Birliktelik Kuralı Analizi İçin FP-TREE Algoritmasını Kullanarak Optimize Edilmiş Bir Algoritma

Meera Narvekara , Shafaque Fatma Syedb

HAZIRLAYANLAR:

BEYZA NUR HORASAN-180303049

ZEHRA İŞÇİ-180303018

ASSOCIATION
RULE MINING

ÖZET

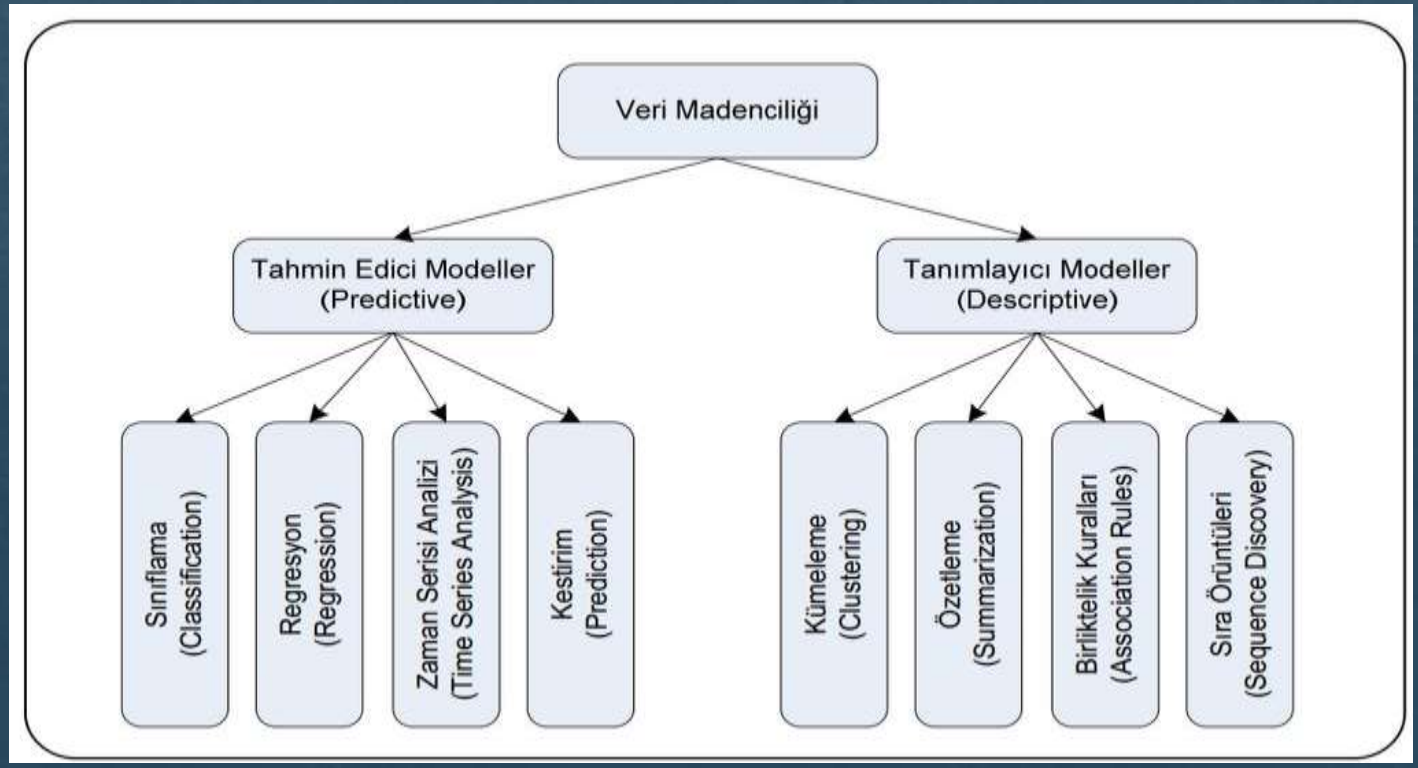
Veri madenciliği, istenen bilgiyi elde etmek için veri tabanında depolanan verilerin büyük boyutuyla başa çıkmak için kullanılır. Verilerin çıkarılması için çeşitli teknikler vardır.

Bunlardan bir tanesi de FP-growth algoritmasıdır. İstenen kuralları bulmada en verimli algoritmadır. İşlem için veri tabanını iki kez tarar. FP-growth algoritmasındaki sorun, çok sayıda koşullu FP ağacı oluşturmasıdır.

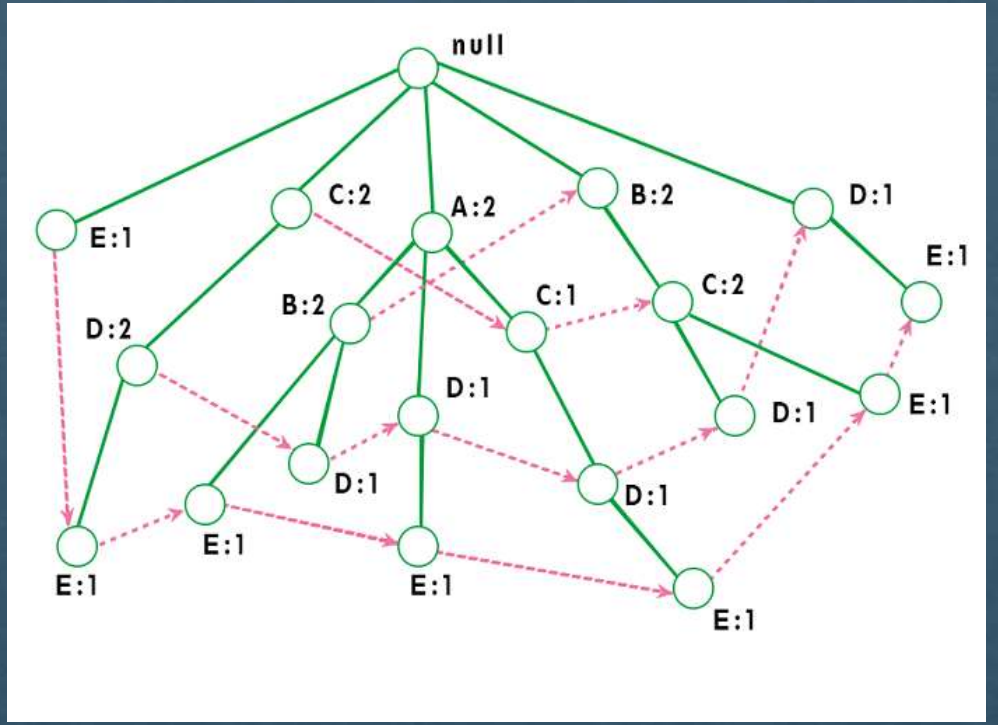
Önerilen çalışmada koşullu FP ağaçları oluşturmadan tüm sık kullanılan öge kümelerini ortaya çıkaran yeni bir teknik tasarladık. FP ağacının aksine, veri tabanını yalnızca bir kez tarar ve bu da algoritmanın zaman verimliliğini azaltır. Ayrıca, istenen ilişkilendirme kurallarını bulmak için sık kullanılan öğe kümelerinin sıklığını da bulur.

1.GİRİŞ

Veri madenciliği, veri ambarlarında ve veri tabanlarında saklanan çok büyük miktarda veriyle uğraşmak, istenen ilgi çekici bilgileri bulmak için kullanılır. Birliktelik kuralları, karar ağaçları, sinir ağları vb. birçok veri madenciliği tekniği önerilmiştir. En iyi bilinen tekniklerden biri birliktelik kuralı madenciliğidir. En verimli veri madenciliği tekniğidir. Verilerin farklı nitelikleri arasındaki ilişkiyi bulmaktan sorumludur.

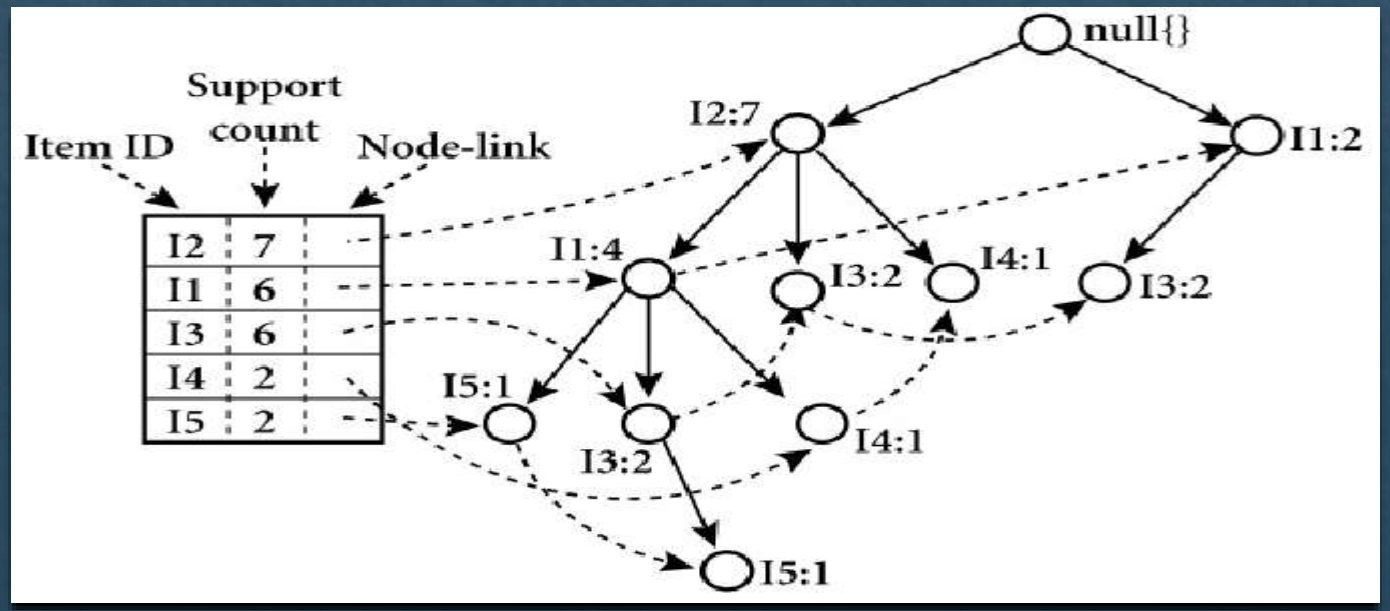


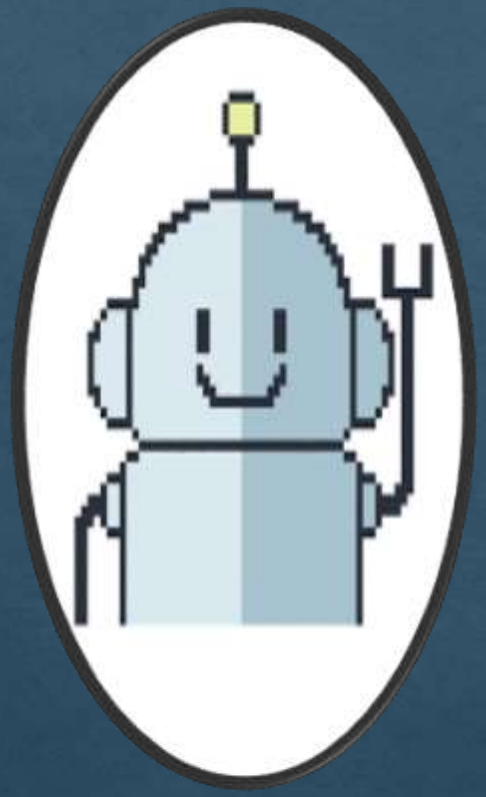
Birliktelik kuralları ilk olarak sonuçları "if-then" ifadeleri şeklinde sunar. Bu kurallar girdi veri kümelerinden oluşturulur. Kurallar, kullanıcıdan girdi olarak verilen destek ve güven değerinden türetilir. Bir birliktelik kuralı X→Y şeklinde ifade edilir. Burada X önce ve Y sonuçtur. Birliktelik kuralı, destek ve güven değerine bağlı olarak X zaten oluşmuşsa, Y'nin kaç kez oluştuğunu gösterir. Birliktelik kuralları oluşturmak için birçok algoritma zaman içinde sunulmuştur. Bazı iyi bilinen algoritmalar Apriori ve FP-Growth'tur.



Agrawal ve diğerleri tarafından 1994 yılında geliştirilen Apriori algoritması, veri madenciliği tarihinde birliktelik kurallarının çıkarılması konusunda elde edilmiş büyük bir başarıdır. Birliktelik kuralları çıkarımında en çok bilinen algoritma olmuştur. Algoritmanın ismi, yaygın nesnelerin önsel bilgilerini kullanmasından yani bilgileri bir önceki adımdan almasından "önceki (prior)" anlamında aprioridir. Bu teknik, yaygın bir nesne kümesinin tüm altkümeleri de yaygın olmalıdır kuralına dayanmaktadır. Ayrıca, bir nesne kümesindeki nesnelerin sözlük sıralı olduğunu varsayar. Öğe kümelerinin desteğini saymak için bir genişlik öncelikli arama tekniği kullanır

FP-growth, aday üretmeksizin yaygın nesne kümelerinin bulunması için geliştirilmiş bir metoddur. İlk olarak veri tabanı, yaygın nesneleri temsil edecek şekilde FP-Tree (Frequent Pattern Tree – FP-Ağacı) denilen ağaç yapısına sıkıştırılır. Bu ağaçta nesne kümelerinin birliktelik bilgileri yer almaktadır. Daha sonra, sıkıştırılmış veri tabanı şartlı veri tabanlarına bölünür. Her biri yaygın bir nesne ile ilişkilendirilmiştir ve bu veri tabanları ayrı ayrı madenlenir.





Etkili bir birliktelik kuralı madenciliği tekniği öneriyoruz. Bunun en büyük avantajı, tüm sık kullanılan öge setlerini daha az çabayla elde edebilmemizdir. Önerilen teknik ayrıca veri tabanını yalnızca bir kez tarar. Herhangi bir koşullu FP ağacı oluşturmadan sık kullanılan öğe kümelerini oluşturur.

ASSOCIATION RULE MINING

2.LİTERATÜR TARAMA

Apriori algoritmasını kullanarak ilişkilendirme madenciliği iyi kurallar verir, ancak veri tabanı boyutu arttığında performans düşer. Bunun nedeni, işlemi her taradığında tüm veri tabanını taraması gerektiğidir. Apriori algoritması, geniş öğe kümelerini elde etmek için geniş bir ilk arama tekniği kullanır. Bu algoritmanın sorunu, doğrudan büyük veri tabanlarına uygulanamamasıdır. Diğer iyi bilinen algoritma, Sık Model (FP) büyüme algoritmasıdır. Böl ve fethet tekniğini kullanır. FP, sık örüntülerden oluşan bir ağaç oluşturur ve sık kullanılan öğe kümelerini hesaplar. FP büyümesi ile Apriori algoritmasını karşılaştırdığımızda, verimlilik açısından FP çok daha üstündür. Ancak FP'nin dezavantajı, çok sayıda koşullu FP-Ağacı üretmesidir.

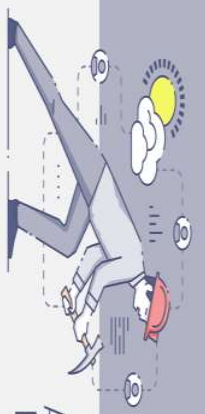
ASSOCIATION RULE MINING

Yan Hu, maksimum sık öğe kümelerini çıkarmak için optimize edilmiş bir algoritma önerdi.

İşlem sırasında sık sık öğe madenciliği yaparken, bazen çok sayıda aday öğe seti ile uğraşmamız gerekir. Ancak, sık öğe kümeleri yukarı doğru kapalı olduğundan, yalnızca tüm maksimum sık öğe kümelerini (MFI) keşfetmek yeterlidir. Sık kullanılan bir üst küme yoksa sık öğe kümesi maksimumdur. Yapı, FP max'ın alt küme kontrol sayısını azaltmasına ve arama süresinden tasarruf etmesine yardımcı olur. Bu algoritmada, belirli koşulları sağlayan bazı koşullu örüntü tabanları için, karşılık gelen maksimum sık öğe kümelerinin mümkün olduğu kadar erken elde edilebileceği düşünülmektedir. Bu, aşağıdaki FP-ağacı yapımında tüketimi azaltacak ve madencilik verimliliğini artıracaktır.

ASSOCIATION RULE MINING

Li Juan QFP algoritmasını önerdi. Önerilen QFP algoritması, veri tabanını yalnızca bir kez tarar QFP algoritması, veri ön işlemesinden sonra işlem veri tabanını bir QFP ağacına dönüştürebilir ve ardından ağacın birliktelik kuralı madenciliğini yapabilir. QFP algoritması, FP büyüme algoritmasından daha fazla bütünlüğe sahiptir ve sık örüntüleri araştırmak için tüm bilgileri korur; herhangi bir işlemin uzun modelini yok etmeyecek ve alakasız bilgileri önemli ölçüde azaltacaktır. QFP algoritmasının girdisi, FP büyüme algoritması veya Apriori algoritmasının girdisi ile aynıdır, bu nedenle QFP algoritması, FP büyüme algoritması veya Apriori algoritması için uygun olabilir.



ASSOCIATION RULE MINING

Jun Tan Etkili bir kapalı sık örüntü madenciliği algoritması, sık öğe kümeleri madencilik algoritma temelli FP büyüme algoritması önerir. Algoritmada yeni bir FP-dizi teknolojisi önerilmiştir, FP-dizisi ile kombinasyon halinde olan FP-ağacı veri yapısının bir varyasyonu kullanılmıştır. Bir işlem veri tabanı çok yoğun olduğunda, yani veri tabanı çok sayıda uzun sık öğe kümesi içerdiğinde, tüm sık öğe kümelerini araştırmak iyi bir fikir olmayabilir.

Bu algoritma, seyrek olan veri tabanlarında iyi çalışır. Syed Khairuzzaman Tanbeer, dinamik ağacı yeniden yapılandırma konseptini tanıttı. Yeniden yapılandırma tekniğini dinamik olarak uygulayarak, tek geçişli bir frekans azalan önek ağacı yapısı elde eden ve bir veri kümesinden sık örüntüleri keşfetmek için madencilik süresini önemli ölçüde azaltan CP ağacını önermişti. CP ağacının genel çalışma süresi açısından kayda değer bir performans kazancı elde ettiği sonucuna vardılar.

ASSOCIATION
RULE MINING

Ye Gao FP büyümesinin kusurlarıyla çelişen bir algoritma önerdi: aynı yolda tekrar tekrar seyahat etmek ve zamanında işlenen düğümleri serbest bırakmamak. Bu arada, FP ağacını silme stratejisini benimser ve belleği azaltmak için işlenen düğümleri serbest bırakır.

ASSOCIATION RULE MINING

3. BİRLİKTELİK KURALI MADENCİLİĞİ KAVRAMLARI

Birliktelik kuralının matematiksel modeli 1993 yılında Agrawal, Imielinski ve Swami tarafından ifade edilmiştir. Bu modele göre; I= { i_1,i_2,…..,i_m}nesnelerin kümesi ve D işlemler kümesi olarak ifade edilir. Her i, bir nesne (ürün) olarak adlandırılır. D veri tabanında her hareket (transaction) T, T ⊆ I olacak şekilde tanımlanan nesnelerin kümesi (nesne küme) olsun. Her hareket bir tanımlayıcı alan olan TID ile temsil edilir. A ve B nesnelerin kümeleri olsun. Bir T işlemler kümesi ancak ve ancak A ⊆ T ise yani A, T'nin alt kümesi ise A'yı kapsıyor denir. Bir birliktelik kuralı A ⇒ B formunda ifade edilir. A önce ve B sonuç olarak adlandırılır. Burada, A ⊂ I , B ⊂ I ve A ∩ B = ∅ dir. İlk olarak, A ⇒ B kuralı için d olasılığı ile kuralın destek değeri tanımlanır. Destek, T işleminin A∪ B 'yi içerme olasılığıdır. İkinci olarak, A ⇒ B kuralının g ile gösterilen güven değeri tanımlanır. Bu olasılık, T işleminin A'yı ve aynı zamanda B'yi içermesidir.

ASSOCIATION RULE MINING

Matematiksel ifade ile kuralın destek ve güven değerleri;

Destek(A ⇒ B) = P(A ∪ B)

Güven(A ⇒ B) = P(B/ A) veya

Güven(A ⇒ B) = Destek(A ⇒ B) /Destek(A)

şeklinde ifade edilir. Burada Destek (A) = Destek (A ⇒ A)' dır.

Başka bir ifade ile destek ve güven değerleri;

Destek(A) = |A|/| D|

Destek(A ⇒ B) =| A.B|/| D|

Güven(A ⇒ B) = Destek(A ⇒ B)/ Destek(A)

Olarak tanımlanır.

ASSOCIATION RULE MINING

Burada;|A|; incelenen kayıtlardaki A ürününü içeren işlemlerin sayısını, |A.B| ; incelenen kayıtlardaki A ve B ürünlerini birlikte içeren işlemlerin sayısını ve D ; veri tabanındaki bütün işlemlerin sayısını ifade etmektedir.

Kuralın destek ve güven değerleri, kuralın ilginçliğini ifade eden iki ölçüdür. Bu değerler sırasıyla keşfedilen kuralların yararlılığını (kullanışlılığını) ve kesinliğini (doğruluğunu) ifade eder.

ASSOCIATION
RULE MINING

4. ÖNERİLEN METODOLOJİ

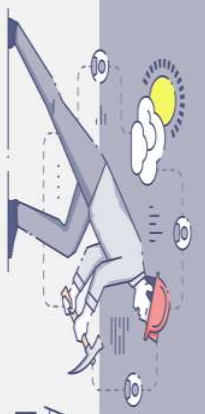
Makalenin ana fikri, sık öğe seti oluşturmanın karmaşıklığını azaltan birliktelik kural madenciliği için bir teknik tasarlamaktır. Birliktelik kuralı madenciliğini azaltmak için işlem veri tabanının taranma sayısını azaltın. FP ağaçlarının oluşturulmasını kaldırın.

ASSOCIATION RULE MINING

Önerilen teknikte yer alan üç adım vardır. İlk adımda, veri tabanından bir D ağacı oluşturmak için veri tabanını bir kez tararız. D-ağacı temel olarak söz konusu veri tabanının kopyasıdır. Veri tabanını taradıktan ve D ağacını çıkardıktan sonra, daha sonraki işlemler için D ağacını kullanırız. Daha sonra saymak için D ağacını tararız. Veri tabanındaki her öğenin oluşumlarını ve her öğenin sıklığını kaydedin. Şimdi sıklığı hesaplamak için veri tabanını değil ağacı tarıyoruz, çünkü bellekte yerleşik ağaç yapısından bir işlemi okumak onu diskten taramaktan daha hızlıdır. İkinci adımda, D-ağacı ve destek sayısını girdi olarak kullanarak, algoritmayı tartışırken ayrıntıları verilecek olan bazı düğümleri ve sıklığını içeren Yeni bir Geliştirilmiş FP ağacı ve bir düğüm tablosu oluşturuyoruz

ASSOCIATION RULE MINING

Yeni geliştirilmiş FP ağacının inşası için, sonraki bölümlerde bahsedilen Algoritma 1'i kullanıyoruz. Üçüncü adımda, destekle birlikte Yeni İyileştirilmiş FP ağacını, destek, düğüm tablosundan bir sayım ve FP ağacındaki her bir öğenin sıklığını girdi olarak alıyoruz ve sık öğe kümelerini elde etmek için Algoritma 2'yi uyguluyoruz. Şimdi her adımı ayrıntılı olarak görelim. Algoritmayı ayrıntılı olarak anlamak için bir örnek ele alalım. Tablo-1, örnek işlem veri tabanını göstermektedir.



ASSOCIATION RULE MINING

4.1. D AĞACININ İNŞASI

D ağacının inşası için Tablo-1’i inceleyin. D ağacını oluşturmak için, veri tabanındaki tüm işlemleri tek tek ele alacağız. Öncelikle ağaç için null olarak bir kök düğüm oluşturacağız. Daha sonra veri tabanındaki her işlem için kökten D ağacında bir yol oluşturuyoruz. Ayrıca her düğüm, bu düğümün kaç kez paylaşıldığını belirleyecek bir sayıma sahip olacaktır. İşlemlerin bazı kısımları verilen iki işlemde ortaksa, ağaçta bazı düğümleri paylaşabilirler. Ancak paylaşılan düğümlerin sayısı her zaman 1'den büyük olacaktır

ASSOCIATION RULE MINING

Şimdi bu fikirle D-ağacı yapmaya devam ediyoruz. İlk a--b--c işlemini ele alalım. Bu işlem için kök düğümden a:1--b:1--c:1 şeklinde bir yol oluşturulacaktır. Daha sonra benzer şekilde ikinci işlem için kökten yeni bir yol b:1--e:1 olarak oluşturulacaktır. Şimdi, üçüncü b-f işlemi için, b düğümünün ikinci işlem için ilk düğüm olarak zaten var olduğunu görebiliriz, bu nedenle bu iki işlem arasındaki ortak düğümdür, böylece paylaşılabilir. Bu nedenle, üçüncü işlem için b düğümünün sayısı 1 artırılacak ve buna yeni bir alt düğüm e:1 eklenecektir. Benzer şekilde veri tabanında kalan işlemlerle ağacı oluşturacağız. Tamamen inşa edilmiş D-ağacı, Şekil 1'de gösterilmiştir. Bu, D-ağacının inşası olan tekniğin ilk adımını tamamlar.

ASSOCIATION RULE MINING

D-ağacının yapımından sonra, ilk adımda veri tabanındaki her bir öğenin sıklığını da hesaplamamız gerekir. Bunu başarmak için, her öğenin ayrı ayrı sayısını elde etmek için D ağacını bir kez taramamız gerekir. Bu amaçla herhangi bir ağaç geçiş algoritmasını kullanabiliriz ve taramanın sonunda her bir öğenin toplam sayısını elde etmiş olacağımız her bir öğenin sıklığını takip edebiliriz. Bu işlemden sonra, ele alınan örnek işlem veri tabanının çıktısı tablo-2'deki gibidir.

Table 1. Sample transactional database

T_id	Transaction
1	a b c
2	b e
3	b f
4	a b e
5	a f
6	b f
7	a f
8	a b f c
9	a b f

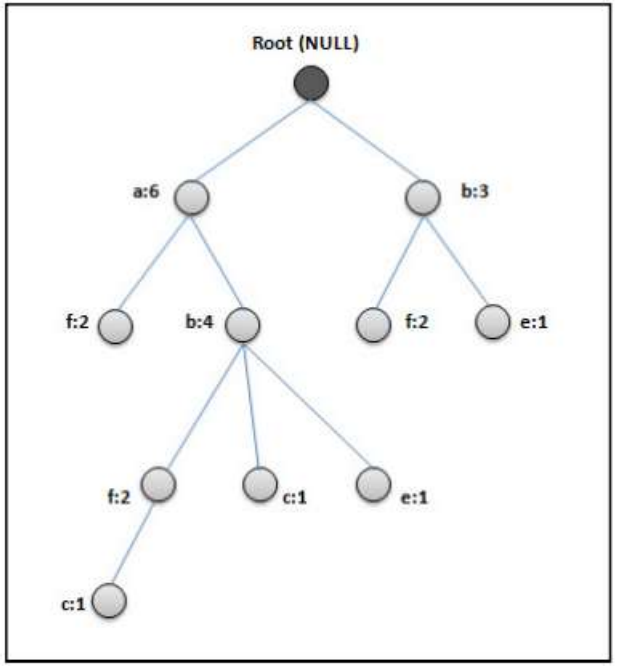


Fig. 1. Construction of D-tree.

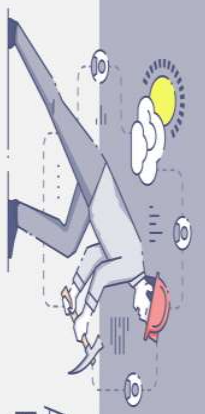
Table 2. Item frequency.

Item	Frequency
b	7
a	6
f	6
c	2
e	2

ASSOCIATION RULE MINING

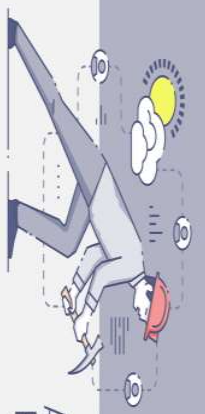
YENİ GELİŞTİRİLMİŞ FP AĞACININ İNŞASI

Önerilen tekniğin ikinci adımında, Algoritma 1'i uyguluyoruz ve Yeni İyileştirilmiş FP ağacını elde ediyoruz. Algoritmanın girdisi D-ağacıdır ve destek sayısıdır ve bunun çıktı kimliği Yeni İyileştirilmiş FP ağacı ve bir düğüm tablosudur. Ağaç, işlem veri tabanındaki öğeler arasındaki korelasyonu daha spesifik olarak temsil etmek için kullanılır. Düğüm tablosu, yedek olan bazı öğeleri depolamak için kullanılır. Düğüm tablosu, öğe_adı ve sıklık olmak üzere iki sütundan oluşur. Düğüm tablosunu tanıtmanın arkasındaki fikir, orijinal FP ağacında bir dizi dalın olması ve aynı öğenin birden fazla düğümde görünmesidir.



ASSOCIATION RULE MINING

Önerdiğimiz teknikte, her benzersiz öğenin ağaçta yalnızca bir düğümü vardır. Bu nedenle, işleme için basit ve verimlidir. Şimdi herhangi bir öğeyi düğüm tablosuna koymak için dikkate alınması gereken belirli durumlar vardır. Herhangi bir koşul altında ağacı yaparken herhangi bir durum gerçekleşirse öğeyi düğüm tablosuna koyarız. Düğüm tablosuna herhangi bir öğe koyduğumuz iki durum vardır. Durum 1: İncelenmekte olan öğenin, dikkate alınan geçerli kökten düğümüne kadar bir kenarı olmadığında. Bu, öğenin FP ağacında zaten mevcut olduğu anlamına gelir. Kök, ağacın oluşumu boyunca değişmeye devam eder, sabit değildir. Durum 2: Bir işlem en sık kullanılan öğeyi içermiyorsa, tüm öğeleri düğüm tablosuna koyarız. Şimdi Algoritma-1'i görelim. Algoritmanın detaylı çalışması şu şekildedir:



ASSOCIATION RULE MINING

Algoritma 1: Yeni Geliştirilmiş FP-ağacının İnşası

Girdi: D-Ağacı, destek sayısı

Çıktı: Yeni Geliştirilmiş FP ağacı, Yeni Geliştirilmiş FP ağacındaki her öğenin sıklığı ve bir sayım

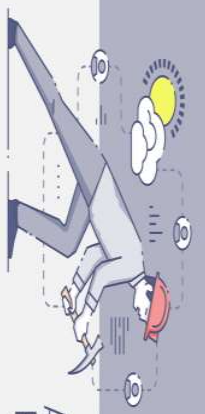
Adım 1: Ağaç için R kök düğümünü oluşturun. Bir önek ağacı olduğundan, R= NULL.

Adım 2: D-ağacından düşünülen her yol için şunları yapın

Adım 3: İşlemdeki oluşma sayısına göre, dikkate alınan yoldaki öğeleri sıklığın azalan sırasına göre sıralayın.

Adım 4: Azalan sıralı yolun [k|L] ile temsil edilmesine izin verin, burada k ilk öğedir ve L yoldaki diğer öğeleri temsil eder.

Adım 5: k= veri tabanındaki en sık kullanılan öğe ise, 6. ve 7. adımları uygulayın, aksi takdirde 8. adıma geçin



ASSOCIATION RULE MINING

Adım 6: Kök R'nin doğrudan bir alt düğümü N varsa, N'nin öğenin_ismi = k'nin öğenin_ismi, sayımı artırın N öğesinin 1 ile 1. Kökü R'den k'ye aktarın

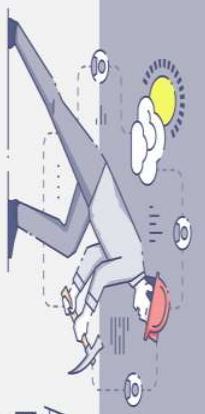
Adım 7: L'de kalan her bir öğe için aşağıdaki adımları uygulayın

[a] Mevcut kökten her yeni öğe için 1 olarak sayılan yeni bir alt düğüm oluşturun, kökü yeni düğüme aktarın.

[b] Bir öğe ise, Li'nin geçerli kökten düğümüne kadar tek bir kenarı yoktur, burada Li, L'de bir öğedir ve yeni FP-ağacında zaten mevcuttur. O zaman Li, veri tabanına konulacak bir öğedir. Li'yi tablodaki sıklık 1 ile saklayın.

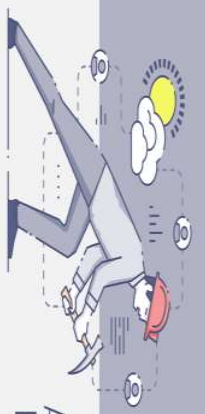
Adım 8: Dikkate alınan yoldaki her öğe için [k | L]. Tablodaki tüm öğeleri sıklık 1 ile saklayın.

Adım 9: Tabloda bir düğüm için birden fazla giriş varsa tablodaki frekansları toplayın



ASSOCIATION RULE MINING

Çıkarlarımızın ilişkilerinin çoğu en sık görülen maddeyle ilgilidir. Bu nedenle, önerilen teknikte azami dikkat, veri tabanındaki en sık kullanılan öğeye odaklanmıştır. Bu nedenle, D ağacından gelen her yolu düşündüğümüzde, tüm öğeleri frekansın azalan sırasına göre düzenleriz, böylece en sık kullanılan öğe her zaman işlemdeki ilk konumda bulunur ve en üst düzey düğümlerde kalır. Yeni Geliştirilmiş FP ağacı. Ağacın kökü algoritmanın 6. adımında değiştirilir, ayrıca bir öğe tekrarlanırsa 6. adımda frekansı 1 artırılır. Ağacın kökü, veri tabanındaki öğeler arasında korelasyonun açıkça görülebilmesi için değiştirilir. Öğeleri düğüm tablosuna koymak için kontrol, algoritmanın 5. adımında gerçekleştirilir. Durum 1 olmuştur. Adım 7'ye [b] koyun ve Durum 2, algoritmanın 8. adımına yerleştirildi. Burada FP ağacını temsil ediyoruz.



ASSOCIATION RULE MINING

4.3 SIK KULLANILAN ÖGE SETLERİ MADENCİLİĞİ

Geleneksel FP büyüme algoritmasının, aday oluşturmayı önlemek için çok sayıda koşullu örüntü tabanı oluşturması ve ardından koşullu FP ağacını hesaplaması gerekir. Ancak daha büyük veri tabanları söz konusu olduğunda verimli değildir ve ana belleği aşırı yığabilecek büyük bir bellek gereksinimine ihtiyaç duyar. Sık öğe kümelerinin oluşturulması için algoritma 2'yi kullanacağız. Bu algoritmanın girdisi, yeni geliştirilmiş FP ağacı ve kullanıcıdan gelen destek sayısı ve sık kullanılan öğe setlerinin çıktısıdır. Algoritma 2'nin uygulanmasından sonra, birliktelik kurallarının geçerli seti, Bölüm III'te verilen kavramlar kullanılarak bulunabilir.

ALGORİTMA 2: SIK ÖGE KÜMESİ MADENCİLİĞİ İÇİN

Girdi: Yeni Geliştirilmiş FP ağacı, destekler, sayım-C, frekans(sıklık)-F

Çıktı: Sık Öğe Seti

Yeni Geliştirilmiş FP ağacındaki her q öğesi için

do {

if (q.F<q.S)

Yeni Geliştirilmiş FP ağacındaki en sık öğe düğümüne kadar, dikkate alınan tüm olası öğe kombinasyonları ve tüm ara düğümler olarak sık öğe kümesini oluşturun, sıklık q.F + C olacaktır.

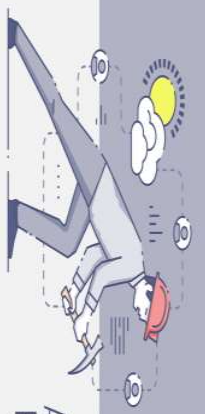
else if (q.F = = q.S)

Sık öğe kümeleri oluşturun dikkate alınan öğenin tüm olası kombinasyonları ve Yeni Geliştirilmiş FP ağacında daha yüksek frekansa sahip düğümler olarak, frekans q.F olacaktır.

else

Yeni Geliştirilmiş FP ağacındaki tüm olası öğe kombinasyonları ve ana düğümü olarak sık öğe kümesi oluşturun, sıklık q.F olacaktır

}

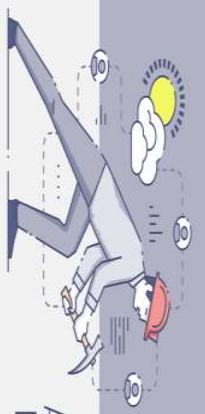


ASSOCIATION RULE MINING

Algoritma 2'nin uygulanmasından sonra sık öğe kümelerini frekanslarıyla birlikte elde ederiz, yani IF (Sık öğe kümesi frekansı). Sık kullanılan öğe kümelerinden ilişkilendirme kurallarını elde etmek için sık kullanılan öğe set frekansını (IF) kullanacağız. Sık öğe seti madenciliği algoritması, FP ağacındaki düğümlerin kombinasyonlarını elde etmek için FP ağacındaki aynı temel kavramı kullanır. Sık kullanılan öğe kümelerini bulmada üç durumu göz önünde bulundurur. Yeni geliştirilmiş FP ağacı F'deki frekans ile kullanıcı tarafından verilen destek S arasında karşılaştırma yapar. Üç olası durum eşittir, küçüktür ve büyüktür. Farklı durumlara dayanarak, sık öğe kümelerinin frekanslarına ve ayrıca kombinasyon elde etmek için dikkate alınacak düğümlere karar verir

ASSOCIATION RULE MINING

Her vakayı ayrıntılı olarak görelim. Yeni geliştirilmiş FP ağacındaki öğenin sıklığını q.F olarak ve desteği q.S ile temsil edeceğiz. Düğüm tablosundaki sayımı dikkate alacağız, çünkü bir öğenin ağaçta daha az sıklığa sahip olması, ancak gerçekte veri tabanında daha yüksek sıklık değerine sahip olması söz konusu olabilir. Sık kullanılan öğe seti, Yeni Geliştirilmiş FP-ağacındaki en sık öğe düğümüne kadar tüm olası öğe kombinasyonları ve tüm ara düğümler olarak üretilecektir, çünkü ağaçtaki ara düğümler öğeler arasındaki korelasyonu temsil eder. İkinci durum için q.F q.S'ye eşit olduğunda IF frekansı q.F olacaktır. Çünkü bu durumda minimum destek kısıtlamasını karşılamaktadır.



ASSOCIATION RULE MINING

Sık kullanılan öğe seti, Yeni Geliştirilmiş FP-ağacında dikkate alınan öğenin ve daha yüksek frekansa sahip düğümlerin tüm olası kombinasyonları olarak üretilecektir. Üçüncü durumda, qF, qS'den büyük olduğunda, IF, q.F olacaktır. Yeni Geliştirilmiş FP-ağacındaki tüm olası öğe ve üst düğüm kombinasyonları olarak sık öğe seti oluşturulacaktır, çünkü üst düğüm her zaman daha yüksek frekanslara sahip olacaktır.

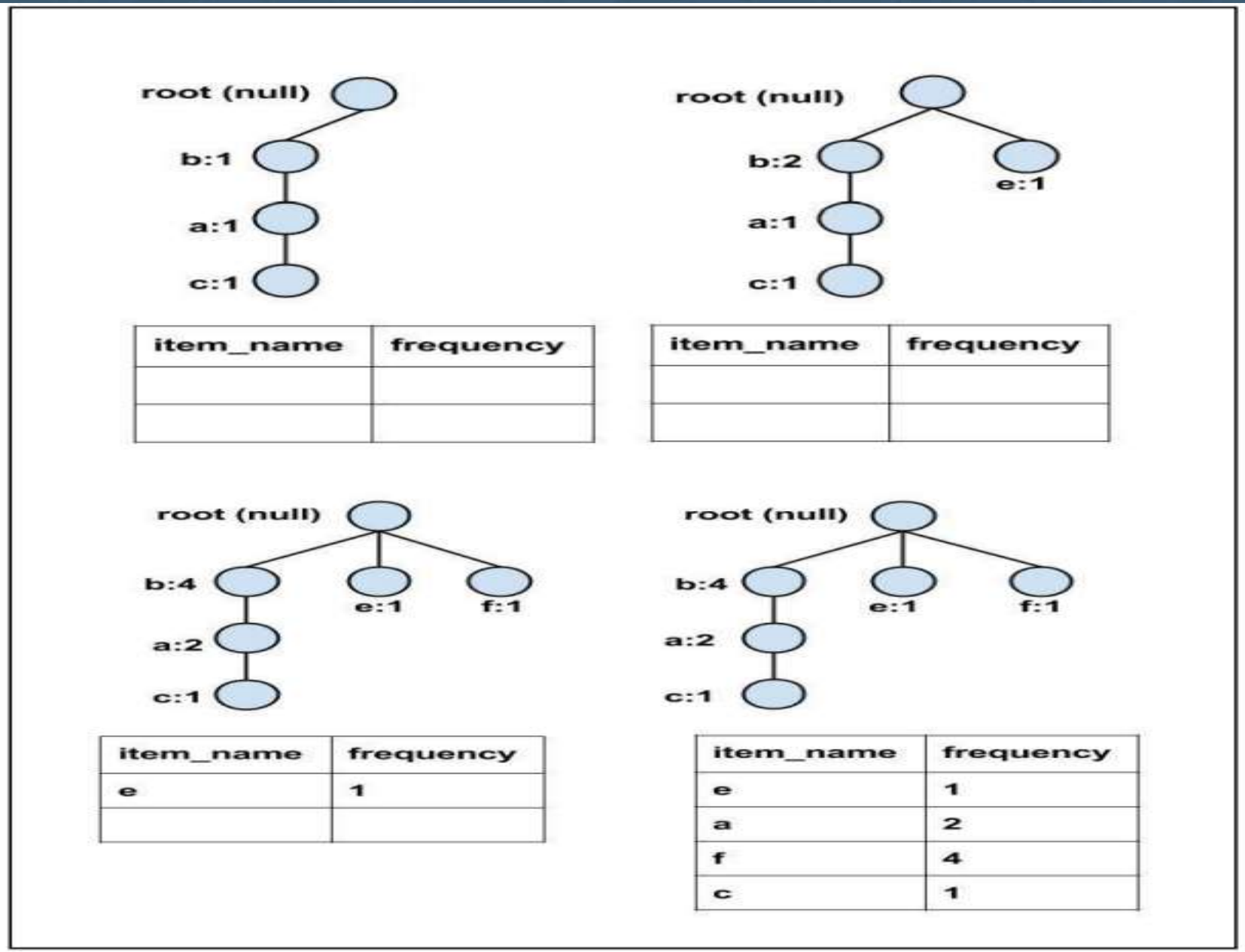
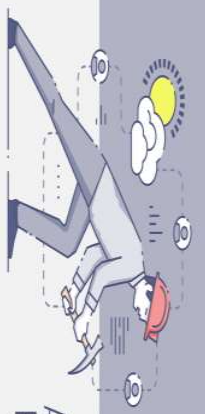


Fig. 2. Construction of the new improved FP tree.



ASSOCIATION
RULE MINING

Tablo-1 veri tabanını daha fazla ele alalım ve sık öğe kümelerini elde etmek için algoritma-1 ve algoritma-2'yi aynı şekilde uygulayalım. Şekil 2, algoritma-1'in çıktısını, yani oluşturulan ağacı ve oluşturulan düğüm tablosunu göstermektedir. Tablo-3, algoritma-2'nin çıktısını, yani her bir düğüm için oluşturulan sık öğe setlerini göstermektedir. Algoritma-2'nin uygulanmasından sonra, birliktelik kurallarını elde etmek için bölüm iii'de verilen kavramları uygulayabiliriz. Sık öğe seti, sıklığı ve kullanıcının destek ve güven girdisi ile birlikte nihayet ilişkilendirme kurallarını verecektir

Table 3. Frequent Item set.

Item	Frequent item set
b	{b:7}
a	{a:4};{a,b:4}
c	{c:2};{c,a:2};{c,b:2};{c,a,b:2}
f	{f:2};{f,a:2};{f,b:2};{f,b,a:2}
e	{e:2};{e,b:2}

ASSOCIATION RULE MINING

5. Sonuçlar

Deney için tablo-1'de verilen verileri dikkate alıyoruz. Karşılaştırma için dikkate aldığımız parametreler, veri tabanı taraması, aday oluşturma, koşullu örüntü tabanları ve koşullu FP ağaçlarıdır. Aşağıdaki grafikler, Apriori, FP büyüme, Önerilen algoritmanın farklı parametrelerle karşılaştırmasını göstermektedir. x ekseni prosedürü temsil eder ve y ekseni numarayı temsil eder.

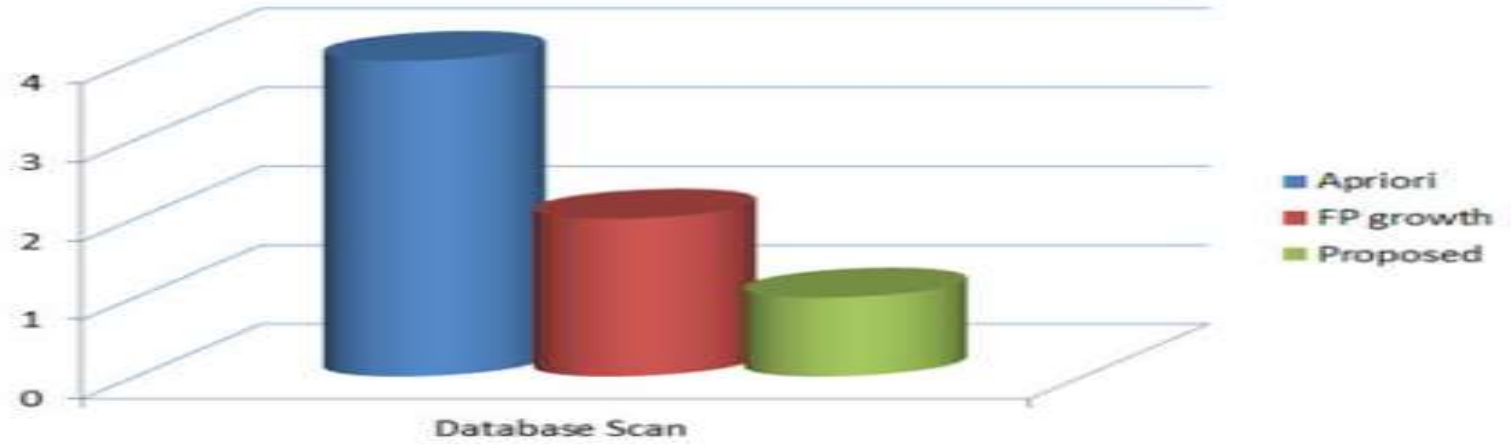


Fig. 3. Comparison graph in terms of database scan.

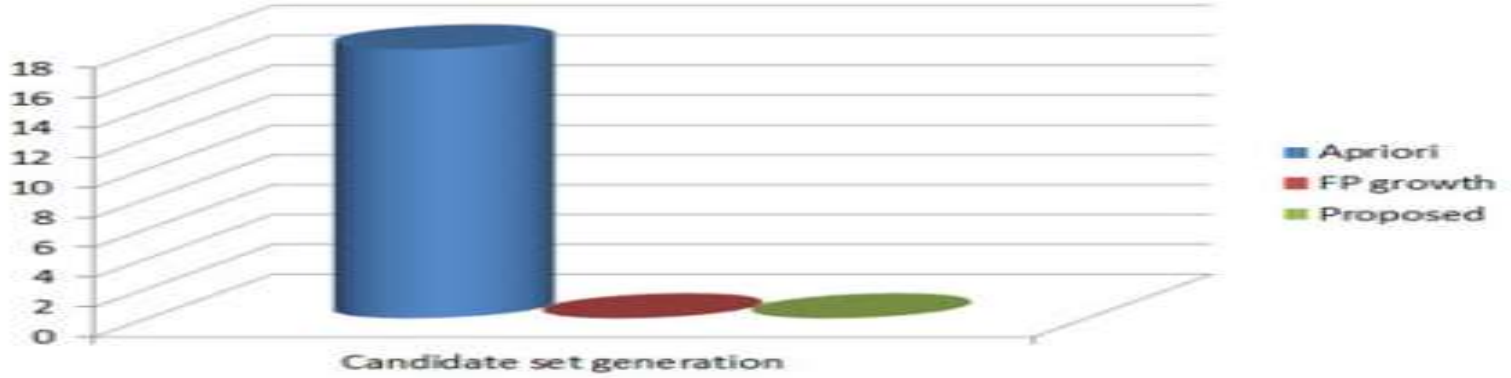
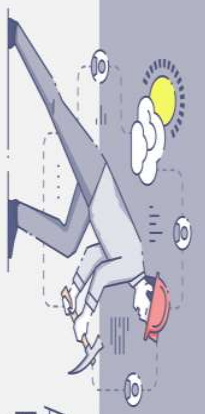


Fig. 4. Comparison graph in terms of Candidate set generation.



ASSOCIATION RULE MINING

Şekil 3,4,5,6 mevcut ve önerilen algoritmaların karşılaştırılmasına yönelik grafikleri göstermektedir. Şekil 3, veri tabanı taraması açısından karşılaştırmayı göstermektedir. Apriori veri tabanını 4 kez, FP büyümesini 2 kez ve önerilen algoritmayı yalnızca 1 kez tarar. Şekil 4, üretilen aday set açısından karşılaştırmayı göstermektedir. Apriori aday kümesini 18 kez üretir, FP büyümesi herhangi bir aday kümesi oluşturmaz ve önerilen algoritma da herhangi bir aday kümesi oluşturmaz. Şekil 5, üretilen koşullu FP ağaçları açısından karşılaştırmayı göstermektedir. Apriori, koşullu FP ağaçlarını üretmez, FP büyümesi bunu 26 kez üretir ve önerdiğimiz algoritma herhangi bir koşullu FP ağacı üretmez. Şekil 6, üretilen koşullu model tabanları açısından karşılaştırmayı göstermektedir. Apriori algoritması herhangi bir koşullu örüntü tabanı üretmez, FP büyümesi 10 kez örüntü tabanını üretir ve önerilen algoritma 5 kez koşullu örüntü tabanını üretir.

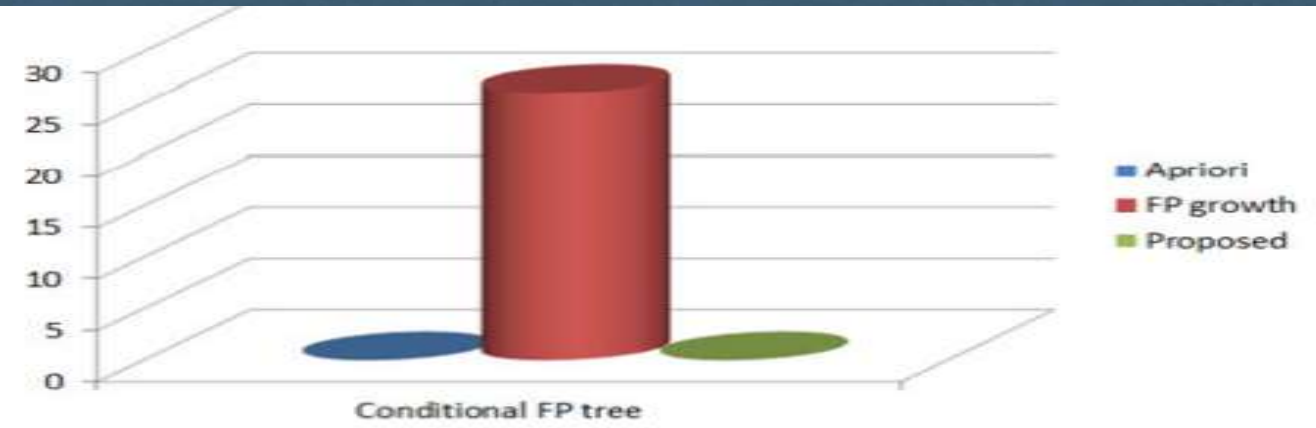


Fig. 5. Comparison graph in terms of Conditional FP tree.

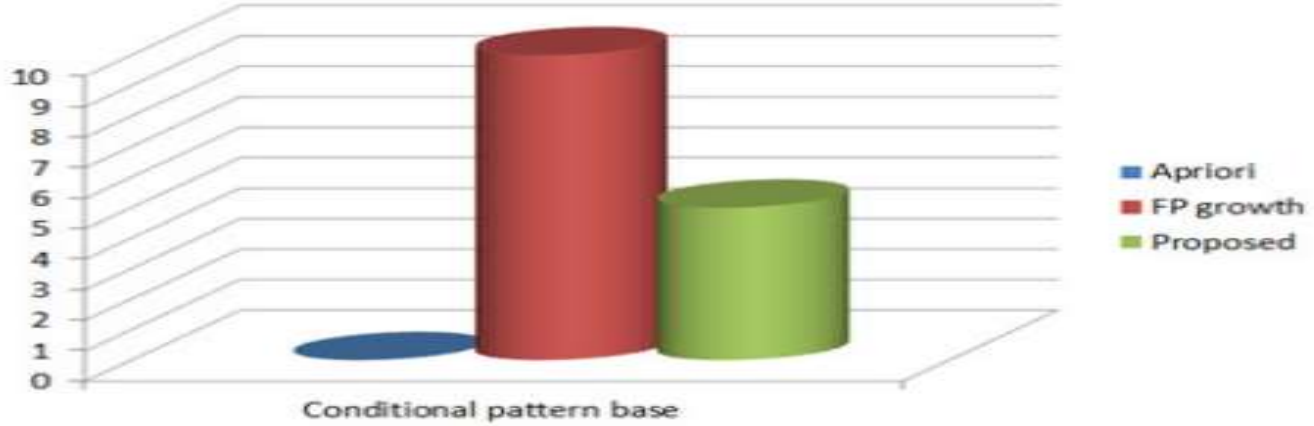
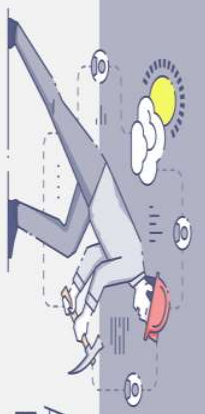
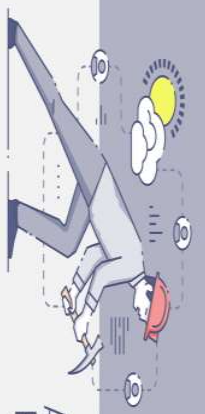


Fig. 6. Comparison graph in terms of Conditional pattern bases.



ASSOCIATION RULE MINING

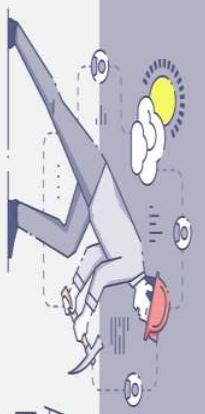
Sonuçlardan, önerilen algoritmanın çok verimli olduğunu kolayca analiz edebiliriz. Veri tabanına yapılan tarama sayısını azaltır, bunun sonucunda zaman kazanılır. Apriori algoritmasına kıyasla hiçbir aday seti oluşturmaz. Geleneksel FP büyüme algoritmasında olduğu gibi çok fazla bellek tasarrufu sağlayan koşullu FP ağaçları oluşturmaz. Koşullu FP ağaçları oluşturulur. Aynı zamanda daha az hayır üretir. Çünkü ilgilenilen kuralların çoğu veri tabanındaki en sık kullanılan öğeyle ilişkilendirildiğinden yalnızca en sık kullanılan öğeyi dikkate alır. Tüm bu noktalar, önerilen algoritmanın etkinliğini kanıtlamaktadır.



ASSOCIATION
RULE MINING

6.SONUÇ:

İlişkilendirme kuralı madenciliği, sınıflandırma, XML madenciliği, mekansal veri analizi ve hisse senedi piyasası ve tavsiye sistemleri dahil olmak üzere kayda değer miktarda pratik uygulamaya sahiptir. İşletmenin kar elde etmesinde fayda sağlayacak ve uygulamalarının diğer birçok alanında da farklı şekillerde yardımcı olacak veri tabanlarından ilginç kurallar bulma görevidir. Bu alanda şimdiye kadar birçok teknik geliştirilmiştir. Koşullu FP ağaçlarının üretimi olan FP büyüme algoritmasının sınırlandırılmasının üstesinden gelen yeni bir teknik geliştirdik. Bu nedenle, tekniğimizi kullanarak, daha önce çok sayıda koşullu FP ağacını depolamak için harcanan çok fazla bellekten tasarruf ediyoruz



ASSOCIATION
RULE MINING

Bu çalışmada, yeni geliştirilmiş bir FP ağacı ve yeni bir Sık Öğe kümesi madenciliği algoritması yardımıyla, no'yu azaltma açısından çok fazla bellek tasarrufu sağlayabiliyoruz. Test veri seti üzerinde deneyler yaptık ve geliştirilen tekniğin mevcut sistemden daha verimli olduğunu gördük. Ayrıca yeni geliştirilmiş FP ağacı, maddeler arasındaki korelasyonu daha net bir şekilde verir.

ASSOCIATION RULE MINING

İleride sistem üzerinde çalışacağımız için bu teknik ile XML verileri üzerinde çalışabiliriz. Web üzerinden veri alışverişi ve yarı yapılandırılmış verileri temsil eden bir standarda ihtiyaç vardır. XML kullanılarak her ikisi de elde edilebilir. Web'de XML verilerinin madenciliği için talep artıyor. XML veri madenciliği için zaman içinde birçok teknik sunulmuş olsa da, daha fazla geliştirmenin önündeki engel, tekniğin basitliği ve verimliliğidir. Gelecekteki çalışma olarak, geliştirilen teknikle XML verileri üzerinde çalışmak ilginç olacaktır.