T.C.
EGE ÜNİVERSİTESİ
Fen Bilimleri Enstitüsü

BÜYÜK VERİ KAVRAMI, K-MEANS, OPTICS VE CURE ALGORİTMALARININ UYGULANMASI VE PERFORMANS ÖLÇÜMÜ

Emin İSMAYILOV

Danışman: Dr. Ögr. Üyesi Arif GÜRSOY

Matematik Anabilim Dalı
Bilgisayar Bilimleri Yüksek Lisans Programı

İzmir
2019

Emin İSMAYILOV tarafından **Yüksek Lisans Tezi** olarak sunulan **"Büyük veri kavramı, K-means, Optics ve Cure algoritmalarının uygulanması ve performans ölçümü"** başlıklı bu çalışma E.Ü. Lisansüstü Eğitim ve Öğretim Yönetmeliği ile E.Ü. Fen Bilimleri Enstitüsü Eğitim ve Öğretim Yönergesi'nin ilgili hükümleri uyarınca tarafımızdan değerlendirilerek savunmaya değer bulunmuş ve 27/08/2018 tarihinde yapılan tez savunma sınavında aday oybirliği/oyçokluğu ile başarılı bulunmuştur.

Jüri Üyeleri:		**İmza**
Jüri Başkanı	: Dr. Öğr. Üyesi Arif GÜRSOY	.....
Raportör Üye	: Doç. Dr. Burak ORDİN	.....
Üye	: Dr. Öğr. Üyesi Refet POLAT	.....

EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

ETİK KURALLARA UYGUNLUK BEYANI

E.Ü. Lisansüstü Eğitim ve Öğretim Yönetmeliğinin ilgili hükümleri uyarınca Yüksek Lisans Tezi olarak sunduğum "**Büyük Veri Kavramı, K-means, Optics ve Cure algoritmalarının uygulanması ve performans ölçümü**" başlıklı bu tezin kendi çalışmam olduğunu, sunduğum tüm sonuç, doküman, bilgi ve belgeleri bizzat ve bu tez çalışması kapsamında elde ettiğimi, bu tez çalışmasıyla elde edilmeyen bütün bilgi ve yorumlara atıf yaptığımı ve bunları kaynaklar listesinde usulüne uygun olarak verdiğimi, tez çalışması ve yazımı sırasında patent ve telif haklarını ihlal edici bir davranışımın olmadığını, bu tezin herhangi bir bölümünü bu üniversite veya diğer bir üniversitede başka bir tez çalışması içinde sunmadığımı, bu tezin planlanmasından yazımına kadar bütün safhalarda bilimsel etik kurallarına uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul edeceğimi beyan ederim.

27 / 08 / 2019

İmzası

Emin İSMAYILOV

ÖNSÖZ

Bu tezde veri madenciliği ve büyük veri konuları ele alınacaktır. "Büyük Veri" terimi çok uzun zaman önce ortaya çıkmadı, ilk olarak 2008'de Nature dergisinde kullanıldı. Bu konuda, okuyuculardan büyük miktarda bilgiyi işlemek ve kullanıcıya anlaşılır bir şekilde sunmak için büyük verileri bir dizi özel yöntem ve araç olarak adlandırmaları istendi. Son yıllarda, ortaya çıkan alandaki araştırmacılar, büyük verilerin yalnızca analiz için uygun olmadığı, ancak birçok alanda yararlı olabileceği sonucuna varılmıştır. Örneğin google'da bir sorgu analizinin sonuçlarından grip salgınlarını tahmin etmekten, çok büyük bir havacılık verisine dayanan uçak biletlerinin karlı maliyetini belirlemeye kadar. Bu tezde büyük veriyi ele almamın nedeni yukarıda belirtildiği gibi yaşam, insanlık için bu kadar önemli bir konuya olan ilgim ve bundan sonra araştıracaklara katkıda bulunma isteğimdir.

Veri madenciliğinin her bir tekniği için çok sayıda algoritma bulunmaktadır. Algoritma sayısının çok fazla olması hangi durumda hangi algoritmanın daha başarılı olduğunun her zaman kestirilememesi veri madenciliğinin zorluklarından biridir. Bu nedenle algoritmalardan bazıları ele alınacak ve hangi durumda hangi algoritmayı kullanmalıyım sorusuna cevap bulmaya çalışılacaktır.

Tez çalışmamda planlanmasında, araştırılmasında, yürütülmesinde ve oluşumunda ilgi ve desteğini esirgemeyen, engin bilgi ve tecrübelerinden yararlandığım, yönlendirme ve bilgilendirmeleriyle çalışmamı bilimsel temeller ışığında şekillendiren sayın hocam Dr. Öğr. Üyesi Arif GÜRSOY'a sonsuz teşekkürlerimi sunarım.

İZMİR

27/08/2019

Emin İSMAYILOV

ÖZET

BÜYÜK VERİ KAVRAMI, K-MEANS, OPTİCS VE CURE ALGORİTMALARININ UYGULANMASI VE PERFORMANS ÖLÇÜMÜ

Emin İSMAYILOV

Yüksek Lisans Tezi, Matematik Anabilim Dalı

Tez Danışman: Dr. Öğr. Üyesi Arif GÜRSOY

Ağustos 2019, 63 Sayfa

Bu tezde veri madenciliği ve büyük veri konuları ele alınacaktır. Büyük veriyi ele almamın nedeni, günümüzde önemli bir kavrama dönüşmesi ve zamanla neredeyse tüm sektörlerin karar aşamalarında büyük rol oynamasıdır. Birçok insan büyük verilerden bahseder, ama ne yazık ki, sadece birkaçı ne olduğunu bilir. Bu tezde, büyük verinin ne anlama geldiğini, modern dünyaya nasıl uygulanacağını ve ne tür özellikleri olduğunu öğreneceğiz.

Veri madenciliği, çok miktarda bilginin analizine dayanan otomatik bir veri aramasıdır. Amaç, genel analizde imkansız olan trend ve kalıpların belirlenmesidir. Verileri bölümlere ayırmak ve sonraki olayların olasılığını değerlendirmek için karmaşık matematiksel algoritmalar kullanılır. Bu da, kurumların performansını daha doğru bir şekilde değerlendirmesine yardımcı olabilir. Dolayısıyla Veri Madenciliği yaşam standartlarımızın artması, hayatımızın kolaylaşması yolunda önemli bir kavramdır.

Veri madenciliğinin her bir tekniği için çok sayıda algoritma bulunmaktadır. Algoritma sayısının çok fazla olması hangi durumda hangi algoritmanın daha başarılı olduğu her zaman kestirilememesi veri madenciliğinin zorluklarından biridir. Bu nedenle algoritmalardan bazıları ele alınacak ve hangi durumda hangi algoritmayı kullanmalıyım sorusuna cevap bulmaya çalışılacaktır.

Anahtar sözcükler: Büyük Veri, Veri Madenciliği, K-means, Optics, Cure

ABSTRACT

BIG DATA CONCEPT, K-MEANS, OPTICS AND CURE ALGORITMS APPLYING AND PERFORMANCE MEASUREMENT

Emin İSMAYILOV

Masters Degree in Mathematics

Supervisor: Asst. Prof. Dr. Arif GÜRSOY

Ağustos 2019, 63 Pages

In this thesis, data mining and big data topics will be discussed. The reason why I deal with big data is that it has become an important concept and plays a major role in the decision-making process of almost all sectors. A lot of people talk about big data, but unfortunately, only a few know what it is. In this thesis, we will learn what big data means, how to apply it to the modern world and what kind of features it has.

Data mining is an automated data search based on the analysis of large amounts of information. The aim is to identify trends and patterns that are impossible in the overall analysis. Complex mathematical algorithms are used to segment the data and evaluate the likelihood of subsequent events. This can help organizations evaluate their performance more accurately. Therefore, Data Mining is an important concept for increasing our living standards and making our lives easier.

There are many algorithms for each technique of data mining. One of the difficulties of data mining is the fact that the number of algorithms is too high and which algorithm is not always more successful. Therefore, some of the algorithms will be discussed and an attempt will be made to find out which algorithm should be used in which case.

Keywords: Big Data, Data mining, K-means, Optics, Cure

İÇİNDEKİLER

Sayfa

İÇİNDEKİLER(DEVAMI)

İÇİNDEKİLER(DEVAMI)

1. GİRİŞ

Bu tezde veri madenciliği ve büyük veri konuları ele alınacaktır. Büyük veriyi ele almamın nedeni büyük veri ne kadar bilgi sahibi olunduğundan ziyade bu bilgiyle ne yapılabileceği üzerine yoğunlaşıyor. Herhangi bir kaynaktan veriyi alıp onları para ve zaman tasarrufu (maliyet azaltma), yeni proje, ürün, hizmet gelişimi ve optimize edilmiş önerileri aynı zamanda da akıllı karar verilmesini sağlayan cevaplar bulmak için analiz edebilirsiniz.

Veri madenciliğinin her bir tekniği için çok sayıda algoritma bulunmaktadır. Algoritma sayısının çok fazla olması hangi durumda hangi algoritmanın daha başarılı olduğunun her zaman kestirilememesi veri madenciliğinin zorluklarından biridir. Bu nedenle algoritmalardan bazıları ele alınacak ve hangi durumda hangi algoritmayı kullanmalıyım sorusuna cevap bulmaya çalışılacaktır.

Bu tez 5 bölümden oluşuyor. İkinci bölümde ilk başta veri evrimi nedir sorusu cevaplanmıştır. Büyük veri hakkında genel bilgi verilecektir. Büyük Verinin hayatımızda rolü ve önemi anlatılmıştır. Büyük verinin bileşenleri ve özellikleri anlatılmıştır. Büyük veri bilimi ile alakalı bilgi verilmiştir. Büyük Veri Teknik ve Teknolojileri anlatılmıştır.

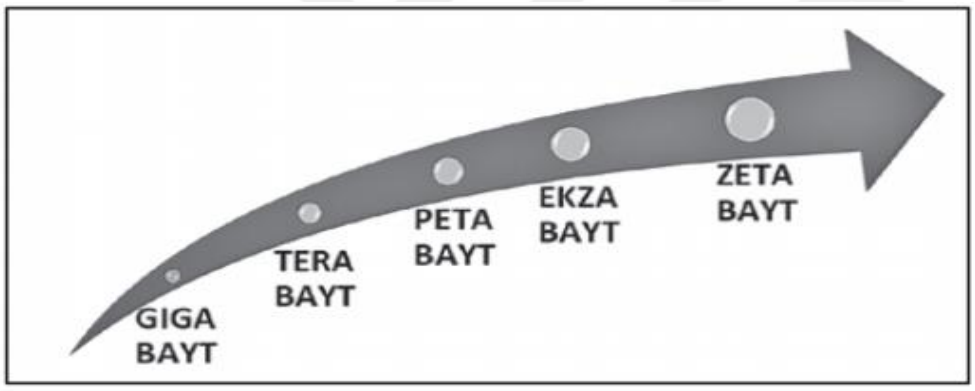
Üçüncü bölümde veri madenciliği hakkında genel bilgi verilmiştir. Veri madenciliği yöntemi sınıflandırma ve kümeleme anlatılmıştır. Kümeleme algoritmalarından birkaçı ele alınmıştır. Genel bilgi verilip, avantaj ve dezavantajları ele incelenmiştir.

Dördüncü bölümde veri madenciliği algoritmalarından K-means, Optics ve Cure algoritmaları ilişkisel hale getirilmiş verilere uygulmış ve performans ölçümü yapılmıştır.

Beşinci bölünde varılan sonuçlar özetlenmiştir.

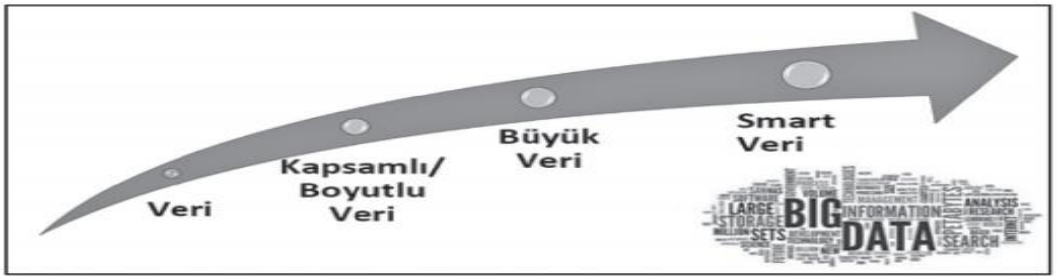
2. VERİNİN EVRİMİ

Elektronik ortamlar gelişip yaygınlaştıkça, verilerin boyutu artmakta (Şekil 2.1.), farklı veri tiplerinin beraber işlenmesi kolaylaşmakta, verilerin değerlendirilmesine bakış açılarımız değişmekte ve gelişmektedir. Verilerin isimlendirilmesinde gelinen en son nokta ise Şekil 2.2.'de verilmiştir. Verilerin isimlendirilmesi değiştikçe ve geliştikçe, kullanılan teknik ve teknolojiler ile yöntem ve çözümlerde de değişiklikler vardır. Hatta bu alan, "veri bilimi" olarak isimlendirilen yeni bir bilim dalı olarak karşımıza çıkmaktadır. Verilerin yönetimi, saklanması, işlenmesi veya aktarılması kadar, verilerden değer elde etmek, verileri bilgiye ve değere dönüştürmek, bunun ekonomisini oluşturmak, üzerinde daha fazla odaklanılmış konulardır. Bu aşamalar aslında verinin evrimleşmesidir. Verilerin bilgiye, bilginin bilgeliğe (hikmete), bilgeliğin kararlara dönüşümü ise bu evrimin en önemli adımlarıdır. Her adım, farklı aşamaları, zorlukları ve kazanımları kapsamaktadır.



Şekil 2.1. Veri miktarı değişimi

Verilerin miktarı, büyüklüğü veya çeşitliliği artarken, yeni kavramlar ortaya çıkarmakta veya kavramlar zamanla değişmektedir. Veriler artık basit bir şekilde veri olmaktan çıkıp belli bir yöntem ve daha fazla katma değer üretecek şekle dönüşmektedir.



Şekil 2.2. Verilerin evrimleşmesi

Günümüzde verilerin en gelişmişi, değerlisi veya faydalısı, belki de zeki unsurlar içeren en günceli ise semantik yapılarıda içerisinde barındıran “smart” verilerdir (Şekil 2.2.). Çağımızın hammaddesi olan veri, bilgi veya özbilgiyi tanımlamak, kavramak veya bunları anlamak ve bunların günümüzün en önemli varlık- ları olduğunu bilmek önemlidir. Veri işlenme sürecinde anlam kazanarak bilgi, özbilgi ve bilgeliğe dönüşmektedir [1]. Veri, bilişim teknolojisi açısından “sayısal ortamlarda bulunan, işlenen veya taşınan sinyaller”, “anlamlı hale dönüştürülmemiş bitler” veya “birbiriyle bağlantısı henüz kurulmamış bilinenler” olarak tanımlanabilir. Verinin “belli bir anlam ifade edecek şekilde işlenip, belirsizliğinin azaltılmasıyla” bilgi elde edilir. Belirli bir konuda elde edilen bilginin, tecrübe veya öğrenme ile farkında olunması ve anlaşılması özbilgi olarak adlandırılır. Bilgelik ise, güvenilir karar vermek için özbilginin nasıl kullanılacağını kavramaktır. Şekil 2.3.’de verinin günlük hayatımızda az veya çok kullandığımız adımları, bunların ilişkileri ve adlandırılması gösterilmiştir. Veri içeriğindeki bilinmezliklerin giderilmesi ile bilgiye, bilginin anlam ve içeriğinin anlaşılması ile özbilgiye, oluşan birikimin içselleştirilmesi ile bilgeliğe ve bu adımda oluşan değerin ise belirlenen amaca dönüştürülmesine veya dönüştürülmüş haline de karar veya hedef denilmektedir.

Şekil 2.3. Verinin evrimleşme süreci adımları ve dönüşümleri

Verileri anlamak, değerlendirebilmek ve farklı bakış açıları ile analiz edebilmek ve sonuçta beklenen hedefe veya istenilen kararlara erişilebilmesi için, bu veriler dört grupta sınıflandırılmış olup aşağıda kısaca açıklanmıştır.

• Veri parçası (data spot), analizlerde dikkate alınan erişilebilir verinin bir alt kümesi olarak gruplandırılır.

• Erişilebilir veri (light data), erişilebilir ve her an kullanıma hazır olan veri grubudur.

• Gri veri (gray data), erişemediğimiz ancak nitelikli varsayımlar yapabileceğimizi ve analiz ettiğimiz sistemin bir parçası olduğunu bildiğimiz veri gruplarıdır.

• Karanlık veri (dark data) ise, nitel veya nicel olup olmadığı anlaşılamayan, bilinmeyen veya gruplandırılamayan veri grubudur. Bu veriler kısaca bilmediğimizin veya bilemediğimizin farkında bile olamadığımız veri gruplarıdır.

Verilerin boyutu ve türü artarken, bunların işlenmesi, taşınması veya değerlendirilmesi de beraberinde ciddi sorunları getirmiş, doğal olarak, bu problemlerin çözülmesi için ise büyük ölçekli verilerin işlenmesi için bugüne kadar grid hesaplama, bulut hesaplama gibi birçok teknik ve teknolojiler de geliştirilmiştir. Grid hesaplama, verinin hacim problemine bir çözüm getirmek için geliştirilmiş iken, bulut hesaplama verinin hacmi, hız, maliyet, yönetilebilirlik gibi hususlar ile başa çıkmak için geliştirilmiştir. Açık kaynak teknolojileri ise maliyeti düşük analizler yapmak için önerilmiştir. Fakat bu teknolojilerin de çeşitli problemleri mevcuttur. Bu problemler; grid hesaplama için maliyetinin yüksek olması, bulut hesaplama için yavaş erişim, açık kaynak teknolojilerinde ise kararlılıktır. Günümüz verileri bu tez kapsamında tekrar değerlendirilmiş, kavramlar gözden geçirilmiş, daha iyi anlaşılması ve kavranması için bilinmesi gerekenler açıklanmıştır. Veri kümeleri, ambarlarında bulunan varlıkların etkin analizi, yeni çözümlerin veya çıktıların üretilebilmesi bilinmesi, yapılması ve sahip olunması gereken teknik ve teknolojilerin tanımları bu bölümde bilinmesi gereken diğer hususlar da farklı bölümlerde açıklanmıştır.

2.1. Büyük Veri

Büyük veriden tez çalışması boyunca BV olarak bahsedilecektir.

BV tanımları:

i. BV, yapılandırılmış ve yapılandırılmamış verilerin çok büyük olduğu ve geleneksel veritabanı ve yazılım teknikleri kullanılarak işlenmesi zor olan çok büyük bir hacim anlamına gelen bir ifadedir.

ii. Modern dünyada, BV, yeni teknolojik kapasitelerin büyük miktarda veriyi analiz ettiği ortaya çıkmasıyla ilgili olan sosyo-ekonomik bir olgudur.

Tanımlardan da görüldüğü gibi BV fiziksel olarak büyük yer kapsayan veriler değildir. Fiziksel olarak çok küçük yer kapsayan veriler de BV olarak adlandırılabilir veya tam tersi fiziksel olarak büyük yer kapsayan veri BV olmayabilir. Bir veri setine BV denebilmesi için bu setin ilişkisel veri tabanlarında tutulması mümkün olmamalıdır.

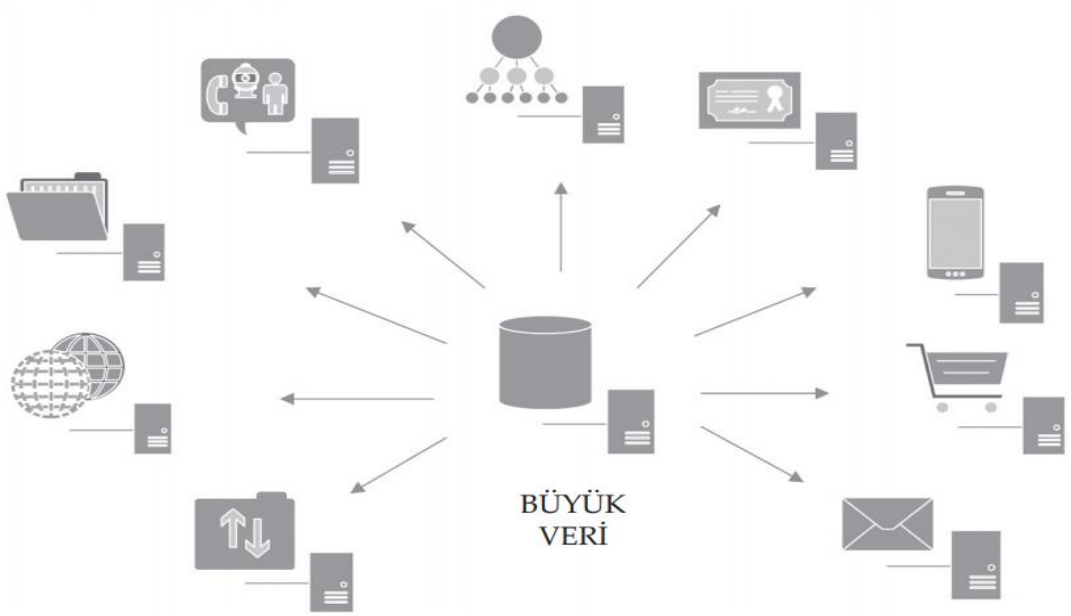
Günümüzde sıkça adını duymaya başladığımız BV, teknolojinin ilerlemesi ve kullanım alanlarının artması ile ortaya çıkmış bir kelime olarak görülse de, yıllardır içinde bulunduğumuz fakat gelişiminden çok haberdar olamadığımız bir olgudur. Bu olguya farkında olmadan sürekli oluşturduğumuz verilerle destek vermekte ve "BV" olarak bildiğimiz bu veri topluluğunana sürekli veri akışı sağlanmasında bizler de katkı sağlamaktayız.

Teknolojinin ve internetin gelişmesinin sonucu olarak günümüzde bilginin gücü de önem taşımaya ve bununla beraber internet dünyasındaki birçok olgu 'Bilgi Çöplüğü' olarak anılmaya başlamıştır. Bu çöplükten anlamlı verilerin de çıkabileceğini düşünen yazılım şirketleri, AR-GE çalışmalarını bu anlamda yürüterek BV olarak isimlendirdiğimiz olguyu ortaya çıkarttılar.

BV olarak bildiğimiz bu olgu, fiziksel olarak çok fazla yer tutan veri çağrışımı yapsa da aslında tam olarak böyle değil. BV, paylaşımlar, fotoğraf arşivleri, takip amaçlı kayıt tuttuğumuz 'log' dosyaları gibi farklı kaynaklardaki verilerin toplanarak anlamlı ve kullanılabilir bir biçime getirilmesidir.

BV kavramı; verilerin saklanmasında, analiz edilmesinde ve yönetilmesinde klasik veritabanı yönetim sistemlerin yetersiz kaldığı durumlarda karşımıza çıkan bir problem olarak tanımlanabilir. Bu kavram, organizasyonlara göre değişiklik gösterebilir. Örneğin; bir organizasyon için gigabaytlarca veriyi işlemek problem iken, bazı organizasyonlarda terabaytlarca veri analizi problem olabilmektedir. "BV farklı formatlarda, hızlı bir şekilde ve büyük hacimde üretilen veriler" olarak adlandırılabilir. Büyük veri tanımından anlaşılacağı üzere bizlere büyük fırsatlar sunarken, beraberinde yeni sorunları da getirmektedir.

Örneğin; büyük verinin saklanması ve bu veriden çeşitli katma değerler üretimi klasik yöntemlerle gerçekleştirilemeyeceği aşikârdır. Bundan dolayı genellikle veriler üzerinden çeşitli örneklemler alınarak çeşitli karar destek sistemlerinde kullanılmak üzere analizler yapılmaktadır. Fakat büyük resim tam anlamıyla görünemediğinden örneklemlerden üretilen değerler, gerçek değerlerin çok altında olabilmektedir. Hadoop gibi büyük veri teknolojileri ile bütün veri üzerinde çalışma imkanı sağlanabilmektedir ve dolayısıyla daha doğru, etkili ve kapsamlı sonuçlar veriden temin edilebilmektedir. Şekil 2.4.'de verildiği gibi, elektronik ortamlarda yapılan işlemler, veri trafiği, uygulamalar, e-postalar, metinler, belgeler, videolar, sesler, resimler, tıklama akışları, sistem günlükleri, arama sorguları, sosyal ağ etkileşimleri, sağlık kayıtları, bilimsel veriler, devlet ve özel sektöre ait kayıtlar, sensörler ve akıllı telefonlardan beslenen büyük veriyi her boyutuyla anlamak büyük önem arz etmektedir. Belirli karakteristik özellikler dâhilinde verinin sınıflandırılması, uygun büyük veri örüntüleriyle eşleşmesinde kolaylık sağlamaktadır. Tablo 2.1'de betimlenen anahtar kategorilerin kombinasyonu ile veriye erişimden tüketim sürecine kadar olan tüm aşamaları kapsamaktadır [4]. Tablo 2.1'de verilen açıklamalara bakıldığında büyük verinin boyutunu, amacını, kapsamını, işin ciddiyetini, yapılabilecekleri ve en önemlisi karşılaşılabilecek zorlukları göre biliriz.



Şekil 2.4. Büyük veriyi oluşturan alanlar

Günümüz sayısal dünyasında sürekli verilerin artışı Şekil 2.1.'de verildiği gibi zetabaytlar mertebesinde olması, mobil cep telefonu sayısının milyarlarca olması, her gün 1 milyarın üzerinde Google araması yapılması, Facebook kullanıcılarının oluşturduğu petabaytlarca verinin saklanıp analiz edilmesi, Akamai firmasının daha iyi hedefli reklamlar için günde 75 milyon olayı analiz etmeleri, Walmart'ın her saat 1 milyondan fazla müşteri verilerini analiz etmesi, Youtube kullanıcılarının dakikada 48 saatlik yeni bir video yüklemeleri, dünyada her yıl 1.8 zetabayt veri üretilmesi, internet üzerinden yapılan işlemlerin milyarların üzerinde olması gibi pek çok örnek, büyük verinin günümüz dünyasındaki yerini daha iyi kavramak için verilebilir.

Tablo 2.1. Büyük Verinin Sınıflandırılması

Veri Türü	Meta Veri, Ana Veri, Geçmiş Veri, İşlemsel Veri
Veri Biçimi	Yapısal Veri, Yarı-Yapısal Veri, Yapısal Olmayan Veri
Veri Kaynağı	Web ve Sosyal Medya, Makine Kaynaklı, Nesnelerin İnterneti (IoT), İnsan Kaynaklı, Dâhili Kaynaklar, İşlem Verisi, Veri Sağlayıcıları
Veri Frekansı	İsteğe Bağlı, Sürekli, Gerçek Zamanlı, Zaman Serileri
Veri Saklama	Sütun Tabanlı, Graf, Anahtar-Değer, Doküman Tabanlı
Veri Tüketicileri	İnsan, İş Süreci, Kurumsal Uygulamalar, Veri Ambarları
Veri Kullanımı	Endüstri, Akademi, Devlet, Araştırma Merkezleri
Analiz Türü	Toplu, Akan, İnteraktif
İşleme Amacı	Tahmine Dayalı, Analitik, Sorgu ve Raporlama, Modelleme
İşleme Metodu	Yüksek Performanslı Hesaplama, Dağıtık, Paralel, Kümeleme

Büyük veri uygulamaları birçok alanda özellikle endüstride kullanılmaktadır. Genel olarak kullanım alanlarına bakıldığında, sağlıktan pazarlamaya, üretimden tüketime, kamudan özel sektöre, bankacılıktan sigorta şirketlerine, telekomdan havacılık sektörüne kadar pek çok örnek verilebilir. Sağlıkta; sağlık alışkanlıklarıyla ilişkili olarak kalite eğilimleri, klinik karar destek sistemleri, uzaktan hasta takip etme, hasta profilleme, hastalık tahmini, vb. Pazarlamada; çapraz satış, konum tabanlı pazarlama, duygu analizi, eğilim analizi, davranış analizi, vb. Üretimde; sensör tabanlı operasyonlar, tedarik zinciri, envanter yönetimi, lojistik, vb. Kamuda; temel ihtiyaç tespiti, trafik sorunları çözümü, gürültü, hava ve su kirliliği önleme, istismar tespiti, vb.

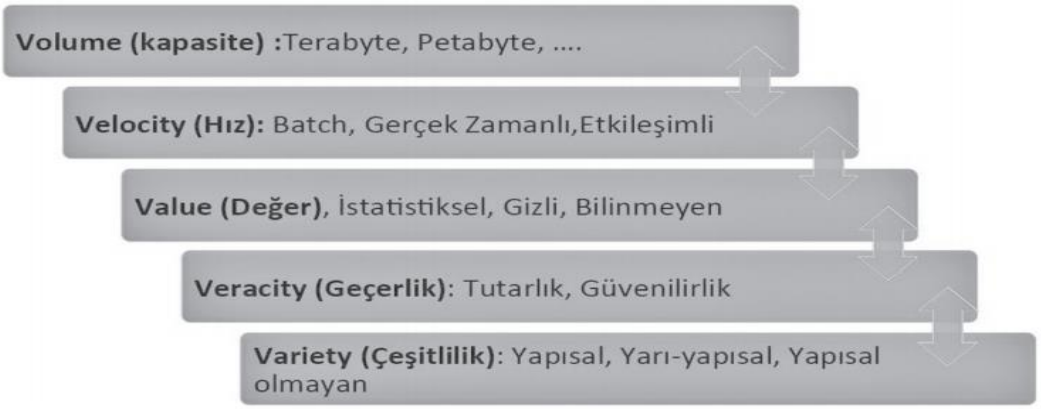
Bankacılık ve Sigortacılıkta; hilelerin, istismarın veya suistimallerin tespiti, müşteri tahminleme, risk analizleri, vb. Telekom sektöründe ise; coğrafi hedefle reklam, acil müdahale, kentsel planlama, ücretlendirme, suistimal önleme, saldırı tespiti, vb. konularda faydalanılmaktadır.

Büyük verinin farklı alanlarda kullanımının artması ve bu teknolojilerden faydalanılmasının önünün açılması için farklı çalışmalar ile bu alan desteklenmiştir. 2012 yılında ABD'de Obama yönetimi bilim, sağlık, enerji ve savunma gibi pek çok kuruma 200 milyon dolarlık bir yatırımın sağlanacağı "Big Data Research and Development Initiative" adlı girişimini duyurmuştur [7].

Avrupa Birliği Komisyonu verinin korunması için 2012'de, IoT'nin geliştirilmesi, makineler arası iletişim, internet güvenliğinin sağlanması gibi konuların önemini ve kamu verilerinin potansiyel ekonomik değerini vurgulayan "Digital Agenda for Europe and Challenges for 2012" adlı doküman yayımlanmıştır. Ülkemizde, kamu kurumlarında gerçekleştirilmiş büyük veri uygulamaları veya büyük veri çalışmalarına girdi sağlayabilecek projeler bulunmaktadır [7, 8]. T.C. Başbakanlık İdareyi Geliştirme Başkanlığı (KAYSİS), Enerji ve Tabii Kaynaklar Bakanlığı (Enerji Tahmin), Sosyal Güvenlik Kurumu (e-Bildirge Sistemi, MEDULA, Aylık Tahsis, ALO 170, Veri Ambarı), Milli Eğitim Bakanlığı (MEBBİS, e-Okul, FATİH, e-YAYGIN, ALO 147) ve Sağlık Bakanlığı (e-Nabız, Sağlık.NET, MHRS, Aşı Takip) gibi çalışmalar örnek olarak verilebilir.

2.2. Büyük Veri Bileşenleri

Büyük veri, kısaca 5V (volume, velocity, variety, veracity, value) olarak adlandırılan beş bileşen ile ifade edilmektedir [6] (Şekil 2.5.). Yapılan yeni çalışmalarda, bu "V" lerin yani bileşenleri sayısının arttığı bilinmekte ve görülmektedir.



Şekil 2.5. Büyük veri bileşenleri

i. Hacim (Volume)

BV olarak isimlendirdiğimiz olgu verilerimiz her geçen gün olağanüstü bir hızlı bir şekilde artıyor olabilir, haliyle gelecekteki durumlarımızı da düşünerek ileride bu veri setlerini nasıl yöneteceğimizi iyi düşünmemiz ve bu doğrultuda doğru planlar yapmamız gerekmektedir. Verinin hacmi verinin boyutu ile doğru orantılıdır ve genel olarak gigabayt, terabayt, petabayt gibi çeşitli veri ölçü birimleri ile ifade edilmektedir.

Büyük verinin bilinen en önemli özelliklerinden ve problemlerinden biri olarak karşımıza çıkmaktadır. Verinin hacminin problem olabilmesi gibi organizasyonun büyüklüğüne göre değişebilmektedir. Veri sadece petabaytlarca verinin olması durumunda büyük veri olarak karşımıza çıkamamaktadır. Genel bir ifade ile eldeki veri miktarının istenilen zamanda analizinin mümkün olmadığı durumlarda veri hacmi büyük veri problemi olmaktadır.

ii. Hız (Velocity)

BV üretimi her geçen gün hızına hız katmakta ve bu veriler saniyede inanılmaz boyutlara ulaşmaktadır. Hızlı büyüyen veri, o veriye muhtaç olan işlem sayısının ve çeşitliliğinin de aynı hızda artması sonucunu ortaya çıkartmaktadır ve hem yazılımsal hem de donanımsal olarak bu yoğunluğu kaldırabilmeliyiz. Veri değişken hızlarla üretilebilmektedir.

Büyük hacimli durağan veriler bir büyük veri problemi oluştururken, özellikle gündelik hayatta sıklıkla kullandığımız telefon veya IoT cihazları, çeşitli makineler tarafından üretilen sensör verileri ve bunlara benzeyen veri kaynakları çok hızlı bir şekilde veri üretmektedir. Bir akış içerisinde hızlı bir şekilde üretilen verinin gerçek zamanlı olarak analiz edilebilmesi ve yönetilebilmesi ise büyük verinin bir diğer problemi olarak karşımıza çıkmaktadır.

iii. Çeşitlilik (Variety)

Üretilen veriler genel olarak yapısal olmadığı ve birçok farklı ortamdan elde edilen veri formatlarından oluştukları için bütünleşik ve birbirlerine dönüştürülebiliyor olmaları gerekmektedir. Verinin çeşitliliği, veri kaynaklarının farklılığından kaynaklanmaktadır. Üretilen veriler yapısal, yarı-yapısal veya yapısal olmayan formatlarda karşımıza çıkabilmektedir. Yapısal verilere ilişkisel veri tabanlarındaki veriler, yarı-yapısal verilere XML, JSON formatındaki veriler yapısal olmayan verilere ise ses, video, metin dosyaları örnek olarak verilebilir. Bu farklı yapılarda verilerin bir arada kullanılıyor olması durumunda verideki bu çeşitlilik büyük bir problem olarak karşımıza çıkmaktadır.

Özellikle verilerin "Çıkar Dönüştür Yükle" (ETL) işlemlerinde bu verilere özgü birçok yeni büyük veri teknolojilerinin kullanılması bir zorunluluk haline gelmektedir.

iv. Geçerlik (Veracity)

Gerçek hayat problemlerinde kullanılan büyük veri içerisinde verinin doğruluğunu olumsuz olarak etkileyebilecek birçok faktör vardır. Bu faktörler genellikle karşımıza gürültü veya aykırılık olarak çıkmaktadır.

Doğruluğundan emin olunamayan veri üzerinde yapılacak analizler gerçek değerin ortaya konulmasını engellemektedir. Özellikle çeşitli sensörler aracılığı ile üretilen veriler sensör doğası gereği gürültüye çok meyillidir. Bundan dolayı bu ve benzeri durumların oluşması halinde büyük veriden büyük değer üretebilmek mümkün değildir. Büyük veriye özgü teknolojiler ile verinin doğruluğundan ve analize uygunluğundan emin olunmalıdır.

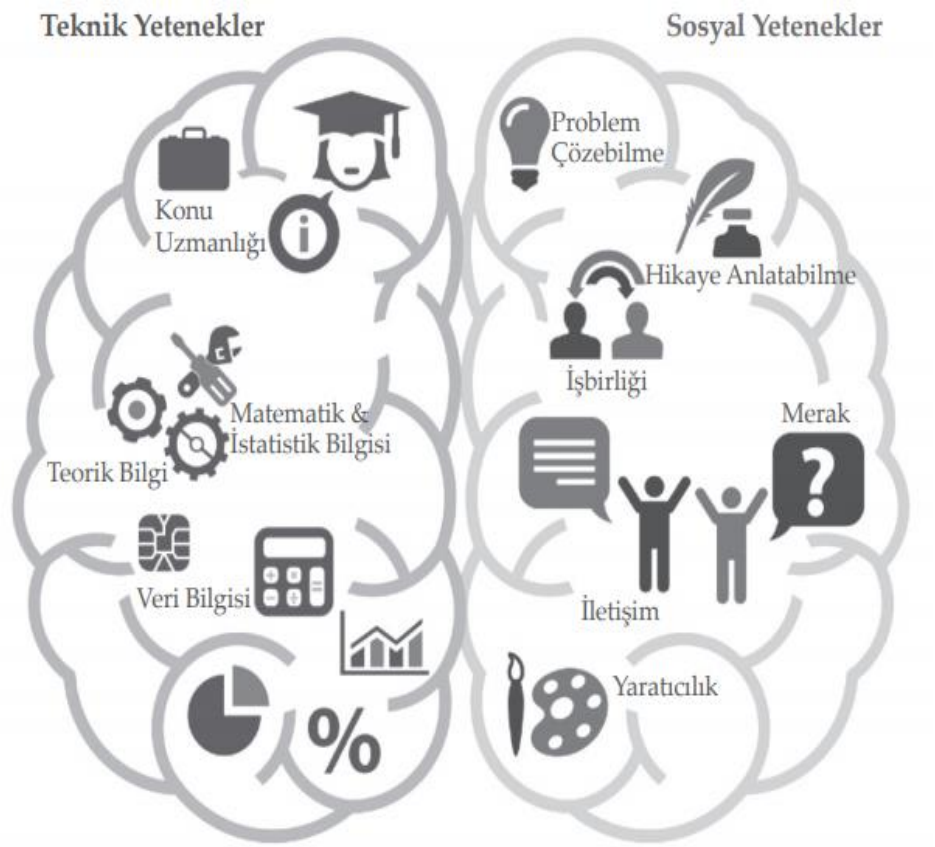
iv. Değer (Value)

Belki de en önemli katmanlardan bir tanesi “Değer” katmanıdır, verilerimiz yukarıdaki veri bileşenlerden filtrelendikten sonra büyük verinin üretimi ve işlenmesi katmanlarında elde edilen verilerin şirketler için artı değer sağlıyor olması gerekir. Veri analizini çeşitli organizasyonlarca önemli hale getiren veriden üretilen değerdir. Değer üretilemeyen herhangi bir veri anlamsızdır. Üretilen değer ise verinin içeriğine, üretilme amacına, uygulama alanına vb. faktörlere göre değişiklik göstermektedir. Var olan verilerin yukarıdaki özelliklere sahip olması durumunda bu veriden geleneksel yöntemler ile değer üretmek çok zor olmaktadır. Dolayısıyla bu verinin büyük veri bakışıyla ele alınıp büyük veri teknolojileri ile analiz edilme ihtiyacı doğmuştur.

2.3. Büyük Veri Bilimi

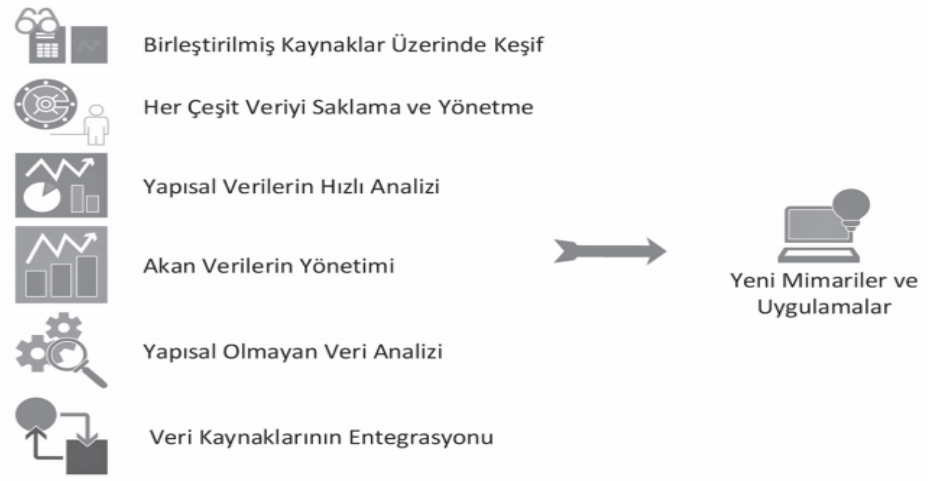
BV hızlı yükselişi, sürecinde gerekli analizlerin yapılması için doğru bakış açısına ve uzman işçi ihtiyacına, modern altyapılara, metotlara ve metodolojilere ihtiyaç vardır. Bu durum; büyük verilerin biçimlendirilmesini ve modellenmesini sağlayarak ürün ve hizmet süreçlerine katkıda bulunan, disiplinler arası bir yaklaşım olan veri biliminin doğmasına sebep olmuştur.

Bu alanda çalışan veri bilimcisi; istatistik, matematik, bilgisayar ve bilgisayar bilimleri mühendisi, makine öğrenmesi, programlama ve veri işleme teknolojileri gibi konulara hakimiyete ek olarak çalıştığı alana özgü bilgi birikimine sahip ve iletişim yeteneği güçlü, kurumunun geleceği için gereken kritik sorunları tespit etme ve öngörülerde bulunabilme yeteneklerine sahip olan uzmandır [9] (Şekil 2.6.). Bu uzmanların, büyük veri analizini yapabilmeleri için Şekil 2.6.’da verilen yeteneklerin yanında verilerin analizinin ve farklı yeni çıktıların üretilebilmesi için ise alan uzmanlıklarına da ihtiyaç duyulmaktadır.



Şekil 2.6. Veri Bilimcisinin Sahip Olduğu Yetenekler

Bu sayede, saklı örüntüler ortaya çıkarılabilmekte, bilinmeyen gerçekler elde edilebilmekte, korelasyonlardan verilerin zenginleştirilmesi yapılabilmekte, bilinmeyenlerden bilinmeyenleri ortaya çıkarılabilmektedir. Bunun için güçlü platformlarda gelişmiş analiz algoritmalarına, bunların büyük veri ortamlarına uygulanmasına ve en önemlisi bunları gerçekleştirilebilecek platformlara ihtiyaç vardır (Şekil 2.7.).



Şekil 2.7. Büyük veri analizi gereklilikleri

Büyük verinin analiz edilerek anlamlandırılması sürecinde; büyük hacim, karmaşıklık, veri kümesinin birçok boyutta genişlemesi, verinin otonom kaynaklardan toplanması ve dağıtık olarak kontrol edilmesi gibi karakteristik özelliklerinden ötürü zorluklar ortaya çıkmaktadır. Bu zorluklar, Veri Depolama ve Analiz, Bilgi Keşfi ve Hesaplama Karmaşıklığı, Verilerin Ölçeklenebilirliği ve Görselleştirilmesi, ve Bilgi Güvenliği gibi 4 kategoride incelenebilir.

i. Veri depolama ve analiz

Son yıllarda veri hacminin üstel olarak artması, verinin depolanması sürecinin daha pahalıya mal olmasına ve yeterli alan olmadığı için bazı verilerin silinmesine sebep olmaktadır. Diske kayıtlı verilerin okuma/yazma hızının yavaş olması, bilgi keşfi ve sunumu sürecinde analizleri zorlaştırmaktadır. Veri madenciliği süreçlerinde kullanılan yaklaşımlar, çok çeşitli ve yüksek boyutlu veriler hızlı ve verimli sonuç üretememektedir.

ii. Bilgi keşfi ve hesaplama karmaşıklığı

Kimlik doğrulama, arşivleme, yönetim, koruma, bilgi çıkarımı ve gösterimi gibi pek çok alt alanı kapsayan bilgi keşfindeki tekniklerin çoğu, probleme özgü sonuçlar üretmektedir ve çok büyük veri setleri için uygun değildirler. Bilgi keşfindeki teknikler, ardışık bilgisayarlar ve paralel bilgisayarlar için farklı karakteristiğe ve performansa sahip olabilmektedirler. Büyük veri setlerinin analizi daha büyük hesaplama karmaşıklıklarına sebep olmaktadır.

iii. Verilerin ölçeklenebilirliği ve görselleştirilmesi

Veri boyutu işlemci hızlarından çok daha hızlı ölçeklendiğinden, işlemci teknolojisinde giderek artan sayıda çekirdekle gömülü bir değişim oluşmaktadır. İşlemcilerdeki bu değişim, veri analizini hızlandırmak için paralel hesaplamanın geliştirilmesi gerekliliğine yol açmaktadır. Veriyi daha iyi sunmak ve yorumlamak amacıyla gerçekleştirilen görselleştirme araçları, büyük veriler için çoğunlukla işlevsellikte, ölçeklenebilirlikte ve zaman içindeki tepkilerde kötü performanslara sahiptir.

iv. Bilgi güvenliği

Ağın büyüklüğü, farklı türden cihazların çeşitliliği, gerçek zamanlı güvenlik izleme sistemlerinin ve saldırı tespit sistemlerinin yetersizliği gibi sebeplerden büyük veri uygulamaları güvenlik sorunlarına maruz kalmaktadır. Hassas bilgilerin korunması da büyük veri analizinde önemli bir sorundur.

2.4. Büyük Veri Teknik ve Teknolojileri

Veriler büyüdükçe, verilerin toplanması, saklanması, işlenmesi ve değerlendirilmesi için yeni algoritmalara, yaklaşımlara, metotlara ve en önemlisi de teknik ve teknolojilere ihtiyaç vardır. 2013 yılında analiz edilebilen veri miktarı 750 eksabyte iken, 2020 yılında bunun boyutunun 13,000 eksabyte olacağı tahmin edilmektedir [12]. Büyük veri ile uğraşırken karşılaşılan problemlerin üstesinden gelirken, depolama ve hesaplama süreçleri klasik yöntemlere göre farklılık göstermektedir. Değerli, gizli veya yeni bilgilerin keşfedilmesi için hem teknik ve teknolojiler için disiplinler arası çalışmalara hem de farklı metotların ve yeni yaklaşımların geliştirilmesine ve bu ortamlarda yeni çözümlerin geliştirilmesine ihtiyaç vardır.

Tablo 2.2. Büyük Veri İşleme Platformları ve Araçları

Platform		**Lokal**	Hadop, Spark, MapR, Cloudera, Hartonworks,InfoSphere, IBM BigInsights, Asterix
		Bulut	AWS EMR, Google Compute Engine, Microsoft Azure, Pure System, LexisNexis HPCC Systems
Veri Tabanı		**SQL**	Greenplum, Astar Data, Verica, SpliceMachine
		IN-MEMORY	SAP HANA
	NoSQL	**Sütun Şeklinde**	Hbase, HadoopDB, Cassandra, HyperTable, BigTable, PNUTS, Cloudera, MonetDB, Accumulo, BangDB
		Anahtar-Değer	Redis, Flare, Sclaris, MemcacheDB, HyperTable, Valdemort, Hibari, Riak, BerkeleyDB, DynamoDB, Tokyo
		Doküman Tabanlı	SimpleDB, RavenDB, ArengoDB, Mango DB, Terrastore, CouchDB, Solr, Apache Jackrabbit, Basex, OrientDB
		Graf Tabanlı	Neo4j, InfoGrid, Infinite Graph, OpenLink, FlockDB, Meronymy, AllegroGraph, WhiteDb, TITAN, Trinity
Veri İşleme			MapReduce, Dryad, YARN, Storm, S4, Big Query, Pig, Impala, Hive, Flink, Spark, Sazma, Heron
Veri Ambarı			Hive, HadoopDB, Hadapt
Veri Birleştirme ve Transfer			Spoop, Flume, Chukwa, Kafka, ActiveMQ
Sorgu Dili			Pig Latin, HiveQL, DryadLINQ, MRQL, SCOPE, ECL, Impala
İstatistik & Makine Öğrenmesi			Mahout, Weka, R, SAS, SPSS, Pyton, Pig, RapidMiner, Orange, BigML, SkyTree, SAMOA, Spark MLLib, H2O
İş Zekası			Talend, Jaspersoft, Pentaho, KNIME
Görselleştirme			Google Charts, Fusion Charts, Tableau Software, QlikView
Sosyal Medya			Radian6, ClaraBridge

Tablo 2.2.'de kategorileştirildiği gibi büyük veri araçları genellikle açık kaynaklı olmakla beraber, dağıtık dosya sistemleri, paralel hesaplama algoritmaları veya NoSQL (Not Only SQL) veri tabanları kullanmaktadırlar.

2.5. Büyük Verinin Güvenlik ve Mahremiyeti

Bilgi varlıklarının hacmi veya boyutu gün geçtikçe artırmakta, bununla beraber bu verilerin mahremiyeti ve güvenliği de her geçen günden daha fazla önem arz etmektedir. Zamanla veriden üretilen katma değerlerin organizasyonlarda, toplumlarda, devletlerde ve hatta dünya genelinde önemi ve bu katma değerlere olan ihtiyaç arttığından büyük verinin güvenliği ve mahremiyeti her zamankinden daha fazla ilgilenilmesi gereken sorunların başında gelmektedir.

Şekil 2.8. Bilgi varlıklarının güvenliği

Büyük veride gizlilik, güvenlik ve mahremiyet normal verilerimizde veya sistemlerimizde olduğu gibi devam etmektedir. Mevcutlardan temel farkları hem normal verilerin işlenmesinden, saklanmasından ve analizinden daha farklı olarak, tahmin edilemeyen pek çok tehlikeleri içerisinde barındırması hem de bu sistemlerde oluşabilecek ihlallerin yapılabilecek analizlerle tespit edilmesidir. Diğer bir ifadeyle, büyük verinin gelişimi, güvenlik, gizlilik ve mahremiyet ihlali endişeleri oluştursa da büyük veri yaklaşımları ile başka sistemlerin güvenliği sağlanmaktadır [14]. Log kayıtları, sistem olayları, ağ trafikleri, SIEM (Security Information and Event Management) uyarıları, siber saldırı örüntüleri, iş süreçleri ve diğer bilgi kaynakları kullanılarak pek çok analiz ve karar destek sistemleri geliştirilmektedir. Bu sistemler, anomali tespiti, saldırı tespiti, zararlı yazılım tespiti, dolandırıcılık tespiti, APT (Advanced Persistent Threats) tespiti, gerçek zamanlı savunma sistemleri geliştirme ve adli bilişim amacıyla yeni sistemler geliştirilmesidir.

Bulut gibi altyapılar sayesinde büyük veriye erişim ucuzlamakta ve kolaylaşmakta fakat beraberinde de güvenlik, gizlilik veya mahremiyet problemlerini getirmektedir. Karşılaşılan veya karşılaşılabilecek problemler ve çözüm önerileri aşağıda verilen başlıklarda özetlenmiştir.

i. Altyapı güvenliği

Büyük veri altyapıları pek çok alt sistemden oluştuğu için bu ortamlarda ihlallerle karşılaşılabilmektedir.

Büyük miktarlardaki verinin işlenmesini ve depolanmasını sağlayan dağıtık işlemlerin güvenliği, işlem düğümlerinin hem bölme hem de birleştirme aşamasında sağlanmalıdır.

İlişkisel olmayan NoSQL gibi veritabanları için hem enjeksiyon gibi klasik tehditlere karşı hem de sisteme özgü ek güvenlik önlemleri alınmalıdır.

ii. Veri gizliliği ve mahremiyeti

Verilerin işlenmesi, taşınması, görselleştirilmesi aşamalarında mevcut yaklaşımlardan farklı ihlallerle karşılaşılabilmektedir. Veri madenciliği ve analitik süreçlerinin, gizliliği ve mahremiyeti koruyarak ve suiistimalleri önleyerek gerçekleştirilmesi gerekmektedir.

Verilerin görünür hale getirilmesi, sistemlere erişimin sınırlandırılması, verilerin anonimleştirilmesi ya da verinin şifrelenmesiyle gerçekleştirilmelidir.

iii. Veri yönetimi

Veri ve işlem günlüklerinin, katmanlı bir yapı içerisinde saklanması yönetimi zorlaştırsa da güvenlik, elastiklik ve ölçeklenebilirlik açısından avantajlar sağlamaktadır. Her ne kadar gerçek zamanlı veri yönetimi yapılsa da bazı saldırılar ya da yanlış olan doğru pozitif (true positive) değerleri sistemde zafiyetlere sebep olmaktadır. Bu yüzden hesapların denetimi, ne olduğunun ve neyin yanlış gittiğinin anlaşılmasını kolaylaştıracaktır. Verinin kaynağının bilinmesi, graflar şeklinde analizinin yapılması, uygulamaların doğrulanmasını, gizlilik ve güvenliğinin sağlanmasını kolaylaştıracaktır.

iv. Bütünlük ve tepkisel güvenlik

Birçok cihazdan ve uygulamadan toplanan verinin sunum aşamasında filtrelenip doğrulanarak kullanılması gerekmektedir.

Tüm koruma ve güvenlik önlemleri alındıktan sonra, büyük veri altyapısının ve analizlerinin gerçek zamanlı olarak gözetilmesi gerekmektedir.

2.6. Açık Veri

Açık veri (open data), "herkesin ücretsiz ve özgürce erişebileceği, kullanabileceği, dağıtılabileceği ve değerler üretebileceği" veridir. Verilerin makine tarafından okunabilir biçimde, toplu olarak ve açık lisanslı bir şekilde bulunması gerekir. Açık verinin temel özellikleri aşağıdaki gibidir.

i. Kullanılabilirlik ve erişim

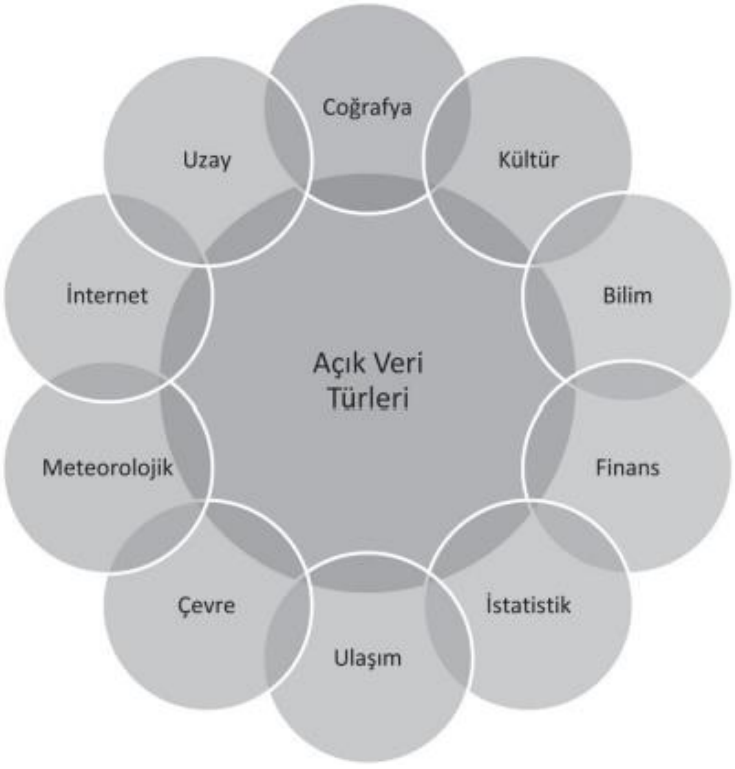
Veri bir bütün olarak, yeniden üretim maliyetini aşmayacak şekilde, tercihen internet üzerinden indirilebilir ve uygun bir biçimde mevcut olmalıdır.

ii. Tekrar kullanım ve yeniden dağıtım

Veri, diğer veri kümeleri ile karıştırılarak kullanılması dahil olmak üzere, yeniden kullanıma ve yeniden dağıtılmaya izin veren şartlar altında sağlanmalıdır.

iii. Evrensel katılım

Veri, kişilere veya gruplara karşı herhangi bir ayrımcılık yapılmadan herkes için kullanılabilir, ve dağıtılabilir olmalıdır.



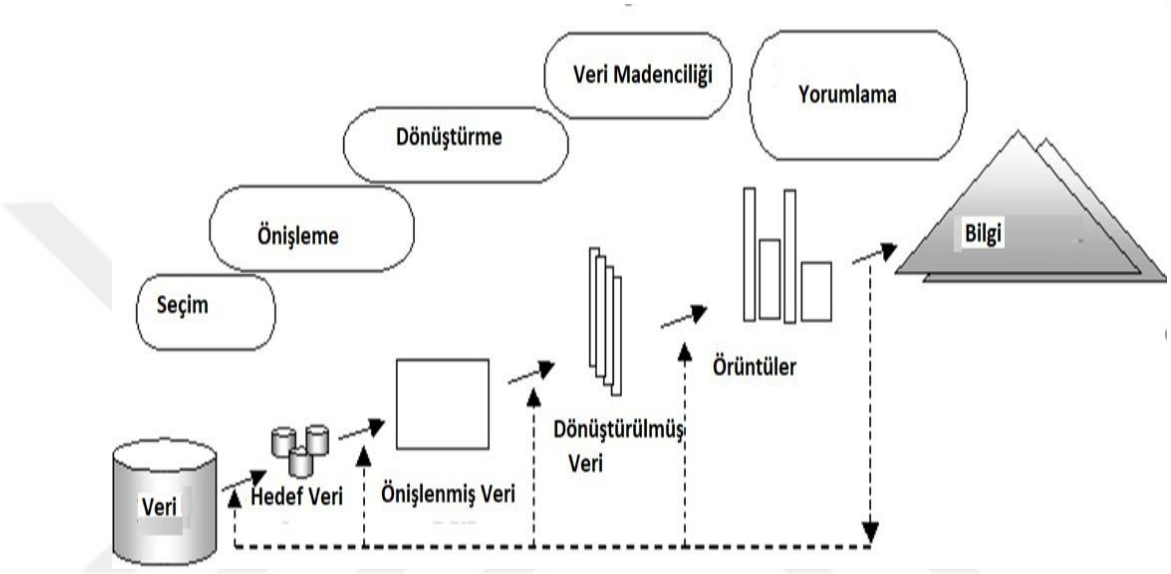
Şekil 2.9. Açık veri türleri

Verilerin üretildiği ve yayınlandığı genel alanlar Şekil 2.9.'da özetlenmiştir. Sadece o alana ait ham veri kaynakları değil, verilerin herkesin kullanımına açılmasıyla, etkileşimli bir ekosistem oluşturulabilecektir. Bu sayede; veriyi açık olarak sunan yayıncılar ile kamu kurumları arasında iletişim ve karşılıklı katkılar artabilecek, şeffaflık, teknolojik yenilik, kamu hizmeti ve ekonomik büyüme odaklı ve etkileşimli yeni bir model oluşturulabilecektir.

Açık veriler sayesinde birçok kazanım farklı kurum veya kuruluşlarca elde edinilebilir. Açık veri üniversitelerde çeşitli yayınlar, kitaplar vb. akademik çıktılarda kullanılabileceği gibi bu veriler ile çeşitli teknik bilgiler ile kullanılarak var olan kazanımlardan daha fazla etki elde edilebilecektir. Yine bu veriler ile çeşitli planlamalarda, devlet veya kurum politikaların geliştirilmesinde, yeni teşvik mekanizmaları geliştirilmesinde ve olası yeni işbirliklerinin oluşturulmasında, yeni süreçlerin geliştirilmesinde, hantal devlet yapılarının iyileştirilmesinde kullanılabilir. Ülkemizde açık büyük veri çalışmalarının önünde, politika eksikliği, insan kaynağı, araştırma olanakları ve altyapı eksikliği ve yetersizliği gibi sebeplerden dolayı, açık büyük veri çalışmaları beklenen düzeyde değildir. Bu konu ile ilgili detaylar takip eden bölümlerde daha detaylı tartışılacaktır.

3. VERİ MADENCİLİĞİ

Veri madenciliği; herhangi tipteki veri depolama alanlarında (veri tabanı, veri depoları vs.) saklanan çok yüksek miktardaki bilgilerin içindeki örüntüler, değişimler, anomaliler veya önemli yapılar gibi ilginç bilgileri bulma işlemidir. Elektronik ortamdaki çok büyük miktardaki verinin işlenme ve anlamlı bir hale getirilmesi ihtiyacından dolayı özellikle piyasa analizi ve karar destek işlemleri için kaçınılamaz bir ihtiyaçtır.[17]



Şekil 3.1. Veri madenciliği süreci

3.1. Veri Madenciliği Yöntemleri

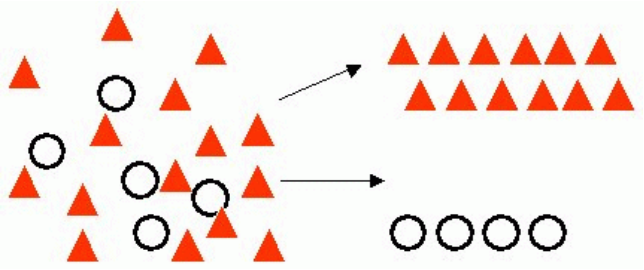
Veri madenciliğinde hem verinin tipine hem de elde edilmek istenilen bilginin türüne (yoğunluk, kategorizasyon, gelecek tahmini vb.) göre pek çok farklı yaklaşım vardır. Bunlardan en fazla kullanılanları şu şekildedir.

3.1.1. Sınıflandırma

Sınıflandırma yönelimi veriler ve meta veriler hakkında önemli ve ilgili bilgileri almak ve bu bilgileri farklı sınıflara ayırmak için kullanılır. Sınıflandırma, aynı zamanda, veri kayıtları sınıflar olarak, kullanıcı verilerin sınıf bilgilerine de sahip olacaklardır.

Sınıflandırma , denetimli bir öğrenme tekniğidir. Bu öğrenme tekniğinde sınıf sayısı ve bir grup verinin hangi sınıfta olduğu bilinir.

Klasik bir sınıflandırma yöntemi uygulaması olarak mail kutularımız verilebilir. Mail kutularımızdaki gerçek ve spam emailler bu tip algoritmalar yardımıyla saplanır.



Şekil 3.2. Sınıflandırma işlemi

3.1.2. Ortaklık Kuralı

Ortaklık, genel olarak örüntüleri izlemekle ilgilidir ama en güçlü tarafı büyük miktardaki verinin içinde birbirleriyle bağımlı ve bağlantılı değişimleri bulmaktır. Bu teknik en fazla marketlerde ürün yerleşimlerinde kullanılır.

3.1.3. Karar Ağacı

Karar ağaçları veriyi kategorize veya tahmin etmek için kullanılır. Karar ağaçları iki veya daha çok cevabı bulunan bir temel soru ile başlar. Her cevap kullanıcıyı bir sonraki soruya yönlendirir ve bu cevaplara göre tahmin ve kategorizasyon yapılır.

3.2. Kümeleme

Kümeleme, verinin benzer nesnelerden oluşturulmuş gruplara bölünmesidir. Kümeleme işleminde küme içindeki elemanların benzerliği fazla, kümeler arası benzerlik ise az olmalıdır. Bir kümeleme yönteminin kalitesi bu prensibi sağlaması ile ölçülür. Kümeleme yöntemi seçimi kullanılacak veri türüne ve uygulamanın amacına göre farklılık gösterir.[17]

Kümeleme, denetimsiz bir öğrenme tekniğidir. Denetimsiz tekniklerde hangi nesnenin hangi sınıfa ait olduğu ve grup sayısı belirsizdir. Kümeleme işlemi tamamen gelen veriye göre değişir.

Kümeleme yöntemleri genel olarak hiyerarşik ve hiyerarşik olmayan(bölücü) kümeleme yöntemleri olarak iki bölümde incelebilirler.

Şekil 3.3. Genel hatlarıyla kümeleme yaklaşımları

3.2.1. Ayırma tipi kümeleme

Hiyerarşik olmayan yöntemler içerisinde en çok kullanılan yaklaşımdır. Bu tip algoritmalar genel olarak tüm noktalar ve bu noktaların merkezlerini inceleyerek bu merkezleri, noktalara olan uzaklığını minimum olana kadar değiştirir. En yaygın kullanılan örneği K-Means algoritmasıdır. Ayırma yaklaşımında uzaklıklar Euclidean uzaklık, manhattan uzaklığı, Minkowski uzaklığı gibi ölçülerle alındığı için dairesel alan kurma becerileri zayıftır.

3.2.2. Yoğunluk tabanlı kümeleme

Hiyerarşik olmayan yöntemler arasında en popüler olan diğer yaklaşımdır. Bu tarz kümelemelerde verilerin sıklığına göre bir kümeleme işlemi yapılır.

Kümelerin seyrek olduğu yerler ise gürültü olarak isimlendirilir. En yaygın örnekleri DBSCAN ve OPTICS algoritmalarıdır.

3.3. K-Means Algoritması

Hiyerarşik olmayan yaklaşımlardan ayırma tipinin en iyi örneği olan K-Means en ilkel ve en çok kullanılan kümeleme algoritması tekniğidir. İsmindeki k, eldeki n adet verinin k merkezlerine bölünerek k adet küme elde edilmesinden gelir.

Algoritma genel olarak tüm noktalar ve bu noktaların merkezleri, noktalara olan uzaklığını minimum olana kadar değiştirir. Verilen bir k değeri için genel algoritma şu şekildedir;

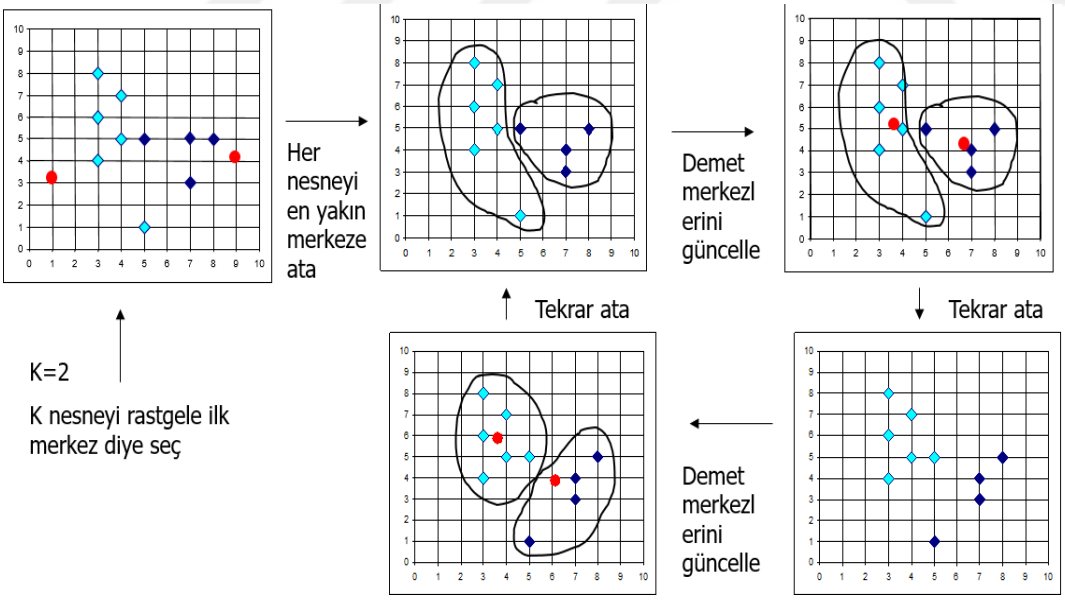
3.3.1. Çalışma Adımları

1- Veriler gilişigüzel k parçaya ayırılır

2-Her parçanın orta noktası hesaplanır.

3-Her nokta kendine en yakın merkezin olduğu parçaya atanır

4-Verilerin parçalanmasında hiç bir değişiklik olmayana kadar adım 2 ye dönülür.



Şekil 3.4. k=2 için K-MEANS algoritması çalışma adımları

3.3.2. Sözde Kod

1. Küme sayısı seçilir (K) ve örneklem noktaları alınır
2. c_1, c_2, ... c_k merkezleri rasgele yerleştirilir
3. Merkezler sabit olana kadar 4. ve 5. adımları tekrarlanır
4. Her bir x_i noktası için
- En yakın merkez bulunur (c_1, c_2 .. c_k)
- Noktalar merkezlere atanır
5. Atanan noktalar için yeni merkezler bulunur.

3.3.3. Avantajları

K-Means en ilkel kümeleme yöntemlerinden olduğu için programlama dillerine implemente edilmesi ve anlaşılması daha kolaydır.

Euclidian uzaklık formülüne göre çalışır.

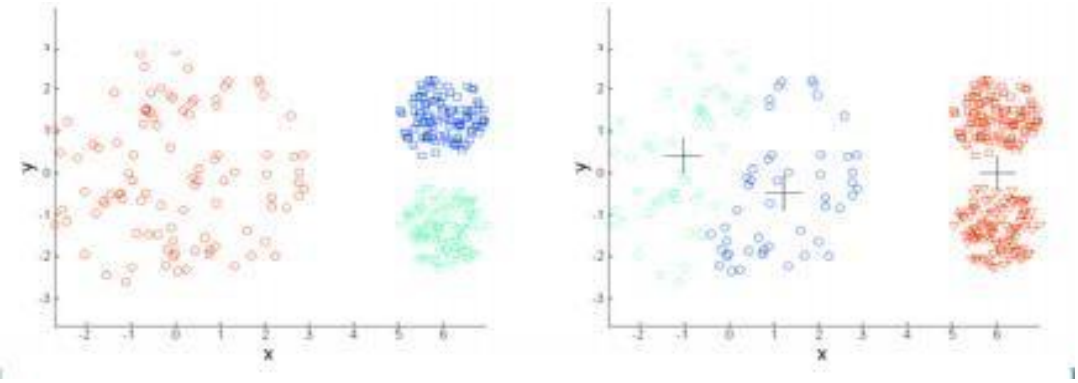
Karmaşıklığı O(tkn)'dir, k: küme sayısı, t: tekrar sayisi, n: veri sayısıdır. Başlangıçta k değeri bilinmelidir.

3.3.4. Dezavantajları

Sadece ortalamanın tanımlı oldluğu verilerde çalışır, kategorik veride kullanılamaz.K değerini dışardan aldığı için her zaman aynı sonucu vermez. Bu yüzden de demetler anlamsız yığınlar olabilir. (Şekil 3.5.)

Gürültü ve sapan veriye karşı zayıftır.

Konveks olmayan şekillerde anlamlı küme üretmez.(Şekil 3.5.)



Orijinal noktalar K-Means (3 küme)

Şekil 3.5. Anlamsız K-Means kümelemeleri

3.4. DBSCAN Algoritması

DBSCAN (Density Based Spatial Clustering of Applications with Noise) yani gürültülü uygulamalara yoğunluk esaslı mekansal kümeleme, bir yoğunluk temelli kümeleme yöntemidir. Martin Ester, Hans-Peter Kriegel, Jörg Sander ve Xiaowei Xu tarafından KDD'96 konferansında sunulmuştur. .[18]

Algoritma genel olarak nesnelerin komşuları ile olan mesafelerini hesaplayarak, belirli bölgede önceden belirlenmiş eşik değerinden fazla nesne bulunan alanları gruplandırarak kümeleme yapar.

3.4.1. Parametreler ve bazı tanımlar

DBSCAN algoritması çalışmaya başlamak için bazı parametreler alır ve bu parametrelere göre çalışma zamanında bazı tanımlamalara göre karar alır.

Algoritma çalışmasını daha iyi kavrayabilmek için bunlar bilinmelidir.

Eps : Bir nesneye komşu olabilmek için gereken maksimum uzaklık,

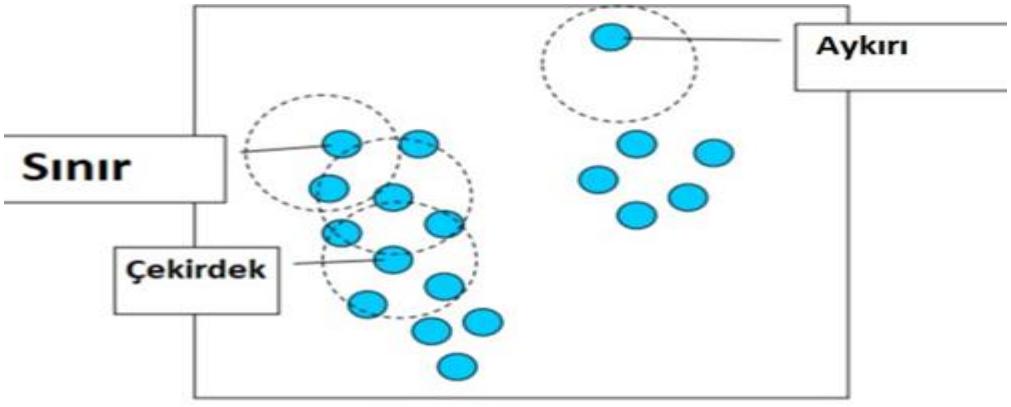
MinPts: Eps yarıçaplı komşulukta bulunması gereken en az nesne sayısı

Bir nesnenin komşuluk bölgesinde MinPts kadar nesne varsa bu nesneye çekirdek nesne denir.

Eps komşuluk: İki noktanın arasındaki mesafe Eps'den küçük olursa o iki küme komşudur denilir.

Sınır Noktası, Eps içinde MinPts'den daha az nesneye sahip ama çekirdeğin komşuluğundaki noktadır.

Gürültü noktası, ne çekirdek ne de sınır noktası olan her noktaya verilen addır.



Şekil 3.6. Sınır, Çekirdek ve Aykırı noktalar (Eps=1 birim, MinPts=5)

Doğrudan erişilebilir nesne: Bir nesne Eps ve MinPts için p nesnesine q nesnesinden doğrudan erişilebilirdir diyebilmek için şu şartlar sağlanmalıdır. (Şekil3.7.)

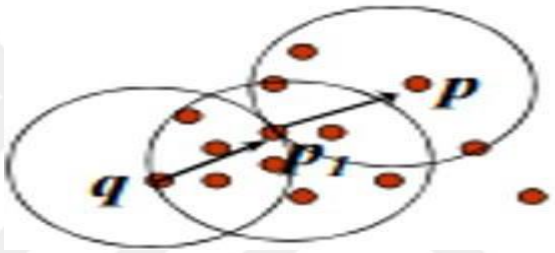
P, q'nun eps komşusudur.

q çekirdek nesnedir.

Erişilebilir Nesne: Eps ve MinPts koşulları altında, p1,p2,...,pn nesne dizisi olması,

p1=q,pn=p,

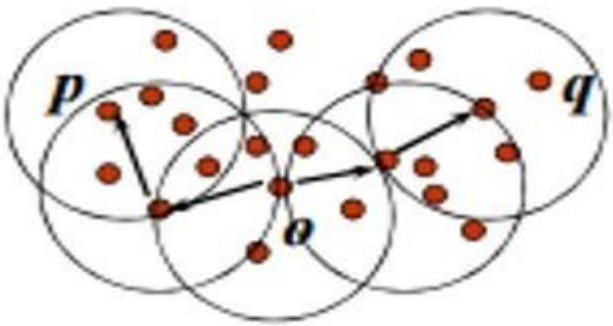
*pi nesnesinin doğrudan erişilebilir nesnesi pi+1 olur(Şekil3.8.)



Şekil 3.8. Erişilebilir nesne

Yoğunluk bağlantılı nesne:Eps ve MinPts koşulları altında q nesnesinin yoğunluk bağlatılı nesnesi şu koşulları sağlar:

p ve q nesneleri Eps ve minPts koşulları altında bir o nesnesinin erişilebilir nesnesidir.



Şekil 3.9. Yoğunluk bağlantılı nesne

3.4.2. Çalışma adımları

i. Veri setindeki her nesne için Eps uzaklıktaki komşulukları bulunur.

ii. p nesneleri çekirdek nesne olacak şekilde kümeler oluşturulur.

iii. Bulunan p nesnelerin doğrudan erişebildikleri nesneler bulunur.

iv. Çekirdek nesnelerinin erişilebilir nesneleri bulunur.

v. Yoğunluk bağlantılı kümeler birleştirilir.

vi. Hiçbir yeni nesne bir kümeye eklenemeyene kadar bu döngü devam eder.

3.4.3. Sözde Kod

DBSCAN algoritmasının sözde kodu aşağıdaki gibidir.

```
DBSCAN(D, epsilon, min_points):
C = 0
for each unvisited point P in dataset
        mark P as visited
        sphere_points = regionQuery(P, epsilon)
        if sizeof(sphere_points) < min_points
                ignore P
        else
                C = next cluster
                expandCluster(P, sphere_points, C, epsilon, min_points)


expandCluster(P, sphere_points, C, epsilon, min_points):add P to cluster C
    for each point P' in sphere_points
        if P' is not visited
            mark P' as visited
            sphere_points' = regionQuery(P', epsilon)
            if sizeof(sphere_points') >= min_points
                sphere_points = sphere_points joined with sphere_points'
            if P' is not yet member of any cluster
                add P' to cluster CregionQuery(P, epsilon):
    return all points within the n-dimensional sphere centered at P with
radius epsilon (including P)
```

3.4.4. Avantajları

Dairesel olmayan kümeleri bulabilir. (Şekil 2.10.) Karmaşıklığı uzaysal sorgulama gibi çeşitli indeksleme algoritmaları kullanarak O(nlog n)'den O(logn)'e düşürülebildiğinden efektif bir yöntemdir.

Gürültülü veri setlerinde çalışabildiği için bulduğu kümeler daha anlamlıdır.

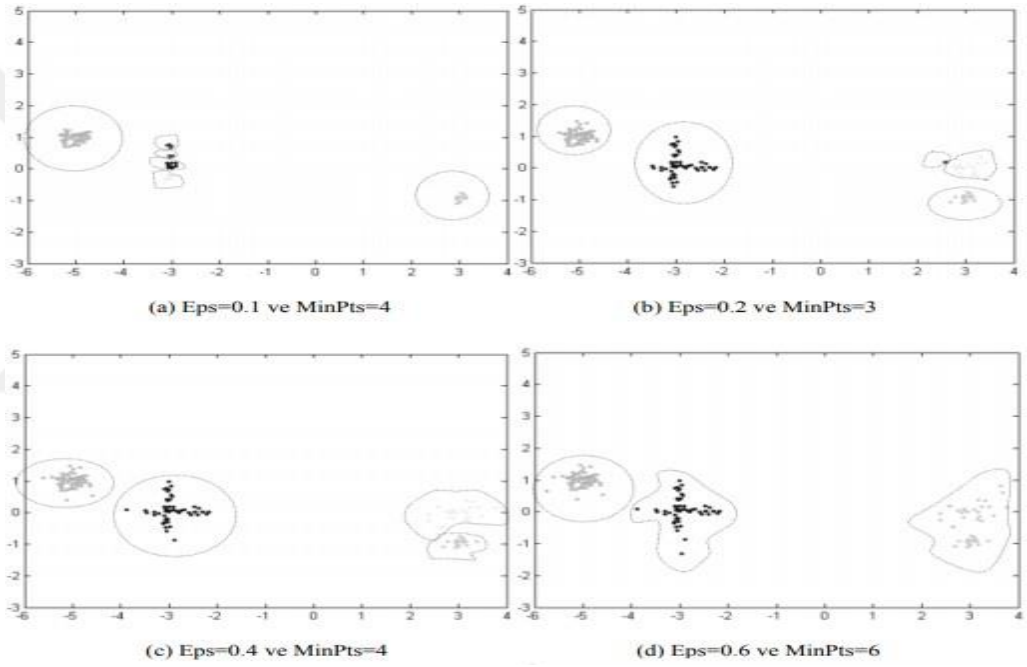
3.4.5. Dezavantajları

Giriş parametrelerini kestirmek zordur.

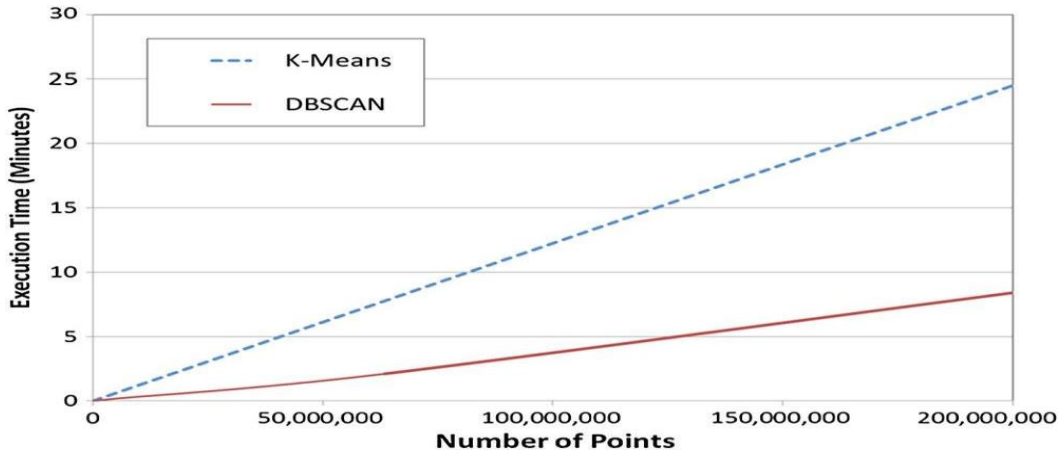
Yapısı gereği tüm veri setini defalarca okuyacağı için çalışma zamanı veri seti büyüdükçe çok fazla artar.

Algoritma Eps ve MinPts değerlerini kullanıcıdan alıp çalışmaya başladığı için farklı yoğunluklardaki kümeleri bulamaz.

Giriş parametrelerine göre sonuçlar çok fazla değişebilir.(Şekil 3.10.) Yeni gelen bir elemanın hangi kümeye dahil olduğunu bulamaz.



Şekil 3.10. Farklı girdi parametrelerine göre aynı veri setinin yorumlanması



Şekil 3.11. Aynı veriseti üzerinde K-MEANS ve DBSCAN algoritmalarının çalışma süreleri

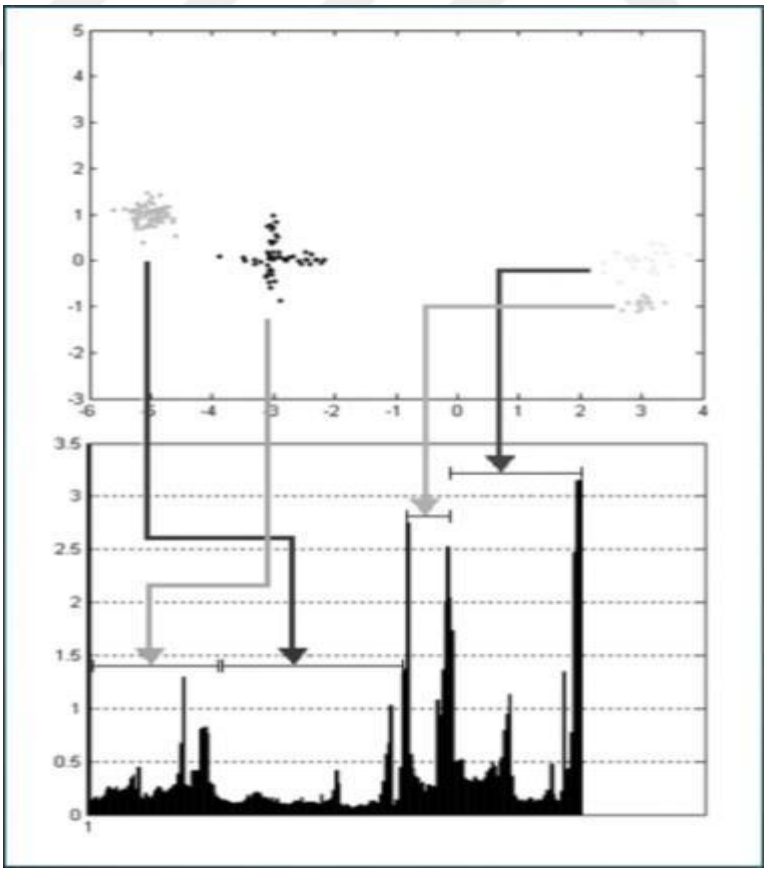
3.5. OPTICS Algoritması

OPTICS (Ordering Points To Identify the Clustering Structure) kümeleme yapısını anlamak için noktaların sıralanmasıdır. Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, Jörg Sander tarafından SIGMOD'99 konferansında sunulmuştur [20].

OPTICS başlı başına bir algoritma olmaktan çok DBSCAN algoritmasının geliştirilmiş hali olarak düşünülebilir. DBSCAN'în en büyük zaafı olanm Eps değerinin kullanıcı tarafından girilmesinden kurtulup bu değeri algoritma çalışırken dinamik bir şekilde değiştirebilir. Bu sayede OPTICS algoritması tek çalışmada, DBSCAN algoritmasının bir fazla uzaklık parametresini bulup gösterme yeteneğine sahiptir. (Şekil 3.12.)

Algoritma çalışmaya başlamak için sadece MinPts parametresine ihtiyaç duyar.

Veritabanındaki nesneleri sıralar ve her bir nesne için iç-mesafe ve ulaşılabilirlik mesafesi değerlerini hesaplar ve kaydeder böylece tek bir kümeleme yapısı yerine üzerine daha fazla analiz yapılabilecek bir sıralama oluşturur.



Şekil 3.12. MinPts=4 için OPTICS sonucu ve bu sonucun DBSCAN ile karşılaştırılması

3.5.1. Sözde Kod

```
procedure OPTICS(X, ε, minpts, O)
pos ← 0
    for each unprocessed point x ∈ X do
        mark x as processed
        N ← GETNEIGHBORS(x, ε)
        SETCOREDISTANCE(x, N, ε, minpts)
        O[pos] ← x; pos ← pos + 1
        RD[x] ← NULL
        if CD[x] <> NULL then
            UPDATE(x, N, Q)
            while Q <> empty do
                y ← EXTRACTMIN(Q)
                mark y as processed
                N! ← GETNEIGHBORS(y, ε)
                SETCOREDISTANCE(y,N! , ε, minpts)
                O[pos] ← y; pos ← pos + 1
                if CD[y] <> NULL then
                    UPDATE(y,N! , Q)

procedure UPDATE(x, N, Q)
    for each unprocessed point x! ∈ N do
        newD = MAX(CD[x], DISTANCE(x! , x))
        if RD[x! ] = NULL then
            RD[x! ] ← newD
            INSERT(Q, x! , newD)
        else if newD < RD[x! ] then
            RD[x! ] ← newD
            DECREASE(Q, x! , newD)
```

3.5.2. Avantajları

Tek bir kümeleme yapısı yerine üzerine daha fazla analiz yapılabilecek bir sıralama oluşturur.

DBSCAN algoritmasından farklı olarak Eps değeri almadığı için farklı yoğunluklardaki kümeleri de dinamik olarak tespit edebilir.

3.5.3. Dezavantajı

DBSCAN algoritmasına göre daha karmaşık bir yapısı vardır bu yüzden uygulama geliştirmesi daha zahmetlidir.

3.6. CURE Algoritması

Cure algoritması, dışardan 3 parametre alarak çalışan bir kümeleme algoritmasıdır. Parametreleri oluşmasını istediğimiz küme sayısı, kümenin örneklem nokta ile temsil edildiği ve her kümeyi ayırdığında kümedeki elemanları merkeze yaklaştırmak için kullanılan daraltma katsayısıdır. İlk başta rastgele sayıda aldığı kümeleri merkezi en yakın diğer bir kümeyle birleştirir bu kümenin merkeze en yakın örneklem sayısı kadar elemanını alır ve daraltma katsayısıyla çarparak elemanları merkeze yaklaştırır. Bu işlem küme sayısı olmasını istediğimiz küme sayınına eşit olana kadar devam eder. Cure algoritmasının optimal kümeleme olasılığı küme sayısı, kümenin örneklem nokta sayısı ve daraltma katsayısına bağlıdır. Algoritmanın sözde kodu aşağıdaki adımlardan oluşur.

1. Girdi olarak aldığımız örneklem sayısı kadar dağınık örneklem noktadan oluşan kümeler seçilir.

2. Seçilen kümeler arasında uzaklık bulunur. (İki küme arasında uzaklığı bulmamız için küme merkezlerini bulup merkezler arasındaki mesafeleri bulmamız yeterlidir.)

3. Merkez uzaklığı en yakın 2 küme birleştirilir.

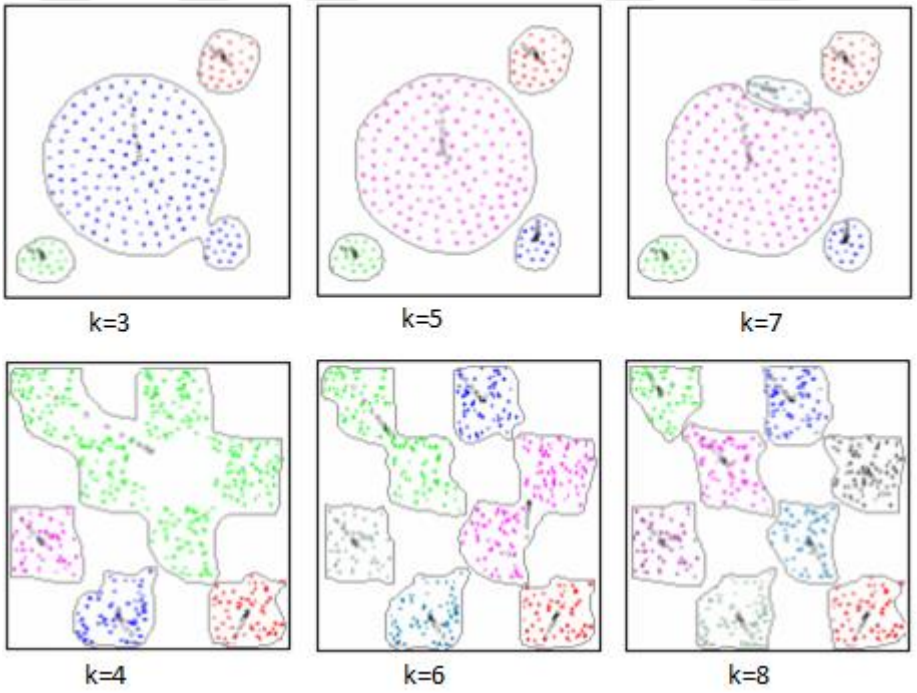
4. Birleşen iki kümeden oluşan yeni kümenin merkeze en yakın c adet elemanı seçilir. Seçilen noktalar α örneklem katsayısıyla çarpılarak noktalar küme merkezine yaklaştırılıyor.

5. Küme sayısı, algoritmada girdi olarak aldığımız k küme sayısı değerine eşitlenene kadar 2, 3 ve 4. adımarı tekrarlanır.

Daha önce belirtildiği gibi, CURE algoritmasının optimal sonuç vermesi, küme sayısı, örneklem nokta sayısı ve daraltma katsayısı gibi üç önemli faktöre bağlıdır Aşağıda bu bağlılığı görebilmemiz için bir matlab çalışmasını inceleyeceğiz.[25]

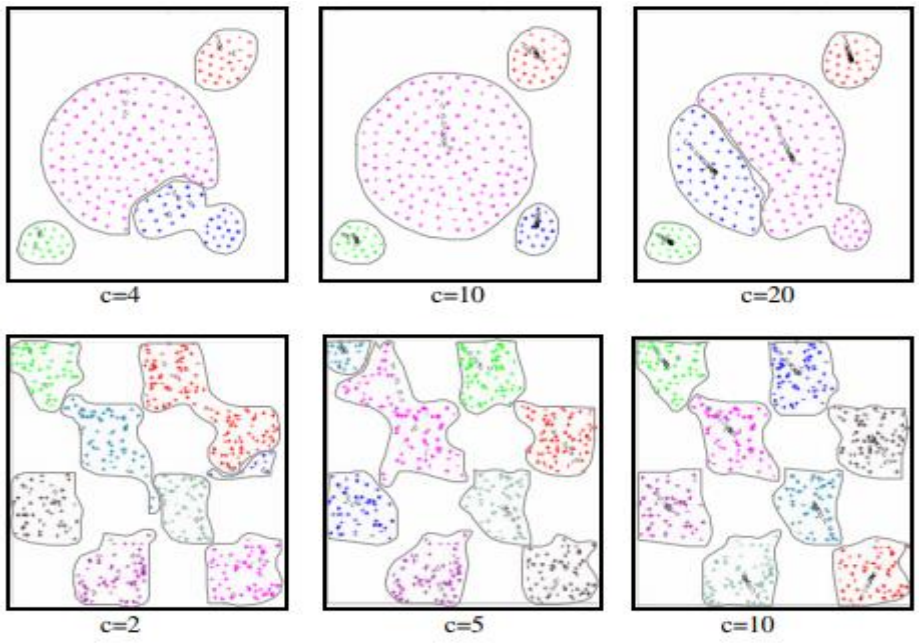
Algoritmanın (k) küme sayısına bağlılığının incelenmesi: Algoritmada k parametresi iterasyonun dolayısıyla algoritmanın sonlanma koşuludur. k değerinin oluşan kümeleme işleminin optimalliği üzerindeki etkisini görmek için k=3, 5, 7 tek sayıları yuvarlaklar veri setinde ve kareler veri setinde k=4, 6, 8 cift sayıları seçilerek CURE algoritması matlabda çalıştırılmıştır. k değerinin algoritmaya etki edip etmediğini açık olarak görebilmemiz için örneklem nokta sayısı ve daraltma katsayısı değerleri sabit tutulmuştur.

Kümelerin örneklem noktaları, kümelerde gri gösterilmektedir. Şekil 3.14.'de CURE algoritmasının yuvarlaklar veri seti üzerinde, daraltma katsayısı α=0.2 ve örneklem nokta sayısı c=10 değerleri sabit tutularak, farklı k değerleri seçilerek uygulanmanın sonuçları görülmektedir. Küme sayısı k=3 için bulunan kümelenme sonucunda 2. ve 4. kümeler birleşmekte, k=4 seçildiğinde ideal sonuç elde edilmekte ve k=5 alındığında veri setindeki ideal bölünmenin en büyük kümesi bölünerek yeni bir küme oluşmaktadır. İnceleme sonucu olarak CURE algoritmasında yuvarlak kümelerin ideal sonuçunu tespit etmek için örneklem nokta sayısı ve daraltma katsayısı faktörlerini farklı değerlerle denenmesi gerekmektedir. Kareler veri setinde farklı k küme sayıları için α=0.3 ve c=10 alınarak algoritma çalıştırılmıştır, sonuçlar Şekil 2.13.'de görülmektedir. k=4 seçildiğinde 1, 2, 3, 4 ve 6. karelerdeki veriler 1. kümeye, 5, 7, ve 8. kareler ise tek başına kümelenerek 2., 3. ve 4. kümeleri oluşturur. k=6 değeri için elde edilen kümelenmede, 1. ve 3. kareleri 1. kümeye, 4. ve 6. kareler birleşerek 3. kümeye gitmiştir. Diğer kümeler şekilden görüldüğü gibi doğru şekilde kümelenmiştir. Veri setindeki en optimal sonuç k=8 durumunda bulunmuştur. CURE algoritması yuvarlak ve kare şeklindeki verileri de başarıyla kümelendirebilmektedir.



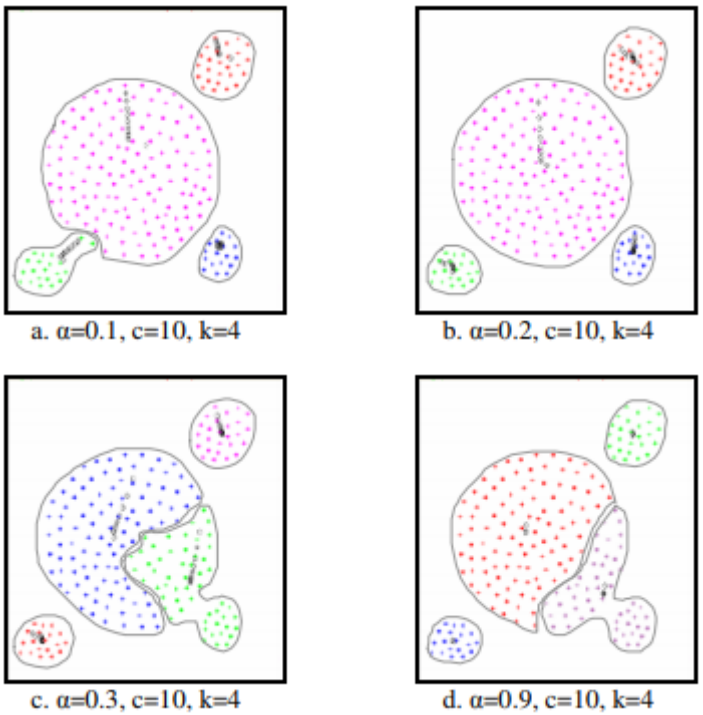
Şekil 3.14. Yuvarlaklar α=0.2, c= 10, k=3, 5, 7
Kareler α=0.3, c= 10, k=4, 6, 8

Örneklem nokta sayısının (c) algoritmadaki etkisinin incelenmesi sonuçları aşağıdadır. Nokta sayısı (c), kümeleri temsil etmek için kullanılan eleman sayısıdır. Şekil 3.14.'de yuvarlaklar veri setinde daraltma katsayısı α=0.2 ve küme sayısı k=4 alınarak c=4, 10, 20 değerleri için algoritmanın sonuçları görülmektedir. Örneklem nokta sayısı c=4 küçük bir değer olduğundan küçük ve ayrık kümelerin geometrik şekillerini bulabilmekte, ancak büyük veri yavarlağını iki kümeye bölmektedir. Örneklem nokta sayısı c=10 için kareler veri setinde de olduğu gibi optimal bir sonuç gerçekleşmektedir. Örneklem nokta sayısı c=20 için, örneklem noktaların kendi aralarında uzun bir küme oluşturarak uzamış kümeler bulma eğiliminde oldukları görülmektedir. c yüksek bir değer aldığında, bu algoritma tüm-noktalar yaklaşımlı algoritmalara benzer davranış gösterir ve uzatılmış biçimde görülen kümeler bulmaya çalışır. Bu durum, algoritmanın yavaş çalışmasına ve aralarında ortak komşu noktalar bulunan ayrık veri gruplarının birleşmesine neden olmaktadır. Şekil 3.15.'de kareler veri setinde α=0.3 ve k=8 olduğunda, c=2, 5, 10 değerleri için CURE algoritmasının çalışma sonuçları görülmektedir. c=2 durumu için 2. ve 4. Kareleri bir kümeye ve 4 no.lu karenin bazı noktaları da yeni bir küme oluşturmuş, 3 nolu kareye ait olan bazı noktalar da 1 ve 6 no.lu karelerin kümelerine girmiştir. c=5 için 1 numaralı küme ikiye bölünmüştür. c= 10 durumunda ise, kümeler iyi bir şekilde oluşmaktadır. Örneklem nokta sayısı (c) küçük seçilme durumunda, algoritma merkez nokta tabanlı algoritmalar gibi çalışmakta ve genellikle kötü sonuç üretemektedir. Bundan dolayı da geometrik şekiller bulunamamaktadır.

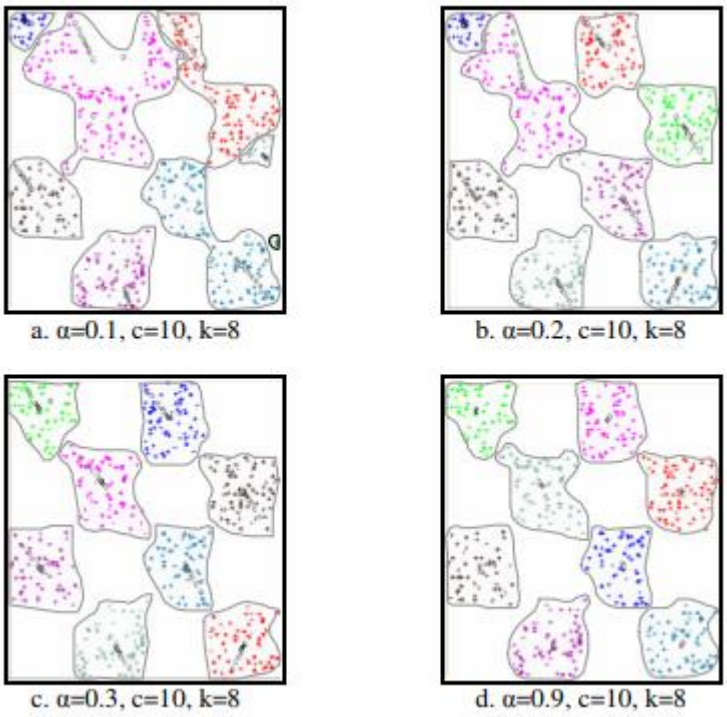


Şekil 3.15. Yuvarlaklar α=0.2, k=4, c=4, 10, 20
Kareler α=0.3, k=8, c=2, 5, 10

Daraltma katsayısının (α) algoritmaüzerinde etkisi aşağıdaki gibidir. CURE algoritmasında α'nın kümeleme işlemini ne kadar etkilediğini incelememiz için yuvarlaklar ve kareler veri setlerinde çalıştırılan uygulamaların sonuçları Şekil 3.16. ve Şekil 3.17.'da yer alıyor. Şekil 3.15a'da olduğu gibi, α=0.1 durumunda birinci kümenin örneklem noktaları gerekenden daha az yaklaşdığı için ikinci kümenin noktaları da birinci kümeye kaymaktadır. Şekil 3.16.a'da α=0.1 durumu için örneklem noktalar merkezden uzak kaldığından büyük yuvarlak bölünerek doğru bir sonuç gerçekleşmemektedir. Sonuçta α=0.1 seçildiğinde merkeze çok az yaklaştığından doğru kümeler bulunamamaktadır. α katsayısı çok küçük seçildiği durumlariçin CURE algoritması dış noktalara karşı daha hassas olmaktadır. Yuvarlaklar veri setinde α=0.2 seçildiği durumda, kareler veri setinde ise α=0.3 olduğu durumda doğru kümeler bulunur. α=0.9 durumnda Şekil 3.16.d ve Şekil 3.17.d'de elde edilen sonuçlara göre örneklem noktaların kümenin merkez noktasına gerekenden çok yaklaşarak algoritmanın merkez tabanlı kümeleme algoritmaları gibi çalışmasına neden olmaktadır.



Şekil 3.16. Yuvarlaklar: Farklı α Değerleriyle Gerçekleştirilen Sonuçlar



Şekil 3.17. Kareler : Farklı α Değerleriyle Gerçekleştirilen Sonuçlar

4. HESAPLAMA DENEMELERI

Algoritmalar C# dilinde MsSql veritabanı kullanarak yazılmıştır. Algoritmaların avantajları ve dezavantajları uygulama ve çıktılar üzerinde açıkca görülmektedir.

4.1. K-means

K-means algoritması aşağıdaki 5 metottan oluşmaktadır. İlk metot (CreateDataPoints()) noktaları datagridview'dan alıp dğer metotların kullanacağı listeye atar. Ardından CreateClusters() metodu çalışır, girdi olarak alınan küme sayısı kadar rastgele merkez belirler. Bir sonraki adımda AssignDataPointsToCloserCluster() metodu listemizde olan tüm noktaları uzaklık kriterine göre bulduğumuz merkezlere atar. Ardından CalculateCenterOfEachCluster() metudu bulunan kümelerin ağırlık merkezini hesaplar. Bu işlemler bittikten sonra FintTheResult() metodu bir önceki merkezle yeni merkezin aynı olup olmadığını teyit eder, farklı olduğu sürece AssignDataPointsToCloserCluster() ve CalculateCenterOfEachCluster() metotları çalışmaya devam eder. Bulunan son iki adımlaki merkezlerın aynı olması durumunda program çalışmayı durdurur ve sonuçları ekrana yansıtır.

CreateDataPoints():

```
private void CreateDataPoints()
{
      listDataPoint = new List<DataPoint>();
      listCluster = new List<Cluster>();
      listDPLimit = new List<DPLimit>();
            for (int i = 0; i < dtVwKmeans.RowCount-1; i++)
      {
          int xPoint =
               int.Parse(dtVwKmeans.Rows[i].Cells[0].Value.ToSt
               ring());
          int yPoint =
               int.Parse(dtVwKmeans.Rows[i].Cells[1].Value.ToSt
               ring());
          DataPoint dataPoint = new DataPoint(i + 1, xPoint,
               yPoint, null);
          listDataPoint.Add(dataPoint);
      }
}
```

Kodlardan da görüldüğü gibi CreateDataPoints() metodu veri tabanından gelen verileri gridden alıp listeye atar yansıtır.

CreateClusters():

```
private void CreateClusters()
{
	listCluster = new List<Cluster>();
	Random rand = new Random();
	for (int i = 0; i < numberOfClusters; i++)
	{
		int xPoint = rand.Next(pnlDrowedData.Size.Width);
		int yPoint = rand.Next(pnlDrowedData.Size.Height);
		Cluster cluster = new Cluster(i + 1, xPoint, yPoint,
			listColor[i]);
		listCluster.Add(cluster);
	}
}
```

CreateClusters() girdi olarak alına küme sayısı kadar rasgele merkez oluşturur.

FintTheResult():

```
private void FintTheResult()
{
	isFinished = false;
	while (!isFinished)
	{
		AssignDataPointsToCloserCluster();
		CalculateCenterOfEachCluster();
	}
}
```

AssignDataPointsToCloserCluster()

```
private void AssignDataPointsToCloserCluster()
{
	iterationNumber++;
	foreach (DataPoint dataPoint in listDataPoint)
	{
		Cluster nearestCluster = null;
		double distance = 999999999999;
		foreach (Cluster cluster in listCluster)
		{
			double tempDistance = GetDistance(dataPoint, cluster);
			if (tempDistance < distance)
			{
				nearestCluster = cluster;
				distance = tempDistance;
			}
		}
		dataPoint.Cluster = nearestCluster;
	}
	pnlDrowedData.Invalidate();
}
```

Uzaydaki tüm noktaların en yakın merkeze atanmasını sağlamaktadır.

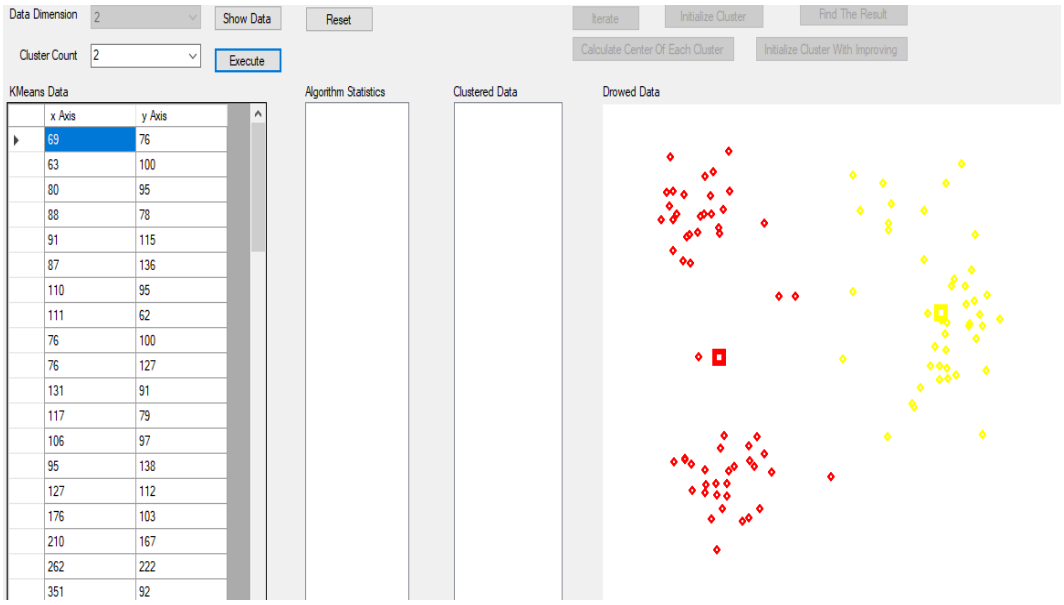
CalculateCenterOfEachCluster():

```
private void CalculateCenterOfEachCluster()
{
      foreach (DataPoint dataPoint in listDataPoint)
      {
            dataPoint.Cluster.XTotal += dataPoint.XPoint;
            dataPoint.Cluster.YTotal += dataPoint.YPoint;
            dataPoint.Cluster.TotalDataPoints++;
      }

      bool isSame = true;
      foreach (Cluster cluster in listCluster)
      {
            if (cluster.TotalDataPoints > 0)
            {
                  cluster.setXPoint(Convert.ToInt16(cluster.XTo
                        tal / cluster.TotalDataPoints));
                  cluster.setYPoint(Convert.ToInt16(cluster.YTo
                        tal / cluster.TotalDataPoints));

                  if (!(cluster.XPoint == cluster.OldXPoint &&
                        cluster.YPoint == cluster.OldYPoint &&
                              cluster.OldTotalDataPoints ==
                              cluster.OldTotalDataPoints))
                  {
                        isSame = false;
                  }

                  cluster.SetToDefaultTotal();
            }
      }

      if (isSame)
      {
            isFinished = true;
            string msg = "Finished" + Environment.NewLine;

            foreach (Cluster cluster in listCluster)
            {
                  msg += "Cluster " + cluster.Number + "(" +
                        cluster.ColorOfPoint.ToString() + ")" +"
                        = " +
                        cluster.OldTotalDataPoints.ToString() +
                        "datapoints" + Environment.NewLine;
            }
            msg += "Iteration Number : " +
                  iterationNumber.ToString();
            MessageBox.Show(msg);
      }
      pnlDrowedData.Invalidate();
}
```

CalculateCenterOfEachCluster() metodu oluşan kümelerin merkezlerini bulur, örneklem noktaları tekrardan ait oldukları kümelere dağıtır, tekrardan küme merkezlerini bulur, yeni merkez eski merkeze eşitlenene kadar bu işlem devam eder. Son olarak bulunan kümeleri farklı renklerde panele yansıtır. Algoritma 2 ve 3 boyutlu noktalar için çalışır.

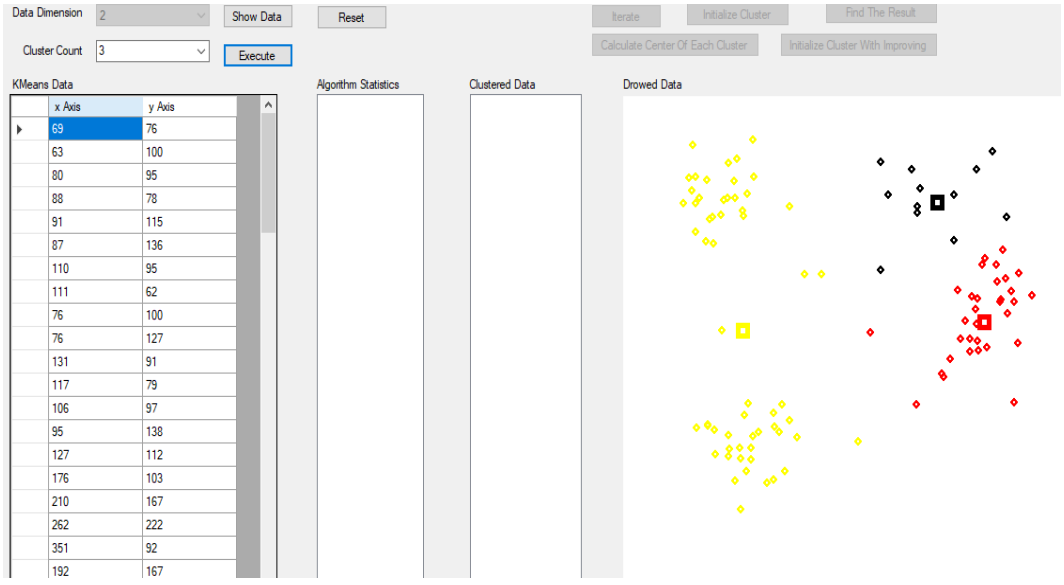
4.1.1. Algoritma çıktıları

Çıktı 1: Algoritma iki boyutlu veri için küme sayısı k=2 olarak verilip çalıştırılmıştır. Sonuç Şekil4.1.



Şekil 4.1. İki boyutlu veride k = 2

Çıktı 2: Algoritma iki boyutlu veri için küme sayısı k=3 olarak verilip çalıştırılmıştır. Şekil4.2.



Şekil 4.2. İki boyutlu veride k = 3

Çıktı 3: Algoritma iki boyutlu veri için küme sayısı k=4 olarak verilerekçalıştırılmıştır. Şekil4.3.

Şekil 4.3. İki boyutlu veride k = 4

Çıktı 4: Algoritma iki boyutlu veri için küme sayısı k=5 olarak verilerek çalıştırılmıştır. Şekil4.4.

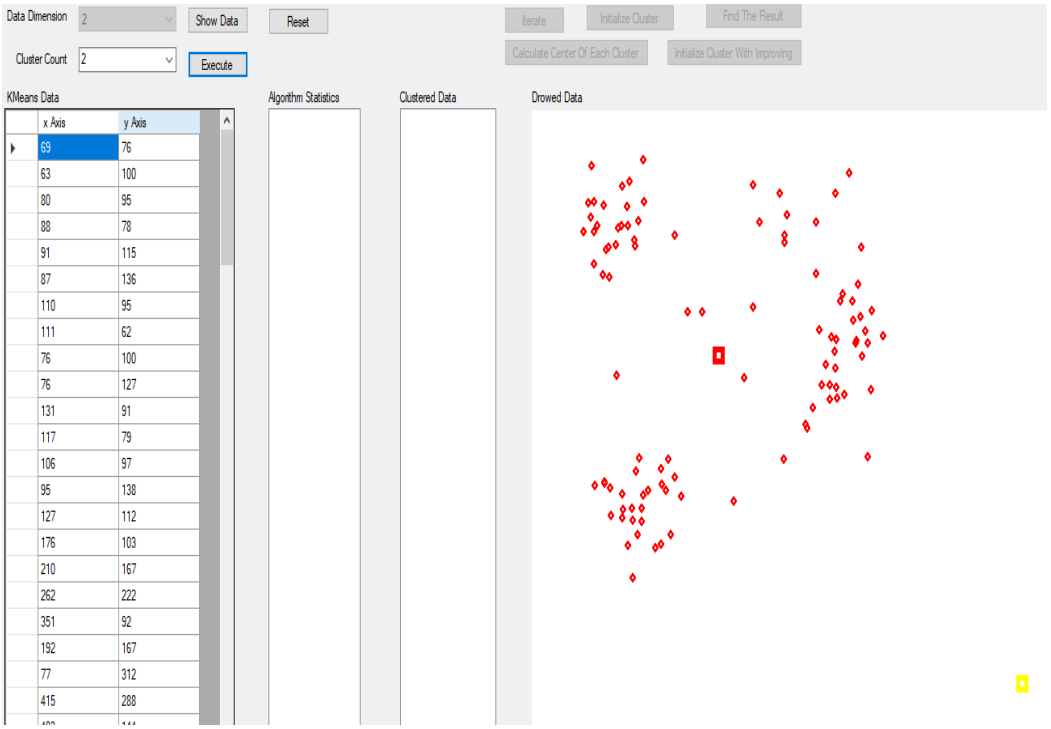
Şekil 4.4. İki boyutlu veride k = 5

Çıktı 5: Algoritma iki boyutlu veri için küme sayısı k=6 olarak verilerek çalıştırılmıştır. Şekil4.5.

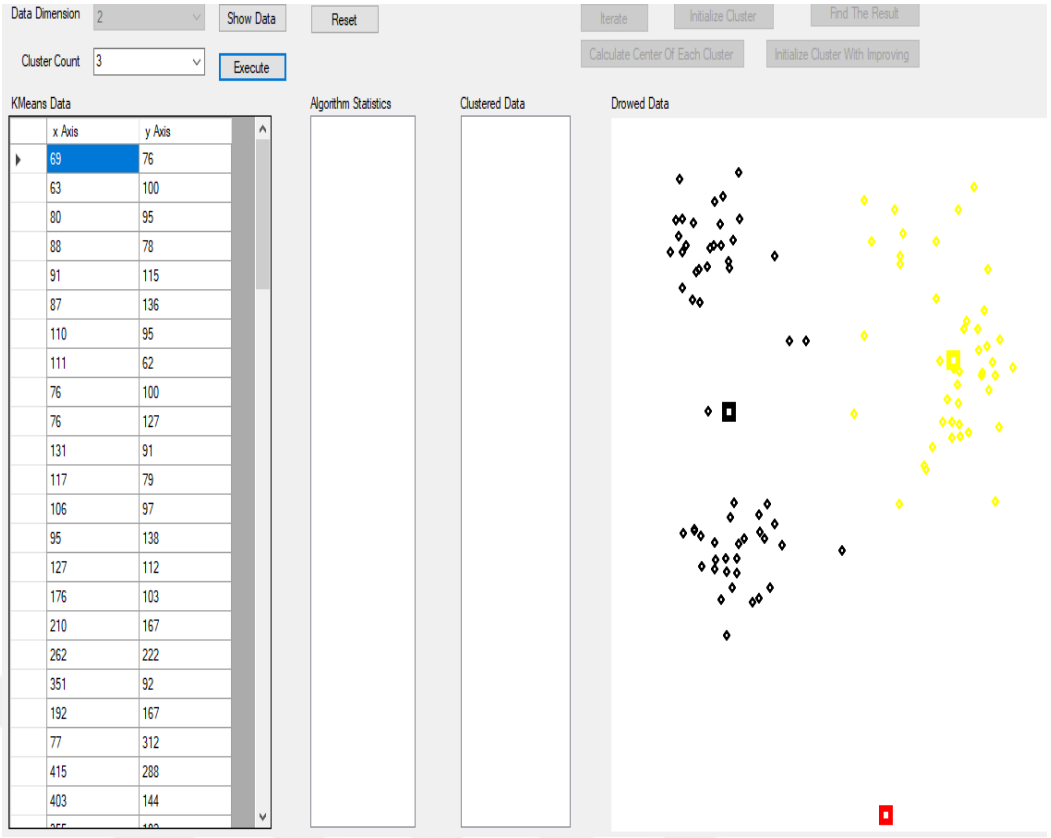
Şekil 4.5. İki boyutlu veride k = 6

4.1.2. Algoritmanın kötü sonuçları

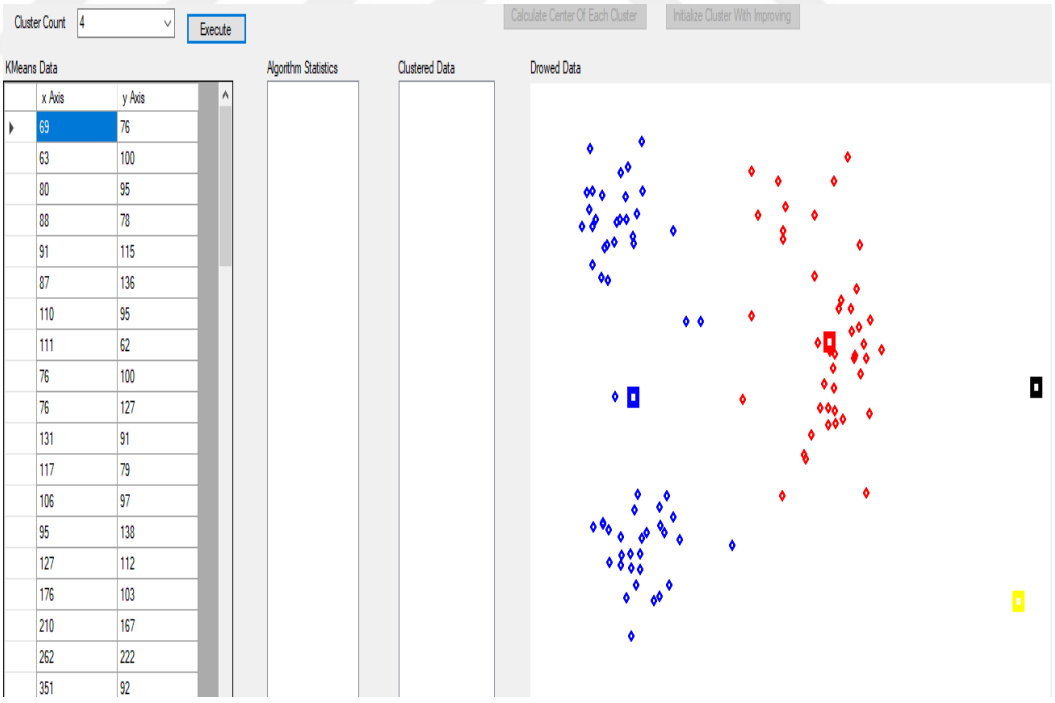
Kötü sonuçlar genellikle örneklem noktaların dağılımında herhangi bir merkezin tek başına bir küme oluşturması veya küme elemanlarının merkeze uzaklıklarının çok olması anlamına gelmektedir. Aşağıda küme merkezlerinin tek başına küme oluşturma durumları gösterilmiştir.



Şekil 4.6. İki boyutlu veride k = 2



Şekil 4.7. İki boyutlu veride k = 3



Şekil 4.8. İki boyutlu veride k = 4

Şekil 4.9. İki boyutlu veride k = 5

4.2. Optics

Optics algoritmasının kaykak kodları aşağıdaki 5 metotdan oluşuyor. getRawData() metodu örneklem kümesini veri tabanından rawDataModel listesi tipinde döndüren metotdur. Ardından getCores() metodu çalışır, core noktaları liste halinde döndürür. Bir sonraki adımda getcoresDistance() uygulama metoduyla core mesafelerini hesaplıyor. Sonraki adımda getRcbDistance() metodu çalışır ve örneklem noktaların ulaşılabilirlik uzaklıklarını döndürüyor. Son adımda drawGraphics() metodu önceki 3 metotdan dönen değerleri kullanarak sonucu grafic şeklinde ekrana yansıtıyor.

getRawData() örneklem noktaları very tabanından çeker.

```
private List<rawDataModel> getRawData()
{
       rawDataModel rawDataInner = new rawDataModel();
       List<rawDataModel> rawListInner = new List<rawDataModel>();
       using (SqlConnection con=new
            SqlConnection(Properties.Settings.Default.conString))
       {
              SqlDataReader dr;
              SqlCommand cmd = new SqlCommand();
              cmd.CommandText= "exec prcBringKmeansData
       @columnsCount";
              cmd.Connection = con;
              cmd.Parameters.Clear();
              cmd.Parameters.AddWithValue("@columnsCount",2);
              con.Open();
              dr = cmd.ExecuteReader();
              while (dr.Read())
              {
                     rawDataInner.x = (int)dr[0];
                     rawDataInner.y = (int)dr[1];
                     rawDataInner.coreX = 0;
                     rawDataInner.coreY = 0;
                     rawDataInner.rcbDistance = 0;
                     rawListInner.Add(rawDataInner);
                     rawDataInner= new rawDataModel();
              }
              con.Close();
              dr.Close();
       }
       return rawListInner;
}
```

getCores() metodu core noktaları bulur.

```
private List<cores> getCores()
{
      List<cores> coresListInner = new List<cores>();
      cores coresInner = new cores();
      int coreCount;
      double coreDistance;

      foreach (var item in rawList)
      {
            coreCount = 0;

            foreach (var itemInner in rawList)
            {
                  if (item.x == itemInner.x && item.y ==
            itemInner.y)
                        continue;

                  coreDistance = Math.Sqrt(Math.Pow((item.x -
                       itemInner.x), 2) + Math.Pow((item.y -
                       itemInner.y), 2));
                  if (coreDistance <= randomEps)
                        coreCount++;

                  if (coreCount >= minPnt - 1)
                  {
                        coresInner.x = item.x;
                        coresInner.y = item.y;
                        coresInner.isCore = true;
                        coresInner.coreDistance = 0;
                        coresListInner.Add(coresInner);
                        coresInner = new cores();
                        break;
                  }
            }
      }
      return coresListInner;
}
```

getcoresDistance() metodu core uzaklıklarını buluyor.

```
private void getcoresDistance()
{
	double coreDistance;
	double[] minPntCountClosest = new double[minPnt - 1];

	foreach (var coreItem in cores)
	{
		for (int i = 0; i < minPntCountClosest.Length; i++)
			minPntCountClosest[i] = 0;
		foreach (var rawItem in rawList)
		{
			if (coreItem.x == rawItem.x && coreItem.y ==
		rawItem.y)
				continue;

			coreDistance = Math.Sqrt(Math.Pow((coreItem.x -
				rawItem.x), 2) + Math.Pow((coreItem.y -
				rawItem.y), 2));
			if (coreDistance > randomEps)
				continue;

#region Get closest points
			if (minPntCountClosest.Contains(0))
			{
				for (int i = 0; i <
			minPntCountClosest.Length; i++)
				{
					if (minPntCountClosest[i] == 0)
					{
						minPntCountClosest[i] =
					coreDistance;
						break;
					}
				}
			}
			else
			{
				Array.Sort(minPntCountClosest);
				for (int i = 0; i <
			minPntCountClosest.Length; i++)
				{
					if (minPntCountClosest[i] >
				coreDistance)
					{
						minPntCountClosest[minPntCount
							Closest.Length - 1] =
							coreDistance;
						break;
					}
				}
			}
#endregion
		}
		Array.Sort(minPntCountClosest);
		coreItem.coreDistance =
			minPntCountClosest[minPntCountClosest.Length - 1];
	}
}
```

getRcbDistance() motodu ulaşılabilir uzaklıkları buluyor.

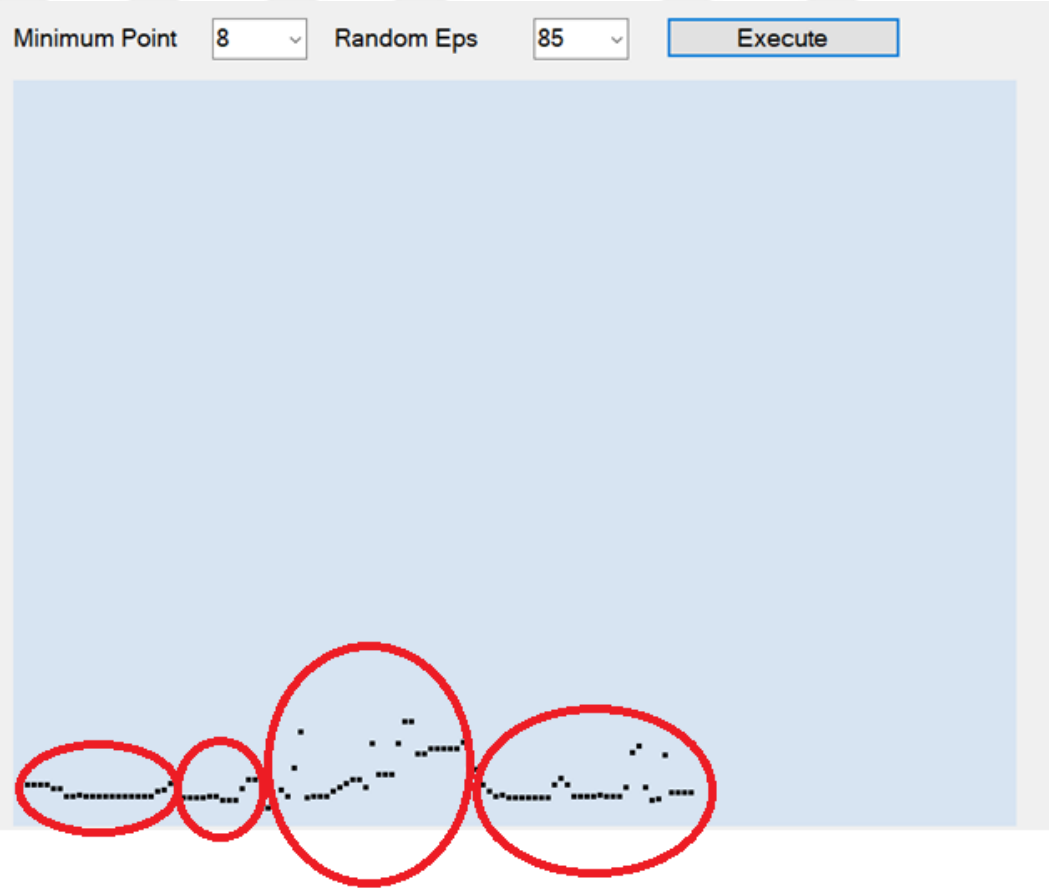
```
private void getRcbDistance()
{
      double rcbDistance;
      foreach (var rawItem in rawList)
      {
            foreach (var coreItem in cores)
            {
                  if (coreItem.x == rawItem.x && coreItem.y
            == rawItem.y)
                        continue;

                  rcbDistance =
                       Math.Sqrt(Math.Pow((coreItem.x -
                       rawItem.x), 2) +
                       Math.Pow((coreItem.y - rawItem.y),
                       2));
                  if (rawItem.rcbDistance==0)
                  {
                        if
                  (rcbDistance<coreItem.coreDistance)
                              rawItem.rcbDistance =
                        coreItem.coreDistance;
                        else
                              rawItem.rcbDistance =
                        rcbDistance;
                        rawItem.coreX = coreItem.x;
                        rawItem.coreY = coreItem.y;
                  }
                  else
                  {
                        if (rawItem.rcbDistance >
                  rcbDistance)
                        {
                              if (rcbDistance <
                        coreItem.coreDistance)
                                    rawItem.rcbDistance =
                                         coreItem.coreDista
                                         nce;
                              else
                                    rawItem.rcbDistance =
                              rcbDistance;
                              rawItem.coreX = coreItem.x;
                              rawItem.coreY = coreItem.y;
                        }
                  }
            }
      }
}
```

drawGraphics() metodu işlenmiş veriyi panele döküyor.

```
private void drawGraphics(List<rawDataModel> rawList)
{
      Graphics graphicsObj;
      rawList.Sort();
      graphicsObj = pnlOpticsGraphics.CreateGraphics();
      Pen myPen = new Pen(Color.Black, 3);
      Rectangle myRectangle;/*= new Rectangle(100, 50, 1,
300);*/
      int i = 10;

      foreach (var item in rawList)
      {
            myRectangle= new Rectangle(i,
                pnlOpticsGraphics.Height-
                (int)item.rcbDistance, 1,1);
            graphicsObj.DrawRectangle(myPen, myRectangle);
            i+=5;
      }
}
```

4.2.1. Çıktılar

Çıktı 1: Algoritma 2 boyutlu veri üzerinde minPoint 8 randomEps 85 br parametreleriyle çalıştırılmıştır.



Şekil 4.9.

Şekil 4.9.'de gördüğümüz gibi Optics algoritması sonucu elimizdeki verileri herhangi bir küme sayısını parametre olarak alan kümeleme algoritmasında çalıştırırsak, küme sayısı 4 olarak çalıştırmamız algoritma ideal sonuç verecektir. Cure ve K-means algoritmalarında bahsettiğimiz gibi kümeleme algoritmalarının ideal sonuç vermesi sadece küme sayısı değil diğer parametrelere de bağlı olduğu için diğer faktörleri de göz önünde bulundurmamız zorunludur.

4.3. Cure

Cure algoritması aşağıdaki metotlardan oluşuyor. Execute butonu alınan sonuçları ekrana yazdıran kodları içerir. Bu buton işlenmiş veriyi GetFinestResult() metodundan alıyor. Diğer metotlar GetFinestResult() metodu tarafından çalıştırılmaktadır. randomCent(), AssignDataPointsToCloserCluster(), getNewCentroid() ve isCentSame() metotları ham veriyi ilkel becerilerle girdi olarak alınan küme sayısı kadar kümelere ayırır. Bu işlem için önemli kriter kümelerin merkeze olan uzaklık olarak açısından bir-birinin elemanlarını içermemesidir. Ilkel kümeleme işlemi bitince GetRepresentitps() metodu işlemmiş veriyi ve getClusters(), getCurePoinst(), getMinDistance() metotları kullanarak kümeleri daha keskin sınırlarla ayırıp, daha optimal bir sonuç sunar.

```
private void btnExecute_Click(object sender, EventArgs e)
        {
            if (dtVwKmeans.RowCount==0 ||
cmbClusterCount.Text=="")
            {
                MessageBox.Show("Data listeleyiniz ve küme
sayısı seçiniz!!");
                return;
            }
            clusterCountBase =
Convert.ToInt32(cmbClusterCount.Text);
            centListBase = randomCentBase();
            cureGenerators cr = new cureGenerators();
            List<centroid> clusteredData =
cr.GetFinestResult();
            Dictionary<int, List<curePoints>> repData=
cr.GetRepresentitps();
            clusteredData =
clusteredData.OrderBy(x=>x.key).ToList();
              Graphics graphicsObj;
            graphicsObj = pnlDrowedData.CreateGraphics();
            string[] colors = new string[9];
            colors[1] = "System.Drawing.Color.Red";
            Pen myPen = new Pen(System.Drawing.Color.Red, 5);
```

```
foreach (var dataPoint in clusteredData)
            {
                switch (dataPoint.key)
                {
                    case 1:
                        myPen = new Pen(System.Drawing.Color.Red, 5);
                        break;
                    case 2:
                        myPen = new Pen(System.Drawing.Color.Green, 5);
                        break;
                    case 3:
                        myPen = new Pen(System.Drawing.Color.Black, 5);
                        break;
                    case 4:
                        myPen = new Pen(System.Drawing.Color.Yellow, 5);
                        break;
                    case 5:
                        myPen = new Pen(System.Drawing.Color.Purple, 5);
                        break;
                    case 6:
                        myPen = new Pen(System.Drawing.Color.DarkBlue, 5);
                        break;
                }

                Rectangle rect = new Rectangle(dataPoint.x, dataPoint.y, 1,
1);
                graphicsObj.DrawEllipse(myPen, rect);
            }

            foreach (var item in repData)
            {
                foreach (var dataPoint in item.Value)
                {
                    switch (item.Key)
                    {
                        case 1:
                            myPen = new Pen(System.Drawing.Color.Red, 10);
                            break;
                        case 2:
                            myPen = new Pen(System.Drawing.Color.Green,10);
                            break;
                        case 3:
                            myPen = new Pen(System.Drawing.Color.Black, 10);
                            break;
                        case 4:
                            myPen = new Pen(System.Drawing.Color.Yellow, 10);
                            break;
                        case 5:
                            myPen = new Pen(System.Drawing.Color.Purple, 10);
                            break;
                        case 6:
                            myPen = new Pen(System.Drawing.Color.DarkBlue,
10);
                            break;
                    }

                    Rectangle rect = new Rectangle(dataPoint.x, dataPoint.y,
1, 1);
                    graphicsObj.DrawEllipse(myPen, rect);
                }
            }
        }
```

findResult() metodu bir iç metot olup kümeleme işlemi bitince geri değer döndürme işlemini yapmaktadır.

```
public List<cureKmeans> findResult()
{
       List<centroid> clusterOld = randomCent();
       List<cureKmeans> dcDataPoint =
            AssignDataPointsToCloserCluster(clusterOld);
       List<centroid> clusterNew = getNewCentroid(dcDataPoint,
            clusterOld);
       bool finished = isCentSame(clusterOld, clusterNew);
       while (!finished)
       {
              clusterOld = new List<centroid>();
              clusterOld = clusterNew;
              dcDataPoint = new List<cureKmeans>();
              dcDataPoint =
       AssignDataPointsToCloserCluster(clusterOld);
              clusterNew = getNewCentroid(dcDataPoint, clusterOld);
              finished = isCentSame(clusterOld, clusterNew);
       }
       return dcDataPoint;
}
```

randomCent() metodu uzaydaki ilk küme listesini bulan metoddur.

```
private List<centroid> randomCent()
{
       List<centroid> centList = new List<centroid>();
       centroid cent;
       Random rd = new Random();
       for (int i = 1; i <= clusterCount; i++)
       {
              cent = new centroid();
              cent.x = rd.Next(600);
              cent.y = rd.Next(500);
              cent.key = i;
              centList.Add(cent);
       }
       return centList;
}
```

AssignDataPointsToCloserCluster() metodu ağırlık merkezi listesini parametre olarak alarak uzay noktalarını yakın oldukları merkezlere dağıtmaktadır.

```
private List<cureKmeans>
     AssignDataPointsToCloserCluster(List<centroid> cluster)
 {
       List<centroid> dataPoint = new List<centroid>();
       List<cureKmeans> dataResultList = new List<cureKmeans>();
       cureKmeans dataResult = new cureKmeans();
       dataPoint = getCureKmeansPoinst();
       foreach (var dtPoint in dataPoint)
       {
              centroid nearestCluster = null;
              dataResult = new cureKmeans();
              double distance = 999999999999;
              foreach (var clsPoint in cluster)
              {
                     double tempDistance = GetDistance(dtPoint,
                          clsPoint);
                     if (tempDistance < distance)
                     {
                            nearestCluster = clsPoint;
                            distance = tempDistance;
                     }
              }
              dataResult.center = nearestCluster;
              dataResult.key = nearestCluster.key;
              dataResult.x = dtPoint.x;
              dataResult.y = dtPoint.y;
              dataResult.z = dtPoint.z;
              dataResultList.Add(dataResult);
       }
       return dataResultList;
 }
```

getNewCentroid() merkezlere atanan noktaların yeni merkezlerini bulur.

```
private List<centroid> getNewCentroid(List<cureKmeans>
    dataPoint, List<centroid> clusterOld)
        {
            List<centroid> cluster = new List<centroid>();
            centroid cent;
            for (int i = 1; i <= clusterCount; i++)
            {
                int minX = 999999, maxX = 0, minY =
                    999999, maxY = 0;
                cent = new centroid();
                foreach (var item in dataPoint)
                {
                    if (item.key == i)
                    {
                        if (minX > item.x)
                            minX = item.x;
                        if (maxX < item.x)
                            maxX = item.x;
                        if (minY > item.y)
                            minY = item.y;
                        if (maxY < item.y)
                            maxY = item.y;
                    }
                }
                if (minX == 999999)
                {
                    foreach (var item in clusterOld)
                    {
                        if (item.key==i)
                        {
                            cent.key = i;
                            cent.x = item.x;
                            cent.y = item.y;
                            cluster.Add(cent);
                            break;
                        }
                    }
                }
                else
                {
                    cent.key = i;
                    cent.x = (int)((minX + maxX) / 2);
                    cent.y = (int)((minY + maxY) / 2);
                    cluster.Add(cent);
                }
        }
        return cluster;
}
```

isCentSame() metodu bulunan her yeni merkezin eski merkezle aynı olup olmadığını kontrol ediyor, aynı olduğu durumda kümelere atanan data CURE işlemleri için aşağıdaki metotlara gönderiliyor.

```
private bool isCentSame(List<centroid> clusterOld, List<centroid> clusterNew)
{
      bool isSame = true;
      foreach (var itemOld in clusterOld)
      {
           if (!isSame)
               break;
           foreach (var itemNew in clusterNew)
           {
                 if (itemOld.key == itemNew.key && (itemOld.x != itemNew.x ||
                    itemOld.y != itemNew.y))
               {
                   isSame = false;
               }
               if (!isSame)
                        break;
           }
       }
       return isSame;
}
```

GetRepresentitps() parameter olarak gelen kümelerin temsilci noktalarını döndürür.

```
public Dictionary<int, List<curePoints>> GetRepresentitps()
{
       Dictionary<int, List<curePoints>> clusteredData = getClusters();
       Dictionary<int, List<curePoints>> newRepPoints = new
            Dictionary<int, List<curePoints>>();
       Dictionary<int, List<curePoints>> newClosestRepPoints = new
            Dictionary<int, List<curePoints>>();
       curePoints newRepPoint = new curePoints();
       foreach (var item in clusteredData)
       {
              if (item.Value.Count < pointCount)
              {
                     continue;
              }
              else
              {
                     newRepPoints.Add(item.Key, preGetReps(item.Value));
              }
       }
       foreach (var item in newRepPoints)
       {
              newClosestRepPoints.Add(item.Key, getCloser(item.Value));
       }
       return newRepPoints;
}
```

getClusters() metodu ilkel kümeleme metodundan gelen kümeleri döndürür.

```
public Dictionary<int, List<curePoints>> getClusters()
{
            Dictionary<int, List<curePoints>> dataFromKK = new Dictionary<int,
List<curePoints>>();
            List<cureKmeans> dcDataPoint = findResult();
            curePoints dtPoint;
            List<curePoints> dtPointList = new List<curePoints>();
            dcDataPoint = dcDataPoint.OrderBy(x => x.key).ToList();
            int lastNo = dcDataPoint.Count;
            int i = 0;
            int j = 1;
            foreach (var item in dcDataPoint)
            {
                j++;
                dtPoint = new curePoints();
                if ((i != item.key && i != 0) || j==lastNo)
                {
                    dataFromKK.Add(i, dtPointList);
                    dtPointList = new List<curePoints>();
                }
                i = item.key;
                dtPoint.x = item.x;
                dtPoint.y = item.y;
                dtPoint.z = item.z;
                dtPointList.Add(dtPoint);
            }
            return dataFromKK;
}
```

getCurePoinst() ham veriyi very tabanından alır.

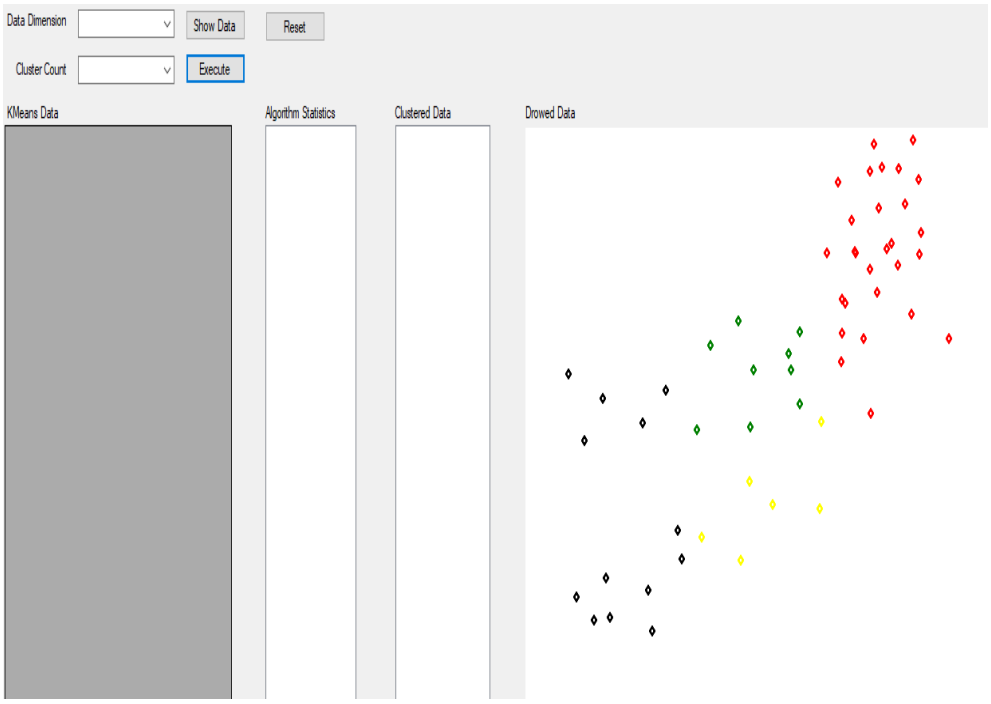
```
private List<centroid> getCurePoinst()
        {
            List<centroid> crListIn = new List<centroid>();
            centroid crIn;
            using (SqlConnection con = new
SqlConnection(Properties.Settings.Default.conString))
            {
                SqlDataReader dr;
                SqlCommand cmd = new SqlCommand();
                cmd.CommandText = "exec prcBringKmeansData @columnsCount";
                cmd.Connection = con;
                cmd.Parameters.Clear();
                cmd.Parameters.AddWithValue("@columnsCount", columnCount);
                con.Open();
                dr = cmd.ExecuteReader();
                int i = 1;
                while (dr.Read())
                {
                    crIn = new centroid();
                    crIn.x = (int)dr[0];
                    crIn.y = (int)dr[1];
                    if (columnCount == 3)
                        crIn.z = (int)dr[2];
                    crListIn.Add(crIn);
                    i++;
                }
                con.Close();
                dr.Close();
            }
              return crListIn;
}
```

getMinDistance() metodu ham veriyi en yakın temlisci noktanın bulunduğu kümeye atar.

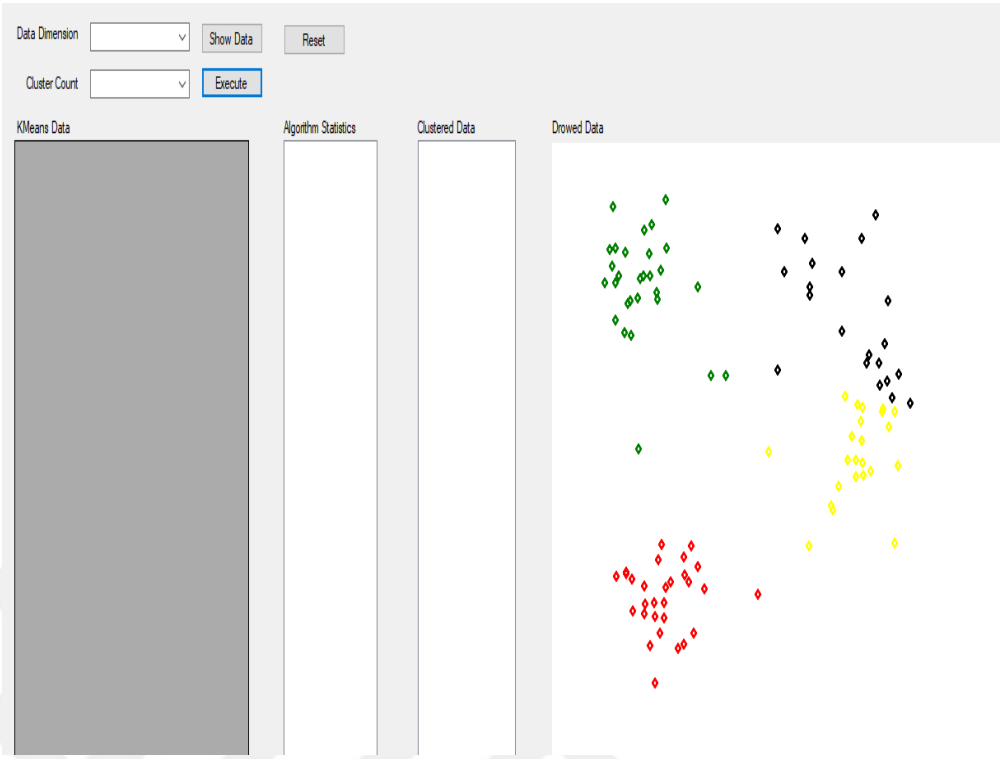
```
public List<centroid> getMinDistance(List<centroid> clusteredData,
Dictionary<int, List<curePoints>> repData)
        {
            List<centroid> resultList = new List<centroid>();
            centroid result;

            foreach (var clData in clusteredData)
            {
                int minKey = 0;
                double minDist = 0;
                foreach (var rpData in repData)
                {
                    if (minKey == 0 || minDist > minDistance(clData,
rpData.Value))
                    {
                        minDist = minDistance(clData, rpData.Value);
                        minKey = rpData.Key;
                    }
                }
                result = new centroid();
                result.x = clData.x;
                result.y = clData.y;
                result.z = clData.z;
```

4.3.1. Çıktılar

Aşağıda 2 ve 3 boyutlu veri üzerinde K küme sayısı 4, P örneklen sayısı 4 ve a daraltma katsayısı 0.5 parametreleriyle CURE algoritmasının çıktıları yer almaktadır.



Şekil 4.10.


Data Dimension
Show Data
Reset
Cluster Count
Execute
KMeans Data
Algorithm Statistics
Clustered Data
Drowed Data

Şekil 4.11.

5. SONUÇ

Çıktılardan görüldüğü gibi K-means algoritması k küme sayısına ve ilk başta rastgele seçilen merkezlere bağlı olarak iyi veya kötü sonuçlar verebilir. Şekil 4.5.'de görüldüğü gibi bazen küme sayısının fazla olması tek küme olması gereken örneklem verisinin bölünmesine neden olur. Bu nedenle hangi küme sayısının ideal sonuç vereceğini deneme yoluyla bulmamız gerekmektedir. Bunun yanı sıra ideal sonuç veren küme sayısını bulduktan sonra bile algoritmayı bu parametreyle tekrar tekrar çalıştırıp farklı sonuçlar alabiliriz. Bu nedenle K-means algoritmasında ideal sonuca ulaşmak için ideal sonuç verecek küme sayısını bulduktan sonra bile tek deneme yetmez, tekrar tekrar deneme daha iyi sonuç verebilir.

Optics algoritması tek başına kümeleme yapan bir algoritma olmayıp bize kümeleme algoritmalarına göndereceğimiz parametreleri vermektedir. Parametre olarak random eps ve minPoint değerleri alır. Bu değerler algoritmanın sonucunu değil sadece çalışma süresini etkilemektedir. Eps ve minPoint parametrelerini çekirdek noktaları bulmak için kullanır. Eps değerinin büyük ve minPoint değerini küçük olması algoritmanı daha çok noktayı çekirdek olarak almasına neden olur bu da algoritmanın çalışma süresini uzatır.

Cure algoritması girdi olarak küme sayısı, örneklem sayısı ve daraltma katsayısı parametreleriyle çalışmaktadır. Verdiği sonuçlar bu 3 girdiye büyük ölçüde bağlıdır. Cure algoritmasının en büyük dezanvantajı aldığı datayı kümelenmiş istemesidir. Dolayısıyla uzayı 2 defa kümelemiş olur. Ilk algoritma ilkel taktiklerle veriyi kümeler, ardından Cure algoritması iyileştirme yapar. K-means algoritmasına benzer olarak bu algoritmanın da girdilerinin ideal değerleri deneme yoluyla bulunur. Yazılan kodlar 2 ve 3 boyutlu veriler için çalışmaktadır.

KAYNAKLAR DİZİNİ

[1] Canbek, G., & Sağıroğlu, Ş. (2006). Bilgi, bilgi güvenliği ve süreçleri üzerine bir inceleme. Politeknik Dergisi, 9(3).

[2] Lugmayr, A., Lugmayr, A., Stockleben, B., Stockleben, B., Scheib, C., Scheib, C., ... & Mailaparampil, M. A. (2017). Cognitive big data: survey and review on big data research and its implications. What is really "new" in big data?. Journal of Knowledge Management, 21(1), 197-212.

[3] Venkatram, K., & Geetha, M. A. (2017). Review on Big Data & Analytics–Concepts, Philosophy, Process and Applications. Cybernetics and Information Technologies, 17(2), 3-27.

[4] Sinanç, D., "Cep telefonu kullanıcı davranışı modelleme", Yüksek Lisans Tezi, Gazi Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı, 2014.

[5] Mulcahy M., Big Data Statistics & Facts for 2017, Waterford Technologies, https://www.waterfordtechnologies.com/big-data-interesting-facts/

[6] Mohammed, A. F., Humbe, V. T., & Chowhan, S. S. (2016, February). A review of big data environment and its related technologies. In Information Communication and Embedded Systems (ICICES), 2016 International Conference on (pp. 1-5). IEEE.

[7] Türkiye Bilişim Derneği, Kamu Bilişim Merkezleri Yöneticileri Birliği Kamu Bilişim Platformu Büyük Veri Uygulamaları Çalışma Grubu Raporu, Mayıs 2016

[8] Koyuncu, E. (2016). Kalkınma İçin Büyük Veri, Türkiye Ekonomi Politikaları Araştırma Vakfı (TEPAV), Değerlendirme Notu, 1-7

[9] Yıldız, U. A., Topal, S., "Büyük Veri Kahramanı Veri Bilimci", Bilim ve Teknik, 76-79, 2015. [10] IT Data Analyst Job Description – All You need to Know, http://itcareercentral.com/it-data-analyst-job-description-all-you-need-to-know/

KAYNAKLAR DİZİNİ (DEVAMI)

[11] Acharjya, D.P. and P. Kauser Ahmed, A Survey on Big Data Analytics: Challenges, Open Research Issues and Tools. Article in International Journal of Advanced Computer Science and Applications, February 2016.

[12] Groupe Speciale Mobile Association. (2014). The Mobile Economy; GSMA Intelligence, London.

[13] Terzi, D.S., Demirezen, U., Sagiroglu, S., "Evaluations of Big Data Processing", Services Transactions on Big Data 3 (1), 44-54 (2016)

[14] Terzi, DS., Terzi, R., Sagiroglu, S., "A Survey on Security and Privacy Issues in Big Data",The 10th International Conference for Internet Technology and Secured Transactions (ICITST-2015) , London, UK, 202-207, (2015).

[15] Rajan S., Ginkel W., Sundaresan N. (2013). Expanded Top Ten Big Data Security and Privacy Challenges. Cloud Security Alliance Big Data Working Group

[16] Open Data Handbook, "What is Open Data?", http://opendatahandbook.org/ guide/en/what-is-open-data/

[17] Han, J., Pei, J., & Kamber, M. (2011). Data mining: concepts and techniques. Elsevier.

[18] Martin Ester, Hans-Peter Kriegel, Jiirg Sander, Xiaowei Xu - A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. Institute for Computer Science, University of Munich Oettingenstr. 67, D-80538 Miinchen, Germany

[19] BİLGİN, T. T., & ÇAMURCU, Y. (2005). DBSCAN, OPTICS ve K-Means Kümeleme Algoritmalarının Uygulamalı Karşılaştırılması. *Politeknik Dergisi*, *8*(2).

[20] Eden W.M. MA ve Tommy W.S. CHOW, "A New Shifting Grid Clustering Algorithm", Pattern Recognition, Vol:37, Issue:3, 2004, s. 504.

KAYNAKLAR DİZİNİ (DEVAMI)

[21][Ankerst, M., Breunig, M. M., Kriegel, H. P., & Sander, J. (1999, June). OPTICS: ordering points to identify the clustering structure. In ACM Sigmod record (Vol. 28, No. 2, pp. 49-60). ACM.

[22] Wang, X., & Wang, X. (2015, September). An OPTICS clustering-based anomalous data filtering algorithm for condition monitoring of power equipment. In *International Workshop on Data Analytics for Renewable Energy Integration*(pp. 123-134). Springer, Cham.

[23] Aljumaily, H., Laefer, D. F., & Cuadra, D. (2017). Urban Point Cloud Mining Based on Density Clustering and MapReduce. *Journal of Computing in Civil Engineering*, *31*(5), 04017021.

[24] Ester, M., Kriegel, H. P., Sander, J., & Xu, X. (1996, August). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd* (Vol. 96, No. 34, pp. 226-231).

[25] Meral DEMİRALAY, A. Yılmaz ÇAMURCU, CURE, AGNES VE K-MEANS ALGORİTMALARINDAKİ KÜMELEME YETENEKLERİNİN KARŞILAŞTIRILMASI, İstanbul Ticaret Üniversitesi Fen Bilimleri Dergisi Yıl: 4 Sayı: 8 Güz 2005/2 s.1-18

TEŞEKKÜR

Bu çalışmanın gerçekleştirilmesinde, üç yıl boyunca değerli bilgilerini bizlerle paylaşan, kullandığı her kelimenin hayatıma kattığı önemini asla unutmayacağım saygıdeğer danışman hocam; Dr. Öğr. Üyesi Arif GÜRSOY'a, çalışmam boyunca benden bir an olsun yardımlarını esirgemeyen sayın hocam Prof. Dr. Urfat NURİYEV'e ve çalışma süresince tüm zorlukları benimle göğüsleyen ve hayatımın her evresinde bana destek olan değerli aileme sonsuz teşekkürlerimi sunarım.

ÖZGEÇMİŞ

01.06.1991 tarihinde Salyanda (Azerbaycan) doğmuştur. İlk okul, orta okul ve lise eğitimini aynı ilde tamamladıktan sonra Azerbaycan Devlet Pedagoji Üniversitesi (Bakü), Matematik ve Enformatik bölümünde eğitim almaya başlamıştır. 2012 yılında lisans eğitimini bitirdikten sonra Aksu ilinde bulunan Abashanlı Devlet Lisesine Matematik öğretmeni olarak atanmıştır. Bir yıl çalıştıktan sonra YÖS sınavına girerek, Ankara Üniversitesi Amerikan Kültürü ve Edebiyatı bölümünü kazanmıştır.

Ankara Üniversitesi Amerikan Kültürü ve Edebiyatı bölümünde bir yıl yoğun İngilizce hazırlıktan sonra bölüme devam etmeyip IELTS sınavı için Azerbaycana dönmüştür. Sınavdan yüksek lisans için geçerlilik puanını alıp 2015 yılında Türkiye İzmirde Ege Üniversitesine Yüksek Lisans başvurusunda bulunmuştur. Aynı yılda Ege Üniversitesinde Yüksek Lisans hakkı kazanmıştır.

Ege Üniversitesinde Yüksek Lisans eğitimiyle beraber İzmirde bulunan farklı yazılım firmalarında Yazılım Uzmanı olarak çalışmıştır.

Hali hazırda Smg Yazılım ve Danışmanlık Ltd. Şti.'de yazılım uzmanı olarak çalışmaktadır.