T.C.
İSTANBUL ÜNİVERSİTESİ-CERRAHPAŞA
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

AKAN VERİNİN MAKİNE ÖĞRENME ALGORİTMALARI KULLANILARAK ÖLÇEKLENMESİ

Önder AYKURT

DANIŞMAN
Doç. Dr. Zeynep ORMAN

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

İSTANBUL-2022

Bu çalışma, 6.01.2022 tarihinde aşağıdaki jüri tarafından Bilgisayar Mühendisliği Anabilim Dalı, Bilgisayar Mühendisliği Programında Yüksek Lisans tezi olarak kabul edilmiştir.

Tez Jürisi

Doç. Dr. Zeynep ORMAN(Danışman)
İstanbul Üniversitesi-Cerrahpaşa
Mühendislik Fakültesi

Doç. Dr. Selçuk SEVGEN
İstanbul Üniversitesi-Cerrahpaşa
Mühendislik Fakültesi

Dr. Öğr. Üyesi Ali HAMİTOĞLU
İstanbul Sabahattin Zaim Üniversitesi
Mühendislik ve Doğa Bilimleri Fakültesi

20.04.2016 tarihli Resmi Gazete'de yayımlanan Lisansüstü Eğitim ve Öğretim Yönetmeliğinin 9/2 ve 22/2 maddeleri gereğince; Bu Lisansüstü teze, İstanbul Üniversitesi-Cerrahpaşa'nın abonesi olduğu intihal yazılım programı kullanılarak Lisansüstü Eğitim Enstitüsü'nün belirlemiş olduğu ölçütlere uygun rapor alınmıştır.

ÖNSÖZ

Tez çalışma sürecinde akademik desteğini esirgemeyen kıymetli hocam Sayın Doç. Dr. Zeynep Orman' a, hayatım boyunca desteklerini esirgemeyen değerli aileme ve tez sürecinde yanımda olan eşim Hayriye Hatıl Aykurt' a ayrıca teşekkür ederim.

Tez çalışmamın, araştırmacı herkese yol göstermesi dileğiyle…

Ocak 2022 Önder AYKURT

İÇİNDEKİLER

ÖNSÖZ ..... **iv**
İÇİNDEKİLER ..... **v**
ŞEKİL TABLOSU ..... **vii**
TABLO LİSTESİ ..... **x**
SİMGE VE KISALTMA LİSTESİ ..... **xi**
ÖZET ..... **xii**
SUMMARY ..... **xiii**
1. GİRİŞ **1**
2. GENEL KISIMLAR **2**
2.1. AKAN VERİ 2
2.1.1. Akan Veri Kaynakları 3
2.1.2. Akan Verinin Temel Özellikleri 4
2.2. MAKİNE ÖĞRENMESİ 5
2.2.1. Makine Öğrenmesi Kavramı ve Gelişimi 5
2.2.2. Makine Öğrenmesi Süreçleri 7
2.2.2.1. Problemin Tespiti *8*
2.2.2.2. Gereken Verinin Toplanması *8*
2.2.2.3. Veri İşleme *8*
2.2.2.4. Eğitici Modelin Oluşturulması *9*
2.2.2.5. Eğitici Modelin Test Edilmesi ve Değerlendirilmesi *9*
2.3. MAKİNE ÖĞRENMESİ METOTLARI 10
2.3.1. Denetimli Öğrenme 10
2.3.2. Denetimsiz Öğrenme 14
2.3.3. Takviyeli Öğrenme 16
2.4. LİTERATÜRDEKİ ÇALIŞMALAR 17
3. MALZEME VE YÖNTEM **19**
3.1. KULLANILAN YÖNTEMLER 19
3.1.1. CluStream-Kmeans++ Kümeleme Algoritması 19
3.1.2. K-means++ Kümeleme Algoritması 21
3.1.3. Test Sürecinde Kullanılan Algoritmalar 22
3.1.3.1. ClusTree Algoritması *23*
3.1.3.2. DenStream Algoritması *23*
3.1.3.3. StreamKM++ Algoritması *24*
3.2. UYGULAMA 24
3.2.1. MOA Kurulumu 24
3.2.2. MOA' nın Kullanımı 25

3.2.2.1. Kurulum Paneli (Setup Panel) *25*
3.2.2.2. Görselleştirme paneli (Visualization Panel) *28*
3.2.3. MOA Uygulama Entegrasyonu 28
3.2.4. CluStream-Kmeans++ Algoritma Entegrasyonu 33
3.2.5. Sistem Durum Tespiti 39
3.2.6. Ölçekleme İşleminin Tamamlanması 41
4. BULGULAR 43
4.1. PLATFORM, DEĞERLENDİRME METRİKLERİ VE VERİ SETİ 43
4.1.1. Kullanılan Platformlar 43
4.1.1.1. Teknik Altyapı *43*
4.1.1.2. Intellij IDEA *43*
4.1.2. Değerlendirme İndeksleri 43
4.1.2.1. F1-Ölçütü (F1-Measure) *44*
4.1.2.2. Saflık (Purity) *44*
4.1.2.3. Hata Kareleri Toplamı (SSE) *45*
4.1.2.4. Rand İndeks *45*
4.1.2.5. Silhouette İndeksi *46*
4.1.3. Veri Seti 46
4.1.3.1. Gerçek Veri Seti *46*
4.1.3.2. Yapay Veri Seti *48*
4.2. DENEY SONUÇLARI 49
4.2.1. Algoritma Performans Değerlendirmesi 49
4.2.1.1 Gerçek Veri Setleri ile Değerlendirme *49*
4.2.1.2. Yapay Veri Setleri ile Değerlendirme *64*
4.2.2. Algoritma Çalışma Zamanı Değerlendirmesi 72
4.2.3. Sistem Yükünün İncelenmesi 75
5. TARTIŞMA VE SONUÇ 79
KAYNAKLAR 81

ŞEKİL TABLOSU

Sayfa No

Şekil 2.1: Akan veri örneği. 2

Şekil 2.2: Derin öğrenme, makine öğrenimi ve yapay zeka karşılaştırılması. 7

Şekil 2.3: Makine öğrenmesi süreci adımları. 8

Şekil 2.4: Doğrusal regresyon örneği. 11

Şekil 2.5: Destek vektör makineleri. 12

Şekil 2.6: Takviyeli öğrenme yöntemi. 16

Şekil 3.1: Geliştirilen CluStream-Kmeans++ algoritma akışı. 22

Şekil 3.2: StreamKM++ algoritma akışı. 24

Şekil 3.3: MOA proje bağımlılığı. 25

Şekil 3.4: MOA uygulamasının başlatılması. 25

Şekil 3.5: Kurulum paneli. 25

Şekil 3.6: Veri kaynağı. 26

Şekil 3.7: MOA örnek algoritma seçimi. 27

Şekil 3.8: Veri örneği. 27

Şekil 3.9: Görselleştirme (Visualization) paneli. 28

Şekil 3.10: MOA mimarisi. 29

Şekil 3.11: Akan veri entegrasyonu Java kodu-1. 30

Şekil 3.12: Akan veri entegrasyonu Java kodu-2. 31

Şekil 3.13: Akan veri entegrasyonu Java kodu-3. 32

Şekil 3.14: Pirimidal zaman çerçevesi örneği. 33

Şekil 3.15: CluStream-Kmeans++ algoritması Java uygulama kodu-1. 34

Şekil 3.16: CluStream-Kmeans++ algoritması Java uygulama kodu-2. 35

Şekil 3.17: CluStream-Kmeans++ algoritması Java uygulama kodu-3. 36

Şekil 3.18: CluStream-Kmeans++ algoritması Java uygulama kodu-4. 37

Şekil 3.19: CluStream-Kmeans++ algoritması Java uygulama kodu-5. 38

Şekil 3.20: CluStream-Kmeans++ algoritması Java uygulama kodu-6. 39

Şekil 3.21: Akan veri işleme sırasında sistemin anlık değerlendirilmesi. ……...40

Şekil 3.22: MOA-Apache Flink bağlantısı. 41

Şekil 4.1: CluStream-Kmeans++ algoritması F1-Kesinlik grafiği. 50

Şekil 4.2: CluStream-Kmeans++ algoritması F1-Hassaslık grafiği. 50

Şekil 4.3: CluStream-Kmeans++ algoritması hata kareleri toplamı grafiği. 51

Şekil 4.4: CluStream-Kmeans++ algoritması Silhouette indeks grafiği. 51

Şekil 4.5: CluStream-Kmeans++ algoritması Rand indeks grafiği. 52

Şekil 4.6: CluStream-Kmeans++ algoritması saflık grafiği. 52

Şekil 4.7: CluStream algoritması F1-Kesinlik grafiği. 53

Şekil 4.8: CluStream algoritması F1-Hassaslık grafiği. 53

Şekil 4.9: CluStream algoritması hata kareleri toplamı grafiği. 54

Şekil 4.10: CluStream algoritması Silhouette indeksi grafiği. 54

Şekil 4.11: CluStream algoritması Rand indeksi grafiği. 55

Şekil 4.12: CluStream algoritması saflık grafiği. 55

Şekil 4.13: StreamKM++ algoritması F1-Kesinlik grafiği. 56

Şekil 4.14: StreamKM++ algoritması F1-Hassaslık grafiği. 56

Şekil 4.15: StreamKM++ algoritması hata kareleri toplamı grafiği. 57

Şekil 4.16: StreamKM++ algoritması Silhouette indeksi grafiği. 57

Şekil 4.17: StreamKM++ algoritması Rand indeks grafiği. 58

Şekil 4.18: StreamKM++ algoritması saflık grafiği. 58

Şekil 4.19: ClusTree algoritması F1-Kesinlik grafiği. 59

Şekil 4.20: ClusTree algoritması F1-Hassaslık grafiği. 59

Şekil 4.21: ClusTree algoritması hata kareleri toplamı grafiği. 60

Şekil 4.22: ClusTree algoritması Silhouette indeksi grafiği. 60

Şekil 4.23: ClusTree algoritması Rand indeks grafiği. 61

Şekil 4.24: ClusTree algoritması saflık grafiği. ……...61

Şekil 4.25: DenStream algoritması F1-Kesinlik grafiği.....62

Şekil 4.26: DenStream algoritması F1-Hassaslık grafiği.....62

Şekil 4.27: DenStream algoritması hata kareleri toplamı grafiği.....63

Şekil 4.28: DenStream algoritması Silhouette indeksi grafiği.....63

Şekil 4.29: DenStream algoritması Rand indeks grafiği.....64

Şekil 4.30: DenStream algoritması saflık grafiği.....64

Şekil 4.31: Gerçek veri ile çalışma zamanı grafiği. 74

Şekil 4.32: Yapay veri ile çalışma zamanı grafiği. 75

Şekil 4.33: Gerçek veri ile ölçekleme öncesi işlemci kullanım oranı. 75

Şekil 4.34: Gerçek veri ile ölçekleme sonrası işlemci kullanım oranı. 76

Şekil 4.35: Gerçek veri ile ölçekleme öncesi bellek kullanım oranı. 76

Şekil 4.36: Gerçek veri ile ölçekleme sonrası bellek kullanım oranı. 76

Şekil 4.37: Yapay veri ile ölçekleme öncesi işlemci kullanım oranı. 77

Şekil 4.38: Yapay veri ile ölçekleme sonrası işlemci kullanım oranı. 78

Şekil 4.39: Yapay veri ile ölçekleme öncesi bellek kullanım oranı. 78

Şekil 4.40: Yapay veri ile ölçekleme öncesi bellek kullanım oranı. 78

TABLO LİSTESİ

Sayfa No

Tablo 4.1: Forest Cover Type veri seti içerisindeki veri dağılımı. 47

Tablo 4.2: Forest Cover Type veri setindeki öznitelik uzayı. 47

Tablo 4.3: Yapay veri setiyle oluşturulan durumlar. 48

Tablo 4.4: Forest Cover Type veri setiyle varsayılan parametreler ile elde edilen sonuçlar.. 49

Tablo 4.5: Durum 1 parametreleri ile sonuçlar. 65

Tablo 4.6: Durum 2 parametreleri ile sonuçlar. 65

Tablo 4.7: Durum 3 parametreleri ile sonuçlar. 66

Tablo 4.8: Durum 4 parametreleri ile sonuçlar. 67

Tablo 4.9: Durum 5 parametreleri ile sonuçlar. 67

Tablo 4.10: Durum 6 parametreleri ile sonuçlar. 68

Tablo 4.11: Durum 7 parametreleri ile sonuçlar. 69

Tablo 4.12: Durum 8 parametreleri ile sonuçlar. 70

Tablo 4.13: Durum 9 parametreleri ile sonuçlar. 70

Tablo 4.14: Durum 10 parametreleri ile sonuçlar. 71

Tablo 4.15: Tüm durumların ortalamaları. 72

Tablo 4.16: Gerçek veri setiyle çalışma süreleri. 73

Tablo 4.17: Yapay veri ile çalışma süreleri. 74

SİMGE VE KISALTMA LİSTESİ

Kısaltmalar	Açıklama
ATM	: Automatic Teller Machine
DBSCAN	: Density Based Spatial Clustering of Applications with Noise
STING	: Statistical Information Grid
MOA	: Massive Online Analysis
CFV	: Cluster Feature Vector
WEKA	: Waikato Environment for Knowledge Analysis
API	: Application Programming Interface
SSE	: Sum of Squared Error

ÖZET

YÜKSEK LİSANS TEZİ

AKAN VERİNİN MAKİNE ÖĞRENME ALGORİTMALARI KULLANILARAK ÖLÇEKLENMESİ

Önder AYKURT

İstanbul Üniversitesi-Cerrahpaşa

Lisansüstü Eğitim Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Zeynep Orman

Teknolojinin gün geçtikçe gelişmesiyle birlikte hayatımızdaki yeri ve önemi artmaktadır. Gelişen teknoloji, birçok cihazın birbirleriyle ve insanlarla olan etkileşimini arttırmıştır. Bu etkileşimin sonucunda ortaya büyük miktarda veri çıkmaktadır. Gerçek zamanlı üretilen bu veriler, üretildiği anda değerlidir. Özellikleri gereği sıralı, değişik boyutlarda ve düzensiz periyotlarda elde edilen bu veriler, akan veri olarak tanımlanmıştır. Akan veriler, hemen işlenmezse değerini kaybedebilir veya tamamen kaybolabilir. Bu nedenle, yapılandırılmamış verileri sürekli olarak alıp analiz edebilen ölçeklenebilir sistemlerin geliştirilmesi önemlidir. Literatürdeki çalışmaların çoğu mevcut şartlarda sistemin nasıl çalışacağı konusuna yoğunlaşmıştır.

Bu tez kapsamında, yukarıdaki problemlerden yola çıkarak, akan veriyi makine öğrenme algoritmaları kullanılarak anlık olarak analiz edebilen ölçeklenebilir bir sistem tasarımı amaçlanmıştır. Geliştirilen sistem ve algoritmalar, gerçek veri ve yapay veriler ile çalıştırılarak değerlendirme metrikleriyle sonuçlar elde edilmiş, ölçeklenme durumu anlık olarak izlenmiştir. Yapılan simülasyon çalışması sonucundaki veriler değerlendirilerek literatüre ve gelecek çalışmalara ışık tutmak amaçlanmıştır.

Ocak 2022, 94 Sayfa.

Anahtar Kelimeler: Akan veri işleme, Büyük veri, Makine öğrenmesi, Akan veri ölçekleme.

SUMMARY

M.Sc. THESIS

SCALING OF DATA STREAM BY USING MACHINE LEARNING ALGORITHMS

Önder AYKURT

İstanbul University- Cerrahpaşa

Institute of Graduate Studies

Department of Computer Engineering

Supervisor: Assoc. Prof. Dr. Zeynep Orman

With the development of technology day by day, its place and importance in our lives is increasing. Developing technology has increased the interaction of many devices with each other and with people. As a result of this interaction, a large amount of data emerges. This real-time generated data is valuable as soon as it is produced. These data, which are sequential due to their characteristics, obtained in different sizes and irregular periods, are defined as streaming data. Streaming data can lose value or be lost forever if not processed immediately. Therefore, it is important to develop scalable systems that can continuously receive and analyze unstructured data. Most of the studies in the literature have focused on how the system will work under current conditions.

Within the scope of this thesis, it is aimed to design a scalable system that can instantly analyze the flowing data using machine learning algorithms based on the above problems. The developed systems and algorithms were run with real data and artificial data, and results were obtained with evaluation metrics, and the scaling status was instantly monitored. It is aimed to shed light on the literature and future studies by evaluating the data as a result of the simulation study.

January 2022, 94 pages.

Keywords: Streaming data processing, Big data, Machine learning, Scaling streaming data.

1. GİRİŞ

Teknolojinin günden güne gelişmesiyle birlikte hayatımızdaki yeri ve önemi artmaktadır. Gelişen teknoloji, birçok cihazın birbirleriyle ve insanlarla olan etkileşimini arttırmıştır. Bu etkileşimin sonucunda ortaya büyük miktarda veri çıkmaktadır. Gerçek zamanlı üretilen veriler, ulaştığı anda değerlidir ve karar verme aşamalarını destekler. Özellikleri gereği sıralı, değişik boyutlarda ve düzensiz periyotlarda elde edilen bu veriler, akan veri olarak tanımlanmıştır. Akan veriler, hemen işlenmezse değerini kaybedebilir veya sonsuza dek kaybolabilir. Bu nedenle, yapılandırılmamış verileri sürekli olarak alıp analiz edebilen ölçeklenebilir sistemlerin geliştirilmesi önemlidir.

Akan veriyi analiz edip işleyebilen bir sistem kurulumu ve sürekliliği açısından, kaynak kullanımı önemli bir iştir. Sistemin başarısı, sistem kaynaklarının doğru kullanımına bağlıdır. Üretilen veriler çok hızlı arttığı için klasik veri analiz yöntemleri birçok açıdan yetersiz kalmaktadır. Bu verileri kaydetme gereksinimiyle birlikte, büyük kaynaklara duyulan ihtiyaç yüksek maliyetleri beraberinde getirir. Bu nedenle bu sistemler mali açıdan da verimli olmalıdır [1].

İnternet teknolojilerindeki gelişmelerle birlikte hayatımıza giren birçok uygulama gerçek zamanlı veri kullanmaktadır. Finansal uygulamalar, sosyal medya uygulamaları, e-ticaret, çevrimiçi reklamcılık uygulamalarında verinin gerçek zamanlı olarak işlenmesi güvenlik, kazanç gibi konularda önemli avantajlar sağlamaktadır. Gerçek zamanlı verilerin analizine ve bu verilerden sonuç üretmeye yönelik birçok yaklaşım mevcuttur. Sınıflandırma ve kümeleme yaklaşımları, akan veriden anlamlı çıktılar üretmek için kullanılmaktadır.

Naive Bayes [2] ve karar ağaçları [3] kullanılan sınıflandırma algoritmalarına örnek olarak gösterilebilir. Bunlara ek olarak, yapay sinir ağları [4] ve genetik algoritmalar [5] ile bunların hibrit yapıları kümeleme işlemlerinde kullanılmaktadır. Bu yöntemler statik verileri alarak ya da rastgele kabul ile kümeleme yapar. Bu açıdan bakıldığında, gelişen teknolojiyle birlikte artan veri hacmi için yeni yaklaşımlara ihtiyaç duyulmaktadır. Bu ihtiyaçla birlikte akan verilerin analiz edilip ölçeklendirilmesi konusu popüler bir çalışma konusu haline gelmiştir. Bu tez çalışmasında akan verinin, kümeleme algoritmaları kullanılarak ölçeklendirilmesi konusu incelenmiş olup, bu konuda çalışma yapmayı düşünen araştırmacılara ışık tutması amaçlanmıştır.

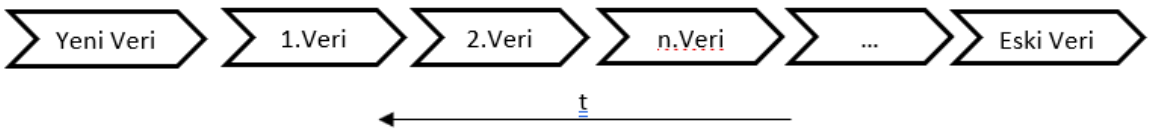
2. GENEL KISIMLAR

Bu bölümde akan verinin tanımı, makine öğrenmesi gibi temel bilgilerle birlikte literatürde yer alan çalışmalar incelenmiştir.

2.1. AKAN VERİ

Günümüzde internet teknolojilerindeki gelişmelerle birlikte finans, telekomünikasyon, web, sosyal ağ gibi kaynaklardan büyük miktarda veri üretilmektedir. Akan veri, büyük hacme sahip, gerçek zamanlı olarak işlemlerin durumunu tanımlayan, sonsuz, zamanla değişebilen ve öngörülemeyen veri akışları olarak tanımlanmaktadır [6].

Akan veriler, statik verilerden farklı özelliklere sahip veri kümeleridir. Akan verilerde algoritmanın işlem süresi, statik veri işleyen algoritmanın işlem süresine göre daha kritiktir. Akan veri, sisteme geldiği anda değerlidir ve hızlıca işlenip değerlendirilmesi gerekmektedir. Örneğin; finansal uygulamaların güvenliği konusunda uygulamaya gelen veriler, işlem güvenliği açısından o an değerlendirilmesi gerekmektedir. Statik verilerde tasarlanan veri modeli kalıcıdır, akan veride veriye göre kendini güncelleyebilen veri modeli kullanılır. Gelecek veri boyutu öngörülemediğinden, kullanılan veri modelinin, zamanla değişen veri akışına uyumlanması gerekmektedir. Akan verinin işlenip değerlendirilmesi için gerekli olan süre, statik veriye göre daha sınırlıdır. Değerlendirme süresinin kısalması, akan verinin kullanıldığı uygulamada verinin değeri bakımından kritik önem taşır [7].



Şekil 2.1: Akan veri örneği.

Veri miktarının artmasıyla birlikte, verilerin sürekli olarak depolandığı uygulamalar yerine akan verinin modellenip kullanıldığı uygulamalar ön plana çıkmaktadır. Bu modellemede, akan veri için hacim, doğruluk, çeşitlilik ve hız kavramları önemlidir. Hacim, akan verinin toplam büyüklüğünün bilinmemesini ve tüm verinin işlenmesinin büyüklüğünü ifade etmektedir. Büyük hacimdeki akan verinin tamamını depolamak veya belirli zaman aralıklarıyla taramak sistem performansına olumsuz etki etmektedir. Hız, algoritmanın akan

veriyi bir kez işleyebileceği bir periyot için veri üretilmesi anlamına gelmektedir. Akan veri zaman içinde değişebileceğinden kullanılan algoritmanın, veriyi birden fazla kez işlemesi durumunda modelin güncellenmesi gerekmektedir. Doğruluk, verilerin güvenirliliğini ve incelemeye ihtiyaç duyulup duyulmamasını ifade eder. Akan veri, genellikle heterojen yapıdadır ve farklı türlerdeki birçok veri birlikte işlenir. Çeşitlilik kavramı akan verinin bu özelliği hakkında bilgi vermektedir [8].

2.1.1. Akan Veri Kaynakları

Teknolojinin gelişmesiyle birlikte gittikçe artan boyutlarda üretilen verileri analiz etmek, işlemek ve oluşan çıktıları sunmak için çeşitli araçlar kullanılmaktadır. Verinin değerlendirilmesi için, akan veriye özgü bir analiz yöntemi ve sistem gerekmektedir. Akan veri kaynağı olarak sosyal medya uygulamaları, e-ticaret, mobil uygulamalar, nesnelerin interneti, operasyonel takip, reklamcılık gibi alanlar örnek olabilmektedir [9].

E-ticaret ve web uygulamalarının gelişmesiyle birlikte, web analizleri giderek önemli bir rol kazanmıştır. İlgili web sitesine gelen ziyaretçi sayısı, incelenen ürünler ile kullanıcı profili arasındaki ilişki gibi verileri analiz edebilmek için büyük veri işleme araçları kullanılmaktadır. Böylece verinin gerçek zamanlı olarak toplanıp işlenmesi mümkün olmuştur [10].

Fiziksel olarak izlenen operasyonel sistemler, akan verilerin temel veri kaynaklarından biridir. Temel olarak ayrık bilgisayar sistemlerinin genel performansına etki eden metrikler izlenir. Disk sürücülerinin durumu, işlemci yükü ve performansı, ağ kullanımı, depolama birim performansı ve erişim süreleri gibi birçok veri işlenip kaydedilmektedir. Bu sistemlerin izlenmesi, hem genel sistem performansı ve olası problemlerin tanımlanması konusunda önem taşımaktadır [11].

Verinin gerçek zamanlı olarak üretilip değerlendirildiği en önemli alanlardan biri de reklam uygulamalarıdır. Farklı ortamlarda satın alma işlemleri, reklam tıklanma sayısı gibi metrikler, gerçek zamanlı teklif verme sistemleriyle birlikte doğru müşteri kitlesine, doğru zamanda ulaşma fırsatı sunmaktadır. Bunun için üretilen veri, sistemden belirlenen metriklerle toplanır, işlenir. Çıktı olarak üretilen değerli veri, yeni tekliflerde kullanılır [12].

Akan verinin kaynağı olan diğer bir alan da sosyal medya uygulamalarıdır. Sosyal medya uygulamaları, sağladığı kolaylık ve avantajlarla yaşamın vazgeçilmez bir parçası haline

gelmiştir. Gelişen teknolojiyle birlikte sadece fotoğraf, video paylaşımı için kullanımın dışında finansal, reklamcılık, pazarlama gibi alanlarda da kullanılan bu uygulamalar büyük miktarda veri üretmektedir. Örneğin; Twitter verilerine göre, günde 500 milyon paylaşım yapılmaktadır. 2,5 milyar aktif kullanıcısı bulunan Facebook şirketi, her saniye ortalama 5 yeni kullanıcı kazanmaktadır. Tüm bu veriler, gerçek zamanlı olarak toplanıp işlenerek gerektiği durumlarda haber siteleri, portallar diğer platformlarla paylaşılmaktadır. Bu başarılı sonuçlar üretmekte önemli bir sorun olmuştur [13].

Nesnelerin interneti ve mobil uygulama verileri, akan veri elde edilen diğer bir alandır. 2020 yıl sonu itibariyle dünyada 5,7 milyar tekil mobil kullanıcı sayısı bulunmaktadır. Web trafiğinin yarısından fazlası da mobil cihazlar tarafından oluşturulmaktadır. Giyilebilir teknolojiler ve mobil cihazlar online servislere geri bildirimde bulunarak iletişim kurmaktadır [14]. Bu iletişim sırasında üretilen her bir veri, akan veri kaynağı olarak işlenip değerlendirilmektedir.

2.1.2. Akan Verinin Temel Özellikleri

Verinin temel özellikleri, verinin temel karakteristiğini ifade etmektedir. Örneğin; ATM uygulamalarından alınan operasyonel takip verisinde ATM ID, ilgili makinenin temel bir özelliğidir [15]. Akan veriler, bazı özellikleriyle statik verilerden ayrışmaktadır. Akan veride, sıralı bir şekilde veriye erişim sağlanırken, statik veride erişim sıralı ya da düzensiz olabilmektedir. Veri hacmindeki büyüklükten dolayı bellek kullanımı, akan veride daha sınırlıdır. Veriyi işlemek için kullanılan algoritmalarda, işlem süresi statik verilere göre daha önemlidir. Verinin varış hızı yüksek olduğundan, algoritma süresindeki olası uzamalar sistem performansına doğrudan etki etmektedir. Veri sisteme geldiği anda değerli olduğundan, veri işleme hızının yüksek olması gerekmektedir. Akan verinin özelliği, veri aktarımı sırasında ortam özelliklerinden etkilenmektedir. Bu yüzden akan verilerin özelliği, sıralı iki veri arasındaki zaman, çeşitlilik ve sürekliliğe göre değişmektedir. Sabit zaman aralığına sahip iki sıralı akan veri paketi aynı uzunluğa sahipse, bu akan veri güçlü periyodik akan veri olarak sınıflandırılır. Zaman aralıklarının güçlü olmadığı iki veri paketi arasındaki zaman periyodunun değişebildiği akan veriler, periyodik olmayan akan veri olarak sınıflandırılır.

Sıralı iki akan veri paketi arasındaki veri boyutunun sürekli sabit kaldığı akan veriler, düzenli akan verilere örnektir. Videolardaki görüntü ve ses akışı düzenli akan veriye örnek olarak

gösterilebilir. Veri miktarının periyodik ya da sabit olmadığı akan veriler, düzensiz akan veri olarak adlandırılır. Düzensiz akan verilerin, akışı ve işlenmesi daha komplekstir.

Akan veriler, veri hacminin büyük olması sebebiyle standart veritabanlarında saklanamaz ve sorgu işlemleriyle kullanılamaz. Bu yüzden klasik çözümler akan veriler için yetersiz kalmaktadır. Akan verinin temel özellikleri, verinin doğal yapısı hakkında bilgi vermektedir. Bu özellikler; verinin detay seviyesi, kirliliği, etki alanı, tutarlılığı açısından izlenmektedir [9].

- **Verinin Detay Seviyesi:** Veri paketi kümesindeki detay seviyesi, veri işlendikten sonra oluşacak çıktı değeri için de mümkün olan ayrıntı seviyesini belirtir. Verideki detay seviyesinin bilinmesi, verilerin destekleyebileceği olası tahminlerin değerlendirilmesine olanak sağlar.
- **Verinin Kirliliği:** Veri kirliliği genellikle kullanıcıların sistem fonksiyonlarını esnetmeye çalıştıklarında oluşmaktadır. Örneğin; veride "kimlik numarası" alanında çoğu zaman kullanıcılar tarafından "11111111111" ya da "ab111111111" standart dışı bilgiler gönderilebilir.
- **Verinin Tutarsızlığı:** Verinin tutarsızlığı, farklı sistemlerde farklı özelliklerin aynı isimle işlenmesi ve aynı özelliğin farklı sistemlerde farklı isimlerle işlenmesi anlamına gelmektedir. Veriler genelde kaynak olarak tek bir uygulama sisteminden alındığında tutarsızlık problemi oluşabilmektedir. Örneğin; bir araba parça otomasyon sisteminde, "BMW" ve "Bavarian Motor Works" aynı markayı temsil etmektedir.

2.2. MAKİNE ÖĞRENMESİ

Makine öğrenmesi, öğrenebilen ve bir problemi o probleme ait veriler üzerinden tahminleyip modelleyebilen algoritmaların genel adıdır. Bu algoritmalar, standart bilgisayar programlarını birebir takip etmek yerine örnek girdileri kullanarak tahminleme ve modelleme yapar. Kurulan modeldeki amaç, en yüksek performansı verebilmektir.

2.2.1. Makine Öğrenmesi Kavramı ve Gelişimi

1960' lı yıllardaki teknolojide yaşanan gelişmelerle birlikte insanların hayatını kolaylaştıran sistemler geliştirilmiştir. Bu sistemler zamanla insan yapısını, hareketlerini örnek alan yapay zeka sistemlerine evrilmeye başlamıştır. Verinin değerli kalabilmesi ve kullanılabilmesi için bu sistemlere öğretilerek yararlı çıktılar alınmaya başlanmıştır.

Makine öğrenmesi, sürekli gelişen ve teknolojik gelişmelerle ön plana çıkan bir kavram olsa da ilk tanımları geçmişte yapılmıştır [32]. Berkeley' in 1950' li yılların başlarında, Giant Brains or Machine That Think (Dev Beyinler veya Düşünen Makine) isimli kitabında; yapısı beyne benzeyen, hesaplama yapabilen, veri işleyebilen büyük makinelerin tanımlarını yapmıştır.

1950' li yıllardan itibaren, istatiksel modellemelerin gelişmesiyle birlikte makine öğrenmesi kavramına ilişkin çalışmalar ilerlemiştir. Alan Turing' in düşünebilen makineler hakkındaki çalışmalarını yayınladığı Computing Machinery and Intelligence isimli çalışmasında tanımladığı Turing Testi, makine öğrenmesi çalışmalarında atılan en büyük adımlardan biridir.

1960 yılından sonra, yapay sinir ağlarının gelişmesiyle birlikte ilk makine öğrenmesi oyunu olarak dama oyunu geliştirilmiştir. 1965 yılında, matematikçi Ivakhnenko teoride kalmış ilk çalışan derin öğrenme ağını oluşturmuştur. Ivakhnenko, modelleri parametrik ve yapısal olarak optimize eden, çok parametreli veri kümelerinin bilgisayar tabanlı matematiksel modelleri için tümevarım algoritması olarak bilinen Veri İşlemede Grup Metodu' nu geliştirmiştir.

1980 yılına doğru, sinir ağları ile ilgilenen Fukushima, görsel kalıpları tanımayı öğrenen Neocognitron' u geliştirmiştir. Neocognitron, doğal dil işleme ve el yazısı karakteri tanıma gibi işlemlerde de kullanılmıştır.

1985 yılında, sinir bilimci Terry Sejnowski öğrenme süreciyle ilgili olarak NetTalk programını oluşturmuştur. Bu program, kelimeleri bir insanın öğrenme sürecine benzer şekilde öğrenmiştir ve metni konuşmaya çevirme sırasında kendini sürekli geliştirmiştir.

1990 yılından sonra denetimli ve denetimsiz makine öğrenmesi kavramları netlik kazanmaya başlamış, veriyi temel alan yaklaşımlar ön plana çıkmaya başlamıştır. Bilim insanları, büyük hacimli verilerin analizi, bu verilerden öğrenebilen programlar geliştirmeye başlamıştır. 1995 yılında, Corinna Cortes ve Vladimir Vapnik destek vektör makinelerini geliştirmiştir. Bir destek vektör makinesi, benzer verileri tanıyıp haritalayan ve görsel sınıflandırma, el yazısı tanıma gibi alanlarda kullanılabilen bir sistemdir.

2006 yılı itibariyle, teknolojideki gelişmelere paralel olarak makine öğrenmesi çalışmalarında da gelişmeler yaşanmıştır. Facebook tarafından geliştirilen DeepFace derin öğrenme sistemi, fotoğraflardaki bireylerin insan doğrulamasını yüksek başarı oranıyla gerçekleştirmiştir. Google' ın makine öğrenmesi ve ağaç tarama tekniklerini kullanan programı AlphaGo, ünlü bir Go oyuncusunu yenmeyi başarmıştır.

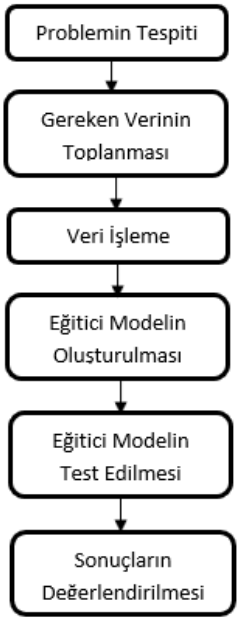
Gün geçtikçe artan veri akışı sebebiyle, verileri doğru bir şekilde değerlendiren makine öğrenimi metodları hayatımıza girmiştir. Amazon ve Netflix gibi uygulamaların öneri yöntemleri, Google'ın kullandığı yüz ve ses tanıma yöntemleri buna örnek olarak gösterilebilir.



Şekil 2.2: Derin öğrenme, makine öğrenimi ve yapay zeka karşılaştırılması.

2.2.2. Makine Öğrenmesi Süreçleri

Makine öğrenmesi süreci, birden fazla adımdan oluşmaktadır. Her adımın, doğru ve tutarlı bir çıktıya ulaşabilmek için belirli koşulları sağlaması gerekmektedir. Makine öğrenmesinde her adım, sıradaki adımın daha kolay şekilde uygulanmasına imkan sağlamaktadır. Makine öğrenmesi süreci Şekil 2.3' te gösterilmiştir.



Şekil 2.3: Makine öğrenmesi süreci adımları.

2.2.2.1. Problemin Tespiti

Makine öğrenmesi sürecinde, problemin tespiti çalışmanın ilk ve en önemli adımıdır. Çalışma hedefinin tam olarak belirlenmesi; çözülecek probleme ait ön analiz, uygulanacak yöntem ve model seçimini etkilemektedir. Problem tespitinde hata yapılması durumunda çalışmanın kalan kısmı da yanlış sonuçlar üretip, karar verme sürecini kötü yönde etkilemektedir.

2.2.2.2. Gereken Verinin Toplanması

Problem tespitinden sonra, belirlenen çalışma hedefine yönelik veri toplama adımına geçilmektedir. Tespit edilen problemin sonucuna ulaşmayı sağlayacak verilerin doğru bir şekilde seçilebilmesi gerekmektedir. Çalışma sonucunun genellenebilmesi için yeterli hacimde veri toplanması gerekmektedir. Veri toplama sürecinde, kaynakların doğru ve güvenilir olması gerekmektedir. Farklı kaynaklardan alınan verilerden oluşan veri kümesinde aynı standartlar korunmalıdır.

2.2.2.3. Veri İşleme

Veri işleme aşamasında, bir önceki adımda toplanan verilerin analiz çalışmaları için hazırlanması gerekmektedir. Veri tiplerinde farklılıklar bulunması, veri eksikliği olması, farklı veri dağılımlarının bulunması gibi etkenler analiz metodunun seçimini etkileyecektir.

Verilerin farklı kaynaklardan elde edildiği durumda, veri kümesinde format olarak aynı standartların sağlanması gerekmektedir. Veri setinde eksiklik bulunması durumunda, önem sırası göz önüne alınarak eksik veriler için analiz uygulanıp veri setinden çıkarılması gerekmektedir. İncelenen veri özellikleri sonucunda, ilerleyen adımlarda sorun oluşturabilecek uyumsuz verilerin kontrolü ve önlenmesi bu aşamada önemlidir.

2.2.2.4. Eğitici Modelin Oluşturulması

Bu aşamada, önceki adımlarda problemin çözümüne göre seçilen analiz metodu uygulanmaktadır. Önceki adımda hazırlanan veri seti üzerinde, makine öğrenmesi modeli oluşturulmaktadır. Bu modelin problem çözümüne yönelik en iyi çıktıyı vermesi beklenmektedir. Modelin başarısı, belirlenen test veri seti üzerinde uygulanıp başarı kriterlerine göre değerlendirilmektedir. Problem tespitinden sonra, belirlenen çalışma hedefine yönelik veri toplama adımına geçilmektedir. Tespit edilen problemin sonucuna ulaşmayı sağlayacak verilerin doğru bir şekilde seçilebilmesi gerekmektedir. Çalışma sonucunun genellenebilmesi için yeterli hacimde veri toplanması gerekmektedir. Veri toplama sürecinde, kaynakların doğru ve güvenilir olması gerekmektedir. Farklı kaynaklardan alınan verilerden oluşan veri kümesinde aynı standartlar uygulanmalıdır.

2.2.2.5. Eğitici Modelin Test Edilmesi ve Değerlendirilmesi

Bu aşamada, önceki adımda oluşturulan model üzerinde test verisi uygulanıp üretilen çıktılar gerçek değerler ile kıyaslanmaktadır. Bu işlem sonunda, kullanılan modelin başarısını gösteren çıktılar elde edilmektedir. Çıktı ölçümleri değerlendirilerek kullanılan modelin, problemin amacına uygun çalışıp çalışmadığı değerlendirilmektedir.

2.3. MAKİNE ÖĞRENMESİ METOTLARI

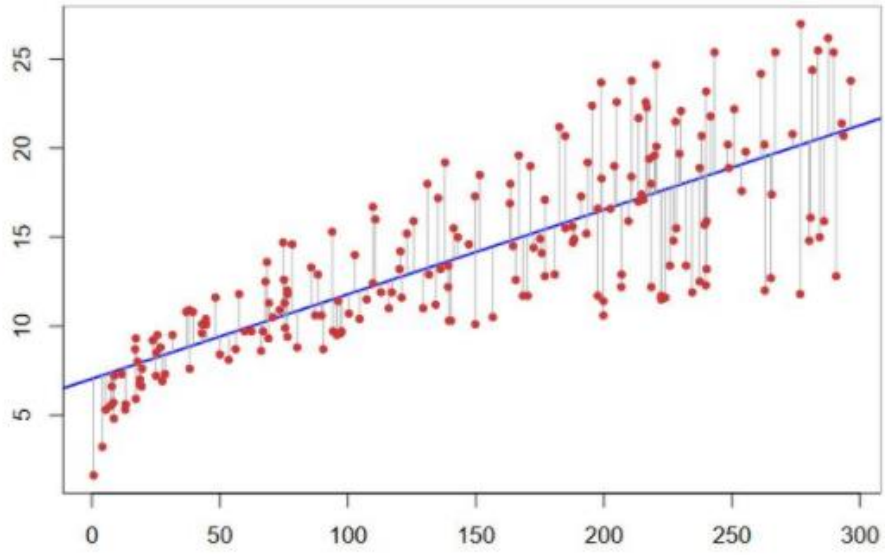
Makine öğrenmesi, öğrenebilen ve bir problemi o probleme ait veriler üzerinden tahminleyip modelleyebilen algoritmaların genel adıdır. Bu algoritmalar, standart bilgisayar programlarını birebir takip etmek yerine örnek girdileri kullanarak tahminleme ve modelleme yapar. Makine öğrenmesi metodları üç ana teknik gruba ayrılmaktadır.

2.3.1. Denetimli Öğrenme

Denetimli öğrenmede, algoritmalar sağlanan etiketli örnekler üzerinden tahminleme yapar. Bu öğrenme modeli, sonucun nasıl görüleceği bilinen durumlarda faydalıdır. Örneğin; şehirlerin son 50 yıla ait nüfuslarını içeren bir küme sağlayıp, 5 yıl sonrası için belirlenen bir şehrin nüfusunun ne olacağının öğrenileceği bir sistemde; veri kümesinde mevcut olan şehir, yıl ve nüfus etiketleri kullanılır. Denetimli öğrenmede doğrusal regresyon, destek vektör makineleri, Naive Bayes, lojistik regresyon, doğrusal diskriminant, k-en yakın komşu ve karar ağaçları bilinen en yaygın algoritmalardır. Denetimli makine öğrenmesinde, geliştirilen algoritmaların işleyiş biçimleri ve modellendiği çevreler farklı olsa da tahminleme ve öğrenmede getirdikleri yaklaşımlar benzerdir. Bu bağlamda tüm algoritmalarda öğrenme için dış mekanizma aracılığıyla, gereken giriş çıkış verilerinin tanımlanması gerekmektedir. Yapılan işlemlerde bu veri yapıları, eğitici set olarak kullanılmaktadır. Eğitici setteki parametreler, problemin tüm özelliklerini kapsayacak şekilde tanımlı olması gerekmektedir. Makine öğrenmesinde elde edilecek başarı, bu eğitici setin kapsayıcı olmasına bağlıdır [17].

Doğrusal Regresyon: Regresyon analizi, bağımlı ve bağımsız değişkenler arasındaki ilişkileri tahminleyen istatiksel bir yöntemdir. Bu yöntemde önemli olan nokta, bağımlı değişkenlerin bağımsız değişkenler tarafından açıklanma oranının yüksek olması gerekir. Yüksek veri miktarının olduğu uygulamalarda başarılı sonuç vermektedir.

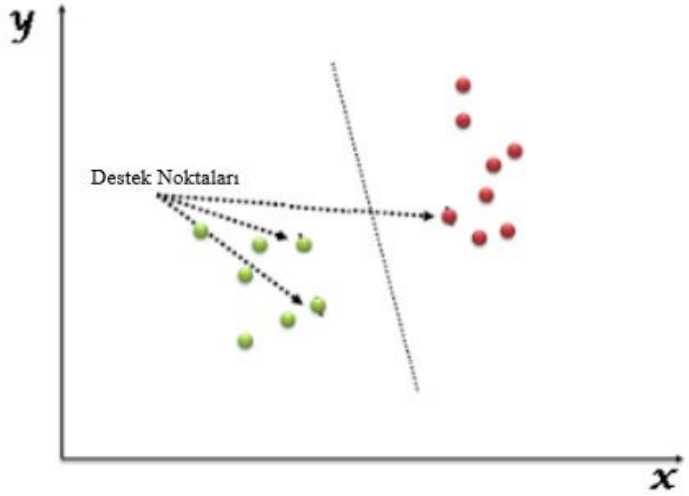
Doğrusal regresyon, değişkenler arasındaki doğrusal ilişkiyi inceleyen istatistik tabanlı bir yöntemdir. Değişkenler arasındaki ilişkiye dayanarak, bir değişkenin değerini diğer değişkenden tahminlemeyi sağlayan bir model oluşturmayı içermektedir. Doğrusal regresyon, bir grup noktaya en uygun olan düzlemi bulmak için kullanılır. En uygun düz çizgiyi kullanarak bağımlı değişkenle, bağımsız değişkenler arasında ilişki kurar. Aşağıdaki Şekil 2.4’ deki mavi çizgi, en uygun düz çizgi olarak adlandırılmaktadır [18].



Şekil 2.4: Doğrusal regresyon örneği.

Destek Vektör Makineleri: Destek vektör makineleri, istatiksel öğrenme teorisine dayanan bir denetimli öğrenme algoritmasıdır. Temel olarak iki sınıfa ait verileri en uygun şekilde birbirinden ayırmak için kullanılmaktadır. Bu işlem için karar sınırları belirlenir. Destek vektör makineleri; yüz tanıma sistemleri, ses analizi gibi birçok sınıflandırma probleminin çözümünde kullanılmaktadır.

Destek vektör makineleri, yüksek boyutlu uzaylarda ve boyut sayısının örneklem sayısından fazla olduğu durumlarda etkilidir. Karar fonksiyonunda bazı eğitim noktaları kullanmasıyla belleği verimli kullanmaktadır. Destek vektör makineleri, veri kümesinin doğrusal olarak ayrılıp ayrılmamasına göre iki parçada değerlendirilmektedir. Doğrusal destek vektör makinelerinde, iki sınıfa ait örneklerin bir karar fonksiyonu kullanılarak birbirinden ayrımı amaçlanmaktadır. Veri kümesini ayıran doğru, karar doğrusu olarak adlandırılmaktadır. Sonsuz sayıda karar doğrusu çizilebilse de, önemli olan en uygun karar doğrusunu belirleyebilmektir. Karar doğrusunun yeni verilere karşı da başarılı olabilmesi için sınır çizgisi, iki sınıfın sınır çizgilerine en yakın uzaklıkta olmalıdır. Sınır çizgisine en yakın olan bu noktalar, destek noktaları olarak adlandırılır [19].



Şekil 2.5: Destek vektör makineleri.

Doğrusal olmayan veri kümesinde destek vektör makineleri, doğrusal bir düzlem çizemez. Bu sebeple çekirdek numaraları kullanılır. Bu yöntem, doğrusal olmayan veri kümelerinde makine öğrenmesi başarısını yüksek oranda arttırmaktadır. En çok kullanılan çekirdek yöntemlerine Polynomial Kernel ve Gaussian RBF Kernel örnek olarak verilebilir.

Naive Bayes Sınıflandırma: Denetimli öğrenme algoritmalarından Bayes sınıflandırma algoritması, istatiksel sınıflandırma tekniğine dayanır. Bayes teoremini temel alan olasılığa dayalı bu yöntemde, test verisi temel alınarak öğrenme gerçekleştirilir ve en yüksek orana sahip örnek ilgili sınıfa dahil edilir. Bayes algoritması, diğer algoritmalara göre daha az hesaplama içerir. Bu sayede giriş ve tahmin sütunları arasındaki ilişki daha hızlı bulunmaktadır. Bu yöntemin daha iyi anlaşılabilmesi için sık kullanılan bir örnek aşağıdaki gibidir:

“ Tenis maçının oynanıp oynanmadığı bilinmektedir. Bu bilgiyle birlikte nem, rüzgar, sıcaklık gibi çevresel faktörler de bilinmektedir. Bu bilgiler kullanılarak tenis maçının oynanıp oynanmayacağına karar verilebilmektedir. Algoritma, kriterler ile ilgili olan bu bilgileri kullanarak tenis maçının yapılıp yapılmaması hakkında bilgi verir.”

Naive Bayes algoritması, işlediği kümedeki verilerin sınıfını bulmayı amaçlayan bir yöntemdir. Sisteme ilk olarak eğitim verileri gönderilir, sistem bu verileri öğrendikten sonra yeni veriler sisteme gönderilerek sınıflandırılır. Bu noktada, öğretilen veri hacminin büyük olması önem taşımaktadır [20].

Karar Ağaçları: Karar ağaçları, sınıflandırma problemlerinde karmaşık veri setlerinde kullanılabilen ağaç tabanlı algoritmalardan biridir. Karar ağacında ilk hücre kök hücredir, her gözlem kökteki koşula göre sınıflandırılmaktadır. Düğümler, kök hücrelerinin altında bulunur. Düğümler, gözlemlerin sınıflandırılmasında kullanılır. Modelin karmaşıklığı, düğüm sayısının artmasına paralel olarak artar. Karar ağacının en altında yaprak diye adlandırılan, sonucu veren kısımlar bulunur.

Karar ağacında önemli olan nokta, kök hücre seçiminde seçilen hücrenin veri setini detaylı açıklaması gerekmektedir. Karar ağacında temel amaç; veri kümesini kümeleme algoritmasıyla gruplanarak bölünüp, grubun tüm elemanları aynı sınıf etiketinde olana kadar işlemin devam etmesidir. Karar ağacı, temel olarak dal ve yapraklardan oluşur. Bu yapıdaki karar düğümleri, gerçekleşecek olan testleri ifade eder. Bu testlerin sonucunda, ağaç dallara ayrışmaktadır. Her düğümde test ve dallanma işlemi ardışık olarak devam etmektedir. Dalın ucunda sınıflandırma işlemi tamamlandıysa, karar düğümünü oluşmuş demektir [21].

Karar ağacı algoritmalarından M5P, C4.5, ID3, J48 en çok tercih edilen algoritmalardır. Bu algoritmalar, hem ayrık hem sürekli veride çalışabilmektedir. C4.5 ile J48 algoritmaları benzer özellikler taşımaktadır. J48 algoritmasında entropi hesaplanıp karar ağacı çıkarıldıktan sonra, öznitelikler için bilgi kazancı hesaplanarak veri kümesi bölünmektedir. En çok kazanç sağlayacak öznitelik karar vermede kullanılır. C4.5 algoritması; entropi indirgemeye dayanarak veri özniteliğinin, veri kümesini alt kümelere ayırarak karar vermede kullanabileceğini temel alır [21].

K-En Yakın Komşu: K-en yakın komşu algoritması, sınıflandırma ve regresyon için kullanılan denetimli öğrenme algoritmalarında biridir. Eğitim ve test aşamaları hemen hemen aynı olduğundan geniş veri setlerini işlemek için ideal bir algoritma değildir. Bu yöntemde temelde, yeni noktanın en yakınındaki noktalar aranmaktadır. K, bilinmeyen noktaya en yakın komşuların miktarını ifade eder. Sonuçların tahmini için, algoritmanın k miktarı seçilir. Yakınlık, Öklid uzaklığı ile tanımlanır. Sonrasında, bilinmeyen örneklem k en yakın komşudaki benzerliği en yüksek sınıfa atanır [22].

2.3.2. Denetimsiz Öğrenme

Denetimsiz öğrenme, veri setinden elde edilecek çıktıyla ilgili bilgi sahibi olunmadığı durumlarda kullanılan öğrenme yaklaşımıdır. Denetimsiz öğrenmede, sisteme veri seti hakkında bilgi verilmeden işlenen veri setinden sonuç üretilir. Denetimsiz öğrenme, olasılık yoğunluklu işlevleri modellemeye ve verilerdeki anormallikleri bulmaya yardımcı olur. Denetimsiz öğrenme yöntemlerinde kümeleme yöntemi ön plana çıkmaktadır [23].

Hiyerarşik Kümeleme: Hiyerarşik kümeleme yönteminde, veri kümesindeki farklı veri noktaları arasındaki benzerlik temel alınarak kümeleme işlemi yapılır. Veri noktalarının çeşitli özellikleri incelenerek benzer bulunanlar birlikte gruplanır. Parçadan bütüne ve bütünden parçaya olmak üzere iki farklı versiyonu bulunmaktadır.

Parçadan bütüne kümeleme yönteminde, veri setindeki tüm veriler için kümeler oluşturulur. Birbirine mesafe olarak yakın olan kümeler birleştirilerek yeni küme oluşturulur. Sistem kararlı hale gelene kadar bu işlem devam eder. Bütünden parçaya yönteminde, veri setindeki tüm veriler için tek küme oluşturulur. Sonrasında bu küme parçalanarak kümeleme işlemi yapılmaktadır [24].

Bölümlemeli Kümeleme: Bölümlemeli kümeleme yönteminde, k giriş parametresi alınarak m adet nesne, k adet kümeye ayrılır. Bu yöntemde, tek seviyeli kümeleri saptayan işlemler gerçekleşir. Bölümlemeli kümeleme yöntemleri, verimli olması ve uygulamasının kolaylığıyla iyi sonuç üretirler. Bu yöntemlerde k-means kümeleme yöntemi ön plana çıkmaktadır.

K-means yönteminde, k değeri belirlendikten sonra k tane merkez noktası seçilir. Veri kümesindeki her bir veri için, belirlenen merkez noktaları arasındaki uzaklık hesaplanarak, veri en yakın merkez noktasının olduğu kümeye atanır. Her küme için tekrardan bir merkez noktası seçilerek kümeleme işlemi yapılır. Sistem kararlı yapıya ulaşıncaya kadar bu işlemler devam eder. K-means yöntemindeki mekanizma, her verinin yalnızca bir kümeye ait olmasına izin verir [25].

Yoğunluk Tabanlı Kümeleme: Bu kümeleme yönteminde, küme seçiminde uzaklık yerine verilerin yoğunluğu temel alınmaktadır. Küme yoğunluğunun az olduğu alanlarda küme sınırlarını oluşturan veriler ya da gürültülü veriler bulunmaktadır. Yoğunluk tabanlı kümeleme yöntemleri, veri setinin içerdiği farklı şekillerdeki kümeleri tespit etmek için

kullanılmaktadır. Bu yöntemlerde temel alınan yoğunluk, veri uzayındaki veri noktalarının yoğunlaştığı kısımlarda kullanılan belirli ölçüm parametrelerine göre belirlenmektedir [27].

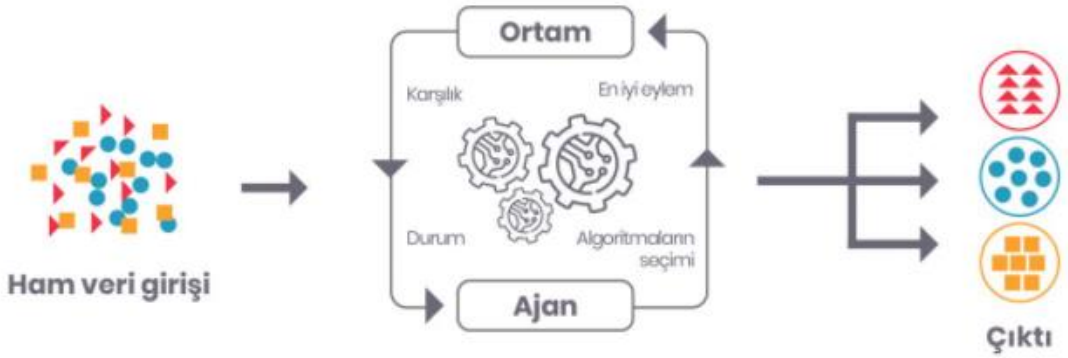
Yoğunluk tabanlı kümeleme yöntemleri gürültülü verilerle çalışırken etkilidir. Küme sayısının sisteme verilmesine ihtiyaç yoktur. Yoğunluk tabanlı kümeleme algoritmalarına örnek olarak DBSCAN(Density-Based Spatial Clustering of Applications with Noise) algoritması örnek verilebilir. Bu algoritmada, veri noktalarının çok boyutlu uzaydaki komşulukları ortaya çıkarılarak uzaysal verilerin analizi yapılır. Temel kavramlar olarak, çekirdek nesne, MinPts, Eps, direkt erişilebilir yoğunluklu nokta, erişilebilir yoğunluklu nokta öne çıkmaktadır. Algoritmada, MinPts ve Eps giriş parametresi olarak alınır. Kontrol edilen nesne, bir kümeye dahil olmuşsa sıradaki nesneye geçilerek işleme devam edilir. Eğer nesne kümelenmemişse, Eps komşulukları bulunur. Bulunan komşu sayısı, MinPts değerinden büyükse, bu nesne ve komşuları yeni bir küme olarak değerlendirilir. Kümelenmemiş her bir komşu için Eps komşulukları bulunarak, kümeye dahil edilir. Komşuluk bulma işlemi, algoritmanın en fazla işlem gören kısmıdır ve algoritmanın performansını önemli ölçüde etkilemektedir [26].

Izgara Tabanlı Kümeleme: Izgara tabanlı kümeleme yöntemi, veri setindeki veri dağılımından bağımsızdır. Veriyi ızgara adı verilen hücreler yardımıyla kümeler. Veri setindeki nesneler, uygun ızgara hücresine atanarak, hücrelerin yoğunluğu hesaplanır. Yoğunluğu tanımlanan eşik değerinin altında kalan hücreler elenir. Algoritma performansı ve kümeleme süresi açısından hızlı bir yöntemdir. STING(Statistical Information Grid)[28], CLIQUE(Clustering High-Dimensional Space)[28] ve WaveCluster[29] örnek olarak verilebilir.

STING algoritmasında uzaysal alan dikdörtgen hücrelere bölünmüştür. Farklı çözünürlüğe karşılık gelen bu tür dikdörtgen hücrelerin birkaç farklı seviyesi bulunmaktadır. Bu hücreler hiyerarşik bir yapı oluşturur. Yüksek seviyedeki bulunan her hücre, bir sonraki alt seviyeden bir dizi hücre oluşturmak için bölünür. Her hücrenin istatistiksel bilgileri önceden hesaplanır, saklanır ve sorguları yanıtlamak için kullanılır [28].

2.3.3. Takviyeli Öğrenme

Takviyeli öğrenme, insan davranışlarından esinlenen ve bulunduğu ortamı algılayıp kendi kendine karar alabilen bir sistemin doğru karar alabilmesi için izlenmesi gereken adımlarla ilgilenen makine öğrenmesi yöntemidir. Bu öğrenme yöntemi, klasik öğrenme yöntemleri gibi ön eğitim verisine ihtiyaç duymadan ve kesin yöntemlerin kullanılamadığı durumlarda kullanılmaktadır [40].



Şekil 2.6: Takviyeli öğrenme yöntemi.

Takviyeli öğrenme ajanı, ortamla etkileşime girerek ortaya çıkan sonuçlardan gözlemleme ile öğrenir. Öğrenme süreci, neden sonuç ilişkisine dayanan bu yöntemde ajan, aşağıdaki adımları uygulamaktadır:

- Ajan, giriş adı verilen başlangıç durumunu gözlemlemektedir.
- Politika olarak adlandırılan karar verme işleviyle, gerçekleştirilecek adım belirlenir.
- Belirlenen eylem gerçekleştirilir.
- Eylem sonucuna göre ajan ortamdan bir ödül alır.
- Eylem ve ödül hakkındaki bilgiler kaydedilir.

Takviyeli öğrenme kullanılarak çeşitli problemler çözülebilir. Oyun sistemleri, robotik kontrol sistemleri takviyeli öğrenme ajanları kullanılarak eğitilebilir.

2.4. LİTERATÜRDEKİ ÇALIŞMALAR

Günümüzde teknolojideki gelişmelerle birlikte üretilen veri miktarı gittikçe artmakta ve büyük miktarda veri üretilmektedir. Üretilen bu canlı verinin analizi, yeni bir kavram olsa da bu konuda da çalışmalar mevcuttur. İlk çalışmalarda akan verinin karakteristiğini belirleme, farklı uygulama alanlarında akan verinin değerlendirilmesi ve karşılaşılan sorunlara karşı uygulanan çözüm yöntemlerinin başarısı değerlendirilmiştir. 2005 yılındaki [8] çalışmada akan verinin karakteristik tanımları ve akan verinin ihtiyaçlarına yönelik geliştirilecek çalışmalar hakkında bilgi verilmiştir.

Akan veri analiz yöntemlerinde karşılaşılan en büyük problemlerden biri, akan veriyi oluşturan verilerin çoğunluğunun etiketsiz olmasıdır. [33] çalışmasında etiketsiz olan veriler için öğrenim kümesi temelinde etiketleme yapılarak çalışmaya dahil edilmiştir. K-means kümeleme algoritması kullanılarak öğrenim kümesi etiketli verilerdeki etiket sayısı kadar kümeye ayrılıp, etiketsiz olan verilere bulundukları kümelerde etiketleme yapılmıştır. Çalışma sonucunda %11 etiketli, %89 etiketsiz verilerin bulunduğu öğrenim kümesinde %89 oranında doğruluk sağlanmıştır.

Akan veri analizi konusunda yapılan önemli çalışmalardan birisi de [16] çalışmasıdır. Bu çalışmada sınıflandırma, regresyon ve kümeleme algoritmaları, akan veri analiz problemleri için kullanılmıştır. Akan verinin anlık izlenmesi ve aynı veri üzerinde birden fazla algoritmayla çalışma olanağı sağlanmıştır. Uygulamada sağlanan kullanıcı arayüzü ile akan veri üzerinde farklı algoritmalar ile analiz çalışmasına imkan sağlanmıştır.

[34] çalışmasında geliştirilen yöntem veri toplama, değerlendirme, analiz ve işleme olmak üzere dört adımdan oluşmaktadır. Veri akışını değerlendirmek üzere farklı hesaplama işlemleri yapılmaktadır. Bu hesaplamalar, planlama ve değerlendirme adımlarında kullanılmaktadır. Her bir veri mesajı için boyut, işlem zamanı, bant genişliği gibi veriler ölçümlenmektedir. Veri kümesinin hafıza kullanımı ve işlemci tüketimi değerleri karşılaştırılarak mevcut sistemlere göre tasarlanan yeni sistemin performansı değerlendirilmektedir.

[35] çalışmasında akan veri kümesi için veri, analiz, operasyon ve genel küme seviyesi olmak üzere dört özellik çıkarımıyla makine öğrenmesi yöntemleri kullanılmıştır. Birden fazla sorgu ile, değişken iş yükü ve veri kümesiyle tasarlanan model eğitilmiştir. Eğitilen model için, yeni

gelecek iş yüklerinde tüketilecek kaynak miktarı tahminlenmiştir. Yapılan testlerde daha az kaynak tüketimi ve işlemci yükü tespit edilmiştir.

[36] çalışmasında sisteme gelen akan verinin dağıtımı ele alınmıştır. Oluşturulan dağıtım planlayıcı bir modül ile sisteme gelen akan verinin hızı dinlenerek yapılacak işlemler için fonksiyonlar oluşturulmuştur. Bu fonksiyonlar ile veri yükü ve kaynak tahmini yapılarak ölçekleme yapılmıştır. Sistemden alınan çıktının birim zamana göre ölçümlenmesiyle sistemin verimi değerlendirilmiştir.

[37] çalışmasında akan verinin istasyonlar arasındaki trafiğini ve istasyonlardaki yük dengelerini temel alarak bir zamanlama algoritması geliştirilmiştir. Hafıza kullanımı, tahminlenen ve gerçekte kullanılan işlemci yükü, gecikme ve istasyonlar arasındaki trafik yoğunluğu test edilerek verimlilik değerlendirilmiştir.

Akan veri analiziyle ilgili yapılan diğer bir çalışmada [38] karar ağacı yönteminin öğrenim aşamasında değişiklikler yapılarak akan veriye göre çalışması amaçlanmıştır. Öğrenim sürecinde hesaplamada kullanılan endeks değeri, her bir yeni veri örneği için yaprak düğüm yaratmadaki etkisiyle bağlantılı olarak hesaplanmıştır. Bu yöntemde endeks değeri, her bir veri örneği için yeniden hesaplanmadığından akan verinin analizi klasik karar ağacı algoritmalarına göre %20 oranında hızlandığı görülmüştür.

Akan veri işleme çalışmalarındaki önemli noktalardan biri de kaynak verimliliğinin artırılmasıdır. [39] çalışmasında hızlı ve dağıtık akan veriyi işleyen, tahmine dayalı bir model tasarlanmıştır. Çalışma zamanı istatislikleri kullanılarak belirli bir çözüm zamanı için, modelin ortalama veri demeti işleme süresini tahminleyen topolojiye duyarlı bir algoritma kullanılmıştır. Denetimli öğrenim algoritması kullanılarak bellek, iş yükü, işlemci kullanımı yükü gibi metriklerle modelin öğrenim aşamasından geçmesi sağlanmıştır. Kelime sayısı iş yüküyle yapılan testlerde oluşturulan topolojinin %83,7 oranında başarılı olduğu değerlendirilmiştir.

3. MALZEME VE YÖNTEM

Akan veri işleyen sistemler, genellikle Apache Kafka, Apache Spark, Apache Storm, Massive Online Analysis(MOA) üzerinde geliştirilmektedir. Bu tez çalışmasında Java programlama diliyle, MOA kullanılarak akan verinin işlenmesi ve ölçeklenmek üzere değerlendirilmesi ele alınacaktır. Kullanılan bu araçların yapısal özellikleri ve işleyişi aşağıda açıklanmıştır.

Bu bölümde makine öğrenmesi metotları, akan verinin nasıl elde edildiği, düzenlendiği ve sonrasında verinin değerlendirilmesinde kullanılan MOA programının nasıl çalıştığı anlatılmıştır.

3.1. KULLANILAN YÖNTEMLER

Bu çalışmanın gerçekleştirilmesinde yukarıda anlatılan denetimsiz öğrenme yöntemlerinden, kümeleme yöntemi kullanılmıştır. Bu yöntemle akan verinin işlenmesi ve sistemdeki akan veri yüküne bağlı olarak meydana gelen değişimler incelenmiştir.

3.1.1. CluStream-Kmeans++ Kümeleme Algoritması

Bu çalışmada kümeleme yöntemlerinden CluStream algoritmasının özelleştirilmesiyle elde edilen CluStream-Kmeans++ algoritması kullanılmıştır. Akan veri kümeleme algoritmaları, akan veri setlerine karşı klasik kümeleme yöntemlerinin bir adaptasyonu olarak karşımıza çıkmaktadır. CluStream kümeleme algoritması, çevrimiçi ve çevrimdışı bileşenler olarak iki adımda karakterize edilebilir. Çevrimiçi bileşen olarak veri soyutlama, çevrimdışı bileşen olarak veri kümeleme ifade edilebilir. Çevrimiçi bileşen, daha sonra çevrimdışı bileşen adımında kullanılmak üzere yalnızca belirli veri yapılarındaki ilgili bilgilerin çıkarılmasıyla ilgilenir.

CluStream algoritması, akan veri setini etkili bir şekilde kümelemek için kullanılabilecek bir algoritmadır. Kümeleme problemi, veri setini bir mesafe ölçüsü kullanarak bir veya daha fazla benzer nesne grubuna bölmeyi amaçlamaktadır. Akan veri kümeleme, bellek sınırlaması nedeniyle kullanılan tüm bilgileri koruyamadığından ve geçmiş bilgileri yeniden işleyemediğinden, algoritma alınan verilerin küçük bir özetini saklamak zorundadır. CluStream, çevrimiçi ve çevrimdışı bileşenler kullanarak bu sorunu verimli bir şekilde aşmaktadır. Çevrimiçi bileşen, verileri tek geçişte analiz eder ve özet istatistikleri depolarken, çevrimdışı bileşen, kullanıcı tarafından zaman içindeki küme gelişimini sorgulamak için

kullanılabilir. Bu özet istatistikleri korumak için CluStream, bir mikro kümeleme tekniği kullanır. Bu mikro kümeler, daha yüksek seviyeli makro kümeler oluşturmak için çevrimdışı bileşen tarafından daha fazla kullanılır [29].

Her algoritmada çalışmak için belirli veri yapıları kullanılır. CluStream algoritmasında, çevrimiçi bileşende büyük miktarda veriyi özetlemek için Küme Özellik Vektörü (Cluster Feature Vector) adı verilen belirli veri yapıları kullanılır. Bu veri yapısı ilk olarak BIRCH [30] algoritmasında kullanılmıştır. Küme özellik vektörleri, veri nesnelerinin sayısı (N), veri nesnelerinin doğrusal toplamı (LS) ve veri nesnelerinin karelerinin toplamından (SS) oluşur. Yeni veri örneği, n değişkeni ile ifade edilmiştir. LS ve SS n boyutlu dizilerdir. Küme özellik vektörü, vektöre nokta eklemek veya vektörleri birleştirmek için arttırım ve toplamsallık özelliklerini korur. CluStream algoritmasında, küme özellik vektörleri, ek zaman bileşenine sahip mikro kümeler olarak adlandırılırlar. Zaman bileşenleri, zaman damgalarının(LST) ve zaman damgalarının karelerinin(SST) toplamıdır. Arttırım ve toplamsallık özellikleri aşağıdaki gibidir:

Arttırım: Yeni gelen bir veri noktası, küme özellik vektörüne vektörün aşağıdaki bilgilerini güncelleyerek eklenebilir.

$$LS \leftarrow LS + n \tag{3.1}$$

$$SS \leftarrow SS + n^2 \tag{3.2}$$

$$N \leftarrow N + 1 \tag{3.3}$$

Toplamsallık: Yeni bir küme özellik vektörü, iki ayrık küme özellik vektörünün aşağıdaki şekilde birleştirilmesiyle oluşturulabilir.

$$N = N_1 + N_2 \tag{3.4}$$

$$LS = LS_1 + LS_2 \tag{3.5}$$

$$SS = SS_1 + SS_2 \tag{3.6}$$

Yukarıdaki bilgiler kullanılarak merkez ve yarıçap bilgileri aşağıdaki eşitlik 3.7 ile hesaplanır.

$$\text{Merkez} = LS/N$$

$$\text{Yarıçap} = \sqrt{\frac{SS}{N} - \left(\frac{LS}{N}\right)^2} \tag{3.7}$$

Yukarıda belirtilen özellikler, CluStream algoritmasında, mikro kümeleri birleştirmek veya oluşturmak için kullanılır. Oluşturma veya birleştirme kararı, en yakın kümeye olan uzaklığına göre bir sınır faktörüne dayanmaktadır. Yeni bir mikro küme oluşturma

durumunda, istenen miktarda küme sağlamak için küme sayısının bir azaltılması gerekir. Bu işlem, bellekten tasarruf etmek için yapılır ve eski kümeleri kaldırarak veya birleştirerek gerçekleştirilebilir. Bu şekilde CluStream, bir zaman kapsamı boyunca büyük miktarda baskın mikro kümelerin istatistiksel bilgilerini koruyabilir. Ayrıca bu mikro kümeler, daha yüksek seviyeli makro kümeler oluşturmak için algoritmanın çevrimdışı bir aşamasında kullanılır [29].

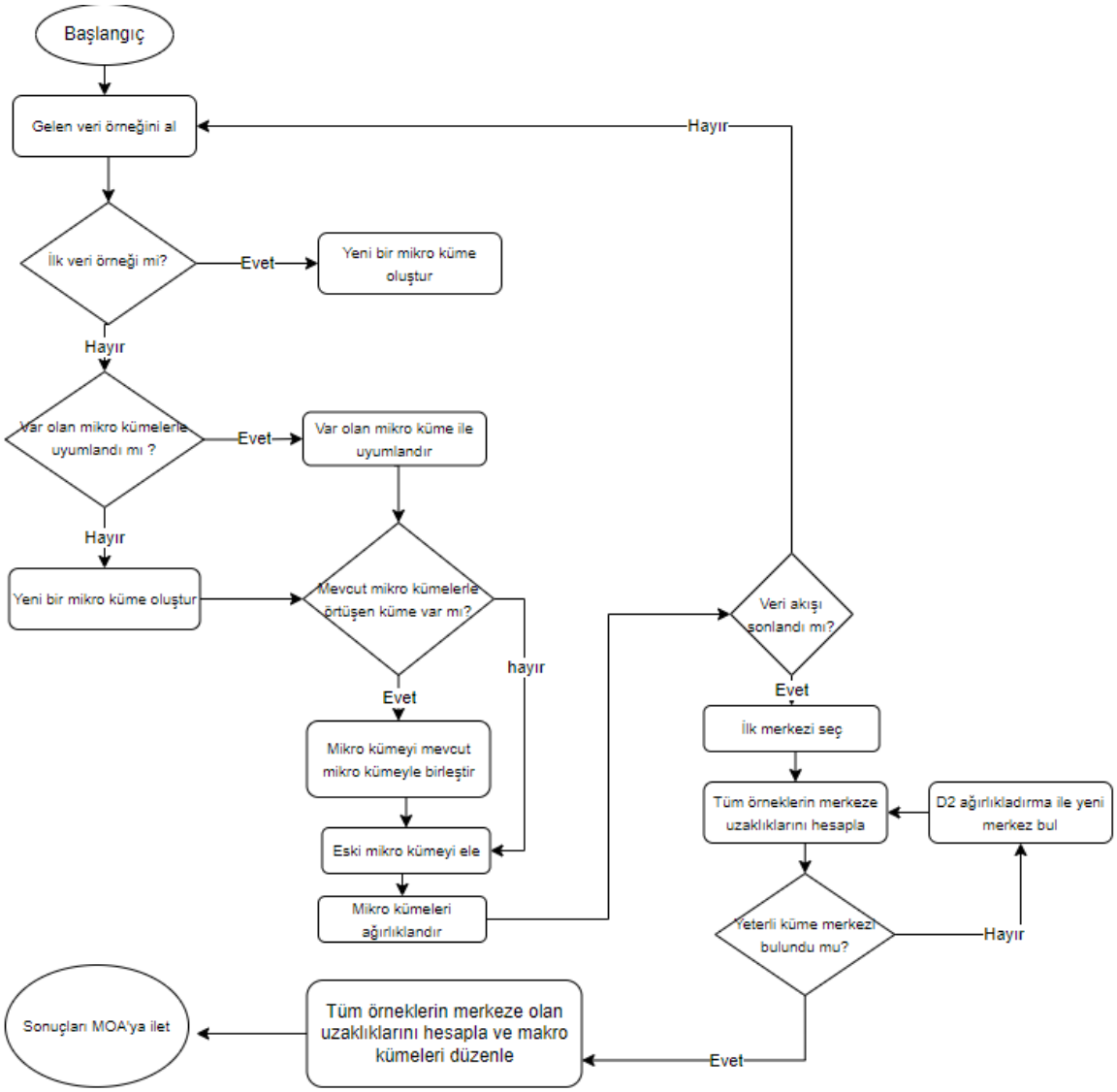
Algoritmanın çevrimdışı aşaması, girdi olarak bir zaman kapsamı olarak h değeri ve bir dizi yüksek seviyeli k tane küme alır. K değeri, son kümelerin ayrıntı düzeyini, h değeri algoritma tarafından ne kadar geçmişin kapsanması gerektiğini belirler. Daha düşük h değerleri, daha yeni bilgileri alır ve daha yüksek k değeri, daha ayrıntılı kümeler oluşturur. Bu aşamada makro kümeler, klasik CluStream algoritmasında kullanılan k-means [29] algoritması yerine, daha hızlı ve performanslı sonuç üreten k-means++ algoritmasının kullanılmasıyla belirlenir. K-means++ algoritmasının detayları sonraki kısımda verilmiştir.

3.1.2. K-means++ Kümeleme Algoritması

K-means++ [48] algoritması, CluStream algoritmasındaki çevrimdışı aşamada yukarıdaki problemlere çözüm getirebilecek yeni bir makro kümeleme yöntemi olarak kullanılmıştır. K-means++ algoritması, D^2 ağırlandırma olarak bilinen bir teknik kullanarak küme merkezlerini belirler. D^2 ağırlıklandırma, küme merkezine en yakın noktanın bu merkeze uzaklığını ifade eder. K-means++ algoritmasını, k-means algoritmasından ayıran en önemli kısım başlangıç kümelerinin belirlendiği kısımdır. Bu yöntemde; ilk seçilen noktaya göre belirlenecek merkez, önceki belirlenen en yakın merkeze uzaklığının karesiyle orantılı bir nokta seçilerek belirlenir. K-means algoritmasında; ilk aşamada k tane küme merkezi belirlenerek çalışmaya başlanırken, k-means++ yönteminde bir adet küme merkezi belirlenerek işleme başlanır ve sonraki küme merkezlerinin belirlenmesinde ilk küme merkezi ve eşitlik 3.8' deki olasılık dağılımları kullanılır. Eşitlik 3.8' de x', seçilecek bir sonraki küme merkezini temsil eden, mevcut kümeye en yakın nokta; x ise Y adet örnek içeren veri kümesindeki her bir veriyi temsil eder.

$$P = \frac{D(x')^2}{\sum_{X \in Y} D(x)^2} \tag{3.8}$$

Veri işleme akışı Şekil 3.1' deki algoritmaya göre yapılmaktadır.



Şekil 3.1: Geliştirilen CluStream-Kmeans++ algoritma akışı.

Geliştirilen bu kısımla birlikte, tezin geri kalan kısmında kullanılan algoritmadan CluStream-Kmeans++ olarak bahsedilecektir.

3.1.3. Test Sürecinde Kullanılan Algoritmalar

Bu kısımda tez çalışmasında geliştirilen CluStream-Kmeans++ algoritmasıyla birlikte, test aşamasındaki doğruluk değerlendirmede kullanılan literatürdeki akan veri algoritmaları anlatılmıştır.

3.1.3.1. ClusTree Algoritması

ClusTree algoritması [45], çevrimiçi ve çevrimdışı olmak üzere iki aşamadan oluşan veri özetleme yöntemi olarak mikro kümeleri kullanan akan veri kümeleme algoritmalarından biridir. Çevrimiçi kısımda bulunan mikro kümeler, hiyerarşik kümeleme prensibiyle birleştirilmektedir. Küme birleşimi ve küme ayırma işlemleri sürekli olarak yapılmaktadır. Algoritmanın işlediği bir veri örneği, ağaç biçimindeki veri yapısında yönlenerek en uygun yaprağa ulaştırılmaktadır. Veri örneğine uygun bir yaprak bulunmuyorsa, veri örneği için yeni bir küme oluşturularak ağaç yapısına eklenmektedir.

ClusTree algoritması, çalışma hızını işlediği akan verinin geliş hızına göre ayarlayıp herhangi bir anda kümeleme yapabilmektedir. Esnek kümeleme özelliği, herhangi bir anda kümeleme sonucu üretebilme ve bu sonuçları gözlemleyip düzeltme imkanı da vermektedir.

3.1.3.2. DenStream Algoritması

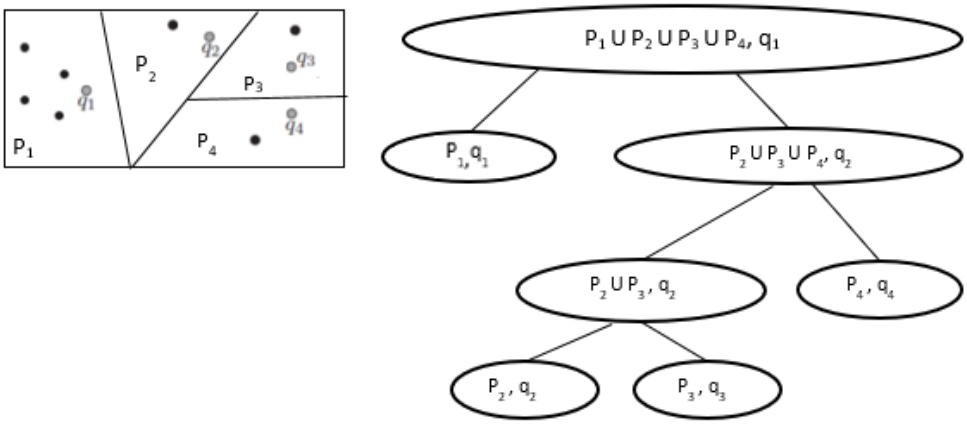
DenStream [46] algoritması, ClusTree algoritmasına benzer olarak çevrimiçi ve çevrimdışı olmak üzere iki kısımdan oluşmaktadır. İlk kümeleme aşaması olan çevrimiçi kümeleme başlamadan önce başlangıç kümeleme işleminin tamamlanmış olması gerekmektedir. ClusTree' den ayrı olarak başlangıç kümeleme işleminde, yoğunluk tabanlı kümeleme yöntemi olan DBSCAN kullanılmaktadır.

DenStream algoritması, başlangıç kümeleme işlemini tamamladıktan sonra çevrimiçi kümeleme adımını gerçekleştirir. Gelen akan veri örneği için, en yakındaki mikro küme aranır ve veri örneği o kümeye atanır. Eğer veri örneği herhangi bir mikro kümeye yakın değilse, bu veri örneği için yeni bir mikro küme oluşturulur. Çevrimiçi kümeleme kısmında belirlenen mikro kümeler ile çevrimdışı kümeleme aşamasında gerçek kümeleme işlemi yapılmaktadır. DenStream algoritması, hesaplamalarında akan veri içerisinde en son gelen veriye daha çok önem vermektedir. Bu ayrımı, eşitlik 3.9' daki hesaplamayla yapmaktadır. Eşitlik 3.9' da a değeri, etken değerini, r ise veri örneğinin geliş süresini ifade etmektedir.

$$f(r) = 2^{-a.r} \tag{3.9}$$

3.1.3.3. StreamKM++ Algoritması

StreamKM++[47] algoritması, akan veri özetleme işleminde yukarıda bahsedilen algoritmalardan farklı olarak mikro küme yerine çekirdek ağacı yapısını kullanmaktadır. Bu yapıda veriler, ikili ağaç yapısında hiyerarşik olarak kümelenir. StreamKM++ algoritması, Öklid uzaklığı ve k-means tabanlı bir akan veri kümeleme algoritmasıdır.



Şekil 3.2: StreamKM++ algoritma akışı.

Şekil 3.2' de aktarıldığı üzere, StreamKM++ algoritması, gelen veri örneklerini ilk olarak bir küme gibi işlemektedir. Sonraki adımda birbiriyle benzerliği olmayan gruplar hiyerarşik olarak ayrılarak asıl kümeler belirlenir.

3.2. UYGULAMA

Massive Online Analysis (MOA), açık kaynak kodlu veri işleme uygulamasıdır. Veriyi işleyen ana temel sistem, Java programlama dili ile geliştirilmiştir. Makine öğrenimi algoritmalarının sınıflandırma, regresyon, kümeleme, aykırı değer algılama, kavram sapması algılama ve öneri sistemlerini içeren bir koleksiyonunu ve değerlendirme araçlarını içerir.

3.2.1. MOA Kurulumu

MOA; Java tabanlı geliştirmesi yapılan, akademik çalışmalarda tercih edilen, açık kaynak kodlu geliştirmeler için uygun olan, çevrimiçi ve çevrimdışı olarak makine öğrenmesi algoritmaları ile verileri işleyebilen bir programdır. MOA; akan verinin işlenmesi, veri üzerinde makine öğrenmesi yöntemlerinin istatistiksel olarak değerlendirilmesi, akan verinin ve bu veriden öğrenimle çıkarılan modelin görsel olarak izlenmesi gibi adımları destekler.

Kurulum için, [16] çalışmasındaki adresten uygulamanın kaynak kodu indirilir. Kaynak kod indirildikten sonra aşağıdaki kod bloğu projeye maven bağımlılığı olarak eklenir.

```
<dependency>
  <groupId>nz.ac.waikato.cms.moa</groupId>
  <artifactId>moa</artifactId>
  <version>2019.4.0</version>
 </dependency>
```

Şekil 3.3: MOA proje bağımlılığı.

3.2.2. MOA' nın Kullanımı

MOA programının indirme ve kurulum işlemleri tamamlandıktan sonra, programa verinin aktarılması gerekmektedir. Program, komut satırından Şekil 3.4' teki kod bloğuyla başlatılır. MOA, iki temel kısımdan oluşmaktadır.

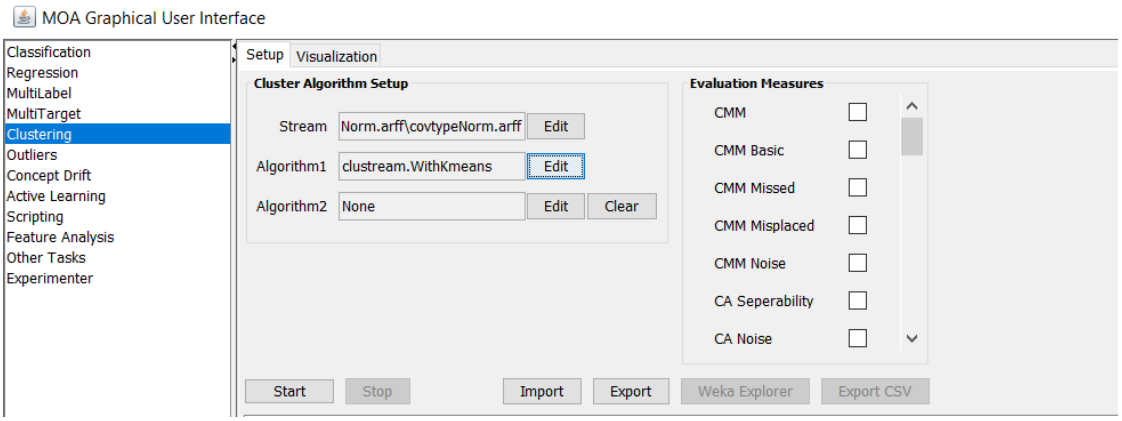
- **Kurulum(Setup)**: İşlenecek veri akışının belirlendiği, veriyi değerlendirirken kullanılacak algoritma ve değerlendirme metriklerinin bulunduğu kısımdır.
- **Görselleştirme(Visualization)**: Çalışma sonrasında oluşacak çıktının görsel olarak izlenebildiği kısımdır.

```
java -cp moa.jar -javaagent:sizeofag.jar moa.gui.GUI
```

Şekil 3.4: MOA uygulamasının başlatılması.

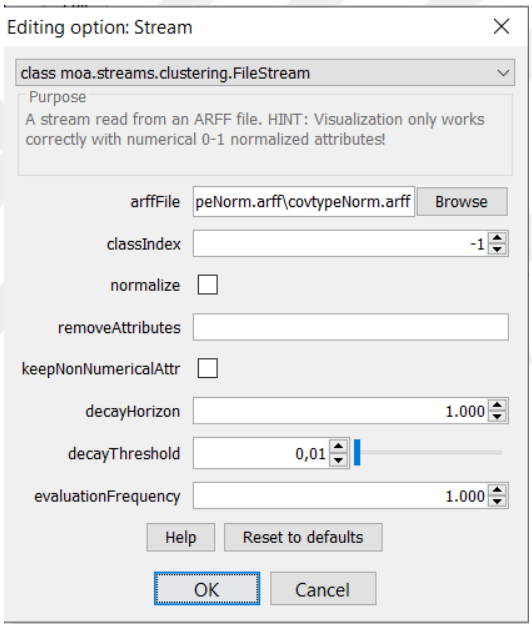
3.2.2.1. Kurulum Paneli (Setup Panel)

Kurulum panelinde, ilk olarak uygulamada işlenecek veri akışı oluşturulur. Bu işlem için, Şekil 3.5' teki "Stream" seçeneği kullanılır.



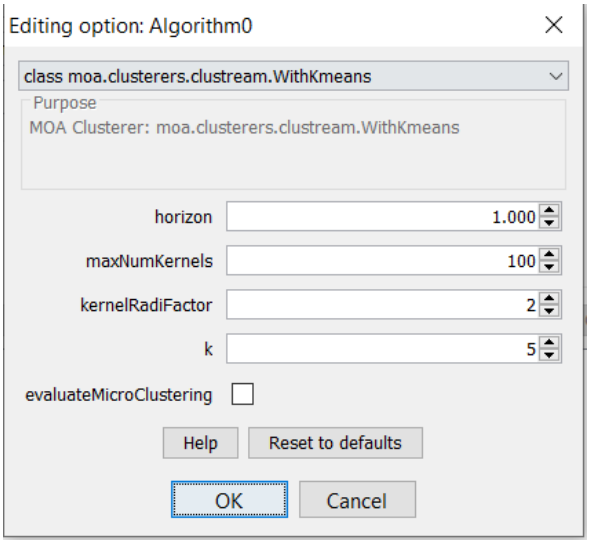
Şekil 3.5: Kurulum paneli.

Açılan Şekil 3.6' daki ekranda üç seçenek mevcuttur. İlk seçenekte FileStream olarak .arff uzantılı dosya kaynağında veri girişi yapılabilmektedir. İkinci seçenekte RandomRBFGeneratorEvents kullanılarak rastgele akan veri üretilebilmektedir. Son seçenekte SimpleCSVStream kullanılarak .csv uzantılı dosya kaynağında veri girişi sağlanabilmektedir. İkinci seçenekteki üretilen veri sentetik veri olmasına karşın, diğer veri seçenekleriyle gerçek zamanlı veri programa aktarılabilir. Bu çalışmada FileStream ile covtypeNorm.arff dosyasındaki veri kullanılacaktır. Şekil 3.6' da gösterilen ekranda, veri değerlendirme sıklığı (evaluationFrequency), bozulmuş veri olma olasılığı(decayThreshold), ortalama eski veri sayısı(decayHorizon) değerleri çalışma ve veri seti kapsamında düzenlenebilir [16].



Şekil 3.6: Veri kaynağı.

Veri kaynağı ayarlandıktan sonra, akan veri kümeleme işlemi için kullanılacak algoritma seçimi Şekil 3.7' deki ekranda yapılır. Horizon ile makro küme hesaplaması için gereken zaman aralığı belirlenir. MaxNumKernels ile mikro küme sayısı, kernelRadiFactor ile bir mikro kümenin yarıçapı belirlenir. Tez çalışmamızda algoritma olarak Clustream-Kmeans++ algoritması ile çalışılacaktır.



Şekil 3.7: MOA örnek algoritma seçimi.

MOA' da kullanılacak veri kümesinden örnek veri parçası Şekil 3.8' de gösterilmiştir.

covtypeNorm.arff

```
@attribute Soil_Type15 {0,1}
@attribute Soil_Type16 {0,1}
@attribute Soil_Type17 {0,1}
@attribute Soil_Type18 {0,1}
@attribute Soil_Type19 {0,1}
@attribute Soil_Type20 {0,1}
@attribute Soil_Type21 {0,1}
@attribute Soil_Type22 {0,1}
@attribute Soil_Type23 {0,1}
@attribute Soil_Type24 {0,1}
@attribute Soil_Type25 {0,1}
@attribute Soil_Type26 {0,1}
@attribute Soil_Type27 {0,1}
@attribute Soil_Type28 {0,1}
@attribute Soil_Type29 {0,1}
@attribute Soil_Type30 {0,1}
@attribute Soil_Type31 {0,1}
@attribute Soil_Type32 {0,1}
@attribute Soil_Type33 {0,1}
@attribute Soil_Type34 {0,1}
@attribute Soil_Type35 {0,1}
@attribute Soil_Type36 {0,1}
@attribute Soil_Type37 {0,1}
@attribute Soil_Type38 {0,1}
@attribute Soil_Type39 {0,1}
@attribute Soil_Type40 {0,1}
@attribute class {1,2,3,4,5,6,7}

@data
0.368684,0.141667,0.045455,0.184681,0.223514,0.071659,0.870079,0.913386,0.582677,0.875366,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0.365683,0.155556,0.030303,0.151754,0.215762,0.054798,0.866142,0.925197,0.594488,0.867838,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0.472736,0.386111,0.136364,0.19184,0.307494,0.446817,0.92126,0.937008,0.531496,0.853339,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,
0.463232,0.430556,0.272727,0.173228,0.375969,0.434172,0.937008,0.937008,0.480315,0.865886,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0.368184,0.125,0.030303,0.10952,0.222222,0.054939,0.866142,0.92126,0.590551,0.860449,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0.36018,0.366667,0.090909,0.214746,0.204134,0.009414,0.905512,0.933071,0.551181,0.840792,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0.373687,0.125,0.106061,0.193271,0.229974,0.088942,0.874016,0.885827,0.543307,0.872159,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0.373187,0.136111,0.060606,0.167502,0.232558,0.080511,0.874016,0.905512,0.566929,0.868256,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0.37919,0.125,0.136364,0.171797,0.295866,0.093579,0.877953,0.870079,0.523622,0.870487,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0.376688,0.163889,0.151515,0.176807,0.237726,0.089363,0.897638,0.862205,0.488189,0.868535,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0.376688,0.558333,0.060606,0.128848,0.289406,0.103274,0.858268,0.956693,0.633858,0.867419,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0.513757,0.419444,0.166667,0.265569,0.257106,0.738092,0.92126,0.944882,0.535433,0.564757,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0.441721,0.372222,0.333333,0.107373,0.312661,0.451735,0.976378,0.88189,0.362205,0.849157,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0.375188,0.594444,0.106061,0.107373,0.282946,0.108332,0.838583,0.972441,0.669291,0.865886,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0.322161,0.436111,0.060606,0.04796,0.228682,0.094703,0.88189,0.944882,0.594488,0.780705,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
```

Şekil 3.8: Veri örneği.

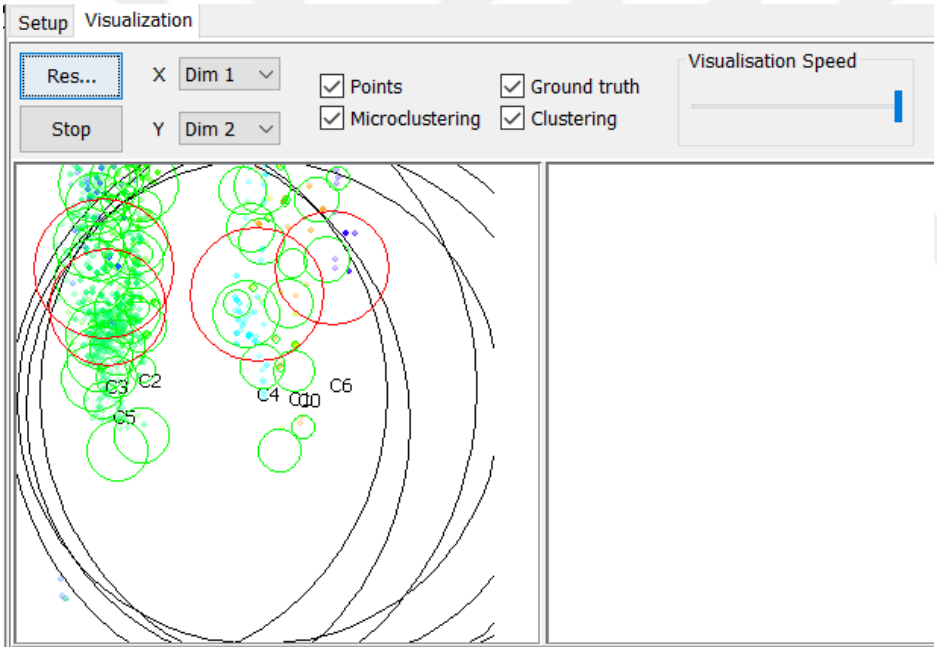
Programa gönderilecek veri kümesi dosyası aşağıdaki özellikleri taşımalıdır.

1. Her kolondaki bilginin veri tipini tanımlayabilmek için @Attribute ifadesi kullanılır.
2. @Data ifadesiyle birlikte veriler gönderilir.

Programa gönderilecek veri akışı sağlandıktan sonra, çalışma sonucu üretilen çıktının görsel olarak takibi görselleştirme(visualization) panelinde yapılabilir.

3.2.2.2. Görselleştirme paneli (Visualization Panel)

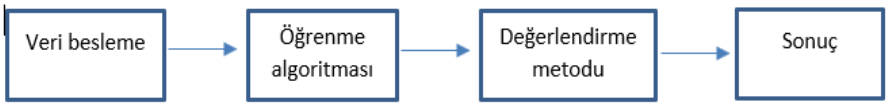
Görselleştirme panelinde, işlenen veri kümesi üzerinden bir çizim gösterilmektedir. Nokta ve hücrelerin boyutları, Şekil 3.9' daki ekrandaki panelden belirlenebilir. Algoritmanın işlediği veri sayısına göre, görselleştirme hızı da ayarlanabilmektedir.



Şekil 3.9: Görselleştirme (Visualization) paneli.

3.2.3. MOA Uygulama Entegrasyonu

MOA, akan veri analizi için en popüler açık kaynaklı Java frameworklerinden biridir. Makine öğrenimi alanında hem denetimli hem de denetimsiz öğrenmeyi destekleyen açık kaynaklı bir yazılım olan Waikato Environment for Knowledge Analysis (WEKA) tabanlıdır ve regresyon, sınıflandırma, kümeleme ve öznitelik seçimi için algoritmalar içerir.



Şekil 3.10: MOA mimarisi.

Şekil 3.10' daki MOA mimarisi incelendiğinde, dış ortamdan alınan veri bağlantıları, veri besleme adımında işlenerek veri akışı bağlantısı oluşturulur. İkinci adımda, alınan veri kümesi üzerinde öğrenme algoritmaları entegre edilir. Bu adımda, MOA' nın sunduğu hazır öğrenme algoritmaları olmakla birlikte, geliştirilen yeni öğrenme algoritmaları da entegre edilebilmektedir. Tez çalışmamızda geliştirilen CluStream-Kmeans++ algoritması uygulama kodu, MOA' ya bu adımda entegre edilmiştir. Değerlendirme adımında uygulamada kullanılan yöntemin, akan veri üzerindeki performans değerlendirilmesi yapılır. Sonuç adımında, çalışma çıktıları elde edilip sonuç analizi yapılmaktadır.

MOA uygulamasında, akan veri entegrasyonu sağlanırken radial basis fonksiyonu kullanılır. Bu fonksiyonda, ilk olarak belli sayıda rastgele küme merkezi belirlenir. Belirlenen küme merkezleri, belirli bir ağırlık ve standart sapma ile uzaya yerleşmektedir. Veri kümesindeki her bir veri, ağırlığının olasılıkla doğru orantılı artacağı şekilde küme merkezlerinin yakınında konumlandırılır. Küme merkezlerine göre yapılan bu dağılımda atanan noktalar, Gauss dağılımı kullanılarak merkezden ayrılırlar. Bu fonksiyonda kullanılan farklı parametreler aşağıdaki gibidir [41]:

- numCluster: Yaratılacak küme sayısı.
- kernelRadius: Yaratılacak küme yarıçap bilgisi.
- noiseLevel: Verinin sahip olacağı gürültü bilgisi.
- kernelRadiusRange: Yaratılacak farklı büyüklükteki kümelerin yarıçap değer aralıkları.
- eventFrequency: Yaratma, ayrılma, birleşme gibi işlemlerin ne kadar sıklıkla gerçekleşeceği bilgisi.
- instanceRandomSeed: Veri kümesi için rastgele besleme değeri.
- densityRange: Kümeler arasındaki yoğunluk farkı değeri.

MOA; sınıflandırma, regresyon, kümeleme, aykırı değer tespiti ve öneri sistemleri gibi çevrimdışı ve çevrimiçi yöntemler içerir. MOA hem bir akan veri işleme aracı hem de bir akan veri işleme algoritması geliştirmek için bir ortam olarak kullanılabilir. Oluşturulan StreamDataSampleGenerator.java isimli Java sınıfı ile uygulamaya akan veri akışı sağlanmaktadır.

```
package moa.streams;

import ...

public class StreamDataSampleGenerator extends AbstractStreamHandler implements
        InstanceStream, ConceptDriftGenerator, CapabilitiesHandler {

    private static final long serialVersionUID = -1255548965556448451;
    protected int numberOfInstancesRead;
    protected boolean hitEndOfFile;
    protected InstanceExample lastInstanceRead;
    protected Reader dataStreamReader;
    protected Instances instanceObjects;
    protected InputStreamProgressMonitor fileProgressMonitor;

    public FileOptionReader arffFileOption = new FileOptionReader("arffFile", 'f',
            "ARFF file to load.", null, "arff", false);

    public IntOption classIndexOption = new IntOption(
            "classIndex",
            'c',
            "Class index of streaming data. 0 for none or -1 for last attribute in file.",
            -1, -1, Integer.MAX_VALUE);

    @Override
    public String getPurposeofOperation() {
        return "A data stream is reading from an ARFF file.";
    }

    public StreamDataSampleGenerator() {
    }

    public ArffFileDataStream(String fileName, int classIndex) {
        this.classIndexOption.setValue(classIndex);
        this.arffFileOption.setValue(fileName);
        restart();
    }

    @Override
    public void preparingForImplementation(TaskMonitor monitor,
                                           ObjectRepository repository) {
        restart();
    }
```

Şekil 3.11: Akan veri entegrasyonu Java kodu-1.

Şekil 3.11’ de, akan veri entegrasyonundan kullanılacak Java sınıf parametreleri ve yapılandırıcı metotlar gösterilmiştir.

```java
@Override
public InstancesHeader getHeaderofInstances() {
    return new InstancesHeader(this.instanceObjects);
}

@Override
public long estimatingRemainingInstances() {
    double progressFraction = this.fileProgressMonitor.getProgressFraction();
    if ((progressFraction > 0.0) && (this.numberOfInstancesRead > 0)) {
        return (long) ((this.numberOfInstancesRead / progressFraction) - this.numberOfInstancesRead);
    }
    return -1;
}

@Override
public boolean fileHasMoreInstances() {
    return !this.hitEndOfFile;
}

@Override
public InstanceExample nextInstance() {
    InstanceExample prevInstance = this.lastInstanceRead;
    this.hitEndOfFile = !readNextInstanceFromFile();
    return prevInstance;
}

@Override
public boolean isRestartEnable() {
    return true;
}
```

Şekil 3.12: Akan veri entegrasyonu Java kodu-2.

Şekil 3.12' de verilen estimatingRemainingInstances() metoduyla kalan veri miktarı alınmıştır. NextInstance() metoduyla akan veri kümesindeki sıradaki örnek alınmaktadır.

```
@Override
public void restart() {
    try {
        if (this.dataStreamReader != null) {
            this.dataStreamReader.close();
        }
        InputStream fileStream = new FileInputStream(this.arffFileOption.getFile());
        this.fileProgressMonitor = new
                InputStreamProgressMonitor(
                fileStream);
        this.dataStreamReader = new BufferedReader(new InputStreamReader(
                this.fileProgressMonitor));
        int classIndex = this.classIndexOption.getValue();
        this.instanceObjects = new Instances(this.dataStreamReader, 1, classIndex);
        if (classIndex < 0) {
            this.instanceObjects.setClassIndex(this.instanceObjects.numAttributes() - 1);
        } else if (this.classIndexOption.getValue() > 0) {
            this.instanceObjects.setClassIndex(this.classIndexOption.getValue() - 1);
        }
        this.numberOfInstancesRead = 0;
        this.lastInstanceRead = null;
        this.hitEndOfFile = !readNextInstanceFromFile();
    } catch (IOException ioe) {
        throw new RuntimeException("Data Stream reading restart failed.", ioe);
    }
    this.clusterEvents = new ArrayList<ClusterEvent>();
}

protected boolean readNextInstanceFromFile() {
    try {
        if (this.instanceObjects.readInstance(this.dataStreamReader)) {
            this.lastInstanceRead = new InstanceExample(this.instanceObjects.instance(0));
            this.instanceObjects.delete();
            this.numberOfInstancesRead++;
            return true;
        }
        if (this.dataStreamReader != null) {
            this.dataStreamReader.close();
            this.dataStreamReader = null;
        }
        return false;
    } catch (IOException ioe) {
        throw new RuntimeException(
                "Operation failed to read instance from stream.", ioe);
    }
}
```

Şekil 3.13: Akan veri entegrasyonu Java kodu-3.

Şekil 3.13’ te, her bir periyot sonunda yeniden veri okunmasını sağlayan restart() ve veri dosyasından sıradaki örneği okuyan readNextInsatnceFromFile() metodları gösterilmiştir.

3.2.4. CluStream-Kmeans++ Algoritma Entegrasyonu

Clustream-Kmeans++ algoritması, örnek tabanlı bir kümeleme yöntemidir ve zaman penceresi modelini kullanır. Algoritmadaki mikro kümeler, küme özellik vektörü (CFV) kullanılarak temsil edilir. CFV, her öznitelik boyutu için tüm demetlerin karelerinin toplamını ve ayrıca zaman özniteliği için karelerin toplamını tutar. CFV, her özellik boyutunda ve zamandaki her mikro küme için ortalama ve varyansı tahmin etmek için kullanılır.

CluStream-Kmeans++ algoritmasında, gelen veri kümesinin özeti oluşturulan mikro kümelerde tutulur. Veri kümesindeki her bir veri, bu veriye en yakın olan mikro kümeye dahil edilir. Gelen bu veriler, bir mikro kümeye dahil edilemiyorsa ayrı bir küme oluşturulur. Yeni oluşturulacak bu mikro kümeye yer açabilmek için, güncellik durumuna göre mikro küme silme veya en yakın mikro kümelerde birleştirme işlemi yapılır. CluStream-Kmeans++ algoritması, diğer kümeleme algoritmalarından farklı olarak anlık sistem durumunu snapshot olarak belleğe kaydeder. Hangi zamandaki bilginin kaydedileceğine de piramidal zaman çerçevesi kullanarak karar vermektedir. Bu yaklaşımda yeni verilerin eski verilere oranla daha çok saklanması prensibi dikkate alınmaktadır.

Anlık Sistem Durumu	Son 5 Zaman Dilimi
0	58 ~~54~~ 53 ~~52~~ 51
1	54 ~~52~~ 50 ~~48~~ 46
2	52 ~~48~~ 44 ~~40~~ 36
3	~~48~~ 40 ~~32~~ 24 ~~16~~
4	48 ~~32~~ 16
5	32

Şekil 3.14: Pirimidal zaman çerçevesi örneği.

Şekil 3.14’ te örnek olarak gösterilen piramidal zaman çerçevesinde, anlık zaman 58’ tir. 0 satırında, son zaman aralığından gelen veriler gösterilmektedir. Son zaman aralığında gelen 5 farklı dilimdeki veriler ile hesaplama yapılacaktır. Bu verilerden önceki anlık sistem

durumlarında yer alanlar bellekte tekrardan tutulmaz, şekildeki çizili durumlar bu kısmı ifade etmektedir.

Uygulamada CluStreamKmeansPlusClusterOperations.java isimli Java sınıfında ClusStream-Kmeans++ kümeleme algoritması geliştirilmiştir.

```
public class CluStreamKmeansPlusClusterOperations extends AbstractClusterer{

    private static final long serialVersionUID = -2563645897521451L;

    public IntOption maximumKernelsOptionNumber = new IntOption(
            "maximumKernelsOptionNumber", 'k',
            "Maximum micro kernels numbers", 100);

    public IntOption kernelRadiFactorInfo = new IntOption(
            "kernelRadiFactorInfo", 't',
            "Factor for the kernel radius", 2);

    public IntOption timeWindowOptionInfo = new IntOption("horizon",
            'h', "window range", 1000);

    private long timestamp = -1;
    private int timeWindowInfo;
    private ClustreamKmeansPlusKernel[] listOfKernels;
    private boolean isTrainingStarted;
    private List<ClustreamKmeansPlusKernel> buffer;
    private int sizeOfBuffer;
    private double t;
    private int m;

    public CluStreamKmeansPlusClusterOperations() {
    }


    @Override
    public void initializeLearningImpl() {
        this.listOfKernels = new ClustreamKmeansPlusKernel[maximumKernelsOptionNumber.getValue()];
        this.buffer = new LinkedList<ClustreamKmeansPlusKernel>();
        this.timeWindowInfo = timeWindowOptionInfo.getValue();
        this.isTrainingStarted = false;
        this.sizeOfBuffer = maximumKernelsOptionNumber.getValue();
        t = kernelRadiFactorInfo.getValue();
        m = maximumKernelsOptionNumber.getValue();
    }
```

Şekil 3.15: CluStream-Kmeans++ algoritması Java uygulama kodu-1.

Şekil 3.15' te CluStream-Kmeans++ algoritmasının kullandığı parametreler ve öğrenme işleminin başladığı initializeLearningImpl() metodu gösterilmiştir.

```java
@Override
public void trainingInstanceImpl(Instance instance) {
    int dim = instance.numValues();
    timestamp++;
    // 1. Starting training
    if ( !isTrainingStarted ) {
        if ( buffer.size() < sizeOfBuffer ) {
            buffer.add( new ClustreamKmeansPlusKernel(instance,dim, timestamp, t, m) );
            return;
        }

        int k = listOfKernels.length;
        assert (k <= sizeOfBuffer);

        ClustreamKmeansPlusKernel[] centers = new ClustreamKmeansPlusKernel[k];
        for ( int i = 0; i < k; i++ ) {
            centers[i] = buffer.get( i );
        }
        Clustering kmeansPlus_clustering = callKmeansPlus(k, centers, buffer);

        for ( int i = 0; i < kmeansPlus_clustering.size(); i++ ) {
            listOfKernels[i] = new ClustreamKmeansPlusKernel( new DenseInstance(1.0,centers[i].getCenter()), dim, timestamp,
                    t, m );
        }

        buffer.clear();
        isTrainingStarted = true;
    }
    // 2. Determine closest kernel
    ClustreamKmeansPlusKernel closestKernel = null;
    double minDistance = Double.MAX_VALUE;
    for ( int i = 0; i < listOfKernels.length; i++ ) {
        double distance = distance(instance.toDoubleArray(), listOfKernels[i].getCenter() );
        if ( distance < minDistance ) {
            closestKernel = listOfKernels[i];
            minDistance = distance;
        }
    }
```

```java
    // 3. Check whether instance fits into closestKernel
    double radius = 0.0;
    if ( closestKernel.getWeight() == 1 ) {
        radius = Double.MAX_VALUE;
        double[] center = closestKernel.getCenter();
        for ( int i = 0; i < listOfKernels.length; i++ ) {
            if ( listOfKernels[i] == closestKernel ) {
                continue;
            }

            double distance = distance(listOfKernels[i].getCenter(), center );
            radius = Math.min( distance, radius );
        }
    } else {
        radius = closestKernel.getRadius();
    }

    if ( minDistance < radius ) {
        closestKernel.insert( instance, timestamp );
        return;
    }

    long threshold = timestamp - timeWindowInfo;
```

Şekil 3.16: CluStream-Kmeans++ algoritması Java uygulama kodu-2.

Şekil 3.16’ da, trainingInstanceImpl() metodu ile algoritmanın sistemi eğittiği kısım geliştirilmiştir. Yine aynı kısımda, en yakın kümenin bulunması ve ilgili örneğin bulunan kümeye uyup uymadığının kontrol edildiği aşamalar gösterilmiştir.

```
    // 3.1 Try to forget old kernels
    for ( int i = 0; i < listOfKernels.length; i++ ) {
        if ( listOfKernels[i].getRelevanceStamp() < threshold ) {
            listOfKernels[i] = new ClustreamKmeansPlusKernel( instance, dim, timestamp, t, m );
            return;
        }
    }

    // 3.2 Merge closest two kernels
    int closestX = 0;
    int closestY = 0;
    minDistance = Double.MAX_VALUE;
    for ( int i = 0; i < listOfKernels.length; i++ ) {
        double[] centerA = listOfKernels[i].getCenter();
        for ( int j = i + 1; j < listOfKernels.length; j++ ) {
            double dist = distance( centerA, listOfKernels[j].getCenter() );
            if ( dist < minDistance ) {
                minDistance = dist;
                closestX = i;
                closestY = j;
            }
        }
    }
    assert (closestX != closestY);

    listOfKernels[closestX].add( listOfKernels[closestY] );
    listOfKernels[closestY] = new ClustreamKmeansPlusKernel( instance, dim, timestamp, t,  m );
}
```

Şekil 3.17: CluStream-Kmeans++ algoritması Java uygulama kodu-3.

Şekil 3.17’ de, eski kümenin unutulduğu ve en yakın iki kümenin birleştirildiği kod kısımları gösterilmiştir.

```java
@Override
public Clustering getMicroClusteringResult() {
    if ( !isTrainingStarted ) {
        return new Clustering( new Cluster[0] );
    }

    ClustreamKmeansPlusKernel[] res = new ClustreamKmeansPlusKernel[listOfKernels.length];
    for ( int i = 0; i < res.length; i++ ) {
        res[i] = new ClustreamKmeansPlusKernel( listOfKernels[i], t, m );
    }

    return new Clustering( res );
}

@Override
public boolean implementsMicroClusterer() { return true; }

private static double distance(double[] pointA, double [] pointB){
    double distance = 0.0;
    for (int i = 0; i < pointA.length; i++) {
        double d = pointA[i] - pointB[i];
        distance += d * d;
    }
    return Math.sqrt(distance);
}

public static Clustering callKmeansPlus( int k, List<? extends Cluster> data ) {
    Random random = new Random(0);
    Cluster[] centers = new Cluster[k];
    for (int i = 0; i < centers.length; i++) {
        int rid = random.nextInt(k);
        centers[i] = new SphereCluster(data.get(rid).getCenter(),0);
    }
    Clustering clustering = kmeansPlus(k, centers, data);
    return clustering;
}
```

Şekil 3.18: CluStream-Kmeans++ algoritması Java uygulama kodu-4.

Şekil 3.18' de, çevrimiçi aşamada yapılan mikro kümeleme işleminin sonucunu getiren getMicroClusteringResult() ve çevrimdışı aşamada k-means++ algoritmasının uygulanmasını gösteren callKmeansPlus() metotları geliştirilmiştir.

```java
public static Clustering kmeansPlus( int k, Cluster[] centers, List<? extends Cluster> data ) {
    assert (centers.length == k);
    assert (k > 0);

    int dimensions = centers[0].getCenter().length;

    ArrayList<ArrayList<Cluster>> clustering = new ArrayList<ArrayList<Cluster>>();
    for ( int i = 0; i < k; i++ ) {
        clustering.add( new ArrayList<Cluster>() );
    }

    int repetitions = 100;
    while ( repetitions-- >= 0 ) {
        // Assign points to clusters
        for ( Cluster point : data ) {
            double minDistance = distance( point.getCenter(), centers[0].getCenter() );
            int closestCluster = 0;
            for ( int i = 1; i < k; i++ ) {
                double distance = distance( point.getCenter(), centers[i].getCenter() );
                if ( distance < minDistance ) {
                    closestCluster = i;
                    minDistance = distance;
                }
            }

            clustering.get( closestCluster ).add( point );
        }

        // Calculate new centers and clear clustering lists
        SphereCluster[] newCenters = new SphereCluster[centers.length];
        for ( int i = 0; i < k; i++ ) {
            newCenters[i] = calculateCenter( clustering.get( i ), dimensions );
            clustering.get( i ).clear();
        }
        centers = newCenters;
    }

    return new Clustering( centers );
}
```

Şekil 3.19: CluStream-Kmeans++ algoritması Java uygulama kodu-5.

Şekil 3.19’ da, çevrimdışı aşamada kullanılan k-means++ algoritmasının uygulaması gösterilmiştir.

```
    private static SphereCluster calculateCenter( ArrayList<Cluster> cluster, int dimensions ) {
        double[] res = new double[dimensions];
        for ( int i = 0; i < res.length; i++ ) {
            res[i] = 0.0;
        }

        if ( cluster.size() == 0 ) {
            return new SphereCluster( res, 0.0 );
        }

        for ( Cluster point : cluster ) {
            double [] center = point.getCenter();
            for (int i = 0; i < res.length; i++) {
                res[i] += center[i];
            }
        }

        // Normalize
        for ( int i = 0; i < res.length; i++ ) {
            res[i] /= cluster.size();
        }

        // Calculate radius
        double radius = 0.0;
        for ( Cluster point : cluster ) {
            double dist = distance( res, point.getCenter() );
            if ( dist > radius ) {
                radius = dist;
            }
        }
        SphereCluster sc = new SphereCluster( res, radius );
        sc.setWeight(cluster.size());
        return sc;
    }
}
```

Şekil 3.20: CluStream-Kmeans++ algoritması Java uygulama kodu-6.

Şekil 3.20' de, küme merkezi hesaplayan calculateCenter() metodu ve küme yarıçap bilgilerinin hesaplandığı detaylar verilmiştir.

3.2.5. Sistem Durum Tespiti

Bu kısımda, Clustream-Kmeans++ algoritmasıyla akan verinin işlenmesi sırasında belirlenen metriklere göre sistemin durumu belirli periyotlarla izlenmektedir. MOA uygulamasındaki anlık durum sonuçlarını girdi olarak alan bu yöntemde, sistem parametrelerinde meydana gelen değişimler izlenerek akan veri yükünün daha müsait istasyonlara aktarımı için bir analiz yapılmıştır.

Geliştirilen uygulamada bu kısımda Clustream-Kmeans++ algoritmasıyla akan verinin işlenmesi sırasında belirlenen metriklere göre sistemin durumu belirli periyotlarla izlenmektedir.

```java
public class CluStreamKmeansPlusObserver extends AbstractMOAObserver, TaskThread
implements StreamObserver{
    private String[] evaluationMetrics;
    private double evaluationMetricValue = 0;
    private HashMap<String, Integer> map;
    private int numEvaluations = 0;

    public EvaluationResult getEvaluationResult(ThreadTask task) {
        TaskThread taskThread = new TaskThread(task);
        evaluationMetrics = getEvaluationMetrics(taskThread.getTaskId());
        numEvaluations = evaluationMetrics.length;
        map = new HashMap<String, Integer>(numEvaluations);
        for (int i = 0; i < numEvaluations; i++) {
            evaluationMetricValue = evaluationMetrics[i].getName(i,evaluationMetrics[i].getMetricEvaluationOption());
            map.put(evaluationMetrics[i],evaluationMetricValue);
        }
        EvaluationResult evaluationResult = new EvaluationResult();
        evaluationResult.add(map);
        return evaluationResult;
    }

    public ResponseEvaluateClusteringPerformance evaluateClusteringPerformance(Clustering clustering) throws Exception{
        EvaluationResult evaluationResult = getEvaluationResult(clustering.getTask());
        // AbstractMOAObserver includes getSystemMeasures and evaluateSystemPerformance
        ResponseSystemMeasures responseSystemMeasures = getSystemMeasures();
        ResponseCheckSystem responseCheckSystem = evaluateSystemPerformance(clustering,
                evaluationResult, responseSystemMeasures);
        return new ResponseEvaluateClusteringPerformance(responseCheckSystem.getIsAlertForScaling(), clustering);
    }
}
```

Şekil 3.21: Akan veri işleme sırasında sistemin anlık değerlendirilmesi.

Şekil 3.21' deki uygulamamızda geliştirilen CluStreamKmeansPlusObserver.java sınıfındaki getEvaluationResult metodu ile belirlenen metriklere göre CluStream-Kmeans++ algoritmasının akan veri işleme sırasındaki değerlendirme sonuçları anlık olarak alınmıştır. AbstractMOAObserver.java sınıfından sistemin işlemci ve hafıza yükü bilgilerini getSystemMeasures() metodu ile elde edilmiştir. Değerlendirme adımında AbstractMOAObserver.java sınıfındaki evaluateSystemPerformance() metodu önemlidir. Bu metot içerisinde işlemci yükü, kalan bellek miktarı, kullanılan bellek miktarı bilgileri hesaplanmaktadır. Bu değerler, sistemde belirlenen eşik değerinin altına düştüğünde ResponseCheckSystem sınıfında sistemin ölçeklenmesi için cevap dönülür.

3.2.6. Ölçekleme İşleminin Tamamlanması

Bu bölümde, akan veri işleme sırasında izlenen sistemin bir önceki adımda yapılan analiz sonucunda ölçeklenme ihtiyacı oluştuğunda yapılan işlemler ve akış anlatılmıştır. Uygulamamızı entegre ettiğimiz MOA uygulamasıyla birlikte bu adımda, Apache Flink[49] uygulamasının API' leri kullanılmıştır.

Apache Flink; sınırlı ve sınırsız veri işleyebilen, durum bilgisi olan hesaplamalarda kullanılabilen dağıtık işlem mimarisidir. Akan veriler gibi yüksek verimlilik ve düşük gecikme beklenen, işlemci ve bellek problemlerinin ortaya çıktığı projelerde büyük yarar sağlamaktadır. Apache Flink, DataStream API üzerinden uygulamalara bir veya daha fazla kaynaktan paralel şekilde veri işleme imkanı sunmaktadır. Flink mimarisinde bir akışın çalışması sırasında üç operatör görev alır. Kaynak operatörü, akışa girecek olan veri kaynağını belirler. İşlem operatörü, akış sırasında işlemleri gerçekleştiren operatördür. Çıkış operatörü, üretilen çıktı akışlarının ilgili alıcıya ulaştırılmasında görevlidir. Bu alıcı, bu akışı dinleyen bir kod bloğu veya dosya olabilir.

Apache Flink çalışma sürecinde iş yöneticileri, iş akışlarını ve alt operasyonları gerçekleştiren işçi düğümleri takip etmektedir. İş yöneticileri, çeşitli metrikler ile anlık akış durumunu kontrol etmektedir. Status.JVM.CPU etiketi ile, ilgili iş akışını çalıştıran JVM' in işlemci yükü elde edilir. Status.JVM.Memory etiketi ile, kullanılan hafıza bilgisi elde edilir. Birim zamanda işlenen kayıt sayısı ve gecikme süresi değerlerini de içinde bulunduran bir değerlendirmeyle ölçeklenme yapılır. Bu çalışma sonucunda ihtiyaç duyulursa uygun düğümler varsa iş yükü o düğümlere iletilir.

```
public class CluStreamKmeansScalingOperationsHandler extends AbstractMOAObserver, TaskThread implements
        InstanceStream, CapabilitiesHandler{
    private static final long serialVersionUID = -1255548965556448845L;

    public ResponseScaleOperation connectMOAToFlink(ResponseEvaluateClusteringPerformance responseEvaluateClusteringPerformance) {
        // gets executionEnvironment from InstanceStream
        StreamExecutionEnvironment executionEnvironment = getStreamExecutionEnvironment().getExecutionEnvironment();
        executionEnvironment.setStreamTimeCharacteristic(getTimeCharacteristic().getProcessingTime());
        executionEnvironment.enableCheckpointing(getTimeCharacteristic().getEnableCheckPoint());
        executionEnvironment.getConfig().setMultiProcess(getTimeCharacteristic().getMultiProcess());
        TaskThread taskThread = new TaskThread(responseEvaluateClusteringPerformance.getClustering().getTask());
        PeriodicSourceGenerator sourceGenerator = new PeriodicSourceGenerator(
                responseEvaluateClusteringPerformance.getClusteringInstance(), taskThread.getDuration());
        executionEnvironment.addSource(sourceGenerator);
        executionEnvironment.execute();
    }
```

Şekil 3.22: MOA-Apache Flink bağlantısı.

Şekil 3.22’ de ölçekleme isteğinin MOA’ dan Apache Flink’ e nasıl yapıldığı gösterilmiştir. CluStreamKmeansScalingOperationsHandler.java sınıfındaki connectMOAToFlink() metodunda, DataStreamAPI’ nin beklediği executionEnvironment parametreleri ayarlanarak execute() metodu çalıştırılmıştır.

4. BULGULAR

Bu bölümde CluStream-Kmeans++ algoritması ve literatürdeki diğer algoritmaların akan veri kümesi üzerindeki çalışma detayları, test verileri, test sırasında kullanılan metrikler ve test sonuçları incelenmiştir.

4.1. PLATFORM, DEĞERLENDİRME METRİKLERİ VE VERİ SETİ

4.1.1. Kullanılan Platformlar

Bu bölümde, geliştirme ve test ölçümlerinde kullanılan teknoloji ve platformlar anlatılmıştır.

4.1.1.1. Teknik Altyapı

Geliştirilen uygulama Java8 tabanlı bir uygulamadır. Geliştirme sırasında, Java8 yeniliklerinden faydalanılmıştır. Aşağıda detayları paylaşılan bilgisayar üzerinde geliştirme ve testler tamamlanmıştır.

- İşlemci: Intel(R) Core(TM) i7-3630QM CPU @ 2.40GHz 2.40 GHz.
- RAM: 32.0 GB - 64 bits, x64.
- İşletim sistemi: Windows 10, 2020.

4.1.1.2. Intellij IDEA

Intellij IDEA, Java yazılım dilinde uygulama geliştirmek için kullanılan bir yazılım geliştirme ortamıdır. JetBrains şirketinin geliştirdiği uygulamanın, hem topluluk hem ticari sürümleri bulunmaktadır. Geliştirme kapsamında topluluk sürümüyle ilerlenmiştir. Her iki sürümde proje çalışmalarında verimli olarak kullanılabilmektedir. Bu çalışmaya sağladığı kullanım kolaylığı, hız ve geliştirme desteği nedeniyle kullanılmıştır [42].

4.1.2. Değerlendirme İndeksleri

Değerlendirme indeksi, bir veri kümesi hakkında ön bilgiye sahip olup olmamaktan bağımsız olarak verinin ne kadar başarılı işlendiğini gösteren ölçüm değerleridir. Değerlendirme indeksleri, değişken kriterlere göre iç değerlendirme ve dış değerlendirme olmak üzere ikiye ayrılmaktadır [43].

- **İç Değerlendirme**: Veriler hakkında ön bilgiye sahip olmadan, veriyi işlediği andaki görüntüsüyle değerlendiren yöntemlerdir. Silhouette indeksi, Davies-Bouldin indeksi, hata kareleri toplamı (Sum Squared Distance) en çok bilinen ve yaygın kullanılan iç değerlendirme indeksleridir.
- **Dış Değerlendirme**: İşlenen veri hakkında ön bilgiye sahip olarak, veriler hakkındaki etiket bilgilerini de kullanarak değerlendirme yapan yöntemlerdir. F1- Kesinlik (F1-Precision), Saflık(Purity), F1-Hassaslık (F1-Recall), Rand indeksi en çok bilinen ve yaygın kullanılan dış değerlendirme indeksleridir.

Tez çalışması kapsamında, geliştirilen algoritmanın test edilmesi için altı adet değerlendirme indeksi kullanılmıştır. Kullanılan iç değerlendirme indeksleri; Silhouette indeksi ve hata kareleri toplamı (SSE), kullanılan dış değerlendirme indeksleri; F1-Hassaslık, F1-Kesinlik, Rand indeksi ve saflıktır.

4.1.2.1. F1-Ölçütü (F1-Measure)

Gerçek etiketler kullanılarak bulunan, istatiksel anlamda ikili sınıflandırmada kullanılan bir ölçüttür. Yapılan tahminin hangi oranda başarılı olduğunu tespit eder. F1-Hassaslık ve F1-Kesinlik değerleriyle hesaplanır. Hesaplama ile [0,1] aralığında bir değer elde edilir. Hesaplanan değerin yüksek olması, başarılı bir kümeleme yapıldığını ifade eder [43].

- **F1-Hassaslık (F1-Recall)**: Elde edilen doğru bilginin, elde edilmesi gereken doğru bilgiye olan oranını ifade eder.
- **F1-Kesinlik (F1-Precision)**: Elde edilen doğru bilginin, gelen tüm bilgiye olan oranını ifade eder. Yapılan kümelenin doğruluğu hakkında bilgi verir.

4.1.2.2. Saflık (Purity)

Saflık, kullanılan modelde yapılan kümeleme yaklaşımında her küme içinde bulunan verilerden sayısının en fazla olan verinin toplam veri sayısına olan oranını ifade etmektedir. Saflık değeri, [0,1] aralığında bir değer alır, yüksek saflık değeri başarılı kümeleme yapıldığını göstermektedir. Saflık değeri, eşitlik 4.1 ile hesaplanmaktadır [43].

$$P = \frac{\sum_{i=1}^{k} c_i}{N} \tag{4.1}$$

Eşitlik 4.1' de N toplam veri örneği sayısını, c_i kümedeki en fazla sayıda bulunan örnek veri sayısını ifade eder.

4.1.2.3. Hata Kareleri Toplamı (SSE)

Hata kareleri toplamı, kümede bulunan her bir elemanın, kümenin merkezine olan uzaklığının karelerinin toplamını ifade etmektedir. Elde edilen küçük değerler, yapılan kümeleme çalışmasının başarılı olduğunu göstermektedir. Küme sayısına bağımlı olan bu yöntem, eşitlik 4.2 ile hesaplanmaktadır [43].

$$SSE = \sum_{i=1}^{N} (x_i - \bar{x})^2 \tag{4.2}$$

Hesaplama formülünde, x_i kümedeki küme eleman sayısını, N kümedeki toplam eleman sayısını, $\bar{x}$ ilgili küme merkezini gösterir.

4.1.2.4. Rand İndeks

Rand indeks, kümedeki verilerin değerini hesaplama sırasında ilgili veriyi dahil olduğu kümedeki diğer verilerle ikili olarak değerlendirerek, ne kadar benzerlik olduğunu hesaplayan dış değerlendirme metriğidir. Rand indeks değeri [0,1] aralığında değer alır. Hesaplanan değer büyüdükçe, iki kümelemenin benzerliği artmaktadır [43].

K, n adet elemana sahip ve kümeleme işlemi yapılacak bir veri grubunu temsil etsin K = $\{k_1,\ldots,k_n\}$. K üzerinde M tahminlenen kümeleme, R gerçek kümeleme bilgilerini temsil etsin M = $\{m_1, \ldots, m_n\}$ ve R = $\{r_1, \ldots., r_n\}$

a, M ve R' de aynı kümelerde bulunan eleman sayısı b, M ve R'de farklı kümelerde bulunan eleman sayısı c, M' de aynı kümede R' de farklı kümede yer alan eleman sayısı d, M' de farklı kümede R' de ayı kümede bulunan eleman sayısı olarak aşağıdaki eşitlik 4.3 ile Rand indeks hesaplanır.

$$R = \frac{a+b}{a+b+c+d} = \frac{a+b}{\binom{n}{2}} \tag{4.3}$$

Burada a + b ikili bölümlemenin benzerlik gösterdiği eleman sayısı toplamını, c + d değeri ise ikili bölümlemenin benzerlik göstermediği eleman sayıları toplamını ifade etmektedir.

4.1.2.5. Silhouette İndeksi

Bir kümenin sahip olduğu elemanların diğer kümelerle karşılaştırıldığında kendi kümesine ne kadar benzediğini ölçmek için hesaplanan bir indeks değeridir. Silhouette değeri, [-1, 1] aralığında değer almaktadır. Hesaplama sonucunda elde edilecek yüksek değerler, kümeleme işleminin başarılı olduğunu göstermektedir [43]. Silhouette indeksi aşağıdaki eşitlik 4.4 ile hesaplanmaktadır.

$$S(\mathrm{i}) = \frac{k(\mathrm{i}) - m(\mathrm{i})}{max(k(\mathrm{i}), m(\mathrm{i}))} \tag{4.4}$$

X_i kümesindeki i elemanına ait Silhouette indeksi hesaplarken; k(i), ilgili noktanın ait olduğu küme haricindeki diğer küme elemanlarına olan Öklid uzaklık ortalamalasını; m(i), i noktasının ait olduğu kümedeki elemanlara olan Öklid uzaklık ortalamasını ifade etmektedir.

4.1.3. Veri Seti

Bu kısımda tez çalışması kapsamında yapılan çalışmanın test edilmesi için kullanılan veri setlerinin detayları verilmiştir. Gerçek ve yapay veri setleri kullanılarak, algoritmaların performansı izlenmiştir.

4.1.3.1. Gerçek Veri Seti

Gerçek veri seti olarak, akan veri işleme çalışmalarında kullanılan Forest Cover Type veri seti kullanılmıştır.

Forest Cover Type Veri Seti: Forest Cover Type veri seti, ABD Orman Servisi Kaynak Bilgi Sistemi verilerinden elde edilen, Kuzey Colarado' da bulunan Roosevelt Ulusal Park' ındaki baskın tür olan yedi bitki türü hakkında veri içermektedir [44]. Bitki türleri ve veri seti hakkında Tablo 4.1' de detay bilgiler paylaşılmıştır.

Tablo 4.1: Forest Cover Type veri seti içerisindeki veri dağılımı.

Bitki Türü	**Veri Setindeki Sayısı**
Lodgepole çamı	283.301
Ladin : Köknar	211.840
Kavak : Söğüt	2.747
Douglas Köknar	17.367
Titrek Kavak	9.493
Ponderosa çamı	35.754
Büyümesi duran çalılaşan ağaç	20.510
Toplam	581.012

Bu veri setinde, 2’ si ikili veri seti, 10’ u devamlı olmak üzere 12 öznitelik uzayı bulunmaktadır. Öznitelik detayları Tablo 4.2’ de verilmiştir.

Tablo 4.2: Forest Cover Type veri setindeki öznitelik uzayı.

Öznitelik Adı	**Birim**
Yön	derece
En yakın su kaynağına yatay uzaklık	metre
En yakın su kaynağına dikey uzaklık	metre
Eğim	derece
Yükseklik	metre
Yaz gündönümü sabah saat 09:00’da güneş ışığı ölçüsü	indeks
Yaz gündönümü saat 12:00’de güneş ışığı ölçüsü	indeks
Yaz gündönümü saat 15:00’de güneş ışığı ölçüsü	indeks
En yakın tarihli yangın başlangıç noktasına yatay uzaklık	metre

Uygulamada testleri sırasında, Forest Cover Type veri seti akan veri olarak kullanılmıştır. Bu yüzden veri seti içerisinde farklı sınıflar bulunsa da herhangi bir zamanda yapılan bir değerlendirme için sistemde $d \leq 7$ olarak d adet sınıf sistemde bulunmaktadır. Test sürecinde, herhangi bir anda gerçekte kullanılan minimum sınıf sayısı 2, maksimum sınıf sayısı 7' dir.

4.1.3.2. Yapay Veri Seti

Bu kısımda, önceki bölümde aktarılan gerçek veri setinden farklı olarak belirli senaryolar için üretilen yapay veri setinin detayları açıklanmıştır. Belirlenen konfigürasyonlarla oluşturulan yapay veri seti üzerinde çalışılan algoritma test edilmiştir. Gürültü oranı değeri, küme sayısı, küme yarıçapı, boyut bilgisi, küme sayısı aralığı, oluşturma/silme, ayırma/birleştirme, küme yarıçapı standart sapması özellikleri değiştirilerek farklı durumlar oluşturulmuştur. Tablo 4.3' te paylaşıldığı gibi ortam değişkenlerinin bir veya daha fazlasının değiştirilerek oluşturulan durumlar üzerinde çalışılmıştır.

Tablo 4.3: Yapay veri setiyle oluşturulan durumlar.

Durum	Küme Sayısı Aralığı	Küme Sayısı	Boyut Bilgisi	Küme Yarıçap Sapması	Oluşturma ve Silme	Ayrıştırma ve Birleştirme	Küme Yarıçapı	Gürültü Bilgisi
1	-	5	2	0	Hayır	Hayır	0,07	0,1
2	-	7	2	0	Hayır	Hayır	0,07	0,1
3	-	5	2	0	Hayır	Hayır	0,07	0,2
4	-	5	10	0	Hayır	Hayır	0,07	0,1
5	-	5	2	0	Hayır	Hayır	0,10	0,1
6	-	5	2	0,4	Hayır	Hayır	0,07	0,1
7	+4, -4	5	2	0	Hayır	Evet	0,07	0,1
8	+4, -4	5	2	0	Evet	Hayır	0,07	0,1
9	+4, -4	5	2	0	Evet	Evet	0,07	0,1
10	-	5	10	0,4	Hayır	Hayır	0,07	0,1

4.2. DENEY SONUÇLARI

Bu kısımda, tez çalışmasında geliştirilen CluStream-Kmeans++ algoritması test edilmiş, literatürdeki diğer algoritmalarla ve farklı veri setleriyle çalıştırılıp sonuçları ele alınmıştır. Kullanılan veri setleri, önceki bölümde gerçek veri setleri ve yapay veri setleri olmak üzere iki kısımda açıklanmıştır. Yapay veri setleri üzerindeki algoritma çalışmaları, gerçek veriye göre daha kolay ve yüksek doğruluk değerleri ile çalışmaktadır. Bu sebeple algoritmaları karşılaştırmak zorlaşsa da detaylı bir değerlendirme için yapay veriyle yapılacak çalışmada kullanılmak üzere önceki bölümde farklı durumlar oluşturulmuştur.

4.2.1. Algoritma Performans Değerlendirmesi

Bu bölümde, tez çalışmasında geliştirilen CluStream-Kmeans++ algoritması ve literatürde yer alan akan veri kümeleme algoritmaları, gerçek veri ve yapay veri kullanılarak önceki bölümlerde bahsedilen değerlendirme metrikleriyle değerlendirilip yorumlanmıştır.

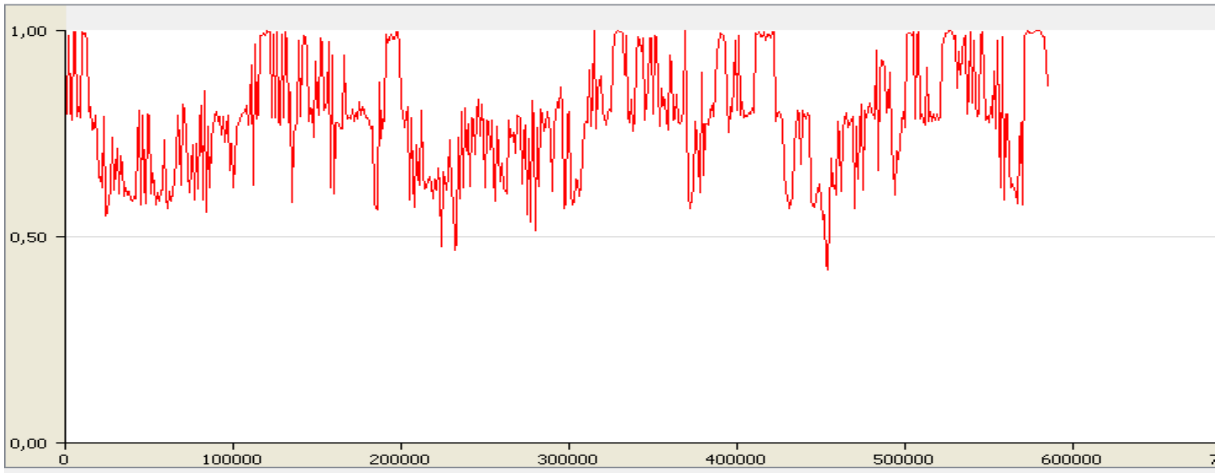
4.2.1.1 Gerçek Veri Setleri ile Değerlendirme

Tez çalışmamızda akan veri akışı oluşturmak için gerçek veri seti olarak Forest Cover Type kullanılmıştır. Forest Cover Type veri seti, 581.012 veriden oluşup ormandaki ağaçların gözlem bilgilerini içermektedir. Veri setinde üzerinde yapılan deneysel çalışmalarda sayısal 10 öznitelik verisi kullanılmıştır. Ağaçların boyu, yönü, en yakın su kaynağına uzaklığı gibi ölçüm bilgileri veri setinde yer almaktadır.

Tablo 4.4 : Forest Cover Type veri setiyle varsayılan parametreler ile elde edilen sonuçlar.

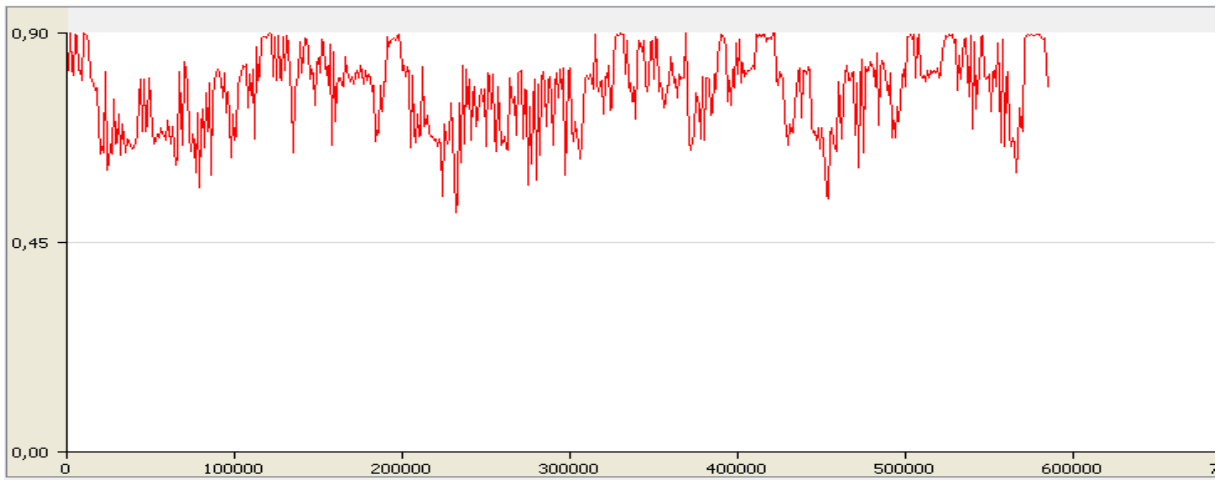
Algoritma	F1-P	F1-R	SSE	Silhouette İndeksi	Rand İndeks	Saflık
CluStream-Kmeans++	0,74	0,69	11,12	0,79	0,60	0,76
CluStream	0,44	0,41	20,23	0,6	0,55	0,66
StreamKM++	0,09	0,08	42,35	0,65	0,52	0,62
ClusTree	0,40	0,28	25,75	0,70	0,51	0,64
DenStream	0,38	0,21	29,03	0,66	0,48	0,59

Tablo 4.4’ te görüldüğü üzere, gerçek akan veri üzerinde literatürdeki diğer algoritmalarla birlikte yapılan test çalışmalarında, CluStream-Kmeans++ algoritmasının diğer algoritmalara göre daha iyi sonuçlar ürettiği görülmüştür. Bu durumun en büyük nedeni, testlerde kullanılan diğer algoritmaların veri boyutunu algılayacak ve kullandıkları parametreleri düzenleyecek bir yapılarının olmayışıdır. Kullanılan gerçek verilerin, yapay verilere kıyasla daha düzensiz olması, aynı grupta bulunan iki veri noktasının yapay verilere kıyasla birbirine benzememesi de bu sonuçların çıkmasında etkili olmuştur. Aşağıdaki bölümde algoritmalarla yapılan testlerin, değerlendirme sonuçları grafiklerle paylaşılmıştır. Grafiklerdeki yatay eksen, işlenen kayıt sayısını; dikey eksen, hesaplanan indeks değerini göstermektedir.



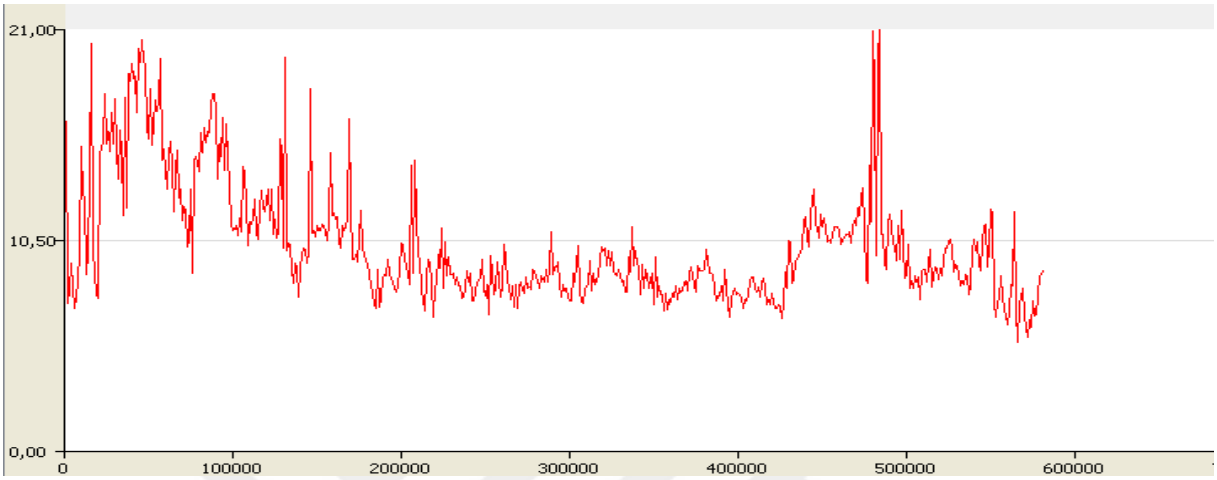
Şekil 4.1: CluStream-Kmeans++ algoritması F1-Kesinlik grafiği.

Şekil 4.1’ de görüldüğü üzere, CluStream-Kmeans++ algoritması ile gelen veri üzerinde yapılan kümelemede ortalama 0,74 F1-Kesinlik değerine ulaşılmıştır.



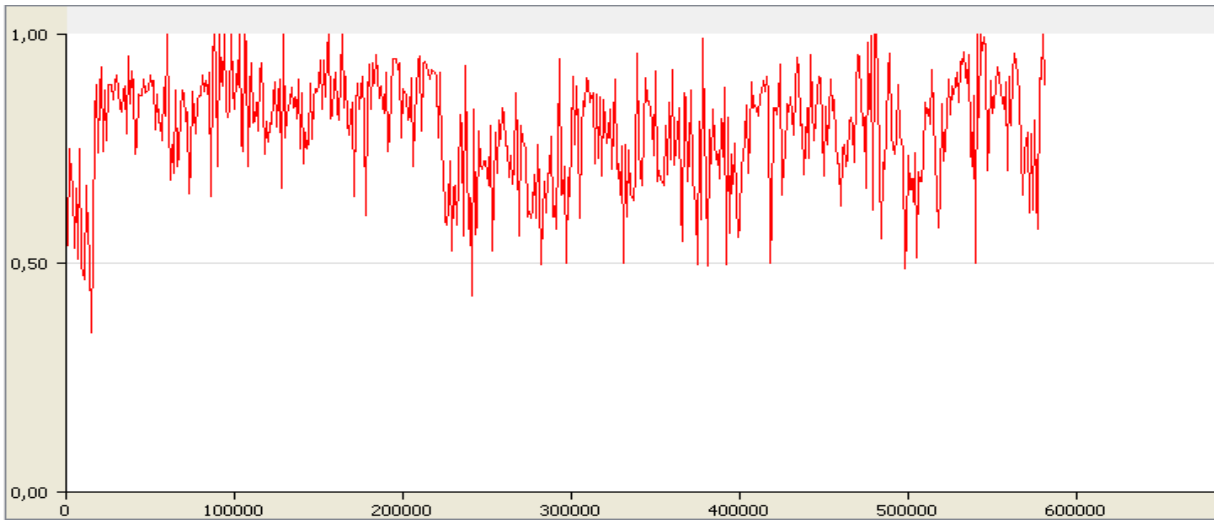
Şekil 4.2: CluStream-Kmeans++ algoritması F1-Hassaslık grafiği.

Şekil 4.2’ de görüldüğü üzere, CluStream-Kmeans++ algoritması ile gelen veri üzerinde yapılan kümelemede ortalama 0,69 F1-Hassaslık değerine ulaşılmıştır. Gerçek verilerle yapılan çalışmada görece düşük bir değer olsa da, diğer algoritmalara göre üstünlük sağlamıştır.



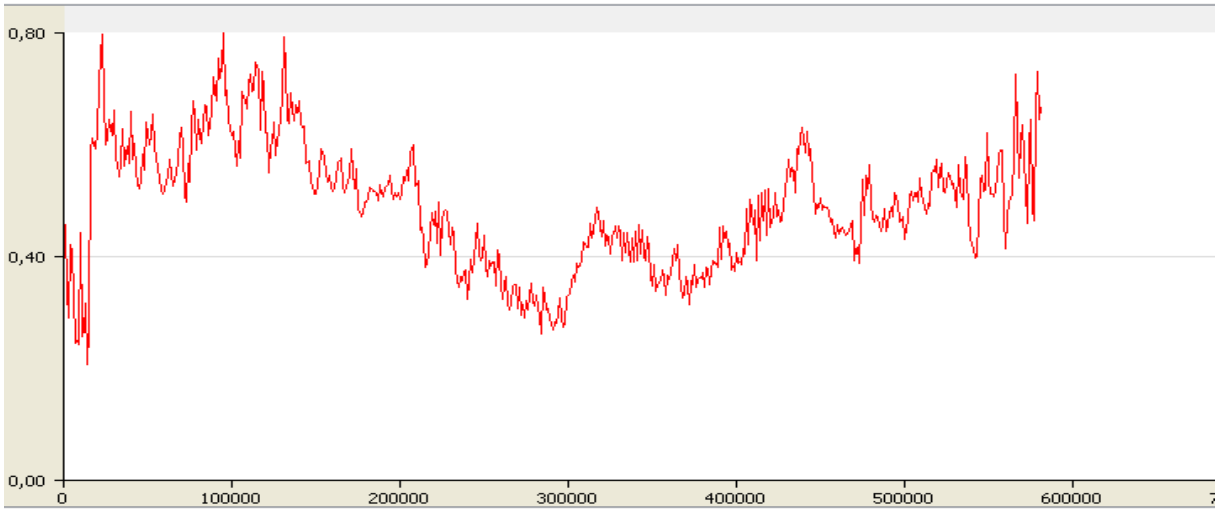
Şekil 4.3: CluStream-Kmeans++ algoritması hata kareleri toplamı grafiği.

Hata kareleri toplamı(SSE) kümede yer alan elemanların, küme merkezine olan uzaklıklarının karelerinin toplamını ifade etmektedir. Şekil 4.3’ de görüldüğü üzere, CluStream-Kmeans++ algoritması için bulunan SSE değeri, ortalama 11,12’ dir.



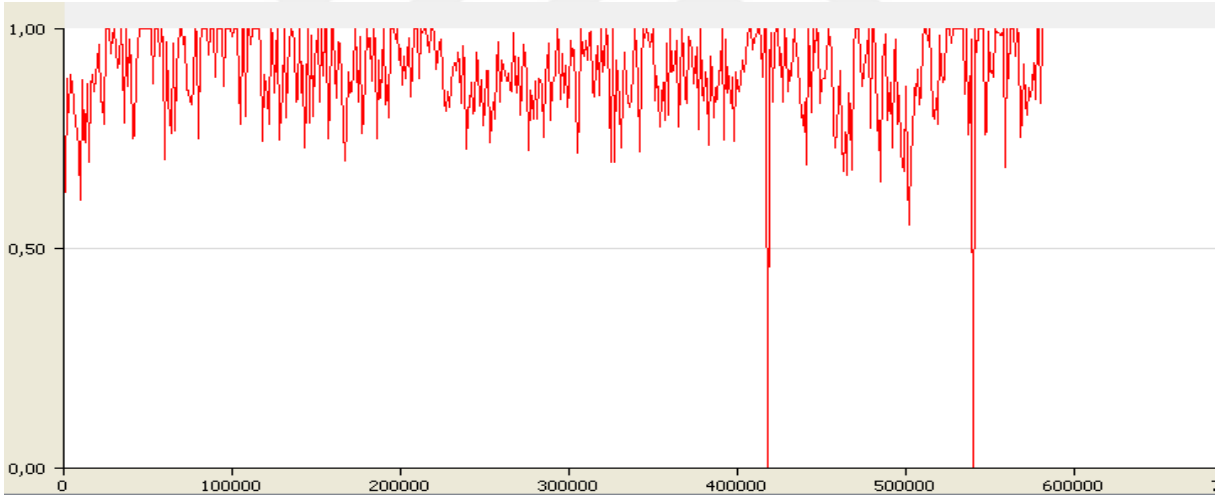
Şekil 4.4: CluStream-Kmeans++ algoritması Silhouette indeksi grafiği.

Şekil 4.4’ te görüldüğü üzere, CluStream-Kmeans++ algoritması için hesaplanan Silhouette indeksi değeri, ortalama 0,79’ dur. Silhouette indeksi, kümedeki elemanların birbirleriyle ne oranda birleşmiş olduğunu ve diğer kümelerden ne oranda ayrıldığını ifade etmektedir.



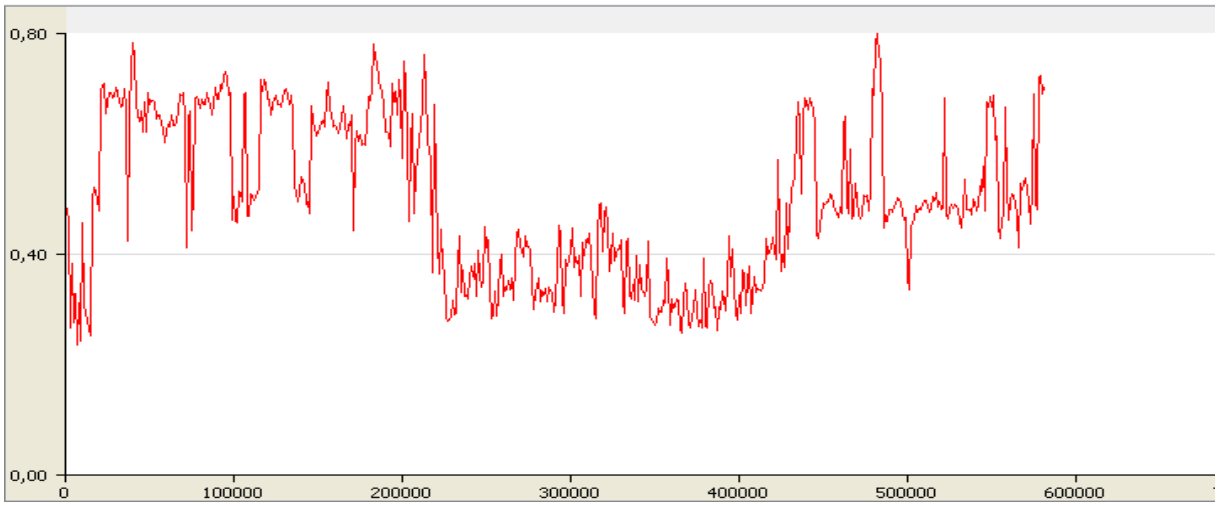
Şekil 4.5: CluStream-Kmeans++ algoritması Rand indeks grafiği.

Şekil 4.5' te görüldüğü üzere, CluStream-Kmeans++ algoritması için hesaplanan Rand indeks değeri, ortalama 0,60' tır. Rand indeksi, kümelemede kullanılan ve iki bölüm arasındaki benzerliği tespit etmekte kullanılan bir indekstir.



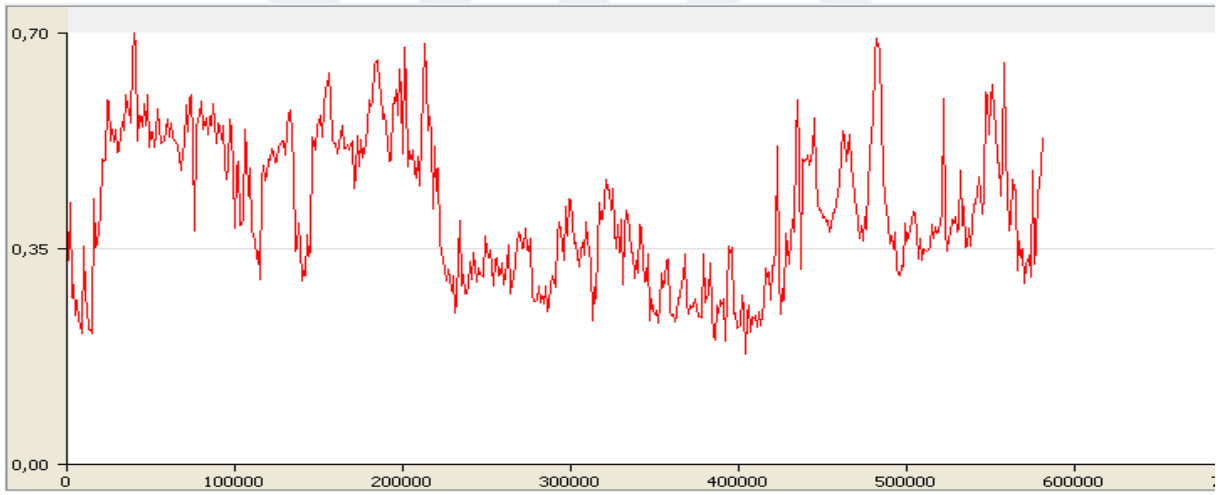
Şekil 4.6: CluStream-Kmeans++ algoritması saflık grafiği.

Şekil 4.6' da görüldüğü üzere, CluStream-Kmeans++ algoritması için hesaplanan saflık değeri, ortalama 0,76' dır. Saflık değeri, kümede bulunan baskın elemanların oranını ifade etmektedir.



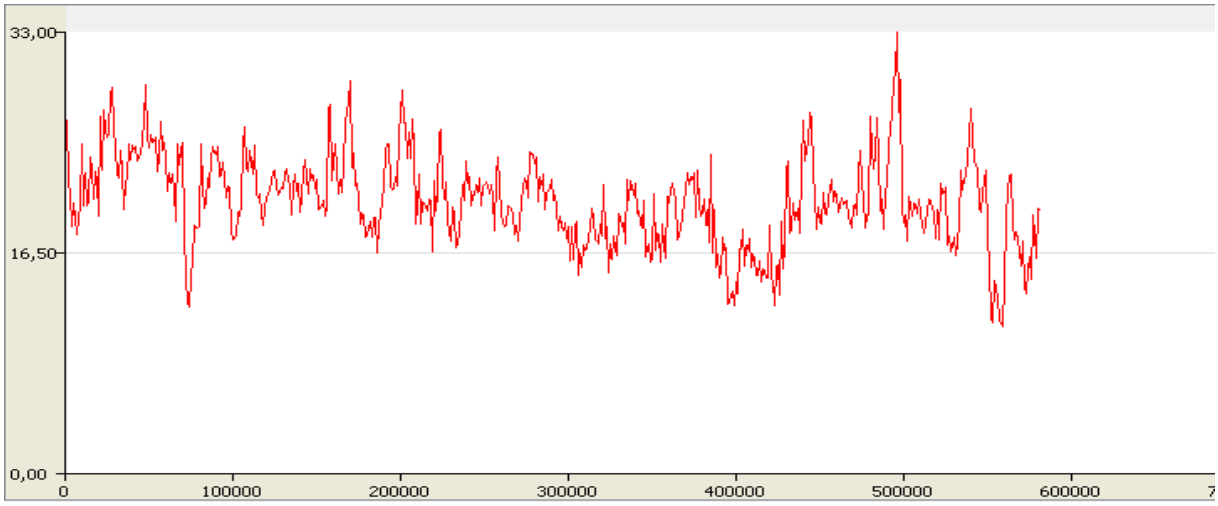
Şekil 4.7: CluStream algoritması F1-Kesinlik grafiği.

Şekil 4.7' de görüldüğü üzere, CluStream algoritması ile akan veri üzerinde yapılan kümelemede ortalama 0,44 F1-Kesinlik değerine ulaşılmıştır. Bu düşük değer, algoritmanın değişen ortam koşullarına uyum sağlayamadığını göstermektedir.



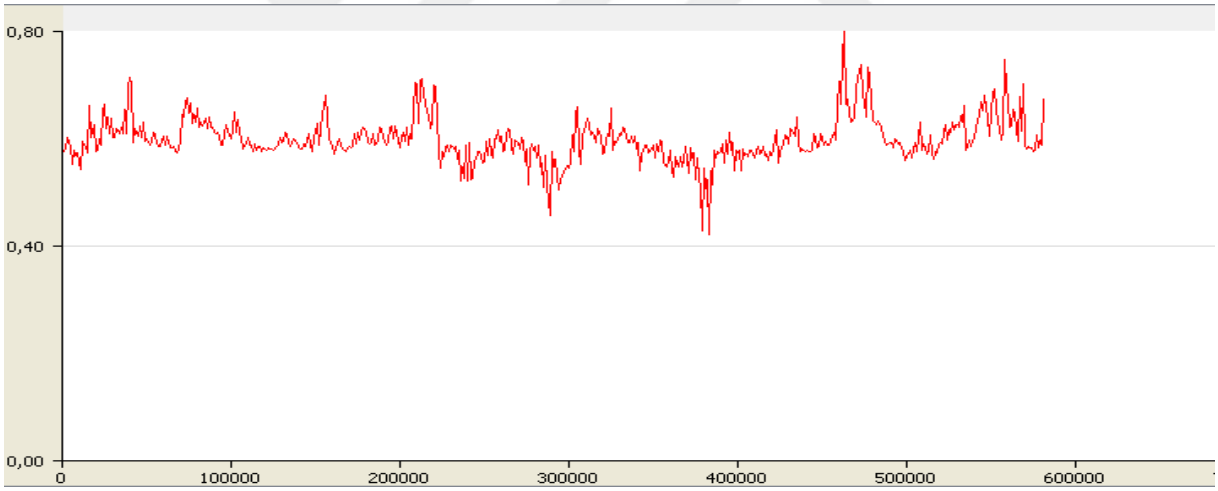
Şekil 4.8: CluStream algoritması F1-Hassaslık grafiği.

Şekil 4.8' de görüldüğü üzere, CluStream algoritması ile akan veri üzerinde yapılan kümelemede ortalama 0,41 F1-Hassaslık değerine ulaşılmıştır.



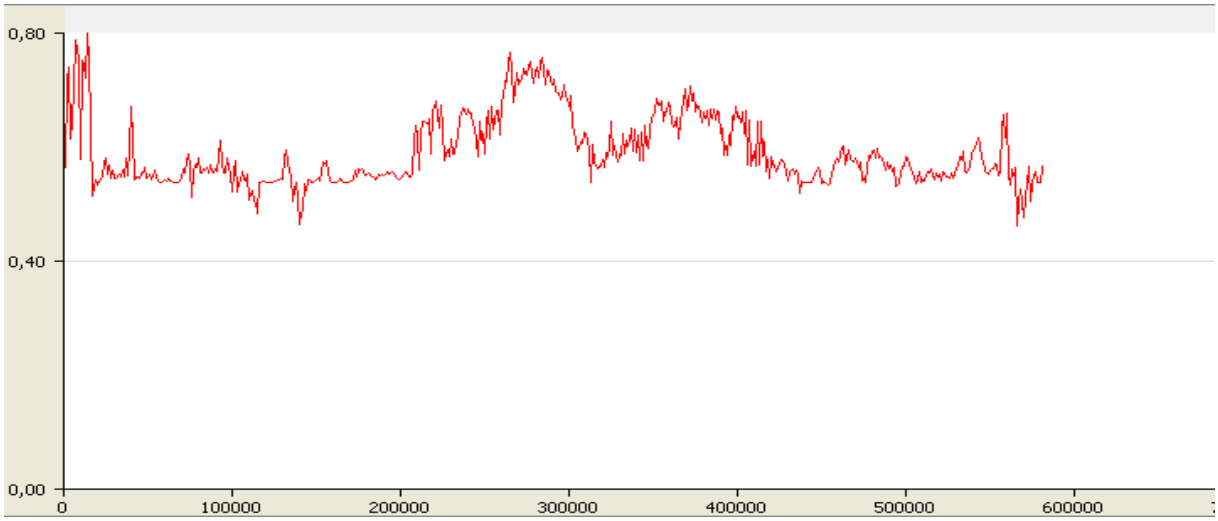
Şekil 4.9: CluStream algoritması hata kareleri toplamı grafiği.

Şekil 4.9' da görüldüğü üzere, CluStream algoritması için bulunan hata kareleri toplamı değeri, ortalama 20,23' tür. Bu değerin yükselmesi, yapılan kümelemenin başarısının düştüğünü göstermektedir.



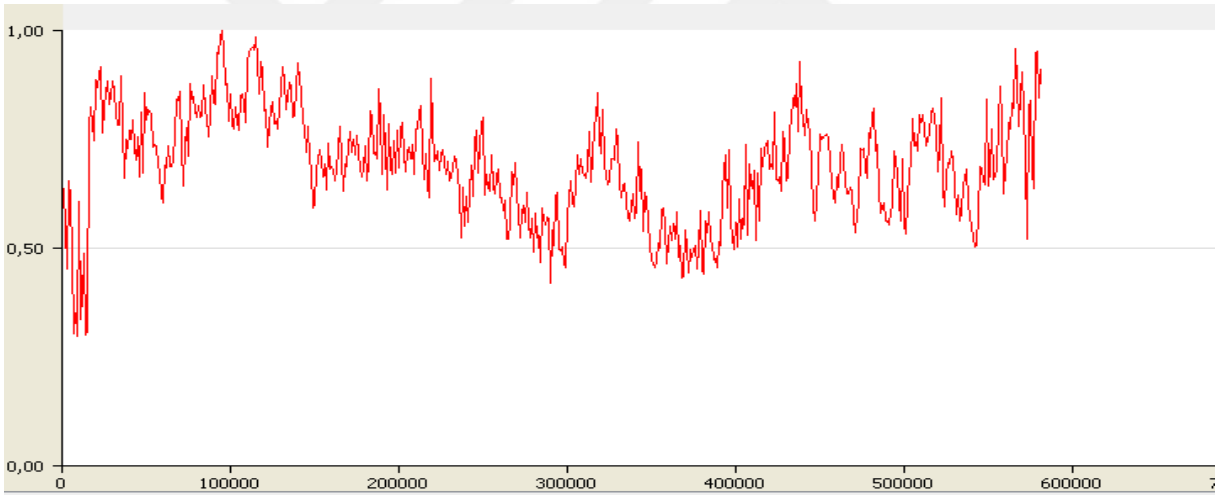
Şekil 4.10: CluStream algoritması Silhouette indeksi grafiği.

Şekil 4.10' da görüldüğü üzere, CluStream algoritması için bulunan Silhouette indeksi değeri, ortalama 0,60' tır. Silhouette indeksi değerinin yükselmesi, kümeleme başarısının arttığını göstermektedir.



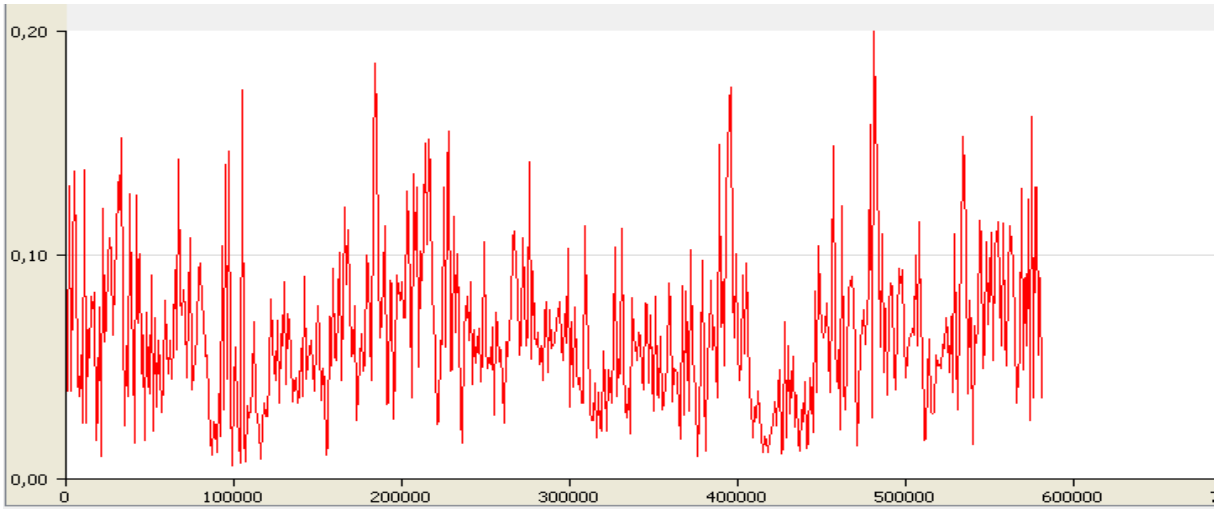
Şekil 4.11: CluStream algoritması Rand indeksi grafiği.

Şekil 4.11' de görüldüğü üzere, CluStream algoritması için bulunan Rand indeksi değeri, ortalama 0,55' tir.



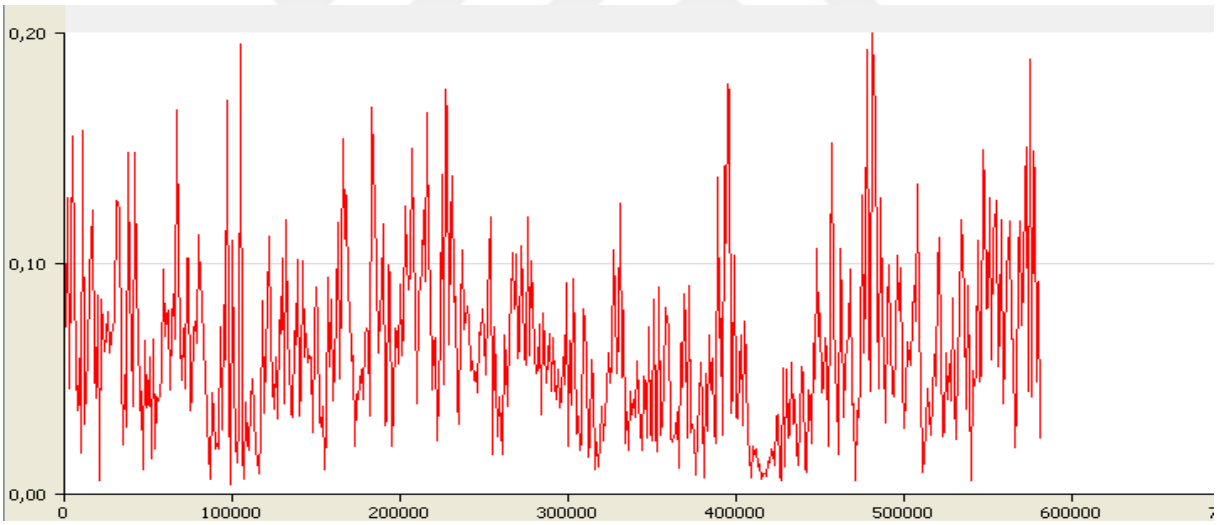
Şekil 4.12: CluStream algoritması saflık grafiği.

Şekil 4.12' de görüldüğü üzere, CluStream algoritması için hesaplanan saflık değeri, ortalama 0,66' dır. Saflık değeri, kümede bulunan baskın elemanların oranını ifade etmektedir.



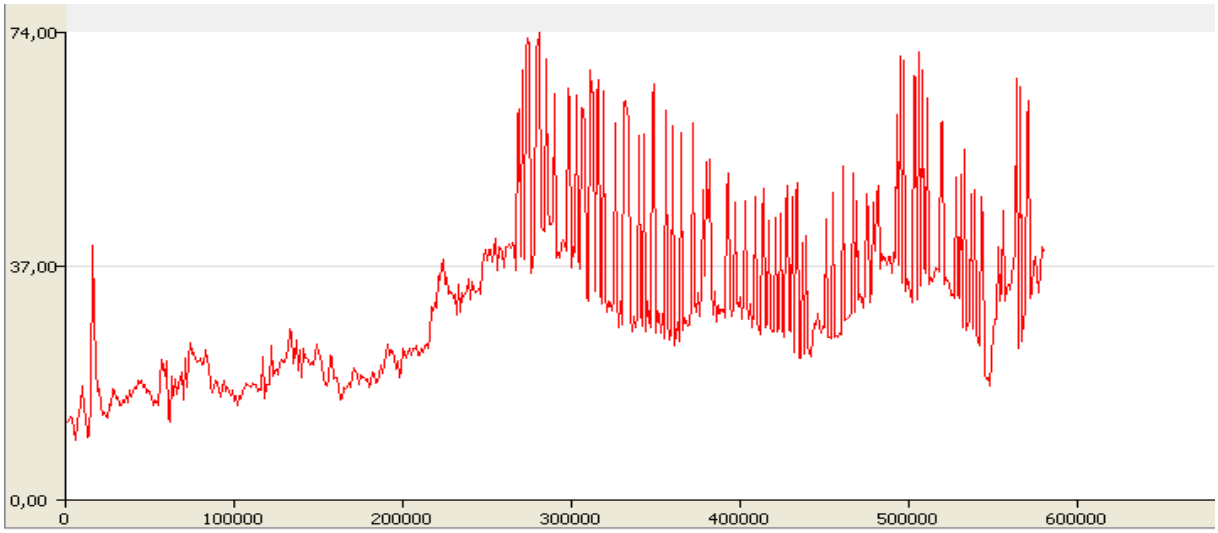
Şekil 4.13: StreamKM++ algoritması F1-Kesinlik grafiği.

Şekil 4.13' de görüldüğü üzere, StreamKM++ algoritması ile akan veri üzerinde yapılan kümelemede ortalama 0,09 F1-Kesinlik değerine ulaşılmıştır. Bu düşük değer, algoritmanın değişen ortam koşullarına uyum sağlayamadığını göstermektedir.



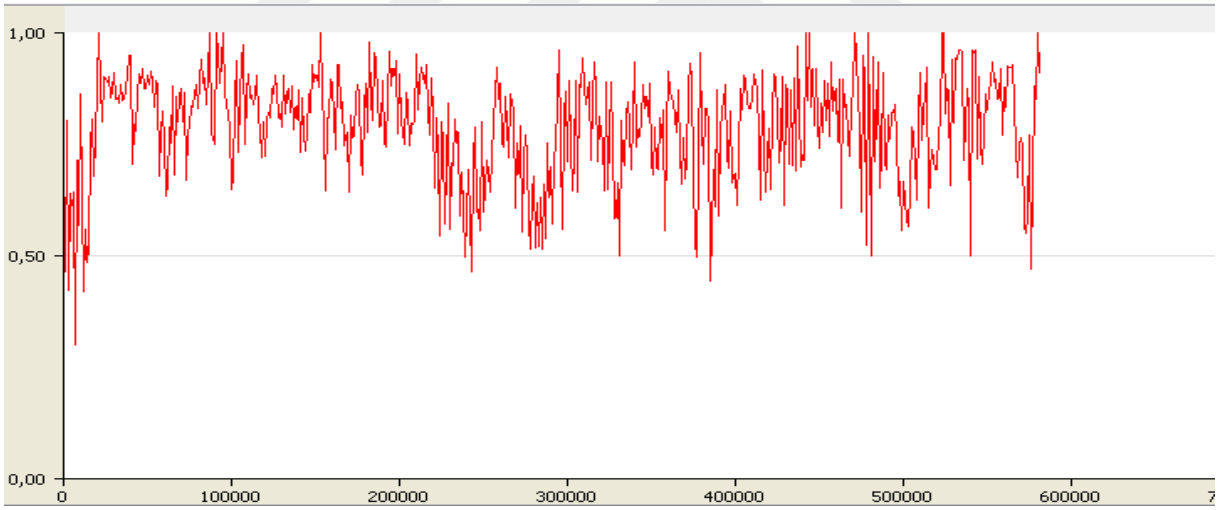
Şekil 4.14: StreamKM++ algoritması F1-Hassaslık grafiği.

Şekil 4.14' te görüldüğü üzere, StreamKM++ algoritması ile akan veri üzerinde yapılan kümelemede ortalama 0,08 F1-Hassaslık değerine ulaşılmıştır.



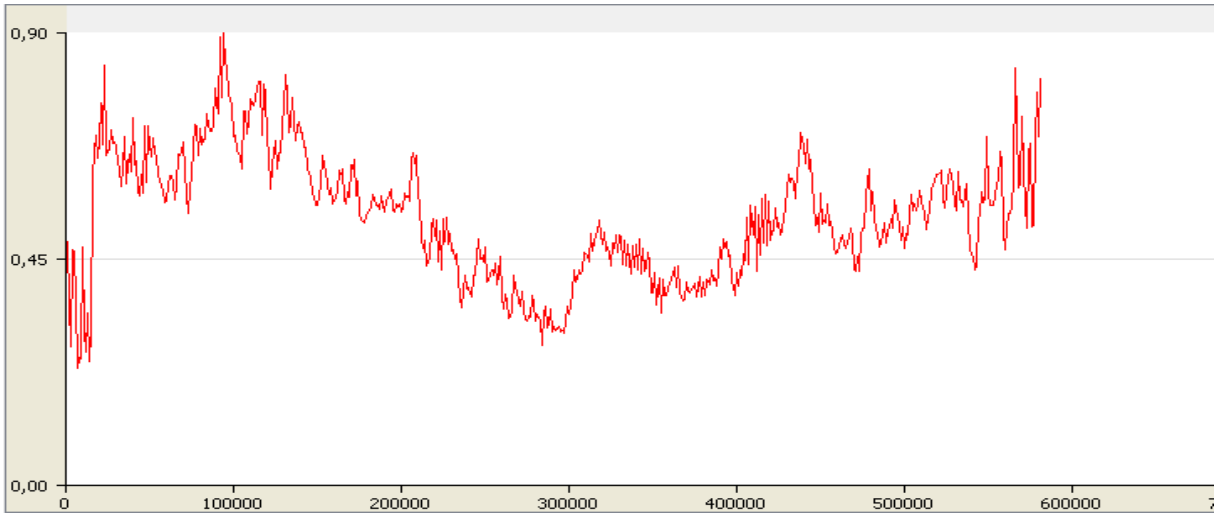
Şekil 4.15: StreamKM++ algoritması hata kareleri toplamı grafiği.

Şekil 4.15' te görüldüğü üzere, StreamKM++ algoritması için bulunan hata kareleri toplamı değeri, ortalama 42,35' tir. Bu değerin yükselmesi, yapılan kümelemenin başarısının düştüğünü göstermektedir.



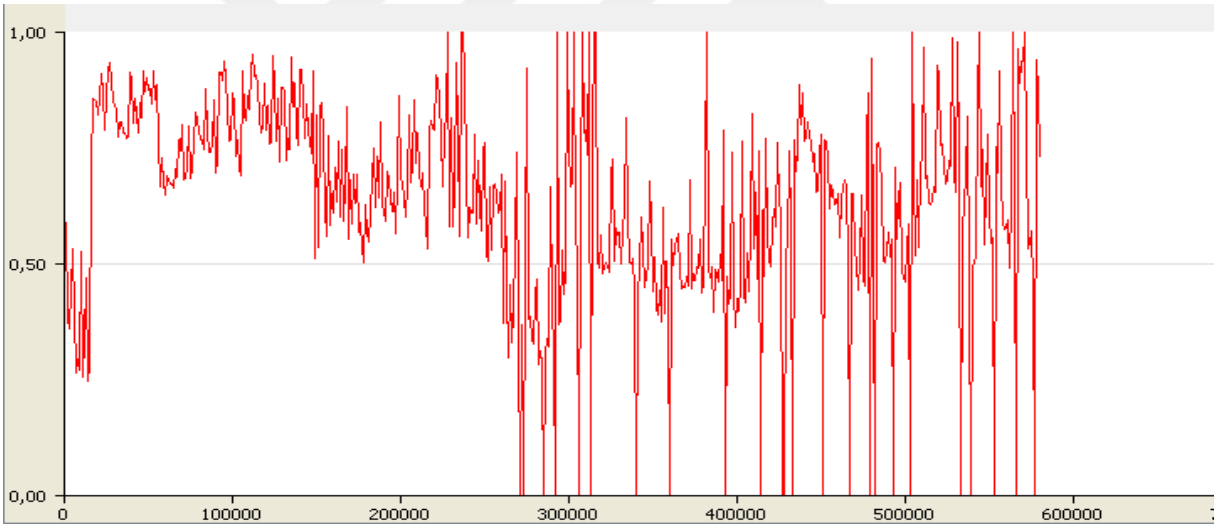
Şekil 4.16: StreamKM++ algoritması Silhouette indeksi grafiği.

Şekil 4.16' da görüldüğü üzere, StreamKM++ algoritması için bulunan Silhouette indeksi değeri, ortalama 0,65' tir. Silhouette indeksi değerinin yükselmesi, kümeleme başarısının arttığını göstermektedir.



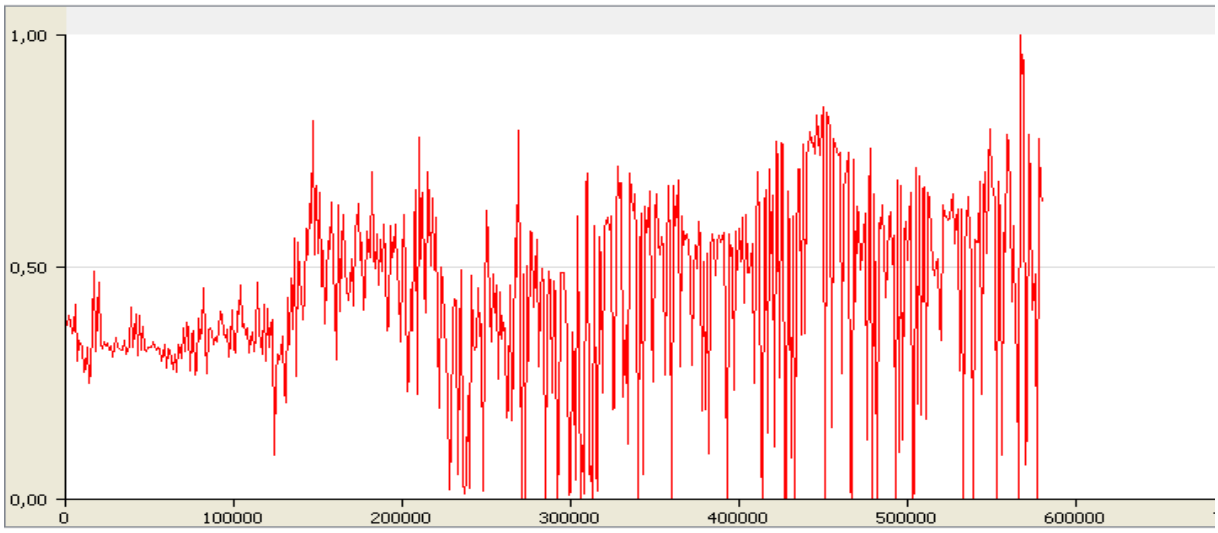
Şekil 4.17: StreamKM++ algoritması Rand indeks grafiği.

Şekil 4.17' de görüldüğü üzere, StreamKM++ algoritması için bulunan Rand indeksi değeri, ortalama 0,52' dir.



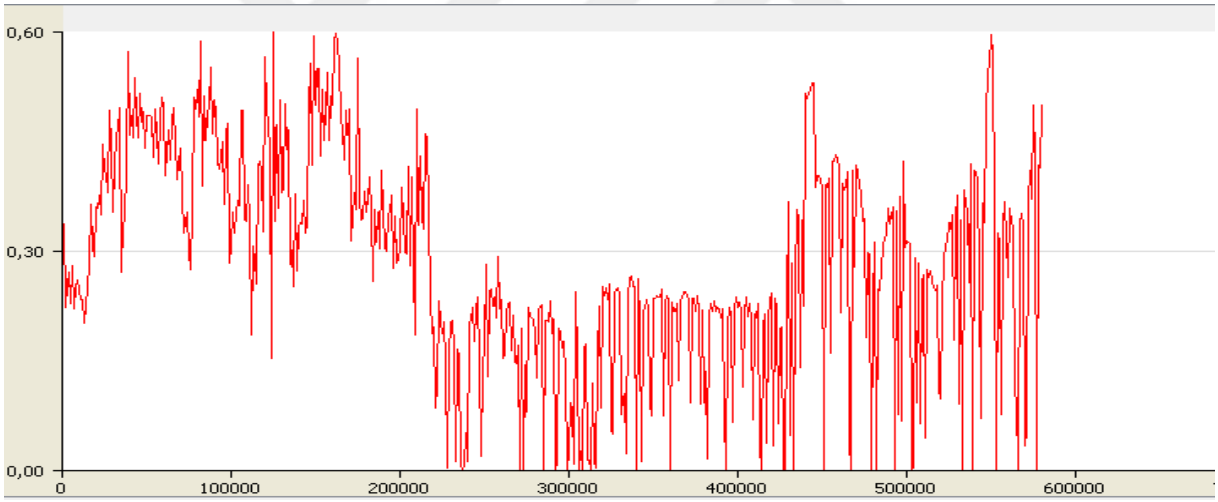
Şekil 4.18: StreamKM++ algoritması saflık grafiği.

Şekil 4.18' de görüldüğü üzere, StreamKM++ algoritması için hesaplanan saflık değeri, ortalama 0,62' dir. Saflık değeri, kümede bulunan baskın elemanların oranını ifade etmektedir.



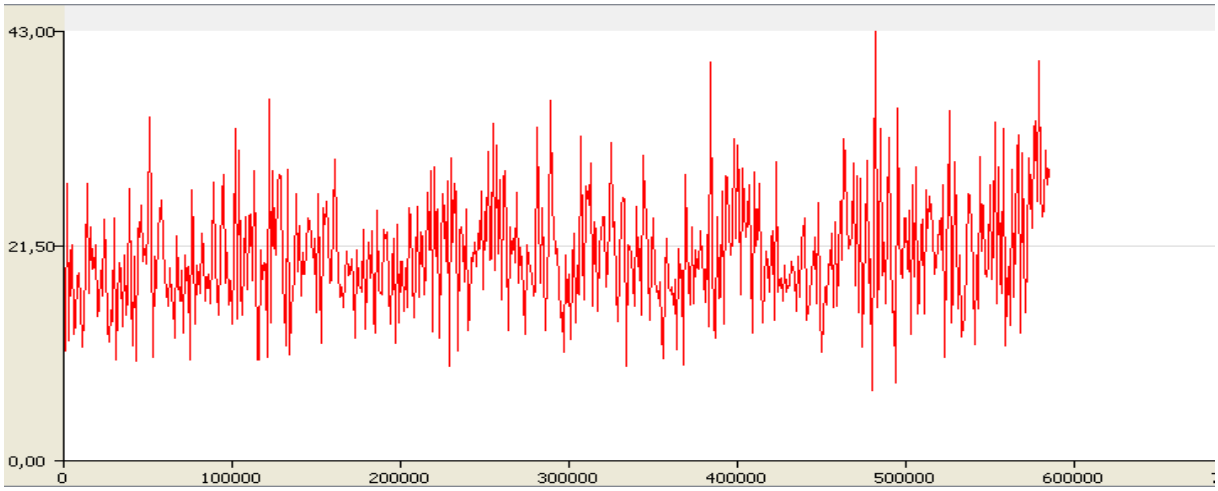
Şekil 4.19: ClusTree algoritması F1-Kesinlik grafiği.

Şekil 4.19' da görüldüğü üzere, ClusTree algoritması ile akan veri üzerinde yapılan kümelemede ortalama 0,40 F1-Kesinlik değerine ulaşılmıştır.



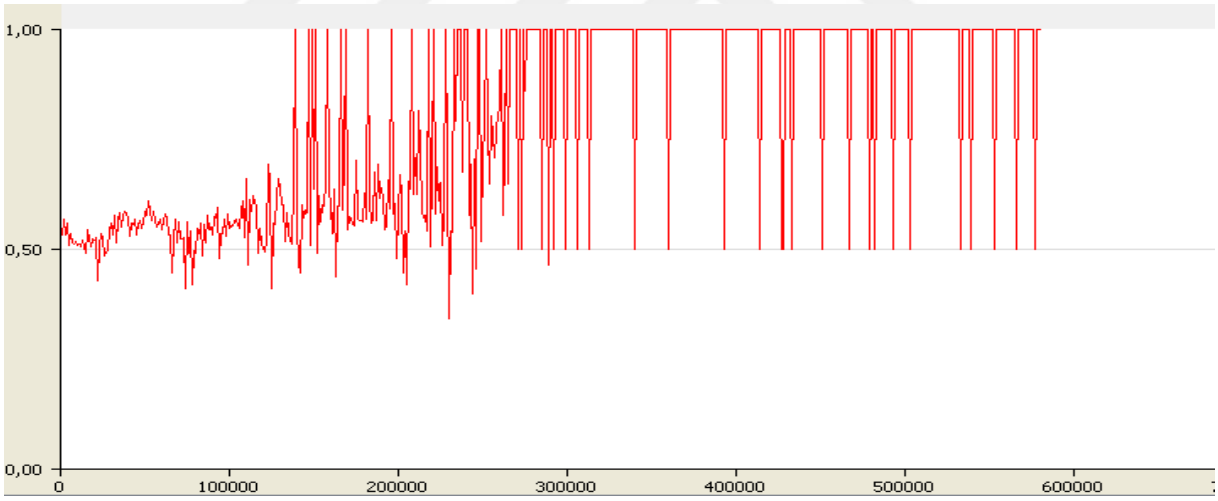
Şekil 4.20: ClusTree algoritması F1-Hassaslık grafiği.

Şekil 4.20' de görüldüğü üzere, ClusTree algoritması ile akan veri üzerinde yapılan kümelemede ortalama 0,28 F1-Hassaslık değerine ulaşılmıştır.



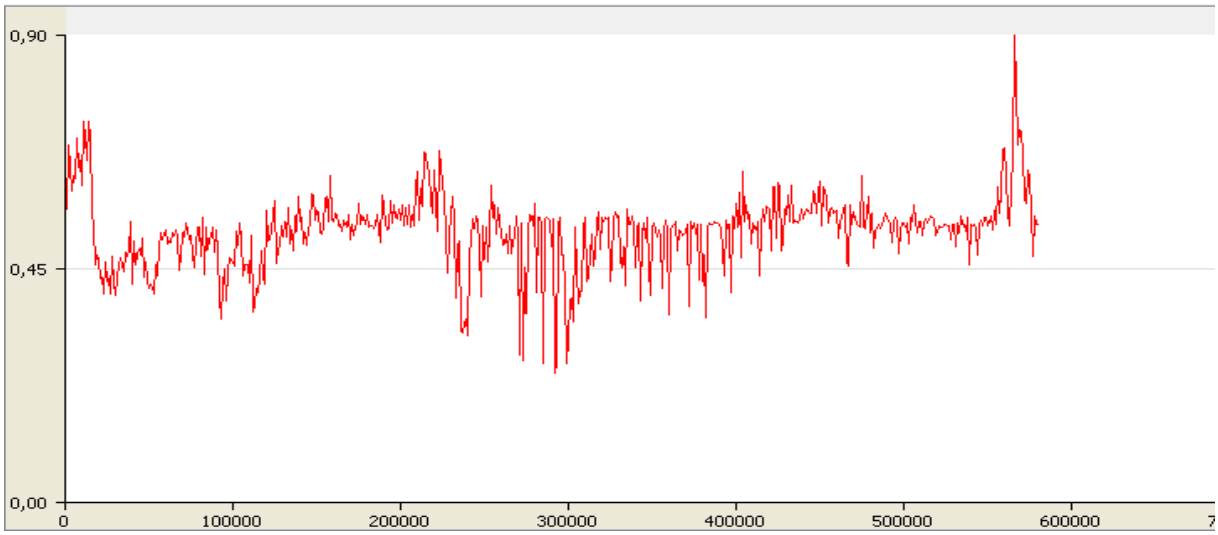
Şekil 4.21: ClusTree algoritması hata kareleri toplamı grafiği.

Şekil 4.21' de görüldüğü üzere, ClusTree algoritması için bulunan hata kareleri toplamı değeri, ortalama 25,75' tir. Bu değerin yükselmesi, yapılan kümelemenin başarısının düştüğünü göstermektedir. ClusTree algoritması bu çalışmada, DenStream algoritmasına yakın performans göstermiştir.



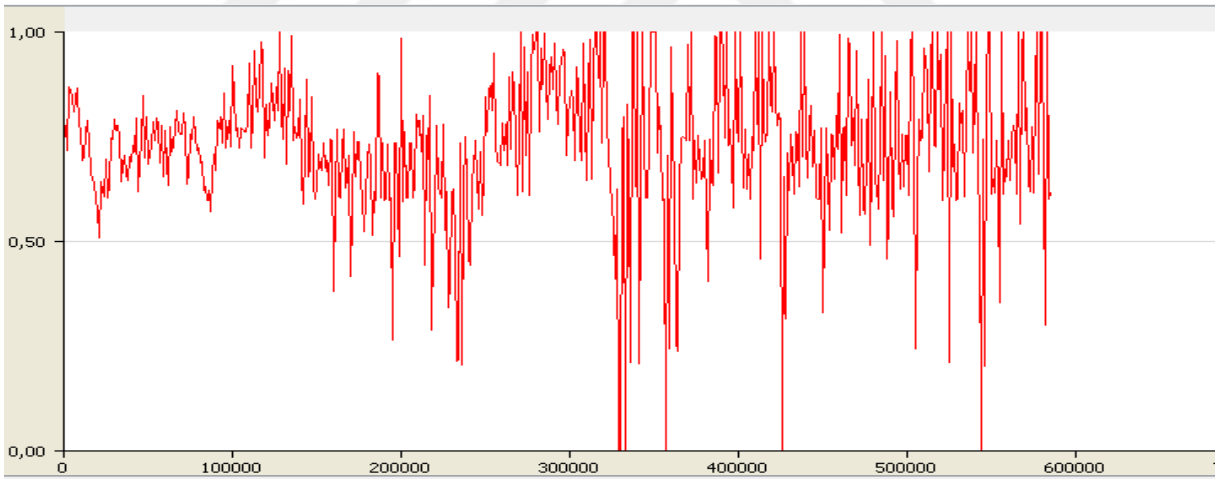
Şekil 4.22: ClusTree algoritması Silhouette indeksi grafiği.

Şekil 4.22' de görüldüğü üzere, ClusTree algoritması için bulunan Silhouette indeksi değeri, ortalama 0,70 olarak hesaplanmıştır. Silhouette indeksi değerinin yükselmesi, kümeleme başarısının arttığını göstermektedir.



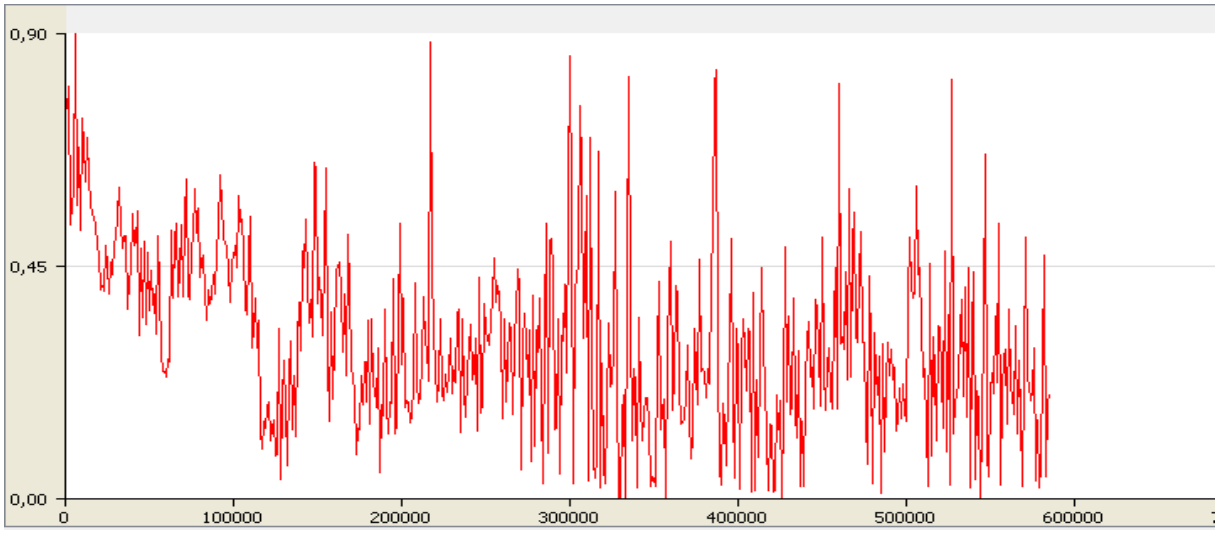
Şekil 4.23: ClusTree algoritması Rand indeks grafiği.

Şekil 4.23’ te görüldüğü üzere, ClusTree algoritması için bulunan Rand indeksi değeri, ortalama 0,51’ dir.



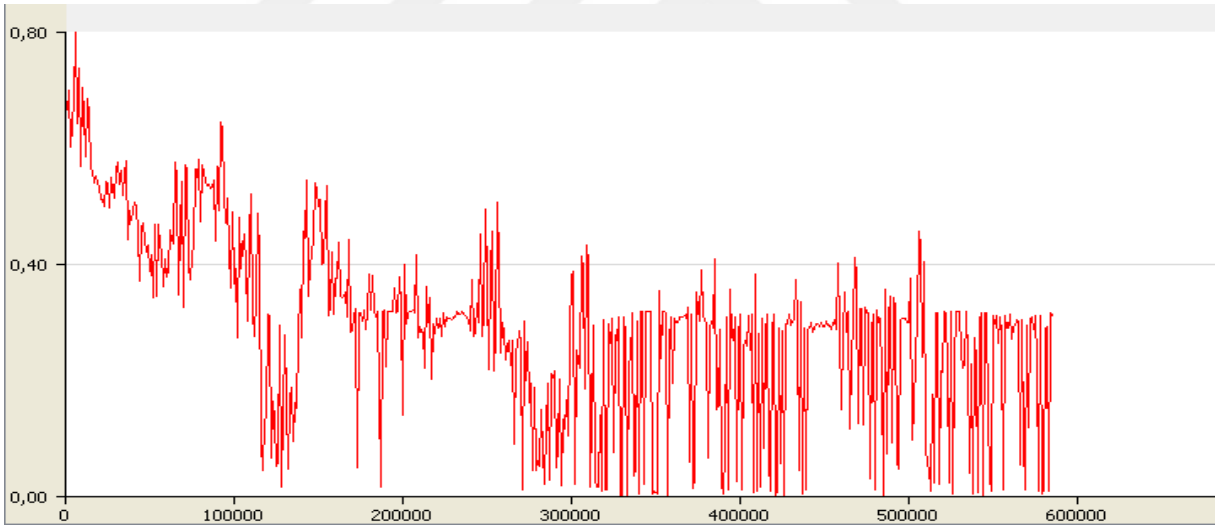
Şekil 4.24: ClusTree algoritması saflık grafiği.

Şekil 4.24’ te görüldüğü üzere, ClusTree algoritması için hesaplanan saflık değeri, ortalama 0,64’ tür. Saflık değeri, kümede bulunan baskın elemanların oranını ifade etmektedir.



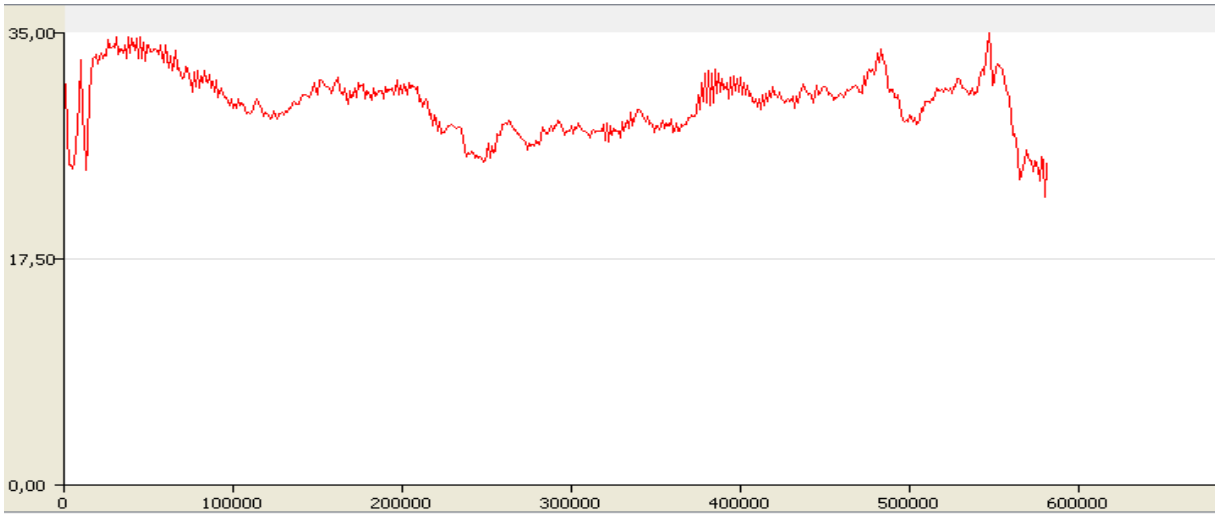
Şekil 4.25: DenStream algoritması F1-Kesinlik grafiği.

Şekil 4.25' te görüldüğü üzere, DenStream algoritması ile akan veri üzerinde yapılan kümelemede ortalama 0,38 F1-Kesinlik değerine ulaşılmıştır. Bu değer, algoritmanın değişen ortam koşullarına uyum sağlayamadığını göstermektedir.



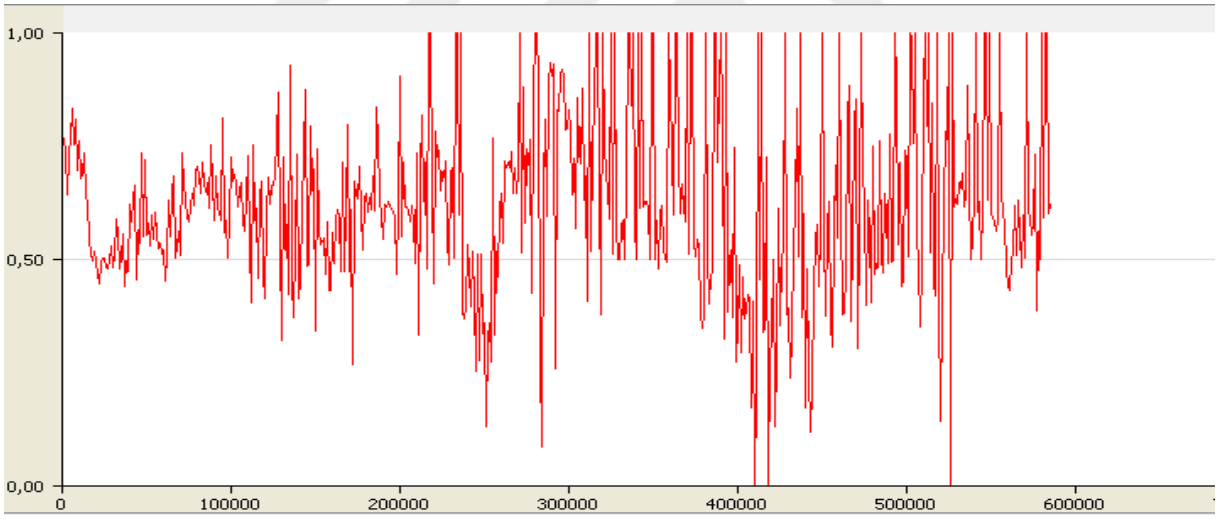
Şekil 4.26: DenStream algoritması F1-Hassaslık grafiği.

Şekil 4.26' da görüldüğü üzere, DenStream algoritması ile akan veri üzerinde yapılan kümelemede ortalama 0,21 F1-Hassaslık değerine ulaşılmıştır.



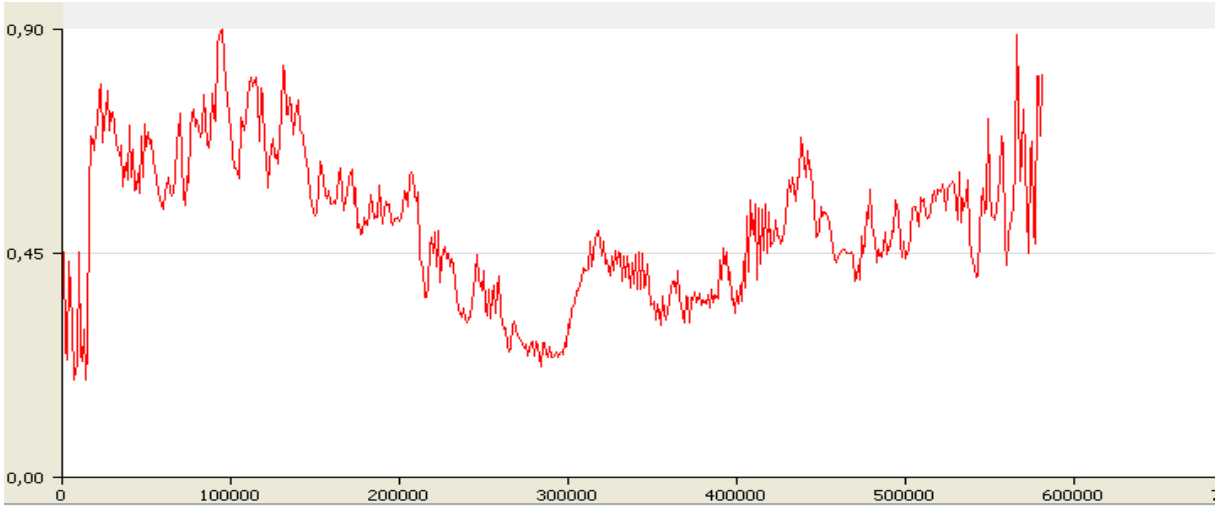
Şekil 4.27: DenStream algoritması hata kareleri toplamı grafiği.

Şekil 4.27' de görüldüğü üzere, DenStream algoritması için bulunan SSE değeri ortalama 29,03' tür. Bu değerin yükselmesi, yapılan kümelemenin başarısının düştüğünü göstermektedir.



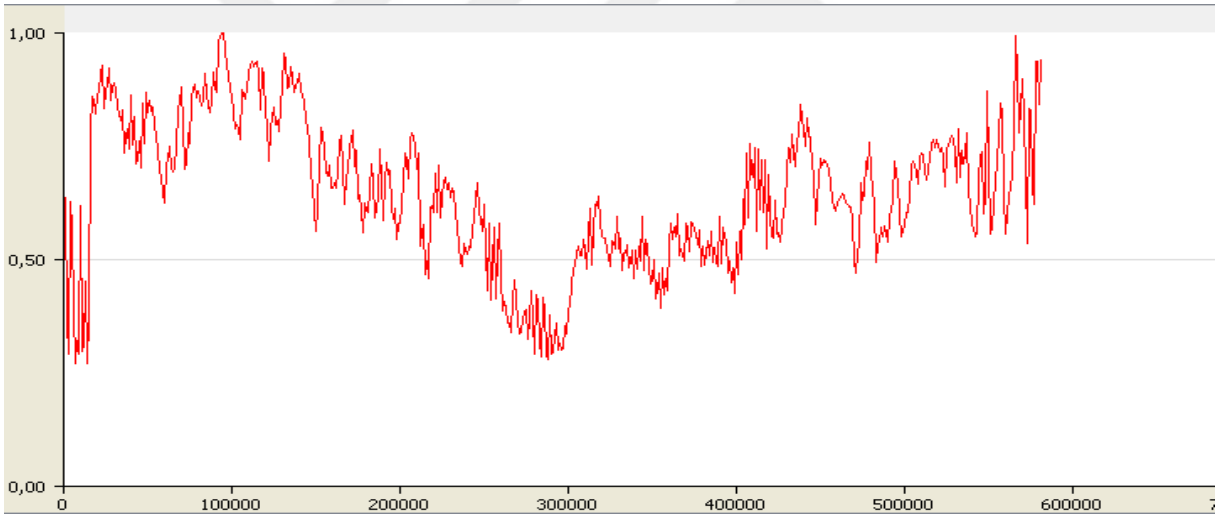
Şekil 4.28: DenStream algoritması Silhouette indeksi grafiği.

Şekil 4.28' de görüldüğü üzere, DenStream algoritması için bulunan Silhouette indeksi değeri ortalama 0,66 olarak hesaplanmıştır. Silhouette indeksi değerinin yükselmesi, kümeleme başarısının arttığını göstermektedir.



Şekil 4.29: DenStream algoritması Rand indeks grafiği.

Şekil 4.29' da görüldüğü üzere, DenStream algoritması için bulunan Rand indeksi değeri ortalama 0,48' dir.



Şekil 4.30: DenStream algoritması saflık grafiği.

Şekil 4.30' da görüldüğü üzere, DenStream algoritması için hesaplanan saflık değeri ortalama 0,62' dir. Saflık değeri, kümede bulunan baskın elemanların oranını ifade etmektedir.

4.2.1.2. Yapay Veri Setleri ile Değerlendirme

Tez çalışmamızdaki önceki kısımlarda anlatılan, oluşturulan farklı durumlar için verilen yapay veri seti üzerinde CluStream-Kmeans++, CluStream, ClusTree, DenStream, StreamKM++ algoritmaları çalıştırılmış ve üretilen sonuçların karşılaştırması yapılmıştır. Bu çalışma kısmında, rastgele oluşturulan 1.200.000 örnekten oluşan veri seti üzerinde çalıştırılan algoritmalar, detayları önceki kısımda verilen altı değerlendirme kriterine göre performansları

değerlendirilmiştir. Bu yapay veri setini, akan veri olarak sisteme gönderebilmek için uygulama tarafında algoritmalara 1500/periyot olmak üzere veri gönderilmiştir.

Tablo 4.5: Durum 1 parametreleri ile sonuçlar.

Algoritma	F1-P	F1-R	SSE	Silhouette İndeksi	Rand İndeks	Saflık
CluStream-Kmeans++	0,79	0,73	8,44	0,80	0,91	0,88
CluStream	0,69	0,68	10,69	0,74	0,87	0,84
ClusTree	0,76	0,71	9,22	0,78	0,89	0,87
DenStream	0,61	0,61	10,19	0,76	0,79	0,85
StreamKM++	0,39	0,27	43,1	0,73	0,56	0,85

Tablo 4.5’ te algoritmalar, 1.200.000 veri örneği üzerinde durum 1’ e göre çalıştırılmıştır. Tablo 4.5’ te görüldüğü üzere, altı farklı değerlendirme indeksi kullanılmış ve bu indekslerde tez çalışmamızda kullanılan CluStream-Kmeans++ algoritması en iyi sonuçları üretmiştir.

Tablo 4.6: Durum 2 parametreleri ile sonuçlar.

Algoritma	F1-P	F1-R	SSE	Silhouette İndeksi	Rand İndeks	Saflık
CluStream-Kmeans++	0,74	0,72	6,85	0,73	0,91	0,86
CluStream	0,69	0,59	13,49	0,62	0,82	0,75
ClusTree	0,70	0,70	7,37	0,71	0,90	0,85
DenStream	0,49	0,54	9,39	0,64	0,75	0,84
StreamKM++	0,40	0,22	83,68	0,64	0,51	0,70

Tablo 4.6’ da algoritmalar, 1.200.000 veri örneği üzerinde durum 2’ ye göre çalıştırılmıştır. Durum 2’ de, durum 1’ den farklı olarak küme sayısı 7 olarak belirlenmiştir. Durum 1’ e göre incelendiğinde; aynı alanda daha fazla küme oluşması, birbirlerine daha yakın kümelerin oluşmasına neden olmuştur. Bu durumda, kümeleri ayırt etmek zorlaşmış ve algoritmaların değerlendirme çıktılarına yansımıştır.

Tablo 4.7: Durum 3 parametreleri ile sonuçlar.

Algoritma	F1-P	F1-R	SSE	Silhouette İndeksi	Rand İndeks	Saflık
CluStream-Kmeans++	0,69	0,65	15,23	0,69	0,85	0,81
CluStream	0,62	0,61	16,93	0,66	0,79	0,79
ClusTree	0,68	0,64	14,67	0,69	0,86	0,81
DenStream	0,46	0,55	15,54	0,62	0,72	0,91
StreamKM++	0,21	0,14	99,28	0,66	0,38	0,84

Tablo 4.7’ de, yapay veri üzerinde algoritmalar durum 3’ e göre çalıştırılmıştır. Yüksek gürültü oranıyla birlikte algoritmaların yüksek gürültüye karşı dirençleri gözlemlenmiştir. Değerlendirme sonuçlarına göre, algoritmaların gürültü sebebiyle değerlendirme değerlerinin azaldığı görülmüştür. Ayrıca ClusStream-Kmeans++ algoritmasının gürültülü ortamlara diğer algoritmalara göre daha kolay uyumlandığı görülmüştür.

Tablo 4.8: Durum 4 parametreleri ile sonuçlar.

Algoritma	F1-P	F1-R	SSE	Silhouette İndeksi	Rand İndeks	Saflık
CluStream-Kmeans++	0,99	0,82	7,79	0,98	0,99	0,99
CluStream	0,73	0,45	24,31	0,95	0,65	0,98
ClusTree	0,98	0,78	7,61	0,98	0,98	0,98
DenStream	0,45	0,53	71,34	0,87	0,67	0,97
StreamKM++	0,39	0,14	48,53	0,82	0,39	0,61

Tablo 4.8’ de, yapay veri üzerinde algoritmalar durum 4’ e göre çalıştırılmıştır. Durum 4’ te, boyut sayısı 2’ den 10’ a yükseltilmiştir. Boyut sayısının artmasıyla birlikte çalışma uzayı sınırları da genişlemiştir. CluStream-Kmeans++ ve ClusTree algoritmalarının doğruluk değerlerinin arttığı gözlenmiştir. Boyut sayısının artmasıyla birlikte yapay verilerde, aynı kümede bulunan veriler arasındaki bağ artarken, farklı kümede yer alan verilerin bağı zayıflamaktadır. Bu sayede, uyumsuz bir verinin küme tarafından sahiplenilme ihtimali azalmaktadır. Bu sayede algoritmalar, farklı veri gruplarını daha rahat tespit edebilmiştir.

Tablo 4.9: Durum 5 parametreleri ile sonuçlar.

Algoritma	F1-P	F1-R	SSE	Silhouette İndeksi	Rand İndeks	Saflık
CluStream-Kmeans++	0,74	0,69	10,14	0,75	0,88	0,84
CluStream	0,67	0,66	11,37	0,71	0,83	0,82
ClusTree	0,70	0,67	10,91	0,73	0,86	0,83
DenStream	0,27	0,47	9,63	0,48	0,68	0,91
StreamKM++	0,26	0,23	10,51	0,60	0,47	0,74

Tablo 4.9’ da, yapay veri üzerinde algoritmalar durum 5’ e göre çalıştırılmıştır. Durum 5’ te, kümelerin yarıçap değerleri 0,07’ den 0,10 ’a çıkarılarak daha büyük kümelerin bulunması beklenmiştir. Bu çalışmada, çalışılan tüm algoritmalarda doğruluk değerlendirme sonuçlarında önemli ölçüde azalma gözlemlenmiştir. Bu sonuçlar, durum 2’ deki değerlendirme sonuçlarıyla benzerdir. Aynı alanda daha geniş ve birbirine yakın kümelerin olması, kümelerin birbirinden ayrımını zorlaştırmıştır. Bu durum, değerlendirme sonuçlarına kötü yönde etki etmiştir.

Tablo 4.10: Durum 6 parametreleri ile sonuçlar.

Algoritma	F1-P	F1-R	SSE	Silhouette İndeksi	Rand İndeks	Saflık
CluStream-Kmeans++	0,72	0,66	8,35	0,72	0,86	0,82
CluStream	0,66	0,62	13,62	0,68	0,84	0,80
ClusTree	0,70	0,64	11,41	0,71	0,86	0,82
DenStream	0,24	0,47	8,75	0,63	0,69	0,91
StreamKM++	0,25	0,16	61,3	0,67	0,39	0,83

Tablo 4.10’ da, algoritmalar yapay veri seti üzerinde durum 6’ ya göre çalıştırılmıştır. Durum 6’ da, oluşturulan kümeler sabit boyutlu kümeler yerine değişken boyutludur. Bu test çalışması ile algoritmaların farklı boyutlu kümeleri tespit etme durumları gözlemlenmiştir. Tez çalışmamızda kullandığımız CluStream-Kmeans++ algoritması; F1-P, F1-R, SSE, Silhouette indekslerinde, DenStream algoritması ise saflık indeksinde daha iyi sonuçlar üretmiştir. DenStream algoritması çok sayıda kümeyi keşfetmeye odaklandığından saflık değerlendirme ölçütüyle iyi sonuçlar üretmesi normal bir durumdur.

Tablo 4.11: Durum 7 parametreleri ile sonuçlar.

Algoritma	F1-P	F1-R	SSE	Silhouette İndeksi	Rand İndeks	Saflık
CluStream-Kmeans++	0,68	0,65	8,39	0,71	0,88	0,85
CluStream	0,66	0,68	10,69	0,69	0,85	0,84
ClusTree	0,66	0,64	8,95	0,70	0,87	0,84
DenStream	0,48	0,52	10,45	0,68	0,75	0,78
StreamKM++	0,21	0,12	8,48	0,66	0,42	0,80

Tablo 4.11’ de, algoritmalar yapay veri seti üzerinde durum 7’ ye göre çalıştırılmıştır. Durum 7’ de, her 25.000 verilik periyotlarda ayrılma veya birleşme olayı gerçekleşecektir. Başlangıç değeri 5 olan küme sayısı değeri, +4 -4 aralığında oynama gösterebilecektir. Bu test çalışmasıyla, algoritmaların değişken ortam şartlarına uyumlanmadaki başarı durumu gözlemlenmiştir. Tablo 4.11’ de görüldüğü üzere, CluStream algoritması değişen ortam şartlarında diğer algoritmalara göre daha başarılı uyumlanarak doğruluk değerlendirme değerlerinde üstünlük sağlamıştır. CluStream-Kmeans++, CluStream ve ClusTree algoritmaları, tüm çalışma uzayında gerçekleşen değişimleri algılayarak önemli sonuçlar elde etmiştir.

Tablo 4.12: Durum 8 parametreleri ile sonuçlar.

Algoritma	F1-P	F1-R	SSE	Silhouette İndeksi	Rand İndeks	Saflık
CluStream-Kmeans++	0,86	0,71	5,36	0,86	0,89	0,90
CluStream	0,74	0,62	8,42	0,83	0,81	0,82
ClusTree	0,84	0,64	8,63	0,86	0,88	0,88
DenStream	0,66	0,61	14,6	0,85	0,83	0,81
StreamKM++	0,35	0,18	9,31	0,83	0,59	0,81

Tablo 4.12’ de, algoritmalar yapay veri seti üzerinde durum 8’ e göre çalıştırılmıştır. Durum 8 ’ de, her 25.000 verilik periyotlarda silme ve oluşturma olayı gerçekleşmektedir. Küme sayısı 5 olarak başlayıp +4 -4 olabilecektir. Bu nedenle, durum 7’ deki oluşan sonuçlarla, durum 8 sonuçları benzerlik göstermektedir. CluStream ve ClusTree algoritmalarının değişen ortam şartlarına daha kolay uyumlandığı görülmektedir.

Tablo 4.13: Durum 9 parametreleri ile sonuçlar.

Algoritma	F1-P	F1-R	SSE	Silhouette İndeksi	Rand İndeks	Saflık
CluStream-Kmeans++	0,74	0,70	7,71	0,71	0,90	0,86
CluStream	0,68	0,60	12,33	0,60	0,81	0,79
ClusTree	0,52	0,68	8,27	0,70	0,89	0,85
DenStream	0,50	0,54	10,10	0,65	0,75	0,72
StreamKM++	0,35	0,19	7,95	0,67	0,47	0,83

Tablo 4.13’ de, tez çalışmamızda kullanılan CluStream-Kmeans++ algoritması ve literatürdeki algoritmaların, oluşturulan durum 9 verileriyle çalıştırıldığında gösterdiği performans ölçümü verilmiştir. Durum 9, durum 7 ve durum 8 ‘in birleştirilmesi gibi düşünülebilir. Bu aşamada, her 25.000 veride bir oluşturma, silme, ayırma ya da birleştirme olayı gerçekleşecektir. Algoritmaların çalışma sonuçlarına bakıldığında CluStream-Kmeans++ algoritmasının daha iyi sonuçlar ürettiği gözlemlenmiştir. CluStream-Kmeans++ algoritması, özellikle F1-P ve Silhouette indeks değerlerinde üstünlüğü ön plana çıkmıştır. Bu durum, CluStream-Kmeans++ algoritmasının eski küme elemanlarını hızlı unutup yeni elemanlara daha rahat uyum sağlamasıyla ilgilidir. CluStream-Kmeans++ algoritması, hızlı değişen ortam şartlarına daha rahat uyumlandığı görülmüştür.

Tablo 4.14: Durum 10 parametreleri ile sonuçlar.

Algoritma	F1-P	F1-R	SSE	Silhouette İndeksi	Rand İndeks	Saflık
CluStream-Kmeans++	0,99	0,77	9,10	0,99	0,98	0,99
CluStream	0,66	0,62	13,62	0,72	0,79	0,80
ClusTree	0,98	0,75	9,68	0,99	0,97	0,98
DenStream	0,68	0,58	10,12	0,98	0,74	0,98
StreamKM++	0,29	0,08	53,64	0,71	0,30	0,53

Tablo 4.14’ de, tez çalışmamızda kullanılan CluStream-Kmeans++ algoritması ve literatürdeki algoritmaların, oluşturulan durum 10 verileriyle çalıştırıldığında gösterdiği performans ölçümü verilmiştir. Durum 10’ da , durum 4 ve durum 6’ daki çalışmadaki parametrelerin birleşimi kullanılmıştır. Boyut sayısı 10, küme yarıçapı sapması 0,4 olarak çalıştırılmıştır. Boyut sayısının artmasıyla birlikte, algoritmaların iş kolaylaşmış ve DenStream haricindeki algoritmalarda 2 boyutlu çalışmalara göre daha başarılı sonuçlar elde edilmiştir.

Tablo 4.15: Tüm durumların ortalamaları.

Algoritma	F1-P	F1-R	SSE	Silhouette İndeksi	Rand İndeks	Saflık
CluStream-Kmeans++	0,79	0,71	8,73	0,79	0,81	0,88
CluStream	0,68	0,61	13,54	0,72	0,80	0,82
ClusTree	0,75	0,68	9,67	0,78	0,89	0,87
DenStream	0,48	0,54	17,01	0,71	0,73	0,86
StreamKM++	0,31	0,17	42,57	0,69	0,44	0,75

Tablo 4.15’ de, yapay veriler üzerinde oluşturulan 10 farklı duruma göre gözlemlenen doğruluk değerlendirme ölçümlerinin ortalamaları alınmıştır. Bulunan değerlere bakıldığında CluStream-Kmeans++ algoritmasının, literatürdeki diğer algoritmalara karşı F1-P, F1-R, SSE, Silhouette indeksi ve saflık ölçütleriyle değerlendirmesinde daha iyi sonuçlara ulaştığı gözlemlenmiştir. Rand indeks ölçütüyle yapılan değerlendirmede ClusTree algoritması ve sonrasında CluStream-Kmeans++ algoritmasının daha iyi sonuçlar ürettiği görülmüştür.

4.2.2. Algoritma Çalışma Zamanı Değerlendirmesi

Bu bölümde, tez çalışmasında geliştirilen CluStream-Kmeans++ algoritması ve literatürde yer alan akan veri kümeleme algoritmalarıyla yapılan gerçek ve yapay veri üzerindeki çalışma zamanı değerlendirmesi yapılmıştır.

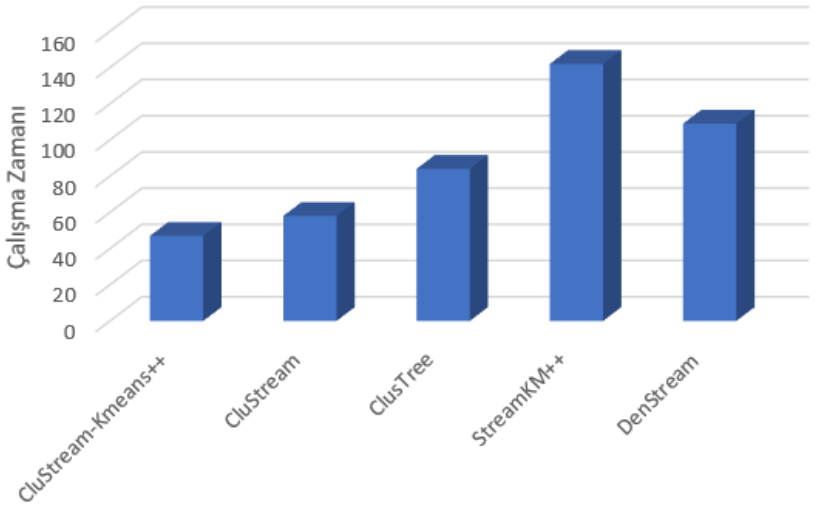
Algoritmaların tutarlı ve doğru sonuçlar üretebilmesiyle birlikte, akan veriyi işlerken harcadıkları süre de önemlidir. Akan verinin anlık değişimi ve üretilen sonuçların kullanımının değerli olması da bu veriyi işlerken geçen sürenin önemini artırmaktadır. Bu kısımda algoritmaların üzerinde çalıştıkları veri tipi ve sayısına göre değerlendirmesi yapılmıştır.

Tez çalışmamızda kullanılan CluStream-Kmeans++ uygulaması ve literatürdeki diğer uygulamaların çoğu çevrimdışı aşamada kümeleme yaptıklarından algoritma işlem süreleri uzamaktadır. Algoritma çalışma sürelerini etkileyen faktörlerden en önemlisi işlenen veri sayısıdır.

Tablo 4.16: Gerçek veri setiyle çalışma süreleri.

Algoritma	Çalışma Zamanı(s)
CluStream-Kmeans++	47
CluStream	58
ClusTree	84
StreamKM++	142
DenStream	109

Tablo 4.16’ da , gerçek veriler üzerinde yapılan çalışmada algoritmaların harcadıkları süreler verilmiştir. İşlenen gerçek verideki gürültü, boyut bilgisi gibi faktörler çalışma süresinin uzamasına sebep olmaktadır. Çalışma sonucunda CluStream-Kmeans++ algoritması çalışma zamanı olarak diğer algoritmalardan daha iyi sonuç göstermiştir. DenStream, StreamKM++ algoritmalarının çevrimdışı kümeleme adımında, çevrimiçi kümeleme adımında bulunan mikro küme sayısına bağlı olarak yaptıkları işlemler çalışma süreleri uzatmaktadır. ClusTree algoritmasında, ağaç yapısıyla hiyerarşik kümeleme yapılması sebebiyle çalışma süresi DenStream ve StreamKM++ algoritmasından daha kısa sürede tamamlanmıştır. Gerçek veriler ile yapılan çalışmanın çalışma süresi karşılaştırması Şekil 4.31’ de verilmiştir.



Şekil 4.31: Gerçek veri ile çalışma zamanı grafiği.

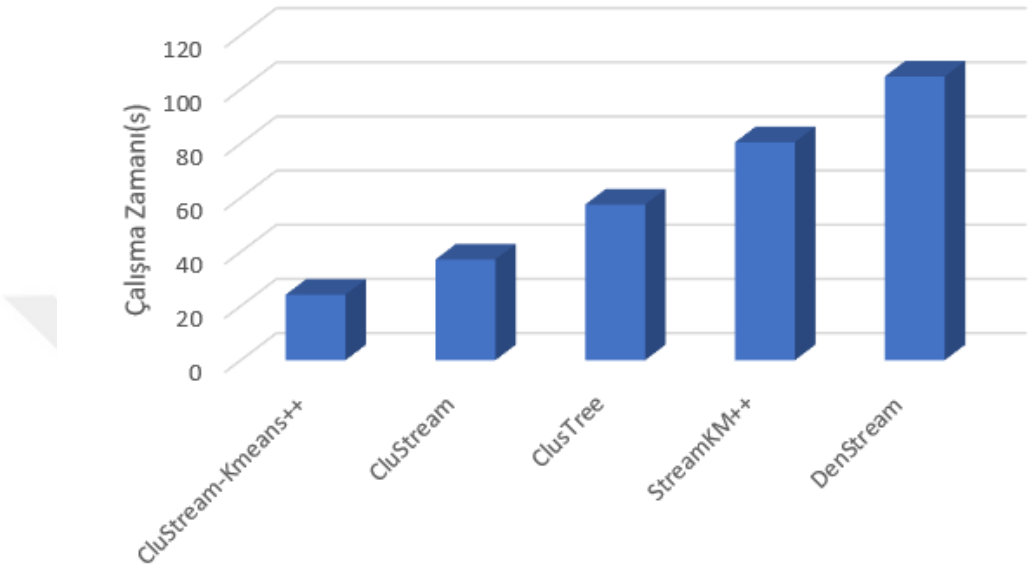
Tez çalışmamızda kullanılan CluStream-Kmeans++ algoritması ve literatürdeki CluStream, StreamKM++, DenStream ve ClusTree algoritmalarının 1.200.000 yapay veriyle 10 farklı durum için çalışması sunucunda elde edilen çalışma süreleri Tablo 4.17' de verilmiştir.

Tablo 4.17: Yapay veri ile çalışma süreleri.

Algoritma	**Durum**										
	1	2	3	4	5	6	7	8	9	10	**Ortalama**
CluStream-Kmeans++	22	26	28	20	23	24	27	24	28	20	24,2
CluStream	35	37	45	34	36	38	36	35	42	34	37,2
ClusTree	54	61	72	48	55	56	57	55	66	51	57,5
DenStream	73	78	89	77	87	86	84	80	86	65	80,5
StreamKM++	95	99	109	102	110	111	114	101	103	104	104,8

Yapay veriler ile yapılan çalışmada, tez çalışmasında geliştirilen CluStream-Kmeans++ algoritmasının ortalama 24,2 saniye ile en kısa sürede çalıştığı gözlemlenmiştir. Gerçek verilere kıyasla yapay veri ile yapılan çalışmada tüm algoritmalar daha başarılı sonuçlar üretmiştir. Yapay verinin, gerçek veriye göre daha stabil olması bu farkın en önemli sebebidir.

Yukarıda anlatılan DenStream ve StreamKM++ algoritmalarının çevrimdışı kümeleme yapması ve ClusTree algoritmasının ağaç yapısıyla birlikte hiyerarşik olarak çalışması çalışma sürelerine etki etmiştir. Yapay veriler ile yapılan çalışmanın çalışma süresi karşılaştırması Şekil 4.32' de verilmiştir.



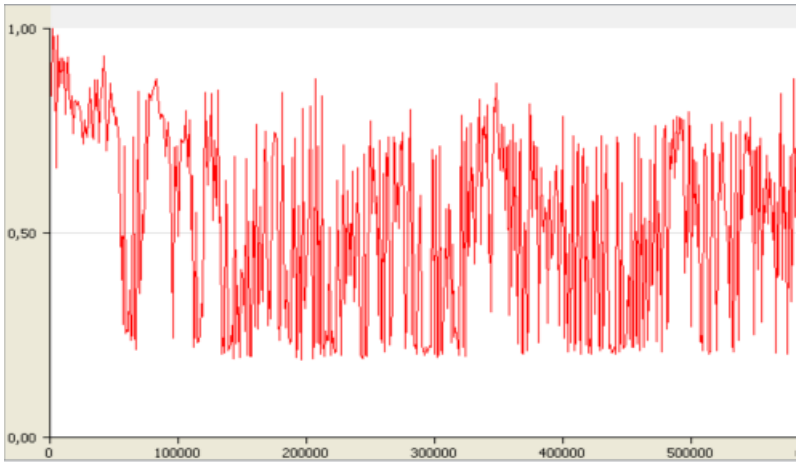
Şekil 4.32: Yapay veri ile çalışma zamanı grafiği.

4.2.3. Sistem Yükünün İncelenmesi

Bu bölümde, tez çalışmamız sırasında gerçek veri ve yapay verilerle yapılan test çalışmalarında kullanılan sistem kaynaklarının incelemesi yapılmıştır. Grafiklerdeki yatay eksen, işlenen kayıt sayısını; dikey eksen, hesaplanan kullanım oranı değerlerini göstermektedir.

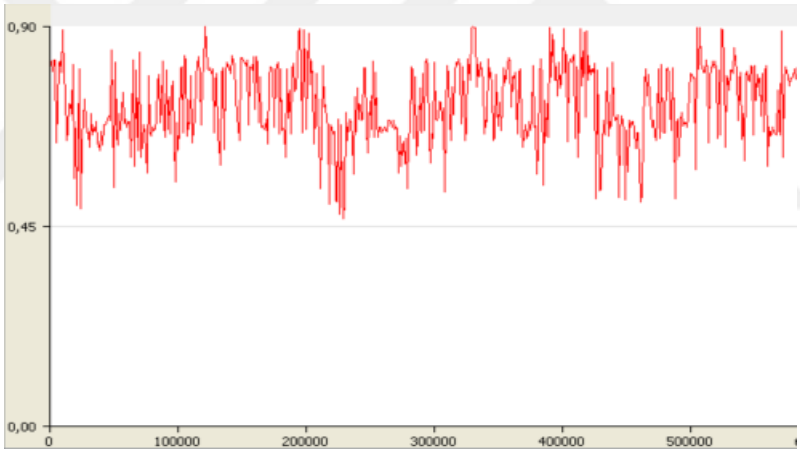


Şekil 4.33: Gerçek veri ile ölçekleme öncesi işlemci kullanım oranı.

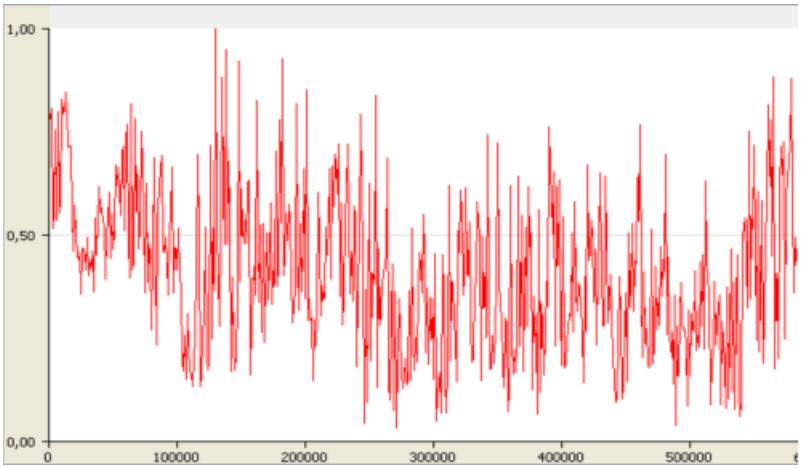


Şekil 4.34: Gerçek veri ile ölçekleme sonrası işlemci kullanım oranı.

Gerçek verilerle yapılan çalışmada, işlemci kullanım oranının ölçekleme öncesi ve sonrası değerleri Şekil 4.33 ve Şekil 4.34' te gösterilmiştir. Yapılan çalışmanın işlemci kullanım oranını azaltmada başarılı olduğu görülmüştür.

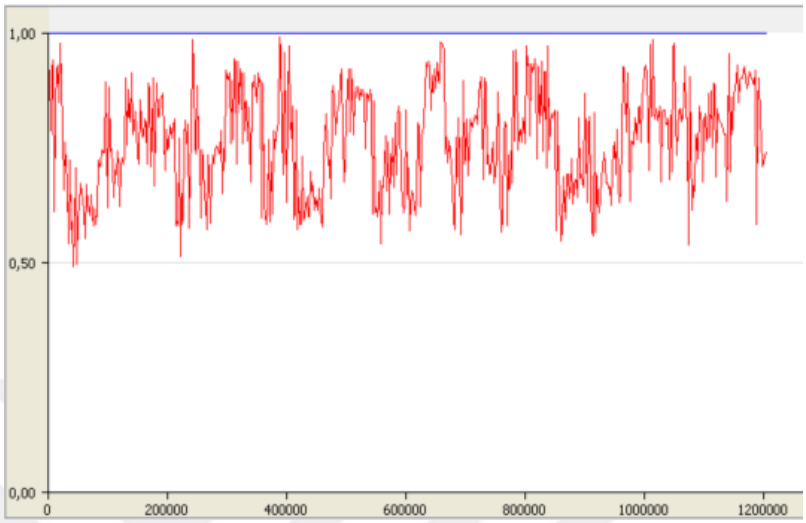


Şekil 4.35: Gerçek veri ile ölçekleme öncesi bellek kullanım oranı.

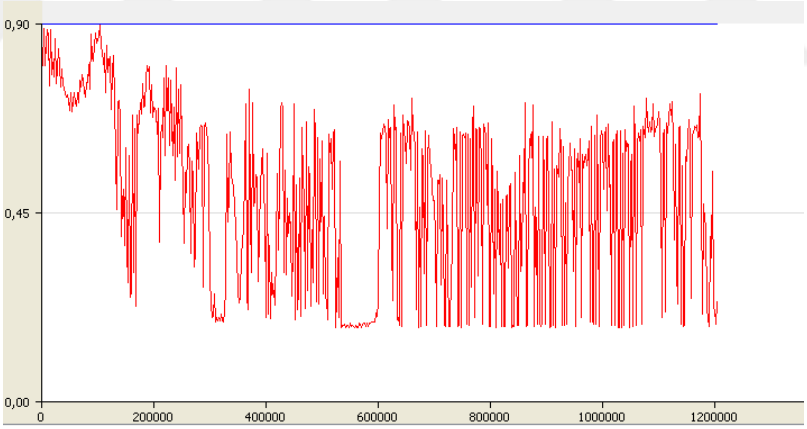


Şekil 4.36: Gerçek veri ile ölçekleme sonrası bellek kullanım oranı.

Gerçek verilerle yapılan çalışmada, bellek kullanım oranının ölçekleme öncesi ve sonrası değerleri Şekil 4.35 ve Şekil 4.36' da gösterilmiştir. Yapılan çalışmanın bellek kullanım oranını azaltmada başarılı olduğu görülmüştür.



Şekil 4.37: Yapay veri ile ölçekleme öncesi işlemci kullanım oranı.

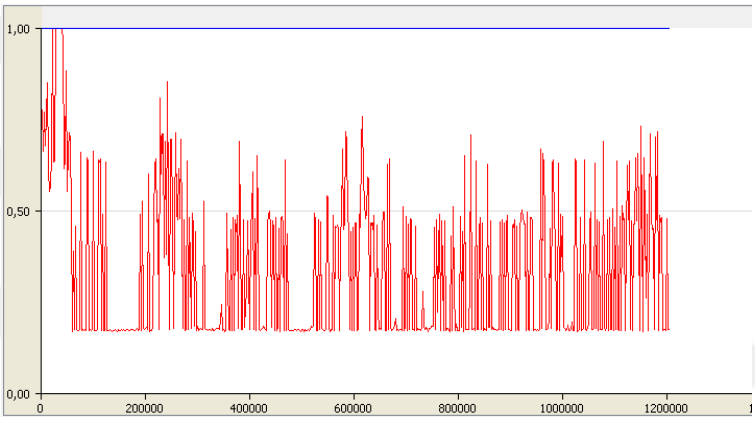


Şekil 4.38: Yapay veri ile ölçekleme sonrası işlemci kullanım oranı.

Yapay verilerle yapılan çalışmada, işlemci kullanım oranının ölçekleme öncesi ve sonrası değerleri Şekil 4.37 ve Şekil 4.38' de gösterilmiştir. Yapılan çalışmanın işlemci kullanım oranını azaltmada başarılı olduğu görülmüştür.



Şekil 4.39: Yapay veri ile ölçekleme öncesi bellek kullanım oranı.



Şekil 4.40: Yapay veri ile ölçekleme sonrası bellek kullanım oranı.

Yapay verilerle yapılan çalışmada, bellek kullanım oranının ölçekleme öncesi ve sonrası değerleri Şekil 4.39 ve Şekil 4.40' da gösterilmiştir. Yapılan çalışmanın bellek kullanım oranını azaltmada başarılı olduğu görülmüştür.

5. TARTIŞMA VE SONUÇ

Teknolojinin gün geçtikçe gelişmesiyle birlikte hayatımızdaki yeri ve önemi artmaktadır. Gelişen teknoloji, birçok cihazın birbirleriyle ve insanlarla olan etkileşimini arttırmıştır. Bu etkileşimin sonucunda ortaya büyük miktarda veri çıkmaktadır. Gerçek zamanlı üretilen bu veriler, üretildiği anda değerlidir. Özellikleri gereği sıralı, değişik boyutlarda ve düzensiz periyotlarda elde edilen bu veriler, akan veri olarak tanımlanmıştır. Akan veriler, hemen işlenmezse değerini kaybedebilir veya sonsuza dek kaybolabilir. Bu nedenle, yapılandırılmamış verileri sürekli olarak alıp analiz edebilen ölçeklenebilir sistemlerin geliştirilmesi önemlidir.

Tez kapsamında akan verinin temel özellikleri ve kaynakları incelenerek detaylandırılmıştır. Sonrasında makine öğrenmesi gelişimi ve süreçleri hakkında detaylı bilgi verilerek literatürde kullanılan algoritmalar ve yapılan çalışmalar analiz edilmiştir. Tez çalışmamızda kullanılan makine öğrenmesi algoritmalarının özellikleri hakkında detaylı bilgi verilmiştir. Tez çalışmasında geliştirilen CluStream-Kmeans++ algoritması detaylandırılıp uygulama entegrasyonu hakkında teknik bilgiler paylaşılmıştır. Test aşamasında kullanılan literatürdeki diğer algoritmalar CluStream, ClusTree, DenStream ve StreamKM++ ‘ın çalışma prensibi ve özellikleri anlatılmıştır.

Tez çalışmasında geliştirilen CluStream-Kmeans++ algoritması, MOA uygulamasıyla entegre çalışmaktadır. MOA uygulamasının kurulum ve kullanım detayları anlatılarak uygulama entegrasyonunda dikkat edilmesi gereken önemli noktalar detaylandırılmıştır. Geliştirilen uygulamada anlık sistem durumunun nasıl izlendiği hakkında detaylı bilgi verilmiştir.

Uygulamanın değerlendirilme adımında kullanılan akan veri değerlendirme metrikleri anlatılarak, kullanılan gerçek veri ve yapay veri setleri hakkında bilgilendirme yapılmıştır. Forest Cover Type veri seti ile yapılan test çalışmasında algoritmaların yapay verilere kıyasla daha çok zorlandığı görülmüştür. Bu duruma gerçek verilerdeki tutarsızlık, gürültü gibi durumların etki ettiği görülmüştür. CluStream-Kmeans++ algoritması, çevrimdışı kümeleme adımında kullandığı k-means++ yöntemiyle CluStream, DenStream, ClusTree ve StreamKM++ algoritmalarına karşı daha iyi sonuçlar ürettiği gözlemlenmiştir.

MOA uygulamasındaki akan veri oluşturma özelliği ile oluşturulan 1.200.000 yapay veri ile yapılan test çalışmasında algoritmaların 10 farklı değişken parametre içeren durum karşısında ürettiği sonuçlar gözlemlenmiştir. Algoritmaların her bir durum sonucunda değerlendirme metrikleri ile ürettikleri sonuçlar değerlendirilmiş ve detaylı olarak paylaşılmıştır. Oluşturulan durumların özelliklerine göre farklı algoritmalar, durum bazında daha iyi sonuçlar üretse de ortalama sonuçlarda CluStream-Kmeans++ algoritmasının daha performanslı sonuçlar ürettiği görülmüştür.

Bu tez çalışması kapsamında akan verinin geliştirilen CluStream-Kmeans++ algoritması kullanılarak makine öğrenme algoritmalarıyla analiz edilmesi ve akış sırasında sistemin anlık izlenmesi, gerekli durumlarda ölçeklenebilirlik analizinin yapılması noktalarına yoğunlaşılmıştır. Gerçek ve yapay veriler ile yapılan testlerde bu kapsamda başarılı sonuçlar üretildiği görülmüştür. Bu çalışmanın daha da ileriye taşınabilmesi noktasında daha güçlü kaynaklara sahip özelleştirilmiş bilgisayarlar ile dağıtık sistemler üzerinde, daha fazla istasyona sahip sistemler kurulabilir. Bu kapsamda Apache Spark, Apache Hadoop gibi büyük veri işleme sistemlerinden destek alınarak yapılabileceği düşünülmektedir.

KAYNAKLAR

[1]. Dhobi, J. S. and Ankleshwaria, T. B., 2014, Mining Data Streams: A Survey, *International Journal of Advance Research in Computer Science and Management Studies*, 2(2), 379-386.

[2]. Mitchel, T. M., 1997, *Machine Learning*, McGraw-Hill, Amerika Birleşik Devletleri, ISBN : 0071154671, 9780071154673

[3] Patel, N. and Singh D., 2015, An Algorithm to Construct Decision Tree for Machine Learning based on Similarity Factor , *International Journal of Computer Applications*, 111(10), 22-26.

[4]. Zhang, Z., 2016, A Gentle Introduction to Artificial Neural Networks, *Annals of Translational Medicine*, 4(19):370-370.

[5]. Bhasin, H. and Bhatia, S., 2011, Application of Genetic Algorithms in Machine Learning, *International Journal of Computer Science and Information Technologies*, 2(5), 2412-2415.

[6]. Tantalaki, N., Souravlas S. and Roumeliotis M., 2019, A Review on Big Data Real-Time Stream Processing and Its Scheduling Techniques, *International Journal of Parallel Emergent and Distributed Systems*, 35(5), 571-601.

[7]. Jinlong W., Congfu X., Weidong C. and Yunhe P., 2004, Survey of The Study on Frequent Pattern Mining in Data Streams, *IEEE International Conference* on *Systems, Man and Cybernetics*, 10-13 Oct. 2004, Netherlands, IEEE , ISBN**:**0-7803-8566-7, Volume 6.

[8]. Krishnaswamy, S., Gaber, M. M. and Zaslavsky, A., 2005, Mining Dat Streams: A Review, *ACM Sigmod Record*, 34(2), 18-26.

[9]. Gama, J. and Rodrigues R.P., 2007, Data Stream Processing, *Learning from Data Streams*, Chapter 3, Springer, Germany, 25-39.

[10]. Kolajo, T., Daramola O. and Adebiyi A., 2019, Big Data Stream Analysis: A Systematic Literature Review, *Journal of Big Data*, 6(1), 47.

[11]. Nittel, S., 2015, Real-time Sensor Data Streams, *Sigspatial Special*, 7(2), 22-28.

[12]. Nair, L. R., Shetty S.D. and Shetty S.D., 2017, Streaming Big Data Analysis for Real-Time Sentiment based Targeted Advertising, *International Journal of Electrical and Computer Engineering*, Vol. 7, 402-407.

[13]. Bartolini, I. and Patella M., 2019, Real-Time Stream Processing in Social Networks with RAM3S, *Future Internet*, 11(12), 249.

[14]. Couceiro, M., Fuentes D.S., Manzano D. and Lafuente L., 2011, Data Stream Processing on Real-Time Mobile Advertisement: Ericsson Research Approach, *12th IEEE International Conference on Mobile Data Management*, 6-9 June 2011, Sweden, Volume 1.

[15]. Bramer, M., 2007, *Principles of Data Mining*, Springer, London, ISBN: 978-1-84628-765-7.

[16]. Bifet, A., Holmes, G., Kirkby, R. and Pfahringer, B., 2011, *Data Stream Mining a Practical Approach*, https://moa.cms.waikato.ac.nz/downloads/, [Ziyaret tarihi: 15 Kasım 2020].

[17]. Binici, K., 2019, Makine Öğrenmesi Yaklaşımıyla e-Belgelere Standart Dosya Plan Numaralarının Otomatik Olarak Atanması Üzerine Bir Çalışma, *Bilgi Yönetimi Dergisi*, 2(2), 116-126.

[18]. Dar, U. , 2013, *Doğrusal (Lineer) Regresyon*, https://bookdown.org/ugurdar/dogrusalregresyon/, [Ziyaret tarihi: 14 Şubat 2021].

[19]. Küçüksille, E. U., ve Ateş, N., 2016, Destek Vektör Makineleri ile Yaramaz Elektronik Postaların Filtrelenmesi, *Türkiye Bilişim Vakfı Bilgisayar Bilimleri ve Mühendisliği Dergisi*, 6(1).

[20]. Kaviani, P. and Dhotre S., 2017, Short Survey on Naive Bayes Algorithm, *International Journal of Advance Engineering and Research Development*, 4(11), 607-611.

[21]. Kavzoglu, T. ve Colkesen I., 2017, Karar Ağaçları ile Uydu Görüntülerinin Sınıflandırılması: Kocaeli Örneği (Classification of Satellite Images Using Decision Trees: Kocaeli Case), *Harita Teknolojileri Elektronik Dergisi*, 2(1), 36-45.

[22]. Saini, I., Singh, D. and Khosla, A., 2013, QRS detection using K-Nearest Neighbor algorithm (KNN) and evaluation on standard ECG databases, *Journal of Advanced Research*, 4(4), 331-344.

[23]. Şeker, A., Diri, B. ve Balık, H.H., 2017, Derin Öğrenme Yöntemleri ve Uygulamaları Hakkında Bir İnceleme, *Gazi Mühendislik Bilimleri Dergisi*, 3(3), 47-64.

[24]. Takaoğlu, M. ve Takaoğlu, F., 2019, K-means ve Hiyerarşik Kümeleme Algoritmanın Weka ve Matlab Platformlarında Karşılaştırılması, *İstanbul Aydın Üniversitesi Dergisi*, 11(3), 303-317.

[25]. Sarıman, G., 2011, Veri Madenciliğinde Kümeleme Teknikleri Üzerine Bir Çalışma: K-Means ve K-Medoids Kümeleme Algoritmalarının Karşılaştırılması, *Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü Dergisi*, 15(3), 192-202.

[26]. Ram, A., Sunita, J., Jalal, A.S. and Manoj, K., 2010, A Density Based Algorithm for Discovering Density Varied Clusters in Large Spatial Databases, *International Journal of Computer Applications*, 3(6), 739-1038.

[27]. Kaya, A.G., 2018, *Yoğunluk tabanlı veri kümeleme algoritmaları üzerine*, Yüksek Lisans, Ege Üniversitesi Fen Bilimleri Enstitüsü.

[28]. Rani, P. and Suman, 2017, A Survey on STING and CLIQUE Grid Based Clustering Methods, *International Journal of Advanced Research in Computer Science*, 8(5), 1510-1512.

[29]. Sheikholeslami, G., Chatterjee, S. and Zhang A, 2000, WaveCluster: A Wavelet Based Clustering Approach for Spatial Data in Very Large Databases, *The VLDB Journal*, 8(3), 289-304.

[30]. Sayed, D., Rady, S. and Aref, M., 2020, Enhancing CluStream Algorithm for Clustering Big Data Streaming over Sliding Window, *12th International Conference on Electrical Engineering (ICEENG)*, Cairo, Egypt, 108-114.

[31]. Zhang, T., Ramakrishnan, R. and Livny, M., 1997, BIRCH: A New Data Clustering Algorithm and Its Applications, *Data Mining and Knowledge Discovery*, 1, 141–182.

[32].Özdemir, R., 2021, Makine Öğrenmesindeki Sınıflandırma Yöntemlerinin Karşılaştırılması ve E-Ticaret Üzerine Bir Uygulama, Yüksek Lisans Tezi, İstanbul Ticaret Üniversitesi Fen Bilimleri Enstitüsü.

[33]. Jing, G., Clay, W., Jiawei, K., Nikunj, C., Mohamad, M., Latifur, K. and Kevin W., 2009, Facing the Reality of Data Stream Classification: Coping with Scarcity of Labeled Data, Knowledge and Information Systems.

[34]. Lindburg, K., Stern, R., Buddhika, T., PallicaraS. and Ericson, K., 2017, Online Scheduling and Interface Alleviation for Low-Latency, High-Troughput Processing of Data Streams, IEEE Transactions on Parallel and Distributed Systems.

[35]. Meng, X., Wang, C., Guo, Q., Weng, Z. and Yang, C., 2017, Automating Characterization Deployment in Distributed Data Stream Management Systems, IEEE Transactions on Knowledge and Data Engineering, 29(12), 2669 - 2681.

[36]. Liu, X. and Buyya, R., 2019, Performance-oriented deployment of streaming applications on cloud, IEEE Tr. on Big Data, 5(1), 46-59.

[37]. Zhang, J., Zhu, L., Li, C., and Liu Y., 2016, The Real-time Scheduling Strategy Based on Traffic and Load Balancing in Storm, *IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems*.

[38]. Hulten, G. and Domingos, P., 2000, Mining high-speed data streams, *In : Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, 71-80.

[39]. Li, T., Xu, J. and Tang, J., 2015, A Predictive Scheduling Framework for Fast and Distributed Stream Data Processing, *IEEE International Conference on Big Data*, 333-338.

[40]. Candan, F., Doğan, M., Emir, S. and Kumbasar, T., 2018, Takviyeli Q-Öğrenme Yöntemiyle Labirent Problemi Çözümü, *Theory of Kowledge Conference.*

[41]. Machine Learning for Data Streams, https://moa.cms.waikato.ac.nz/ , [Ziyaret Tarihi: 15.09.2021].

[42]. IntelliJ Idea, https://www.jetbrains.com/idea/, [16.09.2021].

[43]. Singh, A., 2017, An Efficient Hybrid- Clustream Algorithm for Stream Mining, *13th International Conference on Signal-Image Technology and Internet-Based Systems*, 430-436.

[44]. Machine Learning Repository, https://archive.ics.uci.edu/ml/datasets/covertype, [16.09.2021].

[45]. Assent, I., Kranen, P., Baldauf, C. and Seidl, T., 2011, The ClusTree: Indexing micro-clusters for anytime stream mining, Knowledge and Information Systems, 29(2), 249-272.

[46]. Ester, M., Cao, F., Qian, W. and Zhou A., 2006, Density-Based Clustering over an Evolving Data Stream with Noise, *in Proceedings of the 2006 SIAM International Conference on Data Mining*, 328-329.

[47]. Ackermann, M., Lammersen, C., Martens, M. and Raupach, C., 2010, StreamKM++: A clustering, Journal of Experimental Algorithmics, 17(1), 173-187.

[48]. Vassilvitskii, S. and Arthur, D., 2007, k-means++: The advantages of careful seeding. In Proceedings of the eighteenth ACMSIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics, 1027-1035.

[49]. Apache Flink, https://flink.apache.org/, [18.09.2021].