Journal of the Faculty of Engineering and Architecture of Gazi University 39:1 (2024) 217-231

Mühendislik Mimarlık Fakültesi Dergisi
Journal of The Faculty of Engineering and Architecture of Gazi University

Elektronik / Online ISSN : 1304 - 4915
Basılı / Printed ISSN : 1300 - 1884

A comparison of tree data structures in the streaming data clustering issue

Ali Şenol[1]*, Mahmut Kaya[2], Yavuz Canbay[3]

[1]Department of Computer Engineering, Engineering Faculty, Tarsus University, 33400, Tarsus, Mersin, Türkiye
[2]Department of Computer Engineering, Engineering Faculty, Siirt University, 56100, Siirt, Türkiye
[3]Department Computer Engineering, Faculty of Engineering and Architecture, Kahramanmaraş Sürçü İmam University, 46100, Kahramanmaraş, Türkiye

Highlights:

- Two new algorithms for streaming data clustering were developed.
- Comparative analyses were performed on benchmark datasets.
- The proposed new algorithms have achieved successful clustering quality in a reasonable runtime.

Keywords:

- Streaming data
- Clustering
- Tree data structures

Article Info:

Research Article
Received: 17.07.2022
Accepted: 25.01.2023

DOI:

10.17341/gazimmfd.1144533

Acknowledgement:

The authors would like to thank Prof. Dr. Hacer Karacan for her support in conducting this study.

Correspondence:

Author: Ali Şenol
e-mail: alisenol@tarsus.edu.tr
phone: +90 324 600 0033

Graphical/Tabular Abstract

Processing streaming data is a challenging issue because of the limitation of time and resources. Clustering data streams is an efficient technique to analyze this kind of data. This study proposes two new streaming data clustering algorithms, BT-AR Stream and VP-AR Stream, inspired by the KD-AR Stream clustering algorithm [32]. Our algorithms used Ball-Tree and Vintage Tree data structures instead of KD-Tree. To reveal the efficiency of the proposed algorithms, we tested the algorithms on 18 benchmark datasets in terms of clustering qualities and runtime complexities. Then we compared obtained results with the results of the KD-AR Stream algorithm. According to the results, the BT-AR Stream algorithm was the most successful in terms of clustering quality and runtime complexity, as illustrated in Figure A.

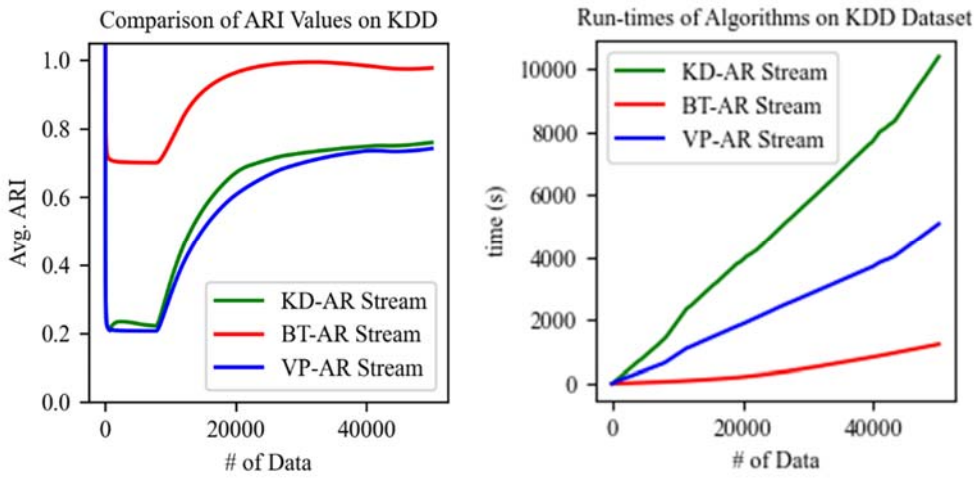


Figure A. Comparison of algorithms on the KDD dataset in terms of clustering quality and runtime complexity

Purpose: This study aims to analyze and compare the efficiency of tree data structure in data stream clustering issues. We aim to reveal the efficiency of tree data structures in both clustering quality and runtime performance.

Theory and Methods: To compare the efficiency of tree data structures in data stream clustering, we proposed two stream clustering algorithms inspired by KD-AR Stream. For this reason, we used Ball-Tree and Vintage-Tree data structures instead of KD-Tree and proposed two new stream clustering algorithms named BT-AR Stream and VP-AR Stream. To compare the success of algorithms, we tested them on 18 benchmark datasets and compared them in aspects of clustering quality and runtime complexity.

Results: According to the results obtained in the experimental study, the BT-AR Stream algorithm, which uses Ball-Tree, was the most successful in both clustering quality and runtime complexity on the KDD, which is a high-dimensional dataset. On the other hand, the clustering quality of all algorithms was good on the other datasets.

Conclusion: Although the clustering quality of all three algorithms was good, the BT-AR Stream algorithm was the most successful because KDD is high-dimensional. Furthermore, it is the fastest algorithm compared to the others.

Journal of the Faculty of Engineering and Architecture of Gazi University 39:1 (2024) 217-231

Mühendislik Mimarlık Fakültesi Dergisi
Journal of The Faculty of Engineering and Architecture of Gazi University

Elektronik / Online ISSN : 1304 - 4915
Basılı / Printed ISSN : 1300 - 1884

Akan veri kümeleme probleminde ağaç veri yapılarının performans karşılaştırması

Ali Şenol[1]*, Mahmut Kaya[2], Yavuz Canbay[3]

[1]Tarsus Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, 33400, Tarsus, Mersin, Türkiye
[2]Siirt Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, 56100, Siirt, Türkiye
[3]Kahramanmaraş Sütçü İmam Üniversitesi, Mühendislik ve Mimarlık Fakültesi, Bilgisayar Mühendisliği Bölümü, 46100, Kahramanmaraş, Türkiye

ÖNEÇIKANLAR

- Akan veri kümeleme için iki yeni algoritma geliştirilmiştir
- Farklı veri kümeleri üzerinde karşılaştırmalı analizler gerçekleştirilmiştir
- Önerilen yeni algoritmalar çalışma zamanı ve kümeleme kalitesinde oldukça başarılı sonuçlar elde etmiştir

Makale Bilgileri
Araştırma Makalesi
Geliş: 17.07.2022
Kabul: 25.01.2023

DOI:
10.17341/gazimmfd.1144533

Anahtar Kelimeler:
Akan veri,
kümeleme,
ağaç veri yapıları

ÖZ

Teknolojik gelişmeler, insanların pek çok farklı kaynaktan üretilen verileri toplamasına ve analiz etmesine imkân tanımaktadır. Sensörler, mobil cihazlar, nesnelerin interneti gibi yapılarda üretilen veriler akan veri formatında olup, bu tür verilerin işlenmesi ve faydalı bilgilerin elde edilmesi zor bir problemdir. Akan verileri analiz etmek için sıklıkla kullanılan yöntemlerden birisi olan akan veri kümelemede, veriler dağılımlarına göre çeşitli gruplara ayrılarak analiz edilir. Ancak bu analizleri gerçekleştiren algoritmalar gerek çalışma zamanı gerekse de kümeleme performansı açısından problemlerle karşılaşmaktadır. Bu çalışmada, akan veri kümelemede kullanılan KD-AR Stream algoritmasına alternatif olarak küre ağacı veri yapısını kullanan BT-AR Stream algoritması ve bakış noktası ağacı veri yapısını kullanan VP-AR Stream algoritmasıı önerilmektedir. Önerilen bu yeni algoritmalar, hem çalışma zamanı hem de kümeleme kalitesi açısından KD-AR Stream algoritması ile karşılaştırılmıştır. Karşılaştırma işlemi farklı özelliklere sahip 18 veri kümesi üzerinde gerçekleştirilerek karşılaştırmaya ait sonuçlar tablolar ve grafiklerle sunulmuştur. Deneysel çalışmalara göre, her üç algoritmanın da genel anlamda başarılı sonuçlar verdiği görülmüştür. Ancak, BT-AR Stream algoritmasının KDD veri kümesinde %89,15 ARI, %96,83 Saflık ve %67,18 Silhouette değerleri ile diğer iki algoritmadan daha başarılı sonuçlar verdiği tespit edilmiştir. Ayrıca BT-AR Stream algoritması çalışma zamanı açısından da diğer iki algoritmadan pozitif yönde ayrışmaktadır.

A comparison of tree data structures in the streaming data clustering issue

HIGHLIGHTS

- Two new algorithms for streaming data clustering were developed
- Comparative analyses were performed on benchmark datasets
- The proposed new algorithms have achieved very successful results in runtime and clustering quality

Article Info
Research Article
Received: 17.07.2022
Accepted: 25.01.2023

DOI:
10.17341/gazimmfd.1144533

Keywords:
Streaming data,
clustering,
tree data structures

ABSTRACT

Advances in technology have allowed people to collect produced data in many different sources and analyze it. The collected data from sensors, mobile devices, and the internet of things are in the form of streaming data, and it is a hard problem to retrieve useful information from such data. In streaming data clustering which is one of the most frequently used methods to analyze streaming data, data is analyzed by dividing it into various groups according to their distribution. However, algorithms that perform these analyzes encounter problems in terms of both runtime and clustering performance. In this study, BT-AR Stream algorithm using ball-tree data structure and VP-AR Stream algorithm using vantage-point tree data structure are suggested as an alternative to KD-AR Stream algorithm used in streaming data clustering. These new proposed algorithms are compared with the KD-AR Stream algorithm in terms of both runtime and clustering quality. The comparison process is performed on 18 data sets with different characteristics and the results of the comparison are presented with tables and graphs. According to experimental studies, it has been seen that all three algorithms give successful results in general. However, BT-AR Stream algorithm achieved better results than the other two algorithms with 89.15% ARI, 96.83% Purity and 67.18% Silhouette values on the KDD dataset. In addition, BT-AR Stream algorithm differs positively from the other two algorithms in terms of runtime.

*Sorumlu Yazar/Yazarlar / Corresponding Author/Authors : *alisenol@tarsus.edu.tr, mahmutkaya@siirt.edu.tr, yavuzcanbay@ksu.edu.tr / Tel: +90 324 600 0033

1. Giriş (Introduction)

Günümüzde teknolojinin hızla gelişmesiyle ve sosyal medya ortamlarının hayatımıza girmesiyle beraber dijital verilerin hacimleri inanılmaz boyutlara ulaşmıştır [1]. Büyük hacimli bu verilerin işlenmesi ile verilerden anlamlı örüntülerin elde edilmesi ve bu örüntülerin yararlı bilgilere dönüştürülmesi mümkündür. Statista firmasının raporlarına göre 2020 yılında 64.2 zetabayt verinin üretildiği, bu miktarın 2021 yılında yaklaşık 79 zetabayta ulaştığı ve 2025 yılında ise bu miktarın 180 zetabaytı aşacağı tahmin edilmektedir [2]. Veri hacmindeki bu inanılmaz artışta video, görüntü ve metin veri türleri oldukça fazla yer edinmektedir [3]. Bu tür veriler genel itibariyle sürekli bir şekilde üretildiği ve aktarıldığı için akan veri formunda olup, analiz edilmesi zordur.

Makine öğrenmesi, veriyi modelleyebilen ve veriden öğrenen yapay zekânın bir alt dalıdır. Makine öğrenmesinin kapsadığı algoritmalar dinamik yapıda olup, mevcut veriye göre modeller oluşturmayı amaçlamaktadır [4]. Kümeleme, makine öğrenmesi kapsamında çalışılan problemlerden birisi olup, veri kümesi içerisindeki verilerin benzerliklerine göre gruplandırılmasını sağlamaktadır [5]. Akan veri kümeleme, bir sisteme sürekli olarak gelen verilerden anlamlı bilgiler çıkarmayı amaçlayan makine öğrenmesinin önemli konularından biridir. Akan veri kümeleme yaklaşımları, dinamik bir yapıda sistemle etkileşim halinde olan veri üzerinde oldukça değerli bilgiler sunabilmektedir [6]. Aykırı veri tespiti [7], elektrik tüketim tahmini [8, 9], istenmeyen veri tespiti [10], metin kümeleme [11], nesnelerin interneti uygulamaları [12], medikal uygulamalar [13] ve finansal uygulamalar [14] akan veri kümelemenin uygulandığı çeşitli alanlar olarak karşımıza çıkmaktadır.

Geleneksel makine öğrenmesi yaklaşımlarından farklı olarak akan veriyi dikkate alan yaklaşımlarda sınırlı hafıza ile gerçek zamanlı çözümler sunulmaktadır. Literatürde, akan veri kümeleme yöntemleri 5 kategori altında incelenmektedir. Bunlar; hiyerarşi-tabanlı (hierarchical-based), parçalama-tabanlı (partitioning-based), ızgara-tabanlı (grid-based), yoğunluk tabanlı (density-based) ve model-tabanlı (model-based) yaklaşımlardır [15-17]. Akan veri kümeleme yaklaşımlarında kümeleme işlemi gerçekleştirilirken çeşitli problemlerle karşılaşılmaktadır. Bu problemler genel itibariyle; verileri tekrar tekrar işleyememe, sınırlı süre, sınırlı bellek, çoklu etiket sorunları, gerçek zamanlı yanıt, evrimsel veriler (evolving data), esnek küme sayısı ve boyutu, sürüklenme ve algılama olarak karşımıza çıkmaktadır [3, 16, 18]. Büyük veri tabanları için geliştirilmiş bir kümeleme yaklaşımı olan BIRCH [19] (Balanced Iterative Reducing and Clustering using Hierarchies) yöntemi örüntülerin yoğun olarak bulunduğu kısımlara odaklanarak tek taramada oldukça iyi bir kümeleme yaklaşımı sunmaktadır. Lang ve Schubert [20], BIRCH'de kullanılan kare değerlerinin toplamının akan verileri kümelemede başarısının zayıf olduğunu belirtmektedirler. Bu sorunun üstesinden gelmek için BIRCH kümeleme özelliğinin geliştirilmiş bir hali olan BETULA yaklaşımını önermişlerdir. BIRCH gibi geleneksel bir yaklaşımdan farklı olarak akan veriler için daha iyi kümeleme yaklaşımı sunmayı hedefleyen CluStream [21] mikro kümeleri kullanmaktadır. Zhou vd. [22], CluStream algoritmasının uzun süre çalışma sonrasında kümeleme sonuçlarında azalmalar olduğunu belirlemiş ve bunun üstesinden gelmek için mikro kümeler yaklaşımını esas alan SWClustering yöntemini akan veri kümeleme için önermişlerdir.

CHAMELEON kümeleme algoritması [23], kümeleri oluştururken benzer küme çiftlerinin hem birbirlerine olan bağlılığını hem de yakınlığını esas almaktadır. CHAMELEON algoritmasını temel alan REPSTREAM algoritması [24], akan verileri kümelemek için grafik tabanlı artırım yapmaktadır. E-Stream [25], görünüm, kaybolma, kendi kendine evrim, birleştirme ve bölünme olarak bilinen beş küme gelişimini destekleyebilen hiyerarşik bir algoritmadır. HUE-Stream [26] heterojen belirsizliğe sahip çeşitli akan veriler için E-Stream'in bir uzantısı olarak sunulan hiyerarşik bir algoritmadır. Nikpour ve Asadi [27], hiyerarşik bir yaklaşımla akan verileri kümelemek için kavram kayması dikkate alınarak akan verinin dinamik kümelenmesi yöntemini önermişlerdir. Sangma vd. ise [28], akan veriler arasındaki farklılığı ölçme ve eşitsizlik derecesini hesaplama bilgilerini dikkate alan bütünleştirici hiyerarşik kümeleme yaklaşımı önermişlerdir. Hyde vd. [29], CEDAS ismini verdikleri algoritma ile çevrimiçi akan veri kümeleme işlemi gerçekleştirilmiştir. Geliştirilen bu algoritma iki aşamadan oluşup, doğru, gürültüye karşı dayanıklı, hesaplama ve hafıza tüketimi açısından etkin bir yaklaşıma sahiptir. Algoritmanın ilk aşamasında mikro kümeler üretilirken, ikinci aşamasında bu kümeleri birleştirerek makro kümeler oluşturmaktadır. Mirsky vd. önerdiği pcStream2 [30] akan veri kümeleme algoritması ise, hesaplama karmaşıklığını azaltmak ve hafıza gereksinimi düşürmek için arttırımsal temel bileşen analizi yönteminden faydalanmaktadır. KDD veri kümesi üzerinde yapılan deneylerde başarılı sonuçlar elde edildiği raporlanmıştır. Al-Amri vd. [31], akan veriyi çevrimiçi kümeleme için TSHC isimli algoritmayı önermişlerdir. Uzaysal hiper küpler yapısını kullanarak geliştirilen algoritma ile mevcut algoritmaların veri özetlemedeki istatistiki kabullerinin kümeleme kalitesini düşürme sorununa çözüm sunmuşlardır. KDD veri kümesinin kullanıldığı çalışmada kabul edilebilir sonuçlar elde edilmiştir.

Rowanda vd. [32], dairesel olmayan kümeleri bulamama ve sapan verileri ele alma gibi bazı sınırlamalardan dolayı akan veri kümelemede mevcut yöntemlerin başarısının düşük olduğunu belirtmişlerdir. Yazarlar bunun üstesinden gelmek için; çevrimiçi-çevrimdışı ızgara ve yoğunluğa dayalı akan veri kümeleme algoritması olan DGStream'i önermişlerdir. ESA-Stream [33], akan verileri gerçek zamanlı işleme ve sabit parametrelerle ilgili problemleri gidermek için önerilen bir akan veri kümeleme algoritmasıdır. Huang vd. [34], birden fazla yetersiz görünümden gelen bilgilerin entegre edilebildiği ve çıktı verilen destek vektörler ile akan veri kümeleme algoritmasının özet istatistiklerinin soyutlamada kullanılabildiği MVStream yöntemini önermişlerdir. Laohakiat ve Saing [35], hem geleneksel hem de akan veri kümelemesinde çalışabilen, tek geçiş şemasını kullanan Bulanık Artan Yoğunluk Tabanlı Kümeleme isimli bir yöntem önermişlerdir.

Hiyerarşi tabanlı yaklaşımlar, veri nesneleri arasında bir örüntü elde etmek için hiyerarşik kümeler ağacında gruplar oluşturmaktadır [36]. Oluşturulan her bir katman örüntü gruplarına ait veri kümesinin bir bölümüne karşılık gelmektedir. Ağaç temelli olan bu yaklaşımlarda örüntüler aşağıdan yukarıya doğru bir araya getirme veya bölme şeklinde gruplar halinde olmaktadır. Bu çalışmada akan veri kümeleme için ağaç tabanlı iki yeni algoritma geliştirilerek uygulanmıştır. Geliştirilen bu algoritmalar daha önce Şenol ve Karacan tarafından önerilen KD-AR Stream [37] algoritmasından esinlenmektedir. Yapılan deneysel çalışmalarda, akan veri kümeleme için farklı sayıda kayıt ve boyutlara sahip 18 açık veri kümesi kullanılmıştır. Geliştirilen algoritmalar olan Bakış Noktası Ağacı (Vantage-Point Tree) tabanlı VP-AR Stream ve Küre Ağacı (Ball-Tree) tabanlı BT-AR Stream algoritmalarının kümeleme başarısı ve çalışma zamanı performansları KD-AR Stream algoritması ile karşılaştırılmıştır.

2. Çalışmada Kullanılan Ağaç Veri Yapıları (Tree Data Structures Used in the Study)

Bu makalede, ağaç veri yapılarının akan veri kümeleme problemindeki etkinliklerini araştırmak üzere bir çalışma

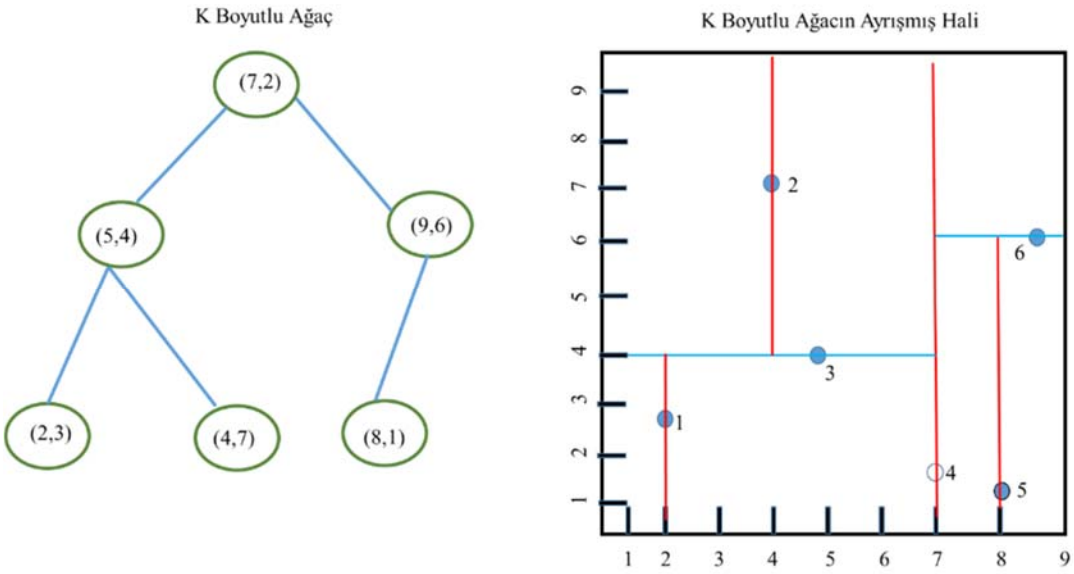
gerçekleştirilmiştir. Bu amaçla, K-Boyutlu Ağaç, Küre Ağacı ve Bakış Noktası Ağaçları kullanılarak performansları çeşitli metrikler dikkat alınarak karşılaştırılmıştır. Bu bölümde, kullanılan ağaç veri yapılarının özellikleri ile ilgili temel bilgiler verilmektedir.

2.1. K-Boyutlu Ağaç (Kd Tree)

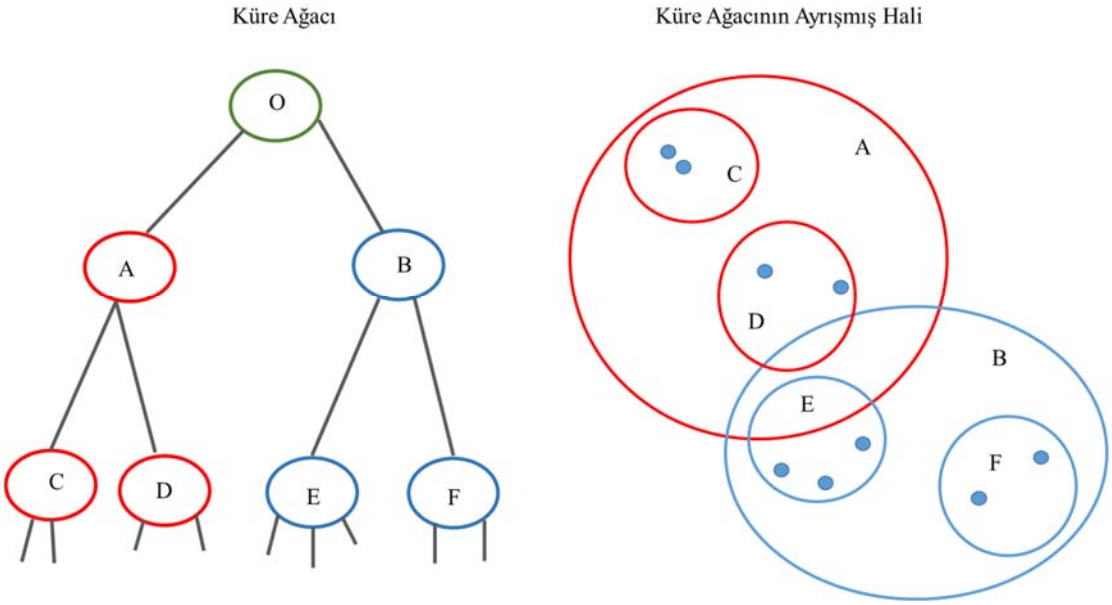
K-Boyutlu Ağaç yapısı 1975 yılında Jon Louis Bentley tarafından geliştirilmiş olup çok boyutluluğu destekleyen bir yaklaşımdır [38]. Bir uzayda ikili düğüm yapısına sahip düğümlerden oluşan çok boyutlu ağaç yapısıdır. Ağaç yapısında her düzeyde bir boyutta karşılaştırma gerçekleştirilmektedir. Her bir ebeveyn düğümün sadece iki tane çocuk düğümü bulunmaktadır. Bu ağaç yapısı k boyutlu bir uzayda en yakın komşuları bulmak için kullanılmaktadır. Şekil 1'de K-Boyutlu Ağacın iki boyutlu düzlemde ayrışmış hali ile ikili düğüm yapısına sahip ağaç yapısı görülmektedir. Şekil 1'de görüldüğü üzere veriler ağaca yerleştirilirken her adımda bir niteliğe göre bölünmektedir.

2.2. Küre Ağacı (Ball Tree)

Küre Ağaç yapısı çok boyutlu bir uzayda geometrik öğrenme görevlerine yönelik uzaysal bölümlemeye dayanan bir geometrik veri yapısı türüdür [39]. Şekil 2'de küre ağacının ayrışmış hali ile küre yapısına sahip ağaç yapısı görülmektedir. Küre ağacı adını, veri noktalarını Şekil 2'de görüldüğü gibi "küre" olarak bilinen iç içe kesişen hiper küreler kümesine bölmesinden alır. Küre Ağaç yapısı, K-Boyutlu ağaç yapısında olduğu gibi çok boyutluluğu desteklemektedir. Bu özelliğinden dolayı bu çalışmada uygulanan BT-AR Stream algoritmasında kullanılmıştır.



Şekil 1. K-Boyutlu ağaç veri yapısı ve ayrışımı (KD-Tree and its deconstruction)



Şekil 2. Küre ağacı veri yapısı ve ayrışımı (Ball-Tree and its deconstruction)

2.3. Bakış Noktası Ağacı (Vantage-Point Tree)

Bakış Noktası Ağaç yapısı, uzaysal düzlemde bir noktayı seçerek veri uzayını belirlenmiş noktaya uzak olan ve yakın olan noktalar olarak iki parçaya ayırmaya dayanmaktadır [40]. Şekil 3'te bakış noktası ağacının ayrışmış hali ile bakış noktası ağaç yapısı görülmektedir. Şekil 3'te görüldüğü gibi özyinelemeli olarak veri uzayı daha küçük parçalara ayrılmakta ve böylelikle komşuluk ilişkisine dayanan bir ağaç yapısı oluşmaktadır.

3. Akan Veri Kümeleme Probleminde Ağaç Veri Yapılarının Uygulanması
(Application of Tree Data Structures on Streaming Data Clustering Problem)

Bu çalışmada akan veri kümeleme probleminde ağaç veri yapılarının etkinliklerinin değerlendirilmesi amaçlanmıştır. Bu kapsamda, daha önce Şenol ve Karacan [32] tarafından önerilmiş olan KD-AR Stream algoritması referans alınarak iki yeni algoritma önerilmiştir. KD-AR Stream algoritmasından farklı olarak k-boyutlu ağaç (Kd-Tree) yapısı yerine farklı ağaç yapılarının kullanılmasının hem kümeleme başarısına, hem de çalışma zamanı performanslarına etkisi karşılaştırmalı olarak analiz edilmiştir. Bu kapsamda, k-boyutlu ağaç veri yapısı kullanılarak geliştirilen KD-AR Stream algoritmasının genel aşamalarından faydalanılarak Bakış Noktası Ağacı (Vantage-Point Tree) tabanlı VP-AR Stream ve Küre Ağacı (Ball-Tree) tabanlı BT-AR Stream algoritmaları geliştirilmiştir.

VP-AR Stream ve BT-AR Stream algoritmalarının geliştirilmesinde KD-AR Stream algoritmasındaki çeşitli alt fonksiyonlardan faydalanılmasına rağmen ilgili ağaç yapılarına ait ağaç oluşturma ve oluşturulan ağaçlar üzerinde alan arama algoritmaları için yeni alt-algoritmalar geliştirilmiştir. KD-AR Stream algoritmasında; küme tanımlamak amacıyla kullanılan YeniKümeOluştur fonksiyonunda ve var olan bir kümeyi bölmek amacıyla kullanılan KümeBöl fonksiyonunda k-boyutlu ağaç yapısından faydalanılır. Bu çalışmada k-boyutlu ağaç veri yapısı yerine; BT-AR Stream algoritmasında küre ağacı, VP-AR Stream algoritmasında ise bakış noktası ağacı veri yapısı kullanılmıştır. Alan arama işlemi için ise BT-AR Stream algoritmasında KA_AlanArama ve VP-AR Stream algoritmasında ise BNA_AlanArama fonksiyonları geliştirilmiştir. Şenol ve Karacan [32] tarafından önerilen akan veri kümeleme için genel bir iş akış diyagramı Şekil 4'te verilmekte olup, aynı diyagram bazı aşamalarda güncellenerek bu çalışma kapsamında da kullanılmıştır. Şekil 4'te sunulan aşamalar alt başlıklar halinde aşağıda açıklanmış ve bu çalışma kapsamında yapılan güncellemeler sunulmuştur.

3.1. Yaşlandırma (Aging)

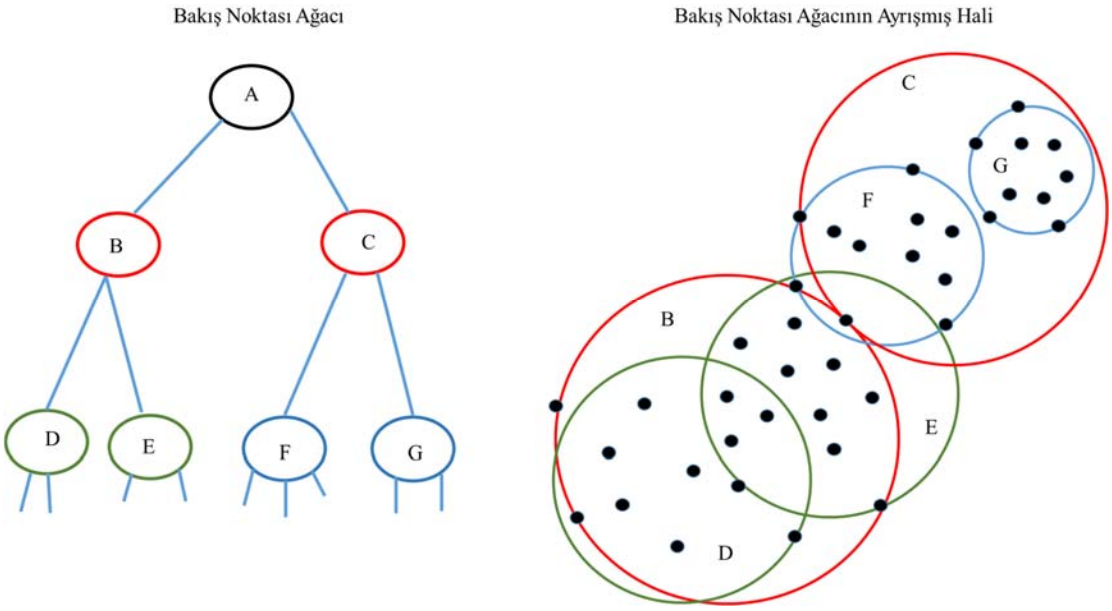
Akan veri kümelemede zamana bağlı bir özetleme yaklaşımı kullanılmaktadır. Bu yaklaşımda her bir verinin işlenmesi için belirli bir süre bulunmaktadır. Bu süre sonunda işleme konulan veriler yaşlandırılarak işlenen verilerden silinmektedir. Bu işlem sonrasında kümeler üzerinde gerekli güncellemeler yapılmaktadır.

3.2. Yeni Küme Oluşturma (Creating New Clusters)

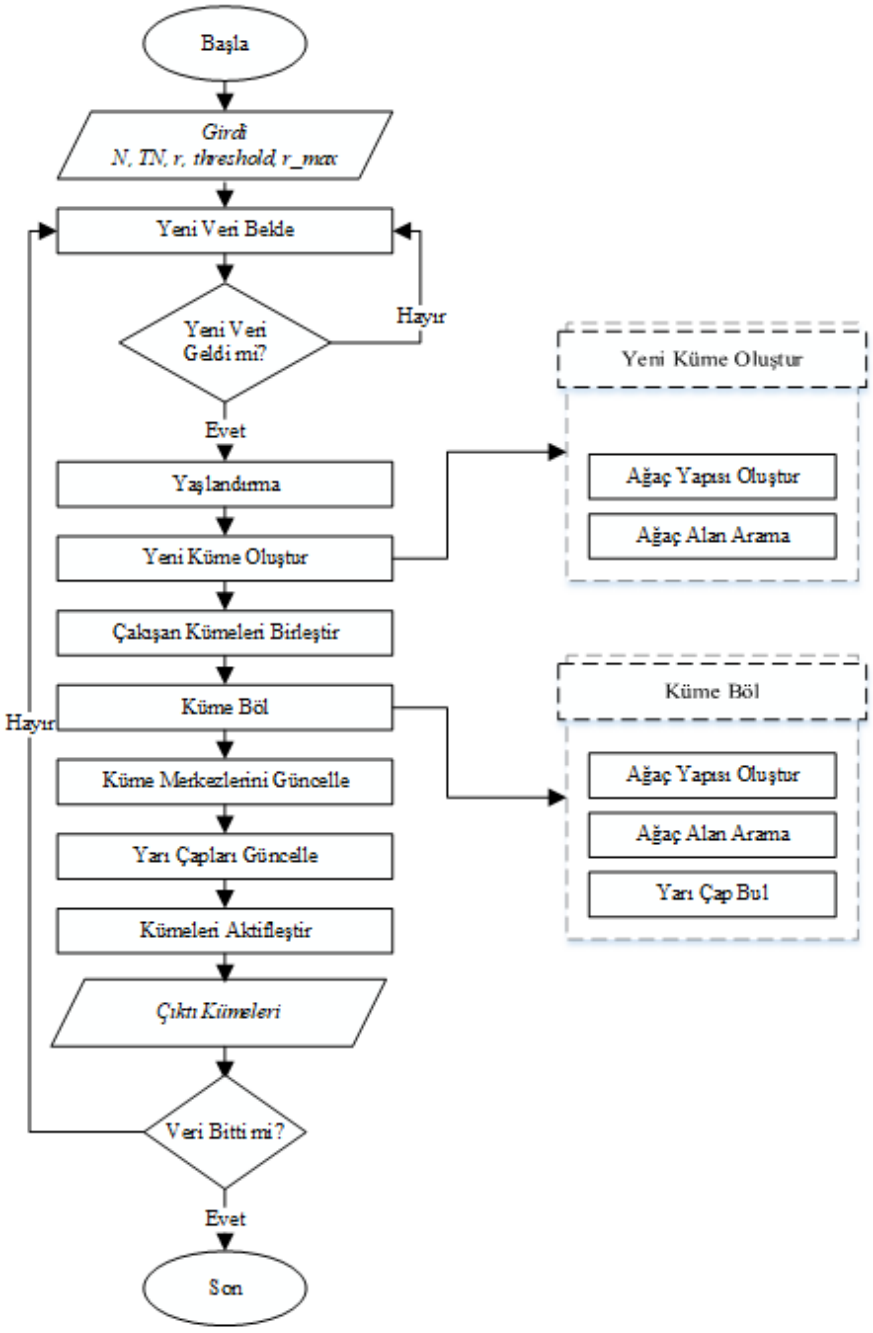
Bakış Noktası Ağacı ve Küre Ağacı temelli akan veri kümeleme yaklaşımlarında veriler bir yarıçap etrafında kümelenir. Burada yapılan işlem herhangi bir kümeye ait olmayan verilerin kullanılan ağaç yapısına bağlı olarak AgacOlustur ve AlanArama yaklaşımlarıyla bir küme içerisine yerleştirilmesi sürecinden oluşmaktadır. Bu işlemler gerçekleştirilirken her bir ağaç yapısı için belirlenmiş bir yarı çarp içerisinde olan veriler için bir küme oluşturulmasıdır.

Şekil 5'te verilen Algoritma 1'de işlemler gerçekleştirilirken her bir ağaç yapısında belirlenmiş bir yarıçap içerisinde olan veriler için bir küme oluşturulmaktadır. Oluşturulacak olan küme yapısında bir ağırlık merkezi belirlenerek küme oluşumları gerçekleştirilmektedir. Küme oluşumunda kullanılacak olan ağırlık merkezine ait denklem yapısı Eş.1'de verilmektedir. Denklem yapısında X kümeye eklenecek olan veriyi temsil ederken; X_i^j ifadesinde i'inci veriye ait j'inci niteliği temsil edilmektedir. Kümede veri sayısı N ile ifade edilirken; μ^j oluşturulacak olan kümeye ait j'inci niteliğe ait ağırlık merkezine karşılık gelmektedir.

$$\mu^j = \frac{1}{N}\sum_{i=1}^{N} X_i^j \quad (1)$$



Şekil 3. Bakış noktası ağacı veri yapısı ve ayrışımı (VP-Tree and its deconstruction)



Şekil 4. Kullanılan ağaç veri yapıları ile akan veri kümeleme yaklaşımı (Stream data clustering approach with used tree data structures)

3.3. Ağaç Yapısı Oluşturma (Creating Tree Structure)

Bu çalışmada kullanılan her bir ağaç türünün farklı veri yapıları bulunmaktadır. Algoritma 1'de kullanılan AgacOlustur algoritmasına karşılık olarak, K-Boyutlu Ağaç (Kd Tree), Küre Ağacı (Ball Tree) ve Bakış Noktası Ağacı (Vantage-Point Tree) veri yapılarının ağaç oluşturma algoritmaları sırası ile Şekil 6, Şekil 7 ve Şekil 8'de verilmiştir. AgacOlustur yöntemlerinin tamamında özyinelemeli bir ağaç oluşturma yaklaşımı bulunmaktadır.

3.4. Ağaçta Alan Arama İşlemi (Range Search Operation in the Tree)

Ağaçlarda arama işlemi, ilgili ağaca yerleştirilmiş olan veri kümesi üzerinde arama işlemini ifade etmektedir. Literatürde ağaç üzerinde arama işlemi iki şekilde yapılmaktadır. Birincisi en yakın komşu yaklaşımı üzerinden arama işlemidir. Bu tür yaklaşımlarda en yakın komşu algoritması üzerinden arama işlemi gerçekleştirilmektedir. Diğeri ise, bizim de algoritmalarımızda kullandığımız, belirli bir yarıçapta bulunan verileri arama işlemi olan alan arama (range search)

işlemidir. Bu işlemde ağaca yerleştirilmiş olan veriler üzerinden belirlenmiş olan *r* yarıçapta verileri bulan işlem gerçekleştirilmektedir. Alan arama işleminde arama işlemi ağacın türüne göre farklılık göstermektedir. K-Boyutlu Ağaç (Kd-Tree), Küre Ağacı (Ball-Tree) ve Bakış Noktası Ağacı (Vantage-Point Tree) veri yapılarının alan arama algoritmaları sırası ile Şekil 9, Şekil 10 ve Şekil 11'de verilmiştir.

Algoritma 1: YeniKümeOlustur

```
Input: C, işlenenVeri, r, N;
Output: C;
işlenenVeri;
Herhangi bir kümeye ait olmayan Veri'yi al;
ağaç ← AgaçOluştur(Veri);
if Veri'nin boyutu > N then
    foreach Veri_i ∈ Veri do
        C_temp ← AlanArama(ağaç, r);
        if C_temp'in boyutu > N then
            C_temp'i yeni bir küme olarak tanımla;
        end
    end
end
return C;
```

Şekil 5. Algoritma 1: yeni küme oluştur
(Algorithm 1: Create a new cluster)

3.5. Çakışan Kümeleri Bulma (Finding Overlapping Clusters)

Gelen verileri kümelere ekleme işlemi yapılırken bazı durumlarda kümeler arasında geçirgenlik durumu olabilir. Bu gibi durumlarda oluşturulan kümelerin yarıçapları arasındaki mesafeye bakılarak kümelerin birleşiminin yapılması daha doğru bir seçenek olur. Cao vd. tarafından yapılan çalışmada sunulan yaklaşıma göre [41], bir kümenin kabuk yarıçapı diğer bir kümenin çekirdek yarıçapı ile örtüşüyorsa bu kümeler arasında birleştirme işlemi gerçekleştirilmektedir. Şekil 12'de küme kesişimleri sonrasında oluşan küme birleşimi görülmektedir (r1:küçük kümeye ait çekirdek yarıçapı, r2: büyük kümeye ait kabuk yarıçapı, R: küme birleşimleri sonrası oluşan yeni kabuk yarıçapı).

Algoritma 3: KAgaciOlustur

```
Input: Veri;
Output: ağaç;
if Veri sayısı = 1 then
    return Veri;
end
else
    c en büyük yayılımın boyutu,
    p, c'ye bağlı olarak seçilen merkez noktası,
    L ve R de c medyana göre sağ ve sol çocuklar olsun.
    agac.pivot ← p;
    agac.cocuk1 ← KAgaciOlustur(L);
    agac.cocuk2 ← KAgaciOlustur(R);
    return agac;
end
```

Şekil 7. Algoritma 3: Küre ağacı oluştur (Algorithm 3: Create ball tree)

Algoritma 4: BNAgaciOlustur

```
Input: Veri;
Output: ağaç;
if Veri sayısı = 1 then
    return Veri;
end
else
    dugum ← yeni bir düğüme işaret etsin;
    dugum.p ← Veri'nin rastgele seçilmiş elemanı;
    dugum.μ ← medyan d(p, v), ∀ v ∈ Veri;
    L ← {v ∈ Veri{p}|d(p, v) < μ}
    R ← {v ∈ Veri {p}|d(p, v) ≧ μ}
    dugum.sol ← BNAgaciOlustur(L)
    dugum.sag ← BNAgaciOlustur(R)
    return agac;
end
```

Şekil 8. Algoritma 4: Bakış noktası ağacı oluştur
(Algorithm 4: Create vantage-point tree)

Algoritma 2: KBAgaciOlustur

```
Input: Veri;
Output: ağaç;
if Veri sayısı = 1 then
    return Veri;
end
else if derinlik çift ise then
    Veriyi X'in medyanına göre düşey şekilde Veri1 ve Veri2 olarak iki alt kümeye böl
end
else
    Veriyi Y'nin medyanına göre yatay şekilde Veri1 ve Veri2 olarak iki alt kümeye böl
end
v düğümünü oluştur;
v_left ← KBAgaciOlustur(Veri1,derinlik+1);
v_right ← KBAgaciOlustur(Veri2,derinlik+1);
v_left'i v'nin sol alt çocuğu, v_right'ı ise sağ alt çocuğu olarak ata;
return v;
```

Şekil 6. Algoritma 2: K boyutlu ağaç oluştur (Algorithm 2: Create kd tree)

```
Algoritma 5: KBA_AlanArama
Input: v, r;
Output: r yarıçap içerisinde bulunan veriler;
if v bir yapraksa then
    v'yi r içerisinde olarak belirt;
end
else
    Solaltagacı değerlendir;
    if Solaltagac r içerisinde ise then
        Solalatagac r içerisinde olarak belirt;
    end
    else if Solaltagac r ile kesişiyor ise then
        KBA_AlanArama(Solaltagac, r);
    end
    Sagaltagacı değerlendir;
    if Sagaltagac r içerisinde ise then
        Sagalatagac r içerisinde olarak belirt;
    end
    else if Sagaltagac r ile kesişiyor ise then
        KBA_AlanArama(Sagaltagac, r);
    end
end
return r yarıçap içerisinde bulunan veriler;
```

Şekil 9. Algoritma 5: K boyutlu ağaç alan arama
(Algorithm 5: Kd tree range search)

```
Algoritma 6: KA_AlanArama
Input: v, k;
Output: r yarıçapta bulunan veriler;
if v bir yapraksa then
    v'yi r içerisinde olarak belirt;
end
else
    if Eğer v ile kesişme sözkonusu ise then
        KA_AlanArama(Solaltagac, r);
        KA_AlanArama(Sagaltagac, r);
    end
end
return r yarıçap içerisinde bulunan veriler;
```

Şekil 10. Algoritma 6: Küre ağacı alan arama
(Algorithm 6: Ball tree range search)

3.6. Küme Bölme (Splitting a Cluster)

Ağaç yapıları içerisinde akan veriyi kümelemek için çeşitli yarıçapta kümeler oluşturulur. Oluşturulan bu kümelerin boyutunun çok fazla olduğu durumlarda gereksiz işlem yükleri ve algoritma üzerinde performans kayıpları söz konusu olabilmektedir. Bu tür durumların üstesinden gelebilmek için belirli şartlar altında küme bölme işlemleri yapılmaktadır. Küme oluşturulurken üç farklı koşulun gerçekleşmesi gerekmektedir [37]:

- Oluşturulan kümedeki veri sayısının minimum küme oluşturmak için gerekli veri sayısının iki katı olması,
- Kabuk yarıçapının iki katı olması,
- Çekirdek yarıçapının en az *r* kadar olması

Küme bölme işlemi gerçekleştirilirken gereken şartlar sağlandıktan sonra öncelikle yukarıda verilen algoritmalardan ağaç türüne göre uygun olanı seçilerek bir ağaç yapısı oluşturulur. Daha sonra bölünecek olan kümedeki veriler için uygun alan araması gerçekleştirilerek uygun aday kümeler tespit edilir ve oluşturulacak kümeler için yarıçaplar güncellenerek yeni küme oluşumları gerçekleştirilir. Küme bölmeye ait sözde kod Şekil 13'te verilmektedir.

```
Algoritma 7: BNA_AlanArama
Input: node;
Output: r yarıçapta bulunan veriler;
if node = Ø then
    return;
end
x ←d(q,node);
if x < τ then
    τ ← d(q, node);
    best ← node;
end
orta ← (node.sol[enyuksek]+node.sag[endusuk])/2;
if x < orta then
    if x ∈ I_L then
        BNA_AlanArama(node.sol);
    end
    if x ∈ I_R then
        BNA AlanArama(node.sag);
    end
end
else
    if x ∈ I_R then
        BNA_AlanArama(node.sag);
    end
    if x I_L then
        BNA_AlanArama(node.sol);
    end
end
return r yarıçapta bulunan veriler;
```

Şekil 11. Algoritma 7: Bakış noktası ağacı alan arama
(Algorithm 7: Vantage-point tree range search)

3.7. En Yakın Kümeyi Bulma (Finding Nearest Cluster)

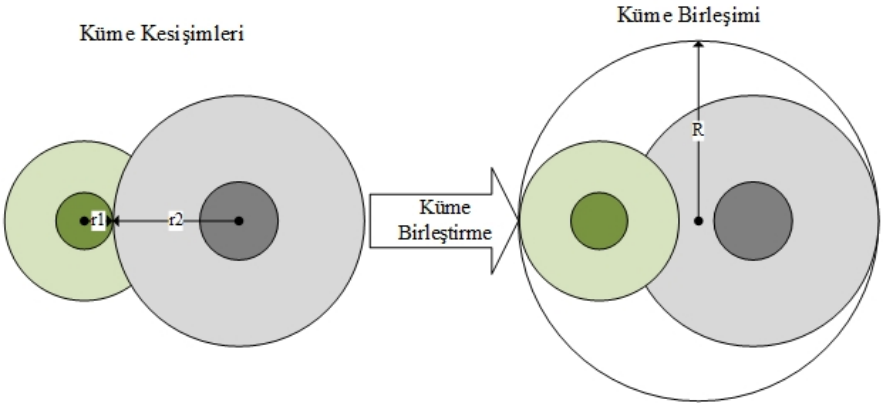
Akan veri kümelemede kümeler oluşturulurken yeni gelen verilerin bir kümeye eklenmesi durumu oluşmaktadır. Bu amaçla yeni gelen verinin mevcut kümelere olan uzaklığına bağlı olarak en yakın kümeye eklenme işlemi gerçekleştirilmektedir. Yeni gelen verilerin haricinde oluşturulan yeni kümeler içerisinde bazı veriler boşta kalabilmektedir. Bu durumda da boşta bulunan veriler mevcut küme uzaklığına bakılarak en yakın kümeye dâhil edilmektedir.

3.8. Küme Merkezlerini Güncelleme (Updating Centers of Clusters)

Her yeni gelen veri veya kümelerin değişim durumuna bağlı olarak küme merkezleri değişebilmektedir. Kümelere yeni eklenen verilere aktif verilere bakılarak aktif küme merkezleri güncellenmektedir. Kümenin pasif duruma geldiği durumlarda ise kümeye ait olan aktif veriler ve silinmiş verilerle güncellenmektedir.

3.9. Yarı Çapları Güncelleme (Updating Radius)

Zaman içinde oluşturulan kümelerde gerçekleşen değişimlerle beraber kümelere ait yarıçaplar güncellenmektedir. Kümelerin



Şekil 12. Küme birleştirme (Merging of two clusters)

Algoritma 8: KümeBöl

Input: C, işlenenVeri, r, N, d;
Output: C;
işlenenVeri;
foreach aktif kümeler $C_i \in C$ *do*
  if C_i'nin veri sayısı>2*N ve $C_{i(KYarıçap)}$>=2*r and $C_{i(CYaricap)}$>=r *then*
    ağaç ← AgaçOlustur(C_i'nin verileri);
    foreach C_i verisi *do*
      C_{temp} ← AlanArama(ağaç, r);
      if C_{temp}'in veri sayısı >= N *then*
        mesafe ← MesafeBul(C_{temp}, C_i-C_{temp}, d);
        r_{temp} ← yarıçapBul(C_{temp});
        $r_{C\,i}$ ← yarıçapBul(C_i-C_{temp});
        if dist > r_{temp}+r_{Ci} *then*
          C_{temp}'i yeni bir küme olarak tanımla;
          C_i ← (C_i-C_{temp});
          break;
        end
      end
    end
  end
end
return C;

Şekil 13. Algoritma 8: Küme böl (Algorithm 8: Divide cluster)

birleşmesi ve bölünmesi durumları ile yeni gelen verilere bağlı olarak kümelerin büyümesi söz konusu olabilmektedir. Bu tür durumlarda küme merkezleri ve buna bağlı olarak her bir kümenin çekirdek yarıçapı ve kabuk yarıçapları tekrar hesaplanarak güncellenmektedir.

3.10. Kümelerin Aktif-Pasif Dönüşümü (Active-Passive Transformation of Clusters)

Kümelerin aktif olma durumu, küme oluşturmak için gerekli minimum veri sayısına bağlıdır. Eğer bir küme içerisinde minimum sayıda veriye ulaşılmışsa küme aktifleştirme işlemi gerçekleştirilmektedir. Aksi durumda küme pasif hale getirilmektedir.

3.11 Çalışma Zamanı Karmaşıklığı (Run-Time Complexity)

Her ağaç yapısının çalışma yaklaşımı kendisine avantaj ve dezavantajlar sağlamaktadır. Bu çalışmada kullanılan ağaç veri yapılarının zaman karmaşıklıkları birbirlerine paralellik göstermesine rağmen pratikte farklı sonuçlar elde edilebilmektedir. Özellikle boyut arttıkça küre ağacı veri yapısı daha avantaj teşkil etmektedir. *n* işlenen veri sayısı, *d* her verinin sahip olduğu nitelik sayısı ve *k* ise geri döndürülen veri sayısı olmak üzere önerdiğimiz algoritmaların kullandığı ağaç oluşturma ve alan arama algoritmalarına ait çalışma zamanları Tablo 1'de verilmiştir.

Tablo 1. Algoritmaların zaman karmaşıklıklarının hem ağaç oluşturma hem de alan arama açısından karşılaştırılması
(The comparison of time complexities of the algorithms in the aspects of both constructing the tree and range search)

	Ağaç Oluşturma	Alan Arama
KD-AR Stream	$O(dn\,logn)$	$O(dn^{1-\frac{1}{d}}+k)$
BT-AR Stream	$O(dn\,logn)$	$O(d\,logn)$
VP-AR Stream	$O(dn\,logn)$	$O(d\,logn)$

4. Deneysel Çalışma (Experimental Studies)

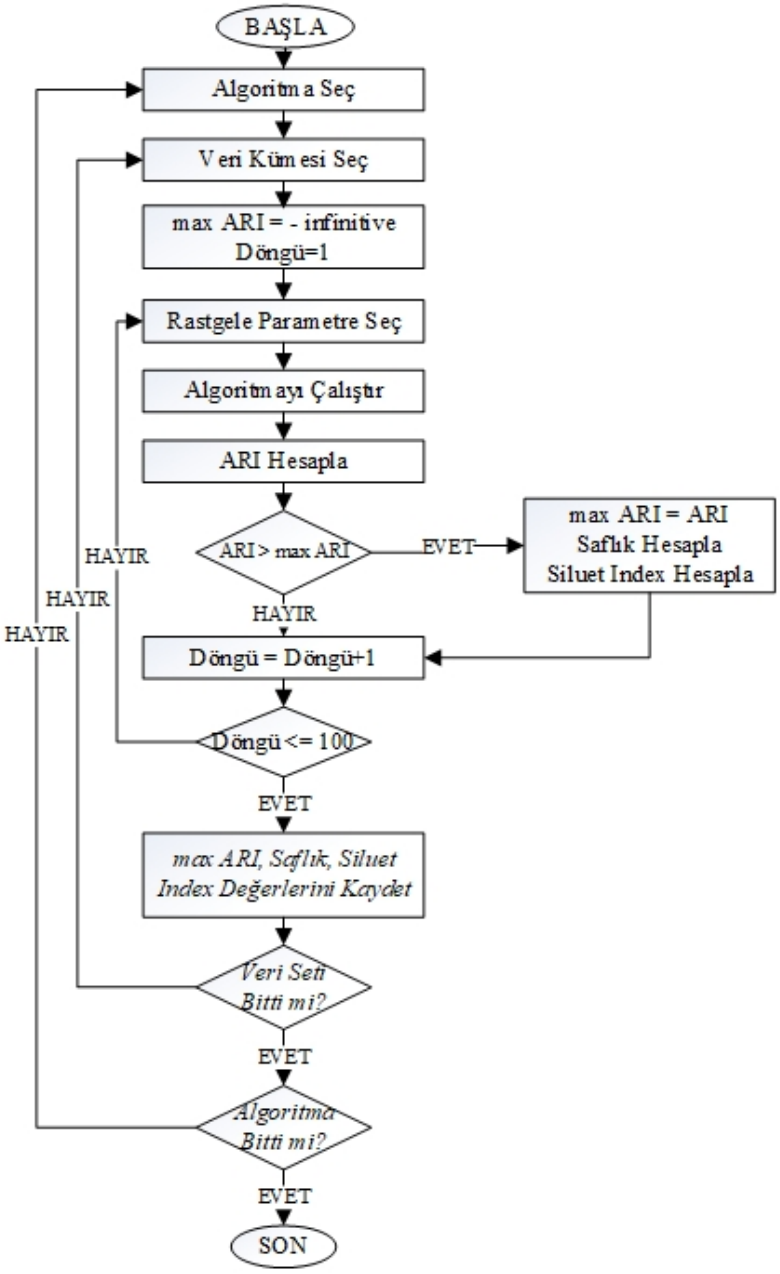
4.1. Deneysel Tasarım (Experimental Design)

4.1.1. Deneysel kurgu ve ortamı (Experimental setup and environment)

Bu çalışmada kullanılan ağaç yapıları ve ilgili diğer algoritmalar, Python dili ile Anaconda Spyder ortamında ScikitLearn kütüphanesi gibi çeşitli kütüphaneler kullanılarak geliştirilmiştir. Veri kümeleri Pandas ve Numpy kütüphaneleri ile içe aktarılmış, matematiksel işlemler ise yine Numpy kütüphanesi ile gerçekleştirilmiştir. Tüm deneysel çalışmalar 16 GB RAM, Intel i7 işlemcili ve Windows 11 işletim sistemi yüklü bir bilgisayarda gerçekleştirilmiştir.

Geliştirilen BT-AR Stream ve VP-AR Stream algoritmalarını KD-AR Stream algoritması ile karşılaştırmak için Şekil 14'teki prosedür kullanılmıştır. Adjusted Rand Index (ARI) metriği değerlendirme yöntemi olarak kullanılmış ve en yüksek kümeleme başarısını veren parametreleri tespit etmek amacıyla da rastgele arama (random search) yöntemi kullanılmıştır.

KD-AR Stream algoritması gibi BT-AR Stream ve VP-AR Stream algoritmaları da 5 ön tanımlı değişken (parametre) kullanmaktadır. Bunlar: bir kümenin sahip olması gereken minimum veri sayısı (*N*), kayan pencere genişliği (tampona alınacak veri sayısı - *W*), küme tanımlama yarıçapı (*r*), yarıçap arttırme/azaltma eşik değeri (*r_threshold*) ve yarıçapın ulaşabileceği maksimum yarıçap (*r_max*)



Şekil 14. Algoritma karşılaştırma prosedürü (Used procedure for comparison of the algorithms)

değeridir. Daha hızlı sonuç almak ve karşılaştırma işleminin aynı değerler üzerinden yapılmasına imkân tanımak amacıyla algoritmalarda zaman penceresi yerine kayan pencere genişliği (sliding window) kullanılmıştır. Ayrıca verilerin aynı aralıkta olmasını sağlamak ve parametre belirleme işlemini kolaylaştırmak amacıyla Eş. 2'de verildiği gibi verilere MinMax Normalizasyonu uygulanmıştır.

$$z_{ij} = \frac{x_{ij} - min x_j}{\max x_j - \min x_j} \quad (2)$$

4.1.2. Kullanılan Veri Kümeleri (Used Datasets)

Algoritmaların kümeleme başarılarını karşılaştırmak amacıyla iki kaynaktan (UC Irvine Machine Learning Repository [42] ve Tomas Barton's repository) alınmış olan 6'sı gerçek olmak üzere 18 veri kümesi kullanılmıştır. Algoritmaların büyük veri kümeleri üzerindeki başarılarını test etmek amacıyla KDD veri kümesi kullanılmıştır. Bununla beraber algoritmaların sapan verilere karşı dirençlerini ölçmek amacıyla %10'u sapan veri olan MrData veri kümesi kullanılmıştır. Deneysel çalışmada kullanılan veri kümelerine ait özellikler Tablo 2'de verildiği gibidir.

4.1.3. Kümeleme değerlendirme metrikleri (Clustering evaluation metrics)

Bu çalışmada kümeleme algoritmalarının başarılarını ölçmek için ARI, Saflık ve Silhouette metrikleri kullanılmıştır. Rand Index (RI) iki veri kümesi arasındaki benzerliği ölçmeye çalışan istatistiksel bir yöntemdir. Bu yöntem aynı zamanda sınıf etiketleri üzerinden başarı tayini yapmak için de kullanılmaktadır. RI bir kümedeki verinin değerini hesaplarken o veriyi aynı kümede olan diğer verilerle ikili olarak karşılaştırır. Bu işlemi aynı şekilde gerçek sınıf etiketleri için de yapar. Sonrasında bu yapılan işlemi ikili olarak karşılaştırarak aynı mı yoksa farklı mı olduğunu bir tabloda tutar. Bu tablo üzerinden RI değerini hesaplar. $X_{Gerçek}$, X parçasına ait gerçek sınıf etiketleri ve X_{Tahmin} X parçasına ait tahmin, $Uyumlu_{Aynı}$ ikili karşılaştırmanın hem $X_{Gerçek}$'de hem X_{Tahmin}'de aynı olmasını, $Uyumlu_{Farklı}$ ikili karşılaştırmanın birinde farklı olmasını, $Uyumsuz_{Aynı}$ ikili karşılaştırmanın hem $X_{Gerçek}$'de hem X_{Tahmin}'de aynı olmasını, $Uyumsuz_{Farklı}$ ikili karşılaştırmanın birinde farklı olmasını ifade etmek üzere RI, Eş. 3 ile hesaplanır.

$$RI = \frac{Uyumlu_{Aynı} + Uyumlu_{Farklı}}{Uyumlu_{Aynı} + Uyumlu_{Farklı} + Uyumsuz_{Aynı} + Uyumsuz_{Farklı}} \quad (3)$$

ARI ise RI değeri üzerinden hesaplanan bir kümeleme başarısı değerlendirme yöntemidir ve Eş. 4 ile hesaplanır.

$$ARI = \frac{RI - Beklenen\ RI}{Max\ RI - Beklenen\ RI} \quad (4)$$

Purity kümelerin saflık derecelerini ölçmeye çalışan bir test türüdür. K küme sayısı ve $|C_i^d|$, $|C_i|$ kümesindeki baskın küme etiketi olmak üzere ortalama Purity Eş. 5 ile hesaplanır.

$$Purity = \frac{\sum_{i=1}^{K} \frac{|C_i^d|}{|C_i|}}{K} x100 \quad (5)$$

Silhouette metriği, kümeleme başarısını ilgili verinin bulunduğu küme ile bulunmadığı kendisine en yakın kümedeki verilere olan mesafe üzerinden hesaplar. Bu metrik küme verilerinin kendi içerisinde ne kadar kompakt ve diğer kümelere de ne kadar uzak olduğuna göre başarı tayini yapar. Elde edilen sonuç -1 ile 1 aralığında olur. Metrik değerinin -1 olması kümeleme başarısının minimum olduğunu ve +1 olması ise kümeleme başarısının maksimum olduğunu gösterir. i verisinin ait olduğu kümenin diğer elemanlarına olan uzaklıklarının ortalaması a ve i verisinin ait olmadığı en yakın kümenin elemanlarına olan uzaklıkların ortalaması b olmak üzere S_i, ortalama Silhouette metriği Eş. 6 ile hesaplanır.

$$S_i = \frac{b - a}{\max(a,b)} \quad (6)$$

4.2. Kümeleme Başarısının Ölçülmesi (Measuring Quality of Clustering)

KD-AR Stream, BT-AR Stream ve VP-AR Stream algoritmalarının kümeleme başarılarını karşılaştırmak için Şekil 14'te verilmiş olan prosedür kullanılmıştır. Her bir algoritma için en yüksek kümeleme başarısını (en yüksek ARI) veren parametreler tespit edilmiş ve ilgili parametreler ile elde edilen ARI, Saflık ve Silhouette metriği değerleri karşılaştırılarak hangisinin kümeleme başarısı açısından daha iyi olduğu değerlendirilmiştir. Her bir algoritmanın her bir veri seti için elde ettiği en iyi parametreler ve bu parametrelere karşılık elde edilen kümeleme başarıları Tablo 3, Tablo 4, Tablo 5 ve Tablo 6'da verilmiştir. Elde edilen sonuçlara göre KD-AR Stream algoritmasının kümeleme başarısı açısından daha iyi olduğu söylenebilir. KD-AR

Tablo 2. Deneysel çalışmada kullanılan veri kümelerinin özellikleri (Features of the datasets used in the experimental study)

Veri Kümesi	Türü	Nitelik Sayısı	Örnek Sayısı	Sınıf Sayısı
IdealData	*Sentetik*	*3*	*210*	*5*
ExclaStar	*Sentetik*	*2*	*755*	*3*
Fisher Iris	*Gerçek*	*4*	*150*	*3*
Breast Cancer	*Gerçek*	*9*	*699*	*2*
Wine	*Gerçek*	*13*	*178*	*3*
Occupancy	*Gerçek*	*5*	*8143*	*2*
Aggregation	*Sentetik*	*2*	*788*	*7*
KDD	*Gerçek*	*38*	*50000*	*20*
Thyroid	*Gerçek*	*4*	*215*	*3*
MrData	*Sentetik*	*2*	*42470*	*5*
Zelnik4	*Sentetik*	*2*	*622*	*5*
Xclara	*Sentetik*	*2*	*3000*	*3*
Zelnik2	*Sentetik*	*2*	*303*	*3*
Sizes	*Sentetik*	*2*	*1000*	*4*
Long1	*Sentetik*	*2*	*1000*	*2*
Long3	*Sentetik*	*2*	*1000*	*2*
Cure-t0-2000n	*Sentetik*	*2*	*2000*	*3*
Chainlink	*Sentetik*	*2*	*1000*	*2*

Tablo 3. KD-AR Stream algoritmasında kullanılan parametreler (Test parameters of KD-AR Stream algorithm)

Algoritma	Veri Kümeleri	N	W	r	r_threshold	r_max
KD-AR Stream	*IdealData*	*7*	*22*	*0,1*	*0,02*	*0,2*
	ExclaStar	*11*	*48*	*0,5*	*0,03*	*0,62*
	Fisher Iris	*14*	*71*	*0,1*	*0,1*	*0,3*
	Breast Cancer	*23*	*91*	*0,5*	*0,1*	*0,69*
	Wine	*12*	*33*	*0,8*	*0,02*	*0,83*
	Occupancy	*24*	*72*	*0,3*	*0,03*	*0,35*
	Aggregation	*18*	*75*	*0,1*	*0,01*	*0,22*
	KDD	*28*	*89*	*0,8*	*0,07*	*1,07*
	Thyroid	*5*	*38*	*0,2*	*0,03*	*0,26*
	MrData	*23*	*76*	*0,1*	*0,01*	*0,23*
	Zelnik4	*21*	*87*	*0,1*	*0,03*	*0,22*
	Xclara	*12*	*39*	*0,2*	*0,1*	*0,37*
	Zelnik2	*16*	*51*	*0,1*	*0,04*	*0,3*
	Sizes	*11*	*41*	*0,1*	*0,02*	*0,21*
	Long1	*9*	*46*	*0,6*	*0,1*	*0,79*
	Long3	*29*	*71*	*0,2*	*0,07*	*0,38*
	Cure-t0-2000n	*16*	*56*	*0,3*	*0,03*	*0,43*
	Chainlink	*10*	*78*	*0,9*	*0,01*	*1.07*

Tablo 4. BT-AR Stream algoritmasında kullanılan parametreler (Test parameters of BT-AR Stream algorithm)

Algoritma	Veri Kümeleri	N	W	r	r_threshold	r_max
BT-AR Stream	*IdealData*	*6*	*72*	*0,8*	*0,02*	*0,84*
	ExclaStar	*6*	*60*	*0,20*	*0,07*	*0,45*
	Fisher Iris	*15*	*32*	*0,70*	*0,03*	*0,75*
	Breast Cancer	*29*	*82*	*0,60*	*0,10*	*0,84*
	Wine	*8*	*26*	*0,50*	*0,07*	*0,61*
	Occupancy	*17*	*82*	*0,20*	*0,08*	*0,47*
	Aggregation	*10*	*66*	*0,10*	*0,01*	*0,25*
	KDD	*25*	*46*	*0,08*	*0,02*	*1,06*
	Thyroid	*5*	*53*	*0,30*	*0,02*	*0,40*
	MrData	*16*	*73*	*0,20*	*0,01*	*0,34*
	Zelnik4	*23*	*56*	*0,10*	*0,01*	*0,24*
	Xclara	*5*	*20*	*0,20*	*0,08*	*0,44*
	Zelnik2	*20*	*78*	*0,10*	*0,03*	*0,20*
	Sizes	*6*	*61*	*0,10*	*0,03*	*0,26*
	Long1	*24*	*62*	*0,20*	*0,09*	*0,29*
	Long3	*28*	*56*	*0,20*	*0,07*	*0,33*
	Cure-t0-2000n	*18*	*52*	*0,10*	*0,01*	*0,19*
	Chainlink	*5*	*73*	*0,90*	*0,01*	*1,10*

Stream algoritması yapılan 60 testin 35'inde en yüksek değere ulaşırken, BT-AR Stream ve VP-AR Stream algoritmaları ise sırasıyla 28 ve 29 olarak tespit edilmiştir. Şekil 15'te ise algoritmaların KDD veri kümesi üzerinde test edilmesi ile elde edilen sonuçlar sunulmuştur. Ayrıca önerilen algoritmaların literatürdeki farklı algoritmalarla KDD veri kümesi üzerinden karşılaştırılması Tablo 7'de gösterilmektedir. Elde edilen sonuçlar dikkate alındığında, BT-AR Stream algoritmasının diğer algoritmalara göre %89,15 ARI skoru ile daha iyi bir sonuç ürettiği görülmektedir.

4.3. Çalışma Zamanı Karmaşıklığı (Comparision of Run-time Complexity)

Tablo 1'de verildiği gibi KD-AR Stream, BT-AR Stream ve VP-AR Stream algoritmalarının çalışma zamanları birbirlerine benzerlik göstermektedir. Ancak, seçilen parametreler çalışma zamanını doğrudan etkilemektedir. Şekil 15'te görüldüğü gibi KDD veri setinde en yüksek ARI değerlerini veren parametrelere karşılık algoritmaların çalışma zamanları karşılaştırıldığında BT-AR Stream algoritmasının çalışma zamanının daha iyi olduğu Şekil 16'da görülmektedir.

Tablo 6. Algoritmaların kümeleme başarısı (Clustering performance of the algorithms)

Veri Kümeleri	ARI			Saflık			Silhouette Metriği [-1, 1]		
	KD-AR Stream	BT-AR Stream	VP-AR Stream	KD-AR Stream	BT-AR Stream	VP-AR Stream	KD-AR Stream	BT-AR Stream	VP-AR Stream
IdealData	*0,6339*	*0,5570*	*0,575*	*0,8190*	*0,7000*	*0,7571*	*0,4023*	*0,4956*	*0,8185*
ExclaStar	*0,9739*	*0,9704*	*0,9739*	*0,9907*	*0,9907*	*0,9907*	*0,5539*	*0,4581*	*0,5539*
Fisher Iris	*0,5681*	*0,7087*	*0,7853*	*0,6667*	*0,8867*	*0,9200*	*0,6295*	*0,3234*	*0,3398*
Breast Cancer	*0,8502*	*0,8502*	*0,8556*	*0,9614*	*0,9614*	*0,9628*	*0,5794*	*0,5794*	*0,5781*
Wine	*0,8965*	*0,5238*	*0,7406*	*0,9663*	*0,7247*	*0,8989*	*0,2755*	*0,2191*	*0,2286*
Glass	*0,2311*	*0,2374*	*0,1998*	*0,4533*	*0,4579*	*0,4439*	*0,4876*	*0,4936*	*0,5193*
Aggregation	*0,9053*	*0,8930*	*0,8371*	*0,9518*	*0,9480*	*0,8629*	*0,4580*	*0,4241*	*0,4051*
KDD	*0,6818*	*0,8915*	*0,6629*	*0,9629*	*0,9683*	*0,9552*	*0,2056*	*0,6718*	*0,4968*
Thyroid	*0,8670*	*0,8123*	*0,8002*	*0,9488*	*0,9163*	*0,8837*	*0,4134*	*0,4341*	*0,3007*
MrData	*0,9742*	*0,9688*	*0,9647*	*0,9859*	*0,9826*	*0,9802*	*0,5475*	*0,5512*	*0,5560*
Zelnik4	*0,9920*	*0,9920*	0,7806	*0,9968*	*0,9968*	0,8167	*0,5669*	*0,5669*	0,4584
Xclara	*0,9905*	*0,9909*	*0,9906*	*0,9970*	*0,9973*	*0,9970*	*0,5805*	*0,5860*	*0,6584*
Zelnik2	*1,0000*	*1,0000*	*1,000*	*1,0000*	*1,0000*	*1,000*	*0,4699*	*0,4699*	*0,4699*
Sizes	*0,9571*	*0,9891*	*0,9891*	*0,9770*	*0,9960*	*0,9960*	*0,5711*	*0,5560*	*0,5480*
Long1	*1,0000*	*1,0000*	*1,000*	*1,0000*	*1,0000*	*1,000*	*0,6820*	*0,6820*	*0,6820*
Long3	*1,0000*	*1,0000*	*1,000*	*1,0000*	*1,0000*	*1,000*	*0,6795*	*0,6795*	*0,6795*
Cure-t0-2000n	*0,9976*	*1,0000*	*1,000*	*0,9990*	*1,0000*	*1,000*	*0,4119*	*0,4144*	*0,4144*
Chainlink	*0,7290*	*0,5266*	*0,7020*	*0,9270*	*0,8630*	*0,9190*	*0,1153*	*0,1306*	*0,1763*

Tablo 7. Önerilen algoritmaların KDD veri seti üzerindeki performanslarının literatürdeki algoritmalarla karşılaştırılması (Comparison of the performances of the proposed algorithms on the KDD dataset with the algorithms in the literature)

Çalışma	Yıl	Veri Kümesi Uygunluğu	Veri Sayısı	Yöntem	ARI
CEDAS [29]	2017	Benzer	495020	Mikro-küme yaklaşımı	0,2415
pcStream2 [30]	2017	Aynı	50.000	Temel bileşenler analizi	0,7625
KD-AR Stream [37]	2019	Aynı	50.000	K-boyutlu ağaç	0,6818
TSHC [31]	2022	Benzer	495020	Uzaysal hiper küpler	0,2265
VP-AR Stream (Önerilen)	Önerilen	Aynı	50.000	Bakış noktası ağacı	0,663
BT-AR Stream (Önerilen)	Önerilen	Aynı	50.000	Küre ağacı	0,8915

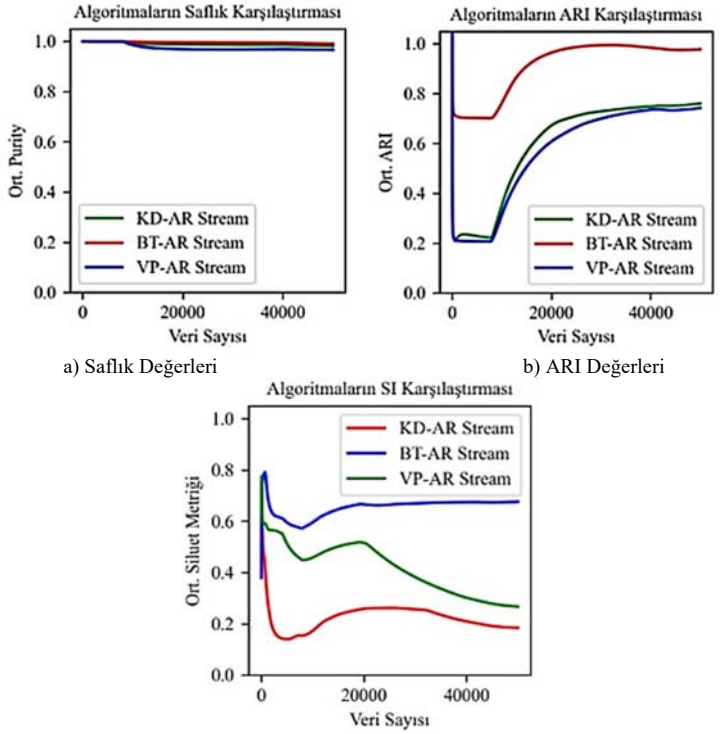
5. Sonuçlar ve Tartışmalar (Results and Discussions)

Bu çalışmada KD-AR Stream algoritmasına alternatif olarak geliştirilen Küre Ağacı tabanlı BT-AR Stream ve Bakış Noktası Ağacı tabanlı VP-AR Stream algoritmaları, KD-AR Stream algoritması ile hem kümeleme hem de çalışma zamanı açısından karşılaştırılmıştır. Yapılan deneysel çalışmaların sonucuna göre; kümeleme başarısı açısından her üç algoritmanın da genel olarak başarılı sonuçlar verdiği, bununla beraber her bir algoritmanın belirli veri kümelerinde daha başarılı olduğu tespit edilmiştir. ExclaStar, Breast Cancer, Occupancy, Zelnik2, Long1 ve Long3 veri setlerinde her üç algoritma da birbirine yakın sonuçlar vermektedir. Ancak Fisher Iris, Wine, KDD, Thyroid ve Chainlink veri setlerinde oldukça farklı sonuçlar elde edilmiştir. Buna rağmen Tablo 6'da da görülebileceği gibi kümeleme başarısı açısından KD-AR Stream algoritmasının diğer iki algoritmaya göre daha iyi sonuç verdiği söylenebilir.

Algoritmalar çalışma zamanı açısından kıyaslandığında Şekil 16'da görüldüğü üzere BT-AR Stream algoritması daha avantajlıdır. KD-AR Stream algoritması ise çalışma zamanı açısından performansı en düşük algoritma olarak tespit edilmiştir. Özellikle akan veri kümeleme uygulamalarının hedefinin büyük veri setlerini gerçek zamanlı olarak kümeleme olduğu göz önünde bulundurulduğunda, BT-AR Stream algoritmasının daha avantajlı olduğu söylenebilir. Deneysel çalışmada kullandığımız MrData veri setinin %10'u sapan verilerden oluşmaktadır. Her üç algoritma da bu veri setinde yüksek kümeleme başarısına ulaşmaktadır. Bununla birlikte KD-AR Stream %97,4'ü aşan başarı ile en yüksek ARI değerine ulaşan algoritmadır. Ancak Silhouette metriği olarak ise en yüksek değere VP-AR Stream algoritması ulaşmaktadır. Genel bir değerlendirme yapacak olursak her üç algoritmanın sapan verilere karşı çok dirençli bir yapıya sahip olduğunu söylemek mümkündür.

6. Sonuçlar (Conclusions)

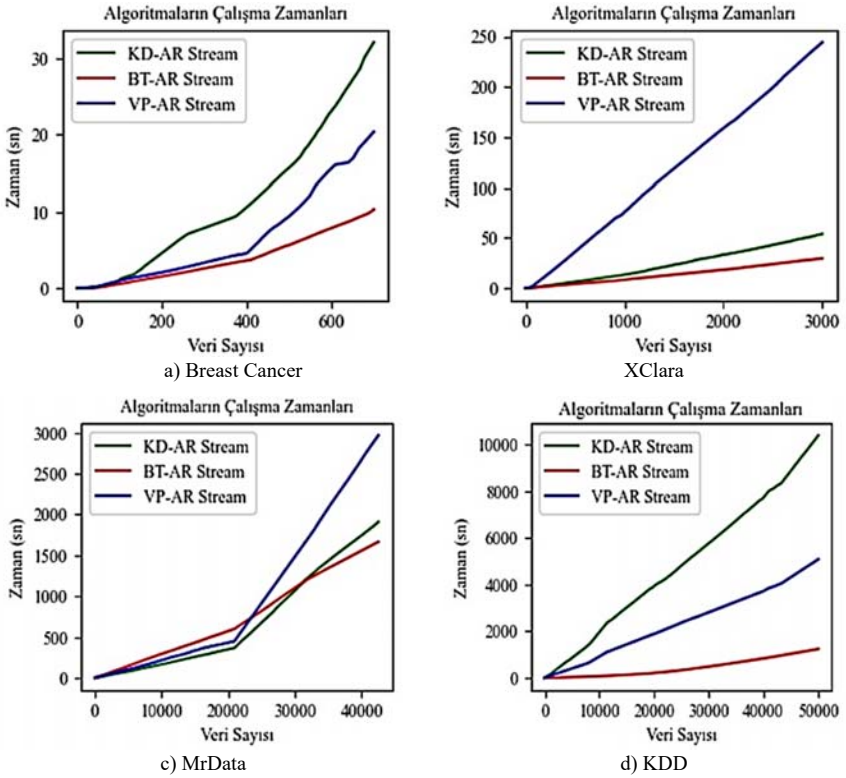
Bu çalışmada, akan veri kümeleme için iki yeni algoritma geliştirilmiş ve başarıyla uygulanarak test edilmiştir. Geliştirilen BT-AR Stream ve VP-AR Stream algoritmaları, daha önce Şenol ve Karacan tarafından geliştirilen KD-AR Stream algoritması ile çeşitli veri kümeleri kullanılarak ARI, Saflık ve Silhouette metrikleri üzerinden kıyaslanmıştır. Yapılan deneysel çalışmalar neticesinde, geliştirilen algoritmaların başarılı sonuçlar verdikleri görülmüştür. Elde edilen sonuçlar analiz edildiğinde, her üç algoritma da başarılı sonuçlar vermesine rağmen BT-AR Stream algoritmasının daha başarılı sonuçlar verdiği gözlemlenmektedir. KDD veri seti özelinde, BT-AR Stream algoritmasının %89,15 ARI, %96,83 Saflık ve %67,18 Silhouette değerleri ile diğer iki algoritmadan daha başarılı sonuçlar verdiği tespit edilmiştir. Bununla beraber BT-AR Stream algoritması çalışma zamanı açısından da diğer iki algoritmaya göre daha hızlı çalışmaktadır. Ayrıca, BT-AR Stream algoritmasının literatürdeki



a) Saflık Değerleri b) ARI Değerleri

c) Siluet Değerleri

Şekil 15. Algoritmaların KDD veri seti üzerinde test edilmesi ile elde edilen sonuçlar
(Obtained results by testing the algorithms on the KDD dataset)



a) Breast Cancer XClara

c) MrData d) KDD

Şekil 16. Algoritmaların çalışma zamanlarının karşılaştırması (Run-time comparison of algorithms)

güncel diğer algoritmalara kıyasla en iyi ARI değerini verdiği de gözlemlenmektedir. Gelecek çalışmalarında, farklı öznitelik seçim tekniklerinin geliştirilen algoritmaların performansına etkilerinin araştırılması ve yeni ağaç türlerinin kullanılması planlanmaktadır.

Teşekkür (Acknowledgment)

Yazarlar, bu çalışmanın yapılmasında desteklerini esirgemeyen Prof. Dr. Hacer Karacan'a teşekkür eder.

Kaynaklar (References)

1. AlNuaimi N., Masud M. M., Serhani, M. A., Zaki N., Streaming feature selection algorithms for big data: A survey, Applied Computing and Informatics, 18 (1), 113-135, 2020.
2. Das, A., Das S., Rathee N., Roles of Big Data, Data Science, Artificial Intelligence in Entrepreneurships, International Conference on Advances in Management Practices (ICAMP 2021), Delhi-India, 1-10, 17-18 Aralık, 2021.
3. Zheng X., Li P., Chu Z., Hu X., A survey on multi-label data stream classification. IEEE Access, 8, 1249-1275, 2019.
4. Hançer E., Differential evolution based multiple kernel fuzzy clustering, Journal of the Faculty of Engineering and Architecture of Gazi University, 34 (3), 1281-1294, 2019.
5. Datlıca M.T., E. Çakıt, Estimation of clustering parameters and anomaly detection in tracking devices with changeable position time Journal of the Faculty of Engineering and Architecture of Gazi University, 36 (1), 373-394, 2021.
6. Jain A. K., Data clustering: 50 years beyond K-means. Pattern recognition letters, 31 (8), 651-666, 2010.
7. Yin C., Zhang S., Yin Z., Wang J., Anomaly detection model based on data stream clustering, Cluster Computing, 22, 1729-1738, 2019.
8. Laurinec P., M. Lucká, Interpretable multiple data streams clustering with clipped streams representation for the improvement of electricity consumption forecasting, Data Mining and Knowledge Discovery, 33, 413-445, 2019.
9. Gajowniczek K., Bator M., Ząbkowski T., Whole time series data streams clustering: dynamic profiling of the electricity consumption. Entropy, 22 (12), 1414, 2020.
10. Tajalizadeh, H., R. Boostani, A novel stream clustering framework for spam detection in Twitter. IEEE Transactions on Computational Social Systems, 6 (3), 525-534, 2019.
11. Yin J., Chao D., Liu Z., Zhang W., Yu X., Wang J., Model-based clustering of short text streams. 24th ACM SIGKDD international conference on knowledge discovery & data mining, London-England, 19-23 Ağustos, 2018.
12. Diaz-Rozo J., Bielza C., Larrañaga P., Clustering of data streams with dynamic gaussian mixture models: an IoT application in industrial processes. IEEE Internet of Things Journal, 5 (5), 3533-3547, 2018.
13. Al-Shammari A., Zhou R., Naseriparsaa, Liu C., An effective density-based clustering and dynamic maintenance framework for evolving medical data streams. International journal of medical informatics, 126, 176-186, 2019.
14. Hendricks D., Using real-time cluster configurations of streaming asynchronous features as online state descriptors in financial markets. Pattern Recognition Letters, 97, 21-28, 2017.
15. Zubaroğlu A., Atalay V., Data stream clustering: a review. Artificial Intelligence Review, 54 (2), 1201-1236, 2021.
16. Kokate U., Deshpande A., Mahalle P., Patil P., Data stream clustering techniques, applications, and models: comparative analysis and discussion. Big Data and Cognitive Computing, 2 (4), 32, 2018.
17. Mansalis S., Ntoutsi E., Pelekis N., Theodoridis Y., An evaluation of data stream clustering algorithms. Statistical Analysis and Data Mining: The ASA Data Science Journal, 11 (4), 167-187, 2018.
18. Kranen P., Assent I., Baldauf C., Seidl T., The clustree: indexing micro-clusters for anytime stream mining. Knowledge and information systems, 29, 249-272, 2011.
19. Zhang T., Ramakrishnan R., Livny M., BIRCH: an efficient data clustering method for very large databases. ACM sigmod record, 25 (2), 103-114, 1996.
20. Lang A., Schubert E., BETULA: Fast clustering of large data with improved BIRCH CF-Trees. Information Systems, 108, 101918, 2022.
21. Aggarwal C.C., Philip S. Y., Han J., Wang J., A framework for clustering evolving data streams. 29th Annual International Conference on Very Large Data Bases, Berlin-Germany, 81-92, 9-12 Eylül 2013.
22. Zhou A., Cao F., Qian W., Tracking clusters in evolving data streams over sliding windows, Knowledge and Information Systems, 15, 181-214, 2008.
23. Karypis G., Han E. H., Kumar V., Chameleon: Hierarchical clustering using dynamic modeling. Computer, 32 (8), 68-75, 1999.
24. Lühr S., Lazarescu M, Incremental clustering of dynamic data streams using connectivity based representative points. Data & knowledge engineering, 68 (1), 1-27, 2009.
25. Udommanetanakit K., Rakthanmanon T., Waiyamai K, E-stream: Evolution-based technique for stream clustering. Advanced Data Mining and Applications: Third International Conference, ADMA 2007, Harbin-China, 605-615, 6-8 Ağustos, 2007.
26. Meesuksabai W., Kangkachit T., Waiyamai K., Hue-stream: Evolution-based clustering technique for heterogeneous data streams with uncertainty. Advanced Data Mining and Applications: Seven International Conference, ADMA 2011, Beijing-China, 27-40, 17-19 Aralık, 2011.
27. Nikpour S., Asadi S., A dynamic hierarchical incremental learning-based supervised clustering for data stream with considering concept drift. Journal of Ambient Intelligence and Humanized Computing, 13 (6), 2983-3003, 2022.
28. Sangma J.W., Sarkar M., Pal V., Agrawal, Hierarchical clustering for multiple nominal data streams with evolving behaviour. Complex & Intelligent Systems, 8, 1737-1761, 2022.
29. Hyde R., Angelov P., MacKenzie A. R., Fully online clustering of evolving data streams into arbitrarily shaped clusters. Information Sciences, 382, 96-114, 2017.
30. Mirsky Y. Halpern T., Upadhyay R. Toledo S., Elovici Y., Enhanced situation space mining for data streams, SAC 2017: Symposium on Applied Computing, Marrakech-Morocco, 842-849, 3-7 Nisan 2017.
31. Al-amri R., Murugesan R. K., Almutairi M., Munir K., Alkawsi G., Baashar Y., A Clustering Algorithm for Evolving Data Streams Using Temporal Spatial Hyper Cube. Applied Sciences, 12 (13), 6523, 2022.
32. Ahmed R., Dalkılıç G.,, Erten Y., DGStream: High quality and efficiency stream clustering algorithm. Expert Systems with Applications, 141, 112947, 2020.
33. Li Y., Li H., Wang Z., Liu B., Cui J., Fei H., Esa-stream: Efficient self-adaptive online data stream clustering. IEEE Transactions on Knowledge and Data Engineering, 34 (2), 617-630, 2022.
34. Huang L., Wang C. D., Chao H. Y., MVStream: Multiview data stream clustering. IEEE Transactions on neural networks and learning systems, 31 (9), 3482-3496, 2020.
35. Laohakiat S., Sa-Ing V., An incremental density-based clustering framework using fuzzy local clustering. Information Sciences, 547, 404-426, 2021.
36. Nguyen H. L., Woon Y. K., Ng W. K., A survey on data stream clustering and classification. Knowledge and information systems, 45, 535-569, 2015.
37. Şenol A., Karacan H., Kd-tree and adaptive radius (KD-AR Stream) based real-time data stream clustering, Journal of the Faculty of Engineering and Architecture of Gazi University, 35 (1), 337-354, 2020.
38. Bentley J.L., Multidimensional binary search trees used for associative searching. Communications of the ACM, 18 (9), 509-517, 1975.
39. Omohundro S. M., Five balltree construction algorithms. Technical Report TR-89-063, International Computer Science Institute Berkeley, 1989.
40. Yianilos, P.N. Data Structures and Algorithms for Nearest Neighbor. Fourth annual ACM-SIAM Symposium on Discrete algorithms. , Texas-USA, 311-321, 25-27 Ocak, 1993.
41. Cao F., Estert M., Qian W., Zhou A., Density-based clustering over an evolving data stream with noise. 2006 SIAM international conference on data mining, Maryland-USA, 328-339, 20-22 Nisan, 2006.
42. Center for Machine Learning and Intelligent Systems, UCI Machine Learning Repository, https://archive.ics.uci.edu/ml/index.php, Erişim Tarihi Mayıs 30, 2022