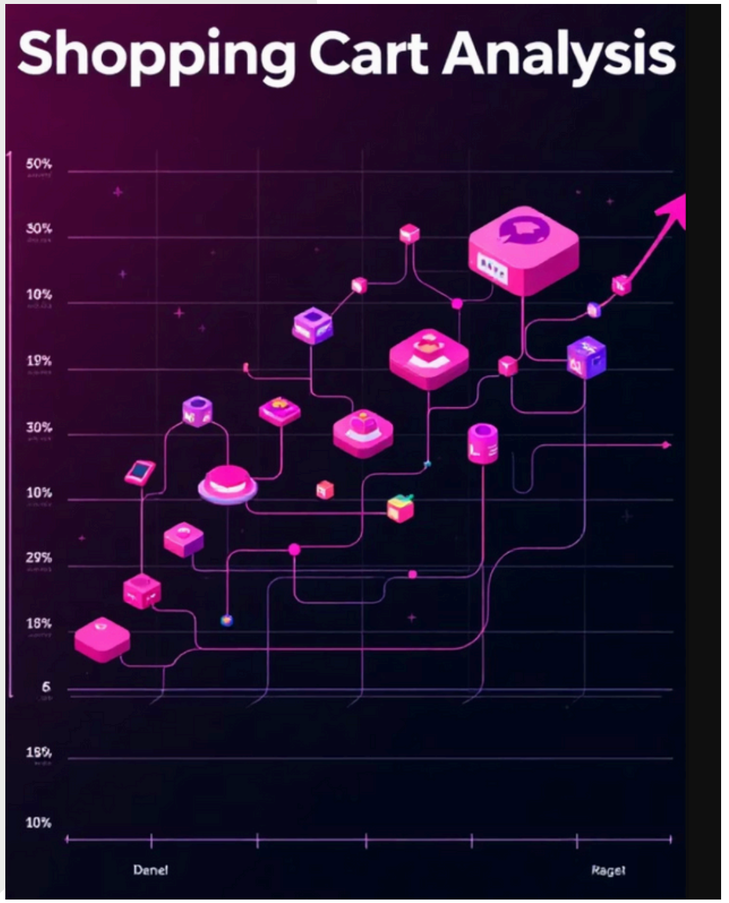
Apriori ve FP-Growth algoritmaları

03

Uygulama

İlişki Kuralları Madenciliği Nedir?

İlişki kuralları madenciliği, büyük ölçekli veri kümeleri içerisinde öğeler arasındaki sık tekrarlayan ilişkileri keşfetmek amacıyla kullanılan güçlü ve etkili bir veri madenciliği tekniğidir. Bu yöntem, özellikle ticaret ve perakende sektöründe müşterilerin alışveriş davranışlarını analiz etmek için yaygın bir şekilde tercih edilmektedir. Müşteri alışveriş verileri üzerinde yapılan analizler "market basket analysis" (pazar sepeti analizi) olarak adlandırılır. Bu analiz sayesinde, müşterilerin alışveriş sepetlerinde hangi ürünlerin genellikle birlikte satın alındığı, hangi ürünlerin birbirini tamamlayıcı nitelikte olduğu veya belirli müşteri gruplarının hangi ürün kombinasyonlarına yöneldiği ortaya çıkarılabilir. Elde edilen bu bilgiler işletmelere çapraz satış stratejileri geliştirme, promosyon kampanyaları planlama ve ürün yerleşim düzenini optimize etme gibi konularda önemli avantajlar sağlar. Dolayısıyla ilişki kuralları madenciliği, yalnızca veri içerisindeki gizli kalmış ilişkileri açığa çıkarmakla kalmaz, aynı zamanda işletmelere rekabet avantajı kazandıran stratejik kararlar almalarında da kritik bir rol oynar.



Samsun Üniversitesi
Yazılım Mühendisliği Bölümü

Temel Kavramlar

İlişki kuralları madenciliğinde üç temel ölçüt vardır.Bunlar, kuralların anlamlılığını ve kullanışlılığını değerlendirmek için kullanılır

Destek (Support) → Bir öğe setinin veri kümesinde kaç kez geçtiğini gösterir. Yüksek destek değeri, ilgili öğelerin veri setinde sık görüldüğünü ifade eder.

Güven (Confidence) → X → Y kuralının doğruluk oranını gösterir. X ürünü alındığında Y ürününün de alınma olasılığını ifade eder.

Artış (Lift) → X ve Y` nin birlikte görülme olasılığının, birbirinden bağımsız olma durumuna göre ne kadar arttığını gösterir. 1` den büyük değerler pozitif ilişkiyi gösterir.

Destek (Support):

- **Tanım:** Bir öğe kümesinin toplam işlemler içinde ne kadar sıklıkla görüldüğünü ölçer.
- **Amacı:** Hangi öğe veya öğe gruplarının "önemli" olduğunu belirlemeye yardımcı olur.

$$Support(X \Rightarrow Y) = \frac{\text{X ve Y'nin birlikte geçtiği işlem sayısı}}{\text{Toplam işlem sayısı}}$$

Örnek:
Toplam 5 işlem var:
{Ekmek, Süt}
{Ekmek, Bez, Süt}
{Ekmek, Bez}
{Süt, Bez}
{Ekmek, Süt}
X = {Ekmek}, Y = {Süt}
Birlikte görüldükleri işlem sayısı = 3 (T1, T2, T5)
Support = 3 / 5 = 0.6

Güven (Confidence):

- **Tanım:** X alındığında Y'nin de alınma olasılığını gösterir.
- **Amacı:** Kuralın doğruluğunu ölçer, yani X'in gerçekleşmesi Y'yi ne kadar tahmin edebilir.

$$Confidence(X \Rightarrow Y) = \frac{Support(X \cup Y)}{Support(X)}$$

Örnek:
Support({Ekmek, Süt}) = 0.6
Support({Ekmek}) = 0.8
Confidence({Ekmek} → {Süt}) = 0.6 / 0.8 = 0.75
Yani, ekmek alanların %75'i süt de alıyor.

Kaldırma (Lift):

- **Tanım:** X ve Y'nin birbirinden bağımsız mı yoksa ilişkili mi olduğunu ölçer.
- **Amacı:** Kuralın anlamlı olup olmadığını değerlendirir.

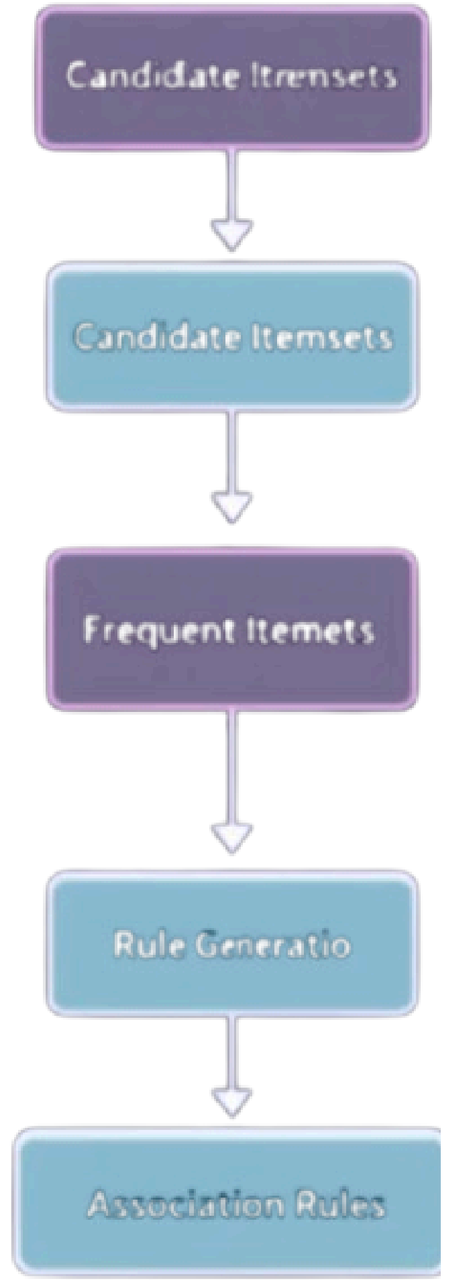
$$Lift(X \Rightarrow Y) = \frac{Confidence(X \Rightarrow Y)}{Support(Y)}$$

Örnek:
Support(Y = Süt) = 0.8
Confidence({Ekmek} → {Süt}) = 0.75
Lift = 0.75 / 0.8 = 0.9375
Yorum: Lift < 1 → ilişki pozitif değil, yani ekmek almak süt alma olasılığını artırmıyor (nispeten bağımsız).

Ölçüt	Anlamı	Amaç	Örnek sonucu
Support	Ne kadar sık görülüyor	Önemli öğeleri bulmak	0.6
Confidence	X alındığında Y'nin olasılığı	Kuralın doğruluğu	0.75
Lift	X ve Y'nin bağımlılığı	Kural anlamlı mı?	0.9375

Apriori Algoritması Nedir?

Apriori algoritması, ilişki kuralları madenciliğinde en sık kullanılan yöntemlerden biridir. Temel amacı, büyük veri kümelerinde sık geçen öğe gruplarını (frequent itemsets) belirlemek ve bu gruplardan anlamlı birliktelik kuralları türetmektir. Algoritma, "Apriori özelliği" olarak bilinen prensibe dayanır; bu prensibe göre bir öğe kümesi sık değilse, bu kümeyi içeren daha büyük hiçbir öğe kümesi de sık olamaz. Bu sayede gereksiz aday kümelerin oluşturulması engellenir ve işlem yükü azaltılır. Algoritma, önce tek öğeli kümelerin destek değerini (support) hesaplayarak sık olanları belirler, ardından bu kümelerden daha büyük kombinasyonlar türetir. Bu işlem, destek eşiğini sağlamayan aday kümeler kalmayana kadar tekrarlanır. Son aşamada, elde edilen sık öğe kümelerinden güven (confidence) ve kaldırma (lift) değerleri kullanılarak güçlü ilişki kuralları çıkarılır.



Apriori Algoritması Adımları

Destek (Support) Eşiğinin Belirlenmesi

Öncelikle, "sık" kabul edilecek öğe kümelerinin belirlenmesi için bir destek eşiği (min support) tanımlanır.

$$Support(X) = \frac{\text{X'in bulunduğu işlem sayısı}}{\text{Toplam işlem sayısı}}$$

1-Öğeli Sık Kümelerin Belirlenmesi

- Tüm ürünlerin (tek tek) destek değerleri hesaplanır.
- Belirlenen destek eşiğinden küçük olanlar elenir.
- Kalanlar L1 (frequent 1-itemsets) kümesini oluşturur.

Aday 2-Öğeli Kümelerin (C2) Oluşturulması

- L1'den hareketle 2'li kombinasyonlar oluşturulur (candidate sets).
- Bu kombinasyonların destekleri hesaplanır.
- Eşiğin altındaki aday kümeler elenir → L2 elde edilir.

Samsun Üniversitesi
Yazılım Mühendisliği Bölümü

Apriori Algoritması Adımları

Kombinasyonların Genişletilmesi (k-Itemsets)

→

- L2'den hareketle 3 öğeli aday kümeler (C3) oluşturulur.
- Aynı şekilde destek hesaplanır → eşiğin altındakiler elenir → L3 elde edilir.
- Bu işlem, yeni sık küme kalmayana kadar devam eder.

Güçlü İlişki Kurallarının Çıkarılması

→

- Sık öğe kümelerinden kurallar türetilir:

$$X \Rightarrow Y$$

şeklinde, "X alındığında Y de alınır" kuralı.

- **Confidence (Güven):**

$$Confidence(X \Rightarrow Y) = \frac{Support(X \cup Y)}{Support(X)}$$

→ X alındığında Y'nin de alınma olasılığı.

- **Lift (Kaldırma):**

$$Lift(X \Rightarrow Y) = \frac{Confidence(X \Rightarrow Y)}{Support(Y)}$$

→ X ve Y'nin bağımsız mı yoksa gerçekten ilişkili mi olduğunu gösterir.

Samsun Üniversitesi
Yazılım Mühendisliği Bölümü

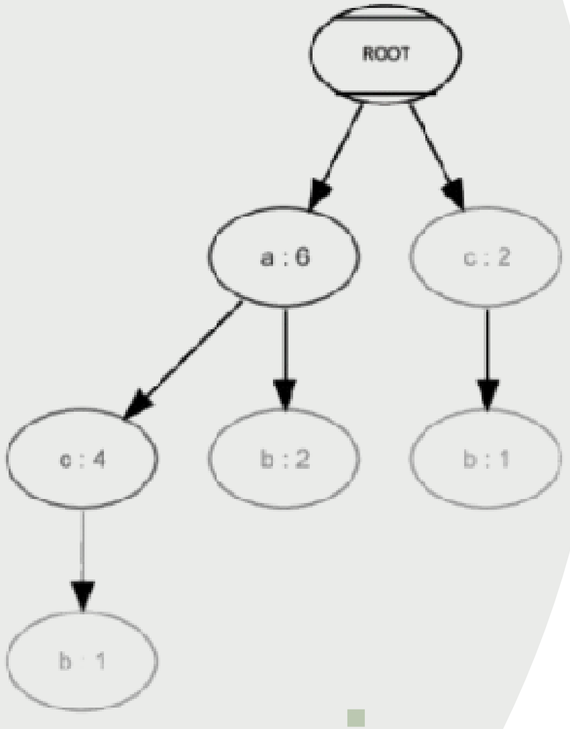
Apriori Algoritmasında Hesaplama Verimliliğini Etkileyen Faktörler

- **Veri kümesinin büyüklüğü** → Kayıt sayısı ve ürün çeşitliliği arttıkça aday küme sayısı da katlanarak artar, bu da algoritmanın yavaşlamasına neden olur.
- **Destek (support) eşiği** → Düşük destek eşiği seçildiğinde daha fazla aday küme oluşur, bu da hesaplama maliyetini artırır. Yüksek destek eşiği ise daha az aday küme üretilmesini sağlar ama bazı değerli kuralların kaybolmasına yol açabilir.
- **Öğe (item) sayısı** → Ürün veya özellik sayısı fazla olan veri kümelerinde kombinasyon sayısı çok artar, bu da bellek ve zaman maliyetini yükseltir.
- **Veri yoğunluğu (density)** → Eğer veri setindeki alışveriş sepetleri çok sayıda ürün içeriyorsa (yoğun veri), daha fazla öğe kombinasyonu oluşur ve algoritmanın çalışma süresi uzar.
- **İyileştirme teknikleri** → Hash tabanlı teknikler, aday küme azaltma stratejileri veya paralel hesaplama yöntemleri kullanılmazsa algoritma çok daha yavaş çalışabilir.

Samsun Üniversitesi
Yazılım Mühendisliği Bölümü

FP-Growth Algoritması Nedir?

FP-Growth algoritması, sık öğe kümelerinin bulunmasında kullanılan, Apriori algoritmasına alternatif hızlı ve verimli bir yöntemdir. Apriori'nin aksine tüm aday kümeleri tek tek üretmek yerine, veriyi FP-Tree (Frequent Pattern Tree) adı verilen kompakt bir ağaç yapısına dönüştürerek saklar. Bu yapı sayesinde veri tabanındaki ortak desenler sıkıştırılır ve gereksiz tekrarların önüne geçilir. Algoritma veri tabanını yalnızca iki kez taradığı için bellek ve zaman açısından önemli avantajlar sağlar. Özellikle büyük ölçekli veri kümelerinde, market sepeti analizi gibi uygulamalarda sıkça tercih edilmektedir.



a, b, c, d
a, c
a, b, d
a, c
$c,$
b, c, d
a, b
a, c

Samsun Üniversitesi
Yazılım Mühendisliği Bölümü

FP-Growth Algoritması Adımları

Veri tabanının ilk taraması →

- Tüm öğelerin (ürünlerin) destek (support) değerleri hesaplanır.
- Minimum destek eşiğini karşılamayan öğeler elenir.
- Geriye kalan öğeler, destek değerine göre azalan sırada sıralanır.

FP-Tree (Frequent Pattern Tree) oluşturma →

- Filtrele: Tüm öğelerin support'unu say, min support altındakileri at.
- Sırala: Kalan öğeleri azalan supporta göre tek bir global sıraya koy.
- İşlemleri hazırla: Her işlemde yalnız sık öğeleri tut, global sıraya göre diz.
- Ağaca ekle: Kökten başla; ortak prefix varsa mevcut düğüm sayaçlarını artır, yoksa yeni düğüm oluştur.
- Header table: Her öğe için ağaçtaki düğümlerine bağlantı listesi (node-link) tut.
- Sonuç: Sık ortak yollar sıkıştırılmış FP-Tree + Header table; bunlar sonraki adımda koşullu FP-Tree üretmek için kullanılır.

Samsun Üniversitesi
Yazılım Mühendisliği Bölümü

FP-Growth Algoritması Adımları

Koşullu desen tabanı oluşturma →

- Her öğe için, FP-Tree üzerinde o öğeye ulaşan yollar çıkarılır (conditional pattern base).
- Bu yollar üzerinden koşullu FP-Tree'ler kurulur.

Sık öğe kümelerinin çıkarılması →

- Koşullu FP-Tree'ler kullanılarak öğelerin birlikte görülme sıklığı hesaplanır.
- Böylece minimum destek değerini karşılayan sık öğe kümeleri elde edilir.

İlişki kurallarının oluşturulması →

- Sık öğe kümelerinden, güven (confidence) ve kaldıraç (lift) gibi ölçütlerle ilişki kuralları üretilir.

Samsun Üniversitesi
Yazılım Mühendisliği Bölümü

FP-Growth Algoritması

Avantajları:

- **Daha hızlıdır:** Apriori gibi aday küme üretimi yapmaz, bu yüzden büyük veri kümelerinde daha verimlidir.
- **Bellek kullanımında etkili:** FP-Tree sayesinde tekrar eden öğeler sıkıştırılarak depolanır.
- **Ölçeklenebilir:** Çok büyük veri kümeleri ve uzun işlem listelerinde daha iyi performans verir.
- **Sık öğeleri net gösterir:** Ağaç yapısı, verideki sık öğe gruplarını görsel olarak da kolayca anlamaya yardımcı olur.

Dezavantajları:

- **Ağaç yapısının karmaşıklığı:** FP-Tree oluşturmak ve yönetmek, Apriori'ye göre daha zordur.
- **Bellek sınırlaması:** Çok büyük ve farklı öğe çeşitliliğine sahip veri kümelerinde FP-Tree bellekte çok yer kaplayabilir.
- **Uygulama zorluğu:** Gerçekleştirilmesi Apriori'ye göre daha karmaşık olduğundan, algoritmanın kodlanması daha zahmetlidir.
- **Dinamik veride zor:** Veri sürekli değiştiğinde ağacı yeniden kurmak gerekebilir, bu da zaman alıcıdır.

UYGULAMA

Market Sepeti Analizi I

Kod Parçası

```python
# Gerekli kütüphaneler
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rules

# Örnek veri seti: Market sepeti
dataset = [
    ['Ekmek', 'Süt'],
    ['Ekmek', 'Bez', 'Süt'],
    ['Ekmek', 'Bez'],
    ['Süt', 'Bez'],
    ['Ekmek', 'Süt']
]

# Veriyi one-hot encoding formatına çevirme
te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
df = pd.DataFrame(te_ary, columns=te.columns_)

print("One-hot encoding ile veri:")
print(df)
print("\n")
```

Kod Çıktısı

```
One-hot encoding ile veri:
     Bez  Ekmek    Süt
0  False   True   True
1   True   True   True
2   True   True  False
3   True  False   True
4  False   True   True
```

UYGULAMA

Market Sepeti Analizi II

Kod Parçası

```python
# Gerekli kütüphaneler
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rules

# Örnek veri seti: Market sepeti
dataset = [
    ['Ekmek', 'Süt'],
    ['Ekmek', 'Bez', 'Süt'],
    ['Ekmek', 'Bez'],
    ['Süt', 'Bez'],
    ['Ekmek', 'Süt']
]

# Veriyi one-hot encoding formatına çevirme
te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
df = pd.DataFrame(te_ary, columns=te.columns_)
# Apriori ile sık öğe kümelerini bulma
frequent_itemsets = apriori(df, min_support=0.4, use_colnames=True)
print("Sık öğe kümeleri (min_support=0.4):")
print(frequent_itemsets)
print("\n")
```

Kod Çıktısı

```
Sık öğe kümeleri (min_support=0.4):
   support      itemsets
0      0.6         (Bez)
1      0.8       (Ekmek)
2      0.8         (Süt)
3      0.4  (Bez, Ekmek)
4      0.4    (Bez, Süt)
5      0.6  (Süt, Ekmek)
```

UYGULAMA

Market Sepeti Analizi III

Kod Parçası

```python
# Gerekli kütüphaneler
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rules

# Örnek veri seti: Market sepeti
dataset = [
    ['Ekmek', 'Süt'],
    ['Ekmek', 'Bez', 'Süt'],
    ['Ekmek', 'Bez'],
    ['Süt', 'Bez'],
    ['Ekmek', 'Süt']
]

# Veriyi one-hot encoding formatına çevirme
te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
df = pd.DataFrame(te_ary, columns=te.columns_)
frequent_itemsets = apriori(df, min_support=0.4, use_colnames=True)
# İlişki kuralları çıkarma
rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.5)
print("İlişki kuralları (confidence >= 0.5):")
print(rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']])
```

Kod Çıktısı

```
İlişki kuralları (confidence >= 0.5):
  antecedents consequents  support  confidence      lift
0     (Ekmek)       (Bez)      0.4    0.500000  0.833333
1       (Bez)     (Ekmek)      0.4    0.666667  0.833333
2       (Süt)       (Bez)      0.4    0.500000  0.833333
3       (Bez)       (Süt)      0.4    0.666667  0.833333
4     (Ekmek)       (Süt)      0.6    0.750000  0.937500
5       (Süt)     (Ekmek)      0.6    0.750000  0.937500
```

Samsun Üniversitesi
Yazılım Mühendisliği Bölümü

VERİ MADENCİLİĞİ (DATA MİNİNG)

Dr. Öğr. Üyesi Alper Talha Karadeniz

2025-2026