BİTİRME PROJESİ FİNAL RAPORU

IOT İçin Gerçek Zamanlı İlişkilendirme Kuramı

161180908 Dursun Ece Gökçay

161180913 Mustafa Öcal

161180918 Fatih Yücel

Gazi Üniversitesi
Bilgisayar Mühendisliği Bölümü

Proje Danışmanı
Doç. Dr. Suat Özdemir

8 Haziran 2017

ÖZET

İlişkilendirme/Birliktelik kuralı, büyük veri kümeleri arasındaki ilişkileri bulan, olayların birlikte gerçekleşme ihtimallerini geçmiş verileri analiz edip ortaya koyarak geleceğe yönelik çalışmaları destekleyen, veri madenciliğinin vazgeçilmez yöntemidir.

İlişkilendirme/Birliktelik Kuralı; ekonomi, eğitim, e-ticaret, pazarlama, telekomünikasyon gibi birçok sektörde geniş kullanıma sahiptir. Haber sitelerinde dolaştıktan sonra bize tavsiye edilen haberler ilişkilendirme/birliktelik kuralı sonucudur. Ya da ücretsiz telefon uygulamalarında çıkan reklamlar, aslında birilerinin geçmiş alışverişlerimizi analiz ederek ilişkilendirme/birliktelik kuralı ile ortaya çıkmış reklamlardır.

İlişkilendirme/Birliktelik kuralı hayatımızın her alanında karşımıza çıkarken bilgisayar bilimi de bu kurala yönelik geliştirmeler uğraşına girmiştir. İlk olarak karşımıza çıkan Apriori ve FP-Growth algoritmaları her ne kadar etkili olsalar da veri miktarı arttıkça hız ve eliminasyon gibi etkileri azalmıştır. Bu yüzden zamanla bilgisayar bilimcileri ve veri bilimcileri bu algoritmalara çeşitli eklemeler yaparak geliştirmişlerdir.

Biz de projemizde geliştirilen bu ilişkilendirme kuralı algoritmalarını inceleyeceğiz. İncelediğimiz ilişkilendirme kuralı algoritmalarını veri setleri üzerinde deneyerek analiz edeceğiz. Analizimizin sonucunda IOT verileri için en optimal ilişkilendirme kuralı algoritmasını bulacağız.

Anahtar kelimeler: İlişkilendirme kuralı, birliktelik kuralı, IOT, FP-Growth, Apriori

Sayfa Adedi: 29

Danışman: Doç. Dr. Suat Özdemir

İÇİNDEKİLER

1.GİRİŞ

1.1.Problemin Tanımı

İlişkilendirme/Birliktelik kuralı, birbiriyle ilişkili olan değişkenlerin ortaya çıkarılması ve aralarındaki bağlantının büyüklüğünün tespit edilmesine yöneliktir. Birliktelik kuralı belirli türlerdeki veri yapıları arasındaki ilişkileri tanımlamaya çalışan bir yöntemdir.

İlişkilendirme/Birliktelik kuralı gerçek hayatta fazlasıyla kullanılmaktadır. Örneğin, Google karşımıza reklam sunarken her zaman bize yönelik reklamları seçer. Bunun nedeni, Google geçmiş verilerimizi irdelemekte ve bize en uygun en ilgi çekici reklamı göstermektedir. Bunu ise ilişkilendirme/birliktelik kuralı ile yapmaktadır.

Bu durum öyle bir hal aldı ki, artık firmalar reklam verirken veri bilimcilerine başvurmaktadır. Google, Yandex, Microsoft gibi üst firmalar rekabeti ilişkilendirme/birliktelik kuralı üzerine taşımaktadır. Bu yüzden artık klasik ilişkilendirme/birliktelik kuralı üzerine olan Apriori, FP-Growth gibi algoritmalar artık yetersiz kalmaktadır.

Bu algoritmaları geliştirmek için çeşitli korelasyon yöntemleri geliştirilmektedir. Bu korelasyon türleri çok yoğun veri setlerinde güçlü ilişkiler kurmakta ve aynı zamanda bunu kısa sürede sağlamaktadır. Ancak her geçen gün verinin artması bazı korelasyon tekniklerinin de yetersiz kaldığını göstermektedir. Bu sebeple korelasyon türleri de geliştirilmeye çalışılmıştır. Örnek olarak lineer korelasyonun bazı veri setlerinde yetersiz kalmasından dolayı, bilgi teorisindeki entropi özelliği kullanılarak lineer olmayan veri setlerinde ilişkilendirme kuralı hesaplanmıştır.

Her geçen gün verinin artması ilişkilendirme/birliktelik kuralına yönelik geliştirilen algoritmaların yetersiz kalmasına neden olmuştur. Bu algoritmaların hız, ilişkilendirme gibi etmenlere bağlı olarak gelişmeye ihtiyacı olmuştur.

Projenin amacı; geliştirilen ilişkilendirme kuralı algoritmalarını inceleyerek, bu tür algoritmaların nasıl geliştiğini incelemek, aralarındaki farkları ortaya koymak, daha sonra ilişkilendirme kuralı algoritmalarını çeşitli veri setlerinde deneyerek, bu algoritmaların çalışma hızları ve doğruluk oranlarını göre analiz yapmak ve sonuç olarak da IOT verileri için en optimal algoritmayı bulmaktır.

1.2.Literatür

Literatür taramasını yaparken, incelediğimiz makalelerin kıyaslama yöntemini kullanarak özetlerini çıkardık. Bu makalelerden yararlanılabilecek teknikleri, makalelerin eksik veya geliştirmeye açık olan yönlerini incelemeleyerek bu durumları bu kısımda belirttik.

- **Mining Frequent Patterns without Candidate Generation[1]**

  Bu makalede, yaygın öğelerin önemli bilgilerini sıkıştırılmış bir biçimde tutmak için yeni bir model ağaç (FP-Tree) veri yapısı ve bu veri yapısını taban olarak alan Apriori'den farklı bir madencilik metodu (FP-Growth) geliştirilmiştir.

  Apriori algoritması yapısı gereği aday kümesi büyük olduğunda maliyeti yüksektir. Ayrıca database tekrar tekrar taramak da maliyeti artırıcı etkenlerdendir. Apriori algoritmasının darboğaz kısmı (maliyeti arttıran kısmı) aday küme oluşturma ve bu kümenin test edilmesidir. Fp-Tree 'de bu problemi ortadan kaldıran 3 özellik bulunmaktadır.

  1. Frequent pattern hakkındaki önemli ve sayısal bilgileri tutan bir ağaç yapısıdır. Ağaç yapısı, daha yaygın öğelerin daha az yaygın öğelere oranla paylaşılma ihtimalinin daha fazla olacağı şekilde düzenlenmiştir.
  2. Fp-Tree parça genişletme(fragment growth)tekniğine dayanır. Burada sadece koşullu öğe yapısı incelenir ve Fptree bu yapı dikkate alınarak oluşturulur. Parça genişletme, 1 uzunluğundaki parçalarla başlayarak yeni oluşturulan ekler ile birleştirilerek yeni bir parça üretme tekniğidir.
  3. Fp-Tree'de kullanılan divide-and-conquer metodu ile koşullu öğe boyutunu azalttığı gibi koşullu Fp- Tree boyutunu da azaltarak katkıda bulunur.

  Yapısı: Fptree; null isimli root node, item-prefix subtree'ler ve frekans başlık tablosundan oluşur. Bu item- prefix subtree'lerde öğe isimleri, adetleri ve bir sonraki node'u tutan node link bulunur. Frekans başlık tablosunda ise öğe isimleri ve Fp-Tree de bu öğe ismini taşıyan ilk node'a işaret eden node link bulunur.

İnşa Edilmesi:

1. Veritabanı bir kere taranıp minimum destek değeri dikkate alınarak yaygın öğe listesi sıralı bir biçimde oluşturulur.
2. Ağacın kök olan null node'u oluşturulur. Veritabanı ikinci kez tarandığında her bir işlemde yaygın öğe listesinde bulunan öğeler seçilerek sıralanır. İlk işlemde bu öğeler ağaca direk eklenir. Diğer işlemlerde ise sıralanmış öğelerde prefix aranır. Eğer sıralanmış öğelerde bulunan prefix ağaçta da var ise bu prefix içindeki öğelerin sayıları 1 arttırılır. Olmayan öğeler ise ağaca eklenir.

FP-Tree'nin boyutu taranan veritabanından büyük olamaz. FP-Tree derinliği ise işlemlerde bulunan maximum yaygın öğe sayısına eşittir. İşlemlerde ne kadar çok öğe paylaşılırsa, FP-Tree boyutu da o kadar küçük olur.

FP-Growth algoritmasında, öncelikle içerisinde algoritmaya input olarak verilen öğenin geçtiği yollar belirlenir. Eğer tek bir dal varsa yaygın öğeler kümesi, dalı oluşturan öğelerin kombinasyonudur. Eğer birden fazla yol varsa, destek değer o yoldaki minimum destek değeri olarak belirlenir. Daha sonra bu yollar o öğe için koşullu örüntü temelini oluşturur. Her bir koşullu örüntü temelinden koşullu örüntü ağacı oluşturulur. Daha sonra bu şartlı örüntü ağacı üzerinde algoritma recursive olarak yeniden çalıştırılır.

Apriori, TreeProjection ve FP-Tree algoritmaları C++ dilinde yazılarak performans karşılaştırması yapılmıştır. Destek eşiği düşük olduğunda FPGrowth ve TreeProjection algoritmalarının performansları Apriori'den daha yüksek olduğu görülmüştür. İkisi arasında, veritabanı büyüdükçe ve destek eşik değeri küçüldükçe FPGrowth algoritması daha yüksek performans göstermiştir.

- **Top Down FP-Growth for Association Mining[2]**

Yazarlar bu makalede frequent patternleri FP-tree kullanarak bulmak için yeni ve deneylerine göre daha efektif bir algoritma sunuyorlar. Bu yeni algoritmaya Top Down FP-growth'un baş harflerinden esinlenerek TD-FP-Growth adını vermişlerdir. Daha önce sunulan metodojilerde FP-tree aşağıdan yukarıya doğru aranıyordu, fakat bu çalışmada tam tersi olarak tepeden aşağı tarama gerçekleştirilerek şartlı pattern tabanlarının ve sub-FP-tree'lerin oluşması engelleniyor ve bu sayede zamandan ve alandan tasarruf edilmektedir. Yazarlar çalışmada aynı zamanda bu algoritmayı ilişki kuramları bulmak

adına da genişletiyorlar.

TD-FP-Growth, FP-tree'yi kurmak için ilk olarak veritabanını iki defa tarar. İlk taramada, her itemin tekrar sayısı elde edilir, ikinci aramada ise kayıtlarda bulunan frequent itemlere göre nodelar ağaca eklenir öyle ki iki kayıtın en çok tekrarlanan itemleri aynı ise aynı ata node'un soyundan gelmektedirler.

TD-FP-Growth'un daha öncede söylediğimiz gibi ayırıcı özelliği ağacı yukarıdan aşağıya taramasıdır. Yani üst seviyedeki nodeları işledikten sonra alt seviyedeki nodelar işlenir. Bu yapılırken akla takılan sorulardan birisi, ilk işlenen üst katmanlarda gerçekleşen bir değişikliğin alt katmanları etkileyip etkilemeyeceğidir ki yazarlar bunun olmasını engelleyecek önlemler de almışlardır. Geri dönecek olursak, algoritmanın işleyişinden çıkarım yapılabileceği üzere herhangi bir katmandaki bir node'u işleme alacağımız sırada bu node'un tüm atalarının daha önce işlendiği bilgisini tutulmaktadır. Bu yüzden, bir patternin şartlı pattern base'ini elde etmek için sadece o node'dan atalarına doğru olan yolu takip etmek ve yol üzerindeki sayıları yenilemek yeterli olacaktır. Böylece, farklı bir yol oluşturmadan bu yolların güncellenmesi yapının kendi içerisinde gerçekleşmiş olacaktır ki bu da hem zaman hem alan açısından verimi kesinlikle artıracaktır. Çünkü bildiğimiz gibi bottom-up FP-growth'larda işlem süreci boyunca sürekli şartlı pattern base'lerin ve alt- ağaçların kurulması gerekmektedir.

Deneylerde TD-FP-Growth; Apriori ve FPG-tree'lerle karşılaştırılmıştır. Tüm veri setlerinde TD-FP-Growth en etkili algoritma olmuştur.

- **Feature Selection for High-Dimensional Data: A Fast Correlation-Based Filter Solution**[3]

Feature Selection algoritmaları iki kategoriye ayrılır: Filter Model ve Wrapper Model. Filter modelde, makine öğrenme algoritması kullanmaksızın feature'lar seçilir. Wrapper modelde ise predetermined öğrenme algoritmaları kullanılır. Yoğunluk arttıkça wrapper modeli kullanmanın maliyeti artar. Bu çalışma, çok yoğun datalara yönelik olduğu için filter model üzerinden ilerlemişletilmiştir.

Filter modeldeki algoritmaları 2 grubu ayırabiliriz: Feature Weighting algoritmaları ve subset search algoritmaları. Feature Weighting algoritmaları her ne kadar hızlı olsa da redundant featureleri temizlemede yetersiz kalmaktadır. Subset search algoritmaları ise yüksek yoğunluktaki verilerde çok yavaş kalmaktadır. Bu çalışmada ise amaçlanan hem

redundant featureları optimal bir şekilde temizlemek hem de zaman karmaşıklığı bakımından subset search algoritmalarından daha iyi olan bir algoritma sunmak.

2 değişken arasındaki korelasyonu bulmak için 'lineer korelasyon' ve 'bilgi teorisi temelli yaklaşım' olmak üzere başlıca 2 yaklaşım vardır. Lineer korelasyonda 2 değişken arasındaki ilişki lineer şekilde bulunarak redundant feature'lar temizlenir. Her ne kadar bu yaklaşım iyi gibi görünse de her zaman ilişkiler lineer değildir. Bu yüzden bilgi teorisi temelli yaklaşımı çalışmalarında kullanmışlardır. Bu yaklaşımda entropi kullanılarak bilgi kazanımı elde edilmekte ve bu bilgi kazanımı ile redundant featurelar belirlenmektedir. Bu bilgi kazanımlarının normalize edilmiş hali olan symmetrical uncertanity'yi kullanmışlardır. Böylelikle değerler 0 ile 1 arasında olucak. Örnek olarak, SU(x,y) = 0 olursa, X ve Y birbirinden bağımsız demektir.

Çalışmalarında 2 soruya cevap aramışlardır: 1) Feature'ların sınıfla ilişkili (relevant) olup olmadıklarına nasıl karar verilecek? 2) İlişkili (relevant) feature'ların, diğer ilişkili featurelar göz önünde bulundurularak redundant olup olmadıklarına nasıl karar verilecek? İlk sorunun çözümünü threshold SU değeri kullanılarak, bir çok feature weighting algoritması ile çözülebilir. Ancak 2. Sorunun çözümünde kıyaslama yapılması gerektiğinden dolayı zaman karmaşıklığı artacaktı. Bu yüzden Predominant Korelasyonu'nu tanımlamışlardır.

Bir F_i feature'u ile C sınıfının korelasyonun, Predominant Korelasyon olması için aşağıdaki önermeyi takip etmelidir:

"İff $SU_{i,c} \geq$ threshold_value and $\forall F_j \in S'$ ($j \neq i$), there exist no F_j such that $SU_{i,j} \leq SU_{i,c}$"

Eğer bu önermeye karşı bir F_j değeri varsa, buna redundant peer deriz ve bu redundant peer'leri ifade etmek için S_{Pi} kümesi tanımlanır. Bu küme S_{Pi}^{+}($SU_{j,c} > SU_{i,c}$) ve S_{Pi}^{-}($SU_{j,c} \leq SU_{i,c}$) olmak üzere ikiye ayrılır. Redundant peer'ler temizlendikten sonra korelasyon, predominant olabilir.

Geliştirdikleri algoritma aşağıdaki 3 heuristic'i takip etmekte:

Heuristic 1 - (S_{Pi}^{+} kümesi boş bir küme ise) F_i'yi predominant feature olarak işle, S_{Pi}^{-} kümesindeki featureları sil ve onlar için redundant peer hesaplama

Heuristic 2 - (S_{Pi}^{+} kümesi boş bir küme değil ise) F_i hakkında karar vermeden önce S_{Pi}^{+} deki her feature'u işle, eğer hiçbiri predominant olmazsa Heuristic 1'e dön; aksi taktirde F_i'yi sil, S_{Pi}^{-} dekileri silip silmemek için karar ver.

Heuristic 3 – (başlangıç noktası) En yüksek $SU_{i,c}$ değerli feature her zaman predominant feature'dur ve başlangıç noktası olarak kullanılabilir.

Geliştirdikleri bu FCBF algoritmasının maliyeti O(M*N*logN) olmuştur. Deneysel çalışmalarında 10 ayrı data set kullanmışlardır. ReliefF, CorrSF, ConsSF algoritmaları ile kendi algoritmalarını kıyaslamışlardır ve hem reduction bakımından, hem accuracy bakımından, hem de zaman bakımından çok daha üst performanslara ulaşmışlardır.

- **A Space Optimization for FP-Growth**[4]

FP-Growth algoritmaları veri yapılarının avantajlarını kullanarak frequency mining problemlerini divide-conquer metodu ile çözer. Klasik bir DFS ile BFS'den farklıdır. Ancak orijinal FP-Growth algoritmasının hafıza tüketimi fazladır. Bu çalışmada hem hafıza tüketimini azaltmak hem de arama hızını artırmak için arama alanı optimizasyonu ve bir kaç minor geliştirmeler yapılmıştır.

Arama alanı optimizasyonunda geliştirdikleri algoritma ile FP-Tree'yi bir Tree'ye aktarıp, prefix path'leri de hesaplamışlardır. Geliştirdikleri prosedür ile intermediate structure üretmeksizin conditional tree'deki sembolleri saymışlar ve en çok tekrar eden elemanları oluşturmuşlardır. Tree'deki transformed prefix path'leri örüntü (pattern) olarak Treeβ'ye eklemişlerdir. COUNT-PREFIX-PATH algoritması geliştirmişler ve bu algoritma verilen düğümün prefix path'lerini taramıştır. Bu algoritma conditional tree'deki sembol sayılarını hesaplar ve optimizasyon uygulanabilirliğini kontrol etmek için bir gözlemcidir. Başka bir algoritma ile oluşturulan patternler FP-Tree'ye eklenir.

Intermediate Conditional Pattern Base Structure implemantasyonları önce ağacı tarar ve transformed prefix path(TPP)'li linked list'ler oluşturur. Sonra linked listteki en sık kullanılan elemanlardan itemset oluşturur. 2. Iterasyonda ise tüm TPP'leri ekler. Bu gibi implementasyonlarda, TPP'ler 2 defa kopyalanır ve 3 kez iterate edilir. (Biri ağaca, iki tanesi conditional pattern list'e). Buna karşılık bu çalışmada geliştirilen algoritmalar ile pahalı kopyalama işlemleri bulunmamaktadır ve Pattern Base'ler ağaçta 2 defa taranmaktadır. Bunun yanında ekstra depolama alanları yok edilerek hafıza tüketimi ciddi oranda azaltılır.

Yapılan minör değişiklikler ise single path'deki seyrek düğümlerin ağaçtan budanması, tüm olası kombinasyonların single path'de listelenmesi yerine sonraki nesillerin kompakt gösterimden faydalanmasıdır.

Algoritmalarını GNU g++ compiler'da C++ dili ile yazmışlar ve test için 5 ayrı veri seti kullanmışlardır. Hafıza tüketimini en iyi durumda 2.4 kat azaltmışlardır, en kötü durumda ise hafıza tüketimi azalmamıştır. Hız bakımından ise 3 veri setinde %28.5'e kadar daha hızlı çalıştırmışlardır ancak diğer 2 datasette %10-27 arası daha yavaş çalışma ile karşılaşılmıştır. Genel olarak bakarsak hafıza tüketimini azaltmışlardır ve arama alanı optimizasyonunun hızı çok fazla etkilemediğini gözlemlemişlerdir.

- **PFP: Parallel FP-Growth for Query Recommendation[5]**

Bu çalışmada, sorgu önerisi için dağıtılmış makinelerde paralel olarak çalışan FP-Growth algoritması yapısı amaçlanmış ve bunun için FP-Growth'dan türetilen algoritma PFD olarak adlandırılmıştır. PFP, problemi, her makinenin bağımsız bir madencilik grubunu yürütecek şekilde bölümlere ayırır. Bu bölme, makineler arasındaki hesaplamalı bağımlılıkları ve dolayısıyla aralarındaki iletişimi ortadan kaldırır.

Büyük ölçekli bir madencilik görevini bağımsız hesaplama görevlerine akıllıca kesen ve MapReduce işlerine eşleyen, PFP-Growth algoritmasına uygun bir MapReduce yaklaşımı benimsenmiştir.

PFP 5 adımdan oluşur.

1. **Parçalama**: Veriseti ardışık olarak P adet parçaya bölünüp P farklı bilgisayarda saklanır.
2. **Paralel Sayım** : Verisetinde görünen tüm öğelerin destek değerlerini saymak için bir MapReduce geçişi yapılır. Her mapper bir veri seti parçasını girer. Sonuç F-listesinde saklanır.
3. **Maddelerin Gruplandırılması**: F-Listte bulunan tüm öğeleri Q adet gruba ayırır. Bu grupların olduğu listeye grup listesi(G-list) adı verilir ve her grubun unique id değeri vardır. Bu adım bir bilgisayarda birkaç saniyede tamamlanabilir.
4. **Paralel FP-Growth**: PFP'nin kilit basamağıdır. Bu adım, bir MapReduce geçişini gerçekleştirir; burada, eşleştirme aşaması ve indirgeme aşaması farklı önemli performans gösterir.

   Eşleştirici - Grup bağımlı işlemler üretme: Her mapper örneği 1. Adımda oluşturulan bir veriseti parçasına sahiptir. İşlemleri gerçekleştirmeden önce G-listesini okur. Ardından eşleştirme algoritmasını(mapping algorithm) çalıştırır.

İndirgeyici – Bölünmüş parçalar üzerinde FP-Growth : Her indirgeyici örneği, bir veya daha fazla gruba bağlı parçayı tek tek işlemek üzere atanır. Her parça için, indirgeyici örneği yerel bir FP ağacı oluşturur ve şartlı FP ağaçlarını tekrar tekrar büyütür.

5. **Toplama**: Adım 4'te üretilen sonuçlar son olarak bir araya getirilir.

PFP algoritmasını, 802939 Web sayfası ve 102 107 etiket içeren geniş bir veri seti üzerine test edildiğinde, PFP'nin neredeyse doğrusal bir hızlanma sağlayabileceği gösterilmiştir. Ampirik çalışma da PFP'nin arama motorlarında kullanılan sorgu önerisini desteklemek için umut verici olduğunu göstermiştir.

- **An Improved Association Rule Mining with FP-Tree Using Positive and Negative Integration[6]**

Geleneksel FP-Growth algoritması, Appriori algoritmasına oranla çok daha hızlı çalışır. Ancak FP-Growth algoritmalarının temel sıkıntısı yüksek miktarda conditional FP-Tree oluşturmasıdır. Bu durum pozitif - negative entegrasyon ile çözülebir.

Yapılan bu çalışma 2 aşamadan oluşmaktadır. İlk aşamada itemlerin korelasyon değeri hesaplanırken, 2. aşamada ise bu korelasyon değerlerine göre sınıflandırma yapılmaktadır. Bu sınıflandırma sonucunda ise ağaç budama işlemi gerçekleştirilmektedir.

Sınıflandırma yapılırken aşağıdaki korelasyon denklemine bağlı kalınarak işlem gerçekleştirilir:

$$\mathrm{Corr}(x,y) = \sup(x \cup y)/\sup(x)\sup(y) \qquad (1)$$

$$\mathrm{Corr}(x,y) = \begin{cases} \mathrm{Corr}(x,y) = 1, & x \text{ ve } y \text{ bağımsızdır.} \\ \mathrm{Corr}(x,y) > 1, & x \text{ ve } y \text{ pozitif korelasyona sahiptir.} \\ \mathrm{Corr}(x,y) < 1, & x \text{ ve } y \text{ negatif korelasyona sahiptir.} \end{cases} \qquad (2)$$

Bu makalede çalışma java ile kodlamışlardır. Elde ettikleri sonuçlara göre, geliştirdikleri algoritma daha büyük datasetlerle başa çıkabilmekte ve orijinal algoritmaya göre daha iyi performans sunmaktadır.

- **Improved Algorithm for Frequent Item sets Mining Based on Apriori and FP-Tree[7]**

APFT algoritmaları, FP-Growth yaklaşımı ile Apriori yaklaşımının kombinlenmesi ile oluşur. Korelasyonun temelinde, ilişkilendirme kuralının temelini sağlasa bile aralarındaki ilişkinin zayıf olmamasına bakar. Örneğin, 100 transaction'dan 50 tanesinde A, 50 tanesinde B ve 5 tanesinde AB olsun. Bu örnekte minimum support değeri 5 ise, örnek ilişkilendirme kuralını sağlar ancak dikkatli baktığımızda A varken B'nin olması, B varken A'nın olması %10 ihtimaldir ve bu zayıf bir bağdır. Korelasyonun temelinde bu zayıf bağların ilişkilendirme kuralını engellemek vardır.

Bu çalışmada, yapılan APFT algoritmalarına korelasyon ekleyip bazı küçük değişikler yaparaki ilişkilendirme işleminde kullanılabiliecek yeni bir algoritma olan APFTC (APFT with correlation) algoritması elde edilmiştir.

Apriori algoritmasının genel sıkıntısı yüksek sayıda aday örneğinin olması ve tekrar tekrar database'in taranmasıdır. APFT algoritmalarında ilk olarak FP-Tree oluşturulur daha sonra ise Apriori algoritmasını kullanarak sık tekrar eden elemanlar tespit edilir. Yapılan deneylerde APFT'nin Apriori'den daha hızlı ve neredeyse FP-Growth kadar da hızlı olduğu gözlemlenmiştir.

Bu çalışmada a ile b'nin arasında ilişkilendirme olması için aşağıdaki korelasyon denklemini sağlaması gerekmektedir:

$P(ab) > P(a)P(b)$ (3)

Bu denklemde bulunan;

P(ab): a ve b'nin birlikte olduğu transaction sayısı / toplam transaction sayısı

P(a): a'nın bulunduğu transaction sayısı / toplam transaction sayısı

P(b): b'nin bulunduğu transaction sayısı / toplam transaction sayısı

ifade etmektedir.

Formülden dolayı ilişkilendirmelere ikili olarak bakılmaktadır. Çalışmada gerçekleştirilen bir başka değişiklik ise original paper'ın aksine her branch'ın support değerlerini hesaplarken, traversing tree yöntemini kullanmayarak bir N-Table ile hesaplamanın yapılmasıdır.

Çalışma 2 dataset (989 item) kullanarak test edilmiş ve geliştirdikleri APFTC Algoritmasının, APFT ve FP-Growth algoritmalarından daha hızlı olduğu gözlemlenmiştir. Ayrıca APFTC'nin diğer algoritmalarla aynı sayıda yüksek ilişkili item bulduğu da gözlemlenmiştir.

- **A Comparative Study of Association Rules Mining Algorithms[8]**

  Bu makalede Apriori, FP-Growth ve Dynamic FP-Growth methodlarının performansı karşılaştırılmıştır. Apriori algoritması, veritabanını tekrarlı olarak tarayarak yaygın aday öğe kümesi oluşturup bunları test etmeye dayalıdır. Aday küme oluşturma ve bu kümeyi test etme olayları sebebiyle maliyeti çok yüksektir.

  FPGrowth algoritması, verilen ağaç yapısının tek yol veya hiç öğe bulunmayasıya kadar koşullu ağaç oluşturmasını sağlayıp, tek yol (single path) içerdiğinde yaygın öğeleri bu dalı oluşturan öğelerin kombinasyonu olarak bulur.

  DynFP-Growth ise FP-Growth algoritmasının sıralama kısmında dinamik işlemler kullanılarak geliştirilmiş halidir. Bu algoritma sayesinde veritabanı güncellendiğinde FP-tree yapısını tekrar inşa etmek gerekmez.

  Çalışmada test aşamasında, algoritmalar Java dilinde kodlanmış ve birçok data üzerinde denenmiştir. 10.000 - 500.000 arasında işlem içeren verisetleri kullanılmıştır. Test verisinde öğe sayısı 100 olarak, maximum sık tekrarlanan öğe sayısı 3000 olarak, ortalama işlem boyutu ise 10 olarak belirlenmiştir.

  İşlem sayısı 40000 olan veriseti üzerinde destek faktörünün etkisi incelenmek için bir deney yapılmıştır. Burada destek faktörü %5 ile %40 arasındaki değerler olarak belirlenmiştir. Düşük destek faktörü değerlerinde en kötü performans Apriori algoritmasının olurken en iyi performans DynFP- Growth algoritmasının olmuştur. Fakat destek faktörü %30'dan fazla olduğunda en iyi performans yine DynFPGrowth olurken, en kötü performans FPTree algoritmasında gözlenmiştir.

  Bir başka deneyde destek faktörü %40 olarak sabitlenip işlem sayısı 150000 olarak değiştirilmiştir. Burda en kötü performans Apriori algoritmasında çıkarken en iyi performans DynFPGrowth algoritmasının olmuştur.

  Bu deneyler ile algoritmalar üzerinde destek faktörü ve veriseti boyutu etkisini

incelenmiştir. Apriori algoritmasında destek faktörü arttıkça performans yükselirken, veriseti boyutu arttıkça performans düşmüştür. FPGrowth algoritmasında destek faktörünün performansa etkisinin çok az olduğu, veriseti boyutu arttıkça performansın azaldığı görülmüştür. DynFPGrowth algoritmasında ise destek faktörünün performansa etkisinin neredeyse hiç olmadığını, veriseti boyutu arttıkça performansın azaldığı görülmüştür.

2.İLİŞKİLENDİRME/BİRLİKTELİK KURALLARI (ASSOCIATION RULES)

Veri kümesi içindeki yaygın örüntülerin (pattern) ve nesneleri oluşturan öğeler arasındaki ilişkileri bulan kuraldır. İlişkilendirme kuralına günlük hayatta fazlasıyla rastlarız. Örnek olarak, bir markete gittiğimizde ketçap ile makarnanın aynı rafta olduğunu görürüz. Bunun nedeni daha önceki müşterilerin yaptığı alışverişlerde, makarna alanların büyük çoğunluğu ketçap da aldığı gözlemlenmiştir. Pazarlama açısından bu iki ürün aynı rafta sunulmaktadır.

İlişkilendirme kurallarında kullanılan temel birkaç terimi açıklayalım.

Destek sayısı (support count), bir öğeler kümesinin veri kümesindeki görülme sıklığıdır. Destek (support) ise bir öğeler kümesinin içinde bulunduğu hareketlerin toplam hareketlere oranıdır. Güven (confidence), 2 öğeler kümesinin birlikteliğinin ikisinden birine olan oranıdır.

Örneğin, elimizde 5 tane market alışveriş örneği olsun:

TID	*Öğeler*
1	Ekmek, Süt
2	Ekmek, Bez, Bira, Yumurta
3	Süt, Bez, Bira, Kola
4	Ekmek, Süt, Bez, Bira
5	Ekmek, Süt, Bez, Kola

Tablo 2.1: Alışveriş listesinden destek ve güven değeri hesaplama[9]

Bu örnek için;

- Destek sayısı = $^{\delta}$ =({Süt, Bira, Bez}) = 2
- Destek = s({Süt, Bira, Bez}) = 2/5
- Güven c = $^{\delta}$(Süt, Bez, Bira) / $^{\delta}$(Süt,Bez) = 2/3

olarak bulunmuştur.

Bir öğe grubunun yaygın öğe grubu (Frequent itemset) olabilmesi için minimum destek, eşik değerine eşit veya daha büyük olmalıdır. İlişkilendirme kuralının geçerli olabilmesi için, minimum destek ve güven değerlerini sağlaması gerekir. İlişkilendirme kuralı X → Y şeklinde gösterilir.

Örnek olarak;

Transaction-id	Items bought
10	A, B, D
20	A, C, D
30	A, D, E
40	B, E, F
50	B, C, D, E, F

Tablo 2.2: Itemlerin satın alımına göre ilişkilendirme kuralı hesaplanması[9]

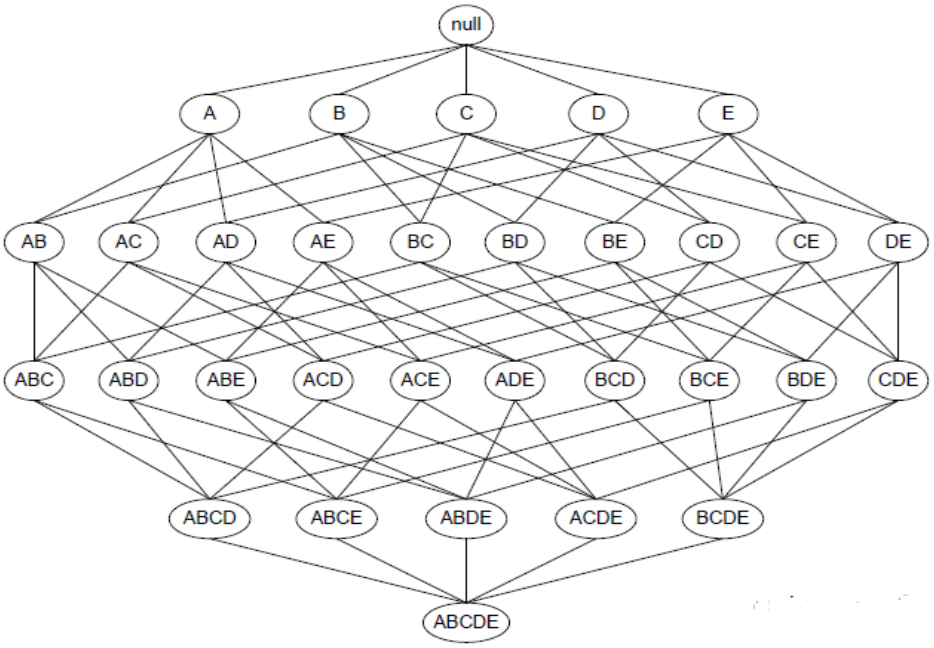
Verilerine göre minimum destek = %50 ve minimum güven = %50 için,

- A → D (destek = %60, güven = %100)
- D → A (destek = %60, güven = %75)

İlişkilendirme kuralı başlıca 3 yaklaşımla oluşturulabilir:

2.1. Brute-force Yaklaşım

Bu yaklaşımda olası tüm kurallar listelenir. Her kural için destek ve güven değerleri hesaplanır. Ardından minimum güven ve destek eşik değerlerini sağlamayan kurallar silinir. Her ne kadar bu yaklaşım doğru sonucu verse de çok fazla aday öğe oluşturmaktadır (n öğe için 2^n-1 öğe oluşturabilir). Bu da maliyeti son derece artırmaktadır.



Şekil 2.1: A,B,C,D,E elemanlarından oluşan kümenin ilişkilerinin brute-force algoritması ile çıkarılması[9]

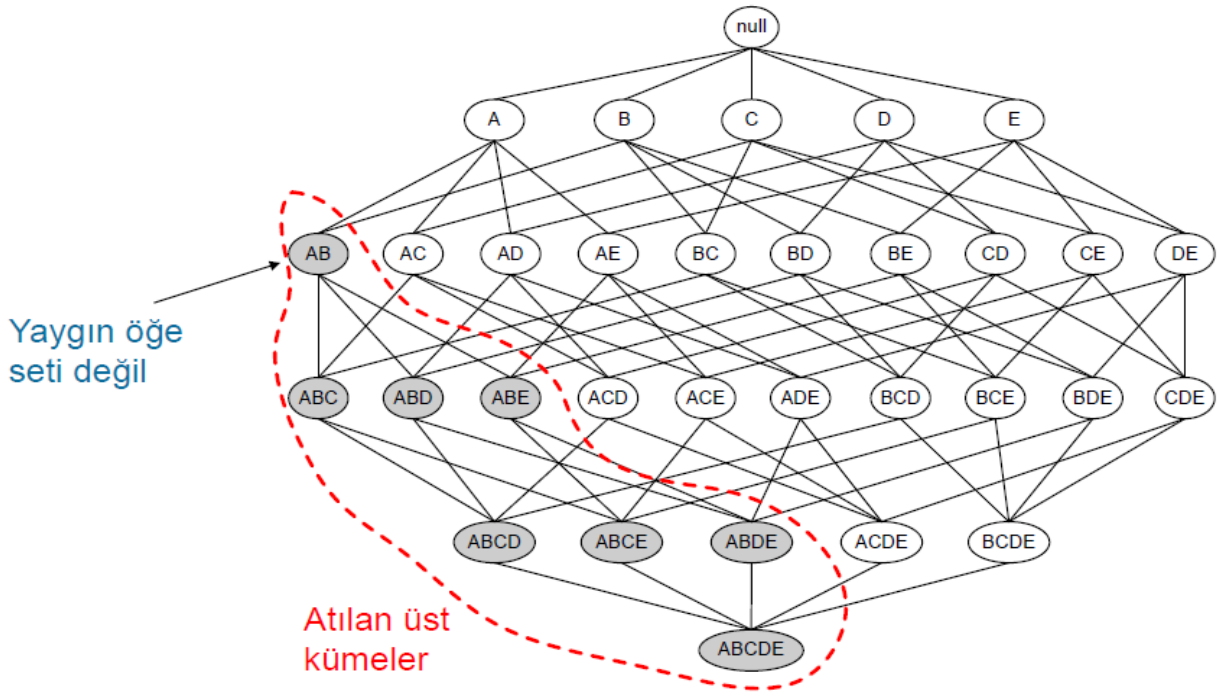
2.2.Apriori Yöntemi

Agrawal ve Srikant tarafından 1994 yılında geliştirilmiştir. Yaklaşım, bir öğeler kümesinin destek değeri, bu kümenin herhangi bir altkümesinin destek değerinden büyük olamaz mantığına dayanmaktadır.

$$\forall X,Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

Yukarıdaki önermeden şu sonucu çıkarabiliriz, bir yaygın öğenin herhangi bir alt kümesi de yaygın öğedir. {x,y,z} kümesi bir yaygın öğe kümesi ise {y,z} kümesi de yaygın öğedir.

Bu yaklaşımda, yaygın öğe olmayan bir kümenin üst kümelerinin de yaygın öğe olmayacağı kabul edilir ve o kümelerin destek değeri hesaplanmaz. Böylelikle yaygın olmayan öğeler elenir.



Şekil 2.2: A,B,C,D,E elemanlarından oluşan kümenin ilişkilerinin apriori algoritması ile çıkarılması[9]

Algoritma Adımları:

1. Uzunluğu k=1 olan yaygın öğe setlerini oluştur

2.yeni yaygın öğe seti kalmayana kadar tekrarla

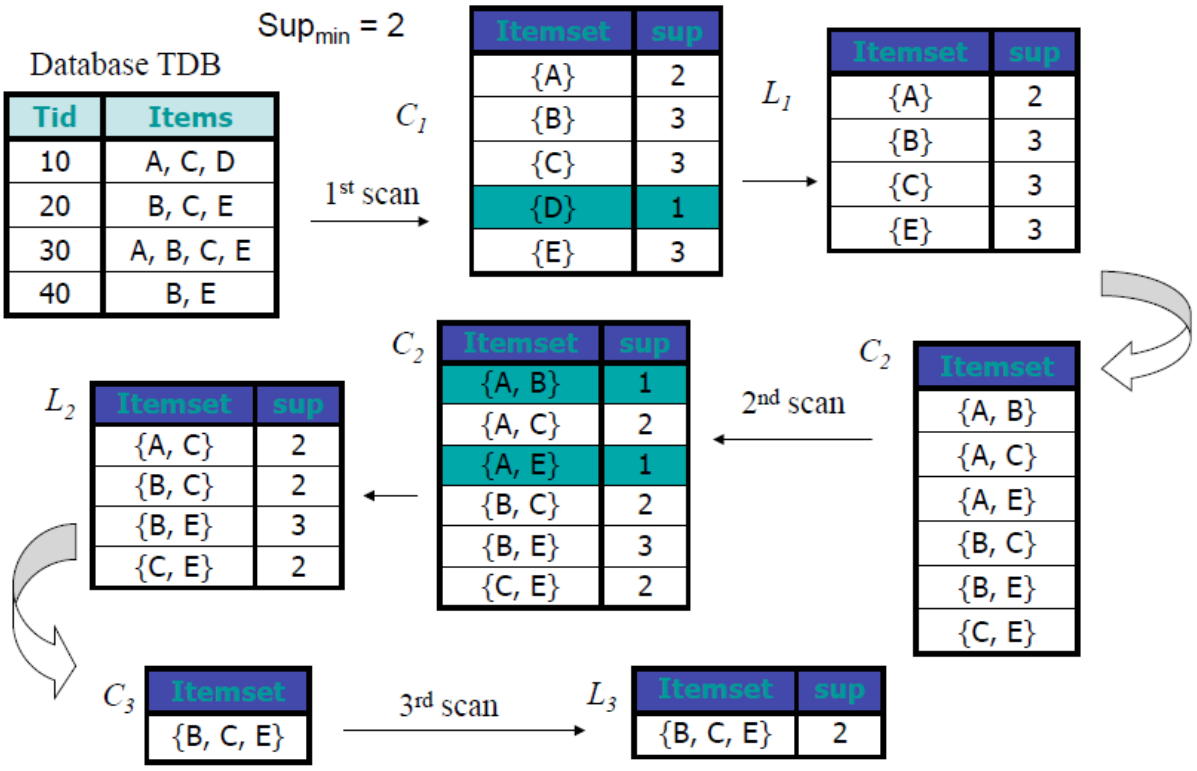
2.1. k uzunluğundaki yaygın öğe setlerinden k+1 uzunluğundaki aday öğe setlerini oluştur

2.2. k uzunluğundaki yaygın olmayan altkümelere sahip aday öğe setlerini ele

2.2.1. veri tabanını tarayarak aday öğe kümelerinin destek değerlerini hesapla

2.2.2yaygın olmayan aday öğe kümelerini ele ve sadece yaygın olanlarla devam et

Örnek olarak:



Tablo 2.3: Apriori algoritmasının uygulanma örneği[10]

Apriori sayesinde sadece güçlü ilişkilendirme kuralları oluşur. Üstelik brüte-force yaklaşıma göre çok daha hızlıdır. Ancak sürekli veri tabanını taradığımız için yine de yüksek bir maliyeti vardır.

2.3.FP-Tree Algoritması

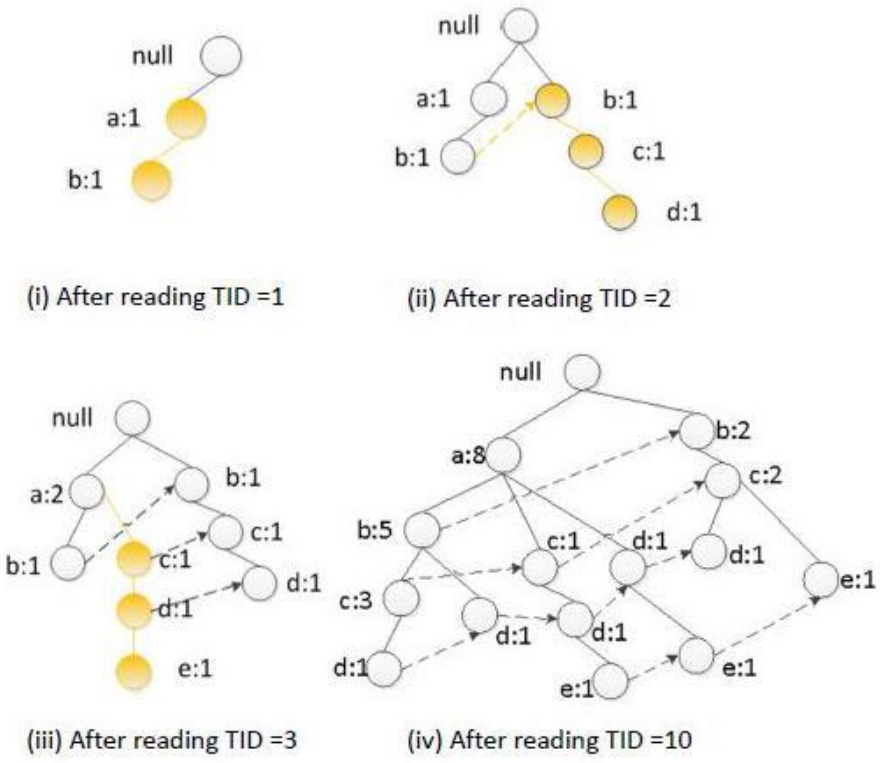
Fp-Tree algoritması büyük veri kümelerinde çalışmaktadır ve sistem kaynaklarını verimli bir şekilde kullanan birliktelik kuralı çıkarım algoritmasıdır. Fp-Tree algoritması kendisi ile aynı işi yapan diğer birçok algoritmaya göre daha performanslı çalışır ve maliyeti azaltır, algoritma böl ve yönet stratejisini kullanmaktadır. En önemli özelliği tüm veri tabanını sıkıştırılmış bir ağaç veri yapısı şeklinde tutmasıdır. Algoritma veri tabanını toplamda iki kez tarar, ilk taramada bütün öğelerin destek değerini hesaplar, ikinci taramada ağaç veri yapısını oluşturur. Fp-Tree algoritması yaygın öğeler kümesini bulurken aday öğeler kümesi kullanmaz, yaygın öğeler Fp-Tree yapısından bulunur.

FP-Tree Algoritması 3 aşamadan oluşur:

1. Veri tabanı bir kez taranarak 1 yaygın öğe bulunur.
2. Yaygın öğeler destek sayısına göre büyükten küçüğe doğru sıralanır.
3. Veri tabanı bir kez daha taranarak ağaç yapısı oluşturulur.

Örneğin;

TID	Items
1	{a,b}
2	{b,c,d}
3	{a,c,d,e}
4	{a,d,e}
5	{a,b,c}
6	{a,b,c,d}
7	{a}
8	{a,b,c}
9	{a,b,d}
10	{b,c,e}



Şekil 2.3: FP-Tree algoritmasının uygulanma örneği[11]

Böl ve yönet (divide and conquer) mantığıyla çalışır. Toplamda veri tabanı 2 kez tarandığı için diğer algoritmalara göre çok daha hızlıdır.

3.PROJENİN GELİŞTİRİLMESİNDE KULLANILAN ARAÇLAR

Projemizde belirli ilişkilendirme/birliktelik kuralı algoritmalarını iki adet veri seti üzerinde çalıştırarak, bu algoritmalar arasında doğruluk oranları ve çalışma hızları bakımından karşılaştırmalar yapılıp bu sonuçlar analiz edilmiştir. Bu bölümde bu verisetleri ve kullanılan algoritmalardan bahsedilmiştir.

3.1.Kullanılan İlişkilendirme/Birliktelik Kuralı Algoritmaları

3.1.1.FCBF

Projemizde, Arizona State University'deki Lei Yu ve Huan Liu'nun yazdığı "Feature Selection for High-Dimensional Data: A Fast Correlation-Based Filter Solution" isimli çalışmasında[3] bilim dünyasına duyurdukları FCBF algoritmasını kullanacağımız merkez algoritma olarak belirledik. Çalışmamızda diğer ilişkilendirme kuralı algoritmalarını, bu algoritma ile karşılaştıracağız. Bu algoritmayı seçmemizde diğer algoritmalara göre daha az zaman karmaşıklığına sahip olması, lineer olmayan ilişkilerde güçlü bağlar kurabilmesi gibi özellikler başlıca nedenlerimizdir.

Kullandıkları bilgi teorisi temelli yaklaşım korelasyonu, diğer korelasyon türlerine göre daha iyi ön eleme yapmaktadır. Çok yoğun veri setleri üzerinde bile hızlı ilişkilendirme yapabilmektedir.

FCBF algoritmasının IOT verileri için uygun bir algoritma olduğunu ispatlamak için, FCBF algoritması ile başka algoritmaları iki veritabanında deneyerek doğruluk oranları ve çalışma hızları bakımından kıyaslayıp analiz ettik.

FCBF algoritmasının performansını kıyaslamak için kullandığımız algoritmalar aşağıda anlatılmıştır.

3.1.2.NDFS(Nonnegative Discriminative Feature Selection)

Yi Yang ve Jing Liu tarafından geliştirilen gözetimsiz öğrenme türüdür. Hayali bulutlama yöntemiyle verileri sınıflandırıp arasındaki korelasyonu bulmaktadır. Aralıklı öğrenme (sparse learning) temelli yaklaşımdır.[12]

3.1.3.Lap_Score(Laplacian Score for Feature Selection)

Xiaofei He, Deng Cai ve Partha Niyogi tarafından geliştirilmiştir. Herhangi bir makine öğrenme algoritmasından bağımsız bir şekilde çalışabilen, verilerin laplace skorlarına göre

sınıflandıran bir algoritma sunmuşlardır. Algoritmaları hem gözetimli öğrenme hem de gözetimsiz öğrenme olarak çalışabilmektedir.[13]

3.1.4.JMI (Joint Mutual Information)

John Moody ve Howard Hua Yang tarafından geliştirilmiştir. Yüksek boyutlu verilerde gereksiz verileri elemek için entropiden yararlanmayı temel alan yaklaşıma sahiptir. Gözetimli öğrenme türüdür.[14]

3.1.5.MRMR (Minimal-Redundancy-Maximal-Relevance)

Hanchuan Peng ve Fihui Long tarafından, karşılıklı bilgi edinimde maksimum bağlılığı temel alan bir filtre geliştirilmiştir. Bu filtre ile özellik eleyici olan "wrapper" kombin edilip 2 aşamalı bir gözetimli öğrenme algoritması oluşturulmuştur.[15]

3.1.6.MIM (Mutual Information Maximization)

David Lewis tarafından text kategorizasyonu için geliştirilmiştir. İstatiksel sınıflandırma ve orantılı atama stratejisi kullanılarak oluşturulan gözetimli öğrenme algoritmasıdır.[16]

3.1.7.MIFS (Mutual Information for Selecting)

Yapay sinir ağları verilerinin korelasyonu için oluşturulan Roberto Battiti'nin geliştirdiği yöntemdir. Yöntem, bilgi teorisi tabanlı olup verilerin karşılıklı bilgilerine "greedy" yaklaşımını içerir.[17]

3.1.8.DISR (Double Input Symmetrical Relevance)

Patrick Meyer ve Colas Schretter tarafından ortaya konan değişken tamamlayıcılık ölçüsüne dayanan ilişkilendirme kuralı algoritmasıdır. Bir dizi değişkenin çıktısının bir değişkene olan bağlılığının toplamına göre hareket eden algoritma, entropiden yararlanır.[18]

3.1.9.CIFE (Conditional Informative Feature Extraction)

Bu algoritma 2 temel üzerinde duruyor: "Class-Relevance" ve "Redundancy". CIFE, özellikler boyunca "redundancy"i azaltıp "class-relevance"ı maksimize ediyor. "Parzen window" yardımıyla da bize gözetimli öğrenme temelli ilişkilendirme kuralı algoritması sunuyor.[19]

3.1.10.CMIM (Conditional Mutual Information Maximization)

Bilgi teorisi temelli olan bu algoritma François Fleuret tarafından geliştirilmiştir. CMIM özellik seçerken, sınıfın önceden seçilmiş herhangi bir özelliğin koşullarını tahmin etmesi ile karşılıklı bilginin maksimize edilmesine dikkat eder.[20]

3.1.11.low_variance

Veri sütunlarının ne kadar bilgi içerdiği sütunun varyans değerine bağlıdır. Bu algoritma da özellik elerken her satırın varyans değerini hesaplayıp, verilen değerden küçük olan sütunları silmektedir.[21]

3.1.12.SPEC (Spectral Feature Selection)

Huan Liu ve Zheng Zao tarafından geliştirilmiştir. Verilerin doğal (intrinsic) değerleriyle ilgilenen bu algoritma hem gözetimli öğrenme hem de gözetimsiz öğrenme algoritmaları altında çalışabilmektedir. Çalışması Spektral Graph Teoremine dayanmaktadır.[22]

3.1.13.RFS (Robust Feature Selection)

Verilere ayrık $l_{2,1}$-norm minimizasyonu uygulayarak anlamlı özellikleri çıkarıp gürültülü verilere temizlemeye yarayan RFS algoritmasını Xiao Cai ve Chris Ding geliştirmiştir. Aynı zamanda ayrık aralıklar yöntemini de kullanarak daha güçlü bir eliminasyon yapmıştırlar.[23]

3.2.Veri Setleri

Kullandığımız veri setlerinden biri, Princeton Üniversitesi Onkoloji bölümü tarafından toplanan kolon kanseri veri setidir. Bu veri seti kolon kanseri teşhisinde kullanılan verileri barındırır. Her numune için, bunun bir tümör biyopsisinden gelip gelmediği gösterilir. Bu veri, 62 örneğe sahip olan ve her örnekte de 2000 özellik bulunduran bir yapıya sahiptir.[24]

Üzerinde çalışma yaptığımız diğer veri seti ise, Golub TR, Slonim DK, Tamayo P, Huard C, Gaasenbeek M, Mesirov JP, Coller H, Loh ML, Downing JR, Caligiuri MA, Bloomfield CD, Lander ES tarafından "Molecular classification of cancer: class discovery and class prediction by gene expression monitoring" çalışması ile toplanmış, lösemi hastalığı veri setidir. Bu veri setinde löseminin teşhisinde kullanılan özellikleri barındırır. Bu veri 72 örneğe sahip olup, her örnekte 7070 özellik barındırmaktadır.[25]

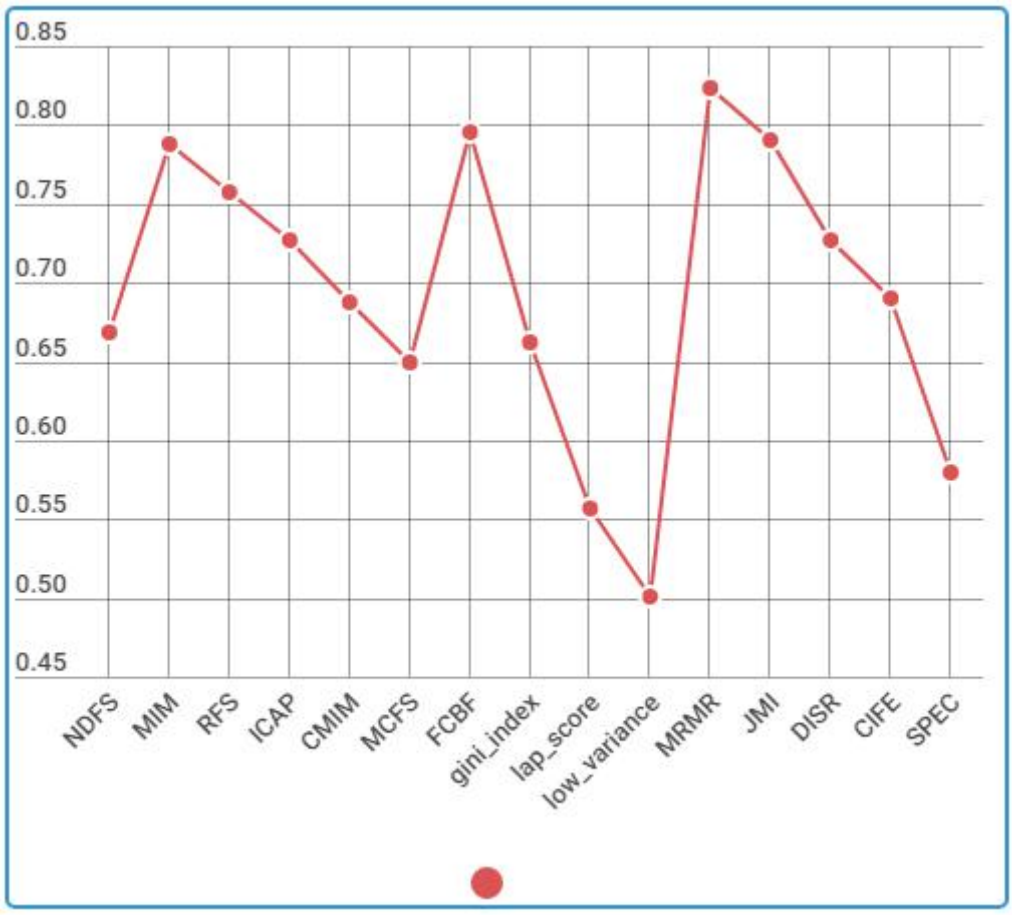
4.DENEY SONUÇLARI

4.1. Kolon Kanseri Veri Seti Üzerinde Yapılan Deney Sonuçları

Kolon kanserleri teşhisinde kullanılan veri seti üzerinde, 15 adet farklı ilişkilendirme kuralı algoritmasını deneyerek, bu algoritmalar aralarında karşılaştırma yapılarak analiz edilmiştir.

4.1.1.Doğruluk Oranı Karşılaştırması

Bu algoritmalar arasında karşılaştırma yaparken esas alınan kriterlerden biri, algoritmanın veri seti üzerinde çalıştığında elde ettiği doğruluk oranıdır. Bir problem için kullanılacak algoritma seçiminde doğruluk oranı, bu seçimdeki belirleyici etkenlerden biridir. Algoritmalardan beklenen, doğruluk oranının kabul edilebilir bir değer olması ve bu değerin olabildiğince yüksek olmasıdır. En yüksek doğruluk değerine sahip olan algoritma, bu veri seti üzerinde en az hata yapan ve en fazla doğru sonuca ulaşan algoritma denebilir.

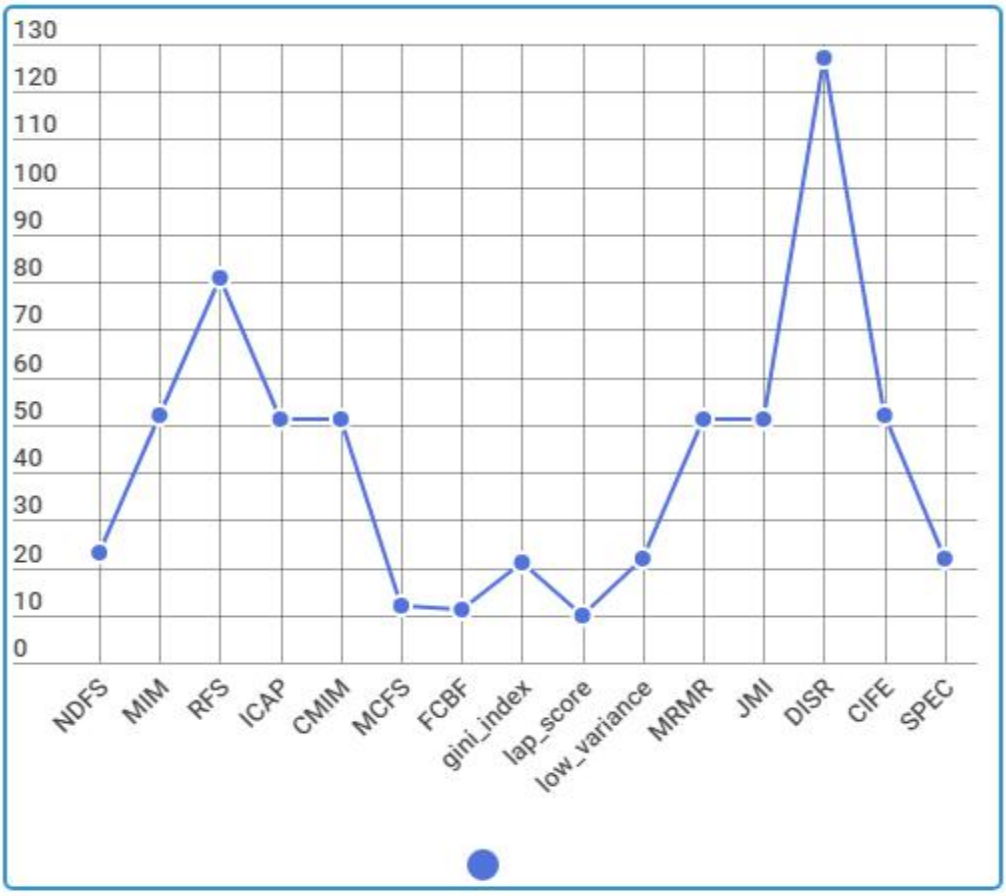


Grafik 1: İlişkilendirme Algoritmalarının Kolon Kanseri Veri Seti Üzerindeki Doğruluk Oranı Grafiği

Grafik 1'de kullanılan 15 ilişkilendirme kuralı algoritmasının doğruluk oranları verilmiştir. Bu grafiğe bakılarak en iyi doğruluk skoruna sahip algoritmanın MRMR, ardından FCBF algoritması olduğu görülmüştür. MRMR algoritmasının doğruluk oranı 0.82 iken FCBF algoritmasının doğruluk oranı 0.80 olmuştur. Aynı şekilde en düşük doğruluk oranlarına sahip algoritmalar ise 0.50 ile low_variance, 0.56 ile lap_score ve 0.58 ile SPEC algoritmasıdır. Bu algoritmalar ile en yüksek sonuçlar veren algoritmalar arasında ciddi farklar bulunmaktadır.

4.1.2.Çalışma Hızı Karşılaştırması

Bu algoritmalar arasında karşılaştırma yaparken esas alınan bir diğer kriter ise algoritmanın veri seti üzerinde çalıştığında geçen süredir. Bir problem için kullanılacak algoritma seçiminde doğruluk oranı kadar çalışma süresi de önemli bir belirleyici etkenlerden biridir. Sonuçların hızlı üretilmesi gibi durumlarda çalışma hızı, doğruluk oranından daha önemli hale gelmektedir. Algoritmalardan beklenen, çalışma süresinin kabul edilebilir bir değer olması ve bu sürenin olabildiğince az olmasıdır. Kabul edilebilir sürenin üzerinde çalışan algoritmalar, genellikle çözüm için seçilmezler.



Grafik 2: İlişkilendirme Algoritmalarının Kolon Kanseri Veri Seti Üzerindeki Çalışma Süreleri Grafiği

Grafik 2'de kullanılan 15 ilişkilendirme kuralı algoritmasının çalışma süreleri(sn cinsinden) verilmiştir. Bu grafiğe bakılarak en az çalışma süresine sahip yani en hızlı çalışan algoritmanın lap_score, ardından FCBF algoritması olduğu görülmüştür. Lap_score algoritması 10.9 saniyede çalışırken FCBF algoritmasının çalışma süresi 11.1 olmuştur. İki algoritma arasında 0.2 saniye vardır. Aynı şekilde en yüksek çalışma süresine sahip olanlar yani en yavaş çalışan algoritmalar ise 127.9 saniye ile DISR, 81.5 saniye ile RFS ve 52.9 saniye ile MIM algoritmasıdır. Bu algoritmalar ile en yüksek sonuçlar veren algoritmalar arasında ciddi farklar bulunmaktadır. FCBF algoritması, en yavaş algoritma olan DISR algoritmasına kıyasla yaklaşık 13 kat daha hızlı verileri ilişkilendirmiştir.

Çalışmamızda temel olarak ilgilendiğimiz ve diğerleri ile kıyasladığımız algoritma FCBF algoritmasıdır. Bu algoritmayı yukarıda bahsi geçen iki kriteri birlikte dikkate alarak diğer algoritmalar ile kıyaslayalım.

FCBF algoritması, bu veri setinde 0.80 doğruluk oranına ulaşarak en iyi doğruluk skoruna sahip 2. algoritma olmuştur. En iyi doğruluk oranına sahip algoritma 0.82 ile MRMR algoritmasıdır. Ancak MRMR verileri 51,5 saniyede ilişkilendirirken, FCBF algoritması ilişkilendirme işlemi için sadece 11.1 saniye harcamıştır. Bu da FCBF algoritmasının doğruluk oranı bakımında en yakın rakibinden yaklaşık 5 kat daha hızlı çalıştığının göstergesidir.

Diğer kriterimiz olan çalışma hızına göre ise FCBF algoritması 11.1 saniye ile en hızlı çalışan 2. Algoritma olmuştur. Daha hızlı çalışan ilişkilendirme kuralı algoritması olan lap_score sadece 0.2 saniye daha erken ilişkilendirmeyi tamamlayarak 10.9 saniyede çalışmıştır. Ancak lap_score algoritmasının doğruluk oranı 0.56 gibi çok düşük bir değere sahipken, FCBF algoritması doğruluk oranında 0.8 gibi bir değere sahiptir. Bu da FCBF algoritmasının hız bakımından rekabet ettiği algoritmalara göre çok daha yüksek başarı oranlarına sahip olduğunu ispatlamaktadır.

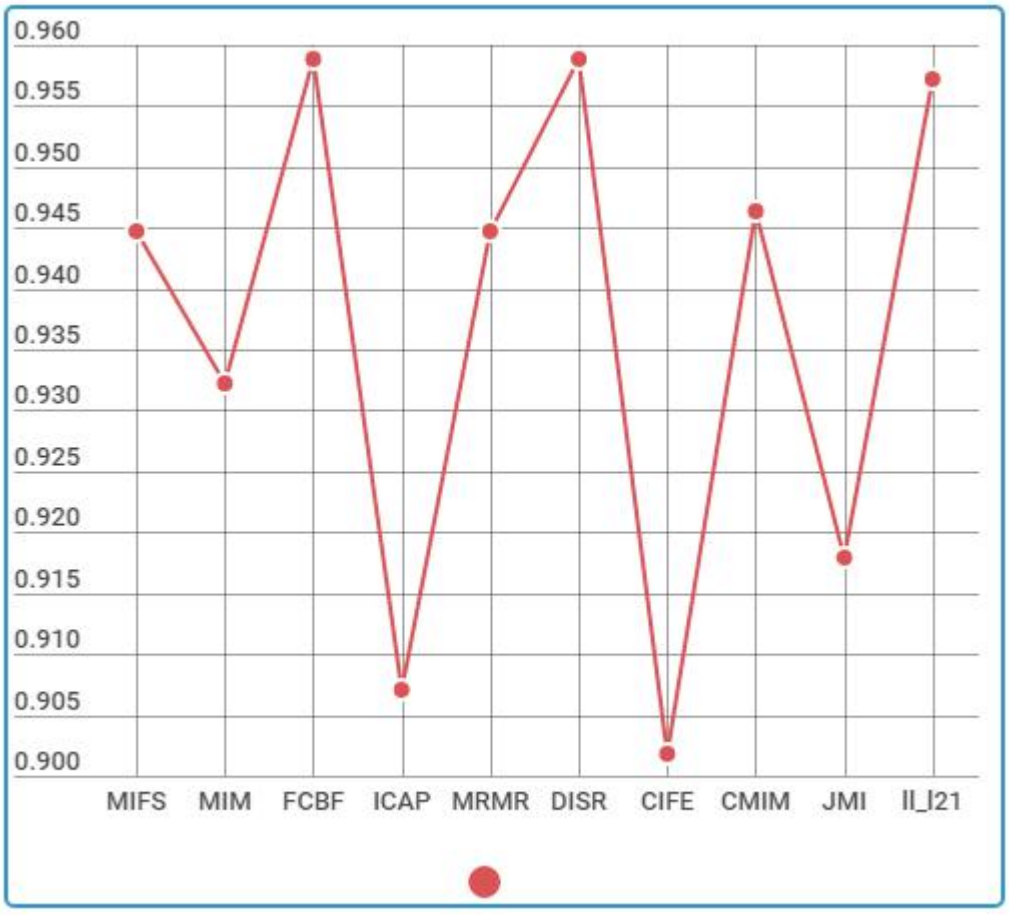
İki kritere birlikte bakıldığında kolon kanseri veri seti üzerinde ilişkilendirme için kullanılmaya en uygun algoritmanın hem hız hem de doğruluk oranı değerlerinin gayet iyi olması sebebiyle FCBF olduğu görülmektedir.

4.2. Lösemi Hastalığı Veri Seti Üzerinde Yapılan Deney Sonuçları

Lösemi hastalığı veri seti, kolon kanseri veri setine göre çok daha büyüktür. Böylece büyük bir veri seti üzerinde FCBF algoritmasının ve diğer algoritmaların performanslarını inceleme fırsatımız oldu. Lösemi hastalığı teşhisinde kullanılan bu veri seti üzerinde, 10 adet farklı ilişkilendirme kuralı algoritmasını deneyerek, bu algoritmalar aralarında karşılaştırma yapılarak analiz edilmiştir.

4.2.1.Doğruluk Oranı Karşılaştırması

Bu algoritmalar arasında karşılaştırma yaparken esas alınan kriterlerden biri, algoritmanın veri seti üzerinde çalıştığında elde ettiği doğruluk oranıdır. Bir problem için kullanılacak algoritma seçiminde doğruluk oranı, bu seçimdeki belirleyici etkenlerden biridir. Algoritmalardan beklenen, doğruluk oranının kabul edilebilir bir değer olması ve bu değerin olabildiğince yüksek olmasıdır. En yüksek doğruluk değerine sahip olan algoritma, bu veri seti üzerinde en az hata yapan ve en fazla doğru sonuca ulaşan algoritma denebilir.

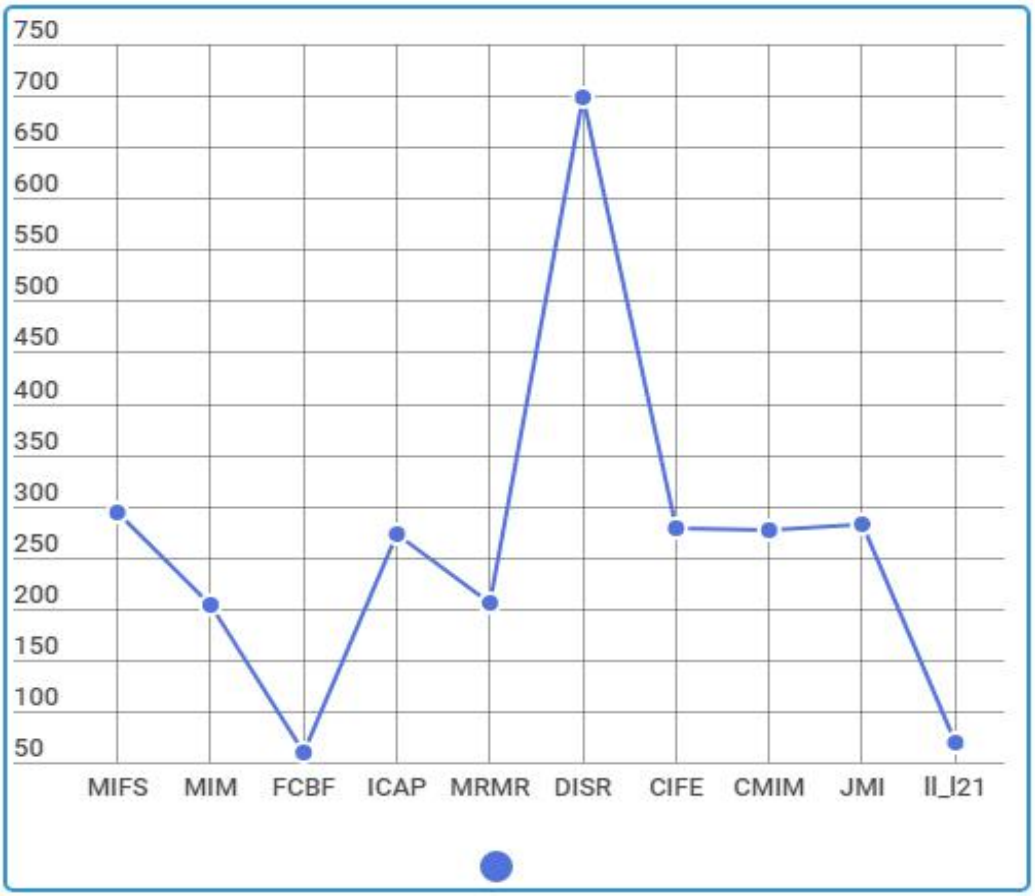


Grafik 3: İlişkilendirme Algoritmalarının Lösemi Hastalığı Veri Seti Üzerindeki Doğruluk Oranı Grafiği

Grafik 3'de kullanılan 10 ilişkilendirme kuralı algoritmasının doğruluk oranları verilmiştir. Bu grafiğe bakılarak en iyi doğruluk skoruna sahip algoritmanın FCBF, ardından DISR algoritması olduğu görülmüştür. FCBF algoritmasının doğruluk oranı 0.9589 iken DISR algoritmasının doğruluk oranı 0.9585 olmuştur. Aynı şekilde en düşük doğruluk oranlarına sahip algoritmalar ise 0.90 ile CIFE, 0.91 ile ICAP ve 0.92 ile JMI algoritmasıdır. Bu algoritmalar ile en yüksek sonuçlar veren algoritmalar arasında bir miktar fark bulunmaktadır. Bu fark az gibi gözükse de çalışma büyük veri üzerinde gerçekleştirildiği için, farkın fazla olduğu anlaşılmaktadır.

4.2.2.Çalışma Hızı Karşılaştırması

Bu algoritmalar arasında karşılaştırma yaparken esas alınan bir diğer kriter ise algoritmanın veri seti üzerinde çalıştığında geçen süredir. Bir problem için kullanılacak algoritma seçiminde doğruluk oranı kadar çalışma süresi de önemli bir belirleyici etkenlerden biridir. Sonuçların hızlı üretilmesi gibi durumlarda çalışma hızı, doğruluk oranından daha önemli hale gelmektedir. Algoritmalardan beklenen, çalışma süresinin kabul edilebilir bir değer olması ve bu sürenin olabildiğince az olmasıdır. Kabul edilebilir sürenin üzerinde çalışan algoritmalar, genellikle çözüm için seçilmezler.



Grafik 4: İlişkilendirme Algoritmalarının Lösemi Hastalığı Veri Seti Üzerindeki Çalışma Süreleri Grafiği

Grafik 4'de kullanılan 10 ilişkilendirme kuralı algoritmasının çalışma süreleri(sn cinsinden) verilmiştir. Bu grafiğe bakılarak en az çalışma süresine sahip yani en hızlı çalışan algoritmanın FCBF, ardından ll_l21 algoritması olduğu görülmüştür. FCBF algoritması 60.8 saniyede çalışırken ll_l21 algoritmasının çalışma süresi 69.2 olmuştur. Aynı şekilde en yüksek çalışma süresine sahip olanlar yani en yavaş çalışan algoritmalar ise 699.4 saniye ile DISR, 283.6 saniye ile JMI ve 278.2 saniye ile CIFE algoritmasıdır. Bu algoritmalar ile en yüksek sonuçlar veren algoritmalar arasında ciddi farklar bulunmaktadır.

Çalışmamızda temel olarak ilgilendiğimiz ve diğerleri ile kıyasladığımız algoritma FCBF algoritmasıdır. Bu algoritmayı yukarıda bahsi geçen iki kriteri birlikte dikkate alarak diğer algoritmalar ile kıyaslayalım.

FCBF algoritması, bu veri setinde 0.9589 doğruluk oranına ulaşarak en iyi doğruluk skoruna sahip algoritma olmuştur. En iyi doğruluk oranına sahip 2. algoritma 0.9585 ile DISR algoritmasıdır. Ancak DISR verileri 699.4 saniye gibi çok fazla bir sürede ilişkilendirirken, FCBF algoritması ilişkilendirme işlemi için sadece 60.8 saniye harcamıştır. Bu da FCBF algoritmasının en yakın rakibine kıyasla 10 kattan daha fazla hızlı ilişkilendirme yaptığını göstermektedir.

Diğer kriterimiz olan çalışma hızına göre ise FCBF algoritması 60.8 saniye ile en hızlı çalışan algoritma olmuştur. 2. en hızlı çalışan ilişkilendirme kuralı algoritması olan ll_l21 algoritması ise 69.2 saniyede işlemi tamamlamıştır. Ancak ll_l21 algoritmasının doğruluk oranı 0.957 gibi bir değere sahipken, FCBF algoritması doğruluk oranında 0.9589 gibi bir değere sahiptir. Bu da FCBF algoritmasının hız bakımından rekabet ettiği algoritmalara göre çok daha yüksek başarı oranlarına sahip olduğunu ispatlamaktadır. 3 dakikanın altında ilişkilendirme yapabilen sadece 2 algoritma vardır. Bu da FCBF algoritmasının diğer rakiplerine kıyasla ne kadar hızlı olduğunun ispatıdır.

İki kritere birlikte bakıldığında lösemi hastalığı veri seti üzerinde ilişkilendirme için kullanılmaya en uygun algoritmanın hem hız hem de doğruluk oranı değerlerinin en iyi olması sebebiyle FCBF olduğu görülmektedir.

5.SONUÇ

IOT, nesnenin bir şekilde internete erişip, diğer cihazlarla iletişim halinde olmasıdır. Doğal olarak, bu iletişimin akıcı olması için IOT verileri çok hızlı bir şekilde işlenmelidir. Sadece hızlı işlenmesi yetmez aynı zamanda bu iletişimin doğru bir şekilde olması gerekir. Bu yüzden IOT verileri için doğruluk oranı da çok önemli bir etmendir.

İlişkilendirme kuralı algoritmaları genel olarak veriler arasındaki lineer ilişkiye göre korelasyon yapar. Ancak veriler arasındaki ilişkiler her zaman lineer olmayabilir. FCBF algoritması diğer ilişkilendirme kuralı algoritmalarından farklı olarak entropi kazanımlı wrapper filtrelemesi kullanır. Bu da lineer olmayan verilerde de korelasyon kurup hızlı bir şekilde ilişkilendirme yapar.

Biz de projemizde FCBF algoritmasının IOT veri setlerinin ilişkilendirilmesi için en iyi seçenek olduğunu ispatladık. Bunun için biri normal yoğunlukta diğeri de çok yoğun olan iki veri seti üzerinde çeşitli algoritmaları kıyasladık. İki veri setinde elde ettiğimiz sonuçlara göre FCBF algoritması hem normal veriler için hem de büyük veriler için diğer ilişkilendirme kuralı algoritmalarına göre çok daha yüksek doğruluk oranlarına sahiptir. Aynı zamanda FCBF algoritması diğer ilişkilendirme kuralı algoritmalarından 10 kata kadar daha hızlı çalışmaktadır. Bu da FCBF algoritmasını, IOT verilerinin ilişkilendirilmesi için diğer ilişkilendirme kuralı algoritmalarına göre en iyi seçenek yapmaktadır.

6.KAYNAKLAR

[1] Jiawei, H. , Jian, P. , Yiwen, Y., (2004)"Mining Frequent Patterns without Candidate Generation", Simon Fraser University

[2] Ke, W., Lui, T., Jiawei, H., Junqiang, L., (2002)"Top Down FP-Growth for Association Mining" Ke Wang, Lui Tung Simon Fraser University

[3] L. Yu and H. Liu, "Feature Selection for High-Dimensional Data: A Fast Correlation-Based Filter Solution," Proc. 20th Int'l Conf. Machine Leaning, vol. 20, no. 2, pp. 856-863, 2003.

[4] E. Ozkural, C. Aykanat, and C. Aykanat, "A space optimization for fp-growth." In FIMI, 2004.

[5] Li Haoyuan, Yi Wang, Zhang Dong, Zhang Ming, Chang Edward, (2008)"PFP: Parallel FP Growth for query Recommendation".

[6] Rashmi, S.,Nitin, S., (2012)"AN IMPROVED ASSOCIATION RULE MINING WITH FP TREE USING POSITIVE AND NEGATIVE INTEGRATION", Shri Ram Institute of Technology Jabalpur, India.

[7] Dandu, S., Deekshatulu, B.L., Priti, C., (2013)"Improved Algorithm for Frequent Item sets Mining Based on Apriori and FP-Tree", Global Journal of Computer Science and Technology, 0975-4172

[8] Cornelia. G., Robert G., Stefan H. "A Comparative Study of Association Rules Mining Algorithms". University of Oradea, University of Timisoara.

[9] İnternet : "http://w3.gazi.edu.tr/~suatozdemir/teaching/dm/slides/04.DM.AR.pdf"

[10] İnternet: "https://image.slidesharecdn.com/associations_130924135811_phpapp01/95/associations1-11-638.jpg?cb=1380031212"

[11] İnternet: "http://thardes.de/wp-content/uploads/2013/06/contruction_fp_tree.jpg"

[12] Z. Li, Y. Yang, J. Liu, X. Zhou and H. Lu. "Unsupervised Feature Selection Using Nonnegative Spectral Analysis". AAAI Conference on Artificial Intelligence 2012 (AAAI-2012).

[13] X. He, D. Cai, P. Niyogi, Laplacian score for feature selection, in: Advances in neural information processing systems, 2005, pp. 507-514.

[14] J. Moody, H.H. Yang, “Data Visualization and Feature Selection: New Algorithms for Nongaussian Data”. Advances in NIPS, pp.687-693,2000

[15] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 27, No. 8, pp.1226- 1238, 2005.

[16] D.L. Lewis “Feature selection and feature extraction for text categorization”.Proc. of Speech and Natural Language Workshop, (1992) 212-217

[17] R. Battiti “Using mutual information for selecting features in supervised neural net learning”. IEEE Transactions on Neural Networks, 5 (1994), pp. 537–550

[18] P.E. Meyer, C. Schretter, G. Bontempi, “Information-theoretic feature selection in microarray data using variable complementarity”. In IEEE Journal of Selected Topics in Signal Processing (JSTSP) Special Issue on Genomic and Proteomic Signal Processing, Volume2, Issue 3, 2008

[19] Lin D., Tang X. (2006) Conditional Infomax Learning: An Integrated Framework for Feature Extraction and Fusion. In: Leonardis A., Bischof H., Pinz A. (eds) Computer Vision – ECCV 2006. ECCV 2006. Lecture Notes in Computer Science, vol 3951. Springer, Berlin, Heidelberg

[20] Fleuret, F.: Fast binary feature selection with conditional mutual information. Journal of Machine Learning Research 5, 1531–1555 (2004)

[21] Fabian Pedregosa, Ga¨el Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, et al.. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, Journal of Machine Learning Research, 2011.

[22] Z. Zhao, H. Liu, Spectral feature selection for supervised and unsupervised learning, in: Proceedings of the 24th international conference on Machine learning, ACM, 2007, pp. 1151-1157.

[23] Nie, F., Huang, H., Cai, X., & Ding, C. (2010). Efficient and robust feature selection via joint ℓ2;1-norms minimization. In Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010, NIPS 2010

[24]Kolon Kanseri Veri Setinin Temin Edildiği Site:

http://featureselection.asu.edu/download_file.php?filename=colon.mat&dir=files/datasets/

[25] Lösemi Hastalığı Veri Setinin Temin Edildiği Site:

http://featureselection.asu.edu/download_file.php?filename=leukemia.mat&dir=files/datasets/