Veri Madenciliğinde Birliktelik Kuralları ve Ardışık Zamanlı Örüntüler

Association Rules and Sequential Patterns in Data Mining

Koçak Mert GÜÇGÖTÜREN- KIRIKKALE ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSİ
kocakmertg@gmail.com
Alper Şahin ÖNER- KIRIKKALE ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSİ
alpersahinoner@gmail.com

Özet

Bilgi keşfi sürecinde,veri madenciliği aşamasına kadar veriler temizlenmiş,gerektiği yerde birleştirilmiş veya indirgenmiştir.Bunun sonucunda veri madenciliği ile bu veriler işleme tabi tutularak anlamlı sonuçlar çıkartılmalıdır.Birliktelik kuralları ve ardışık zamanlı örüntüler bu anlamlı sonuçlar çıkarma,modelleme yöntemlerinden biridir.Modellerin nasıl algoritmalarla verileri birbirinden ayırdığı veya birleştirdiği,neden niçin kullanıldığı,performansı ölçütleri bu makalede anlatılmıştır.

Anahtar Kelimeler: Veri, Veri Madenciliği, Veri Madenciliğinde Birliktelik Kuralları ve Ardışık Zamanlı Örüntüler, Apriori,Apriori-TID,FreeSpan,GSP

1-GİRİŞ

Şuan bulunduğumuz bu zamana baktığımızda sürekli verilerle temas halindeyiz.Günlük işlerimizde,karşıdan karşıya geçerken trafik ışıklarının rengini bildiğimizde veya internet ortamında yaptığımız her şeyde verileri kullanarak yaparız.Her alışverişte,her sosyal medya ortamına resim yüklediğimizde,her bankacılık işleminde bilgilerimiz bir veritabanına kaydedilmektedir.Baktığımızda birçok veri sürekli olarak depolanmak zorundadır.Fakat bu verileri işlemeye çalıştığımızda istenilen şekilde değerlendirilmemekte saniyeler içinde milyonlar hatta milyarlarca şekilde bilgi yığını olmaktadır.

Bu büyük verilerden anlamlı sonuçlar çıkararak bilgiyi keşfetmeliyiz.Veritabanlarının içerisinden anlamlı ve yararlı örüntülerin birbirlerine bağlı veya bağımsız olarak küçültmeli,hızlı sonuçlar almalıyız.Kullandığımız veri madenciliğinde de birliktelik kuralları ve ardışık zamanlı örüntüler diye adlandırdığımız veri madenciliği yöntemleri kullanılır. Kullanıcıların,müşterilerin veya önceden yapmış olduğu tercihlerlerle karşısına, birbirleriyle bağımlı sonuçlar çıkartma,birbirini izleyen dönemlerde belirli başlı olayların olması gibi. Kısaca şöyle örnek üzerinden anlatılırsa,alışveriş yaparken birbirleriyle bağımlı ürünler alınıyor mu? Ekmekle su,bebek beziyle bebek maması gibi birlikte seçilen ürünlerin tercih edilme olasılığı (birliktelik kuralları),yüksek sesli işyerlerinde çalışanların işitme kaybının da olacağının (ardışık zaman örüntüleri) bilinmesi. Bu çalışmada, veri madenciliği tekniklerinden birliktelik kuralları ve ardışık zamanlı örüntüleri kullanarak örnekler üzerinde durulmuş ve algoritmaları detaylandırılmıştır.

2-BİRLİKTELİK KURALLARI ,ARDIŞIK ZAMANLI ÖRÜNTÜLER VE ALGORİTMALARI

2.1.Birliktelik Kuralları ve Ardışık Zamanlı Örüntüler Nedir , Niçin Kullanılır ?

Birliktelik kuralları,birlikte yani eş zamanlı olarak birbirleriyle ilişki içinde olma anlamına gelmektedir.Yani bir eylem gerçekleştirildiğinde diğer bir eyleminde olmasıdır.Örneğin bir izleyici Netfilix üzerinden X filmini beğenip Y filmini de izlediyse,sonra gelen başka bir izleyici X filmini izleyince şirketin Y filmini "Bak insanlar bu filmide izlemiş."demesi gibidir.Aslında birbirleriyle bağımlı olan tercihlerin bir arada sunulması anlamını çıkarmak yanlış olmayacaktır.

Kullanılmasının sebebi ise sürekli büyüyen veritabanları yüzünden şirketler daha fazla ürün satmak daha fazla maliyet kazanmak için ellerinden gelen her şeyi yapabilmektedir.Verilerin içerisinden birbirleriyle bağımlı olarak verileri ortaya çıkarmak istediği için birliktelik kuralları olmak zorundadır.
Spotify kullanıcılara müzikleri tavsiye etmesi,Amazon'nun gelirlerinin neredeyse birçoğunu bu ilişkiye dayanarak elde ettiğini bilmemizde fayda olacaktır.

Ardışık zamanlı örüntüler ise birbirleriyle ilişkisi olup (eş zamansız) bir A olayı gerçekleştiğinde,başka bir B olayının A olayının olduğu duruma göre gerçekleşmesidir. Örneğin: Yol olmayan bir köye yol yaptıktan sonra araba almaya teşvik edilmesi,çekiç alan bir müşterinin çivi satın almaya teşvik edilmesi gibi

Ardışık zamanlı örüntüler için kullanılan algoritmalar; Spade,Spam,GSP,Free-Span (Frequent Pattern-Projected Sequential Pattern Mining), Prefix-Span (Prefix-Projected Sequential Pattern)

Birliktelik Kuralları Analizi için kullanılan bazı algoritmalar:
AIS,SETM,Apriori,Apriori-TID, Carma, Sequence, GRI, Eclat, FP-Growth ve diğerleri. Bunlar içerisinde popüler ve kullanılan **Apriori** Algoritmasıdır.

Bu yazımızda GSP,Free-Span,Apriori ve Apriori-TID algoritmaları anlatılacaktır.

2.2.Apriori Algoritması(Apriori Algorithm)

Apriori algoritması,veritabanlarında sık geçen veya tekrarlayan öğe kümelerinin keşfedilmesinde kullanılır.

Birliktelik kuralı madenciliği, tüm sık geçen öğelerin bulunması ve sık geçen bu öğelerden güçlü birliktelik kurallarının üretilmesi olmak üzere iki aşamalıdır. Birliktelik kuralının ilk aşaması için kullanılan Apriori Algoritması, sık geçen öğeler madenciliğinde kullanılan en popüler ve klasik algoritmadır. Bu algoritmada özellikler ve veri, Boolean ilişki kuralları ile değerlendirilir .[1]

2.2.1 Apriori Algoritması Örneği

X ürünü alan>Y ürününü alıyor
Y ürününü alan>X ürünü alıyorsa X⇔Y birbirleriyle bağımlı(birliktelik kuralına uyuyor.)
Ayrıca
X ürününü alınmış>Y ürününe bakabilirsin.[10]

Adım 1:K adet ürünler kümesi olsun.
Adım 2: Kullanıcılar, K (ürünler kümesi)içerisinden herhangi bir ürünü seçtiğinde en fazla satın alınandan en az satın alınana kadar bir sıralama yapılır.(Ürün-1<Ürün-2<…) Oluşturulan bu küme sayaç koyularak gerçekleştirilir ve adına da L kümesi diyelim.
Adım 3: Bu ürün sıralamasına aday olarak,yani alınan ürünlerin sayısının sıralamasını, önceden oluşturduğumuz L kümesi ile karşılaştırma yapacak ,bu listeyi değişiklik olduğunda güncel tutmak için ise aday kümesi oluşturulmalıdır.Bu kümeye C kümesi adını verelim.(C-Ürün-1<C-Ürün-2<…)
Adım 4:L kümesi(Önerilen ürünler) ile C kümesi(Aday olan ürünler) birleştirme yaparak elemanların birbirleriyle benzerliği aranır.Benzerlikten dolayı alınan öğeler için kümeler oluşturulur.
Adım 5:C kümesi içerisinden benzerliği olmayan elemanlar içerisinden hepsi silinir.Farklı bir ifade ile budama,değerlendirme dışı kalır.[2][3]

Method:

```
(1)   L1 = find_frequent_1-itemsets(D);
(2)   for (k = 2; Lk−1 ≠ φ; k++) {
(3)       Ck = apriori_gen(Lk−1);
(4)       for each transaction t ∈ D { // Sayım için D taraması
(5)           Ct = subset(Ck, t); // Adayların t alt kümelerini elde et
(6)           for each candidate c ∈ Ct
(7)               c.count++;
(8)       }
(9)       Lk = {c ∈ Ck | c.count ≥ min_sup}
(10)  }
(11)  return L = ∪k Lk;

procedure apriori_gen(Lk−1:frequent (k − 1)-itemsets)
(1)   for each itemset l1 ∈ Lk−1
(2)       for each itemset l2 ∈ Lk−1
(3)           if (l1[1] = l2[1]) ∧ (l1[2] = l2[2])
                  ∧... ∧ (l1[k − 2] = l2[k − 2]) ∧ (l1[k − 1] < l2[k − 1]) then {
(4)               c = l1 ⋈ l2; // Birleştirme adımı: Adayları oluştur
(5)               if has_infrequent_subset(c, Lk−1) then
(6)                   delete c; // Budama adımı: sık olmayan adayları sil
(7)               else add c to Ck;
(8)           }
(9)   return Ck;

procedure has_infrequent_subset(c: candidate k-itemset;
          Lk−1: frequent (k − 1)-itemsets); //Ön bilgileri kullan listede var mı?
(1)   for each (k − 1)-subset s of c
(2)       if s ∉ Lk−1 then
(3)           return TRUE;
(4)   return FALSE;
```

Şekil 4.16 Sık öğeler kümesini keşfetmek için Apriori algoritması

#[5] Şekil 4.16 Apriori Algoritması

2.2.2. Apriori Problem Avantaj ve Dezevantajları

Tercihlerin birbiriyle bağlantılı olması şirketler açısından maliyeti artıracak,kolay ve anlaşılır sonuçları üretmeyi sağlayacaktır.[4]Ayrıca bununla beraber ortaya çıkarılan ürünlerin verimli bir şekilde kullanılması sağlanacaktır.Fakat sık geçen öğe kümelerini bulmak için birçok kez veritabanının tarandığı için büyük veri setlerinde problem çıkarabiliceği gibi kayıtlarda da çok az rastlanan tercihleri yok sayabilme durumu oluşaçaktır.

2.3. Apriori-TID Algoritması

Apriori aday değerlerini saymak,kontrol etmek,sıralama yapmak için her geçişte tüm veritabanını taramak zorundadır.Bu taramayı yapmayalım her geçişte sıralama zaman alacaktır diye veritabanını kullanmadan işlem gerçekleştirilmiştir.Bu işlem her geçişte şifrelenmiş kümeler olacaktır.

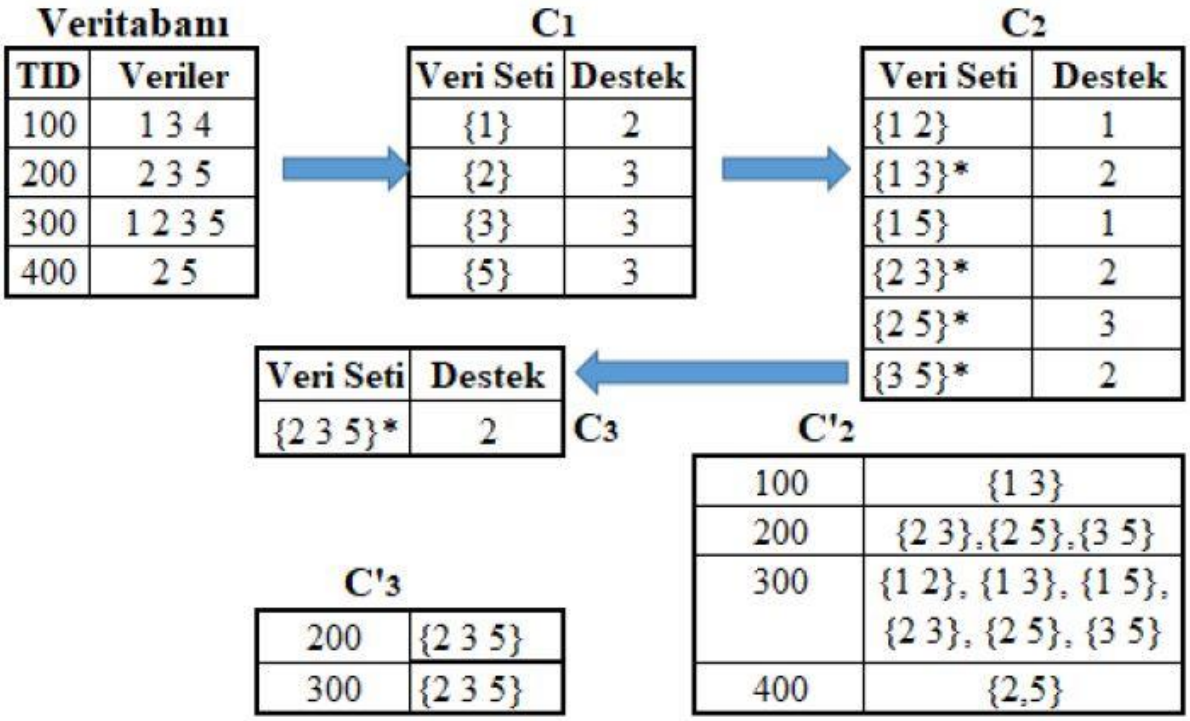
Bir kere veritabanını tarıyarak ve <TID,c> şeklinde c sürekli değiştirilir.TID tanımlayıcı,c ise değerimizdir(<Film-1,5> şeklinde).Oluşan veri kümesinin büyüklüğü küçülmüş olur.

İlk olarak, tüm veritabanı taranır ve C_1 nesne kümeleri elde edilir. C_1 'in her bir kaydı tüm nesneleri TID'leriyle birlikte içerir. 1-nesneli L_1 yaygın nesnekümeleri C_1 'deki kayıtlar sayılarak hesaplanır. Daha sonra C_2 'yi elde etmek için apriori_gen() fonksiyonu kullanılır. Bir T hareketine ilişkin C_2 'deki kayıtlar, T'de bulunan C_2 'nin üyelerinden elde edilir. Bunu gerçekleştirmek için tüm veritabanı yerine C_1 taranır. Sonra, C_2 'deki destek değerleri sayılarak L_2 elde edilir. Bu işlem, boş aday nesnekümeleri oluşuncayadek sürer. [4]

2.3.1. Apriori-TID Algoritması Örnek Üzerinde Anlatımı

1) $L_1 = \{\text{large 1-itemsets}\}$;
2) $\overline{C}_1 = \text{database } \mathcal{D}$;
3) **for** ($k = 2$; $L_{k-1} \neq \emptyset$; k++) **do begin**
4) $C_k = \text{apriori-gen}(L_{k-1})$; // Yeni adaylar
5) $\overline{C}_k = \emptyset$;
6) **forall** entries $t \in \overline{C}_{k-1}$ **do begin**
7) // tanımlayıcı t.TID ile işlemin içerdiği C_k içindeki aday öğeleri belirler
$C_t = \{c \in C_k \mid (c - c[k]) \in t.\text{set-of-itemsets} \wedge (c - c[k-1]) \in t.\text{set-of-itemsets}\}$;
8) **forall** candidates $c \in C_t$ **do**
9) c.count++;
10) **if** $(C_t \neq \emptyset)$ **then** $\overline{C}_k$ += $< t.\text{TID}, C_t >$;
11) **end**
12) $L_k = \{c \in C_k \mid c.\text{count} \geq \text{minsup}\}$
13) **end**
14) Answer = $\bigcup_k L_k$;

#[5] Şekil 4.19 Apriori-TID algoritması



#[5] Şekil 4.20 Apriori-TID Algoritması aşamalarının uygulamalı gösterimi

İlk başta veritabanı taranarak veriler üzerinden 1 değerini içeren sayısı,2 değerini içeren sayısı… hesaplanarak C1 kümesi elde edilir.Burada veri seti bir kez taranmış ve verilerin kimlerle birlikte kullanıldığı bilinmiştir.

Oluşan C1 kümesi 1-ürünlü büyük veri kümeleridir.Bu adımdan sonra C2'yi üretmek için apriori_gen algoritması kullanılır.[5]

C2 için girilen değerler C1 ' de mevcut olan değerlerin sırayla (Apriori algoritmasından dolayı yukarıdan aşağı doğru sıralanmış)ilişkisiyle oluşturulur.1>2 1>3,1>5,2>3,2>5,3>5.Bu ilişkilendirme şöyle ifade edilirse daha kolay anlaşılabilir.1 verisini içerenlerde 2 olması,1 verisini içerenlerde 5 olması…
Şartları yerine getiren yukarıdaki Şekil 4.20 de {1,3}{2,3}{2,5}{3,5} destek(aday) olan değerleri veritabanında içerenler * işaretiyle gösterilmiş ve diğerleri göz ardı edilmiştir.
Devamında oluşan C3 kümesinde sıralama devam ettiğinden {2,3,5} alınarak 2 tane daha destek(aday) bulunmuştur.
Sonucunda C'2 ,C'3 oluşmuş ve birbirleriyle ilişkili öğeler çıkarılmıştır.

Bu işlem aday ürün kümeleri boş küme olarak bulununcaya kadar devam ettirilir. Bu kodlama işlevinin kullanılmasının avantajı, sonraki döngülerde kodlama işlevinin büyüklüğünün veritabanından daha küçük olmasıdır. Böylece veri okuma işlemi daha az çabayla gerçekleşir ve daha kısa sürer. [5]

2.3.2.Apriori-TID Algoritması Problem Avataj ve Dezevantajları

Apriori,adayların sayısını bulmak için tüm veritabanını sürekli taradığından Apriori-TID algoritması farklıdır.Performans olarak tarama işlemlerinden dolayı Apriori – TID performansı daha iyidir.

Bu şifreleme fonksiyonunu kullanmanın avantajı, sonraki geçişlerde şifreleme fonksiyonunun büyüklüğü veritabanına oranla daha küçük olacağından okuma süresi azalır.[4]

2.3.3.FreeSpan(Örnek Üzerinden Algoritma Anlatım)-Ardışık Zamanlı Örüntü

SID	Sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<(eg(af)cbc>

Yandaki veritabanında SID:Tanımlayıcı, Sequence:Veriler'dir
Adım 1:Verilen sıralı veritabanında min_support=2(Aday olarak gösterilmesi için en az ziyaret edilmesi gereken sayı) belirlenir.
Adım 2:*f_list* adında verilerin sıralı olarak kaç defa geçtiği sayaç yardımıyla hesaplanır.
f_list = a:4,b:4,c:4,d:3,e:3,f:3; ~~g:1~~
(a ilk satırda,ikinci,üçüncü ve dördüncü satırda geçtiği için sayaç 4 olarak alınmıştır.g min_support değerinden küçük yani sık ziyaret edilmediği için silinmiştir.Diğer verilerin hesaplaması bu mantıkla hesaplanmıştır.)
Adım 3:Sıralı olarak alınan f_list listesinde ilk eleman ve son eleman haricinde aşağıdaki mantıkla kalanları ayrık kümesi oluşturulur.6 tane ayrık küme oluşmuş olacak.

-Yalnız *a* içerenler
-Yalnız *b* içerenler ve *b* den sonra eleman kabul etmeyenler.
-Yalnız *c* içerenler ve *c* den sonra eleman kabul etmeyenler.
-Yalnız *d* içerenler ve *d* den sonra eleman kabul etmeyenler.
-Yalnız *e* içerenler ve *e* den sonra eleman kabul etmeyenler.
-İçerisinde f olanlar.

Bu şekilde veritabanı içerisinden küçük kümeler oluşturulmuş olur.
Adım 4:Oluşturulan kümelere elemanlar taranarak sırasıyla yerleştirilir.

Sequence id	Sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<e(af)cbc>

=>

{a}-projected database

10	<aaa>
20	<aa>
30	<a>
40	<a>

Frequent Patterns
<a> <aa>

-Yalnız *a* içerenler veritabanında taranır.Sonuçta iki farklı örüntü ortaya çıkar(<a>,<aa>).

Sequence id	Sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<e(af)cbc>

=>

10	<a(ab)a>
20	<aba>
30	<(ab)b>
40	<ab>

Frequent Patterns
<b> <ab> <ba> <(ab)>

-Yalnız *b* içerenler ve *b* den sonra eleman kabul etmeyenler için;
–<b> **-taranıp tahmin edilenler veritabanından alınır.**
• 10:<a(ab)a>, 20:<aba>, 30:<(ab)b>, 40:<ab> (a→b den önce kabul edildiği için b den sonrasında kullanımı dikkate alınmaz.)
Toplam 4 tane oldu a ve b birbirleriyle bağlantılı olmuş oldu.
Adım 5:Diğer kümeler hesaplandıktan sonra birbirleriyle benzer olan (Frequent Pattern) örüntüler dikkate alınır.

Sequence id	Sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<e(af)cbc>

=>

{c}-projected database

10	<a(abc)(ac)c>
20	<ac(bc)a>
30	<(ab)cb>
40	<acbc>

Frequent Patterns
<c> <ac> <bc> <(bc)> <ca> <cb>
<(ab)c> <acc> <acb>

Sonucunda <c>,<ac>,<bc>,<(bc)>,<ca>,<cb>,<(ab)c>,<acc>,<acb> birbirleriyle ardışık zamanlı olarak örüntü kurmuş olur.

2.3.4. GSP(Generalized Sequence Pattern) Algoritması-Ardışık Zamanlı Örüntü

Genelleştirilmiş ardışık desen,veri madenciliği için oluşturulmuştur.Algoritmanın başlangıcı temel olarak Apriori Algoritması ile aynıdır.Yani veritabanımızda tarama yaparken minumum destek(aday olarak gösterilen) alanları buluyoruz.

İsterseniz kaynağını aşağıda vermiş olduğum örnekten yola çıkarak anlatıyım.

Transaction Date	Customer ID	Items Purchased	Customer Sequence
1	01	A	<AB(FG)CD>
3	01	B	
7	01	FG	
9	01	C	
10	01	D	
1	02	B	<BGD>
4	02	G	
6	02	D	
1	03	B	<BFG(AB)>
5	03	F	
8	03	G	
9	03	AB	
2	04	F	<F(AB)CD>
6	04	AB	
8	04	C	
10	04	D	
3	05	A	<A(BC)GF(DE)>
4	05	BC	
7	05	G	
9	05	F	
10	05	DE	

Yandaki tabloda CustomerID sütününda müşterileri bilgileri,A,B,C,D,F,G ise aldığı ürünleri göstermektedir.
Örnegin <BGD> ile ifade edilen B ürününden sonra G ve D ürünleri alınmış demek.

Adım 1:Bütün ürünler alınan ürün(tekrarlanan) sayısı kadar sayaçla sayılır.Apriori algoritmasında olduğu gibi min_support=2(aday olarak gösterilen) seçilir.İsterseniz değiştirebilirsiniz.
Yukarıdaki verileri kullanacağız ve bu durumda minumum desteğin 2 olduğunu söyleyeceğiz.[6]

Item	Support
A	4
B	5
C	3
D	4
~~E~~	~~1~~
F	4
G	4

E ürünü yeterli aday gösterilmediği için biz 2 den düşükleri alma dediğimiz için alınmayacaktır.

	A	B	C	D	F	G
A	AA	AB	AC	AD	AF	AG
B	BA	BB	BC	BD	BF	BG
C	CA	CB	CC	CD	CF	CG
D	DA	DB	DC	DD	DF	DG
F	FA	FB	FC	FD	FF	FG
G	GA	GB	GC	GD	GF	GG

Bu öğelere sahip olduktan sonra, bu öğelerden daha büyük desenler oluşturma ve destek olup olmadığını kontrol etme yinelemeli işleme başlarız. Burada işler gerçekten ilginçleşiyor. Apriori'de olası toplam kombinasyonlar AB, AC, AD, AF, AG, BC, BD, BF, BG, CD, CF, CG, DF, DG, FG olacaktır. Diziler oluştururken, sipariş önemli olduğundan neredeyse yeterli değildir VE tek öğelerin tekrarlandığı dizilere sahip olabiliriz (örneğin A aldım, ertesi gün tekrar A aldım). Bu yüzden şuna benzer tablolar oluşturmalıyız. [6]

	A	B	C	D	F	G
A		(AB)	(AC)	(AD)	(AF)	(AG)
B			(BC)	(BD)	(BF)	(BG)
C				(CD)	(CF)	(CG)
D					(DF)	(DG)
F						(FG)
G						

Sonra alınan ürünlerin olmadığı hücreler boş bırakılır.

Köşegen değerler burada boştur, çünkü zaten ilk 2 maddelik tabloda yer alırlar. Anladığınızdan emin olmak için, (AA) orada değildir, çünkü A ve A aynı işlemde satın alınırsa, yalnızca bir kez sayılır, bu nedenle aynı öğeden ikisini parantez içinde asla bir araya getirmeyiz. Sol alt da boştur, ancak bunun nedeni, her zaman birlikte satın alınan ürünleri artan sırada listelemekten ibaret olduğum konvansiyondan kaynaklanmaktadır.[6]

Oluşturulan tablodaki değerleri(ürünleri) tekrar tarıyoruz.

AA	AB	AC	AD	AF	AG
<AB(FG)CD>	<AB(FG)CD>	<AB(FG)CD>	<AB(FG)CD>	<AB(FG)CD>	<AB(FG)CD>
<BGD>	<BGD>	<BGD>	<BGD>	<BGD>	<BGD>
<BFG(AB)>	<BFG(AB)>	<BFG(AB)>	<BFG(AB)>	<BFG(AB)>	<BFG(AB)>
<F(AB)CD>	<F(AB)CD>	<F(AB)CD>	<F(AB)CD>	<F(AB)CD>	<F(AB)CD>
<A(BC)GF(DE)>	<A(BC)GF(DE)>	<A(BC)GF(DE)>	<A(BC)GF(DE)>	<A(BC)GF(DE)>	<A(BC)GF(DE)>

(AB) için 1 ve 3 müşteri almış oluyor.A ürününü aldıktan sonra B ürününü almış diyoruz.Burada önemli bir nokta var B ürününden sonra A gelmesi (BA),(AB) den farklıdır. Aynı adımlar AC,AD,AF,AG,… şeklinde tarayarak veritabanında birbiri ardına gelen değerleri alıyoruz.min_support=2 aday'ından küçük olanları almıyoruz.

~~AA~~	~~0~~
~~BA~~	~~1~~
~~CA~~	~~0~~
~~DA~~	~~0~~
FA	2
~~GA~~	~~1~~

AB	2
~~BB~~	~~1~~
~~CB~~	~~0~~
~~DB~~	~~0~~
FB	2
~~GB~~	~~1~~

AC	3
BC	2
~~CC~~	~~0~~
~~DC~~	~~0~~
FC	2
~~GC~~	~~1~~

AD	3
BD	4
CD	3
~~DD~~	~~0~~
FD	3
GD	3

AF	2
BF	3
~~CF~~	~~1~~
~~DF~~	~~0~~
~~FF~~	~~0~~
~~GF~~	~~1~~

AG	2
BG	4
~~CG~~	~~1~~
~~DG~~	~~0~~
~~FG~~	~~1~~
~~GG~~	~~0~~

(ör. (AB)). Bu biraz daha kolaydır, çünkü veritabanımızda aynı anda 2 ürün satın alındığında sadece 5 kez vardır. (FG), (AB), (AB), (BC) ve (DE) idi. Sadece (AB) iki kez görünür, bu yüzden onu tutarız ve geri kalanından kurtuluruz.[6]

AB	AC	AD	AF	AG	BC	BD	BF	BG	CD	FA	FB	FC	FD	GD	(AB)

2 değerlileri şimdi 3 değerli dizilerle arayarak birbirleriyle ilişkili olma durumlarına bakalım.

2-seq	-1st	-Last
AB	B	A
AC	C	A
AD	D	A
AF	F	A
AG	G	A
BC	C	B
BD	D	B
BF	F	B
BG	G	B
CD	D	C
FA	A	F
FB	B	F
FC	C	F
FD	D	F
GD	D	G
(AB)	B	A
	A	B

Kalan tüm sekansları son adımdan (AB, AC, AD, AF, AG, BC ...) alırsınız, ardından ilk öğeyi sekanstan kaldırırsınız. AB için A'yı kaldırırız ve B ile kalırız. Tüm diziler için bunu yapın (aşağıdaki tabloda 2. sütuna bakın).[6]

Sonra aynı şeyi yaparsınız, ancak son öğeyi kaldırın (aşağıdaki tabloda 3. sütun). Bu, (AB) gibi birden fazla ürün satın alımında sona eren bir sekansla uğraşmanız dışında oldukça basittir. Bu olduğunda, olası tüm öğeleri birer birer çıkarmanız ve ayrı ayrı ele almanız gerekir. Dolayısıyla, (AB) 'den "ilk" öğeyi kaldırıyorsanız, A'yı kaldırır ve B artıklarını alırsınız ve sonra B'yi kaldırır ve A artıklarını alırsınız. Bu yüzden yukarıdan başlıyorsak, 2 maddelik AB dizisinin ABC, ABD, ABF, ABG ve A (AB) oluşturmak için BC, BD, BF, BG ve (AB) ile eşleştiğini görüyoruz. [6]

Birbirleriyle eşleşmeyen tüm öğeler birer birer çıkarılır.Ve oluşturulan 2 lik ürün tablosunda yukarıdan aşağıya doğru hücreler birleştirilir.

2-seq. (1)	2-seq. -1st	2-seq. (2)	2-seq. -Last	3-seq after join	3-seq. after prune	Support Count	3-seq. Supported
AB	B	BC	B	ABC	ABC	1	
AB	B	BD	B	ABD	ABD	2	ABD
AB	B	BF	B	ABF	ABF	2	ABF
AB	B	BG	B	ABG	ABG	2	ABG
AB	B	(AB)	B	A(AB)			
AC	C	CD	C	ACD	ACD	3	ACD
AF	F	FA	F	AFA			
AF	F	FB	F	AFB	AFB	0	
AF	F	FC	F	AFC	AFC	1	
AF	F	FD	F	AFD	AFD	2	AFD
AG	G	GD	G	AGD	AGD	2	AGD
BC	C	CD	C	BCD	BCD	2	BCD
BF	F	FA	F	BFA			
BF	F	FB	F	BFB			
BF	F	FC	F	BFC	BFC	1	
BF	F	FD	F	BFD	BFD	2	BFD
BG	G	GD	G	BGD	BGD	3	BGD
FA	A	AB	A	FAB	FAB	0	
FA	A	AC	A	FAC	FAC	1	
FA	A	AD	A	FAD	FAD	1	
FA	A	AF	A	FAF			
FA	A	AG	A	FAG			
FA	A	(AB)	A	F(AB)	F(AB)	2	F(AB)
FB	B	BC	B	FBC	FBC	1	
FB	B	BD	B	FBD	FBD	1	
FB	B	BF	B	FBF			
FB	B	BG	B	FBG			
FC	C	CD	C	FCD	FCD	2	FCD
(AB)	B	BC	B	(AB)C	(AB)C	1	
(AB)	B	BD	B	(AB)D	(AB)D	1	
(AB)	B	BF	B	(AB)F	(AB)F	0	
(AB)	B	BG	B	(AB)G	(AB)G	0	
(AB)	A	AB	A	(AB)B			

A (AB) olanı aldatıcıdır çünkü parantez içindeki düzenin sadece konvansiyon olduğunu ve B'nin eşleştiğini görmeyi kolaylaştıran kolayca A (BA) yazılabileceğini hatırlatmanız gerekir. Tablonun geri kalanında bu şekilde çalışarak (yinelenen adaylar oluşturmamaya dikkat ederek) aşağıdaki tabloda "3-seq birleştirildikten sonra" sütununu dolduruyoruz. Bir sonraki adım, bu 3 maddelik dizileri budamaktır. Birçoğu, 2 maddelik dizilerin önceki turunda desteklenmeyen 3 maddelik dizilerinin bölümlerine sahiptir. Bunun iyi bir örneği aday dizi

AFA'dır. AF ve FA 2 öğeli dizileri destekler, ancak AA desteklenen 2 öğeli bir dizidir, bu yüzden bu budanır. [6]
Bu şekilde tarama yaparak 4,5… adımlarda yapılabilir.Ama yapılması daha düşük sonuçlar ortaya çıkaracak ve doğruluk değerinden sapılacaktır.

2.3.4.1.GSP Performas ölçütü

Destek eşiğini karşılayamayanları her adımda attığımız için sürekli veritabanından tarama yapmadığımızdan,zamandan tasarruf ve verimli sonuçlar elde etmemizi sağlamıştır.

3.Birliktelik Kuralları ve Ardışık Zamanlı Örüntüler Performans Ölçütleri

Performans ölçütlerini bir örnek üzerinden anlatmak gerekirse.
Market Sepet Analizi
Bir ürünü aldığımızda başka bir ürünüde almamız veya almaya zorunlu bırakılmamızdır.Market sepet analizi yapan şirketler bu zorunluluğu bize dayatarak ürün aldırmaya zorlamışlardır.Döner yanında ayran içmek,film izlerken mısır yemek tam bu açıklamaya örnek olacaktır.Bu analizde şirket yöneticileri,müşterilerin birlikte satın almaya daha eğilimli oldukları ürünleri birbirlerine yakın raflara koyması satışlarını artırmıştır.

Birliktelik kuralları ve ardışık zamanlı örüntüler ise bu ürünlerden hepsini bir küme halinde bulunması,sık geçen ürün kümelerinden güçlü ilişki kurallarının kurulması işini gerçekleştirir. Bu kurallar minumum destek(aday olabilecek ürünler) ve minumum güven durumunu sağlamalıdır.
Bu analizde,satılan ürünler arasındaki ilişkileri ortaya çıkarmak için:
1-Destek(Support)
2-Güven(Confidence) ölçütleriden yaralanılır.
Bu ölçütlerin büyük olması ürünlerin birbirleriyle ilişkisinin güçlü olduğu anlamını çıkaracaktır. [7][8]
Bu ölçütlerin hesaplanmasında destek sayısı adı verilen değer kullanılır.[7][8]

Destek ölçütü:destek(A→B)=sayı(A,B)/Tüm alışverişlerin sayısı

A ürününü alan B ürünün alıyor ve tüm müşterilerde bu oran yüksek çıkıyorsa A ve B ürünleri birbirlerine bağlı oluyor.

Güven ölçütü:güven(A→B)=sayı(A,B)/sayı(A)

A ürününü satın alan müşterilerin B ürününü alma olasığı bizim güven ölçütümüz olur.Biz ürün alırken ürünlerin güvenilirliğine göre almayı tercih ederiz.

4.SONUÇ

Birliktelik kuralları geçmişte yapmış olunan tercihlere göre eş zamanlı olarak geleceğe dair tercihler,öneriler sunması iken ardışık zamanlı örüntüler eş zamansız yani birbirleriyle bağlı olup belirli bir süre geçmesi veya geçmemesidir.
İki modelde de kullanılan algoritmalarda bu verilerin birbirlerine bağlı olması veya olmaması dikkate alınmıştır.Bağlı olanların ne sıklıkla birbirlerine bağımlı olduğu sonuca etkisinin nasıl değiştirdiği gözlenmiştir.
Ayrıca kullanılan veritabanının boyutlarına göre algoritmaların seçilmesi gereklidir. Performans açısından çok kullanılan algoritmalar bazen doğru sonuçlar vermeyebilir.Bu yüzden veritabanın boyutuna ve içerisindeki verilerin birbirleriyle ilişkisine bakarak uygun algoritmayı bulmak mantıklı olacaktır.

Kaynaklar

[1]Gao, W., (2004), “A Hierarchical Document Clustering Algoritm”, MSc Thesis, Dalhousie University, Halifax, Nova Scotia.

[2]Özçakır, Feridun Cemal, and Ali Yılmaz Çamurcu. "Birliktelik kuralı yöntemi için bir veri madenciliği yazılımı tasarımı ve uygulaması." (2007).

[3]Birant, Derya, et al. "İş Zekası Çözümleri için Çok Boyutlu Birliktelik Kuralları Analizi." *Akademik Bilişim* 10 (2010): 256.

[4]Yüksel, Tuğçe, and Metin Zontul. "Dağıtık Sistemlerde Birliktelik Kuralları İle Sepet Analizi." *Aurum Mühendislik Sistemleri Ve Mimarlık Dergisi* 3.1 (2019): 65-77.

[5]Zerman, Mesut. *Birliktelik kuralı algoritmaları ile büyük veriler üzerinde analitik analizler: Havaalanı örneği*. Diss. Fen Bilimleri Enstitüsü, 2018.

[6] Generalized Sequential Pattern (GSP) Mining, Tuesday, March 10, 2015 http://simpledatamining.blogspot.com/2015/03/generalized-sequential-pattern-gsp.html

[7]Birant, Derya, et al. "İş Zekası Çözümleri için Çok Boyutlu Birliktelik Kuralları Analizi." *Akademik Bilişim* 10 (2010): 256.

[8] *Birliktelik Kuralları Kullanılarak Pazar Sepeti Analizi (Market Basket Analysis Using Association Rules)* Nov 24, 2016 https://www.slideshare.net/uslumetin/birliktelik-kurallar-kullanlarak-pazar-sepeti-analizi-market-basket-analysis-using-association-rules

[9] KARABATAK, Murat, and Melih Cevdet İNCE. "APRIORI ALGORİTMASI İLE ÖĞRENCİ BAŞARISI ANALİZİ." (2004).

[10] Han, Jiawei, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.