BÜYÜK VERİNİN MAKİNE ÖĞRENMESİ YÖNTEMLERİ İLE APACHE SPARK TEKNOLOJİSİ KULLANILARAK SINIFLANDIRILMASI

2017
YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ

Yadigar ERDEM

BÜYÜK VERİNİN MAKİNE ÖĞRENMESİ YÖNTEMLERİ İLE APACHE SPARK TEKNOLOJİSİ KULLANILARAK SINIFLANDIRILMASI

Yadigar ERDEM

Karabük Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalında
Yüksek Lisans Tezi
Olarak Hazırlanmıştır

KARABÜK
Temmuz 2017

Yadigar ERDEM tarafından hazırlanan "BÜYÜK VERİNİN MAKİNE ÖĞRENMESİ YÖNTEMLERİ İLE APACHE SPARK TEKNOLOJİSİ KULLANILARAK SINIFLANDIRILMASI" başlıklı bu tezin Yüksek Lisans Tezi olarak uygun olduğunu onaylarım.

Yrd. Doç. Dr. Caner ÖZCAN .....
Tez Danışmanı, Bilgisayar Mühendisliği Anabilim Dalı

Bu çalışma, jürimiz tarafından oy birliği ile Bilgisayar Mühendisliği Anabilim Dalında Yüksek Lisans tezi olarak kabul edilmiştir. 28/07/2017

Ünvanı, Adı SOYADI (Kurumu)	İmzası
Başkan : Doç. Dr. Ergin YILMAZ (BEÜ)	.....
Üye : Yrd. Doç. Dr. Caner ÖZCAN (KBÜ)	.....
Üye : Yrd. Doç. Dr. Ümit ATİLA (KBÜ)	.....

...../...../2017

KBÜ Fen Bilimleri Enstitüsü Yönetim Kurulu, bu tez ile, Yüksek Lisans derecesini onamıştır.

Prof. Dr. Filiz ERSÖZ .....
Fen Bilimleri Enstitüsü Müdür V.

ii

"Bu tezdeki tüm bilgilerin akademik kurallara ve etik ilkelere uygun olarak elde edildiğini ve sunulduğunu; ayrıca bu kuralların ve ilkelerin gerektirdiği şekilde, bu çalışmadan kaynaklanmayan bütün atıfları yaptığımı beyan ederim."

Yadigar ERDEM

ÖZET

Yüksek Lisans Tezi

BÜYÜK VERİNİN MAKİNE ÖĞRENMESİ YÖNTEMLERİ İLE APACHE SPARK TEKNOLOJİSİ KULLANILARAK SINIFLANDIRILMASI

Yadigar ERDEM

Karabük Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı:
Yrd. Doç. Dr. Caner ÖZCAN

Temmuz 2017, 60 sayfa

Bu çalışmada, teknolojinin ve internetin hızla gelişmekte olduğu bilgi çağında verilerin üretimi, depolanması, analiz edilmesi ve analiz sonuçlarının büyük bir değere sahip olduğundan dolayı büyük veri üzerinde çalışılmıştır. Büyük veri üzerinde sınıflandırma ve kümeleme işlemleri zaman alıcı olabilmektedir. Bu çalışmada, büyük verinin işlenmesi ve analiz edilmesi için geliştirilen Apache Spark teknolojisi kullanılarak farklı büyük veriler üzerinde sınıflandırma, kümeleme ve aykırı değer algılama işlemlerinin yapılması amaçlanmıştır. Bu amaçla, makine öğrenmesi algoritmalarını içeren Apache Spark'ın MLlib kütüphanesinden faydalanılmıştır. Apache Spark teknolojisini kullanarak hataya dayanıklı, güvenilir, tutarlı ve hızlı sınıflandırma ve kümeleme işlemi gerçekleştirmesi amaçlanmaktadır. Bu çalışmada kullanılan MLlib kütüphanesinde yer alan Naïve Bayes, K-means ve Gaussian Mixture yöntemleri ile büyük verilerin başarılı bir şekilde analiz edilmesi

sağlanmış algoritmaların çalışma süreleri farklı veri boyutları kullanılarak tespit edilmiştir. K-means kümeleme algoritmasının uygulaması Spark Standalone modda, 1 master ile 1 master 3 worker şeklinde çalıştırılıp çalışma süreleri tespit edilmiştir.

Anahtar Sözcükler : Makine öğrenmesi, sınıflandırma, büyük veri, Apache Spark.
Bilim Kodu : 924.1.014

ABSTRACT

M. Sc. Thesis

CLASSIFICATION OF BIG DATA WITH MACHINE LEARNING METHODS USING APACHE SPARK TECHNOLOGY

Yadigar ERDEM

Karabük University
Graduate School of Natural and Applied Sciences
Department of Computer Engineering

Thesis Advisor:
Assist. Prof. Dr. Caner ÖZCAN
July 2017, 60 pages

In this study, big data have been studied because the production, storage, analysis and analysis results of the data have a great value in the information age that technology and internet are developing rapidly. Classification and clustering operations on big data is time consuming. In this work, classification, clustering and outlier detection are aimed on different big data sets using Apache Spark technology which is developed for processing and analyzing big data. For this purpose, Apache Spark MLlib library, which contains machine learning algorithms, is used. It is intended to perform fault tolerant, reliable, consistent, and rapid classification and clustering using Apache Spark technology. Naïve Bayes, K-means and Gaussian Mixture methods in the MLlib library are used to successfully analyze big data sets. The working times of the algorithms are determined using different data set sizes.

The application of the K-means clustering algorithm is executed as 1 master and 1 master 3 worker in Spark Standalone mode and the working times are determined.

Key Word : Machine learning, classification, big data, Apache Spark.
Science Code : 924.1.014

TEŞEKKÜR

Bu tez çalışmasının planlanmasında, araştırılmasında, yürütülmesinde ve oluşumunda ilgi ve desteğini esirgemeyen, engin bilgi ve tecrübelerinden yararlandığım, yönlendirme ve bilgilendirmeleriyle çalışmamı bilimsel temeller ışığında şekillendiren sayın hocam Yrd. Doç. Dr. Caner ÖZCAN'a sonsuz teşekkürlerimi sunarım.

Aynı zamanda bu tez çalışmamızı "KBÜ-BAP-17-YL-072" proje numarası ile desteklemeye layık gören Karabük Üniversitesi Bilimsel Araştirma Projeleri Birimi'ne teşekkürlerimi sunarım.

Sevgili eşime ve manevi hiçbir yardımı esirgemeden yanımda oldukları için Sultan UYSAL ve Nilay DEMİR'e tüm kalbimle teşekkür ederim.

İÇİNDEKİLER

Sayfa

Sayfa

ŞEKİLLER DİZİNİ

Sayfa

ÇİZELGELER DİZİNİ

Sayfa

SİMGELER VE KISALTMALAR DİZİNİ

KISALTMALAR

GPS : Global Positioning System (Küresel Yer Belirleme Sistemi)
EC2 : Elastic Compute Cloud (Elastik Hesaplama Bulutu)
SVM : Support Vector Machine (Destek Vektör Makinesi)
IOT : Internet of Things (Nesnelerin İnterneti)
DBSCAN : Density-Based Spatial Clustering of Applications with Noise (Yoğunluk Tabanlı Gürültülü Konumsal Veri Uygulama Kümesi)
DSL : Digital Subscriber Line (Sayısal Abone Hattı)
RDF : Resource Description Framework (Kaynak Tanımlama Çerçevesi)
HPC : High Performance Computing (Yüksek Performanslı Hesaplama)
RDMA : Remote Direct Memory Access (Uzaktan Doğrudan Bellek Erişim)
SEDA : Staged Event-Driven Architecture (Aşamalı Olay Kontrollü Mimari)
HDFS : Hadoop Distributed File System (Hadoop Dağıtılmış Dosya Sistemi)
RDD : Esnek Dağıtılmış Veri Seti (Resilient Distributed Dataset)
GPU : Graphics Processing Unit (Grafik İşleme Ünitesi)
CUDA : Compute Unified Device Architecture (Birleştirilmiş Hesaplama Cihaz Mimarisi-Birleşik Hesaplama Aygıt Mimarisi)
MLlib : Machinelearning Library (Makine Öğrenmesi Kütüphanesi)
BT : Bilişim Teknolojileri
MPP : Multi Parallel Programming (Çoklu Paralel İşleme)
DaaS : Hizmet Olarak Masaüstü Ambarı Hizmeti
PADB : ParAccel Analitik Platformu
SQL : Structured Query Language (Yapılandırılmış Sorgu Dili)
HQL : Hibernate Query Language (Hibernate Sorgulama Dili)
GMM : Gauss Mixtures Modeli
MML : Minimum Message Length (Minimum Mesaj Uzunluğu)
EM : Expectation Maximization (Beklenti Maksimizasyonu Algoritması)

BÖLÜM 1

GİRİŞ

Teknolojinin ve internetin gelişmesiyle birlikte, bilginin gücü ön plana çıkmış ve bilgi fazlalıklarının oluşmasına neden olmuştur. Sürekli bir hızla artan bu bilgi topluluğuna bilgi çöplüğü adı verilmeye başlanmıştır. Yazılım şirketleri, bu çöplükten anlamlı verilerin elde edilebileceğini düşünüyorlardı ve bunu yaparak, büyük veri denilen yapıyı keşfettiler.

Günümüzde bilgi toplumunu oluşturan unsurlar hayatımızın her alanında görülmektedir. Bilgisayarlar hayatımızda büyük önem taşıdıklarından, bilgi topluluklarından anlamlı ve özel nitelikler aranmaya başlanmıştır. Sadece bilgi miktarı artmakla kalmamakta, aynı zamanda bilgiye erişim hızı da artmaktadır. Büyük veriler, sosyal medya paylaşımı, ağ blogları, fotoğraflar, videolar, log dosyaları, uzay verileri, sensör verileri, GPS (Global Positioning System) verileri gibi farklı kaynaklardan elde edilen verilerin anlamlı ve uygulanabilir bir biçimde dönüştürülmüş halidir.

Büyük veriler üzerinde makine öğrenme yöntemleri ile sınıflandırma ve kümeleme işlemleri oldukça önemlidir. Sınıflandırma verileri uygun kriterlere göre kategorilere dahil eder. Sınıflar ayrık olup birbirlerine uzak ya da yakın olma durumu söz konusu değildir. Kümeleme süreci ise, veriyi belirli bir gruba ayırarak çok benzer verilerin bir grubun altına yerleştirilmesini sağlar. Veri kümeleri bölünürse, hileli verileri bulmak için aykırı değer tespiti yapılabilir.

Bu çalışmada, Naive Bayes sınıflandırma yöntemi, K-means kümeleme yöntemi ve Gaussian Mixture yöntemi ile büyük veri üzerinde Apache Spark teknolojisi kullanılarak veri sınıflandırma, veri kümeleme ve aykırı değer algılama işlemlerinin daha hızlı hale getirilmesi amaçlanmaktadır. Büyük veriler üzerinde sınıflandırma ve

kümeleme yapmak zaman alıcı işlemlerdir. Bu nedenle, bu çalışmada Apache Spark hızlı küme bilgisayar mimarisi kullanılmıştır. Bu teknolojiyi kullanarak hataya dayanıklı, güvenilir, tutarlı ve hızlı kümeleme işlemi gerçekleştirmek amaçlanmaktadır. Spark bileşenlerinin makine öğrenmesi kütüphanesi (MLlib) nispeten küçük kod boyutuna ve kullanım kolaylığına sahiptir. Amacı, makine öğrenimini ölçeklenebilir ve kullanışlı kılmaktır. Bu çalışmada kullanılan MLlib kütüphanesinde yer alan Naive Bayes, K-means ve Gaussian Mixture yöntemleri, büyük verilerin başarılı bir şekilde analiz edilmesini sağlamaktadır. Ayrıca farklı boyutlara sahip büyük veri toplulukları üzerinde K-Means ve Gaussian Mixture algoritmalarının zaman kıyaslamaları yapılmıştır.

Bu çalışmanın amacı, büyük veri üzerinde sınıflandırma ve kümeleme algoritmalarının uygulanması ve çalışma zamanı kriteri ile incelenmesidir.

Hazırlanan bu çalışma, genel itibarıyla literatür taraması ile birlikte 6 bölümden oluşmaktadır. Bunlardan birinci bölüm Giriş olup bu kısımda çalışmanın kısa özeti verilmiştir. İkinci bölümde, literatür bilgisi verilmiştir. Üçüncü bölümde büyük veri hakkında genel bilgilere yer verilmiştir. Dördüncü bölümde, çalışmada kullanılan Apache Spark teknolojisi tanıtılmıştır. Beşinci bölümde, kullanılan makine öğrenmesi algoritmalarının genel yapısı anlatılıp uygulamalara ve analizlerine yer verilmiştir.

Altıncı bölümde yapılan çalışmalar sonucu elde edilen bulgular, çalışmanın amacına uygun bir biçimde yorumlanarak sonuçlandırılmıştır.

BÖLÜM 2

LİTERATÜR ÇALIŞMASI

Büyük veriyi oluşturan kaynaklar arasında bloglar, sosyal medya, fotoğraflar, e-mailler, sensör verileri, videolar gibi çeşitli verilerin yer alması şirketler arasında rekabet ortamı oluşturmaktadır. Şirketler veri madenciliği gibi analiz yapılarını kullanarak bu veri kümelerinden yarar sağlayarak kendilerini geliştirmektedirler. Selçuk (2015), yaptığı Büyük veri alanında kullanılabilecek bir uygulama geliştirme stratejisi önerip ve gerçek veriler kullanılarak duygu analizi yapmıştır[1].

Teknolojinin ilerlemesiyle birlikte üretilen, depolanan ve işlenen veri miktarında sürekli bir artış görülmektedir. Çeşitli makine ve cihazlarla sürekli farklı formatlarda veri üretilerek anlık olarak iletilip işlenmeye başlanmıştır. Yapılan çalışmada, Büyük Veri, mevcut analiz ve sosyal bilimlerdeki araştırma yöntem ve teknikleriyle işlenip anlamlı bir yapıya dönüştürülmüştür. Bayrakcı'nın (2015), Büyük Veri alanında yer alan analiz teknik ve yöntemlerinin 2012 yılından sonra akademik alanda yapılan çalışmalarda sıkça yer aldığı tespit edilmiştir [2].

Büyük veri günümüzde yeni bir devrin başlangıcı olarak sayılmaktadır. Kütüphaneler ile ilgili yapılan çalışmada büyük veri iki temel başlık altında incelenmiştir. İlki kütüphanelerde büyük verinin yer alabilmesi diğeri ise kütüphane işlem ve hizmetlerinde büyük veriden yararlanıp bu işlem ve hizmetleri geliştirmek için kullanılabilirliğidir. Bu farkındalığın var olup olmadığı bir anket uygulamasıyla ölçülmeye çalışılmıştır. Çalışmanın sonucuna göre eğitim sunumuna katılan kütüphanecilerin, sunumdan sonra büyük veri konusunda temel bilgi düzeylerinde istatistiksel olarak anlamlı bir artış olduğu gözlenmiştir. Doğan'ın (2015), yaptığı çalışmada Büyük Verinin kütüphanecilik mesleği içinde değerlendirilmesi gerektiği ve büyük veri ile ilgili hazırlanacak eğitim programlarının kütüphanelere olumlu katkılar sağlayacağı görülmüştür [3].

Verilerin farklı formatlarda ve hızlı üretilmeleri, devasa büyüklükte verilerin oluşmasına neden olmaktadır. Klasik veritabanları büyük veri söz konusu olduğunda yetersiz kalmaktadır. Büyük verilerin analiziyle kurum ve kuruluşların, toplumların, devletlerin ve araştırmacılar için önemli değişimler söz konusu olması muhtemeldir. Doğan'ın (2014), yaptığı çalışmada büyük veri yapısının birçok alanda etkili bir şekilde kullanılmasıyla birlikte kurum ve kuruluşların üzerindeki etkileri belirlenmiştir [4].

Son yıllarda büyük veri kavramının dikkat çekmesi ile bu verilerin analiz edilme ihtiyacı artmaktadır. Toplu verilerin analiz edilmesi kolay bir işlem değildir. Özellikle twitter gibi sürekli akan veri kaynaklarının büyük miktardaki verilerini analiz etmek için önemli derecede gayret sarf etmek gerekir. Bu verilerin toplanması, depolanması, işlenmesi ve analiz edilmesinin yanısıra doğruluğunu ve tutarlılığını sağlamak da gereklidir. Apache Spark, akan tweet'leri birkaç saniye içinde çok küçük gecikme ile analiz edebilmektedir. Shoro ve Soomro (2015), makalelerinde gelecek çalışmalardaki başarısıyla adını duyuracak bir teknoloji olan Apache Spark kolay kullanımıyla dikkatleri üzerine çekeceği belirtilmiştir [5].

Veri seli adı verilen yapı, GPS cihazlarının konum verileri, akıllı telefonlar, tabletler, bilgisayarların etkileşimleri ile sosyal medya paylaşımları, sensör verileri, e-mailler, fotoğraflar, videolar, log dosyaları tarafından yapılandırılmamış ya da yarı-yapılandırılmış verilerden oluşan yapıdır. Veri analizlerinin sonuçlarına göre kurum ve kuruluşların alacakları stratejik kararlar, hedef kitleye göre alınabilecek doğru kararların olasılığında artış olacağı öngörülmektedir. Böylece firmalar ve diğer kuruluşlar büyük veriyi analiz ederek gelişmeye yönelik fırsatlar yaratmaya çalışmaktadır. Birçok kurum ve kuruluşa ait yatırım planları, problem çözümleri, müşteri memnuniyeti faaliyetleri ile satış politikalarını oluşturmada büyük veri analizlerinden faydalanılması gerektiği görülmektedir. Dülger'in (2015), yaptığı çalışmada büyük veri analizlerinin yatırımlar üzerindeki etkileri araştırılıp büyük verinin önemli bir yapı olduğu ve yapılacak olan çalışmalarda sıkça adını duyuracağı öngörülmüştür [6].

Hızla artan konum verisi miktarı birçok uygulamanın gelişmesini sağlamıştır. Bulut sistemindeki verilere yönelik geniş ölçekli mekansal sorgularını ve ölçeklenebilirlik performansı uygulaması tasarlanmıştır. You vd. (2015), makalelerinde bulut sistemi dağıtımları için hazır prototip sistemlere sahip olan Apache Spark temelli SpatialSpark ve Impala temelli yöntem kullanılmıştır. Her iki sistem de endeksli uzaysal birleştirmeleri desteklemektedir. Amazon EC2 (Elastik Hesaplama Bulutu) kümesindeki veri kümeleri iyi ölçeklenebilirlik ile gösterildi. Gelecekteki çalışmalar için, ek olarak geometri performansını artırması, verinin paralel tasarımlarını entegre ederek aynı zamanda çok çekirdekli işlemciler üzerinde verimli kullanımı amaçlanmaktadır [7].

GPS cihazları ve akıllı telefonların yaygın kullanımı çok büyük miktarda mekânsal veri üretilmesini sağlamaktadır. Geleneksel veri işleme yöntemlerinin bu tür verilerin işlenebilmesi ve analiz edilebilmesinin yetersiz kaldığı görülerek son yıllarda alternatif çözümler üretilmeye başlanmıştır. Büyük veri teknolojileri büyük miktarlardaki verinin dağıtık bir şekilde ve yüksek performansla analiz edilmesine yardımcı olmaktadır. Kılıç'ın (2015) yaptığı çalışmada, büyük veri teknolojileri kullanılarak mekânsal verilerin analizi ve kümelenmesi incelenmiştir. Milyonlarca mekânsal veri noktalarının kısa zamanda kümelenebilmesi için açık kaynaklı Apache Hadoop ve Apache Mahout kullanılmıştır [8].

Tıp alanında uygulanan ilaç keşfi bağlamında proteinleri etkileyen moleküllerin tanımlanması, makine öğrenme tahmincisi aracılığıyla uzun süren ve paralellik gerektiren uygulamalar yapılmıştır. Harnie vd. (2015), makalelerinde Spark'ın iki büyük ölçekli sınıflandırma algoritmaları Lojistik Regresyon ve Doğrusal Destek Vektör Makinesi (SVM) uygulanmıştır. Spark'ın çalışma performansı düğüm sayılarına bağlı olarak test edilmiştir [9].

Akan veri analizinin, durumsal farkındalık kazanmak için güvenilir olması önemlidir. Gerçek zamanlı veriler için görsel analiz platformu geliştirmek amacıyla yapılan çalışmada bu tür veri akışlarının izlenmesi ve görsel analizi için NStreamAware adlı bir sistem önerilmiştir. Modern akışları analiz etmek için dağıtık işlem teknolojileri ile akış dilimleri kullanılmıştır. NVisAware adı verilen web tabanlı bir görsel analiz

uygulaması, kullanıcıya görsel olarak rehberlik eden bir sistem olup bir sosyal medya veri akışı üzerinde uygulanabilmektedir. NStreamAware, Apache Spark Streaming mimarisi üzerine kurulup web uygulaması olan NVisAware ayrılmaz bir parçası olarak inşa edilmiştir. Tüm sistem, analistlerin etkileşime girmesini sağlamaktadır. Tüm görsel analiz sistemi ile birlikte veri akışlarını anlamlı hale getirmek için kümeleme süreci segmentler ile anlamlı hale getirilmiştir. Fische ve Keim (2014), yaptıkları çalışmada Apache Spark Streaming aracılığıyla yeni ölçeklenebilir veri analitiği yöntemleri içermektedir [10].

Büyük veri, uzun süredir bilgisayar bilimi meraklıları için dünya çapında ilgi uyandıran bir konudur ve son zamanlarda sosyal medyanın beğenilerinden kaynaklanan verilerin sürekli artması ve verilerin daha derinlemesine analizini gerçekleştirmek için yeni ve etkili teknolojilere sahip olma arayışı önem kazanmıştır. Hadoop Map Reduce ve yakın zamanda tanıtılan Apache Spark'ın karşılaştırmaları yapılmıştır. Her ikisi de büyük verilerin analiz edilmesinde bir işleme modeli sağlamaktadır. Bu seçeneklerin her ikisi de büyük veri kavramına dayansa da, performansları uygulanmakta olan kullanım durumuna bağlı olarak önemli derecede değişmektedir. Büyük verilerin dinamik alanda değişkenliği, çeşitliliği ve analiz edilme ihtiyacı bu tür teknolojilerin gelişmesini sağlayacaktır. Gopalani ve Arora (2015), yaptıkları çalışmada, bu iki çerçeve kümeleme için standart bir makine öğrenme algoritması (K-Means) kullanılarak performans analiziyle karşılaştırılmıştır. Bu analiz sonucunda Spark'ın çok güçlü bir rakip olduğu görülmüş ve in-memory kullanarak büyük bir değişimi gerçekleştirdiği kanıtlanmıştır [11].

Büyük veri, çok sayıda özerk kaynağa sahip geniş hacimli, karmaşık, sürekli artan veri kümeleriyle ilgili bir yapıdır. İnternet ve ağ sistemlerinin gelişmesiyle, veri toplama ve depolama kapasitelerinin hızla gelişmesi ile büyük veri, tüm bilim ve mühendislik alanlarında hızla genişlemektedir. Wu vd. (2014), makalelerinde büyük veri devriminin özelliklerini karakterize eden ve veri madenciliği açısından bir büyük veri işleme modeli önermek üzere bir HACE teoremi sunulmaktadır. Bu veri odaklı model, talep kaynaklı bilgi kaynakları toplama, madencilik ve analiz, kullanıcı ilgi modellemesi ile güvenlik ve gizlilik hususlarını içermektedir [12].

Güçlü endüstriyel sistemlerin ve internetin gelişmesiyle nesnelerin interneti (IoT) teknolojisi için geniş ve çeşitli uygulama ortamları sağlanmaktadır. Geniş bir yelpazeye sahip endüstri alanındaki IoT uygulamaları teknolojinin geldiği son durumu özetler niteliktedir. Algılama, tanımlama, işleme, ağ iletişimi gibi işlemlerin özellikle sensörler yardımıyla gerçekleştirilip ayrıca otomatik izleme, kontrol, yönetim ve bakım işlemleri sağlanabilmektedir. Teknolojideki hızlı gelişmeler ve endüstriyel altyapı için IoT'nin yaygın olarak uygulanması beklenmektedir. Xu vd. (2014), yaptıkları çalışmada kayıt altına alınan verilerin analitiği sonucunda elde edilen bilgilerin gelecekteki endüstriyel araştırmacılar için ışık tutacak nitelikte olduğu görülmüştür [13].

Büyük veri analizi, verilerden çıkarımlar elde edilmesini amaçlayan araştırma alanıdır. Birçok farklı cihaz ve araçlardan alınan farklı türdeki veriler, veride çeşitliliği sağlamaktadır. Verideki artış karışıklığa neden olabilmektedir. Veri üzerinde önişleme ve algoritma geliştirme, çözüm geçerliliği ve değerlendirme işlemleri birbiriyle yakından ilişkilidir. Küme analizi yöntemleriyle etiketlenmemiş verileri inceleyip hiyerarşik bir yapı oluşturarak veriler gruplandırılır. Küme algoritmaları için evrensel olabilecek bir kümeleme algoritması yoktur. Xu ve Wunsch (2005), yaptıkları çalışmada veri türlerine ve istenilen sonuçlara ulaşabilmek için algoritmalar geliştirilebilir [14].

Günlük hayatta kullanılan teknolojik aletlerle kaydedilen veriler kayıt altına alınmaktadır. Bu verilerde karşılaşılan aksaklıklar uzun vadede şirketlerin ciddi oranda kayıplar yaşamasına neden olmaktadır. Gittikçe artan verilerin kontrolsüz yönetimi olumsuzluklara sebep olmaktadır. Büyük verilerin saklaması, analiz edilmesi ve bu verilerin görsel olarak ifade edilmesinde güvenilir tekniklerin kullanılmaması pek çok soruna neden olabilmektedir. Bu sorunlar veri kayıplarına da neden olabilmektedir. Yapılan çalışmada büyük verilerle çalışma yöntemleri araştırılmıştır. Büyük verilerle daha hızlı çalışan ve daha kararlı ve doğru sonuçlar alınabilmesine yönelik uygulamalar üzerinde durulmuştur. Kayım (2015), makalesinde büyük verilerle veri madenciliği algoritmalarının birlikte kullanımı ve performans değerlendirmeleri ile ele alınıp veri madenciliği algoritmalarından yoğunluk tabanlı gürültülü konumsal veri uygulama kümesi (DBSCAN) ve K-means

algoritmalarının paralelleştirmesi incelenmiştir. Sonrasında Pig, Hive, Impala performans karşılaştırılması yapılmıştır [15].

Yapılan çalışmada telekom firmaları için geliştirilmiş olan büyük veri sistemi açıklanmıştır. Sayısal Abone Hattı (DSL) adı verilen Telekom alanına özgü bir dil, Map-Reduce programlama modeli içeren paralel programlama platform ve sonuçların kullanıcıya sunulduğu bir arayüz bulunmaktadır. Şenbalcı (2013), makalesinde önerdiği DSL çözümü telekom firmalarına özgü telefon kayıtları, ağ kayıtları, link analizleri gibi verilerin paralel olarak işlenmesine olanak sağlamıştır. Veri merkezinde dağıtık yapıda bulanan cihazlar üzerinde işlemlerin paralel olarak yapılması incelenmiştir [16].

Çok büyük kaynak tanımlama çerçevesi (RDF) veri setlerini doğru bir şekilde sorgulamak karmaşık bir dağıtım stratejisi gerektirmektedir. Cure vd. (2015), makalelerinde RDF dağılımının değerlendirilmesi için Apache Spark üzerinde uygulanan algoritmalar incelenmiştir. Derinlemesine bir analiz ve dağıtım işlemlerinde incelenip Spark'ın hata toleransı, yüksek kullanılabilirlik ve ölçeklenebilirlik bakımından bu tür sistemlerde önemli olduğu görülmüştür. Her yaklaşımın uygulamadan bağımsız temel özellikleri veri hazırlama, yük dengeleme, veri çoğaltma ve bir dereceye kadar cevaplama maliyeti ve performans sorgusu içerir. Spark çerçevesi kullanarak yüksek performans elde edilmektedir [17].

Apache Hadoop ve MapReduce emtia kümeleri üzerinde büyük ölçekli, veri yoğunluklu toplu iş uygulamalarında son derece başarılı olmuştur. Bununla birlikte, düşük gecikmeli etkileşimli uygulamalar ve iteratif hesaplamalar için ortaya çıkan Apache Spark bellek içi işlem çerçevesi ile son zamanlarda dikkatleri üzerine çekmiştir. Lu vd. (2014), yaptıkları çalışmada yüksek performanslı bağlantılar için Spark'ın potansiyeli modern yüksek performanslı hesaplama (HPC) kümeleri üzerinde iyileştirme etkileri analiz edilmiştir. Uzaktan doğrudan bellek erişimi (RDMA) tabanlı bir öneri sunulup Spark'ın performansının arttırdığı belirlenmiştir. Aşamalı olay kontrol mimari (SEDA) ile RDMA tabanlı gerçek zamana dayalı yaklaşım tasarımları hem performans, hem de verimlilik sağlamaktadır [18].

Büyük verilerin standart bilgisayarlarda saklanması veya işlenmesi oldukça zordur. Büyük veri üzerinde analiz yapabilmek için yüksek hesaplama güçlerine ihtiyaç duyulmaktadır. Bu yüzden geleneksel işlemlerin yerine Hadoop Dağıtılmış Dosya Sistemi (HDFS), esnek dağıtımlı veri kümesi (RDD) gibi dağıtık dosya sistemleri ve bilgisayar kümeleri tercih edilmektedir. Yanı sıra Hadoop, Spark gibi platformlarda yaygınlaşmaktadır. Metin ve duygu analizleri, web üzerindeki etkileşimlerin analizleri, doğruluk tespiti, tavsiye sistemlerinin analizi ve sınıflandırılması gibi birçok uygulama makine öğrenmesi algoritmalarını kullanmaktadır. Hallaç (2014), yaptığı çalışmada büyük veri ve bulut bilişim teknolojileri, büyük veri üzerinde paralel algoritmaların çalıştırılması ve dağıtık makine öğrenmesi algoritmalarının büyük veriye uygulanması incelenmiştir [19].

Büyük veri analizinde sıklıkla Hadoop Map-Reduce yapısı kullanılmaktadır. Paralel veri hesaplamalarının sıklıkla kullanıldığı grafik işleme ünitesi (GPU)' lara bağlı görüntü işleme alanında yapılan uygulamalar son yıllarda kapsamlı olarak görülmeye başlamıştır. C tabanlı bir programlama modeli olan birleşik hesaplama aygıt mimarisi (CUDA), GPU birimlerini kullanır. Malakar ve Vydyanathan (2013), yaptıkları çalışmada CUDA'nın hızlandırmasını dağıtık bir sistem olan Hadoop'a entegre edilip yüksek performanslı görüntü işleme sistemi oluşturulmaya çalışılmıştır. Deney sonuçlarına göre Adaboost tabanlı yüz algılama algoritmasıyla CUDA ve Hadoop ile ölçeklenebilir, yüksek verimli, güç ve maliyet açısından verimli bilgi işlem platformları geliştirilebileceği görülmüştür. CUDA ve Hadoop'un birlikte kullanımıyla kümelemede % 25'lik bir verimlilik sağlandığı tespit edilmiştir [20].

Mevcut küresel ekonomi göz önüne alındığında dolandırıcılığı önleme ve tespit etme çabaları önem kazanmkatadır. Analiz edilmesi gereken verilerin hacmi arttıkça, veri madenciliği yöntemleri ve tekniklerinin kullanımı da artmaktadır. Şüpheli işlem izlemede etkili dolandırıcılık tespit yöntemleri ortaya çıkarıldı. Mevcut veri madenciliği tekniklerinden kümeleme algoritması sahtekarlık tespiti için sıkça kullanılan bir çözüm olmuştur. Sabau (2012), yaptığı çalışmada, dolandırıcılık tespitinde son on yılda kullanılan kümeleme teknikleri incelenmiş ve her biri gözden geçirilmiştir [21].

Aykırı değer algılama şu anda veri madenciliği alanında araştırmanın çok aktif olduğu bir alandır. Desen koleksiyonunda şekil bozukluğu bulma, veri madenciliği alanında çok iyi bilinen bir sorundur. Bir aykırı değer, veri kümesindeki kalıpların geri kalanına göre benzer olmayan bir modeldir. Aykırı algılama için önerilen yöntem, hibrid teknoloji kullanır. Yaklaşımın amacı, öncelikle veri kümesine kümeleme algoritmasını uygulamak ve sonra her bir sonuç için mesafe tabanlı yöntem kullanarak değerlerin bulunmasıdır. Aykırı değer olarak tanımlanan veriler eşik değerine bağlıdır. Eşik kullanıcı tarafından ayarlanır. İkinci aşamadaki ana hedef, kütle merkezlerinden uzak nesneleri bulmaktır. Pachgade ve Dhande (2012), önerdikleri yaklaşımda, veri kümesinden sapma değerlerini verimli bir şekilde bulmak için iki teknik birleştirilmektedir. Gerçek veri kümesini kullanan deneysel sonuçlar önerilen yöntemin daha az hesaplama maliyetli ve uzaktan tabanlı yöntemden daha iyi performans gösterdiğini göstermektedir [22].

Kredi kartları günümüzde birçok kişi tarafından kullanılmaktadır. Zaman kaybı nedeniyle insanların çoğu çevrimiçi alışverişe güvenmektedir. Çevrimiçi alışveriş, alışverişi yapan insanlar için en iyi yoldur. Ancak kredi kartı dolandırıcıları sadece rakamları değiştirerek kredi kartlarını kullanmaya çalışmaktadır. Bu gibi kredi kartı dolandırıcılığını önlemek için algoritmalar önerilmiştir. Dolayısıyla, kredi kartı numarasını kabul eden her sistemin, doğru kart numarasını tanımlayacak ve dolandırıcılık kredi kartı numarasını kullanmaktan kaçınacak bir mekanizması olmalıdır. K-means algoritmasını değiştirerek dolandırıcılık kredi kartı numaraları tespit edilebilir. Singh vd. (2014), önerdikleri geliştirilmiş algoritmanın uygulanması kolaydır ve K-means kümeleme algoritmasında kullanılacak başlangıç merkezlerini belirlemek için daha iyi bir yöntem olduğu kanıtlanmıştır [23].

Büyük veri analitiği, verimsiz depolama, işleme gecikmeleri, düşük bilgi alma oranı, geleneksel yöntemlerle ele alınması ve yönetilmesi mümkün olmayan karmaşık algoritmalar gibi çeşitli zorluklarla karşı karşıyadır. Yazılım geliştiricilerine büyük veri zorluklarıyla baş etmelerine yardımcı olması için yeni programlama çerçeveleri gerekmektedir. Yapılan araştırmada, sırasıyla disk üzerinde ve bellek içi hesaplamayı destekleyen Hadoop MapReduce ve Apache Spark incelenmiştir. Kümeleme, bilgi toplama ve bilgi keşfi için kullanılan büyük veri madenciliğinin önemli

görevlerinden biridir. Ketu ve Agarwal (2015), yaptıkları çalışmada dağıtık K-Means kümelemesinin bellek içi ve diskte hesaplama modellerine dayalı performansının analizi yapılmıştır [24].

K-means tabanlı kümeleme algoritmaları mesafe hesaplanması ve merkezlerin güncellenmesi dahil olmak üzere iki aşamalı yenilemeli prosedürü kullanmaktadır. Wang vd. (2016), yaptıkları çalışmada Apache Spark'taki K-means algoritmasını paralelleştirmek için katkıda bulunulmuştur [25].

BÖLÜM 3

BÜYÜK VERİ KAVRAMI

Büyük veri, geleneksel temel yöntemlerle işlenmesi mümkün olmayıp hacimsel olarak diskte fazlasıyla yer tutan verilere verilen isimdir. Teknolojinin ve internetin gelişmesi bilginin gücünü açığa çıkarmanın gereksinimine ve bilginin işlenmesinde yeni teknolojilerin arayışına zemin hazırlamıştır.

Büyük verilerin tüm gücünü açığa çıkarmak için sistematik tasarımlar uygulayan yüksek performanslı bilgi işlem platformları gereklidir. Büyük verinin yeni bir trend olarak görülmesi ve büyük veri madenciliği ihtiyacı tüm bilim ve mühendislik alanlarında ortaya çıkmaktadır. Sagiroglu ve Sinanc (2013), makalelerinde Büyük veri teknolojileri sayesinde, toplumumuzu daha iyi anlamak için gerçek zamanlı olarak en doğru analizler ile geri bildirimler almak için yeni teknolojiler gelişmeye devam ettiği belirtilmiştir [26].

3.1. BÜYÜK VERİ NEDİR?

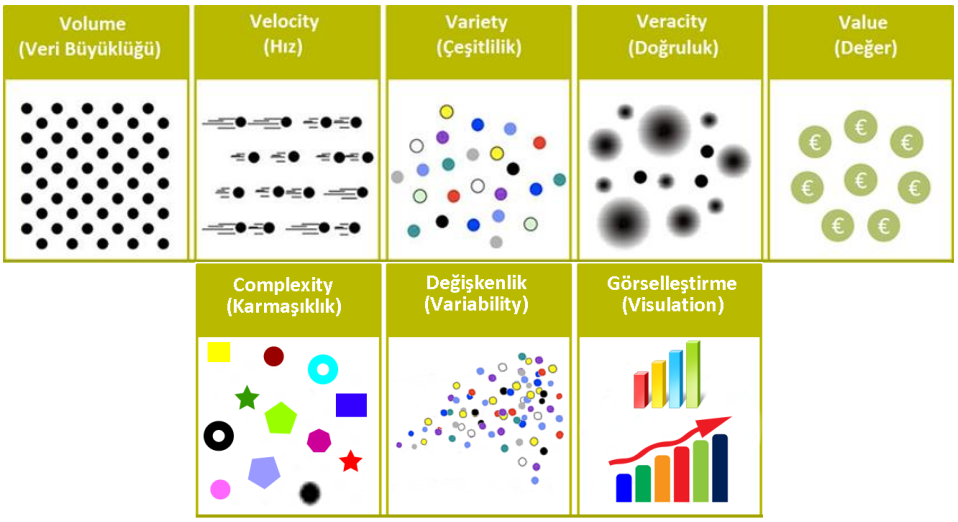
Teknolojinin ilerlemesi ve internetin hızlı gelişimi, bilginin gücünün ön plana çıkmasını sağlamıştır. İnternet dünyasındaki sosyal medya paylaşımları, sürekli kayıt edilen log dosyaları, meteorolojik veriler, sensör verileri, GPS verileri, uzay verileri gibi birçok farklı türde ve sürekli artış gösteren verilerin kapladıkları alanda artmaktadır. Şekil 3.1'de büyük veriyi oluşturan farklı kaynakların benzetimi verilmiştir. Büyük hacimlere sahip olan bu veri kümelerinin çeşitli ve karmaşık yapıya sahip olmaları büyük veri olarak adlandırılır ve işlenmesi geleneksel yöntemlerle mümkün olmayan bir yapıdır. Bu verilerin saklanması, işlenmesi, analizi ve görselleştirilebilmesi için yeni teknikler geliştirilmektedir. Büyük veri topluluklarından anlamlı verilerin elde edilmesi ve veri analitiği sürecinde elde

edilen yararlı bilgilerin şirket ve kuruluşlara daha zengin ve derin anlayışlar kazandırıp rekabette avantaj elde etmelerine destek olur. Bu nedenle, büyük veri uygulamalarının mümkün olduğunca doğru analiz edilmesi ve yürütülmesi gerekir.

Şekil 3.1. Büyük veri gösterimi.

3.2. BÜYÜK VERİNİN ÖZELLİKLERİ

Veriler büyük miktarda verinin (exabytes) işlenmesini zorlaştırarak büyük bir hızla büyümektedir. Böyle büyük miktardaki verilerin taşınmasında başlıca zorluk, bilgi işlem kaynaklarına kıyasla hacimlerin hızla artmasıdır. Günümüzde kullanılan büyük veri terimi, diğer mevcut özelliklerine çok fazla önem vermeyen yalnızca veri boyutunu işaret ettiği için yanlış isimlendirilmektedir. Şekil 3.2'de büyük veri özelliklerinin genel gösterimi verilmektedir.



Şekil 3.2. Büyük veri özelliklerinin genel gösterimi.

Büyük veri yapıları aşağıdaki özelliklerle ilişkilendirilerek tanımlanabilir.

3.2.1. Hacim (Volume)

Büyük verinin büyük kelimesi verinin hacimsel olarak tanımlanmasını sağlar. Günümüzde kullanılan veriler petabayt boyutundadır ve yakın gelecekte zettabayta çıkması beklenmektedir. Mevcut sosyal paylaşım siteleri günlük olarak terabayt boyutunda veri üretmekte ve bu miktarda verinin mevcut geleneksel sistemleri kullanarak ele alınması kolay bir yönetme olarak görülmemektedir.

3.2.2. Hız (Velocity)

Çeşitli kaynaklardan gelen verilerin hızıyla ilgilenen bir kavramdır. Bu karakteristik özellik gelen verilerin hızı ile sınırlı olmayıp, aynı zamanda verinin aktığı hızı da ifade etmektedir. Verinin üretilmesindeki hız, işlem sayısını ve çeşitliliğinin aynı hızda artırması sonucunu doğurmaktadır. Geleneksel sistemler, sürekli hareket halindeki verilere analitik bilgi verme yeteneğine sahip değildir.

3.2.3. Çeşitlilik (Variety)

Üretilen veriler sadece geleneksel veriden değil aynı zamanda web sayfalarının oluşturduğu veriler, web günlük dosyaları, sosyal medya verileri, e-postalar, dokümanlar, uzay verileri, algılayıcı cihazların oluşturdukları veriler gibi çeşitli kaynaklardan gelen ve farklı yapılara sahip verilerden oluşmaktadır. Tüm bu veriler, mevcut geleneksel analitik sistemler tarafından işlenmesi zor olan yapılandırılmış, yarı yapılandırılmış ve hatta yapılandırılmamış verileri içeren tamamen farklı yapılardan oluşabilmektedir. Soyal ağlar, GPS cihazları, sensörler, akıllı telefonlar gibi farklı ve çeşitli kaynaklardan elde edilen verilerin yapısal, yarı yapısal ve yapısal olmayan formlarda bulunmaları çeşitliliği oluşturmaktadır. Özellikle yapısal olmayan veri, önceden tanımlı bir veri modeline sahip olmayan verileri ifade etmektedir. Günümüzde, farklı teknolojik cihazlar farklı formlara sahip veriler üretebilmektedir. Bu verilerin bir araya gelmesi ve miktarının artması sonucu yapısal olmayan veri yığınları oluşmaktadır.

3.2.4. Güvenirlik (Veracity)

Verilerin doğru olduğundan emin olmakla ilgili olup sistemdeki yanlış ve gerçek olmayan verilerin sistemde birikmesinin önlenmesini gerektirir. Veracity, verilere, algoritmalara ve varsayımlara dayanır.

3.2.5. Değer (Value)

Kullanıcılar, saklanan verilerde işlemler yapıp bazı sorguları çalıştırabilir ve böylece elde edilen filtrelenmiş verilerden önemli sonuçları çıkartabilir ve ayrıca, istedikleri boyutlara göre sıralayabilir. Bu raporlar, şirketlerin stratejilerini değiştirebildikleri iş eğilimlerini keşfetmesine yardımcı olur. Farklı kurum ve kuruluşlar tarafından depolanan veriler kendileri tarafından veri analizi için kullanılmaktadır. Elde edilen analiz sonuçları şirketlerin daha fazla kar oranı elde etmeleri için yol gösterici niteliktedir.

Böylelikle, Bilişim Teknolojileri (BT) Uzmanları'nın büyük verileri işlemede önündeki başlıca zorluklar arasında; büyük miktarda veriyi etkili bir biçimde işleyebilecek sistemler tasarlamak, kurum ve kuruluşlar tarafından toplanan tüm verilerin en önemli verilerini filtrelemektir. Bir başka ifadeyle, işe değer katmaktır.

3.2.6. Karmaşıklık (Complexity)

Çeşitli kaynaklardan gelen sistemler arasında veri bağlantısı sağlamak, eşleştirmek, temizlemek ve dönüştürmek oldukça büyük ve zahmetli bir iştir. İlişkileri, hiyerarşileri ve çoklu veri bağlantılarını birbirine bağlamak ve ilişkilendirmenin de gerekli olduğu durumlarda geleneksel sistemler yetersiz kaldığından yeni teknolojilere verilen önem artmaktadır.

3.2.7. Değişkenlik (Variability)

Veri akışının tutarsızlıklarını göz önüne alır. Veri yükleri, özellikle bazı olayların meydana geldiği veri yüklerinde zirveye neden olan sosyal medyanın kullanımındaki

artışla korunması zorlaşmaktadır. Çok sayıda veri kaynağı ve türünden kaynaklanan çok sayıda veri boyutundan dolayı büyük veriler de değişkendir. Değişkenlik ayrıca, büyük verilerin veritabanına yüklendiği tutarsız hız anlamına da gelebilir.

3.2.8. Görselleştirme (Visualization)

Veriyi görselleştirmek, insan beyninin karmaşık yapılardaki verileri grafik ve tablolar ile daha iyi anlamasını sağlar. Veriyi görselleştirerek analiz sonuçlarının daha iyi ve etkili bir şekilde evrensel olarak anlaşılabilir olmasına katkı sağlar.

3.3. BÜYÜK VERİNİN ZORLUKLARI

Büyük verilerin ortaya çıkmasının sebebi, veri yoğunluğuna sahip teknolojilerin kullanımını artıran bir toplumda yaşıyor olmamızdır. Büyük verilerin güncel özelliklerinden birisi, ilişkisel veri tabanları ile istatistik ve görselleştirme kullanılarak paralel yazılımların onlarca, yüzlerce hatta binlerce sunucunun üzerinde çalışmasını gerektiren zorluklarla çalışmasıdır. Büyük veri yönetiminde karşılaşılan çeşitli zorluklar arasında ölçeklenebilirlik, yapılandırılmamış veriler, erişilebilirlik, gerçek zamanlı analitik, hata toleransı ve çok daha fazlası bulunur. Farklı sektörlerde depolanan verilerin çeşitliliğine ek olarak, üretilen ve depolanan veri türleri, yani verilerin video, resim, ses, metin veya sayısal bilgiyi kodladığı da endüstriden endüstriye farklılık göstermektedir. Büyük veriler, büyük miktarda veriyi etkili bir şekilde işlemek için tolere edilebilir. Büyük verilere uygulanan teknolojiler, büyük paralel işleme (MPP) veritabanları, veri madenciliği ızgaraları, dağıtılmış dosya sistemleri, dağıtılmış veritabanlar, bulut bilgi işlem platformları, internet ve ölçeklenebilir depolama sistemleri içerir. Gerçek veya yakın gerçek zamanlı bilgi dağıtımı, büyük veri analizinin belirleyici özelliklerinden biridir. Büyük veri toplamak, yönlendirmek, analiz etmek ve görselleştirmek için çok çeşitli teknikler ve teknolojiler geliştirilmiş ve uyarlanmıştır. Bu teknikler ve teknolojiler istatistik, bilgisayar bilimleri, uygulanan matematik ve ekonomi gibi çeşitli alanlardan yararlanmaktadır. Patel vd. (2012), büyük veriden değer elde etmek isteyen bir organizasyonun esnek, çok disiplinli bir yaklaşım benimsemesi gerektiği anlamına gelmediği belirtilmiştir [27].

3.4. BÜYÜK VERİNİN UYGULAMA ALANLARI

Büyük veriler, geleneksel depolarda depolanan verilerden farklıdır. Burada depolanan veriler temizlenmeli, belgelenmeli ve güvenilir olmalıdır. Dahası depolanacak veritabanının temel yapısına uygun olması gerekir. Daha fazla verinin analizi daha iyi sonuçların elde edilmesini sağlar.

BT endüstri sektöründe, dosyaları çözmek ve nadiren ortaya çıkan sorunlarla başa çıkmak için çok miktarda veriyi log dosyaları olarak depolarlar. Ancak, bu verilerin depolanması, birkaç hafta kadar bir süre boyunca yapılır, ancak bu günlükler, değerlerinden dolayı daha uzun süre saklanmalıdır. Geleneksel sistemler, hacimleri, ham veya yarı yapılandırılmış yapıları nedeniyle bu günlükleri idare etmekte başarılı olamamaktadır.

Büyük veri için büyük miktarda sensör verisinin analizi de büyük bir iştir. Günümüzde bu büyük miktarda veri ile uğraşan bütün endüstriler, depolama altyapısının ve analiz tekniklerinin eksikliği nedeniyle analiz için bunun küçük bir bölümünü kullanmaktadır. Ayrıca, sensör verisi hareket halindeki verilerle ve durağan verileriyle karakterizedir. Bu nedenle güvenlik, kar ve daha iyi iş anlayışı için büyük miktarda verinin analiz edilmesini gerektirir.

Finansal kurumlar, risk hesaplamaları için, verileri kabul edilebilir eşiklerin altına düşecek şekilde modellemek için önemli çaba sarfetmektedir. Çok miktarda verinin potansiyel olarak yetersiz bir şekilde kullanılmasını önlemek ve risk modellerini daha doğru bir şekilde belirlemek için bu potansiyel modelin içine entegre edilmelidir.

Büyük verilerin en çok kullanıldığı alan, sosyal medya ve müşteri analizleri içindir. Müşterilerin ürünleriyle ilgili hareketlerinin dikkate alınması, ticari kuruluşların bir tür müşteri geribildirimi almasına yardımcı olur. Katal vd. (2013), çalışmalarında bu geri bildirim daha sonra şirketlerin kar oranlarını artırma ve işlerine daha fazla değer katmak için kullanıldığı belirtilmiştir [28].

3.5. BÜYÜK VERİ FİRMALARI

Singh ve Singh (2012), makalelerinde büyük veri firmaları hakkında bilgilere yer vermiştir. IBM, Kognitio, ParAccel & SAND gibi büyük veri analitik çözümleri sunan birçok kuruluş vardır. IBM InfoSphere BigInsights, yapısal ve yapısal olmayan verilerin büyük miktarlarını yönetmek ve analiz etmek için kullanılan bir Apache Hadoop tabanlı bir çözümdür. IBM Big Sheet ile açık kaynaklı bir Apache Hadoop üzerine inşa edilmiş ve çeşitli performans, güvenilirlik, güvenlik ve idari özelliklere sahiptir. IBM Big Sheet, veri araştırması için kullanılan karmaşık bir metin analitiği modülüdür. BigInsights, herhangi bir şema ya da yapı yüklemeden kendi verilerinde verileri analiz edebilmekte ve hızlı analiz yapabilmektedir. Bulut üzerindeki BigInsights, hem kamu hem özel hem de hibrit bulut dağıtım seçenekleriyle temel ve kurumsal sürümlerde bulunmaktadır. Temel sürüm, ücretsiz olarak sunulan giriş seviyesinde bir sunumdur ve kuruluşların, büyük veri analizi hakkında bilgi edinmelerine yardımcı olur. Müşteriler hazır olduklarında kurumsal sürüme sorunsuzca geçebilmektedirler.

WX2 Kognitio Analitik Platformu, hızlı ve ölçeklenebilir bir bellek içi analitik veritabanı platformudur. 3 yoldan dağıtılabilir: yalnızca yazılım lisanslaması olarak, endüstri standardı donanımda çalışan tamamen konfigüre edilmiş bir veri ambar cihazı veya Kognitio'nun bulut tabanlı veri ambarı hizmeti olarak (DaaS) çözüm sunulur. WX2 müşterisinin en büyük verileri, 9 ½ terabayt kullanıcı verisi için bir lisans satın aldı. Tipik Kognitio WX2 yapılandırmaları, CPU çekirdeği başına yüzlerce kullanıcı verisi dizisine sahiptir. Düğüm noktaları tarafından ölçülen en büyük mevcut sistem 300 sunucuya sahiptir.

ParAccel Analitik Platformu (PADB), MPP analitik veritabanı platformu sunmaktadır. Sorgu optimizasyonu ve derlemesi, sıkıştırma ve ağ bağlantısı için güçlü özelliklere sahiptir. Herhangi bir uzunluğa ve karmaşıklığa bakılmaksızın yapılandırılmış sorgu dili (SQL) kodunu optimize edebilen Omne optimizer ile birlikte gelir.

IBM InfoSphere Streams, minimum milisaniyelik yanıt süresince büyük miktarlardaki akan verilerin sürekli bir şekilde analiz edilmesini sağlar. IBM'in bu ürünü, çok çeşitli yapısal ve yapısal olmayan veri türlerini destekleyebilen oldukça ölçeklenebilir ve çevik bir altyapı sağlar.

SAND Analitik Platformu, MPP ile doğrusal veri ölçeklenebilirliği sağlayan sütunsal bir analitik veritabanı platformudur. Karışık iş yükleri ve sonsuz sorgu optimizasyonu ile binlerce eş zamanlı kullanıcıyı destekleyecek şekilde tasarlanmıştır. Ayrıca anında veri testi için bellek içi analiz, tam metin arama ve veri testi için SANDbox'ı destekler. SAND Analitik Platformu, müşteri pazarlaması, dönüşüm analitiği ve finansal analitik gibi karmaşık analitik görevlere odaklanmaktadır [29].

3.6. BÜYÜK VERİ ANALİZİ

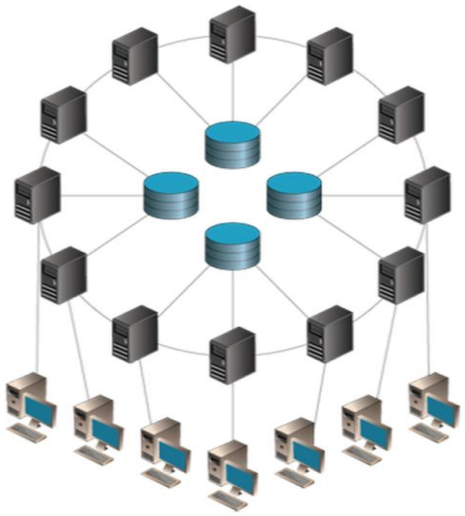
Son on yılda şirketler büyük verileri avantajlara dönüştürmek için stratejik kararlar aldılar. Büyük miktarlardaki verinin mevcut çevrimiçi depolama sistemlerinin potansiyelini ve genel bilgisayar sistemlerini zorladığı görülmektedir.

Her yıl daha fazla artacak olan büyük verileri veri bilimcileri yönetmek zorunda kalacaktır. Büyük veriler bize çeşitli alanlardaki çok miktarda veriyi sunup bu alanları geliştirmek için bir fırsat sunmaktadır. Çeşitli çevrimiçi alışveriş web siteleri EBay.com gibi büyük veri setleri kullanır. 7.5 ve 40 petabayt (PB) boyutundaki ambarlar ile Hadoop arama kümesini kullanmaktadır. Amazon özellikle Linux'u kullanmakta ve veritabanlarının kapasitesi 7.8, 18.5 ve 24,7 terabayt (TB)' tır.

Çevremiz büyük verilerle doludur. Önemli nedenleri arasında; şirketler için kullanıcılar ve artan sosyal medya kullanımları, analog teknolojiler yerine dijital teknolojilerin kullanımındaki artış, internete bağlı cihazların ve sistemlerin artışı görülmektedir.

3.7. DAĞITIK DEPOLAMA SİSTEMLERİ

Bir ağ bağlantısı sayesiyle biligisayarların sahip olduğu yazılım ve donanım bileşenlerinin haberleşebildiği sistemlere dağıtık sistemler denir. Dağıtık depolama ise verilerin bir ya da daha fazla sunucu üzerinde depolanıp istemci bilgisayarların bu verilere erişebilmelerini sağlayan dosya istemidir. Karasulu ve Korukoglu (2008), makalelerinde dağıtık depolama sistemlerinin yapısal karşılaştırması yapılmıştır [30]. Şekil 3.3'te dağıtık sistem mimarisi yapısı gösterilmiştir. Büyük verileri işleme ve değerlendirme için tutarlı verileri ölçeklenebilir özellikte bir depo oluşturmak gerekir. Yüksek hacimli veriyi dağıtık depolama alanlarında depolamak ve kullanmak için bazı özelliklere sahip olması gerekir.



Şekil 3.3. Dağıtık sistem mimarisi.

3.7.1. Tutarlılık

Bu depolama sisteminin en önemli özelliğidir. Özdeş verilerin çok sayıda kopyası alınır. Dağıtık depolama sistemi, verileri depolayacak sayıda sunucuya sahiptir. Kullanılabilirliğini sağlamak için parçalara bölününen veriler farklı sunucu sistemlerinde saklanır. Sunucu hatası olasılığı veri tutarsızlığına neden olabilir.

3.7.2. Kullanılabilirlik

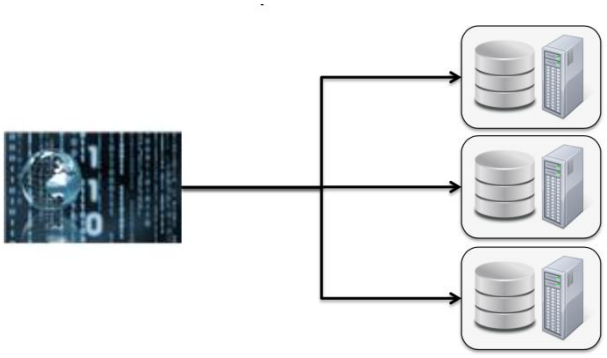
Dağıtık depolama sisteminde sunucuların sunucu hatasıyla karşı karşıya kalması kullanılabilirlik açısından tercih edilmeyen bir durumdur. Sunucu hataları durumunda sistemlerin birbirinden etkilenmemesi istenir.

3.7.3. Bölme Toleransı

Dağıtık depolama sisteminde, sunucular birbirlerine bir ağ aracılığıyla bağlanırlar. Bağlantı, düğüm hatası veya geçici tıkanıklıklar olabilir. Bu sistemler az bir miktarda toleransa sahiptir.

3.8. HADOOP

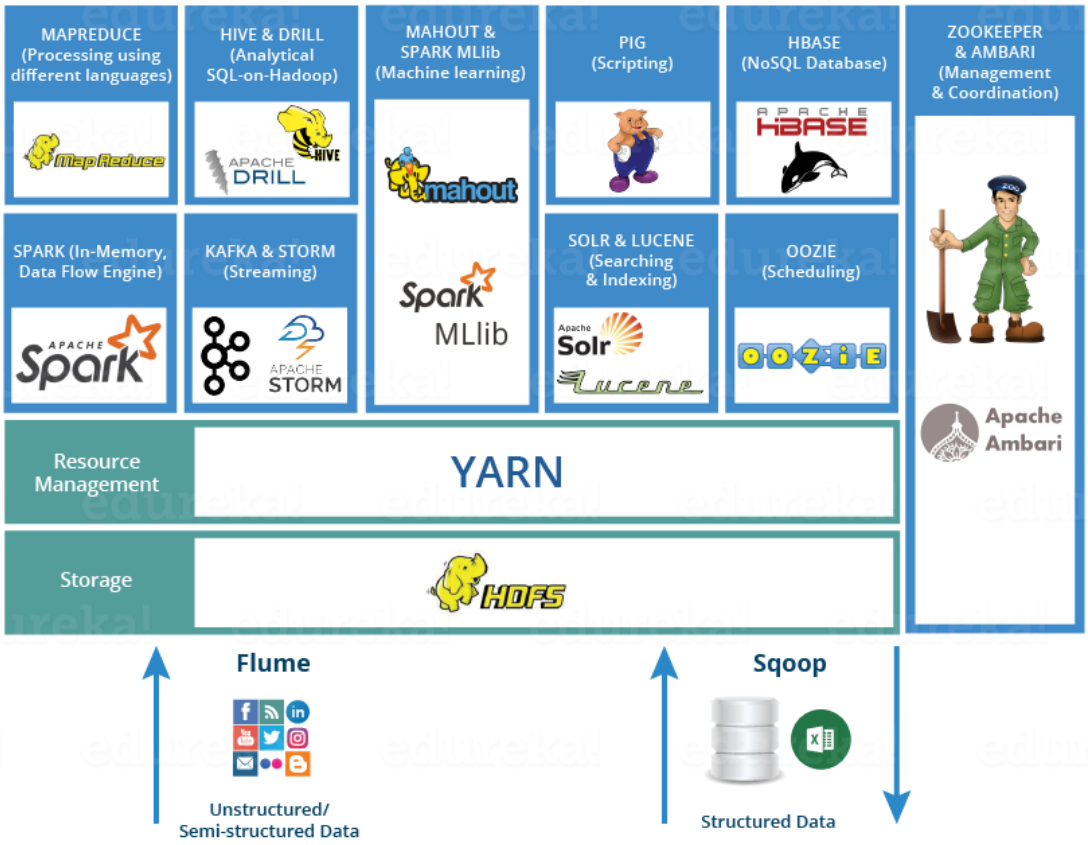
Apache Hadoop projesi, güvenilir, ölçeklenebilir, dağıtılmış bilgi işlem için açık kaynaklı yazılımlar geliştirmektedir. Apache Hadoop yazılım kütüphanesi, basit bir programlama modeli kullanarak geniş veri kümelerinin bilgisayar kümeleri arasında dağıtılmasını sağlayan bir çerçevedir. Uygulamalar, binlerce bağımsız bilgisayar ve petabayt veri ile çalışmaya olanak tanımaktadır. Şekil 3.4'te Hadoop mimari yapısı gösterilmiştir. Hadoop, Google'ın MapReduce ve Google Dosya Sistemi'nden (GFS) elde edilmiştir [27].



Şekil 3.4. Hadoop mimari yapısı.

Hadoop geniş yelpazeli bir ekosisteme sahiptir. Map-Reduce, Pig, Hbase, Zookeeper, Spark, Kafka gibi birçok işlevi bünyesinde bulunduran Hadoop sistemi veri

bilimcileri için çok kullanışlı bir yapı sunmaktadır. Şekil 3.5'te Hadoop ekositemi gösterilmiştir. Hadoop kendi içerinde bir çok projeyi barındırır.Dağıtık dosya sistemi için HDFS,kaynak yönetimini sağlayan YARN, SQL benzeri sorgular için Hive, veritabanı çözümü için HBASE, makine öğrenmesi yöntemleri için Mahout ve MLlib, arama ve dizin oluşturma için Solr ile Lucene, ilişkisel veritabanları ile Hadoop sistemi arasında veri aktarımını sağlayan Sqoop, günlük verilerin toplanmasını, bir araya getirilmesini, taşınmasını sağlayan Flume bunlardan bazılarıdır.



Şekil 3.5. Hadoop ekosistemi.

3.8.1. HDFS Mimarisi

(HDFS), hata toleransını sağlayan, emtia donanımında çalışmak üzere tasarlanmış veri işleme motoruna sahip dağıtılmış bir dosya sistemidir. HDFS, uygulama verisine yüksek verimli erişim sağlar ve büyük veri kümelerine sahip uygulamalar için uygundur. Hadoop, binlerce sunucudaki verileri depolayabilen bir dağıtılmış dosya

sistemi HDFS ve bu makineler arasında işi çalıştıran (Haritalama/İndirgeme işleri (Map-Reduce)) bir araçtır; bu veriler verinin yakınında çalışır. HDFS, master/slave mimariye sahiptir. Büyük veri, otomatik olarak, hadoop kümesindeki farklı düğümler tarafından yönetilen parçalara ayrılır. Hadoop bağımsız, büyük verilerin hesaplanmasını güçlendiren java tabanlı, dağıtık bir bilgi işlem ortamı çerçevesine sahiptir. Apache Software Foundation, Hadoop'u Apache'nin bir parçası olarak desteklemektedir.

HDFS, makineler arasında çok büyük dosyaları büyük bir kümede güvenilir bir şekilde depolamak için tasarlanmıştır. Her dosyayı bir dizi blok halinde depolar ve son blok haricindeki bir dosyadaki tüm bloklar aynı boyuttadır. Bir dosyanın blokları hata toleransı için çoğaltılır. Blok boyutu ve çoğaltma faktörü, dosya başına yapılandırılabilmektedir. Bir uygulama, bir dosyanın çoğaltma sayısını belirleyebilir. Çoğaltma faktörü, dosya oluşturma zamanında belirtilebilmekte ve daha sonra değiştirilebilmektedir. HDFS'deki dosyalar bir kez yazılır ve herhangi bir zamanda kesinlikle bir yazara sahip olurlar. NameNodu, blokların çoğaltılması ile ilgili tüm kararları vermektedir. Kümedeki her DataNode'dan periyodik olarak bir Heartbeat ve bir Blockreport alır. Bir Heartbeat alındığında, DataNode'un düzgün çalıştığı anlaşılır. Bir Blockreport, bir DataNode'daki tüm blokların bir listesini içerir. HDFS'nin görevi, bir sunucu ve iki bağımlı düğüm olan iki paralel küme üzerinde dağıtım işlemidir. Veri analizi görevleri bu kümelerde dağıtılmaktadır.

3.8.2. Map-Reduce

MapReduce, 2004 yılında Google tarafından geniş veri kümeleri üzerinde dağıtık hesaplamayı desteklemek üzere tanıtılan bir yazılım çerçevesidir. MapReduce bir programlama modelidir. Anahtar/değer çifti kümesi ve aynı anahtarla ilişkili tüm ara değerleri birleştiren bir azaltma fonksiyonu üretmek için bir anahtar/değer çiftini işleyen bir harita işlevi belirttiler [27].

Map-Reduce yapısı Java, C++, Python, Perl, Ruby, C vs. gibi birçok dille geliştirilebilmektedir. Apache Hadoop gibi büyük sistemlerle kullanılabilmektedir. Gazal ve Kaur (2015), çalışmalarında büyük veri depolama stratejilerini

incelemişlerdir. Map-Reduce yapılacak işleri parçalara ayırıp ayrılan iş parçacıklarını diğer sunuculara dağıtmaktadır. Diğer sunucularda işlenen verilerin sonuçlarını birleştirip tek bir sonuca indirgeyen bir yapı sunmaktadır [31].

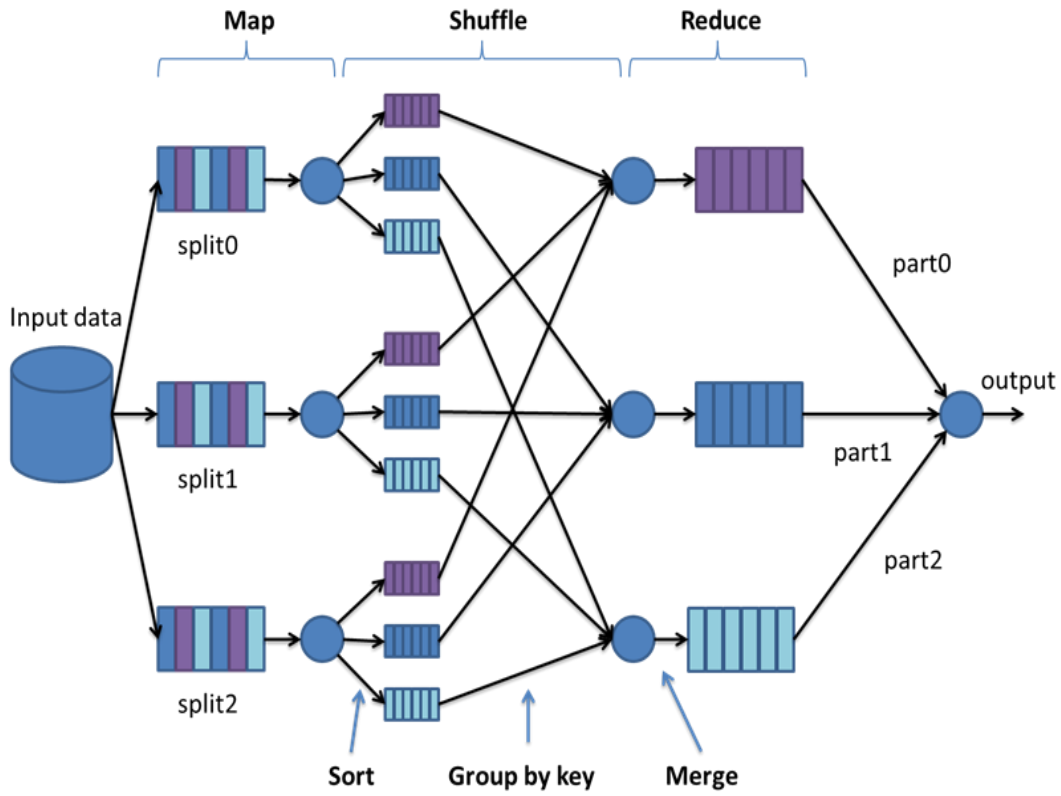
"Map" adımı: Ana düğüm girişi alır, daha küçük alt parçalara ayırır ve onları işçi düğümlerine dağıtır. Bir işçi düğümü bunu tekrar yaparak çok düzeyli bir ağaç yapısına götürür. İşçi düğümü daha küçük olan sorunu işler ve cevabı ana düğüme geri gönderir. Harita, bir veri alanındaki bir türe sahip bir çift veri alır ve farklı bir alandaki çiftlerin bir listesini döndürür:

$$\text{Map } (K1, V1) \rightarrow \text{liste } (K2, V2)$$

"Reduce" adımı: Ana düğüm, daha sonra tüm alt problemlere cevapları toplar ve onları, başlangıçta çözmeye çalışılan sorunun cevabı olan çıktıyı oluşturmak için bir şekilde birleştirir. Redüksiyon işlevi daha sonra her bir gruba paralel olarak uygulanır; bu da aynı alanda bir değer topluluğu oluşturur:

$$\text{Reduce}(K2, \text{liste } (V2)) \rightarrow \text{liste } (V3)$$ [27].

Şekil 3.6'da Map-Reduce sisteminin Map, Shuffle ve Reduce işlemlerinin çalışma yapısı gösterilmiştir.



Şekil 3.6. Map-Reduce mimarisi [32].

3.8.2.1. Map-Reduce Bileşenleri

Ad-Düğümü (Name-Node): HDFS meta verilerini yönetir, doğrudan dosyalarla ilgilenmez.

Veri Düğümü (Data-Node): Her blok için HDFS-defaultreplication düzeyinin bloklarını depolar.

İş İzleyicisi (Job Tracker): Yeni hesaplardaki işyeri tahkiklerini planlar, ayırır ve izler.

Görev İzleyicisi (Task Tracker): Map Reduce işlemleri çalıştırır, genel olarak kullanışlı haritalar, redüktörler ve bölücülerden oluşan bir kütüphane ile birlikte gelir.

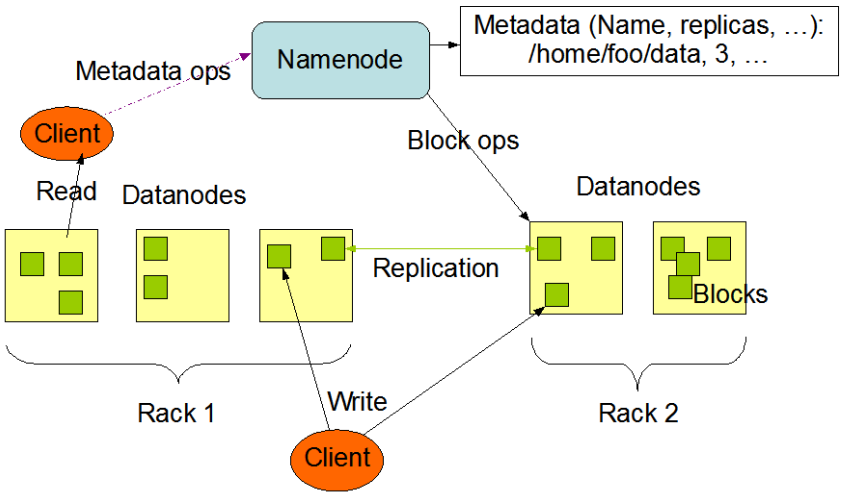
3.8.2.2. Map-Reduce Teknikleri

1. Harita() girişinin hazırlanması - "MapReduce sistemi" Map işlemcilerini, her işlemcinin üzerinde çalışacağı K1 giriş anahtarını atar ve o işlemciye o anahtar değeri ile ilişkili tüm giriş verilerini sağlar.
2. Kullanıcı tarafından sağlanan Map() kodunu çalışması - Map(), her K1 anahtarı değeri için tam olarak bir kez çalıştırılır ve K2 anahtar değerlerine göre düzenlenmiş çıktı üretilir.
3. Reduce işleminin Map çıktısı "Shuffle" - MapReduce sistemi Reduce işlemcilerini belirler, her işlemcinin üzerinde çalışması gereken K2 anahtar değerini atar ve o işlemciye o anahtar değeriyle ilişkili tüm Map tarafından oluşturulan verileri sağlar.
4. Kullanıcı tarafından sağlanan Reduce() kodu - Reduce(), Map adımında üretilen her K2 anahtarı değeri için bir kez çalıştırılır.
5. Son çıktı üretimi - MapReduce sistemi, tüm Reduce çıktısını toplar ve nihai sonucu elde etmek için K2'ye göre sıralar.

Bir HDFS kümesi, dosya sistemini yöneten bir ana sunucu olan tek bir NameNode'dan oluşur. Buna ek olarak, kümede her düğüm başına bir tane olmak üzere, üzerinde çalıştırdıkları düğümlere bağlı depolamayı yöneten bir dizi DataNode vardır. HDFS bir dosya sistemi ad alanını sunar ve kullanıcı verilerinin dosyalarda saklanmasına izin verir. Dahili olarak, bir dosya bir veya daha fazla bloğa bölünür ve bu bloklar bir DataNode kümesinde saklanır. NameNode, blokların Datanodlara eşleştirilmesini belirler. HDFS, makineler arasında çok büyük dosyaları büyük bir kümede güvenilir bir şekilde depolamak için tasarlanmıştır. Her dosyayı bir dizi blok halinde saklar.

3.8.3. Hadoop Kümesi Üst Düzey Mimarisi

Hadoop kümesi tek bir ana ve birden fazla köleden diğer bir deyişle "işçi düğümleri"nden oluşur. Şekil 3.7 'de HDFS mimari yapısı gösterilmiştir.



Şekil 3.7. HDFS mimarisi.

Map, Reduce ve Shuffle işlemleri TaskTracker ve JobTracker süreçlerinden oluşur. Ana düğüm bir JobTracker, TaskTracker, NameNode ve DataNode'dan oluşur. Köle veya işçi düğümü, hem bir DataNode hem de TaskTracker gibi davranır. Daha büyük bir kümede, HDFS, dosya sistemi dizinini barındırmak için özel bir NameNode sunucusu ve ad düğümünün bellek yapılarının anlık görüntülerini oluşturan ve böylece dosya sistemi bozulmasını önleyen ve veri kaybını azaltabilen ikinci bir NameNode ile yönetilir [27].

Büyük Veri özel niteliğinden dolayı, dağıtık dosya sistemi mimarisinde saklanır. Apache'nin Hadoop ve HDFS, Büyük Verileri depolamak ve yönetmek için yaygın bir şekilde kullanılmaktadır. Büyük Verileri analiz etmek, hataya dayanıklı, esnek ve ölçeklendirilebilir geniş dağıtılmış dosya sistemlerini içerdiği için zor bir iştir. Map Reduce, Büyük Verilerin verimli analizi için yaygın olarak kullanılmaktadır. Birleşmeler ve dizin oluşturma, grafik arama gibi diğer teknikler gibi geleneksel DBMS teknikleri, Büyük Verilerin sınıflandırılması ve kümelenmesi için kullanılır. Bu teknikler Map Reduce'da kullanılmak üzere benimsenmektedir. Map Reduce, eşleme, sıralama, karıştırma ve nihai azaltma ile dosya indekslemesini kullanan bir minimizasyon tekniğidir.

Terabayt ve petabayt ölçekli veri setleri hızla yaygınlaşmakla birlikte, aynı zamanda, doğru hesaplama araçları ile kilidini açmayı bekleyen, büyük bir değerin gömülü

olduğu konusunda fikir birliği mevcuttur. Map Reduce, Hadoop ve HDFS gibi Büyük veri analiz araçları, kurumların müşterilerini ve pazarını daha iyi anlamakta, böylelikle daha iyi iş kararları ve rekabet avantajları sağlamasına yardımcı olmayı taahhüt etmektedir. Bilgi işleme araçları ve uygulamaları oluşturan mühendisler için, sürekli veri akışı üreten geniş ve heterojen veri kümeleri, makine çeviriden spam algılamaya kadar geniş bir yelpazede görevler için daha etkili algoritmaların tasarlanmasını sağlar.

Map Reduce tek başına büyük dağıtık veri setlerini analiz etme yeteneğine sahiptir; ancak Büyük Veri'nin heterojenliği, hızı ve hacmi nedeniyle geleneksel veri analizi ve yönetimi araçları için bir zorluktur. Ayrıca, web ölçekli veriler evrensel değildir ve heterojendir. Büyük Verilerin analizi için veritabanı entegrasyonu ve temizlenmesi geleneksel madencilik yaklaşımlarından çok daha zordur.

Veri ambarı, raporlama ve çevrimiçi analitik işleme (OLAP) konusundaki geleneksel deneyim, gelişmiş analitik formlar için farklıdır. Kuruluşlar, özellikle ileri analiz olarak adlandırılan belirli analiz yöntemlerini uygulamaktadırlar. Bunlar, genellikle tahmin analizi, veri madenciliği, istatistiksel analiz, karmaşık SQL, veri görselleştirme, yapay zeka, doğal dil işleme gibi ilgili teknikler ve araç türleri topluluğudur. MapReduce, veritabanı içi analitik, in-memory veritabanları ve sütunlu veri depoları gibi veritabanı analitiği platformları standartlaştırmak için kullanılır. Araştırmacılar ve akademisyenler için zor olan, geniş veri kümelerinin özel işleme sistemlerine ihtiyaç duymasıdır.

HDFS üzerinde yer alan Map Reduce, Veri Bilimcilerine Büyük Verilerin analizinin yapılabileceği teknikleri verir. HDFS, orijinal Google Dosya Sistemi'ni kapsayan bir dağıtılmış dosya sistemi mimarisi olup Map Reduce işlemi, MapReduce'un her aşamasında uygulanabilecek eşleme, karıştırma, dizin oluşturma, gruplama ve azaltma gibi verimli veri işleme teknikleri kullanır.

Günümüzde çok fazla veri doğası gereği yapılandırılmış formatta değildir. Görüntüler ve videolar görüntü için yapılandırılmış ancak semantik içeriğe sahip ve arama için yapılandırılmamışken, tweet'ler ve bloglar zayıf yapılı metin parçalarıdır:

bu tür içeriği daha sonraki analizler için yapılandırılmış bir biçime dönüştürmek büyük bir zorluktur.

Hadoop ve HDFS çerçevesinin birer parçası olan Hive, Pig ve Mahout gibi araçların, verilerin hızını ve heterojenliğini ele alan, ön tanımlı ve kesin veriler kullanılan Büyük Veri analiz araçlarının bir görüntüsü verilmektedir. Kullanılan tüm araçlar için HDFS üzerindeki Hadoop temel mimaridir. Flume ve Zookeeper ile Oozie ve EMR, standart Büyük Veri yönetim araçları olan verilerin hacmini ve doğruluğunu işleme koymak için kullanılır.

Büyük ölçekli veri işlemleri zor bir iştir, yüzlerce veya binlerce işlemci yönetmek ve paralelleştirme ve dağıtılmış ortamları yönetmek daha zordur. Nandimath vd. (2013), makalelerinde, Map Reduce, dağıtılmış ve paralel G/Ç zamanlamasını destekleyen hataya dayanıklı ve ölçeklenebilirliği destekleyen ve durum için dahili süreçlere ve Büyük Veri'deki heterojen ve büyük veri kümelerinin izlenmesine sahip olduğu belirtilen sorunlara çözüm sunan bir yapı olduğu incelenmiştir [33].

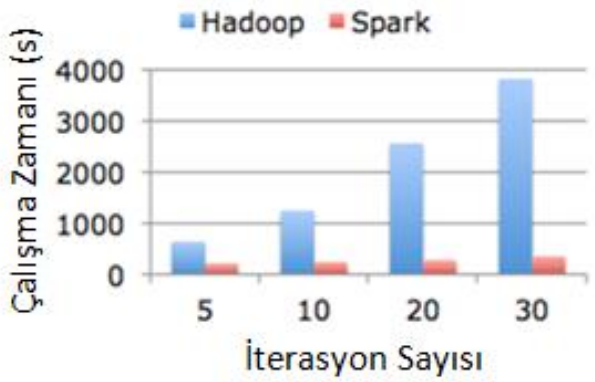
BÖLÜM 4

APACHE SPARK TEKNOLOJİSİ

İlk olarak Kaliforniya Üniversitesi Berkeley AMPLab laboratuvarında geliştirilen Spark çekirdek sürümü daha sonra Apache Software desteği alarak Apache yazılım firmasına bağışlanmıştır. Needell ve Warner (2014), çalışmalarında Spark'ın tüm kümeleri örtük veri paralelleştirme ve hata toleransıyla programlamak için bir arayüz sağladığı belirtilmiştir [34]. Apache Spark dil bağımlılığını en aza indiren bir yapı kullanmaktadır. Temelde hız faktörünü en yükseğe çıkarmak için Java ile yazılmış olan Spark mimarisi R, Scala ve Python dillerini desteklemektedir. Analiz esnasında çekirdek yapıya sahip olan Java, Python dilinden iki kat süre tasarrufu yaparak daha hızlı bir analiz süresi vermekle beraber kod yazım maliyeti olarak Python'ın gerisinde kalmaktadır. Çalışma kapsamında kullanılan MLlib kütüphanesi R ve Scala ile oluşturulmuştur. Spark MLlib kütüphanesi, dağıtık bellek tabanlı Spark mimarisi sayesinde çoğunlukla Apache Mahout tarafından kullanılan disk tabanlı uygulamaya göre dokuz kat daha hızlı olan Spark çekirdeği üzerine geliştirilmiş makine öğrenmesi çerçevesi analizlerini yapar. Michael (1997), çalışmasında makine öğrenmesi algoritmalarının doğru ve hızlı bir şekilde analiz edilmesi gerektiğini belirtir [35]. Spark, atası olan Hadoop mimarisine ait HDFS'e göre yüz kata kadar daha hızlı kümeleme işlemi yapmaktadır. Bu sayede verilerin analiz ortamlarına adaptasyonunda zaman tasarrufu sağlamakla beraber büyük veri analizi olarak adlandırılan çok büyük ölçekte analiz işlemlerinde kaynak kullanımını en iyi seviyeye çekmektedir [36].

Apache Spark, Berkeley Üniversitesinde AMPLab ortamında geliştirilen Hadoop veri işleme platformuna benzer hızlı ve genel amaçlı bir küme bilgi işlem sistemidir. Hadoop'a benzer şekilde, Spark Hadoop Dağıtık dosya sistemi üzerine kurulmuştur. Ancak Hadoop'un aksine, döngüsel bir veriye bağlı değildir. Bunun yerine, Spark bir

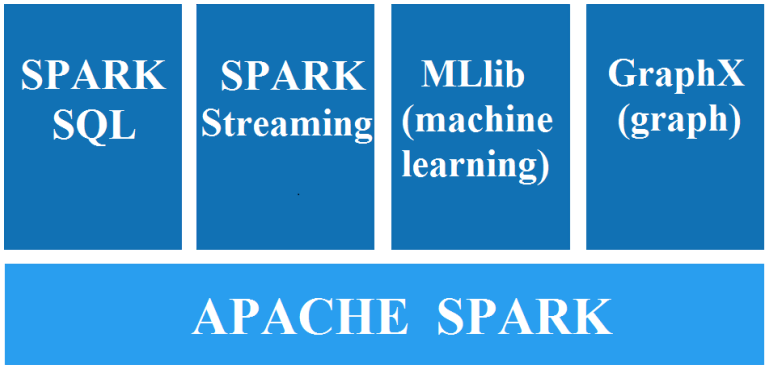
döngüsel veri kullanır. Algoritmanın her aşamasında, önbelleğe alınan verilere doğrudan erişilebilir. Sistemden veya ana düğümden defalarca okumak zorunda kalmaz [34]. Şekil 4.1'de Hadoop ve Spark'ın Lojistik Regresyon çalışma zamanı karşılaştırılması gösterilmiştir.



Şekil 4.1. Apache Spark ve Hadoop Lojistik Regresyon karşılaştırması.

Spark, RDD soyutlaması üzerine merkezlenmiş bir uygulamadır. Programlama arayüzü Java, Python, Scala ve R ile sunulmuştur. Tamamıyla bağımsız olarak çalışabilmekte, bilgisayarda yüklü olan Java Sanal Makinası (Java Virtual Machine, kısaca JVM) üzerinden işlem yapmaktadır. Matei vd. (2012), çalışmalarında Spark'ın Hadoop mimarisi ile karşılaştırılınca herhangi bir platform bağımlılığı olmadığı belirtilmiştir [37]. Hadoop veri analizi işlemleri için HDFS saklama biriminden ve MapReduce programlama modeli olan bir işleme bölümünden oluşur. Hadoop okuduğu dosyaları bloklara böler ve disk üzerine yazar ya da düğümler üzerine dağıtır. Daha sonra paketlenmiş veriyi tekrar okuyarak işlemeye başlar. Hadoop temelde paralel programlama mimarisini kullanarak hız kazanmaktadır ancak işlemlerini HDFS üzerinden diske yazıp tekrar okuyarak analiz eder. Analiz bittikten sonra tekrar diske yazar ve sonuçları verir [38]. Hadoop dağıtılmış bir dosyalama sistemi sunarken Spark bundan farklı olarak bellek içi veri işleme olanağı tanır. Spark'ın sistem belleği üzerinden işlem yürütme yaklaşımı neredeyse yüz kata kadar hız artışı sağlamakta ve analistin sonuçlara ulaşımını hızlandırmaktadır. Ancak arzu edilen büyük veri senaryosu Hadoop ve Spark'ın beraber çalışabilme ortamıdır. Bellekte yetersiz alan olması yani taşma durumunda Spark taşan veriyi bölerek HDFS haline getirmekte ve sistemin geçici klasöründe saklamaktadır. Sonuca

ulaştıktan sonra geçici klasördeki dosyaları silerek analist için dosya alanı açmaktadır. Şekil 4.2.'de Apache Spark'ı oluşturan kütüphaneler gösterilmektedir.



Şekil 4.2. Apache Spark bileşenleri.

Apache Spark, Hadoop sistemine karşı geliştirilmiş bir yapı değil aksine Hadoop sisteminden daha fazla fayda sağlanması amacına yönelik geliştirilmiştir. Apache Spark'ı Hadoop sisteminden bağımsız düşünemeyiz. Çünkü Spark Hadoop'un HDFS dosya yapısını üzerinde çalışmaktadır.

Spark SQL: Yapısal veriler için kullanılan Spark bileşeni olup ayrıca SQL ve Hibernate sorgulama dili (HQL) dili ile Parquet, Json, Hive tabloları sorgulanabilir. Komplex analitik işlemleri RDD'ler ile desteklenen operasyonlar ile kolayca geliştirilebilir.

Spark Streaming: Sürekli akan verileri işlemek için kullanılan Spark bileşenidir. RDD'ler disk ile bellekte bulunan veriler ile anlık akan verileri birlikte işlenebilir. Özellikle sosyal medya verileri bu kütüphane ile işlenebilmektedir. Spark motoru gelen verileri işlemek için verileri mikro yığın halinde gruplar. Bu, ayrık bir akımdır (DStream) ki, RDD setini temsil eder.

MLlib: Sınıflandırma, kümeleme, regresyon gibi makine öğrenmesi algoritmalarını içeren Spark bileşenidir. Makine öğrenme algoritmaları dağıtık çalışabilecek şekilde planlanmış ve optimize edilmiştir.

GraphX: RDD'lerin yönlü graflara dönüştürülüp işlenmesini sağlar. Paralel olarak işlenebilme özelliğine sahip bu kütüphane grafı oluşturan nokta ve bağlantılara farklı özellikler tanımlayabilir.

MLlib: Sınıflandırma, kümeleme, regresyon gibi makine öğrenmesi algoritmalarını içeren Spark bileşenidir. Makine öğrenme algoritmaları dağıtık çalışabilecek şekilde planlanmış ve optimize edilmiştir.

Apache Spark ölçeklenebilir veri işleme için genel bir küme hesaplama altyapısıdır. Hata toleransı ve emtia donanımında çalışacak şekilde tasarlanmıştır. İki aşamalı Map/Reduce ile işlemler arasında hızlı veri paylaşımını destekler. Spark, Amazon S3, Hadoop HDFS ile herhangi bir POSIX dosya sistemi ile uyumludur. Xin vd. (2014), makalelerinde Spark'ın, in-memory özelliği ile iş yüklerini işleme yeteneğine sahip olması belirtilmiştir [39].

Hata toleransı, Yönlendirilmiş düz ağaç grafiği (DAG) yapısına sahiptir. Önemli ölçüde kullanılan okuma/yazma işlemlerinin sayısını azaltıp MapReduce işleminde daha fazla zaman verimliliği sağlar. Spark, HDFS gibi Hadoop elemanları ile uyumludur. Hive ve YARN kullanarak bir Hadoop kümesinin üzerinde çalışabilir. MLlib, Spark Scala'da yazılmıştır ve Scala, Java, SQL, Python ve R yoluyla da kullanılabilir.

Apache Spark, büyük ölçekli veri işleme için popüler bir açık kaynak platformu olup tekrarlı makine öğrenme uygulamaları için çok uygundur. Spark'ın açık kaynaklı dağıtık makine öğrenme kütüphanesi MLlib, geniş bir öğrenim ayarları yelpazesi ve istatistiksel, optimizasyon ile lineer cebirsel özellikler sağlaması işlevsellik özelikleri arasındadır. Spark ile birlikte gelen MLlib, birçok dili desteklemekte ve geliştirmeyi basitleştirmek için Spark'ın zengin ekosistemini güçlendiren üst düzey bir end-to-end pipelines API sağlamaktadır. Meng vd. (2015), çalışmalarında, Mllib katkıda bulunanlarla hızlı bir şekilde büyümekte ve daha fazla büyümeyi desteklemek ve kullanıcıların hızlı bir şekilde hızlanmalarına olanak tanımak için canlı, açık kaynaklı bir topluluk ile geniş dokümanlar içerdiği belirlenmiştir [40].

Son yıllarda Büyük Veri kavramının çok dikkat çekmesi nedeniyle büyük verileri analiz etmek bugünün ortak gereksinimi haline gelmiştir. Toplu verilerin analiz edilmesi kolay bir işlem değildir. Özellikle twitter gibi sürekli akan veri kaynakları gibi büyük miktardaki verileri analiz etmek için büyük bir mücadele gerekir.

Apache Spark, en yaygın kullanılan açık kaynak yapılarından biri olup büyük veri için, zengin dillere entegre edilmiş işleme motorlarına sahip API'ler ve geniş kütüphaneler barındırır. Spark'ın performans kullanımı geliştirilmeye devam etmektedir. Kullanılabilirlik tarafında, Spark'ın geniş veri analizi algoritmalarının ölçeklenebilir sürümlerini içeren standart kütüphane kümeleri geliştirilip artmaktadır. Spark'ın makine öğrenme kitaplığı MLlib, önceki yıllara göre geliştirilmiştir. Harici veri kaynaklarına erişimi kolaylaştıran API'ler geliştirilmiştir. Armbrust vd. (2015), çalışmlarında Spark'ın performans ve kullanabilirliği incelenmiştir [41].

Grafik işleme uygulamalarının performansının iyi olması gerekir. Grafik verilerinin veri akışını sağlayan sistemlerin hızlandırılmasını sağlayan algoritmalar tercih edilmektedir. Apache Spark'ın sunduğu grafik işleme kütüphanesi olan GraphX incelenip dağıtık veri akış sistemini kullanan bu kütüphanenin mevcut grafik API'leri ile düşük maliyet gerektirdiği görülmüştür. Gonzalez vd. (2014), çalışmlarında grafik algoritmaları için, GraphX bir temel veri akışı sisteminden daha hızlı olduğu görülmüştür [42].

MLlib, makine öğrenme algoritmalarını barındıran açık kaynaklı ve geniş kapsamlı bir kütüphanedir. Spark, bellek içi dağıtık veri işleme motorudur ve büyük veri kullanımları için hızlı bir popülerlik kazanmıştır. Spark toplu işlem ve akan veriyi işlemeye olanak tanır. Her iki paradigmayı kullanarak makine öğrenimi gerçekleştirilebilir. MLlib ayrıca birçok matematiksel ve istatistiksel yöntemler ile veri ön işleme ve model değerlendirmeyi destekler. Birçok kullanıma hazır sınıflandırma, regresyon, öneri, kümeleme ve boyut azalma modelini sunar. Spark ve MLlib, Mahout'la kıyaslandığında oldukça hızlı olup bu algoritmaları uygulayan ve değerlendiren akademik çalışmaların sayısında artış görülmektedir.

MLlib kütüphanesi aşağıdaki algoritma ve yardımcı uygulamaları içermektedir:

1. Lojistik regresyon ve doğrusal destek vektör makinesi (SVM)
2. Sınıflandırma ve regresyon ağacı
3. Rasgele orman ve gradyanı güçlendirilmiş ağaçlar
4. Alternatif en küçük karelerle (ALS) öneri
5. K-means kümeleme, Bisecting k-means, Gauss karışımları (GMM) ve kuvvetli iterasyon kümelendirmesi
6. Latent Dirichlet ayırma (LDA) yoluyla konu modellemesi
7. Hızlandırılmış arıza süresi modeliyle hayatta kalma analizi
8. Tekil değer ayrışma (SVD) ve QR ayrışması
9. Temel bileşen analizi (PCA)
10. L1, L2 ile lineer regresyon ile elastik ağ düzenleştirme
11. İzotonik regresyon
12. Multinomiyal / binomiyal Naive Bayes
13. FP-büyüme ve birliktelik kuralları ile araştırma madenciliği
14. PrefixSpan ile ardışık desen incelemesi
15. Özet istatistikler ve hipotez testleri
16. Özellik dönüşümleri
17. Model değerlendirmesi ve hiper parametre ayarı [43].

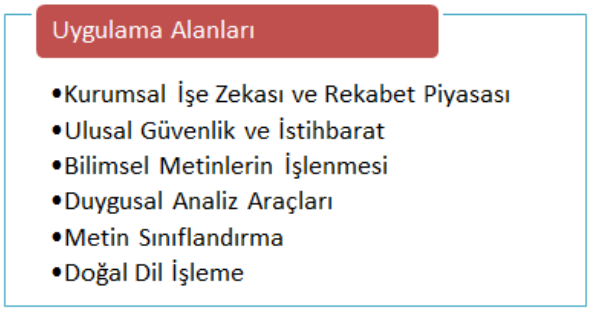
BÖLÜM 5

BÜYÜK VERİNİN APACHE SPARK İLE ANALİZİ

Bu bölümde Apache Spark teknolojisiyle büyük veri üzerinde sınıflandırma kümeleme ve aykırı değer tespiti uygulamaları yapılmıştır. Apache Spark'ın MLlib kütüphanesi kullanılarak büyük veri üzerinde uygulanan Naive Bayes, K-means ve Gaussian Mixture algoritmalarının sonuçları incelenmiştir.

5.1. DENETİMLİ MAKİNE ÖĞRENMESİ YÖNTEMİ NAİVE BAYES İLE APACHE SPARK ORTAMINDA TEXT SINIFLANDIRMA

Veri madenciliğinin önemli bir alanı olan metin madenciliği Apache Spark'ın makine öğrenmesi algoritmaları ile kullanımı oldukça geniş bir alana ulaşmıştır. Apache Spark'ın MLlib kütüphanesinin sağladığı denetimli ve denetimsiz öğrenme mimarisi ile verilerin analiz süresi ve tutarlılığı arttırılmıştır. Spark'ın genel kullanım alanları Şekil 5.1'de gösterilmektedir.



Şekil 5.1. Spark'ın Metin madenciliğindeki kullanım alanları.

Metin madenciliği, metinlerden verimli ve yüksek kalitede anlamların çıkarımı için kullanılmaktadır. Robbs vd. (1982), makalelerinde metinlerden yüksek kalitede bilgi

çıkarımı genel olarak istatistiksel olarak desen tarama ve makine öğrenmesi gibi uygulamalardan geçerek desenlerin oluşturduğu eğilim ile sağlandığı belirlenmiştir [44].

Metin madenciliğinde ilk aşama uygulamanın üzerinde çalıştırılacak olan verinin düzenlenmesi ile başlar. Çoğunlukla türemiş dil özelliklerinin temizlenmesinin yanısıra anlam çıkarılabilecek kelimelerin analiz ortamına uygun veri tabanı ya da kullanılan çerçeve yapısının depolama alanına eklenir. Cohen ve Hunter (2008), yaptıkları metin analizi sırasında belirli kalıplar türetebilmek için algoritmadaki eğitim setleri oluşturmuşlardır [45]. Bu setler, analizi yapılacak olan metin veya metinlerden belirli bir anlam çıkarma işleminde kullanılacak olan indekslerdir. Mohri vd. (2012), çalışmalarında metin madenciliğinde genellikle kullanılan yaklaşım denetimli öğrenme yöntemi olduğunu belirtiler [46].

Denetimli öğrenmede yönteminde eğitim verileri bir dizi eğitilecek olan kelime gruplarının örneğinden oluşur. Brodely ve Friedl (1999), çalışmalarındaki örneklerden her biri, bir giriş nesnesi (vektör) ve bir çıkış nesnesi (denetimli sonuç sinyali) olarak işlem görür [47]. Zaharia vd. (2010), çalışmalarınad bir denetimli öğrenmede, eğitim verileri analiz edilir ve yeni örneklerdeki verilerin haritalanmasında kullanılacak olan bir çıkarım fonksiyonu üretiğini belirttiler [48]. Beklenen durumda makine öğrenmesi algoritmasının eğitim verilerini etiketleme sırasında, örneklerin sınıf etiketlerini doğrusal bir şekilde belirlenmesine izin verir. Vapnik (2000), çalışmasında denetimli makine öğrenme algoritmasının öngörülemeyen durumlar için makul bir şekilde genelleştirme yapı olanak tanındığı incelenmiştir [49]. Duygu ve metin sınıflandırma analizi uygulamalarında yaygın olarak metin madenciliği ve makine öğrenmesi algoritmaları kullanılır. Veri madenciliği alanındaki en önemli gelişme Apache Spark teknoloji mimarisinin ortaya çıkışı olmuştur. Apache Spark açık kaynak kodlu paralel küme işleme çerçevesidir.

Sınıflandırma projesi için Carnegie Mellon University resmi sitesinde yer alan açık kaynak olarak dağıtılan metin öğrenme grubu veri arşivleri kullanılmıştır [50]. Bu veri seti, 8 farklı ağ haber grubundan toplanan yaklaşık 8.000 mesaj grubundan oluşmaktadır. Büyük veri boyutu sıkıştırılmış halde 7,85 MB gibi bir boyutta iken

açıldıktan sonra 28,4 MB boyutana ulaşmaktadır. Çizelge 5.1' de veri seti hakkında genel bilgiler gösterilmiştir. Apache Spark'ı bir Java-Maven tabanlı projede çalıştırmak için pom.xml ismindeki genişletilebilir işaretleme dili dosyasındaki ayarlar ile Maven online kütüphanelerinden Spark 2.10 Java arşiv dosyaları projeye eklenerek analiz ortamı oluşturulur.

Çizelge 5.1. Veri seti özellikleri.

Alan Adı	**Veri Dosyası Adedi**	**Boyut Aralıkları**
sci.crypt	1000	61 KB – 1 KB
soc.religion.christian	1000	19 KB – 1 KB
alt.atheism	1000	51 KB – 1KB
sci.space	1000	40 KB – 1KB
talk.politics.mideast	997	62 KB - 1KB
comp.windows.x	1000	66 KB – 1 KB
sci.med	1000	38 KB – 1 KB
comp.graphics	1000	61 KB – 1 KB

Oluşturulan analiz ortamında metin sınıflandırma için makine öğrenmesi algoritmalarından Naive Bayes algoritması kullanılmıştır. Naive Bayes algoritması makine öğrenmesinin denetimli öğrenme tabanına dayanmaktadır. Her kategoriden alınan 200 adet veri ile test verisi oluşturuldu. Dışarıdan alınan analizi yapılacak olan veri Java<RDD> haline getirilerek analiz için sistem belleğine yüklenir. Veri sistem belleğine yüklendikten sonra boşluk karakteri referans alınarak cümleler veri haritalandırma yöntemi ile kelimelere ayrılır. Ayrılan kelimeler yeni bir Java<RDD> dosyası haline getirildikten sonra analiz için saklanır. Girdi verisi haritalandırma (Map) işleminden sonra, metnin içeriğini sınıflandıracak olan anahtar kelimeler analiz ortamına dahil edilir ve etiketlendirilir. Haritalandırılan girdi verisi bir vektör yapısı haline getirilir. Vektör haline getirilen girdi verisi içinde Naive Bayes modeli ile anahtar kelimeler aranarak kategorilere göre yatkınlık oranları elde edilir. Naive Bayes sınıflandırma teoremi şu şekilde verilmektedir:

$$P(S_i)\prod_{k=1}^{L}P(X_k\,|S_i) > \quad P\left(S_j\right)\prod_{k=1}^{L}P(\,X_k\big|\,S_j\right) \tag{5.1}$$

$P(S_i)$ ve $P(S_j)$ i ve j sınıflarının öncel olasılıklarıdır. Elde olan veri kümesinden değerleri kolayca hesaplanabilir. Naive Bayes sınıflandırıcısının kullanım alanı her ne kadar kısıtlı görünse de yüksek boyutlu uzayda ve yeterli sayıda veriyle x'in (nicelik kümesi) bileşenlerinin istatistik olarak bağımsız olması koşulu esnetilerek başarılı sonuçlar elde edilebilir. Varsayımsal olarak $C = \{c_1, c_2, ..., c_m\}$ kümesi olduğunu düşünelim. Doküman dosyası olan $D = \{d_1, d_2, ... d_n\}$ kümesi için, $K = \{k_1, k_2, ..., k_s\}$ benzersiz kelimelere karşılık gelen K kümesindeki her bir sözcük bir kez D kümesi üzerinde gezdirilir ve olasılıksal sonucun C olması hesaplanarak aşağıda yazılan Bayes kuralı kullanılır.

$$P(c|d) = \frac{P(c).P(d|c)}{P(d)} \tag{5.2}$$

$P(d)$ bilinen sabit veri kümesinin değeridir. Naive Bayes modeli her bir kelime terimi W_k'nın bağımsız olarak "c" sınıfını oluşturduğu varsayılır. Dolayısıyla denklem

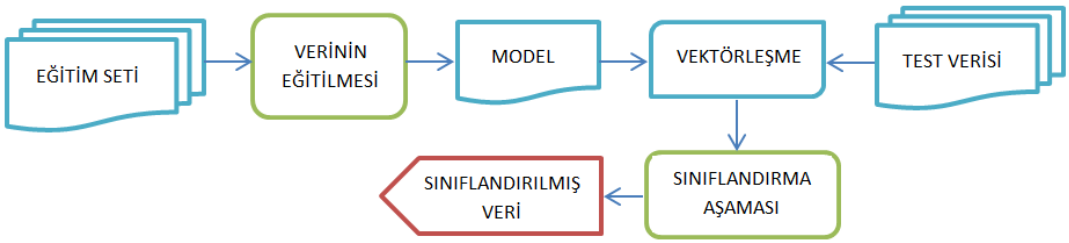
$$P(cP(c|d) \infty P(c) \prod_{k=1}^{n_d} [P(w_k|c]^{t_k} \tag{5.3}$$

n_d burada "d" dökümanı içindeki benzersiz kelimeleri belirtir ve t_k da her kelimenin frekansına karşılık gelmektedir. Navie Bayes algoritmasını kullanılırken $P(c)$ ve $P(w_k|\, c)$'yi

$$\widehat{P}(\mathrm{c}) = \frac{N_c}{N} \text{ ve } \hat{P}(w_k|c) = \frac{N_{w_k}}{\sum w_{i \in W} N_{wi}} \tag{5.4}$$

N doküman sayısını, N_c ise c sınıfı içindeki dokümanı ve N_{wi} ise kelime değişkeni olan W_i 'nin c içindeki frekansını verir [35].

Naive Bayes akış şeması aşağıda verilmektedir. Geliştirilen yöntemin sözde kodu ise şu şekildedir:



Şekil 5.2. Naive Bayes akış şeması.

```
Input: Raw Data
Output: Sınıflandırılma Kategorisi Oranları
1: function FlatMap <String>
2:         for all data
3:   end function
4:   function LabeledPoint(key, String)
5:         for all  training data
6:         Labeled_Points convert to JavaRDD<String>
7:   end function
8:   train model JavaRDD<String>
9:   predict = (pos prob > neg prob) ? pos : neg
10: if Predict : true
11:           Labeled_Count++
12: else
13:           return
14: end function
```

Deneysel araştırmalarda Intel i7 3.4 GHz 4 çekirdekli CPU ile kullanılmıştır. SSD ile hız faktörü arttırılmış, Spark'ın bellek üzerindeki analizinde yeterli alanı kullanabilmesi için 16 Gb bellek üzerinden 13 Gb bellek, JVM üzerine sunulmuştur. Sistemin kararlılığını test etmek için 5 aşamalı bir uygulama yapılmıştır. İlk uygulama Çizelge 5.2'de gösterilmiştir. 3 kategoriye ait 10 test verisi üzerinden doğru tahmin analiz sonuçları elde edildi.

Çizelge 5.2. 3 kategoriye ait 10 test verisi ile Spark analiz sonuçları.

	Kategori	Test verisi adedi	Doğru tahmin
1	alt.atheism tahmini	10	10
2	comp.windows.x tahmini	10	10
3	sci.space tahmini	10	10

İkinci uygulamanın sonuçları Çizelge 5.3.'de gösterilmiştir. İlk uygulama ile kıyaslandığında test veri sayısının sabit kalıp kategori sayının yani modeli oluşturan veri miktarındaki artışa bağlı olarak doğru tahmin sonuçları belirlendi.

Çizelge 5.3. 5 kategoriye ait 10 test verisi ile Spark analiz sonuçları.

	Kategori	Test verisi adedi	Doğru tahmin
1	alt.atheism tahmini	10	9
2	comp.windows.x tahmini	10	10
3	sci.space tahmini	10	10
4	talk.politics.mideast tahmini	10	10
5	sci.crypt tahmini	10	9

Üçüncü uygulamada ikinci uygulamada olduğu gibi test veri sayısı sabit tutulup kategori sayısı 8'e çıkarıldı. Çizelge 5.4' de gösterilen sonuçlara göre doğru tahmin sayısı belirlendi. Kategori sayısındaki artışa bağlı olarak test veriş sayısının sabit tutulması doğru tahmin olasılığını düşürdüğü belirlendi.

Çizelge 5.4. 8 kategoriye ait 10 test verisi ile Spark analiz sonuçları.

	Kategori	Test verisi adedi	Doğru tahmin
1	alt.atheism tahmini	10	8
2	comp.windows.x tahmini	10	10
3	sci.space tahmini	10	10
4	talk.politics.mideast tahmini	10	10
5	sci.crypt tahmini	10	9
6	sci.med tahmini	10	10
7	soc.religion.christian tahmini	10	10
8	comp.graphics tahmini	10	9

Dördüncü uygulamada test veri sayısı 200' e çıkartılıp doğru tahmin sayıları ve bu sayılara göre yüzdelik tahmin sonuçlar Çizelge 5.5'de gösterilmiştir. Doğru tahmin sonuçlarına göre test verisi sayısındaki artış doğru tahmin olasılığını artırdığı belirlendi.

Çizelge 5.5. 5 kategoriye ait 200 test verisi ile Spark analiz sonuçları.

	Kategori	**Test verisi adedi**	**Doğru tahmin**	**Tahmin Yüzdesi**
1	alt.atheism tahmini	200	193	96,5
2	sci.crypt tahmini	200	189	94,5
3	sci.space tahmini	200	198	99
4	soc.religion.christian tahmini	200	186	93
5	talk.politics.mideast tahmini	200	181	90,5

Çizelge 5.6' da kategori sayısı 8'e çıkartılıp test verisi 200 olarak yapılan çalışma sonucu gösterilmektedir. Genel olarak tahmin yüzde sonuçlarına göre kategori sayısının artışı doğru tahmin olasılığını azaltmıştır.

Çizelge 5.6. 8 kategoriye ait 200 test verisi ile Spark analiz sonuçları.

	Kategori	**Test verisi adedi**	**Doğru tahmin**	**Tahmin Yüzdesi**
1	alt.atheism tahmini	200	185	92,5
2	sci.crypt tahmini	200	184	92
3	sci.space tahmini	200	183	91,3
4	soc.religion.christian tahmini	200	184	92
5	talk.politics.mideast tahmini	200	166	83
6	comp.windows.x tahmini	200	155	77,5
7	comp.graphics tahmini	200	167	83,5
8	sci.med tahmini	200	165	82,5

Yapılan analiz veri seti üzerindeki 3, 5 ve 8 farklı kategori belirlenerek sorgulanmıştır. Ateizim içerikli kategori analizi, grafik içerikli kategori analizi, windows içerikli kategori analizi, şifreleme içerikli kategori analizi, tıp içerikli kategori analizi, uzay içerikli kategori analizi, Hristiyan içerikli kategori analizi, Ortadoğu içerikli kategori analizi kelimelerini içeren analiz uygulaması yapılmıştır. Farklı ve birbiriyle ilgili alanlar seçilerek analiz sonuçlarını nasıl etkilendiği

belirlendi. Uygulamalara göre seçilen kategoriler Naive Bayes ile etiketlendikten sonra bir model üzerinde eğitilerek veri kümeleri üzerinde işleme dahil edilerek tahmin oranları elde edilmiştir.

Bu çalışmada metin sınıflandırması için makine öğrenmesinde Naive Bayes yöntemi kullanılarak Apache Spark üzerinde hızlı metin sınıflandırması gerçekleştirilmiştir. Veri seti üzerinde terminolojilerden seçilen anahtar kelimelere göre kategori tahmini yapılmıştır.

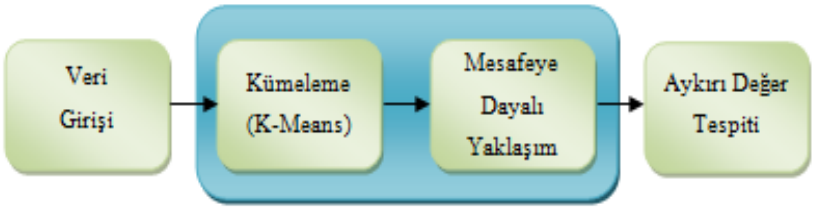
Veri boyutunun büyüklüğü performans ve sonucu etkin bir şekilde değiştirir. Makine öğrenmesi ile güçlendirilen Naive Bayes algoritması ile metin sınıflandırması etkin bir şekilde yapılmaktadır. Bu yöntem ile sınıflandırma, cinsiyet tahmini, istenmeyen mesaj analizi gibi olasılık hesapları yapılabilmektedir. Spark teknolojisi ile standart veri madenciliği geliştirilerek makine öğrenmesi algoritmaları ile işlemler paralel hale getirilmiştir. Çalışma süreleri ve analiz performansları gerçek metin verileri üzerinden gösterilmiştir.

5.2. DENETİMSİZ MAKİNE ÖĞRENMESİ YÖNTEMİ K-MEANS KÜMELEME İLE APACHE SPARK ORTAMINDA HIZLI VERİ KÜMELEME VE AYKIRI DEĞER ALGILAMA

Aykırı değer tespiti için farklı teknikler yıllar boyunca önerilmiştir. Anormallik tespitinin öne çıkması, verilerin anormalliklerinin birçok farklı uygulama alanında değerli bilgiler haline dönüşmesinden kaynaklanmaktadır. Chawla ve Gionis (2013), çalışmalarında K-means yönteminin, verileri kümelemek ve aykırı değerlerin bulunması için genelleştirilmiş olduğu belirtilmiştir [51]. Bu çalışmanın sonucu olarak, yöntemin sapma değerlerine karşı çok hassas olduğu bulundu. K-means kümeleme algoritması ile kredi kartı dolandırıcılığı algılama, Sabau (2012), makalesinde mali dolandırıcılık algılamaya [52], Issa ve Vasarhelyi (2011), makalelerinde dolandırıcılık tespitini geri ödeme gibi çalışmalar yer almıştır [53].

Etiketsiz bir veri kümesi üzerinde gerçekleştirebileceğimiz en basit görevlerden biri, veri kümesindeki birbirine benzeyen veri gruplarını bulmaktır. MacQueen (1967), çalışmasında K-aracı bilinen kümeleme problemini çözen en popüler ve en basit denetlenmeyen öğrenme algoritmalarından biri olduğunu belirtmiştir [54]. İki önemli varsayım, her girdinin yalnızca bir küme ait olabilmesi ve kullanıcı zaten kaç küme bulunduğunun bilinmesidir.

Belirli bir veri kümesinde yeni ve önemli bilgileri bulmak için kümeleme yöntemleri kullanılır. Kümeleme yöntemlerinde, K-means, en eski yöntemlerden biri olarak gösterilebilir. Bu yöntem, büyük veri setlerini işlemek için iyi bilinir ve etkilidir. Kümelenmeyi kullanarak, veri kümeleri bölünür ve dolandırıcılık verilerini bulmak için belirsizlik tespiti kullanılır. Bir aykırı değer, belirli bir veri kümesindeki genel bir kalıba uymayan beklenen bir davranışın bir kaydıdır. Bir boyutlu veride aykırı değer algılaması Şekil 5.3'te verilmiştir.



Şekil 5.3. Sistem mimarisi.

K-Means yöntemi sayısaldır, denetlenmez ve iteratif olarak işlem yapar. Denetlenmemiş teknik, bir veri kümesindeki gözlem sınıfı hakkında önceden bilgi bulunmaması durumunda yararlıdır. K-Means kümeleme yönteminde, n cisim k kümelerine bölünür ve her cisim kümelerin en yakın ortalamasına ait olur. Bu noktada, mümkün olan en büyük varyasyonla farklı kümeler oluşturulacaktır. En büyük mesafeyi sağlayan küme sayısı önceden bilinmediğinden, verilerden hesaplanması gerekir. Son olarak, bu algoritmanın amacı objektif bir işlevi en aza indirgemektir, bu durumda karesel bir hata işlevi. Amaç fonksiyonu:

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^{(j)} - c_j \right\|^2 \tag{5.5}$$

Burada k, kümelenme sayısı ve n sayısı vaka sayısıdır. $\left\|x_i^{(j)} - c_j\right\|^2$ veri noktası $x_i^{(j)}$ c_j ile küme cj'nin merkezi arasındaki seçilmiş bir uzaklık ölçüsüdür. Algoritma aşağıdaki basamakları içermektedir. K-Means işleminin ilk adımı K noktalarını kümelenmiş nesneler tarafından temsil edilen alana yerleştirmektir. Bu noktalar ilk küme merkezleri gibi davranır. İkinci adım, her veri noktasını en yakın ağırlık merkezine sahip olan gruba atamaktır. Üçüncü aşamada, K ağırlık merkezlerinin pozisyonları hesaplanır ve yeni bir bölüm oluşturulur. Daha sonra, merkezlerde herhangi bir değişiklik yapılıncaya kadar Adım 2 ve 3 tekrarlanır. Yinelemeli bir şekilde çalışır ve her iterasyonun sonunda küme merkezleri güncellenir. Jain (2010), makalesinde, K-Means algoritmasının temel amacı, tüm kümeler için kareli hata toplamını en aza indirgemek olduğunu incelemiştir [55]. K-Means algoritmasının sahte kodu aşağıda verilmiştir.

Input: Raw Data
Output: Kümeleme Kategorisi
$\vec{c}_i, \ldots, \vec{c}_k$ değerleri rastgele seçilmeli
1. $\vec{x}_1, \ldots, \vec{x}_n$
2.**repeat**
3.**for** $i = 1$ **to** n
4. $\gamma_{ij} = \begin{cases} 1 & \text{if } j = \text{argmin}_j \left\|\vec{x}_j - \vec{c}_j\right\|^2 \\ 0 & otherwise \end{cases}$
5.**end for**
6.**for** $j = 1$ **to** k
7.$n_j = \sum_{i=1}^{n} \gamma_{ij}$
8.$c_j = \frac{1}{n_j} \sum_{i=1}^{n} \gamma_{ij} \vec{x}_i$
9.**end for**
10.**until convergence**
11.Return $\vec{c}_1, \ldots, \vec{c}_k$

Başlangıçta kümelenme sayısı bilinmediğinden ampirik olarak tahmin edilebilir. Bu durumu çözmek için, farklı küme sayıları için çoklu çalışma sonuçları karşılaştırılabilir ve daha sonra en iyi olanı önceden tanımlanmış bir kritere dayalı olarak seçilir. Büyük bir kümenin muhtemelen hatayı azaltacağı, ancak gereğinden fazla uyma riskini artıracağı görülecektir. Dunham (2003), çalışmasında K-Means algoritmasının, her zaman hedef fonksiyona karşılık gelen en uygun sonucu bulamadığını tespit etmiştir [56].

Üst düzey kümeleme programlama modellerinin geliştirilmesi, K-aracı yöntemlerinin paralel uygulamalarının ortaya çıkmasına yol açmıştır. Spark'ın tercih edilmesinin nedeni K-means gibi yinelemeli algoritmaların paralelleştirilmesi için çok uygun olmasıdır. Spark'da, MLlib kütüphanesi altındaki K-means de dahil olmak üzere çeşitli makine öğrenme algoritmaları geliştirilmiştir. Dolayısıyla, problemimizi çözmek için etkili bir kümeleme çerçevesi kullanılır.

Spark teknolojisi, büyük verilerin ayrıntılı olarak işlenmesi ve analizi için bir makine öğrenme kütüphanesi sağlar. MLlib, kümeleme, sınıflandırma, regresyon, boyut azaltma, eğitim ve diğer istatistiksel uygulamalar için yüksek kaliteli algoritmalar içerir. MLlib yaklaşık olarak bellekte Hadoop MapReduce'dan 100 kat daha hızlı, diskte 10 kat daha hızlı programları çalıştırabilir. K-Means yönteminin paralelleştirilmiş bir versiyonu olan Spark MLlib uygulamasına dahil edilmiştir. Çalışmamızda bu algoritmayı kullanarak analizler yapıldı. K-Means MLlib algoritması Çizelge 5.3'de verilen bazı parametrelere sahiptir.

Çizelge 5.7. MLlib içerisindeki K-means parametreleri.

Parametreler	Açıklamalar
k	İstenen küme sayısı
maxIterations	Çalıştırılacak maksimum yineleme sayısı
initializationMode	K-aracıyla rastgele başlatma
runs	Spark 2.0.0'dan bu yana hiçbir etki yok
initializationSteps	K-means algoritmasında adım sayısı
epsilon	Mesafe eşiği içerisinde k-ortalamasının yakınsak olduğu düşünülür
initialModel	Başlatma için kullanılan isteğe bağlı küme merkezleri seti

Deneysel çalışmalarda, Intel i7 3.4 GHz 4 çekirdekli 8 GB RAM'e sahip bir bilgisayar kullanıldı. Hız faktörü SSD ile artırılmış ve 16 GB bellekte 13 GB bellek JVM üzerinde sunulmuştur, böylece Spark bellek analizi için yeterli alan kullanıldı. Öncelikli adım olan kümeleme işlemi için küme sayısı ve iterasyon sayısı sırasıyla 10 ve 20 olarak belirlenir ve seçilen K-Means algoritması yürütülür. Kümeler

tanımlandıktan sonra, uzaklık belirleme aykırılık tespiti için gerçekleştirildi. Aykırı algılamanın sonucu, elektrik tüketiminde anlık bir sorunun saptanması mümkündür.

Örnek veriler, çeşitli veri kümeleri sağlayan UCI makine öğrenme havuzundan alınan bireysel ev elektriğine ait güç tüketimi verisi olup Lichman (2013) tarafından yüklenmiştir [57]. Bu veri kümesi, kümeleme ve regresyon için kullanılabilir. Öznitelik sayısı dokuz, örnek sayısı 2075259'dur. Veri setini uygulamaya hazır hale getirmek için bir ön çalışma uygulanmıştır. Ön çalışma, doğru sonuçlara ulaşabilmek için sensörlerden alınan kirli verilerin temizlenmesi işlemidir. Böylece hata oranı ciddi derecede azaltılmaktadır. Eksik ve hatalı verilerin ayrıştırılmasından sonra veri seti uygulamaya hazır hale getirilip Apache Spark MLlib kütüphane komutları üzerinde işlenebilir hale getirilmiştir. İlk iki sütun tarih ve saat alanları işleme alınmayıp sadece sensör verileri üzerinde kümeleme çalışması yapılmıştır Veri kümesinin gerçek değer öznitelikleri vardır ve eksik değerleri vardır. Bu değerleri ortadan kaldırmak için veri kümesi önceden işleme konarak toplam 25979 girdi silinir. Bu noktada daha kapsamlı bir çalışma yapılmıştır.

Büyük Veri üzerinde Spark teknolojisi kullanılarak veri kümeleme ve aykırı değer algılama işlemi daha hızlı bir şekilde gerçekleştiği sonucuna varılmıştır. Deney sonuçları, Spark MLlib'de uygulanan K-Means algoritmasını kullandıktan sonra, aykırı değer algılamanın başarıyla gerçekleştirildiği kanıtlanmıştır. Veriler kümelendikten sonra, aykırı değerlerin tespiti bireysel ev elektriğine ait güç tüketimi veri seti üzerinde etkili bir şekilde uygulanmıştır. Kütüphanenin kullanım kolaylığı, farklı boyutlardaki veri setlerine uyarlanabilmesi ve veri işleme hızının dikkate alınmasıyla Apache Spark Teknolojisinin büyük veri alanında önemli bir değer taşıdığı görülmektedir.

Gauss Mixtures Modeli (GMM), Gauss bileşen yoğunluklarının ağırlıklı bir toplamı olarak gösterilen bir parametrik olasılık yoğunluk fonksiyonudur. GMM'ler yaygın olarak, konuşmacı tanıma sistemindeki ses yolu ile ilgili spektral özellikler gibi biyometrik bir sistemdeki sürekli ölçümlerin veya özelliklerin dağılma ihtimalinin parametrik bir modeli olarak kullanılır. GMM parametreleri, eğitimli bir önceki modelden yinelenen Beklenti Maksimizasyonu (EM) algoritması veya Maksimum A

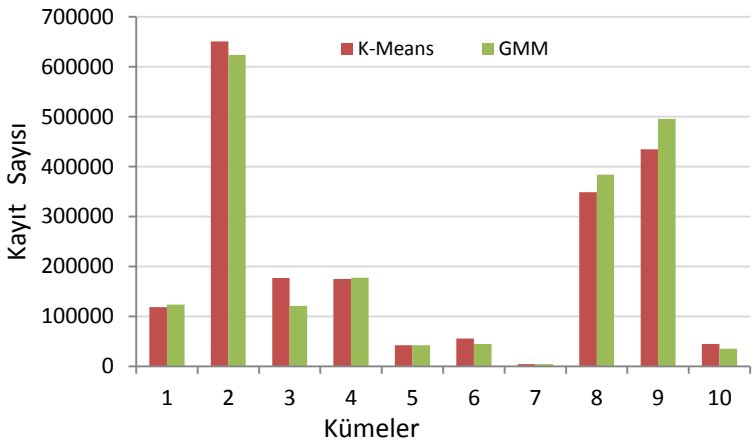
Posterior (MAP) tahminini kullanarak eğitim verilerinden tahmin edilir. GMM'deki M bileşeni Gauss yoğunluklarının ağırlıklı toplamıdır ve aşağıdaki denklemle verilir.

$$P(x|\lambda)f(x) = \sum_{i=1}^{M} w_i \; g(x|\mu_i,\sum_i) \tag{5.6}$$

Burada x, D-boyutlu sürekli değerli veri vektörüdür (yani ölçüm veya özellikler), wi, i = 1, ..., M, karışım ağırlıklarıdır ve g (x | μi, Σi), i = 1, ..., M, bileşen Gauss yoğunluklarıdır. Her bileşen yoğunluğu, formun D-değişkenli Gauss fonksiyonudur. Reynolds (2015), makalesinde Gaussian Mixture modelini inceleyip fonksiyonların işlevleri belirtilmiştir [58].

Gauss Karışım Modellerinin verinin etrafında yumuşak kümeleme sınırları oluşturmak mümkündür, çünkü bir karışım modelinde her numunenin yalnızca bir ihtimal dahilinde bir kümeye ait olduğu söylenmektedir.

GMM'lerin temel dezavantajı, K-Means'taki gibi Gauss karışım bileşenlerinin sayısının önceden olduğu varsayıldığı için tamamen denetlenmeyen kümeleme yöntemi olarak kabul edilemeyeceğidir. Bu duruma yardımcı olmak için Minimum Mesaj Uzunluğu (MML) ölçütleri gibi ek algoritmalar kullanılabilir olduğunu belirtmiştir [59].



Şekil 5.4. K-means ve GMM kümeleme sonuçları.

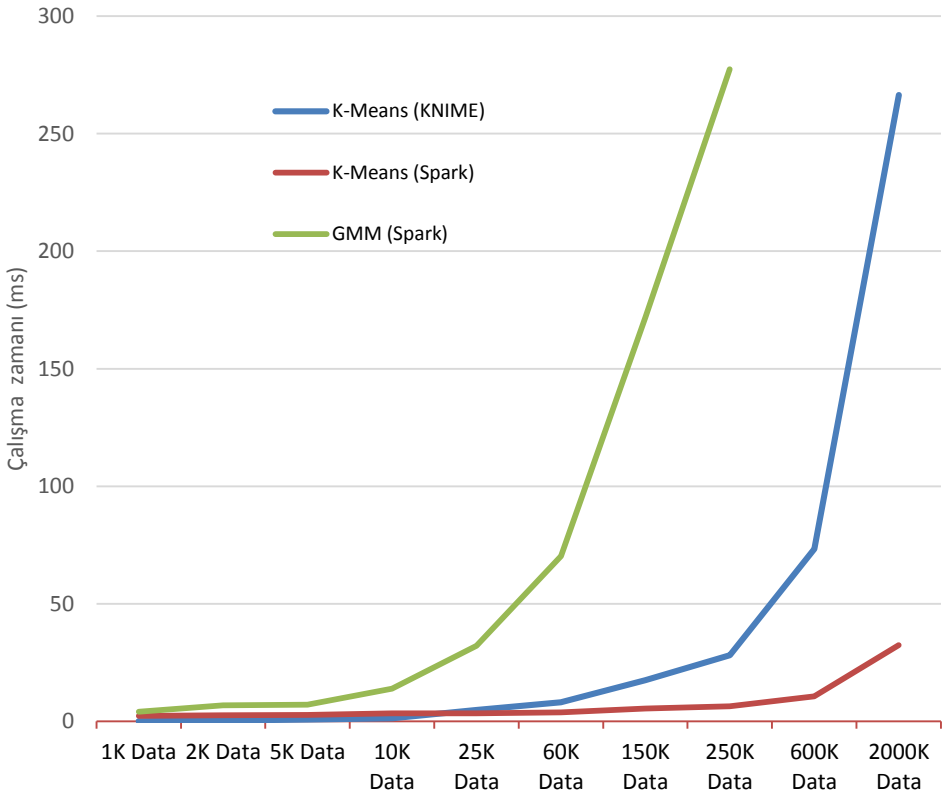
MLlib'de de uygulanan Gauss Karışım Modeli (GMM), K-means ortalamasının üstünlüğünü ortaya çıkarmak için de kullanılır. GMM, her bir noktanın kendi muhtemel Gauss alt dağılımlarından birinden alındığı bir bileşik dağılımı sağlar. K-means ve GMM kullanılarak kümeleşme sonuçları Çizelge 5.4'de verilmektedir. Sonuçlara göre, bu iki yöntemin sonucunda benzer kümelerin oluştuğu görülmektedir. Küme bilgileri ayrıntılı olarak incelendiğinde, evdeki güç tüketimi için farklı desenlerin fark edilebileceği saptanmıştır.

Kümeleme işlemini uyguladıktan sonra, veri kümesindeki aykırı değerlerin saptanması için bir analiz yapıldı. Mesafeye dayalı yaklaşımı kullanarak; Arzu edilen aykırı değer verileri tespit edilebilir. Çizelge 5.4, örnek veri kümesi üzerinde K-means ve GMM kullanılarak saptanan 10 outlier verilerinin değerlerini göstermektedir. Bu satırlardan bazılarının aynı olması, her iki yöntemin de aynı kayıtları aykırı değerlerle algıladığını gösterir. K-Means Spark, K-Means KNIME ve GMM Spark'ın farklı veri boyutlarına göre çalışma süreleri Şekil 5.5'te her 'K', belirli bir bölüm için 1000 kaydı temsil eder. Şekil 5.5'te görüldüğü gibi, K-Means algoritması, etkili sayısal şemasından dolayı KNIME çerçevesinde Spark ve K-means'deki iki Gauss yönteminden çok daha hızlıdır. Veriler katlanarak büyüdükçe, K-means yönteminin başarısı daha da artmaktadır.

Çizelge 5.4'de K-Means ve GMM uygulamalarında 10 aykırı değer bulma sonuçlarının karşılştırılması gösterilmektedir.

Çizelge 5.8. K-Means ve GMM uygulamalarının 10 aykırı değer tespit sonuçları.

K-Means	GMM
[3,3;0,258;242,62;13,8;0,0;24,0;31,0]	[5,086;0,86;236,86;21,8;38,0;1,0;31,0]
[3,65;0,274;242,43;15,2;0,0;28,0;31,0]	[3,3;0,258;242,62;13,8;0,0;24,0;31,0]
[4,422;0,43;238,51;19,0;0,0;28,0;31,0]	[3,65;0,274;242,43;15,2;0,0;28,0;31,0]
[3,568;0,436;246,05;14,8;0,0;23,0;31,0]	[2,582;0,314;242,92;10,8;0,0;11,0;31,0]
[4,328;0,372;238,16;18,2;0,0;33,0;30,0]	[2,922;0,306;241,97;12,2;0,0;14,0;31,0]
[4,172;0,138;238,94;17,4;0,0;37,0;30,0]	[4,422;0,43;238,51;19,0;0,0;28,0;31,0]
[3,192;0,138;240,64;13,4;0,0;22,0;30,0]	[3,568;0,436;246,05;14,8;0,0;23,0;31,0]
[3,858;0,212;236,16;16,4;0,0;32,0;30,0]	[4,592;0,452;239,57;19,2;39,0;0,0;31,0]
[3,846;0,236;235,58;16,6;0,0;29,0;30,0]	[2,478;0,394;237,64;10,6;5,0;0,0;30,0]
[4,844;0,27;236,19;20,4;1,0;35,0;30,0]	[2,898;0,728;238,35;13,2;12,0;0,0;30,0]



Şekil 5.5. Farklı veri boyutlarına göre çalışma süreleri.

Bu çalışmada, büyük veri üzerinde Spark teknolojisi kullanılarak veri kümeleme ve aykırı değer algılama işlemi daha hızlı gerçekleştirildi. Deney sonuçları, Spark MLlib'de uygulanan K-Means algoritmasını kullandıktan sonra, aykırı değer algılamanın başarıyla gerçekleştirildiğini kanıtladı. Veri kümeleri kümelendikten sonra, hileli verileri bulmak için aykırı değer tespiti bireysel elektrik enerjisi tüketimi veri kümeleri üzerinde verimli bir şekilde uygulanır. Dahası, algoritma, veri boyutunu arttırmada dikkatle ölçeklenir. UCI Makine Öğrenme Veri Havuzu veri setindeki deneyler çalışmamızın etkinliğini göstermektedir. Bu çalışma büyük ölçekli gerçek zamanlı veri kümeleri için genişletilebilir.

Ayrıca bu veri seti üzerinde MLlib Kütüphanesinin barındırdığı Gaussian Mixture yöntemi de uygulanarak bu iki algoritmanın çalışma süreleri karşılaştırılmıştır.

Çizelge 5.9. K-means algoritmasının veri büyüklüklerine göre çalışma süreleri.

Boyut	Süre (ms)	K-MEANS
1K Data	224	
2K Data	258	
5K Data	261	
10K Data	336	
25K Data	335	
60K Data	373	
150K Data	533	
250K Data	639	
600K Data	1052	
2000K Data	3234	

Çizelge 5.10. GMM algoritmasının veri büyüklüklerine göre çalışma süreleri.

Boyut	Süre (ms)	GAUSSIAN MIXTURE
1K Data	411	
2K Data	574	
5K Data	920	
10K Data	1688	
25K Data	3328	
60K Data	7351	
150K Data	15717	
250K Data	28757	
600K Data	66563	
2000K Data	212464	

Çizelge 5.9 ve 5.10'da yer alan farklı boyutlardaki veriler üzerinde uygulanan K-Means ve GMM algoritmalarının çalışma süreleri karşılaştırıldı. Tablolar incelendiğinde çalışma süreleri bakımından K-Means algoritmasının daha hızlı çalıştığı görüldü.

Kullanıcıların kesintisiz bir şekilde uygulamalarını gerçekleştirebilmek için geşiltirilen kümeleme (clustering) yapısı K-means uygulaması ile birlikte yürütüldü. K-Means algoritmasının Spark Clustering Standalone modu üzerinde yapılan uygulamada zaman ölçümü yapılmıştır. Oluştutulan 1 Master 3 Worker ile tek Master üzerinden yapılan K-means uygulamasının zaman karşılaştırılması yapılmıştır.

Çizelge 5.11. Clustering 1 Master çalışma süreleri.

Boyut	Süre (ms)	Clustering 1 Master
1K Data	810	
2K Data	970	
5K Data	2178	
10K Data	2245	
25K Data	2499	
60K Data	2828	
150K Data	4038	
250K Data	4345	
600K Data	7275	
2000K Data	15470	

Çizelge 5.12. Clustering 1 Master 3 Worker çalışma süreleri.

Boyut	Süre (ms)	Clustering 1 Master 3 Worker
1K Data	556	
2K Data	652	
5K Data	731	
10K Data	781	
25K Data	1003	
60K Data	1122	
150K Data	2064	
250K Data	2223	
600K Data	5875	
2000K Data	13536	

Çizelge 5.11 ve Çizelge 5.12 incelendiğinde 1 Master yapısının 1 Master 3 Worker yapısına göre daha yavaş öalıştırı görüldü. Worker düğümlerinin sayısının fazlalığı çalışma süresini doğrudan etkilediği belirlendi.

BÖLÜM 6

SONUÇLAR VE DEĞERLENDİRME

Bulunduğumuz teknoloji çağında bilginin depolanmasının, işlenmesinin ve analizinin önemli olması büyük veriyi değerli kılar. Analiz sonuçlarına doğrultusunda elde edilen çıkarımlar şirketlerin, kurum ve kuruluşların böylece toplumumuzun gelişmesine katkıda bulunmaktadır. Çeşitli teknolojiler sayesinde Büyük verinin saklanması ve işlenmesi kolay hale gelmiştir. Yeni geliştirilen Apache Spark teknolojisi sayesinde büyük verinin işlenme süresi kısaltılmıştır.

Bu çalışmada, iki farklı veri türü üzerinde sınıflandırma, kümeleme ve aykırı değer tespiti uygulamaları başarıyla yapılmıştır. Makine öğrenmesi algoritmalarını içeren Apache Spark'ın MLlib Kütüphanesi kullanılmıştır. Apache Spark teknolojisini kullanarak hataya dayanıklı, güvenilir, tutarlı ve hızlı sınıflandırma ve kümeleme işlemi gerçekleştirilmiştir. MLlib kütüphanesindeki Naive Bayes, K-means ve Gaussian Mixture yöntemleri ile büyük verilerin başarılı bir şekilde analiz edilmesi sağlanmış algoritmaların çalışma süreleri farklı veri boyutlarında incelenmiştir.

Naive Bayes yöntemi kullanılarak Apache Spark üzerinde hızlı metin sınıflandırması gerçekleştirilmiştir. Veri seti üzerinde farklı alanlardan seçilen anahtar kelimelere göre kategori tahmini yapılmıştır. Veri boyutunun büyüklüğü performans ve sonucu etkin bir şekilde değiştirir. Makine öğrenmesi ile güçlendirilen Naive Bayes algoritması ile metin sınıflandırma işlemi başarılı bir şekilde yapıldı. Bu yöntem ile sınıflandırma, istenmeyen mesaj (spam) analizi, cinsiyet tahmini gibi olasılık hesapları yapılabilmektedir. Spark teknolojisi ile standart veri madenciliği geliştirilerek makine öğrenmesi algoritmaları ile işlemler paralel hale getirilmiştir. Çalışma süresi ve analiz performansı gerçek metin verileri üzerinden gösterilmiştir.

Büyük Veri üzerinde Spark teknolojisi kullanılarak veri kümeleme ve aykırı değer algılama işlemi daha hızlı gerçekleştirildi. Deney sonuçları, Spark MLlib'de uygulanan K-Means algoritmasını kullandıktan sonra, aykırı değer tespitinin başarıyla gerçekleştirildiğini kanıtladı. Veri kümeleri kümelendikten sonra, hileli verileri bulmak için aykırı değer tespiti bireysel ev elektrik güç tüketimi veri kümeleri üzerinde etkili bir şekilde uygulandı. Algoritmanın, veri boyutunu arttırmada dikkatle ölçeklendiği kanıtlanmıştır. UCI Makine Öğrenme veri havuzu veri setindeki deneyler çalışmamızın etkinliğini göstermektedir.

Bu çalışma büyük ölçekli gerçek zamanlı veri kümeleri için genişletilebilir. K-means kümeleme algoritmasının uygulaması Spark Standalone modda, 1 master ile 1 master 3 worker şeklinde çalıştırılıp çalışma süreleri belirlenmiştir. 1 Master ile yürütülen uygulamanın daha uzun sürede gerçekleştiği görüldü. Apache Spark teknolojisi sayesinde uygulama süreçlerinin minimum düzeyde olması ve işlem sonuçlarının doğruluğu bu teknolojinin işlevselliğini kanıtlamıştır.

İleride yapılabilecek çalışmalarda Spark teknolojisinin dil ve platform bağımsızlığından yararlanılarak, analiz hesaplamaları alanında kullanıcı dostluğu ile bilinen Python dili ile birlikte paralel bilgisayar mimarisi kullanılarak yüksek boyutlu verilerde kümeleme algoritmaları üzerinden çalışmalar yürütülebilir. Ayrıca çeşitli veri tiplerinde aykırı değer algılama, dolandırıcılık tespiti, spam mail tespiti gibi birçok uygulamanın hızlı bir şekilde sonuçlanabilir.

KAYNAKLAR

1. Selçuk, A., Orencik, C., Savas, E., "Private search over big data leveraging distributed file system and parallel processing", ***Sabanci University Faculty of Engineering and Natural Sciences***, Istanbul (2015).

2. Bayrakcı, S., "Sosyal bilimlerdeki akademik çalışmalarda büyük veri kullanımı", ***Marmara Üniversitesi Sosyal Bilimler Enstitüsü***, İstanbul (2015).

3. Doğan, K., "Büyük verinin üniversite kütüphaneleri açısından önemi ve kütüphane hizmetlerinde kullanımı Ankara Üniversitesi Kütüphaneleri örneği", ***Ankara Üniversitesi Sosyal Bilimler Enstitüsü***, Ankara (2015).

4. Doğan, M., "Büyük verinin kişiler ve kurumlar üzerindeki etkileri", ***İstanbul Bilgi Üniversitesi Sosyal Bilimler Enstitüsü***, İstanbul (2014).

5. Shoro, A., G., and Soomro, T., R., "Big data analysis: Ap Spark perspective", ***Double Blind Peer Reviewed International Research Journal***, USA, 0975-4350 (2015).

6. Dülger, Ü., "Stratejik büyük veri yönetiminin yatırımlar üzerindeki etkileri", ***İstanbul Üniversitesi Fen Bilimleri Enstitüsü***, İstanbul (2015).

7. You, S., Zhang, J., and Gruenwald, L., "Large-scale spatial join query processing in cloud", ***Data Engineering Workshops (ICDEW), 2015 31st IEEE International Conference***, Seoul, South Korea (2015).

8. Kılıç, Y., "Stratejik mekansal büyük veri kümeleme", ***Fırat Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı***, Elazığ (2015).

9. Harnie, D., Vapirev, A., E., Wegner, J., K., Gedich, A., Vapirev, and M., Steijaert, "Scaling machine learning for target prediction in drug discovery using Apache Spark", ***15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing***, Vrije Universiteit Brussel, Russia, 871-879 (2015).

10. Fische, F., and Keim, D., A., "NStreamAware: real-time visual analytics for data streams to enhance situational awareness", ***Erschienen in: Proceedings of the Eleventh Workshop on Visualization for Cyber Security***, Paris, France (2014).

11. Gopalani, S., and Arora, R., "Comparing Apache Spark and map reduce with performance analysis using k-means", ***International Journal of Computer Applications***, 113 (1) (2015).

12. Wu, X., Zhu, X., Wu, G., Q., and Ding, W., "Data mining with big data", ***IEEE Trans. Knowl. Data Eng***, 26 (1): 97-107 (2014).

13. Xu, L., D., He, W., and Li, S., "Internet of things in industries: A survey", ***IEEE Trans. Ind. Informat***., 10 (4): 2233-2243 (2014).

14. Xu, R., and Wunsch, D., "Survey of clustering algorithms", ***IEEE Transactions on Neural Networks***, 16 (3): 645-678 (2005).

15. Kayım, F., "K-means ile DBSCAN algoritmasının paralelleştirilmesi ve HADOOP üzerimde büyük veri analizinde kullanılması, performans ve yeterlilik karşılaştırılması", ***Beykent Üniversitesi Fen Bilimleri Enstitüsü***, İstanbul (2015).

16. Şenbalcı, C., "Big Data Platform Development with a Telecom DSL", ***Kadir Has University Graduate School of Science and Engineering***, Cyprus (2013).

17. Cure, O., Naacke, H., Baazizi, M., and Amann, B., "On the Evaluation of RDF Distribution Algorithms Implemented over Apache Spark", ***Sorbonne Universites***, France (2015).

18. Lu, X., Rahman, W., and Islam, N., "Accelerating Spark with RDMA for Big Data Processing: Early Experiences", ***Network-Based Computing Laboratory***, USA (2014).

19. Hallaç, İ., R., "Büyük Veri Analizinde Dağıtık Makine Öğrenmesi Algoritmalarının Kullanılması", ***Fırat Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği***, Elazığ (2014).

20. Malakar, R., and Vydyanathan, N., "A CUDA-enabled Hadoop Cluster for Fast Distributed Image Processing", ***IEEE***, India (2013).

21. Sabau, A., S., "Survey of clustering based Financial Fraud Detection Research", ***Informatica Economica***, Romania, 16 (1) (2012).

22. Pachgade, Ms., S., D., and Dhande, Ms., S., S., "Outlier Detection over Data Set Using Cluster-Based and Distance-Based Approach", ***IJARCSSE***, 2 (6): 12-16 (2012).

23. Singh, M., Aashima, and Raheja, S., "Credit Card Fraud Recognition by Modifying K-means", ***IJARCSSE***, 4 (5): 1291-1296 (2014).

24. Ketu, S., and Agarwal, S., "Performance enhancement of distributed K-Means clustering for big Data analytics through in-memory computation", ***Contemporary Computing (IC3) IEEE*** (2015).

25. Wang, B. Yin, J. Hua, Q. Wu, Z. and Cao, J. "Parallelizing K-Means-Based Clustering on Spark", ***Advanced Cloud and Big Data (CBD)***, 31-36 (2016).

26. Sagiroglu, S., and Sinanc, D., "Big data: A review", ***Collaboration Technologies and Systems (CTS) International Conference on IEEE***, 42-47 (2013).

27. Patel, A., B., Birla, M., and Nair, U., "Addressing Big Data Problem Using Hadoop and Map Reduce", ***Nirma University International Conferance Oon Eng.*** (2012).

28. Katal, A., Wazid, M., and Goudar, R., H., "Big Data: Issues, Challenges, Tools and Good Practices", ***IEEE***, India (2013).

29. Singh, S., and Singh, N., "Big Data Analytics", ***International Conference on Communication, Information & Computing Technology (ICCICT)***, India (2012).

30. Karasulu, B., and Korukoğlu, S., "Modern Dağıtık Dosya Sistemlerinin Yapısal Karşılaştırılması", ***Çanakkale Onsekiz Mart Üniversitesi***, Çanakkale (2008).

31. Gazal, P., D., and Kaur, "A Survey on Big Data Storage Strategies", ***IEEE***, India (2015).

32. İnternet: Soft Computing and Intelligent Information Systems, "Big Data: Algorithms for Data Preprocessing, Computational Intelligence, and Imbalanced Classes", **http://sci2s.ugr.es/sites/default/files/files/TematicWebSites/BigData/mr.png** (2017).

33. Nandimath, J., Patil, A., Banerjee, E., Kakade, P., and Vaidya, S., "Big Data Analysis Using Apache Hadoop", ***IEEE IRI***, San Francisco, California, USA, 14-16 (2013).

34. Needell, D., and Warner, D., N., "Scalable Collaborative Filtering Recommendation Algorithms on Apache Spark", ***Claremont Mckenna College*** (2014).

35. Michael, T., "Machine Learning", ***McGraw-Hill***: ***USA*** (1997).

36. İnternet: What is the Hadoop Distributed File System (HDFS)?", **http://ibm.com** (2017).

37. Matei, Z., Mosharaf, C., Tathagata, D., Ankur, D., Justin, M., Murphy, M., Michael, J., Scott, S., and Ion, S., "Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing", ***USENIX Symp. Networked Systems Design and Implementation*** (2012).

38. İnternet: Chris, E., Computer Weekly, "Big data storage: Hadoop storage basics", **http://computerweekly.com** (2016).

39. Xin.R., Deyhim, P., Ghodsi, A., Meng, X., and Zaharia,M., "GraySort on Apache Spark by Databricks", ***Databricks Inc*** (2014).

40. Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., Freeman, J., Tsai, D., Amde, M., Owen, S., Xin, D., Xin, R., Franklin, M. J., Zadeh,R., Zaharia,M., and Talwalkar, A., "MLib: Machine Learning in Apache Spark", ***Journal of Machine Learning Research*** (2016).

41. Armbrust, M., Das, T., Davidson, A., Ghodsi, A., Or, A., Rosen, J., Stoica, I., Wendell, P., Xin, R., and Zaharia, M., "Scaling Spark in the Real World:Performance and Usability", ***Databricks Inc*** (2015).

42. Gonzalez, J., E., Xin, R., S., Dave, A., Crankshaw, D., and Franklin, M., J., "GraphX: Graph Processing in a Distributed Dataflow Framework", ***Usenix*** (2014).

43. İnternet: Apache Spark, "Machine Learning Library (MLlib) Guide", **https://spark.apache.org/** (2017).

44. Robbs, J. R., Walker, D. E., and Amsler, R. A., "Natural language access to structured text", ***Proceedings of the 9th conference on Computational linguistics*** (1982).

45. Cohen, K. B., and Hunter, L., "Getting Started in Text Mining", ***PLoS Computational Biology*** (2008).

46. Mohri, M., Rostamizadeh, A., and Talwalkar, A., "Foundations of Machine Learning", ***The MIT Press*** (2012).

47. Brodely, C., E., and Friedl, M., A., "Identifying and Eliminating Mislabeled Training Instances", ***Journal of Artificial Intelligence Research***, 11: 131-167 (1999).

48. Zaharia, M., Chowdhury M., Franklin, M. J., Shenker S., and Stoica, I., "Spark: Cluster Computing with Working Sets", ***Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing*** (2010).

49. Vapnik, V. N., "The Nature of Statistical Learning Theory" ***Springer Verlag*** (2000).

50. İnternet: Lang, K., CMU Text Learning Group Data Archives, **https://www.cs.cmu.edu/afs/cs/project/theo-20/www/data/news20.html** (2017).

51. Chawla, S., and Gionis, A., "k-means--: A unified approach to clustering and outlier detection", ***13th SIAM International Conference on Data Mining***, 189-197 (2013).

52. Sabau, A. S. "Survey of clustering based Financial Fraud Detection Research", ***Informatica Economica***, 16 (1) (2012).

53. Issa, H., Vasarhelyi, M., A., "Application of Anomaly Detection Techniques to Identify Fraudulent Refunds", ***SSRN Electronic Journal***, Rutgers Business School (2011).

54. MacQueen, J., B., "Some Methods for classification and Analysis of Multivariate Observations", ***Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability University of California Berkeley***, 1: 281-297 (1967).

55. Jain, A., K., "Data clustering: 50 years beyond K-means", ***Pattern Recognition Letters***, 31 (8): 651-666 (2010).

56. Dunham, M., "Data Mining: Introductory and Advanced Topics", ***Prentice Hall***, USA (2003).

57. Lichman, M., "UCI Machine Learning Repository Irvine", ***School of Information and Computer Science***, California (2013).

58. Reynolds, D., "Gaussian Mixture Models", ***MIT Lincoln Laboratory***, USA (2015).

59. İnternet: Crsouza, "Gaussian Mixture Models and Expectation-Maximization", **http://crsouza.com/2010/10/18/gaussian-mixture-models-and-expectation-maximization** (2010).

ÖZGEÇMİŞ

Yadigar ERDEM 1991 yılında İstanbul'da doğdu; ilk ve orta öğrenimini aynı şehirde tamamladı. Kağıthane Anadolu Teknik Lisesi Bilgisayar-Veritabanı Bölümü'nden mezun oldu. 2009 yılında Karabük Üniversitesi Teknik Eğitim Fakültesi Bilgisayar Sistemleri Öğretmenliği Bölümü'nde öğrenime başlayıp 2014 yılında iyi derece ile mezun oldu.

ADRES BİLGİLERİ

Adres : Kayabaşı Mah. Barbaros Cad. Merkez / KARABÜK
Tel : (370) 712 68 78
E-posta : ydgrerdem@gmail.com