İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

APACHE SPARK TABANLI DESTEK VEKTÖR MAKİNELERİ İLE AKAN BÜYÜK VERİ SINIFLANDIRMA

YÜKSEK LİSANS TEZİ

Barış AKGÜN

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

HAZİRAN 2016

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

APACHE SPARK TABANLI DESTEK VEKTÖR MAKİNELERİ İLE AKAN BÜYÜK VERİ SINIFLANDIRMA

YÜKSEK LİSANS TEZİ

Barış AKGÜN
504111502

Tez Danışmanı: Doç. Dr. Şule GÜNDÜZ ÖĞÜDÜCÜ

HAZİRAN 2016

İTÜ, Fen Bilimleri Enstitüsü'nün 504111502 numaralı Yüksek Lisans Öğrencisi Barış AKGÜN, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı **"**APACHE SPARK TABANLI DESTEK VEKTÖR MAKİNELERİ İLE AKAN BÜYÜK VERİ SINIFLANDIRMA**"** başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı : **Doç. Dr. Şule GÜNDÜZ ÖGÜDÜCÜ** .....
İstanbul Teknik Üniversitesi

Jüri Üyeleri : **Yrd. Doç. Dr. Yusuf YASLAN** .....
İstanbul Teknik Üniversitesi

Yrd. Doç. Dr. Tolga ENSARİ .....
İstanbul Üniversitesi

Teslim Tarihi : 20 Nisan 2016
Savunma Tarihi : 09 Haziran 2016

Aileme,

ÖNSÖZ

Tez çalışmam sürecinde her türlü desteğinden dolayı tez danışmanım sayın Doç. Dr. Şule Gündüz Öğüdücü'ye teşekkür ederim. Tez sürecimdeki katkılarından dolayı çalışma arkadaşlarıma; sağladığı imkanlardan ötürü çalıştığım kuruma, çalışma hayatım boyunca öğrenimime destek olan ve öğrenimim için her türlü imkanı sunan aileme ve desteğini hiçbir zaman esirgemeyen eşime sonsuz teşekkür eder, saygılarımı sunarım.

Haziran 2016

Barış AKGÜN
(Bilgisayar Mühendisi)

İÇİNDEKİLER

Sayfa

KISALTMALAR

LR : Lojistik Regresyon
DVM : Destek Vektör Makineleri
HDFS : Hadoop Distributed File System
MOA : Massive Online Analysis
SAMOA : Scalabled Massive Online Analysis
USA : United States of America
GFS : Google File System
I/O : Input/Output
API : Application Programming Interface
RDDs : Resilient Distributen Datasets
DAG : Directed Acyclic Graph
YARN : Yet Another Resource Negotiator
M : Milyon
TCP/IP : Transmission Control Protocol/Internet
SGD : Sthocastic Gradient Descent
GD : Gradient Descent
RAM : Random Access Memory
RMSE : Root Mean Square Error

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

APACHE SPARK TABANLI DESTEK VEKTÖR MAKINELERİ İLE AKAN BÜYÜK VERİ SINIFLANDIRMA

ÖZET

Klasik veri analizi yöntemlerinde, verilerin herhangi bir veri merkezinde toplanması ve veri madenciliği yöntemleri ile analiz edilerek anlamlı sonuçların çıkarılması gerekmektedir. Son yıllarda, gerek sosyal ağların gerek ise makinelerin ürettikleri veri miktarları çok ciddi boyutlardadır. Üretilen büyük miktardaki verileri saklamak yüksek maliyetli olmakla beraber bu denli büyük miktarda veriyi biriktirerek analiz etmek yine aynı şekilde ciddi zaman gerektirmektedir. Günümüz rekabet ortamında bir saniyenin dahi önemi oldukça büyüktür. Öyle ki sosyal ağlarda bir saniyede firmanız hakkında binlerce yorum yapılırken sizin buna sessiz kalmanız ciddi sonuçlara sebebiyet verecektir. Tüm bu ihtiyaçlar veri analizinin, biriktirilen veriler üzerinden değil verinin geldiği anda anlık veri üzerinden yapılması gerekliliğini doğurmuştur. Anlık veri üzerinden yapılan çalışmalar bilişim dünyasında akan veri analizi başlığı altında toplanmaktadır.

Sınıflandırma yöntemleri, veri madenciliği yöntemleri içinde en yaygın olarak kullanılan yöntemlerden bir tanesidir. Hemen her sektörde sınıflandırma problemlerine rastlamak mümkündür. Sınıflandırma problemlerinin çözümü için kullanılan algoritmaların başında, yüksek doğruluk oranı ve düşük kod karmaşıklığı ile Lojistik Regresyon (LR) ve yüksek doğruluk oranı ile Destek Vektör Makineleri (DVM) gelmektedir. Akan veri analizi yöntemlerinde en çok aranan özellik, gelen anlık verinin boyutuna bakılmaksızın olabildiğince hızlı olarak analiz edilebilmesidir. Bu sebebten LR ve DVM yöntemleri akan veri sınıflandırma problemi için de son derece uygundur.

Çalışmanın amacı büyük akan veri problemleri çözümü için sıklıkla kullanılan Apache Spark Streaming teknolojisi üzerine DVM sınıflandırma yöntemini geliştirmektir. Geliştirilen DVM çözümünün hali hazırda var olan LR yöntemi ile karşılaştırarak hangi yöntemin akan büyük veri sınıflandırmada daha etkin olduğunu bulmak ise çalışmanın bir diğer amacıdır.

Yapılan çalışmada öncelikli olarak, kullanılan sahtecilik veri kümesi için LR ve DVM yöntemlerinin başarımlarını analiz etmek adına birikmiş veri üzerinden SAS ürünü kullanılarak deneyler yapılmıştır. Bunun yanı sıra LR performansını arttırabilmek adına Stepwise yöntemi kullanılarak ek deneyler yapılmıştır. DVM ve LR yöntemlerinin başarımlarını incelemek için ROC değerleri ve çalışma süreleri ele alınmıştır. Akan veri analizi yöntemleri için bahsedildiği üzere doğruluk kadar çalışma zamanı da önemli bir performans metriğidir. Yapılan deneylerde karşılaşılan problemlerin başında SAS ile tek makine de yapılan analizlerin büyük veri kümesi üzerinde bellek yetersizliği sebebi ile sonlanamaması gelmektedir. İlgili problemi çözmek adına veri kümeleri özellik seçimi yöntemi ile küçültülerek deneyler yapılmış ve DVM yönteminin daha başarılı sonuçlar verdiği gözlemlenmiştir.

Sınıflandırma probleminin çözümünde DVM yönteminin başarılı olduğu gözlemlendikten sonra ilgili DVM yöntemi akan veri üzerine uyarlanmaya çalışılmıştır. Akan büyük veri teknolojisi olarak Apache Spark Streaming seçilmiştir. Apache Spark Streaming ve Apache Spark MLlib teknolojileri kullanılarak DVM yöntemi akan veri üzerine uyarlanmıştır. Apache Spark Streaming teknolojisi ile beraber gelen mevcut akan LR yöntemi ile sonuçlar karşılaştırılmış ve geliştirilen DVM yönteminin kullanılan veri kümeleri üzerinde daha başarılı sonuçlandığı gözlemlenmiştir. Öte yandan Apache Spark teknolojisi paralel programalama modeli ile çalışmaktadır. Geliştirilen akan DVM yöntemin dağıtık bilgisayarlara getirdiği yük incelendiğinde mevcut akan LR yöntemi ile benzer sonuçlar çıktığı gözlemlenmiştir.

APACHE SPARK BASED DISTRIBUTED SVM ALGORITHM FOR STREAM DATA CLASSIFICATION

SUMMARY

Nowadays, data is automatically generated at an increasing rate from mobile applications, sensor applications, log records, click-streams, call detail records, email, twitter posts and others. Much of these massive data is valuable at its time of received, therefore these types of data is called data stream and has to be analyzed in real time. Data mining is the best technique for analyzing datasets and then extracting the meaning of the data. The data stream is a real-time, continuous, ordered sequence of items, hence the arrival order of data records are not certain and keeping the all stream data records in local memory are not feasible. Stream data mining involves extracting knowledge from real-time actions. Mostly these real-time actions produce a massive high rate volume of data. The stream data mining approaches must handle these massive (big) and high rate data in very short time.

Classification methods are used for predicting the target class for each records in given dataset. Classification problems are the most common data mining tasks, therefore they are used for solving huge amount of business problems; such us, credit risk analysis, churn analysis etc. Binary classification problems whose target class may take just two values are the simplest type of classification problems in data mining, therefore all stream data mining technologies firstly handle binary classification problems. Logistic Regression and Support Vector Machines analysis are one of the widely used techniques in binary classification. Implementing Logistic Regression and Support Vector Machines are not complex and they are very efficient algorithms, therefore these two algorithms are good choices for modelling and predicting the behavior of massive stream data.

In this work, to satisfy the real time binary classification problem, we present Support Vector Machine algorithm for classifying given datasets. Our steps for presenting the streaming Support Vector Machines can be summarized as follows;

We firstly try to find which technique (Logistic Regression or Support Vector Machines) is the best one for solving binary classification problem. The SAS pre-defined Logistic Regression, Logistic Regression wtih Stepwise and Support Vector Machines functions are used for comparing the performances of these two algorithms on our datasets. Since our dataset is too massive, SAS Support Vector Machine on single machine can not produce any results and it returns memory exception. To compare these two algorithms, Support Vector Machine memory problem must be solved, therefore we choose a subset of given dataset. To standardize the range of independent features of data, we also normalize all the features for in our dataset. After feature scaling, we handle another normalized second training set for measuring the performance. The experiments for comparing these two algorithms on

SAS are executed on reduced dataset. The ROC area and execution time are chosen for performance metrics. After all experiments are completed on SAS, Support Vector Machines is more accurate than Logistic Regression, on the other hand the training execution times are almost similar for both Logistic Regression and Support Vector Machines.

To apply Support Vector Machine on stream data, streaming data challenges and streaming data analysis technologies are investigated deeply. The streaming technologies have to be ready for data analysis at any time and the arrival data rate in the streams may be very fast, which may result in crashing if too many items arrive. The streaming technologies must be process that fast data in very short time with limited memory. The other challenge is that streaming technologies have to include powerful error handling mechanism because the stream data can not be reproduced most of times. These are the two main challenges for streaming data analysis.

The first task is to determine the best stream technologies which can handle effectively these two main challenges. There are lots of open source and commercial effective platforms for solving streaming data analysis problems. Two of the most known ones are Apache Storm and Apache Spark Streaming, hence we focused on these two real time data analysis technologies. Both these two streaming technologies use paralel programing, so they are very fast on real time big data. However, the Apache Spark Streaming has more effective error handling mechanism than Apache Storm. Apache Spark Streaming receives data streams from sources and divides these stream data into mini batches. These mini batches are sent to Spark engine for processing. Since the job is executed on Spark engine, Apache Spark Streaming performs data parallel computations while Apache storm performs task parallel computations. Apache Storm restarts the worker node if any error occurs, on the other hand Apache Spark Streaming can continue with an other copy of data if any error occurs becuase of data paralel mechanism. In this study, Apache Spark Streaming is used for stream data analysis due to its better fault tolerance mehanism. Moreover it uses Apache Spark for job execution which results in a shorter development time comparing to Apache Storm.

After the selection of right algorithm and streaming technology, the same experiments are executed on Apache Spark distributed environment which has six nodes for performing data analysis. Apache Spark MLlib Support Vector Machine and Logistic Regression algorithm performances are compared with using our dataset and Support Vector Machine again shows better performance than Logistic Regression. Apache Spark Streaming has already Logistic Regression algorithm for stream real time data but it has not yet Support Vector Machine implementation for streaming data. Apache Spark MLlib Support Vector Machine algorithm is implemented on Apache Spark Streaming technology with using Scala programming language. To test our Support Vector Machine implementation, we run our experiments on stream data with using existing Logistic Regression and our Support Vector Machine implementation. The performance results of our implementation are very similar with existing Logistic Regression.

The Apache Spark is the paralel programming model, therefore we also look the cluster health when our implemented Support Vector Machine and existing Logistic Regression codes are executed on distributed mode. Actually, the CPU, network and

HDFS I/O performance are examined for seeing the cluster health when the both algorithms' code executed on it. Support Vector Machine and Logistic Regression algorithms cause almost same costs on cluster. At the end the of our work, the results are very promising and we believe that streaming data analysis will become more important at near future.

1. GİRİŞ

Günümüzde teknolojik altyapının gelişimine paralel olarak, üretilen veri miktarı da hızla artmaktadır. Kişiler artık her anlarını sosyal medyada paylaşmaktadır. Öyle ki 2012 yılında bir günde atılan tweet sayısı 340 M olarak kayıt altına alınmıştır [1]. Bunun yanı sıra makineler de milisaniyeler seviyesinde farklı formatta ve hızda veri üretebilmektedirler. Kısa zamanda hızla üretilen veri, bilişim dünyasında büyük veri olarak adlandırılan kavramı yaratmıştır.

Veri analizindeki genel yaklaşım üretilen verilerin herhangi bir veri depolama sisteminde saklanıp sonrasında saklanan veriden çıkarımlar yapılması yönündedir. Büyük veriyi saklamak donanımsal olarak maliyetli olmakla beraber yine bu veriden çıkarım yapmak uzun zaman almaktadır. Büyük veri kavramının getirisi olan bu iki problemi çözmek üzere ortaya bir takım teknolojiler sunulmuştur. Günümüzde Apache Hadoop ve Apache Spark teknolojileri, büyük veri problemi çözümünde kullanılan başlıca yöntemler sunmaktadırlar. Apache Hadoop teknolojisi temel olarak büyük verinin dağıtık sistemlerde maliyet bakımından ucuz şekilde saklanmasına ve paralel olarak işlenmesine olanak tanımaktadır. Apache Hadoop kullanılarak paralel veri işleme sayesinde her ne kadar büyük veri analizi hızlansa da disk tabanlı olmasından ötürü Apache Hadoop gerçek zamanlı analizlerden ziyade toplu (batch) analizler yapılmasına olanak sağlamaktıdır. Apache Spark teknolojisi de yine dağıtık sistemlerde paralel olarak büyük veriyi analiz etmek için kullanılmaktadır. Apache Hadoop teknolojisinden farklı olarak bellek içi çalışması sayesinde büyük veri analizinde performans olarak 100 kata yakın iyileştirme sağlamaktadır [26].

Büyük veri ve teknolojilerinin hayatımıza girmesi ile rekabet ortamı yeni bir boyut kazanmıştır. Firmalar artık sadece kendi verilerini ilişkisel veritabanlarında saklayıp analiz etmek yerine farklı kaynaklardan kendilerine avantaj sağlayacak tüm verileri toplayıp analiz etmek çabası içerisine girmişlerdir. Son zamanlarda gerek büyük veriyi saklama ve işlem maliyetleri gerekse rekabette avantaj sağlayacak anlık analiz ihtiyaçları sebebi ile verinin elde edilme anında işlenmesi ve analiz edilmesi ihtiyacı bilişim dünyasında akan veri analizi kavramını doğurmuştur. Akan veri gerçek

zamanlı, sürekli ve sıralı olarak gelen veri olarak tanımlanabilir [2]. Akan veriler, sosyal medya paylaşımları, bankacılık işlemleri, e-ticaret işlemleri gibi kişi etkileşimine dayanan, işlem bazlı sistemlerden üretilebileceği gibi sensör verileri veya veri günlüğü (data log) gibi makine bazlı da üretilebilmektedir. Özellikle makineler tarafında üretilen akan veriler çok büyük boyutlarda olabilmektedir. Aynı şekilde milyonlarca kullanıcısı olan sosyal medya platformlarından da işlem bazlı üretilen anlık veriler ciddi boyutlarda olmaktadır [3]. Akan veri analizi gerek verinin anlık gelmesi gerekse anlık gelen verinin büyük boyutlarda olması sebebi ile birtakım zorluklar içermektedir. Bunları başlıcaları aşağıdaki gibi sıralanabilir;

1. Veri analizi anlık olarak yapılmalıdır.
2. Veri analizinde kullanılan kaynaklar, özellikle bellek kısıtlıdır.
3. Veri analizde kullanılacak yöntem değişken boyutta ve akış hızında gelecek anlık veri için her an hazır durumda olmalıdır.
4. Anlık veriler genelde büyük boyutlarda olup 2. maddede belirtildiği üzere kısıtlı kaynaklarla çalışacak şekilde optimize edilmelidir.

Veri madenciliği algoritmaları veri analizinde kullanılan yöntemlerde en etkin çözümleri sunmaktadır. Akan veri analizi için de mevcut veri madenciliğinde kullanılan algoritmaların birçoğu, bilişim dünyasında akan veri madenciliği (stream data mining) başlığı altında uyarlanmıştır. Akan veriyi analiz etmek için sunulan tüm teknoloji ve yöntemlerin temel olarak yukarıda verilen 4 maddelik problem listesini çözüme kavuşturması gerekmektedir.

Çalışmanın amacı büyük akan veri problemleri çözümü için sıklıkla kullanılan Apache Spark Streaming teknolojisi üzerine Destek Vektör Makineleri (DVM) sınıflandırma yöntemini geliştirmektir. Geliştirilen DVM çözümünün hali hazırda var olan Lojistic Regresyon (LR) yöntemi ile karşılaştırarak hangi yöntemin akan büyük veri sınıflandırmada daha etkin olduğunu bulmak ise çalışmanın bir diğer amacıdır.

Yapılan bu çalışmada veri analizinde sıkça karşılaşılan sınıflandırma probleminin çözümü öncelikle SAS [57] uygulaması ile LR, LR Stepwise ve DVM algoritmaları kullanılarak çözümlenmeye çalışılmıştır. Veri kümesinin büyüklüğünden kaynaklı olarak belirtilen yöntemler sınırlı iterasyon sayıları ile çok uzun sürede tamamlanabilmiş, DVM algoritması ise yetersiz bellek hatası alarak tamamlanamamıştır. Karşılaşılan büyük veri sınıflandırma problemini çözebilmek

adına ilgili veri kümesi Hadoop Distributed File System (HDFS) dosyalama sistemine aktarılarak Apache Spark teknolojisi ile çözümlenmeye çalışılmıştır. LR ile DVM algoritmaları Apache Spark altyapısı ile beraber çok daha fazla iterasyon sayısı ile çok daha kısa sürede tamamlanabilmiştir. Yapılan çalışmada DVM algoritmasının LR'ye göre daha başarılı sonuçlar verdiği gözlemlenmiştir. Büyük veri sınıflandırma probleminin Apache Spark altyapısı ile kısa sürede tamamlanabiliyor olması LR ve DVM yöntemlerinin akan büyük veri sınıflandırma problemi üzerine uygulanabileceğine ışık tutmuştur. Hali hazırda Apache Spark Streming teknolojisi LR ile beraber kullanılabilirken DVM algoritması için bu şekilde bir uyarlama bulunmamaktadır. Bu kapsamda DVM algoritması Apache Spark MLlib ve Apache Spark Streaming teknolojileri kullanılarak geliştirilmiş ve hali hazırda Apache Spark'ta var olan akan LR algortimasından, kullanılan veri kümesi için daha başarılı sonuçlar elde edilmiştir.

Tez çalışması Giriş bölümü ile birlikte yedi bölümden oluşmaktadır. İlk bölümde tezin amacı ve planlanması özetlenmiştir. Yapılan çalışma ile benzer yapılmış akademik çalışmalar Bölüm 2'de verilmiştir. Büyük veri ve bu alanda günümüzde en etkin olarak kullanılan Apache Hadoop ve Apache Spark teknolojileri Bölüm 3'de açıklanmıştır. Veri analizi ve çalışmada kullanılan sınıflandırma yöntemleri Bölüm 4'te, akan büyük veri analizi ve bu alandaki teknolojiler Bölüm 5'te anlatılmıştır. Bölüm 6'da ise yapılan çalışmanın adımları, detayları ve sonuçları yer almıştır. Son olarak 7. Bölüm'de çalışmanın sonuçlarının yorumları ve gelecek çalışmalara yönelik öneriler yer almıştır.

1.1 Tezin Bilime Katkısı

Akan büyük veri sınıflandırma problemi bilişim dünyasında son dönemde en çok tartışılan konulardan bir tanesidir. Akan büyük veri problemleri için ortaya çıkan teknolojiler ve bunların çeşitli uygulama alanlarına uyarlanması her geçen gün artmaktadır. Apache Spark Streaming ve Apache Storm teknolojileri akan büyük veri problem çözümleri için en yaygın kullanılan teknolojilerdir.

Sınıflandırma yöntemleri akan büyük veri için çeşitli büyük veri teknolojileri üzerine uyarlanmaya devam edilmektedir. Sınıflandırma problemlerinde sıklıkla kullanılan LR yöntemi Apache Spark Streaming teknolojisi üzerine uyarlanmışken DVM için hali hazırda bir uyarlanma bulunmamaktadır. Yapılan bu çalışma ile beraber DVM

yöntemi Apache Spark Streaming üzerine uyarlanarak akan büyük veri sınıflandırma problemi çözümlenmeye çalışılmıştır. Çıkan sonuçlar geliştirilen yöntemin var olan LR yöntemi ile benzer çıktılar gösterdiğini sergilemektedir.

Geliştirilen yöntem ile beraber büyük akan veri sınıflandırma problemi üzerinde DVM ile hızlıca aksiyon alınabilecektir.

2. LİTERATÜR ÇALIŞMASI

Akan veri analizi yeni sayılabilecek bir kavram olmak ile beraber bu konuda çeşitli çalışmalar yapılmıştır. Yapılan ilk çalışmalar akan verinin, veri analizinde kullanılabilmesi için gerekliliklerin neler olduklarını belirleme yönünde olmuştur. Bunlardan bir tanesinde akan verinin karakteristik özellikleri, akan verinin çeşitli uygulama alanlarında detaylı olarak incelenmesi ve ilgili yıllarda sunulan çözümlerin akan veri analizi zorluklarını ne ölçüde karşıladığı ele alınmıştır [4]. İlgili çalışma akan veriyi tanımak ve onun gereksinimlerini karşılayacak teknolojiler geliştirebilmek adına literatürde yer almaktadır.

2003 yılında yapılan bir çalışmada [5] ise sınıflandırma algoritmalarının bazıları ağırlıklandırılarak melez bir yapı kurulmuş ve kurulan melez yapının tek bir sınıflandırma algoritması kullanılması durumundan performans farkı ölçülmüştür. Melez yöntemler kullanılarak, akan veri üzerindeki sınıflandırma probleminin çözümünün hata oranında %3 düzeyinde bir iyileştirme sağlanmıştır. Bu çalışmada akan veri üzerinden 250-1000 kayıtlık bir örneklem alınmıştır. Bu miktarın küçük olmasından kaynaklı tek bir sınıflandırma algoritması yerine melez yöntemle başarımın arttırılması amaçlanmıştır.

Yapılan başka bir çalışmada [6] ise karar ağacı algoritmasının öğrenme süreci hızlandırılarak akan veriye uygun hale getirilmesi amaçlanmıştır. Öğrenme sürecini hızlandırmak adına Gini endeksi her yeni gelen veri için hesaplanmamıştır. Gelen örneğin veri kümesindeki her bir girdinin yaprak düğüm oluşturmadaki etkisine bakılarak Gini endeks hesaplaması yapılmıştır. Aynı şekilde sınıflandırma aşamasında karar ağacı bellek içinde saklanmış, bellek yetmediği durumlarda ise karara etki oranına göre bazı yaprak düğümler pasif ve aktif hale getirilmiştir. Çalışma sonucu, bir karar ağacı algoritması olan C4.5 algoritması ile karşılaştırılmıştır. Gini endeksi her girdi için hesaplanmadığından 100 satırlık veride C4.5 %10 oranında hızlanma sağlanır iken veri boyutları arttığında farkın ortadan kalktığı ve C4.5'in sonuç üretemediği gözlemlenmiştir.

Akan veri madenciliği ile yapılan sınıflandırma çözümlerinde en büyük problem akan veri içersinde yer alan verilerin birçoğunun etiketsiz olarak gelmesidir. Yapılan çalışmaların bir çoğunda [7,8,9] ve bizim çalışmamızda da denetimli öğrenme (supervised learning) yöntemleri kullanılmış ve gelen eğitim verisinin hepsinin etiketli olduğu ortamda çalışma yapılmışır. Etiketsiz olarak gelen her veri için ise öğrenme kümesi baz alınarak etiketleme yapılmıştır. Yapılan bir çalışmada bu problem ele alınmış ve yarı denetimli öğrenme (semi supervised learning) ile akan veri sınıflandırılması yapılmıştır [10]. Yapılan bu çalışma ile etiketsiz gelen veriler ve etiketli gelen veriler, etiket sayısı kadar küçük kümelere ayrılarak, etiketsiz olan verilere bulunduğu kümelerdeki etiketli verilerin etiketi atanmıştır. Kümeleme algoritması olarak K-means algoritması kullanılmıştır. Akan veri kümesi boyutu 100'den 1 M'a kadar değişken olup etiket sayısı ise 5 ile 40 arasında seçilmiştir. Çalışmanın çıktısı olarak %10 etiketli, %90'u etiketsiz olan veri kümesinde %90 düzeyinde doğruluk elde edilmiştir.

Akan veri madenciliği alanında yapılan en geniş kapsamlı çalışmalardan birtanesi ise A.Bifet öncülüğündea başlatılan Massive Online Anaylsis (MOA) [11] adlı çalışmadır. Bu çalışmada kümeleme sınıflandırma ve regresyon alanında yer alan birçok veri madenciliği algoritması akan veri madenciliği problemleri için uyarlanmıştır. Aynı çalışma ile akan verinin durumunu izleme ve aynı akan veri üzerinde birden çok algoritmayı kıyaslama imkanı mevcuttur. İlgili çalışmada kullanıcı arayüzü sayesinde çok kısa zamanda akan veri üzerinde değişik algoritmalar denenerek analizler yapılmasına olanak sağlar.

Yukarıda bahsi geçen çalışmalar, dikkat edildiği üzere küçük veri kümeleri için uyarlanmıştır. Fakat günümüzde akan veri kümelerinin hacimleri de ciddi boyutlarda olmaktadır. Akan büyük veri promlemine yönelik olarak MOA'nın dağıtık sistemlerde çalışan versiyonu olan Scalabled Advanced Massive Online Analysis (SAMOA) adında çalışma yapılmıştır [12]. İlgili çalışmanın amacı veri madenciliği algoritmalarının, akan veri için dağıtık sistemlerde çalışmalarına olanak sağlamaktır. Kullanıcılar SAMOA algoritmalarını Apache Storm, Samza, Flink ve S4 akan veri analizi yazılımları üzerinde çalıştırabilirler. Aynı şekilde, belirtilen benzeri akan veri analizi yazılımları üzerine çalışan algoritmalar geliştirilebilir.

3. BÜYÜK VERİ TEKNOLOJİLERİ

3.1 Büyük Veri Kavramı ve Karakteristikleri

Teknolojinin hızla gelişmesi ile beraber hayatımızdaki herşey gibi verinin de tanım ve büyüklüğü aynı hızla değişim göstermektedir. Verinin her geçen gün katlanarak büyümesi gerek rekabet ortamına yeni bir boyut kazandırmış gerekse bilişim dünyasında yeni tanım ve metriklerin oluşmasına yol açmıştır. Büyük veri kavramı, göreceli olarak değişmekle beraber klasik veri depolama yöntemleri ile saklanması ve analiz edilmesi mümkün olmayan veri kümelerini ifade etmek için kullanılır. Günümüzde her saniye çok büyük boyutlarda veri, sensörler, GPS aygıtları, sosyal medya vb. kaynaklar tarafından üretilmektedir. Öyle ki son bir dakika içerisinde video paylaşım sitesi olan youtube kullanıcıları toplamda 48 saatlik bir video yüklerken fotoğraf paylaşım sitesi olan instagram'da 3600 yeni içerik paylaşılmıştır [13]. Mikroblog olanağı sağlayan Twitter'de ise 100000 yeni tweet atılırken, firma ve kişiler yaklaşık 35000 beğeni almış ve 200 M e-posta gönderilmiştir [13].

2000'li yıllardan itibaren üretilen verilerin, analog ortamlardan dijital ortama kaydığı bir dönüşüm yaşanmaktadır. Büyük veri kavramı da bu dönüşümle beraber 2000'li yılların sonunda endüstri analisti Doug Laney'nin bilişim dünyasına 3 V tanımı ile girmiştir [14]. Doug Lanen'nin kabul görmüş tanımına göre Hacim (Volume), Hız (Velocity) ve Çeşitlilik (Variaty) kavramları büyük veriyi ifade etmektedir. Firmaların büyük veriyi analiz etmesi için bu 3 V'yi optimize edecek olan spesifik analitik yazılımlara ve yeni sistemlere ihtiyaçları vardır [15]. Büyük verinini kabul görmüş karakteristik özellikleri de yine Doug Lanen'in 3 V tanımına dayanmaktadır:

Hacim: Verinin her geçen gün Terabyte'dan Zetabyte, Petabyte seviyesine artan büyüklüğünü ifade etmektedir.

Hız: Verinin büyük hacimlerde sürekli akışkan olarak gelmeye devam etmesini ifade eden bir terimdir. Yüksek hızda gelen bu veriden verim alabilmek için gelen hacimli veriyi kısa zamanda işleme ve saklama yöntemlerinin geliştirilmiş olması gerekmektedir.

Çeşitlilik: Gelen veriler farklı kaynaklardan, farklı format, yapı ve karmaşıklık düzeyi ile gelmektedir. Farklı tiplerde gelen verilerin saklanabilmesi ve işlenmesi büyük verinin bir başka karakteristik özelliğini oluşturmaktadır. Genel kabul gören bu 3 karakteristik özelliğin yanı sıra son yıllarda gelen verinin boyutunun zamansal olarak değişmesi ve farklı kaynaklardan gelen verilerin nasıl birleştirilmesi gerektiği kavramları da büyük verinin karakterestik özellikleri arasına bazı kaynaklarca eklenmiştir.

Günümüzde büyük verinin özel sektör, kamu ve akademik dünyada birçok uygulama alanı vardır. Örneğin parakende zinciri olan WallMart her saat başında 1 M müşteri işlemini saklayacak bir veri depolama sistemi gerçeklemiştir. Aynı şekilde bugün birçok firma mobil teknolojilerden aldığı lokasyon bilgisi ile müşterilerinin bulunduğu konuma göre kampanya üreten sistemler üzerine çalışmaktadır. Yine bir çok internet sitesi, kullanıcı özgü öneri sistemlerini optimize edebilmek adına tüm kullanıcıların gerek demografik gerekse işlemsel bilgilerini analiz etmektedir. 2012 yılında Amerika Birleşik Devletleri hükümeti, büyük verinin ülkede karşılaşılan problemleri çözmek adına kullanılması için bir araştırma başlatmıştır [16]. Yine bugün birçok devletin sosyal medya üzerinden suç ve örgütlenme tespiti yaptğı bilinmektedir.

Son yıllarda kullanım alanı yaygınlaşan büyük verinin getirileri olduğu kadar zorlukları da vardır. Bu denli büyük hacimde üretilen verilerin saklanması ve güvenliğinin sağlanması için teknolojik olarak yeterli altyapı sağlanmalıdır. Öte-yandan yapılan araştırmalar göstermektedir ki üretilen verilerin sadece %25'i saklanabilmektedir. Büyük veriyi analiz etmek mevcut yazılımlarla mümkün olamamaktadır. Bu sebebten büyük veri özelliklerini göz önüne alan yeni yazılımlar geliştirilmektedir. Bir diğer zorluk ise insan kaynağıdır. Büyük veri bilişim dünyasında yeni sayılabilecek bir kavramdır. Büyük veriden değer çıkarabilecek tecrübe birikimi yeni yeni kazanılmakta ve her geçen gün yeni bir problemle yüzleşilmektedir. Bu nedenledir ki her geçen gün bilişim dünyasında yeni teknolojiler ortaya çıkmaktadır.

3.2 Büyük Veri Teknolojileri

Büyük veri kavramı önceki bölümlerde bahsedildiği gibi klasik yöntemlerden farklı teknolojik altyapıya ihtiyaç duymaktadır. Yüksek hacimli olan verinin

saklanabilmesi için tek bir donanım yeterli olmayacaktır. Öncelikle büyük verinin güvenli olarak saklanabilmesi ve işlenmesi için dağıtık donanımsal altyapı ve dağıtık dosyalama sistemine ihtiyaç duyulmaktadır.

Dağıtık sistem kavramı birden çok bilgisayarın tek bir bilgisayar gibi ölçeklendirilmesi ve kullanılmasını ifade eder [17]. Yine aynı şekilde büyük hacimli verilerin dağıtık sistemlerde saklanması için temelde küçük parçalara ayrılması ve dağıtık sistemlerde işlenebilir olarak tutulması gerekmektedir. Bu problemi çözmek için dağıtık dosyalama (depolama) sistemlerine ihtiyaç duyulmaktadır. Bilişim dünyasında 3 popüler (Amazon-S3, HDFS, GFS) dağıtık depolama sistemi vardır. GFS ve Amazon-S3 firma spesfik olup GFS Google arama işlemlerinde, Amazon-S3 ise Amazon bulut altyapsında kullanılan dosyalama sistemlerdir. HDFS ise açık kaynak kodlu olup hata kotarım altyapısı ve ucuz donanımlarla ölçeklenebilirliği ile bilişim dünyasına girmiş bir dosyalama sistemidir [18].

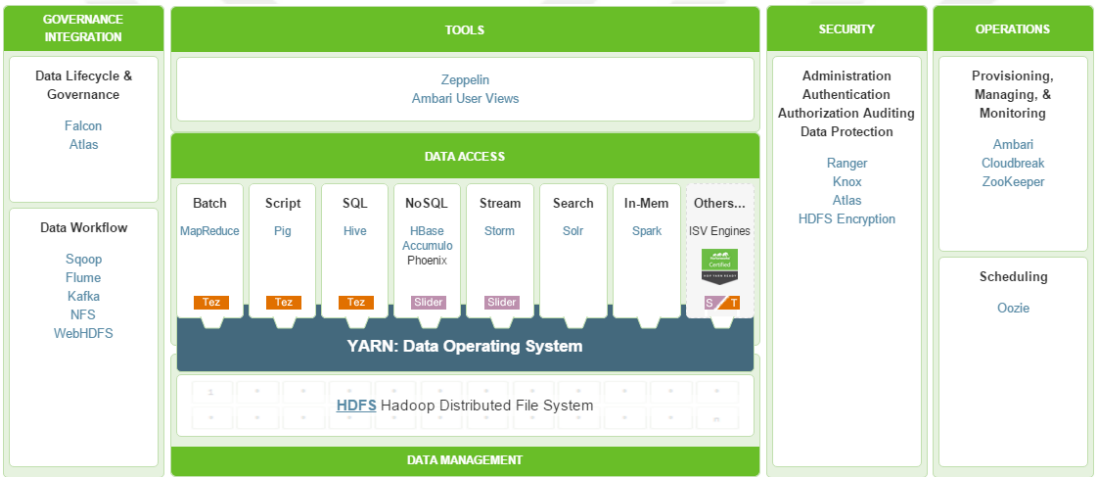
Donanımsal altyapıdaki farklılışma ve büyük veri analizindeki ihtiyaçlar yeni yazılım çözümleri gereksinimlerini de doğurmuştur. Öncelikle dağıtık sistemler ve dağıtık dosyalama sistemleri ile çalışabilecek olan programlama modelleri ve bu modellere dayanan yeni analiz yazılımları ve veritabanı programları ortaya çıkmıştır. Büyük veri dünyasında şu anda en çok konuşulan programlama modelleri MapReduce [19] ve Apache Spark' a ait olan RDDs temelli programlama modeli [20] olarak verilebilir. Her iki programlama modeli de klasik veri programlama modellerinden farklı olarak veriyi işleme taşımak yerine, işlemi verinin tutulduğu bilgisayarlara gönderme temelli çalışmaktadır. Her iki modelin de detayları ileriki bölümlerde açıklanacaktır.

İlgili programlama modelleri kullanılarak bilişim dünyasında veri saklama ve analizi alanında birçok yeni yazılım ortaya çıkmıştır. Bunlardan en önemlileri ise HDFS dosyalama sistemini ortaya çıkartan ve üzerinde MapReduce programlama modeli kullanılmasına olanak sağlayan Apache Hadoop ve dağıtık bellek içi tabanlı da işlem gücü sağlayabilen Apache Spark teknolojileridir. Bu teknolojileri kullanan Mahout ve MLlib gibi veri madenciliği yazılımları ve veri sorgulamaya yarayan Hive, Pig, Spark SQL gibi yazılımlar ortaya çıkmıştır. Bunların dışında büyük hacimli verileri saklamaya ve performanslı sorgulamaya yarayan Hbase [50], MongoDB [51], Cassandra [52] gibi NOSQL veritabanları bilişim dünyasına kazandırılmıştır.

Belirtildiği üzere günümüzde büyük veri denildiği zaman akla gelen iki teknoloji (Apache Hadoop ve Apache Spark) vardır. Sonraki bölümlerde bu teknolojilerin ayrıntılarına yer verilecektir.

3.2.1 Apache Hadoop

Büyük veriyi herhangi özellik gereksinimi olmayan bilgisayarlardan oluşan dağıtık sistemlerde saklamaya ve işlemeye yarayan açık kaynak kodlu java programlama dili ile geliştirilmiş yazılım teknolojisidir. Apache Hadoop son kullanıcının yapısal olan ve olmayan yüksek hacimli veriden anlam çıkarmasına olanak sağlamaktadır. Dağıtık sistemler üzerinde çalışan dağıtık programlar için altyapı sunmaktadır. Öyle ki Apache Hadoop çevresinde ortaya çıkan birçok yazılım mevcuttur. (Şekil 3.1 [22])



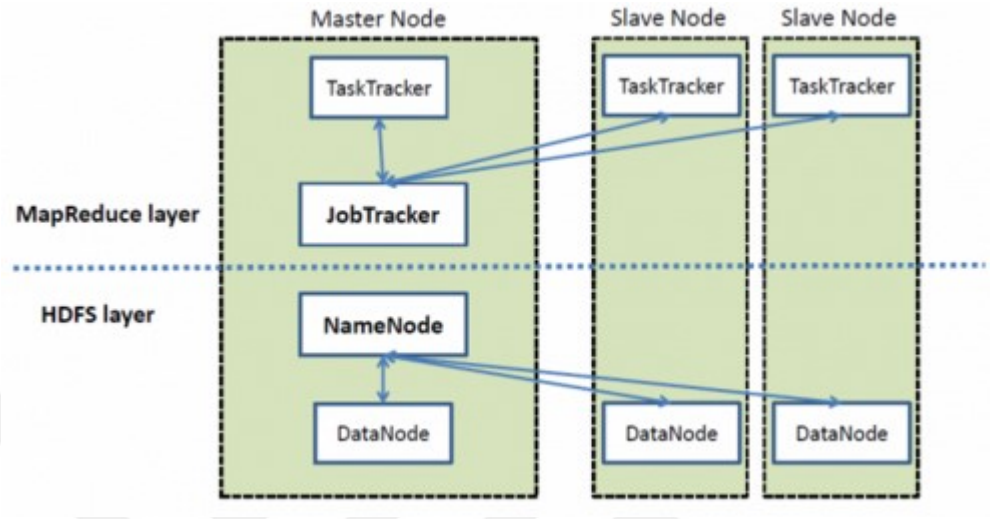
Şekil 3.1: Apache Hadoop ekosistem yazılımları.

Temel olarak hadoop 4 bileşenden oluşmaktadır [21].

1. Hadoop Common: Hadoop tarafından gereksinim duyulan tüm kütüphane ve modülleri ifade eder.
2. HDFS : Hata kotarım mekanizmasına ve düşük ağ trafiğine sahip dağıtık sistemler üzerinde çalışan dosyalama sistemidir
3. Hadoop YARN: Apache Hadoop ekosisteminde kaynakların yönetiminden sorumludur. Bahsedildiği üzere hadoop ekosisteminde çalışan birtakım yazılımlar mevcuttur. Gerek bu yazılımların gerekse kullanıcıların Hadoop ortamında yazmış olduğu kod parçacıklarının kaynak paylaştırılması YARN tarafından yapılmakatadır.

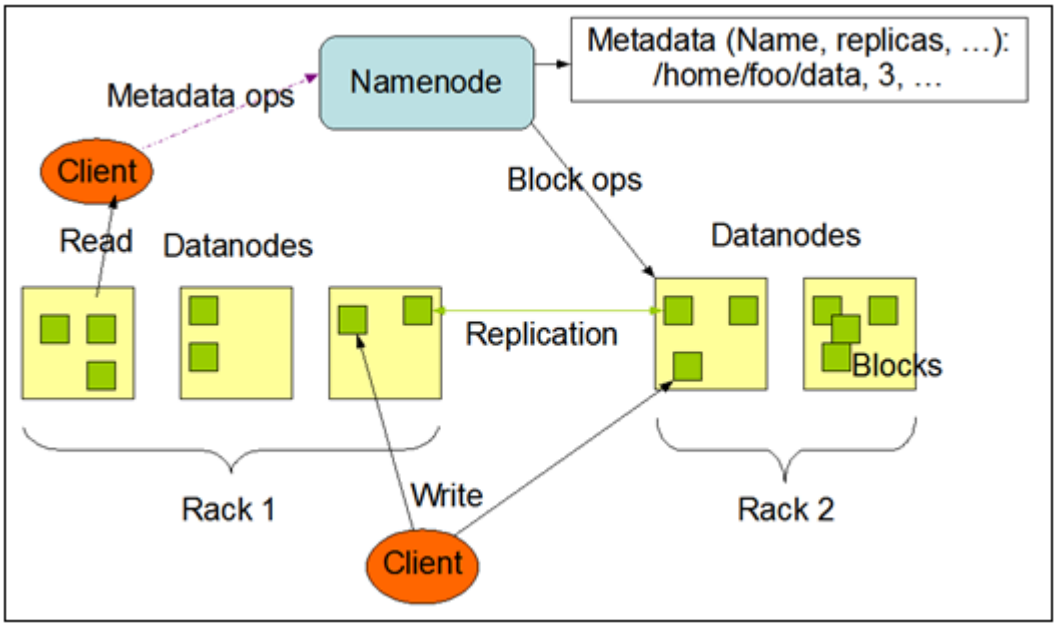
4. Hadoop MapReduce: Büyük veri analizinde kullanılan bir programlama modelidir.

Apache Hadoop her ne kadar bu 4 bileşenden oluşsa da bilişim dünyasında Hadoop denildiğinde akla HDFS ve MapReduce kavramları gelmektedir. (Şekil 3.2 [21])



Şekil 3.2: Apache Hadoop HDFS ve MapReduce Version1 genel yapısı.

HDFS sayesinde dağıtık sistemlerdeki bilgisayarların diskleri bir araya getirilerek tek bir sanal disk oluşturulur. Oluşan bu sanal disk büyük hacimdeki verilerin saklanmasına olanak sağlar. HDFS dosyalama sistemi 1 ana düğüm (namenode) ve istenilen sayıda işçi düğüm'den (datanode) oluşmaktadır. Ana düğüm işçi düğüm'lerde yer alan veri bloklarının adresini tutmakla yükümlüdür. Aynı şekilde işçi düğüm'lerdeki veri bloklarının yaratılması, silinmesi ve takibi ana düğüm tarafından yürütülür. İşçi düğüm'ler HDFS sisteminde veri bloklarının saklanılmasından ve veriler üzerinde yapılan analiz işlerini yürütmekten sorumludur. HDFS sisteminde tüm düğüm'lerin birbirleri ile haberleşmesi TCP/IP protokolü ile yürütülmektedir.



Şekil 3.3: HDFS mimarisi [23].

HDFS dağıtık dosyalama sistemini diğerlerinden ayıran en büyük özellik veri kaybı riskinin yok denilecek kadar az olmasıdır. Öyle ki büyük hacimli veriyi oluşturan her veri bloğunun varsayılan (ayarlanabilir) olarak 3 kopyası saklanmaktadır. Şekil 3.3' ten yola çıkıldığında 2 kopya Küme1'de (Rack1) saklanırken diğer kopya Küme2'de (Rack2) saklanır. Böylelikle hem disk kaybında hem de küme kaybında veri kaybı yaşanmayacaktır. Ayrıca tüm işçi düğüm'ler durumlarını kalp atışı (heartbeat) kavramı adı altıda ana düğüm'e iletirler. Bu sayede ana düğüm işçi düğüm'ler hakkında güncel bilgiye sahip olmakta ve olası işçi düğüm kaybında ana düğüm tarafından aksiyon alınabilmektedir.

HDFS sisteminde ikincil ana düğüm (secondary namenode) ise tam olarak ana düğüm yedeği gibi çalışmamaktadır. Ana düğüm dağıtık dosya sistemin durumunu *fsimage* adı altında saklar ve süreç içinde yapılan tüm değişiklikler günlük (log) dosyasında saklanır. Apache Hadoop'un kurulu olduğu dağıtık sistemler yeniden başlatıldığında günlük dosyasında yer alan değişiklikler fsimage dosyasına uygulanır. Bu sayede ana düğüm üzerinde yer alan I/O trafigi azaltılmış olunur. İkincil ana düğüm ise yeniden başlatma beklemeksizin belirli aralıklarla ana düğüm'ün fsimage dosyasına günlük dosyasında yer alan değişiklikleri uygulamakla yükümlüdür. Herhangi bir yeniden başlatma işleminde sistem ikincil ana düğüm'ün fsimage dosyası kullanılarak yeniden başlatılır. Böylelikle yeniden başlatma işlemi hızlanmış olacaktır.

3.2.1.1 MapReduce programlama modeli

HDFS üzerinde tutulan verilerin güvenli ve dağıtık olarak işlenebilmesine olanak sağlayan programlama modelidir. Temel olarak Map ve Reduce fonksiyonlarına dayanan ve adını da buradan alan programlama modelinde HDFS'teki veri blokları map fonksiyonu sayesinde filtrelenir ve istenilen (anahtar,değer) çiftlerine dönüştürülür. Map fonksiyonun çıktısı ise anahtar bazlı gruplanarak sıralanır ve Reduce fonksiyonu kullanılarak sonuç elde edilir.

Map ve Reduce fonksiyonları yazıldıktan sonra ilgili kod Apache Hadoop ortamında çalıştırılır. Yazılan Map ve Reduce kod blokları Apache Hadoop konfigürasyonları kullanılarak HDFS'te yer alan ilgili veri kümelerinin olduğu yerde çalıştırılır. Verinin olduğu düğüm'lerde çalışan Map fonksiyonu çıktıları dağıtık sistem içersindeki TCP/IP ağ protokolü ile toplanarak sonuçlar yine HDFS'e yazılır [24]. Apache Hadoop'un gücü önceki bölümde bahsedildiği gibi verinin işleme değil işlemin veriye atanmasıdan gelmektedir. Bu sayede dağıtık sistem içersinde ağ trafiği oluşmamakta, sistemdeki işçi düğüm sayısı arttıkça da performans doğru orantı ile artmaktadır.

Apache Hadoop, MapReduce işlerinin yürütülmesinde, MapReduce Version1 ve MapReduce Version2'de farklılıklar bulunmaktadır. MapReduce Version1'de İş Yöneticisi (JobTracker) ve Görev Yöneticisi (TaskTracker) kavramları kullanılır. İş Yöneticisi bir tane olmakla beraber her işçi düğüm'de bir Görev Yöneticisi vardır. İş Yöneticisi, MapReduce kod bloklarını dağıtık sistemde paralel olarak çalıştırılmakla yükümlüdür. Dağıtılan MapReduce kodlarının çalışması sırasında meydana gelebilecek herhangi bir problemin de çözüme kavuşturulması yine İş Yöneticisi tarafından yapılmaktadır. İş Yöneticisi ana düğüm'ün meta verisini kullanarak işçi düğüm'lerin lokal diski hakkında bilgi sahibi olur ve ilgili işçi düğüm'ün Görev Yöneticisine en uygun işi atar. Görev Yöneticisi ise her işçi düğüm'de yer alır ve işçi düğüm'de yer alan veri ve iş yoğunluğuna göre İş Yöneticisinden iş talep eder. MapReduce Version2 de ise İş Yöneticisi ve Görev Yöneticisi kavramları yoktur. YARN kaynak yöneticisi tarafından, çalıştırılan her iş için herhangi bir düğüm üzerinde Uygulama Yöneticisi (Application Master) yaratılır. Uygulama Yöneticisi işin hangi düğüm'lerde çalışacağına her düğüm üzerinde yer alan Düğüm Yöneticisi (Node Manager) kavramlarını kullanarak karar verir. Aynı şekilde Düğüm Yönetici'leri kendilerine verilen işin sonlandırılmasından da sorumludur. Basit

anlamda Uygulama Yöneticileri İş Yöneticisi; Düğüm Yönetici'leri ise Görev Yöneticisi gibi düşünülebilir.

3.2.1.2 Apache Hadoop dezavantajları

Apache Hadoop büyük veri analizinde kullanılan en bilindik yazılım teknolojisi olmasına rağmen kullanımında birtakım zorluklar ve dezavantajlar vardır [25]. Bu zorlukların bir kısmı büyük verinin doğasından kaynaklı olup hemen her büyük veri yazılımı için geçerlidir. Önceki bölümlerde bahsedildiği gibi büyük veri alanında tecrübeli insan kaynağı bulmak zordur. Apache Hadoop kavramlarına hakim kişiler bilişim dünyasında çok fazla olmadığından, yapılacak yanlış bir kodlama verinin analizini mümkün hale getirmeyeceği gibi veri kaybı gibi çok daha ciddi sonuçlar doğurabilir. Yine aynı şekilde veri güvenliği ve kullanıcı yetkilendirme işlemleri de Apache Hadoop ortamı için ciddi sıkıntılar oluşturmaktadır. Apache Hadoop ortamında son zamanlarda Kerberos [55] kullanılarak veri güvenliği sağlanmaktadır. Yeni bir entegrasyon olduğu için bu noktada ciddi sıkıntılar mevcuttur. Apache Hadoop sistemlerini yönetmek ve gerek MapReduce gerekse HDFS çevresinde gelişen ürünleri kullanmak kolay değildir. Bu da Apache Hadoop konusunda bilgili kişi sayısının artamamasındaki en büyük etkendir. Verilen tüm bu dezavantajlar aslında Apache Hadoop'ta olduğu kadar tüm büyük veri yazılımları için de geçerlidir.

Apache Hadoop'a özgü olan en büyük dezavantaj ise MapReduce programlama modelinin bilişim dünyasındaki her iş için uygun olmamasıdır. MapReduce disk tabanlı olarak çalışan bir modeldir. Her MapReduce işinde diskten okuma ve diske yazma yapılır. Bu nedenledir ki itarasyon gerektiren analiz işlemleri zaman alabilmektedir. Apache Hadoop'un bu dezavantajından yola çıkarak Apache Spark teknolojisi geliştirilmiş ve bellek içi tabanlı, dağıtık çalışabilen büyük veri analiz teknolojisi ortaya çıkmıştır.

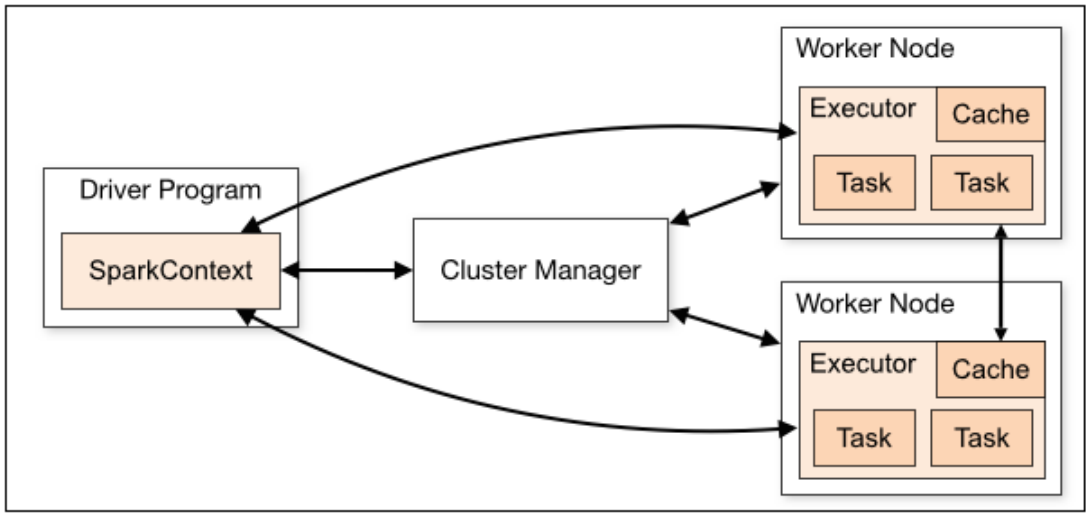
3.2.2 Apache Spark

Apache Spark, Berkeley Üniversitesi AMPLab laboratuvarunda 2009 yılında Matei Zaharia tarafından geliştirilmesine başlanan açık kaynak kodlu, dağıtık sistemlerde veri analizi gerçeklemeye yarayan teknolojidir. Temel olarak Apache Hadoop'un MapReduce disk bazlı programlama modelinden farklı olarak dağıtık bellek içi veri işleme özelliğine sahiptir. Yapılan çalışmalar Apache Spark teknolojsinin Apache

Hadoop'a oranla 100 kat daha hızlı olduğunu göstermektedir [26]. Apache Spark teknolojisinde veriler bellek içinde saklandığından tekrar kullanım sırasında herhangi bir I/O işlemi yapılmayacağı için yüksek performans elde edilir. Bu nedenle iterasyon gerektiren veri madenciliği algoritmaları için Apache Spark, Apache Hadoop teknolojisinden çok daha ileridedir.

Apache Spark teknolojisi Scala, Python ve Java dilleri için API desteği sunmaktadır. Sunmuş olduğu dil seçenekleri sayesinde bilişim dünyasında daha fazla kişiye hitap etmektedir. Her ne kadar 3 dil desteği sunulsa da Apache Spark geliştiricileri önceliği sırasıyla Scala, Python ve Java olarak belirlemişlerdir. Scala dilinin fonksiyonel bir dil olmasından dolayı herhangi bir problemin çözümü Scala dili ile çok daha az kod satırı kullanılarak tamamlanabilmektedir. Her ne kadar son dönemde disk tabanlı versiyonu sunulsa da Apache Spark teknolojisi için doğrudan Apache Hadoop'un yerini alacak bir teknoloji demek doğru olmayacaktır.

Apache Spark teknolojisinin kendisine ait spesifik dağıtık dosyalama sistemi bulunmamaktadır. Öte yandan Apache Spark, HDFS, Cassandra, Amazon S3 ve Kudu gibi dağıtık dosyalama sistemlerinden veriyi okuyabilir. Apache Spark'ın gereksinim duyduğu bir diğer bileşen ise dağıtık sistem kaynak yöneticidir. Apache Spark'ın kendine ait Spark Standalone adı verilen kaynak yöneticisi mevcut olduğu gibi Hadoop YARN, Apache Mesos ürünlerine de kaynak yöneticisi olarak desteklemektedir. Bunların dışında Apache Spark teknolojisini, geliştirme yapmak ya da yazılan kodların test edilmesini sağlamak amaçlı lokal mod'da kullanmak ta mümkündür. Apache Spark, dağıtık sistemler üzerinde çalıştırıldığında ana düğüm (driver node), kaynak yönetim uygulamasının uygun gördüğü düğüm'lerden birisinde ya da doğrudan kodun çalıştırıldığı düğüm'de oluşturulur. Yine aynı şekilde diğer düğüm'lerde ise çalışıtırıcı düğüm'ler (executer node) oluşturulurak ilgili Spark işi çalıştırılır. Apache Spark teknolojisinin çalışmasına ilişkin akış Şekil 3.4'te verilmiştir [26].



Şekil 3.4: Apache Spark çalışma prensibi.

Apache Spark'a ait olan bileşenler aşağıdaki gibi sıralanabilir [27].

3.2.2.1 Spark Core ve Resilient Distributed Datasets

Spark Core kütüphanesi Apache Spark projesinin temeli olarak düşünülebilir. Apache Spark'a ait tüm bileşenler Spark Core kütüphanesi temel alınarak geliştirilmiştir. Apache Spark teknolojinde programlama modeline verilen isim RDDs' dir. RDD's kavramı verinin bellek içersinde tutulan ve paralel olarak işlenebilen halini ifade eder. RDDs'ler yukarıdaki bahsedilen herhangi dış bir dosyalama sisteminde yer alan veriden yaratılabileceği gibi kullanıcılar tarafından Spark programında üretilen veriler de olabilirler. Üretilen her RDDs, bölümlenerek dağıtık sistemde yer alan bilgisayarlara konumlandırılan çalıştırıcı düğüm'ler ile işlenir. RDDs'lerin işlenmesi Apache Hadoop teknolojisinde yer alan Map Reduce fonksiyonlarına benzer bir şekilde gerçeklenir. Apache Hadoop teknolojisnde yer alan Map fonksiyonu Apache Spark teknolojsinde *Transformations*, Reduce fonksiyonunu ise *Actions* olrak ifade edilmiştir. Transformations işlemlerinde var olan RDD's lerden yeni RDDs'ler üretiliriken Actions işlemlerinde ise Transformations sonrası üretilen RDDs'ler toplanarak sonuç elde edilir. Elde edilen sonuç ana düğüm'e (driver program) gönderilebileceği gibi yukarıda bahsi geçen herhangi bir dağıtık dosyalama sistemine de yazılabilir.

3.2.2.2 Spark SQL

Spark SQL [28] Apache Spark'ın 1.3 versiyonu ile beraber ortaya çıkmış bir teknolojidir. Spark SQL ile beraber RDDs'den sonra yeni bir veri saklama yöntemi

olan *DataFrames* yapıları Apache Spark teknolojisine kazandırılmıştır. Spark SQL temel olarak yapısal olmayan verilerin bellek içersinde DataFrames adı verilen kavramla saklanmasını ve saklanan DataFrames yapılarının SQL programlama diline benzer bir dil ile sorgulanmasına olanak sağlar.

Bunun yanı sıra Spark SQL kullanılarak Oracle, MSSQL, MySQL gibi ilişkisel veritabanlarından da okuma yazma yapılabilir. Bu da kurumlara ilişkisel veri tabanlarında sakladıkları verilerle, yapısal olmayan verilerin bir araya getirilip hızlıca analiz edilmesini olanak sağlayan bir kapı açmaktadır. Spark SQL kullanılarak büyük veri dünyasında yer alan hive vb. veritabanlarınıda sorgulamak mümkün olmaktadır.

3.2.2.3 Spark MLlib

Apache Spark teknolojisi, ilgili bölümün başında da belirtildiği üzere dağıtık bellek içi çalışan bir yazılım olduğundan iterasyon gerektiren işlemler için oldukça uygundur. Birçok veri madenciliği algoritması da yüksek sayıda iterasyona ihtiyaç duymaktadır.

Apache Spark büyük veri üzerinde veri madenciliği çalışmasını kolaylaştırabilmek adına MLlib [29] adında makine ögrenmesi kütüphanesine sahiptir. MLlib içerisinde sınıflandırma, kümeleme, öneri sistemleri gibi veri madenciliği algoritmalarının bir çoğunu içermektedir. Yine bir çok istatistiki hesaplama MLlib içinde tanımlanmıştır. Apache Spark MLlib kullanılarak çok kısa süre içersinde birçok veri analizi elde edilebilir.

3.2.2.4 Spark Streaming

Apache Spark Streaming [30] teknolojisi kullanılarak Apache Spark ile yapılabilecek tüm işlemlerin akan veriye uyarlanması mümkün hale getirilmiştir. Akan veri işlemleri için en önemli sorunlar zaman ve bellek kullanımıdır. Apache Spark teknolojisi bu iki soruna da çözüm bulmaktadır.

Apache Spark Streaming teknolojisinde temel olarak akan veriler düzenli olarak Apache Spark Core teknolojisine iletilir. Bu sayede akan veriler Apache Spark tarafından analiz edilmiş olur. Akan veri Apache Spark teknolojisi tarafından analiz edildiği için akan veri analizi ile toplu veri analizi maliyetleri Apache Spark teknolojisinde aynıdır.

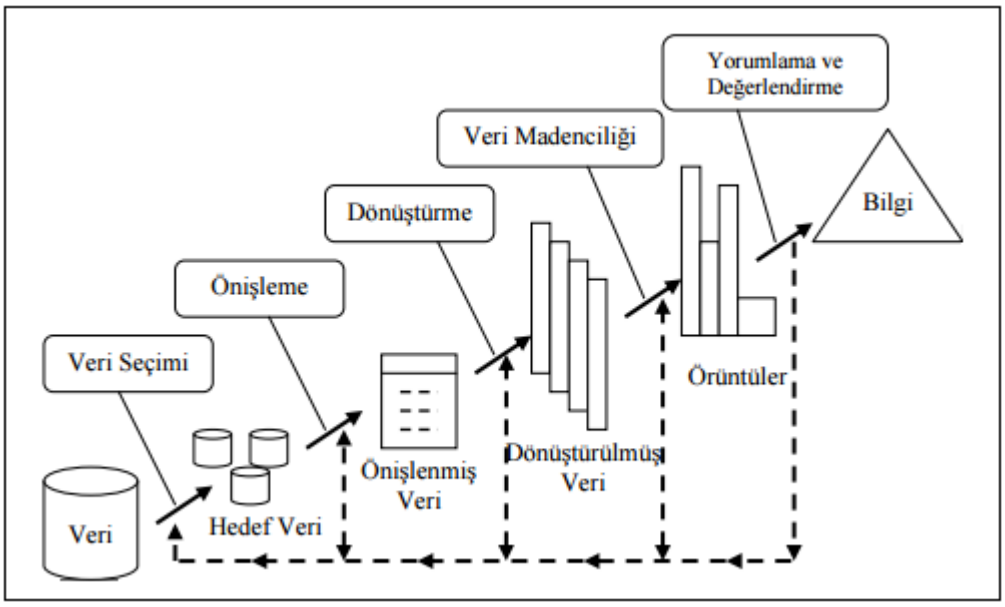
Apache Spark Streaming teknolojisi akan büyük veri kapsamında ilk akla gelen teknolojilerdendir. Spark Streaming teknolojisi ileriki bölümlerde detaylı olarak açıklanacaktır.

4. VERİ MADENCİLİĞİ VE SINIFLANDIRMA PROBLEMİ

4.1 Veri Madenciliği

Veri madenciliği, veri kümesi içersinde yer alan anlamlı verinin bulunması ve veriden çıkarım yapılması için kullanılan bir tekniktir. Veri madenciliği yapılmasının temel amacı insan gözüyle anlamlandırılamayan verilerden istatistik, makine öğrenimi ve bilgisayar kullanılarak sonuca varmaktır.

Veri madenciliğinin adımları aşağıdaki Şekil 4.1'de verilmiştir [37].



Şekil 4.1: Veri madenciliği adımları.

Günümüzde veri madenciliği denildiği zaman öncelikli olarak akıllara makine öğrenmesi kavramı gelmektedir. Veri madenciliği insan ve makine unsurundan oluşmaktadır. İnsan, analiz edilecek problemin belirlenmesi, probleme ait çözümün ve çözüme ait verinin sağlanmasından sorumludur. Problem, çözüm yöntemi ve veri insan tarafından, bir araya getirilmek üzere makinelera verilir. Makineler ise verilen veri ve uygun yöntemlerle problemi çözüme kavuşturmaktan ve çözüm sırasında veriden yapımış olduğu çıkarımları öğrenmekten sorumludur. Çıkan çözümlerin problemle ne kadar örtüştüğünü tespit etmek, örtüşmüyorsa gerekli düzenlemeleri yapmak yine insan sorumluluğundadır. Makine öğrenmesi aşamasında makinelerden

beklenen, verilerden örüntü çıkartabilmesi ve yeni gelen her veriyi değerlendirmeye alarak örüntüyü güncel tutmasıdır.

Veri madenciliği çalışmalarının hepsinde beklenen iki temel şey ise probleme getirilen çözümün doğruluğu ve ilgili çözümün zaman karmaşıklığıdır.

Veri madenciliği yöntemlerini temelde iki kategoride ifade etmek mümkündür [38].

- Tanımlayıcı (Descriptive)
- Tahmin edici veya öngörüsel (Predictive)

Tanımlayıcı yöntemler eldeki verinin özelliklerini tanımak için kullanılır. Yapılacak olan istatistiki çalışmalarla elimizdeki verinin dağılımı, bağımlılıkları ve bize yaptığı etki araştırılmış olur. Temel olarak geleceğe ait kuralların çıkarılabilmesi için eldeki mevcut verilerden örüntüler çıkarmaya dayanır.

Tahmin edici modellerde ise verilerden çıkarılan bilgi gelecekte oluşabilecek trendleri tahminlemede kullanılır. Tahmin edici modellerde amaç bilinmeyeni bulmaktır. Örneğin yapılan bir kredi başvurusunun sahte olup olmadığını anlamak ya da önümüzdeki sene trendin mavi renk olacağını söylemek tahmin edici modelleler ile mümkün olabilmektedir.

Tanımlayacı modelleri kurmak basit olsa da kişilere kazandıracağı değer tahminleme modellerine oranla daha düşük olacaktır. Bu sebebten bilişim dünyasında daha çok tahmin edici modeller ağırlıklı çalışmalar yürütülmektedir.

Veri madenciliği algoritmalarını aşağıdaki gibi gruplamak mümkündür [39];

- Tanımlama ve Ayrımlama (Characterization and Discrimination)
- Birliktelik Analizi (Association Analysis)
- Sınıflandırma ve Öngörü (Classification and Prediction)
- Kümeleme Analizi (Cluster Analysis)
- Sıradışılık (İstisna) Analizi (Outlier Analysis)
- Evrimsel Analiz (Evolution Analysis)

Günümüzde yaygın olarak kullanılan algoritmalar tahminleme modelleri için kullanılan sınıflandırma ve öngörü yöntemleri içerisinde yer almaktadır. Yapılan akademik araştırmada [40] Sınıflandırma ve Öngörü algoritması olarak kullanılan

DVM ve LR analizleri en çok kullanılan algoritmalar arasında gösterilmiştir. İlgili iki sınıflandırma algoritmasının bu denli yaygın kullanılıyor olması ve düşük algoritma karmaşıklığı yapılan bu çalışmaya ışık tutmuştur.

4.2 Sınıflandırma Problemi

Sınıflandırma yöntemi günümüzde hem akademik dünyada hem de gerek iş gerekse günlük yaşantımızda dahi en yaygın kullanılan yöntemdir. Çevremizde gördüğümüz hemen herşeyi başka birisine aktarırken iyi, kötü, güzel çirkin vb. sıfatları kullanarak etiketleriz. Bu sıfatları kullanırken aslında yine o güne kadar edindiğimiz verileri kullanıp çıkarımlar yaparız.

Bilişim dünyasında da, veri madenciliğinde de aslında durum benzerdir. Veri madenciliğinde, elimizdeki etiketli verilerin kullanılarak yeni gelen veri kümelerinin etiketlenmesi işlemi sınıflandırma olarak adlandırılır. Sınıflandırma yöntemlerindeki temel amaç eldeki verileri kullanarak, yeni gelecek verileri yüksek doğrulukta tahminleyecek sınıflandırıcıları oluşturabilmektir. Bazı durumlarda bir sınıflandırma problemi için birden fazla sınıflandırma algoritması kullanılması doğruluk oranını gözle görülür arttırabilir.

Sınıflandırma problemlerinin yaygın kullanım alanları mevcuttur. Bankacılık sektöründe sahtecilik, telekom sektöründe müşteri sadakati analizi, tıp alanında hastalık tespitleri bunlardan sadece birkaçıdır. Bahsedildiği üzere her problem farklı sınıflandırma algoritması ya da algoritmaları gerektirse de en yaygın kullanılan algoritmaların arasında DVM ve LR analizi de yer almaktadır [40] ve bu çalışmada da akan verinin sınıflandırması için bu iki algoritmadan yararlanılmıştır.

4.2.1 Destek Vektör Makineleri

Son zamanlarda sınıflandırma problemlerinin çözümü için kullanılan en yaygın yöntemlerden birisi DVM'dir. Birçok sınıflandırma problemi çözümünde, DVM başarım oranı yüksek çıkmasından ötürü kullanım alanı yaygınlaşmıştır. DVM yöntemini diğer yöntemlerden farklı kılan, algoritma karmaşıklığının düşük olması ve öğrenme sırasındaki işlem sayısının az olmasıdır [41]. Bu sebebten büyük veri sınıflandırma işlemleri için de son derece uygun bir yöntemdir.

DVM, ilk etapta iki sınıflı doğrusal verilerin sınıflandırlması için kullanılmış olup sonrasında çok sınıflı doğrusal olmayan veri kümelerinin sınıflandırılması için de kullanılmıştır. DVM'in temel amacı, sınıfları birbirinden ayıran optimum hiper düzlemin belirlenmesidir. Bu da ayrı sınıfta yer alan en yakın iki örneğin arasındaki mesafenin maksimize edilmesiyle elde edilir.

DVM yöntemlerini doğrusal ayrılabilen veriler ve doğrusal olmayan veriler üzerinde çalışan olarak ikiye ayırmak mümkündür.

4.2.1.1 Doğrusal ayrılabilen veriler için Destek Vektör Makineleri

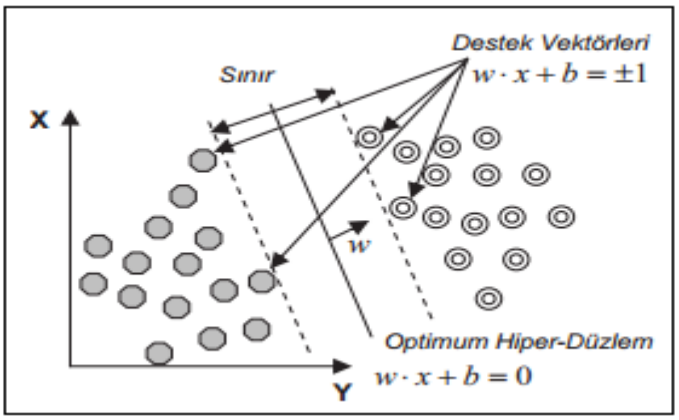
Doğrusal DVM yönteminde amaç iki sınıfı birbirinden ayırabilecek fonksiyonu bulmaktır. Elde edilen fonksiyon ile beraber iki sınıfta yer alan en yakın iki nokta arasındaki mesafeyi maksimize edecek hiper düzlem bulunmuş olur. Doğrusal sınıflandırma problemlerinde iki sınıf arasında birden çok hiper düzlem çizilebilir. DVM yöntemi çizilen hiper düzleme en yakın noktalar arasındaki mesafeyi maksimize etmeyi amaçlar. Maksimum uzaklığı sağlayan bu hiper düzleme optimum hiper düzlem adı verilir. Her iki sınıfa ait sınırları belirleyen noktalardan geçen vektörlere ise destek vektörleri adı verilir.

Doğrusal ayrılabilen iki sınıflı bir problemde DVM algoritmasının eğitilmesi için seçilen n sayıda örnek içeren veri kümesinde yer alan her bir örnek $\{x_i, y_i\}$, i =1,2,…,n ise, optimum hiper düzeleme ait denklemler aşağıdaki gibi tanımlanabilir [42].

$$wx_i + \beta \geq +1, \qquad y_i = +1 \; için \tag{4.1}$$

$$wx_i + \beta \leq -1, \qquad y_i = -1 \; için \tag{4.2}$$

Verilen denklem 4.1 ve 4.2'de, $x \epsilon R^N$ olup N boyutlu bir uzayı, $y \epsilon \{+1, -1\}$ ise ikili sınıf etiketlerini, w ağrılık vektörünü ve β eğilim değerini ifade etmektedir. Optimum hiper düzlemin oluşabilmesi için optimum hiper düzleme paralel olacak ve iki sınıf arasındaki sınırları oluşturacak iki hiper düzlemin belirlenmesi gerekir (Şekil 4.2 [43]). Bu iki hiper düzlemi oluşturan noktalardan geçen vektörler destek vektörleri olarak adlandırılır ve bu vektörler $wx_i + \beta = \pm 1$ şeklinde ifade edilir [43].



Şekil 4.2: Doğrusal olarak ayrılabilen veri kümeleri için optimum hiper düzlemin belirlenmesi [43]

Destek vektörleri arasındaki uzaklık, geometrik uzaklıktan $\frac{2}{\|w\|}$ olarak hesaplanır. Destek vektörleri arasındaki mesafeyi maksimum yapmak $\frac{\|w\|}{2}$ ifadesinin minimize edilmesi ile mümkün olacaktır. Buna göre optimum hiper düzlem aşağıdaki kısıtlara sahip optimizasyon problemi ile bulunabilir [43].

Optimizasyon Problemi;

$$min\frac{[\|w\|]}{2} \tag{4.3}$$

Kısıtlar;

$$y(wx_i + \beta) - 1 \geq 0 \; ve \; y \in \{+1, -1\} \tag{4.4}$$

Optimizasyon probleminin çözümünden yola çıkıldığında verilen bir girdi için etiket ataması basit şekilde Denklem 4.5'teki işaret fonksiyonu ile yapılır.

$$sgn(wx + \beta) \tag{4.5}$$

Belirli bir hata oranı ile doğrusal ayrılma durumu günümüz veri kümeleri için daha yaygındır. DVM yönteminde hata fonksiyonu olarak genelde hinge loss fonksiyonu kullanılmaktadır. Bu durumda yeni DVM denklemi aşağıdaki gibi yazılabilir.

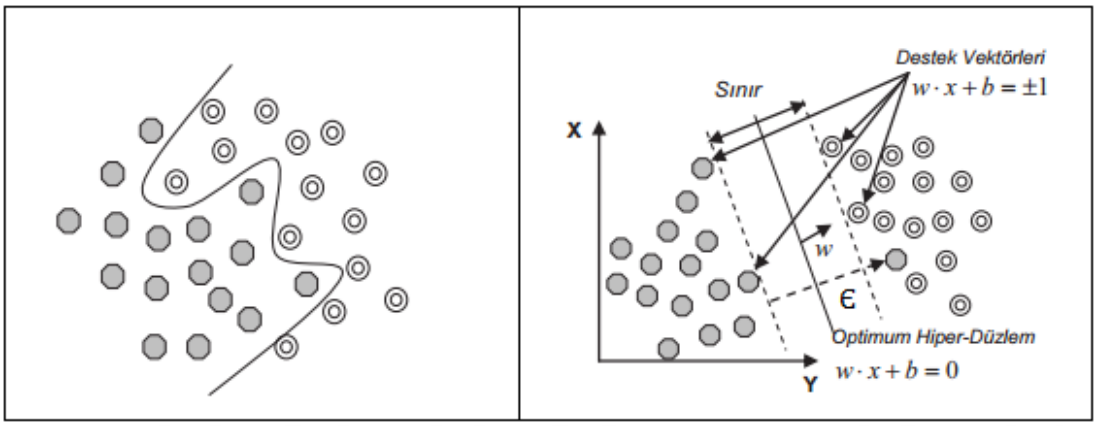
$$f(w, \beta) = \frac{1}{n}\sum_{i=1}^{n} \max(0, 1 - y_i(wx_i + \beta)) + a\frac{\|w\|}{2} \tag{4.6}$$

Denklem 4.6'da a düzenlileştirme parametresini ifade etmektedir.

Bizim yaptığımız çalışmada da doğrusal olarak ayrılabilen veri kümesi için Apache Spark MLlib içersinde yer alan DVM algoritması kullanılmıştır. Hata fonksiyonunu minimize etmek için Stochastic Gradient Descent (SGD) yöntemi kullanılmıştır. SGD yönteminin açıklaması Bölüm 6'da verilmiştir. Doğrusal DVM algoritması hızlı çalışma ve az öğrenme adımı içermesinden ötürü büyük akan veri analizi için de son derece uygundur.

4.2.1.2 Doğrusal olarak ayrılamayan veriler için Destek Vektör Makineleri

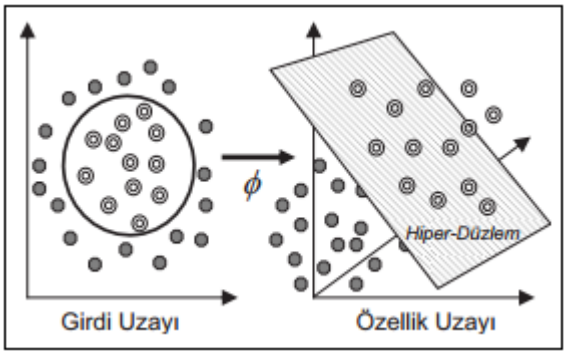
Bilişim dünyasında birçok problemde doğrusal olarak ayrılamayan veriler mevcuttur.(Şekil 4.3 [45])



Şekil 4.3: Doğrusal olarak ayrılamayan veri kümeleri için optimum hiper düzlemin belirlenmesi.

Doğrusal olarak ayrılamayan veri kümesi probleminden kaynaklanan, bir sınıfın sınır hiper düzlemin diğer sınıf tarafında olan kısmında kalması problemi pozitif yapay değişken adı verilen ϵ_i'nin tanımlanması ile çözülür. Doğrusal olarak ayrılabilen yöntemde olduğu gibi destek vektörler arasındaki mesafeyi maksimize ederken sınıflandırma hatalarının minumuma indirilmesi C denge parametresi ile ifade edilir [44, 45].

Doğrusal ayrılamayan veri kümesi probleminin çözümü, ilgili parametreler kullanılarak aşağıdaki optimizasyon probleminin çözümü ile mümkün olur. Optimizasyon probleminin çözümü için Şekil 4.4'te ki [45] gibi doğrusal olarak ayrılamayan veri, özellik uzayı adı verilen çok boyutlu uzaya çevrilir. Böylece sınıfları birbirinden ayırabilen optimium hiper düzlem elde edilebilir.



Şekil 4.4: Çekirdek fonksiyonu ile verinin çok boyutlu uzaya çevrilmesi.

Doğrusal olarak ayrılamayan verilerin çok boyutlu özellik uzayına çevrilmesi çekirdek (kernel) fonksiyonları ile mümkün olmaktadır. Bilişim dünyasında en çok kullanılan çekirdek fonksiyonları polinom, radyal tabanlı fonksiyon, Pearson VII ve normalleştirilmiş polinom olarak sıralanabilir. Herbir çekirdek fonksiyonu kendine ait parametreler içermektedir. Çekirdek fonksiyonlarına ait parametrelerin optimize edilmesiyle ilgili literatürde birçok çalışma bulunmaktadır [46, 47].

4.2.2 Regresyon analizi

Regresyon analizi, veri kümesindeki değişkenler arasında yer alan ilişkiyi belirlemek için kullanılan bir yöntemdir. Regresyon analizinde değişkenler arası ilişki ölçümü bağımlı ve bağımsız değişkenler kavramı ile belirlenir. Regresyon analizindeki temel mantık bağımlı değişken ve bağımsız değişkenlerden oluşan denklem 4.7 'yi oluşturabilmektir.

$$y = f(x) + \epsilon \tag{4.7}$$

Bağımlı değişken y ile ifade edilir ve eşitliğin sol tarafında yer alır, bağımsız değişkenler ise x ile ifade edilirler ve ağırlıklandırılarak eşitliğin sağ tarafında yer alır. Böylelikle bağımsız değişkenler ve bağımlı değişken arasında yer alan ilişki boyutları denklemdeki bu ağırlık katsayıları ile ölçülmüş olur. Denklem 3.6'da verilen ϵ ise bağımlı değişkeni tahmin etmek için kullanılan hata payını göstermektedir.

Regresyon yöntemleri bağımlı değişkenin nitel ya da nicel olmasına göre doğrusal (linear) ve lojistik (logistic) regresyon olarak ikiye ayrılır. Doğrusal regresyon analizinde girdiye ait bağımlı değişkenin değeri tespit edilmeye çalışılırken lojistik regresyon analizinde bağımlı değişkenlerden herhangi birinin olma olasılığı

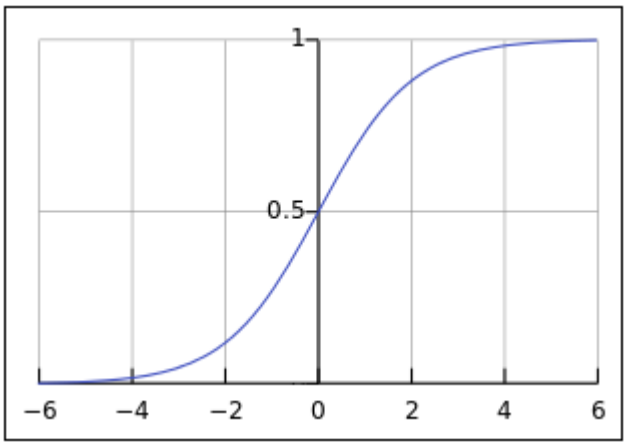
hesaplanır. Yapılan çalışmada da kullanılan lojistik regresyon analizinin detayları aşağıdaki bölümde verilmiştir.

4.2.2.1 Lojistik Regresyon analizi

LR bağımlı değişkenin nitel olduğu analizlerde kullanılır. Bu da LR yöntemini sınıflandırma problemleri için uygun kılmaktadır. LR, ikili sınıflandırma ve çoklu sınıflandırma problemi için kullanılabilir. LR yönteminin ikili sınıflandırma problemi için kullanılmasına ilişkin adımlar aşağıda verilmiştir. Bizim çalışmamızda da LR ikili sınıflandırma problemi için kullanılmıştır.

LR yöntemindeki amaç bağımsız değişkenler kullanılarak herhangi bir bağımlı değişkenin elde edilme olasılığını bulmaktır. Örneğin Y sınıf etiket değerini ifade etsin. İkili sınıflama probleminde Y'nin değeri 1 ve -1 olarak belirlenebilir. Bu durumda LR'de ki amaç $P(Y = 1|X = x)$ olasılığını, x ile ifade edilen bağımsız değişkenleri kullanarak bulabilmektir [48]. İlgili Y olasılığı en çok olabilirlik (maximum likelihood) yöntemi ile bulunabilir. $P(x)$'i, doğrusal regresyon analizi ile çözmeye çalışırsak sonuç 0 ve 1 arasında çıkmayacaktır. Bu sebebten problemin çözümleyebilmek adına sonucu 0 ve 1 arasına indirgeyebilecek bir yöntem gereklidir.

Lojistik fonksiyon Şekil 4.5'ten de [49] anlaşılacığı üzere x hangi değeri alırsa alsın sonuç olarak 0 ve 1 arasında çıktı üretecektir. İlgili çıktı kullanılarak etiketi belirli olmayan bir verinin etiketi belirlenebilecektir.



Şekil 4.5: Lojistik fonksiyon.

LR yönteminin algoritmasının basit olması ve kod karmaşıklığının az olması, büyük akan veri üzerine uygulanabilme ihitmalini arttırmışıtır.

5. AKAN (STRAEM) VERİ ANALİZİ VE TEKNOLOJİLERİ

5.1 Akan Veri Nedir?

Son zamanlarda teknolojinin hızla ilerlemesi verinin geldiği anda analiz edilmesi ihtiyacını doğurmuştur. Akan veriler, özelliği gereği sıralı ve değişik boyutlarda gelmektedir. Değişik boyutlarda ve düzensiz periyotlarla gelen akan veri, zaman ve donanım kısıtı altında anlık olarak analiz edilmeli ve raporlanmalıdır. Akan veri analizlerine, trafik kazalarını önlemek adına sensor verilerinin değerlendirilmesi, farklı makinelerden gelen farklı tipteki günlüklerin birleştirilmesi ve analizi, müşteriye lokasyon bazlı kampanya önerim sistemleri örnek olarak verilebilir. Akan verinin en önemli ve en zorlayacı özelliği verinin farklı tip ve boyutlarda gelmesidir. Günümüz teknolojileri akan veri ve zorluklarını aşacak şekilde güncellenmiştir [31]. Bu kapsamda, belleklerde gerek veri depolama kapasitesini arttıracak gerek ise veri kaybını önleyecek eklentiler getirilmiştir. Belleklerdeki boyut ve veri kaybı önleme yöntemlerinin etkisi ile veriyi bellek içerisinde saklayan veritabanları ve veri ambarı sistemleri geliştirilmiştir. Veriyi analiz etmekteki en etkin yöntemler veri madenciliği başlığı altında toplanmıştır. Bu bağlamda veri madenciliği yöntemleri de akan veri üzerine uyarlanarak bellek içi makine öğrenmesi kavramı doğmuştur. Bellek içi makine ögrenmesi yöntemlerinin sağlıklı çalışması için temiz ve algoritmaya uygun akan veriye ihtiyaç duyulmaktadır.

Gerçek hayattan alınan bir örnekte [32] 100 noktada bulunan 13.000 antenden anlık olarak saniyede 2 Terabyte boyutunda veri gelmektedir. Bu denli büyük verilerin saklanması günümüz teknolojisi ile bile çok yüksek maliyetli olup kimi zaman ise mümkün olamamaktadır. Bu nedenle gelen verilerin saklanmadan doğrudan analiz edilmesi gerekir. Saniyede 2 Terabyte gelen verinin anlık olarak analiz edilmesi tek bir makine ile mümkün olamayacağından gelen verinin dağıtık olarak analiz edilmesi gerekmektedir. Bu problemi çözebilmek adına bilişim dünyasında büyük akan veri analizi için birçok ürün ortaya çıkmıştır. Bunların başlıcaları Apache Spark Streaming ve Apache Storm'dur. Temelde her dağıtık akan veri yazılımı aynı

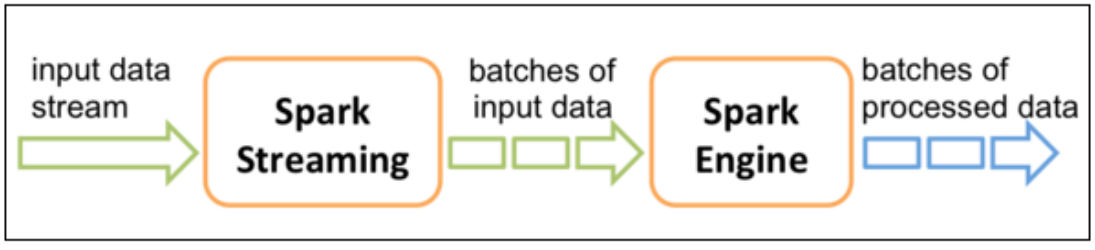
mantıkla çalışmaktadır. Veri herhangi bir kaynaktan dinlenir, paralel olarak sürekli çalışan hazır düğüm'ler tarafından işlenir ve çıktı yine istenilen yere yazılır [33]. Apache Spark Streaming ve Apache Storm teknolojilerinin detayları ileriki bölümde verilmiştir.

5.2 Akan Veri Teknolojileri

5.2.1 Apache Spark Streaming

Apache Spark Streaming son zamanlarda en yaygın olarak kullanılan akan veri analizi teknolojisidir. Bunun sebebi Apache Spark Streaming teknolojisinin diğer akan veri analizi ürünlerinden farklı olarak Apache Spark desteği ile hem toplu hem de anlık analizler için aynı kodu çalıştırabiliyor olmasıdır. Dolayısıyla toplu iş için yazılmış olan Apache Spark kodu akan veri analizi için uyarlanabilir ve dakikalar için de akan veri analizi ile çalışabilir hale getirilebilir. Aynı şekilde Spark MLlib ile akan veri üzerinde makine öğrenmesi gerçeklenebilir ya da Spark SQL ile akan veri üzerinde sürekli sorgular atılabilir.

Apache Spark Streaming'in mimarisine bakıldığı zaman birçok kaynaktan veriyi okuyabildiği görülmektedir. Bu kaynaklara örnek Kafka, Flume, ZeroMQ, Kinesii ya da TCP verilebilir. Apache Spark Streaming genel olarak bahsi geçen kaynaklardan almış olduğu akan veriyi parçalara ayırarak Apache Spark teknolojisine gönderir. Apache Spark, kendisine iletilen parça verileri tıpkı toplu veri işlemede olduğu gibi çalıştırarak çıktı verisini istenilen ortama yazar. (Şekil 5.1 [34])

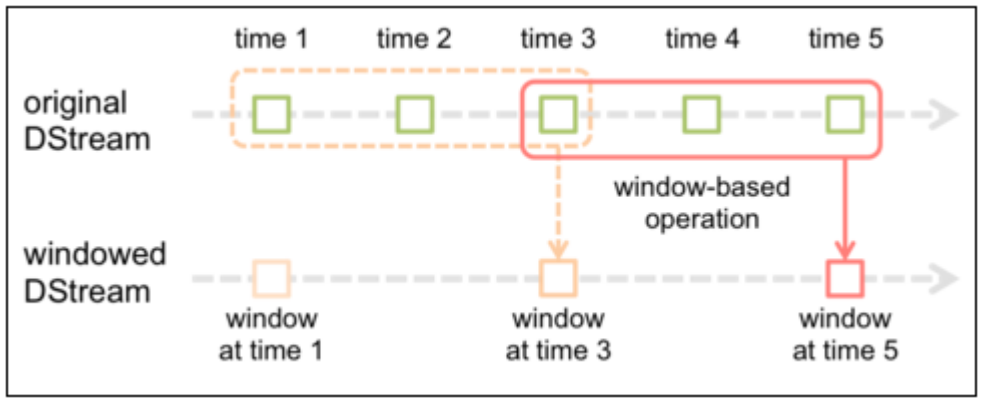


Şekil 5.1: Apache Spark Streaming çalışma prensibi.

Apache Spark Streaming tüm ayrıcalıklarını verileri parçalara bölme ve toplu olarak Apache Spark ortamında çalıştırmaktan alır. Parçalara ayrılan verilerin Apache Spark tarafından işlenmesi sayesinde hata kotarımı ya da kaynakların veri yoğunluğuna göre paylaştırılması doğrudan Apache Spark tarafından yönetilir. Böylelikle Apache Spark Stremaing rakiplerine oranla çok daha güçlü bir yapıya bürünmektedir.

Apache Spark Streaming ile okunan canlı veri Dstream adı verilen programlama modelinde tutulur. İlgili Dstream, Apache Spark Core kütüphaneleri tarafından işlenebilmesi için parçalara ayırılır ve Apache Spark bölümünde de anlatıdığı üzere Apache Spark tarafından programlama modeli olarak kullanılan RDDs'lere dönüştürülür. Veriyi işlemek adına kullanılan Actions ve Transactions'lar bu RDDs'ler üzerinde uygulanırlar.

Apache Spark Streaming, kayan pencere (sliding window) kavramını da desteklemektedir. Pencere operasyonları, bir zaman aralığı belirleyip RDDs'leri birleştirerek, ilgili zaman aralığında gelen RDDs'ler üzerinde işlem yapılmasına olanak sağlar.



Şekil 5.2: Apache Spark Streaming kayan pencere.

Temel olarak kayan pencere işlemlerinde, pencere uzunluğu (window length) ve hangi zaman aralıklarında bu kayan pencerenin yaratılması bilgisi yer alır. Şekil 5.2' ten [34] örneklemek gerekirse, her iki saniyede kaynaktan gelen son üç RDDs birleştirilerek Apache Spark Core fonksiyonlarına iletilecektir.

Tüm akan veri analizi işlemlerinde en önemli problem performanstır. Apache Spark Streaming, performans analizi için birçok parametre içermektedir. Bunların başlıcaları, verinin kaynaktan kaç paralellik ile alınacağı ve yine işleme sırasında kaç bilgisayarda hangi kaynaklarla çalışacağının ayarlanmasıdır. Apache Spark Streaming, bu tez çalışmasının da temelini oluşturmaktadır.

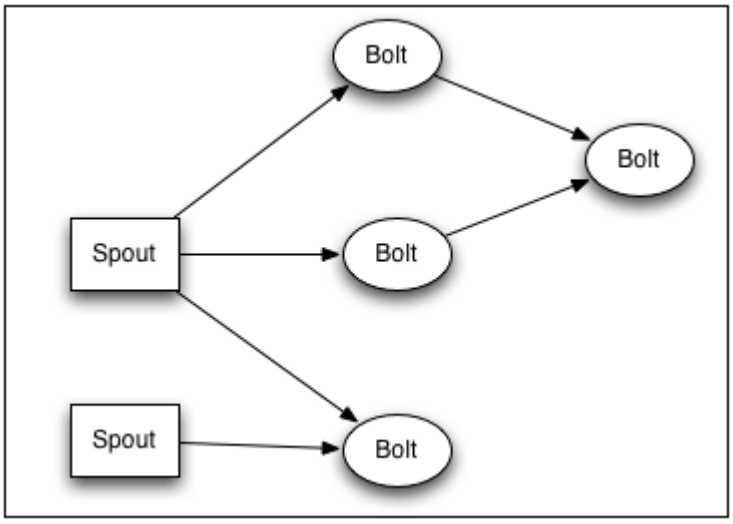
5.2.2 Apache Storm

Apache Storm teknolojisi de Apache Spark Streaming'te olduğu gibi büyük ve yüksek hızda akan veriyi paralel olarak işlemek için ortaya atılmış bir projedir. Apache Storm, akan veri işlemede hızlı işleme yeteneği ile ön plana çıkmaktadır.

Öyle ki, dağıtık sistemlerde tek bir düğüm üzerinde, saniyede 1 milyon veri işleme kapasitesine sahip bir teknolojidir [35]. Apache Storm, temelde Apache Hadoop'un akan veri analizi işleri için özelleştirilmiş hali olarak düşünülebilir.

Apache Storm teknolojisi kuyrukta yer alan akan verinin sadece bir kez işleneceğini garanti altına alır. Apache Storm, hata kotarım mekanizması olarak hata alan düğüm'ü otomatik olarak tekrardan başlatır ve işleme devam eder. Apache Spark Streaming teknolojisi farkını burada ortaya koymaktadır. Apache Spark Streaming'te hata olması durumunda yeniden başlatma beklemeksizin işleme devam edilir.

Apache Storm Clojure programlama dili ile geliştirilmiş olup mimarisi, *Spouts* ve *Bolts* kavramlarını içeren yönlü çevrimsiz çizgedir (Directed Acyclic Graph (DAG)). Spouts kavramı verilerin kaynaklardan alınması için gerekli kodları içerirken Bolts ise akan verinin işlenmesi ve sonuçlarının yazılması için gerekli olan kodları içerir. Spouts ve Bolts kodları dağıtık bilgisayarlar üzerinde DAG çizge konfigrasyonu baz alınarak birlikte çalışırlar.



Şekil 5.3: Apache Storm mimarisi.

Apache Storm işi başlatıldığında, Şekil 5.3'ten de [36] anlaşılacağı üzere, ilk olarak akan veri Spouts'lar tarafından ele alınır ve verinin işleneceği Bolts'lara gönderilir. Bolts'lar, kodlanan işi gerçekledikten sonra çıktıları başka bir Bolts'a gönderebileceği gibi herhangi bir veritabanı ya da dosya sistemine de yazabilirler.

Apache Storm'un en önemli özelliği Spouts ve Bolts'ların programlama dili bağımsız her hangi bir dilde yazılabilmesidir. Verileri bilindik kaynaklardan almak üzere birçok dilde Spouts'lar geliştirilmiştir. Dolayısıyla Twitter, Kafka vb. gibi

bilindik kaynaklardan veriyi almak için ekstra kod yazmadan geliştirme yapılabilir. Apache Storm'un bir diğer önemli özelliği ise kolay genişletilebilir olmasıdır. Bu sayede Apache Storm'un üzerinde çalıştığı dağıtık bilgisayarların sayısı arttırılarak daha büyük akan veriler için analiz sistemleri gerçeklenebilir. Apache Storm teknolojisi her ne kadar hızı ile ön plana çıkıyor olsa da hata kotarım mekanizması ve Apache Spark Streaming'in bütünleşik yapısı, yapılan çalışmada Apache Spark Streaming seçilmesinde etkili olmuştur.

6. GELİŞTİRİLEN YÖNTEM

Yapılan bu çalışmada öncelikli olarak akan büyük veri üzerinde sınıflandırma problemini çözmek esas alınmıştır. Akan eğitim veri kümesi ile gerçek zamanlı olarak kurulan model güncellenmekte, aynı şekilde akan etiketsiz veriler için ilgili model kullanılarak etiketlendirme yapılmaktadır.

Sınıflandırma probleminin hangi teknoloji ile akan büyük veriye uyarlaması gerektiğini anlamak adına büyük veri teknolojileri incelenmiştir. Yapılan araştırmalar sonucunda Apache Hadoop büyük veri teknolojisinin disk tabanlı olmasından kaynaklı yavaşlık probleminden ötürü akan büyük veri uygulamalarına uygun olmadığı kanısına varılmış ve çalışmada Apache Spark büyük veri teknolojisi tercih edilmiştir.

LR ve DVM sınıflandırma algoritmalarının performanslarını inceleyebilmek adına SAS ile deneyler yapılmış SAS'in tek makine üzerinde büyük veri ile çalışamadığı noktada deneyler Apache Spark teknolojisi ile tekrarlanmıştır. Yapılan bu deneysel araştırmaların sonucunda DVM algoritmasının ROC değerlerinin LR yöntemlerine göre daha başarılı çıktığı gözlemlenmiştir.

DVM yönteminin yüksek başarım oranı DVM yönteminin büyük akan veri üzerine uyarlanmasına ışık tutmuştur. Bunun üzerine, Apache Spark MLlib ve Apache Spark Streaming teknolojileri kullanılarak akan büyük veri ile çalışabilecek DVM yöntemi geliştirilmiş ve sonuçlar Apache Spark Streaming üzerinde hali hazırda yer alan LR yöntemi ile karşılaştırılmıştır.

6.1 Veri Kümesi, Kullanılan Kaynaklar ve Algoritma

Deneyler için iki veri kümesi kullanılmıştır. İlk veri kümesi[1] kredi başvurularında yapılan örnek sahtecilik işlemlerini içermektedir. Modele giren değişkenler başvuru yapan kişilere ait bilgiler olmak ile beraber hedef değişken başvurunun sahte olup

[1] İlgili veri kümesi özel bir kuruma ait olup paylaşılması mümkün değildir.

olmadığını ifade etmektedir. 1 değeri başvurunun sahte olduğunu, 0 ise gerçek bir başvuru olduğunu göstermektedir. Tüm girdi değişkenleri nümerik değer almaktadır.

Kullanılan eğitim veri kümesi 870741 kayıt ve 445 özellikten oluşmakta ve yaklaşık 1 GB boyutundadır. Test veri kümesi ise 373174 kayıttan oluşmaktadır. Kullanılan eğitim kümesinde yer alan hedef değişken dağılımı Çizelge 6.1'de verilmiştir.

Çizelge 6.1: Eğitim veri kümesi hedef değişken dağılımı.

Hedef Değişken	Veri Kümesinde Görülme Sayısı	Yüzde Dağılımı
1	43534	%5
0	827207	%95

Geliştirilen akan büyük veri DVM yöntemi herkes tarafından erişilebilir olan KDD Cup 2010 educational data mining competition adlı yarışmanın veri kümesi üzerine [53] uyarlanmıştır. İlgili veri kümesi, online olarak yürütülen cebir derslerinde sorulan problemlerle öğrencilerin etkileşim günlüğünü içermektedir. Veri kümesinde yer alan her bir satır herhangi bir öğrencinin ilgili cebir ünitesindeki $\boldsymbol{x}$ problemine verdiği cevap sırasında gerçeklediği adımların günlüklerini içermektedir (cevap süresi, ipucu alıp almadığı vb.). Hedef değişken bilgisi ise öğrencinin sorulan cebir sorusuna bir kerede cevap verip veremediği bilgisi ile belirlenmiştir. Hedef değişken, öğrenci ilgili soru ile ilk karşılaştığında doğru cevap vermiş ise 1 aksi halde 0 olarak atanmıştır. Veri kümesinde toplamda 8918055 satır kayıt bulunmaktadır.

Deneyler SAS tarafında SAS programlama dili ile, Apache Spark tarafında ise Scala programlama dili ile gerçeklenmiştir. Apache Spark kodları CDH 5.4 versiyonu ile 6 paralel düğüm üzerinde çalıştırılmıştır. Herbir düğüm 2 X 18 Core Intel 2.3 GHz işlemci, 128 Gigabyte RAM içermektedir. SAS üzerinde yapılan çalışmalar 4 thread X 9 Core PowerPC_POWER7 3.8 GHz işlemci, 50 Gigabyte RAM kaynaklı bir makine üzerinde gerçeklenmiştir. Önceki bölümlerde anlatıldığı üzere Apache Spark paralel programlama modelidir.

Çalışmadan kullanılan akan büyük veri DVM'e ait algoritma, Algoritma 1'deki gibidir.

Algoritma1: SGD ile DVM

Başla *w* başlangıç ağırlıkları belirlenir
while (*i*!=*T*) (*T* belirlenen iterasyon sayısı) **do**
  öğrenme katsayısı (*a*) hesaplanır.
  for (j=1 **until** örneklemBoyutu) **do**
    j. örnek için subgradient (*g*) hesaplanır.
    model ağırlıkları güncellenir. (w_{j+1}=$w_j - a * g$)
    j++
  end for
  i++
  currHata, yeni *w* model ağırlıkları kullanılarak hesaplanır.
  if (currHata <= hataEsikDegeri) **do**
    döngünden çık
  end if
end while

6.2 Çalışma Adımları

Çalışmanın ilk adımı olarak SAS veri analizi ürünü ile, kredi sahtecilik veri kümesi kullanılarak DVM ve LR yöntemlerinin performansları incelenmiştir.

Öncelikli olarak orjinal veri kümesi ile deneyler yapılmış olup sonrasında performansı arttırabilmek adına veri kümesinin özelliklerini sınıflandırarak (coarse class) ve özellik değerlerini ölçeklendirerek aynı deneyler tekrarlanmış ve performanstaki değişimler gözlemlenmiştir.

SAS ile yapılan, LR ve DVM yöntemlerinde hatanın minimize edilmesi için iterasyon sayısı 50, hata eşik değeri 0.0001 olarak belirlenmiştir. SAS 'in hata minimizasyonu için hangi algoritmayı kullandığı bilinmemektedir. Öte yandan SAS LR Stepwise yönteminin denenme amacı, eğitim kümesinde yer alan özellikleri, tahminlemeye katkı oranını göz önüne alarak azaltmak ve böylelikle tahminleme süresini kısaltmaktır. Tahminlemeye etkisi fazla olan özellikleri bulabilmek adına Stepwise selection yöntemi seçilmiştir, bu yöntem ile beraber bazı özellikler modele girerken bazı özellikler modelden çıkmıştır. LR Stepwise çalışmasında süreç tamamlandığında 49 özelliğin modele girdiği gözlemlenmiştir. Stepwise yöntemi özellik seçimi ile beraber beklenildiği gibi tahminleme süresini kısaltmıştır öte yandan model seçimi işlemi sebebi ile eğitim süresi uzamıştır.

SAS kullanılarak kredi sahtecilik veri kümesi üzerinde yapılan çalışmanın ROC sonuçları Çizelge 6.2'de verilmiştir.

Çizelge 6.2: SAS , LR ve DVM kredi sahtecilik veri kümesi için ROC değerleri.

	Veri Kümesi	LR	LR Stepwise	DVM
Orjinal Veri	Eğitim	0,830	0,801	bellek hata
Orjinal Veri	Test	0,791	0,775	bellek hata
Özellikleri Sınıflandırılmış Veri	Eğitim	0,843	0,813	bellek hata
Özellikleri Sınıflandırılmış Veri	Test	0,818	0,783	bellek hata
Özellikleri Ölçeklendirilmiş Veri	Eğitim	0,821	0,799	bellek hata
Özellikleri Ölçeklendirilmiş Veri	Test	0,788	0,771	bellek hata

LR yöntemi yüksek ROC değerlerinde çıktılar üretebilirken DVM yöntemi yetersiz bellek hatası alarak tamamlanamamıştır. LR ve DVM yöntemlerini karşılaştırabilmek adına aynı veri kümesinden örneklem alınarak deneylerde ilgili veri kümesinin yaklaşık yarı boyutu kullanılmıştır.(Çizelge 6.3)

Çizelge 6.3: Örneklem sonrası, SAS , LR ve DVM kredi sahtecilik veri kümesi için ROC değerleri.

	Veri Kümesi	LR	LR Stepwise	DVM
Orjianl Veri	Eğitim	0,766	0,735	0,785
Orjinal Veri	Test	0,738	0,719	0,767
Özellikleri Sınıflandırılmış Veri	Eğitim	0,778	0,749	0,801
Özellikleri Sınıflandırılmış Veri	Test	0,751	0,732	0,792
Özellikleri Ölçeklendirilmiş Veri	Eğitim	0,764	0,732	0,782
Özellikleri Ölçeklendirilmiş Veri	Test	0,733	0,715	0,766

Orjinal veri kümesi üzerinde DVM yöntemini çalıştırmak adına Apache Spark MLLib ile dağıtık veri analizi yapılmış ve DVM'in büyük veri üzerinde LR'ye göre daha iyi başarım oranına sahip olduğu gözlemlenmiştir. Apache Spark üzerinde yapılan çalışmalarda iterasyon sayısı olarak 200, hata eşik değeri 0,0001 olarak belirlenmiştir.

Hata fonksiyonunu minimize etmek adına sık kullanılan yöntemler arasında Gradient Descent (GD) ve SGD yöntemleri vardır. GD yönteminde amaç kendisine verilen

fonksiyondaki minumum noktayı bulmaktır. GD yöntemi kendisine verilen fonksiyondaki minumum noktayı sağlayan fonksiyon ağırlıklarını bulmak için seçilen başlangıç noktası değerlerinden başlayarak ilgili iterasyondaki fonksiyon ağırlıklarını $(w_i, i \in 1,..n)$ bir önceki iterasyonda hesaplanan fonksiyon ağırlıklarından Denklem 6.1'i kullanarak hesaplar. Fonksiyondaki her ağırlığın eğim değeri fonksiyonun o ağırlık değerine göre türevinin alınması ile hesaplanır. Denklem 6.1'de $\frac{\partial f}{\partial w_{\text{önceki } i}}$ değeri ilgili fonksiyonun $w_{\text{önceki } i}$ ağırlığına göre türevini *a* ise öğrenme katsayısını ifade etmektedir.

$$w_{yeni\ i} = w_{\text{önceki } i} - a \frac{\partial f}{\partial w_{\text{önceki } i}} \tag{6.1}$$

GD yöntemi ya kendisine verilen sayıda iterasyonu tamamladıktan sonra ya da belirlenen hata aralığını sağlayan $w_i, i \in 1,..n$ katsayıları bulunduğunda sonlanır.

Sınıflandırma yöntemlerinde GD yönteminin amacı, verilen herhangi bir başlangıç noktasından öğrenme katsayısını kullanarak kendisine verilen hata fonksiyonunu minimize eden değeri bulmaktır. SGD yöntemi de GD yöntemi ile aynı prensipte çalışmaktadır. SGD yöntemi ile GD yöntemi arasındaki fark; GD yönteminde hatayı minimize etmek adına model katsayı güncellemesi esnasında tüm veri kümesi ele alınırken SGD yönteminde tüm veri kümesi değil, sadece ilgili tek kaydın ele alınmasıdır.

Yapılan çalışmada kullanılan hinge loss hata fonksiyonu minimizasyonu için SGD yöntemi kullanılmıştır. SGD yönteminde yukarıda bahsedildiği üzere rastgele model ağırlıkları ile başlanarak her iterasyonda hata oranını azaltacak şekilde model ağırlıkları güncellenir. SGD yöntemi için Hinge loss hata fonksiyonun *subgradient*'i [56] alınabilir. Yapılan çalışmada Denklem 6.2'de yer alan subgradient fonksiyonu eğim bulmak için kullanılmıştır. Denklem 6.2'de w^T fonksiyon ağırlıkları matrisinin transpozunu, x_i *i*. özellik değerini y ise ilgili girdinin etiket bilgisini ifade etmektedir.

$$\begin{matrix} -yx & eğer\ y(w^T x) < 1 \\ 0 & diğer\ tüm\ durumlar \end{matrix} \tag{6.2}$$

Yapılan çalışmada SGD seçilip GD seçilmemesinin nedeni akan veri problemlerinde hızlı aksiyon alma gereksinimidir. GD yönteminde hata SGD'ye göre daha fazla

minimize edilse de çalışma süresi daha uzun sürmektedir. SGD yönteminde öğrenme katsayısı $a = \frac{1}{1+mevcutIterasyon}$ olarak hesaplanmıştır. Öğrenme katsayısı yüksek değerde seçilirse daha kısa sürede hata minimize edilirken, düşük değerde seçilirse hata minimizasyonu daha fazla süre alacak fakat daha küçük hata oranı elde edilecektir.

Apache Spark ile kredi sahtecilik veri kümesi üzerinde yapılan çalışmanın sonuçları Çizelge 6.4'te verilmiştir.

Çizelge 6.4: Spark, DVM ve LR kredi sahtecilik veri kümesi için ROC değerleri.

	Veri Kümesi	LR	DVM
Orjianl Veri	Eğitim	0,719	0,741
Orjinal Veri	Test	0,701	0,725
Özellikleri Sınıflandırılmış Veri	Eğitim	0,735	0,755
Özellikleri Sınıflandırılmış Veri	Test	0,716	0,736
Özellikleri Ölçeklendirilmiş Veri	Eğitim	0,713	0,736
Özellikleri Ölçeklendirilmiş Veri	Test	0,698	0,719

Yapılan karşılaştırmalar, Apache Spark ile yapılan sahtecilik veri kümesi için DVM yönteminin LR'ye göre zaman ve doğruluk bakımından daha performanslı olduğunu göstermiştir. Çalışma sürelerine bakıldığında Apache Spark büyük veri yönteminin büyük veri analizinde son derece performanslı olduğu gözlemlenmiştir. (Çizelge 6.5)

Çizelge 6.5: Spark ve SAS , DVM ve LR kredi sahtecilik veri kümesi için eğitim süreleri.

	SAS LR	SAS LR Stepwise	SAS DVM	SPARK LR	SPARK DVM
Orjianl Veri	18d.30sn	9s.42d	-	3d.54sn	4d.11sn
Özellikleri Sınıflandırılmış Veri	17d.32sn	8s.35d	-	2d.32sn	2d.57sn
Özellikleri Ölçeklendirilmiş Veri	18d.25s	9s.25d	-	3d.42sn	4d.05sn

DVM yönteminin başarım oranı gözlemlendikten sonra Apache Spark Streaming ve Apache Spark MLlib teknolojileri kullanılarak akan büyük veri için DVM algoritması kodlanmıştır.

Apache Spark üzerinde geliştirlen DVM ve Apache Spark üzerinde var olan LR yöntemleri 1 Gigabyte boyutunda akan kredi sahtecilik veri kümesi kullanılarak eğitilmiş ardından yaklaşık 0,.42 Gigabyte boyutunda akan test verisi ile test edilmiştir. Akan veri DVM yönteminin çıktısı Apache Spark teknolojisi içerisinde var olan akan veri LR algoritması ile karşılaştırılarak performans ölçümü ayrıca yapılmıştır. (Çizelge 6.6)

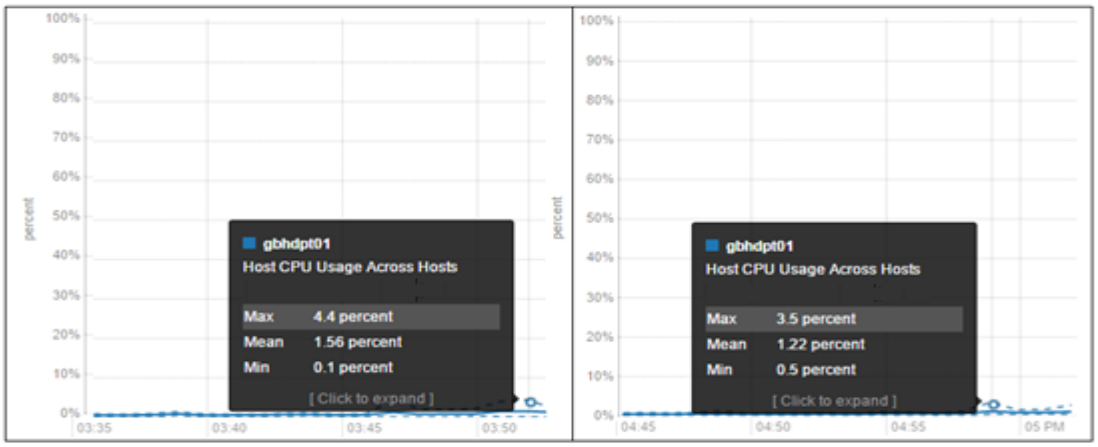
Çizelge 6.6: Akan veri Spark LR ve DVM yöntemeleri kredi sahtecilik veri kümesi için karşılaştırması.

	Veri Kümesi	LR ROC	DVM ROC	LR Süre	DVM Süre
Orjinal Veri	Eğitim	0,718	0,739	4d.48sn	5d.12sn
Orjinal Veri	Test	0,701	0,724	48sn	47sn
Özellikleri Sınıflandırılmış Veri	Eğitim	0,736	0,759	3d.25sn	4d.01sn
Özellikleri Sınıflandırılmış Veri	Test	0,714	0,738	41sn	42sn
Özellikleri Ölçeklendirilmiş Veri	Eğitim	0,711	0,737	4d.32sn	5d.04sn
Özellikleri Ölçeklendirilmiş Veri	Test	0,699	0,718	47sn	49sn

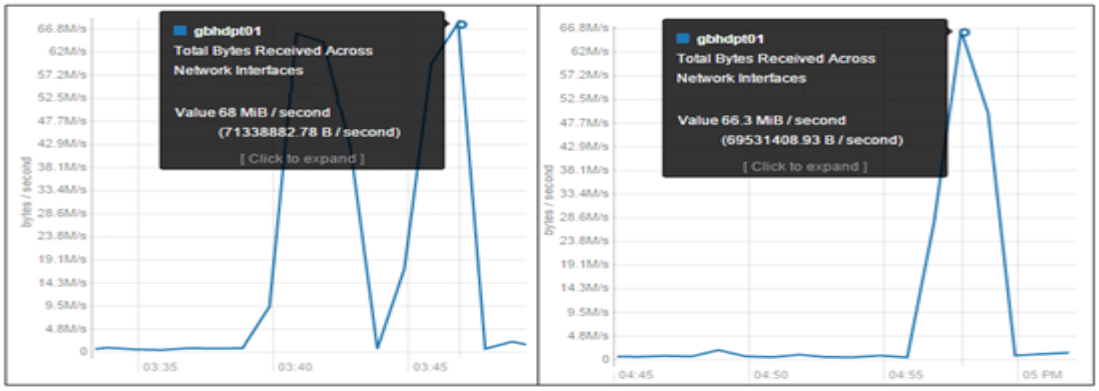
Apache Spark bahsedildiği üzere paralel programlama modelidir. Paralel programlama modellerinde bir diğer önemli unsur ise programın paralel olarak çalışırken dağıtık bilgisayarlara getirmiş olduğu yüktür. Geliştirilen akan veri DVM yönteminin dağıtık bilgisayardaki çalışma performansı ve karşılaştırma yapabilmek adına Apache Spark'ta mevcut akan LR algoritmasının performansı detaylıca incelenmiştir.

Yapılan incelemeler sonucunda hem LR hem de DVM yöntemlerinin dağıtık bilgisayarlara getirdiği yük benzer davranış göstermiştir. (Şekil 6.1, Şekil 6.2, Şekil 6.3) HDFS üzerinde yer alan I/O'nun anlık ve bir kez olduğu gözlemlenmiştir. (Şekil 6.3) Yukarıdaki bölümlerde bahsedildiği üzere Apache Spark teknolojisi bellek içi çalışma prensibi gerektirdiği için bu beklenen bir durumdur. Veri,

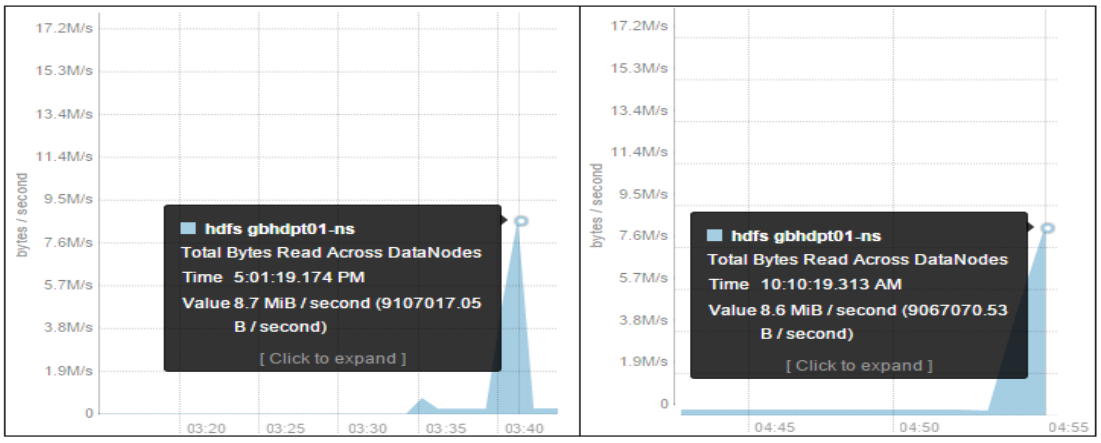
HDFS'ten bir kez belleğe çekildikten sonra tüm işlemler bellek içersinde hızlı bir şekilde tamamlanmıştır.



Şekil 6.1: Apache Spark akan DVM (sol taraf) ve LR (sağ taraf) Merkezi İşlem Birimi (MİB) kullanımı karşılaştırması.



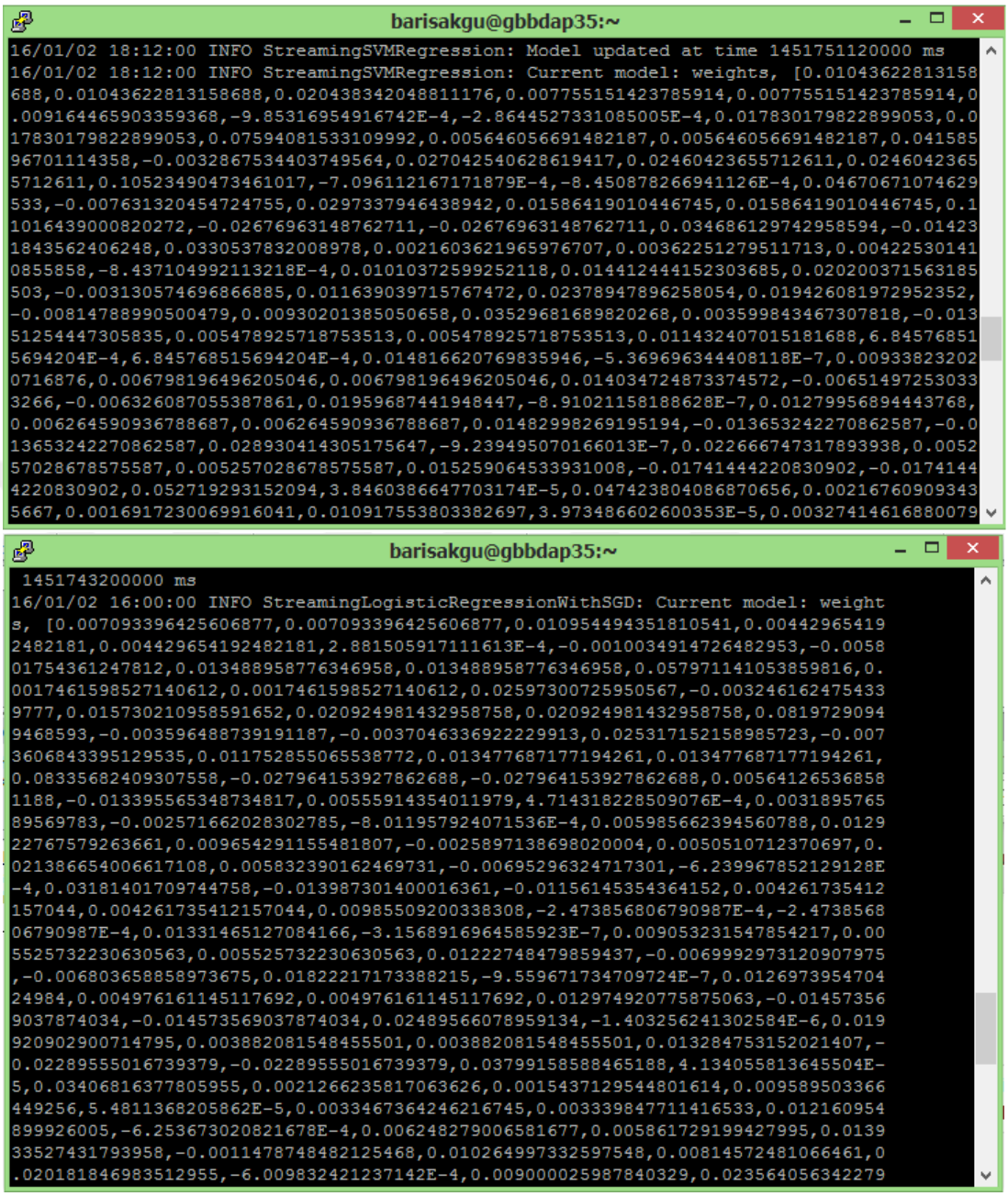
Şekil 6.2: Apache Spark akan DVM (sol taraf) ve LR (sağ taraf) ağ karşılaştırması.



Şekil 6.3: Apache Spark akan DVM (sol taraf) ve LR (sağ taraf) HDFS I/O karşılaştırması.

Yapılan çalışma sonucunda anlık olarak gelen verilere göre model katsayıları güncellenmektedir. Yapılan çalışmada model katsayılarının anlık olarak nasıl

değiştiğini gösteren LR ve DVM yöntemlerine ait ekran görüntüleri Şekil 6.4'te verilmiştir.



Şekil 6.4: Apache Spark akan LR ve DVM ekran çıktıları.

Elde edilen sonuçlar doğrultusunda, geliştirilen akan büyük veri DVM yöntemi KDD Cup 2010 educational data mining competition adlı yarışmanın veri kümesi [53] üzerine uyarlanmıştır. Yarışmanın amacı öğrencilerin geçmiş cebir dersleri başarım günlüklerine bakarak gelecek cebir derslerinde başarılı olup olamayacağını tahminlemektir. İlgili yarışmada test kümesinin etiket bilgileri paylaşılmamıştır. Katılımcılar test kümesi için buldukları etiketleri sisteme yükleyerek RMSE

sonuçları elde etmiş ve buna göre sonuçlar değerlendirilmiştir. Katılımcılar modellerini gerçeklerken verilen veri kümesini eğitim ve test olarak ikiye ayırmıştır. Her bir cebir ünitesinin son problemine öğrenciler tarafından verilen cevapların günlükleri test kümesi, geri kalan soruların cevap günlükleri ise eğitim kümesi olarak ele alınmıştır.

Bu çalışmada yarışmanın birincisi olan National Taiwan Üniversite'sinin yaratmış olduğu eğitim ve test kümesi kullanılmıştır [54]. Buna göre 8918055 satırlık yarışma veri kümesi 8407752 (2.5 GB) satırlık eğitim ve 510303 (151 MB) satırlık test kümesi olarak ikiye ayrılmıştır. Geliştirilen akan büyük veri DVM yöntemine eğitim ve test veri kümeleri akan veri olarak yüklenmiştir.

Çizelge 6.7: KDD Cup 2010 veri kümesi için ROC değerleri ve Çalışma Süreleri

Veri Kümesi	ROC	Çalşma Süre
Eğitim	0,681	11d.12sn
Test	0,662	26sn

Ayrıca yapılan yarışmaya katılım sağlayan yöntemler ile geliştirilen akan veri DVM yönteminin doğruluk performasını karşılaştırabilmek adına RMSE hesaplanması da yapılmıştır. Eğitim kümesi kullanılarak geliştirilen yöntemin test kümesi üzerindeki RMSE değeri 0,396 olarak hesaplanmıştır. Yarışma sonuçları[2] incelendiğinde yarışma birincisinin RMSE değeri 0,273 iken geliştirilen akan büyük veri DVM yönteminin 0,396'lık RMSE değeri ile 29. sırada yer alabilecek olduğu gözlemlenmiştir.

[2] KDD Cup 2010 yarışma sonuçları https://pslcdatashop.web.cmu.edu/KDDCup/results_full.jsp .

7. ÇALIŞMANIN SONUCU VE GELECEK PLANLARI

Yapılan çalışmanın sonucunda akan büyük veri üzerinde ikili sınıflandırma problemini ele alabilecek DVM yöntemi Apache Spark Streaming ve Apache MLlib teknolojileri kullanılarak gerçeklenmiştir.

Yapılan çalışma büyük akan veri yöntemine uyarlanmadan önce DVM yönteminin LR yöntemi kadar etkin olup olmadığını anlamak adına SAS ürünü ile gerçek zamanlı olmayan deneyler yapılmıştır. Yapılan deneylerin sonuçlarında DVM yönteminin karmaşıklık derecesinin LR yönteminden fazla olduğu gözlemlenmiştir. Öte yandan DVM yönteminin çıktılarının LR yönteminin çıktılarına oranla daha doğru olduğu belirlenmiştir. Çalışma kapsamında gerçeklenen araştırmalar doğrultusunda DVM yönteminin akan büyük veri üzerine uyarlanabilmesi için paralel büyük akan veri teknolojisine ihtiyaç duyduğu saptanmıştır.

Bilişim dünyasında yer alan büyük veri ve akan büyük veri teknolojileri DVM yöntemini uyarlayabilmek adına detaylıca incelenmiştir. Akan büyük veri problemlerinin gerçek zamanlı ve kısa sürede tamamlanma gereksinimleri, bu çalışma için en uygun teknolojinin Apache Spark Streaming teknolojisi olduğuna işaret etmiştir. DVM yöntemi Apache Spark Streaming teknolojisi üzerine uyarlanırken akan verinin zorlukları göz önünde bulundurulmuştur. Akan veri analizi ve özellikle de büyük akan veri analizi gereksinimlerini anlayabilmek adına akan veri, büyük veri ve büyük akan veri gereksinimleri detaylıca incelenmiş, bu konuda yapılan çalışmalar ve çıktıları göz önüne alınmıştır.

Geliştirilen akan büyük veri DVM yöntemi ile, Apache Spark Streaming teknolojisi üzerinde var olan akan büyük veri LR yönteminden daha etkin ROC sonuçları elde edilirken, süre ve dağıtık sisteme getirilen yükün her iki yöntemde de paralel olduğu gözlemlenmiştir. Elde edilen sonuçlar ikili akan veri sınıflandırma probleminde DVM yönteminin de etkin bir yöntem olduğuna işaret etmektedir.

Konuyla ilgili olarak gelecek dönemde yapılabilecek bir çalışma, geliştirilen ikili sınıflandırma yönteminin ikiden fazla sınıf için de gerçeklenmesi olabilir. Böylelikle

akan veri sınıflandırma problemi üzerinde genel ve etkin bir yöntem geliştirilmiş olacaktır.

KAYNAKLAR

[1] **Twitter** V*ikipedia.* Erişim: 23 Kasım, 2015, https://en.wikipedia.org/wiki/Twitter

[2] **Golab, L., Özsu, M.** (2003). Issues in data stream management, *ACM SIGMOD Record*, (pp.5-14).

[3] **Koudas, N., Srivastava D.** (2003). Data Stream Query Processing: A Tuorial, *In : Proceedings of the 29th VLDB Conference,* (pp.1149-1149).

[4] **Gaber, M. M., Krishnaswamy, S. & Zaslavsky, A**. (2005). Mining Data Streams : A Review, *ACM SIGMOD Record*, (pp.18-26).

[5] **Jiawei, H., Haixun, W., Wei, F., Phlip, S.Y** (2003). Mining concept-drifting data streams using ensemble classifiers. *In : Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining,* (pp.226-235). .

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

[7] **Gao, J., Fan, W., Han, J. (**2007). On appropriate assumptions to mine data streams. *In : Proceedings of seventh IEEE international conference on data mining (ICDM).* (pp.143-152). Omaha, USA.

[8] **Grossi, V., Turini, F.** (2012). Stream mining: a novel architecture for ensemble-based classification, *Knowledge and Information Systems*. (pp.143-152).

[9] **Kolteri, J., Maloof, M**. (2005). Using additive expert ensembles to cope with concept drift. *In: Proceedings of international conference on machine learning (ICML).* (pp.449-456). Bonn, Germany.

[10] **Mohammad, M. M., Clay, W., Jing, G., Latifur, K., Jiawei, H., Kevin, W. H., Nikunj, C. O. ASTM** (2009). Facing the reality of data stream classification:coping with scarcity of labeled data, *Knowledge and Information Systems*.

[11] **Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.** (2010). MOA: Massive Online Analysis, *In : The Journal of Machine Learning Research*. (pp.1601-1604).

[12] **Gianmarco De Francisci, M.** (2013). SAMOA: a platform for mining big data streams. *In: Proceeding WWW 13 Companion Proceedings of the 22nd International Conference on World Wide Web.* (pp.777-778). Geneva, Switzerland.

[13] **Data Never Sleeps Report,** Erişim: 19 Kasım, 2015, https://www.domo.com/learn/infographic-data-never-sleeps

[14] **What is Big Data**, Erişim: 20 Kasım, 2015, http://www.sas.com/en_us/insights/big-data/what-is-big-data.html

[15] **De, M., Girmaldi, M., Greco, A.** (2015). What is big data? A consensual definition and a review of key research topics. *In: AIP Conference Proceedings 1644*. (pp.97-104).

[16] **Kalil, T.** (2012). *Big Data is a Big Deal*. Retrieved 26 Kasım 2015, from https://www.whitehouse.gov/blog/2012/03/29/big-data-big-deal

[17] **Gonzolez, J.** (2012). *Parallel and Distributed Systems for Probabilistic Reasoning*. Retrieved 5 Ağustos 2015, from http://www.cs.berkeley.edu/~jegonzal/jegonzal_thesis.pdf

[18] **Şenbalcı, C.** (2013). *Big data platform development with a Telecom DSL* (Master's thesis). Available from Ulusal Tez Merkezi (Tez No: 333175)

[19] **Dean, J., Ghemawat, S.** (2008). Mapreduce: simplified data processing on large clusters, *ACM SIGMOD Record*, (pp.107-113).

[20] **Zaharia, M.** (2013). *An Architecture for Fast and General Data Processing on Large Clusters* (Doctoral dissertation). Retrieved from http://digitalassets.lib.berkeley.edu/etd/ucb/text/Zaharia_berkeley_0028E_14121.pdf

[21] **Sachin, P.** (2014). *An introduction to Apache Hadoop for big data.* Retrieved from https://opensource.com/life/14/8/intro-apache-hadoop-big-data

[22] **What is Apache Hadoop?**, Erişim: 18 Kasım, 2015, http://hortonworks.com/hadoop/

[23] **HDFS Architecture Guide**, Erişim 18 Kasım, 2015, https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html

[24] **Apache Hadoop MapReduce Tutorial**, Erişim 10 Eylül, 2015, https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html

[25] **Apache Hadoop, What is it and Why does it matter**, Erişim 12 Eylül, 2015, http://www.sas.com/en_us/insights/big-data/hadoop.html

[26] **Apache Spark**, Erişim 16 Ekim, 2015, http://spark.apache.org/

[27] **Apache Spark Wikipedia**, Erişim 16 Ekim 2015, https://en.wikipedia.org/wiki/Apache_Spark

[28] **Armbrust, M., Kaftan, T., Meng, X., & Diğerleri**. (2015). Spark SQL: Relational Data Processing in Spark, *In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data,* (pp.1383-1394).

[29] **Spark MLLib Project**, Erişim 16 Ekim, 2015, http://spark.apache.org/mllib/

[30] **Das, T., Li, H., Shenker, S., Stoica, I., Zaharia, M**. (2012). Discretized streams: an efficient and fault-tolerant model for stream processing on large clusters. *In: Proceedings of the 4th USENIX conference on Hot Topics in Cloud Ccomputing.* (pp.10-10).

[31] **Real Time Analytics Definition**, Erişim 19 Ekin 2015, http://searchcrm.techtarget.com/definition/real-time-analytics

[32] **Challanges of Streaming Data**, Erişim 20 Ekim 2015, http://www.nesc.ac.uk/talks/1130/ToreRisch.pdf

[33] **Zaharia, M., Das, T., Wendell, P.** (2015). Diving into Spark Streaming's Execution Model, https://databricks.com/blog/2015/07/30/diving-into-spark-streamings-execution-model.html

[34] **Spark Streaming Programming Guide**, Erişim 25 Ekim 2015, http://spark.apache.org/docs/latest/streaming-programming-guide.html

[35] **Apache Storm,** Erişim 25 Ekim 2015, http://hortonworks.com/hadoop/storm/

[36] **Apache Storm Tutorial**, Erişim 26 Ekim 2015, https://storm.apache.org/documentation/Tutorial.html

[37] **Fayyad, U.M.** (1996). Data Mining and Knowledge Discovery: Making Sense Out of Data . *In: IEEE Intelligent Systems.* (pp.20-25).

[38] **Wilhelmiina, H.** (2006*). Descriptive and Predictive Modelling Techniques for Educational Technology.*

[39] **Farboudi, S.** (2009). *Tıp Bilişiminde İstatistiksel Veri Madenciliği*. (Yüksek Lisans Tezi). Fen Bil. Ens., Hacettepe Üniversitesi.

[40] **Wu, X., Kumar, V., Quinlan, J., Ghosh, J., Yang, Q., Motoda, H.** (2007). Top 10 Algorithms in data mining. *In: prooceding Knowledge Information System.* (pp.1-37)

[41] **Osowski, S., Siwekand, K., and Markiewicz, T**. (2004). MLP and SVM Networks. *In: Proceedings of the 6th Nordic Signal Processing Symposium*. (pp.37-40)

[42] **Osuna, E.E., Freund, R., Girosi, F.** (1997). Support Vector Machines: Training and Applications. (Teknik Rapor). Massachusetts Institute of Technology and Artificial Intelligence Laboratory, Massachusetts.

[43] **Vapnik, V.N.** (1995). The Nature of Statistical Learning Theory. (Kitap). Springer-Verlag, New York.

[44] **Cortes, C., Vapnik, V.** (1995). Support-Vector Network. *In: Machine Learning Journal Vol.20.* (pp.273-297)

[45] **Kavzaoğlu, T., Çölkesen, İ.** (2010). Destek Vektör Makineleri ile Uydu Görüntülerinin Sınıflandırılmasında Kernel Fonksiyonlarının Etkilerinin İncelenmesi. *Harita Dergisi, Sayı 144,*73-82.

[46] **Chapelle, O., Vapnik, V., Bousquet, O., Mukherjee, S. (2002).** Choosing Multiple Parameters for Support Vector Machines. *In: 2 Kluwer Academic Publishers.* (pp.131-159).

[47] **Kou-Ping, W., Sheng-De, W.** (2006). Choosing the kernel paramaters for support vector machines by the inter-cluster distance in future space *In: International Joint Conferance on Neural Networks*. (pp.1205-1211).

[48] **Shalizi, C.** Logistic Regression Ders Notu, Department of Statistics, Carnegie Mellon University.

[49] **Logistic Regression** V*ikipedia.* Erişim: 28 Kasım, 2015, https://en.wikipedia.org/wiki/Logistic_regression

[50] **HBase**, Erişim 2 Kasım 2015, https://hbase.apache.org/

[51] **MongoDB**, Erişim 2 Kasım 2015, https://www.mongodb.org/

[52] **Cassandra**, Erişim 2 Kasım 2015, http://cassandra.apache.org/

[53] **Stamper, J., Niculescu-Mizil, A., Ritter, S., Gordon, G.J., & Koedinger, K.R.** (2010). Algebra I 2008-2009. Challenge data set from KDD Cup 2010 Educational Data Mining Challenge. Find it at http://pslcdatashop.web.cmu.edu/KDDCup/downloads.jsp

[54] **Chia-Hua H., Hung-Yi L., Hsiang-Fu Y., … Shou-de L.** (2010). Feature Engineering and Classifier Ensemble for KDD Cup 2010. *In: JMLR: Workshop and Conference Proceedings* (pp.1-16).

[55] **Kerberos: The Network Authentication Protocol**, Erişim 12 Nisan 2016, http://web.mit.edu/kerberos/

[56] **Nedic, A.** (2002). Subgradient Methods for Convex Minimization. (Dokatora Tezi). Department of Electrical Engineering and Computer Science, Massachusetts of Technology.

[57] **SAS:** Erişim 12 Nisan 2016, http://www.sas.com/tr_tr/home.html

ÖZGEÇMİŞ

Ad-Soyad : Barış AKGÜN
Doğum Tarihi ve Yeri : 02.10.1989 / İSTANBUL
E-posta : baris.akgun1@gmail.com

ÖĞRENİM DURUMU:

- **Lisans** : 2011, Sakarya Üniversitesi, Mühendislik Fakültesi , Bilgisayar Mühendisliği
- **Lisans (Çift Anadal Programı)** : 2011, Sakarya Üniversitesi, Mühendislik Fakültesi, Endüstri Mühendisliği

TEZDEN TÜRETİLEN YAYINLAR, SUNUMLAR VE PATENTLER:

- **Akgün B., Öğüdücü Ş**. (2015). Streaming Linear Regression on Spark MLlib and MOA. *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, August 27-28, 2015 Paris, France.