T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BÜYÜK VERİDE MAKİNE ÖĞRENMESİ UYGULAMASI

MUSTAFA VAHİT KESKİN

YÜKSEK LİSANS TEZİ
UYGULAMALI İSTATİSTİK ANABİLİM DALI
İSTATİSTİK PROGRAMI

DANIŞMAN
YRD. DOÇ. DR. DOĞAN YILDIZ

İSTANBUL, 2018

T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BÜYÜK VERİDE MAKİNE ÖĞRENMESİ UYGULAMASI

Mustafa Vahit KESKİN tarafından hazırlanan tez çalışması 30.03.2018 tarihinde aşağıdaki jüri tarafından Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Uygulamalı İstatistik Anabilim Dalı'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Tez Danışmanı

Yrd. Doç. Dr. Doğan YILDIZ
Yıldız Teknik Üniversitesi

Jüri Üyeleri

Yrd. Doç. Dr. Doğan YILDIZ
Yıldız Teknik Üniversitesi _____

Prof. Dr. Ali Hakan BÜYÜKLÜ
Yıldız Teknik Üniversitesi _____

Doç. Dr. Esra AKDENİZ
Marmara Üniversitesi _____

ÖNSÖZ

Çağımızın petrolü olan veri, hızla artan boyutu ve yapısal olmayan formatlarda oluşu ile işlenmesi ve kendisinden anlamlı sonuçlar çıkarılmasını güçleştirmiştir. Ortaya çıkan bu güçlüklerin çözümü için geliştirilen yaklaşımların ele alındığı bu çalışmada büyük veri üzerinde veri manipülasyonu, büyük veri görselleştirmesi ve büyük veride makine öğrenmesi uygulamaları yapılmıştır.

Lisansüstü eğitimim boyunca maddi manevi tüm desteklerinden dolayı saygıdeğer hocam Yard. Doç. Dr. Doğan Yıldız'a, lisans eğitiminde bizleri veri bilimi dünyası ile tanıştıran değerli hocam Doç. Dr. Esra Akdeniz'e, tez süresince hiçbir desteği esirgemeyen sahibinden.com'un değerli yöneticilerinden Güntulu Peker Pamuk ve Emre Kesmen'e, tüm emeklerinden dolayı aileme ve hayatımın her aşamasında yanımda olan çok kıymetli eşim Betül Eren Keskin'e teşekkür ederim.

Mart, 2018

Mustafa Vahit KESKİN

İÇİNDEKİLER

Sayfa

SİMGE LİSTESİ vii
KISALTMA LİSTESİ viii
ŞEKİL LİSTESİ ix
ÇİZELGE LİSTESİ xi
ÖZET xii
ABSTRACT xiii
BÖLÜM 1
GİRİŞ 1
1.1 Literatür Özeti 1
1.2 Tezin Amacı 1
1.3 Hipotez 2
BÖLÜM 2
BÜYÜK VERİ 3
2.1 Büyük Veri Nedir? 3
2.1.1 Büyük Verinin Bileşenleri 4
2.1.2 Büyük Veri Analitiği Türleri 4
2.1.2.1 Betimleyici Analitik 5
2.1.2.2 Tanı Analitiği 5
2.1.2.3 Tahminsel Analitik 5
2.1.2.4 Reçeteli Analitik 5
2.2 Apache Hadoop 6
2.2.1 Apache Hadoop Temel Bileşenleri 6
2.2.1.1 Hadoop Ortak Gereksinimler (Hadoop Common) 7
2.2.1.2 Hadoop Dağıtık Dosya Sistemi (Hadoop Distributed File System) 7
2.2.1.3 Hadoop Kaynak Yönetimi (Yet Another Resource Negotiator) 7
2.2.1.4 Hadoop MapReduce 7
2.2.2 Hadoop Ekosistemi 9
2.2.3 Apache Hadoop'un Değerlendirilmesi 11
2.3 Apache Spark 11

2.3.1 Apache Spark Temel Bileşenleri ve RDD (Spark Core ve Resilient Distributed Datasets) 12
2.3.2 Spark SQL 13
2.3.3 Spark MLlib (Spark Makine Öğrenmesi Kütüphanesi) 14
2.3.4 Spark Akan Veri İşleme (Spark Streaming) 15
2.3.5 GraphX 15
BÖLÜM 3
MAKİNE ÖĞRENMESİ 16
3.1 Makine Öğrenmesi Nedir 16
3.2 Öğrenme Çeşitleri 16
3.2.1 Gözetimli Öğrenme 16
3.2.2 Yarı-Gözetimli Öğrenme 17
3.2.3 Gözetimsiz Öğrenme 17
3.2.4 Pekiştirmeli Öğrenme 17
3.3 Gözetimli Öğrenme Yöntemleri 17
3.3.1 Çoklu Doğrusal Regresyon Modeli 17
3.3.2 Lojistik Regresyon 19
3.3.3 Karar Ağaçları 20
3.3.4 Topluluk Öğrenme Yöntemleri (Ensemble Learning) 21
3.3.4.1 Torbalama Yöntemi (Bagging) 21
3.3.4.2 Boosting Yöntemi 22
3.3.4.3 Rastgele Ormanlar (Random Forests) 23
3.3.5 Naive Bayes 24
3.3.6 Yapay Sinir Ağları 25
3.3.6.1 Biyolojik Sinir Hücresi Yapısı 25
3.3.6.2 Yapay Sinir Hücresi Yapısı 26
3.3.6.3 Çok Katmanlı Yapay Sinir Ağları 27
3.4 Gözetimsiz Öğrenme Yöntemleri 28
3.4.1 Kümeleme Analizi 28
3.4.1.1 Hiyerarşik Olmayan Kümeleme Yöntemleri 29
3.4.1.2 Hiyerarşik Kümeleme Yöntemleri 29
3.4.1.3 Hiyerarşik ve Hiyerarşik Olmayan Yöntemler Arasındaki Farklar 29
3.4.2 Temel Bileşen Analizi 30
3.5 Model Doğrulama Yöntemleri 30
3.5.1 Sınama Seti Yaklaşımı (Holdout) 30
3.5.2 K-Katlı Çapraz Doğrulama 31
3.5.3 Leave-one-out 31
3.5.4 Yeniden Örnekleme Yöntemi (Bootstrap) 32
3.6 Model Başarı Değerlendirme Yöntemleri 32
3.6.1 Karışıklık Matrisi 32
3.6.2 Alıcı İşlem Karakteristiği Eğrisi 33
BÖLÜM 4

BÜYÜK VERİ UYGULAMALARI 35
4.1 Programlama Dilleri ve R Kütüphaneleri 35
4.1.1 SparkR 35
4.1.2 DistributedR 36
4.1.3 sparklyr 36
4.2 Çalışma Ortamının Hazırlanması 37
4.2.1 Amazon Web Servisi Kullanıcı Hesabı Oluşturma 38
4.2.2 Güvenlik Kimliği Ayarlamaları 38
4.2.2.1 Erişim Anahtarının Silinmesi 38
4.2.2.2 Bireysel Kullanıcı Oluşturma 39
4.2.2.3 EC2 Ayarları 40
4.2.3 EMR ile Hadoop Kümesinin Oluşturulması 40
4.2.5 R Studio Server Kurulumu ve Kullanıcı Oluşturma 44
4.2.6 Verinin HDFS Üzerine Dağıtılması 45
4.2.7 R Studio ve Spark Bağlantısı 46
4.3 Büyük Veride Keşifçi Veri Analizi Uygulamaları 48
4.4 Büyük Veri Görselleştirme 52
4.5 Büyük Veride Makine Öğrenmesi Uygulaması 54
4.5.1 Veri Düzenleme ve Değişken Dönüşümleri 55
4.5.2 Modellerin Eğitilmesi 56
4.5.3 Modellere Göre Değişken Önem Düzeylerinin İncelenmesi 57
4.5.4 Model Tahmin Başarılarının Karşılaştırılması 58
BÖLÜM 5
SONUÇ VE ÖNERİLER 61
KAYNAKLAR 62
EK - A
FLIGHTS VERİSİ İÇİN HIVE TABLOSU OLUŞTURMA KODU 67
EK - B
AIRLINES HIVE TABLOSU OLUŞTURMA KODU 68
EK - C
AIRPORTS HIVE TABLOSU OLUŞTURMA KODU 69
ÖZGEÇMİŞ 70

SİMGE LİSTESİ

$\pi(x)$	Lojistik regresyon fonksiyonu
P	Olasılık

KISALTMA LİSTESİ

AUC	Area Under Curve (ROC Eğrisinin Altında Kalan Alan)
AWS	Amazon Web Hizmetleri (Amazon Web Services)
EMR	Elastic Map Reduce (Elastik Map Reduce)
EC2	Amazon Elastic Compute Cloud (Amazon Elastik Bulut Hesaplama Platformu)
ROC	Receiver Operating Characteristic (Alıcı İşlem Karakteristiği)
FN	False Negative (Yanlış Negatif)
FP	False Pozitive (Yanlış Pozitif)
TN	True Negative (Doğru Negatif)
TP	True Pozitive (Doğru Pozitif)

ŞEKİL LİSTESİ

Sayfa

Şekil 2.1 Bir dakikada internette neler oluyor? 3
Şekil 2.2 Apache hadoop temel bileşenleri [17] 7
Şekil 2.3 MapReduce kelime sayma örnek uygulaması [18] 8
Şekil 2.4 Apache Hadoop ekosistemi [19] 9
Şekil 2.5 Apache Spark bileşenleri [33] 12
Şekil 2.6 Spark tembel (lazy) çalışma yapısı 13
Şekil 2.7 MapReduce'ün iteratif işlemlerde çalışma yapısı [35] 13
Şekil 2.8 RDD'nin iteratif çalışma yapısı [35] 13
Şekil 3.1 Örnek karar ağacı [47] 20
Şekil 3.2 Yanlış sınıflandırılan örneklerin ağırlıklandırılması ve iteratif ele alınması .. 23
Şekil 3.3 Biyolojik sinir hücresi 26
Şekil 3.4 Yapay sinir hücresi 27
Şekil 3.5 Çok katmanlı yapay sinir ağı [66] 28
Şekil 3.6 Holdout yöntemi [68] 30
Şekil 3.7 K-Katlı çapraz doğrulama yöntemi [68] 31
Şekil 3.8 ROC eğrisi [70] 34
Şekil 4.1 SparkR Mimarisi 36
Şekil 4.2 sparklyr mimarisi 37
Şekil 4.3 Security Status ekranı 38
Şekil 4.4 Erişim anahtarının silinmesi 39
Şekil 4.5 Kullanıcı oluşturma 39
Şekil 4.6 Grup oluşturma 39
Şekil 4.7 Inbound ayarları 40
Şekil 4.8 EMR servisinin başlatılması 41
Şekil 4.9 EMR servisi ile küme kurulumunun başlatılması 41
Şekil 4.10 Gerekli yazılımların seçilmesi 42
Şekil 4.11 Donanım ayarları 42
Şekil 4.12 Güvenlik Ayarları 43
Şekil 4.13 Küme genel görünüm 43
Şekil 4.14 Uzaktan erişim sonrası ekran görüntüsü 44
Şekil 4.15 Uçuş sayıları R kodu 48
Şekil 4.16 Uçuş sayıları R çıktısı 48
Şekil 4.17 Uçuş sayıları ve tam isimler R kodu 49
Şekil 4.18 Uçuş sayıları ve tam isimler R çıktısı 49
Şekil 4.19 Havaalanlarına göre uçuş sayıları R kodu 49
Şekil 4.20 Havaalanlarına göre uçuş sayıları R çıktısı 49
Şekil 4.21 Gecikme ortalamalarına göre sıralama R kodu 50
Şekil 4.22 Gecikme ortalamalarına göre sıralama R çıktısı 50

Şekil 4.23 Uçaklar ve gecikme süreleri R kodu 50
Şekil 4.24 Uçaklar ve gecikme süreleri R çıktısı 51
Şekil 4.25 İkili karşılaştırmalar R kodu 51
Şekil 4.26 İkili karşılaştırmalar R çıktısı 51
Şekil 4.27 Verinin oluşturulması verinin oluşturulması 52
Şekil 4.28 Verinin görselleştirilmesi 52
Şekil 4.29 Gecikmelerin barplot ile görselleştirilmesi 53
Şekil 4.30 Gerekli verinin oluşturulması 53
Şekil 4.31 Verinin görselleştirilmesi 53
Şekil 4.32 Geç kalma sürelerinn boxplot ile görselleştirilmesi 53
Şekil 4.33 Gecikme süreleri verisinin oluşturulması 54
Şekil 4.34 Gecikme süreleri verisinin görselleştirilmesi 54
Şekil 4.35 Gecikme süreleri histogramı 54
Şekil 4.36 Gerekli kütüphanelerin kurulması 55
Şekil 4.37 Veri setinin okutulması ve değişken isimlerinin düzenlenmesi 55
Şekil 4.38 Spark bağlantısının yapılması ve verinin geçici belleğe taşınması 55
Şekil 4.39 Spark SQL API'si ile değişken dönüşümü 55
Şekil 4.40 Spark MLLib API'si ile değişken dönüşümü 56
Şekil 4.41 Eğitim ve test veri setlerinin oluşturulması 56
Şekil 4.42 Tablo referanslarının oluşturulması 56
Şekil 4.43 Bağımlı ve bağımsız değişkenlerin tanımlanması 56
Şekil 4.44 Modellerin eğitilmesi 57
Şekil 4.45 Modellerin bir araya toplanması 57
Şekil 4.46 Modellere göre önem düzeylerinin hesaplanması 57
Şekil 4.47 Modellere göre önem düzeylerinin görselleştirilmesi 58
Şekil 4.48 Modellere göre değişkenlerin önem düzeyleri 58
Şekil 4.49 Test fonksiyonu 59
Şekil 4.50 Modellerin skorlanması 59
Şekil 4.51 Doğruluk hesaplaması için fonksiyon yazılması 59
Şekil 4.52 AUC ve Doğruluk değerlerinin hesaplanması 59
Şekil 4.53 Sonuçların görselleştirilmesi 60
Şekil 4.54 Model performans karşılaştırması 60

ÇİZELGE LİSTESİ

Sayfa

Çizelge 3.1 Karışıklık Matrisi.....32

ÖZET

BÜYÜK VERİDE MAKİNE ÖĞRENMESİ UYGULAMASI

Mustafa Vahit KESKİN

Uygulamalı İstatistik Anabilim Dalı

Yüksek Lisans Tezi

Tez Danışmanı: Yrd. Doç. Dr. Doğan YILDIZ

Çağımızın petrolü olan veri hızla artan boyutu ve çeşitliliği ile kendisinden anlamlı bilgi çıkarma çalışmalarını zorlaştırmıştır. Geleneksel yöntemler ile işlenemeyen boyut ve türdeki verileri işleyebilmek adına ortaya çıkan büyük veri yazılımları veri analitiği çalışmalarına yeni bakış açıları getirmiştir. Tez kapsamında bulut servisi üzerinde büyük veri araçları kullanılarak büyük veride keşifçi veri analizi, büyük veri görselleştirmesi ve büyük veride makine öğrenmesi uygulamaları gerçekleştirilmiştir. Uygulamada Amazon Web Servisi Elastic Map Reduce, Apache Hadoop, Apache Hive, Apache Spark ve R Studio kullanılacaktır. Makine öğrenmesi literatüründe yer alan ve sık kullanılan algoritmalar ele alınacak ve uygulama içinde performansları karşılaştırılacaktır.

Anahtar Kelimeler: Büyük veri, makine öğrenmesi, Apache Spark, sparklyr, Amazon Web Servisi

ABSTRACT

MACHINE LEARNING APPLICATION ON BIG DATA

Mustafa Vahit KESKİN

Department of Statistics

MSc. Thesis

Adviser: Assoc. Prof. Dr. Doğan YILDIZ

Data, which is the new oil of 21st century, has made it difficult to extract meaningful information from itself because of its increasing size and variety. Big data softwares emerged in order to process data that can not be processed by traditional methods have brought a new perspective on data analysis studies. Within the scope of the thesis, explorer data analysis, visualization, and machine learning applications have been performed in big data by using big data tools on cloud service. Amazon Web Services Elastic Mapreduce, Apache Hadoop, Apache Hive, Apache Spark and R Studio will be used in application. Frequently used algorithms in the machine learning literature will be discussed and their performance in the application will be compared.

Keywords: Big data, machine learning, Apache Spark, sparklyr, Amazon Web Services

BÖLÜM 1

GİRİŞ

1.1 Literatür Özeti

Yapılan Türkçe literatür taramasında genellikle büyük verinin yapısallığı, büyük veri araçlarının karşılaştırılmaları, bulut bilişim ile ilgli çalışmalara rastlanmıştır. Tez kapsamında ele alınan büyük veride makine öğrenmesi uygulamasına yakın olabilecek çalışamalarda gözlenmiştir.

Yengi yüksek lisans tez çalışmasında büyük veride duygu analizine dayalı öneri sistemi uygulaması gerçekleştirmiştir [1]. Salur Apache Hadoop kullanarak veri madenciliği uygulaması çalışmıştır [2]. Özdeş büyük veri araçları ile duygu analizi çalışmıştır [3]. Akgün Apache Spark ile akan veride sınıflandırma üzerine bir uygulama gerçekleştirmiştir [4]. Çetinkaya Apache Hadoop kullanarak hızlı tüketim sektörüne yönelik uygulamalar yapmıştır [5]. Hallaç büyük veri araçlarını kullanarak büyük veride makine öğrenmesi algoritmaların ile ilgili uygulamalar yapmıştır [6]. Bu çalışmalar incelendiğinde tezin farklılaştığı noktalar araç olarak bulut teknolojisinin ve R programlama dilinin kullanılmasıdır. Bununla birlikte uygulamada büyük veride keşifçi veri analizi, büyük veride veri görselleştirmesi ve büyük veride makne öğrenmesi uygulamaları ile de diğer çalışmalardan farklılaşmaktadır.

1.2 Tezin Amacı

Tezin amacı uçtan uca veriden anlamlı bilgi çıkarma çalışmalarını bulut bilişim, büyük veri, makine öğrenmesi ve R kullanarak sağlayabilmektir. Bu amaçla ilgili kavramların tanıtımı ve uygulamaları yapılmıştır.

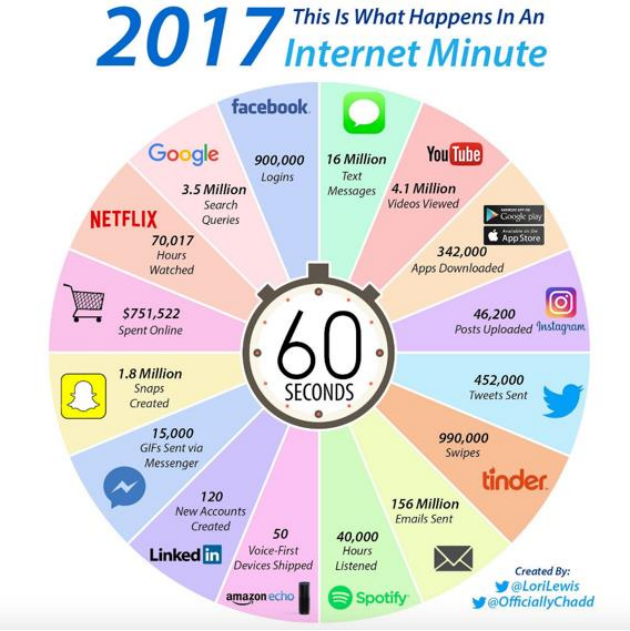
1.3 Hipotez

İstatistik dünyasının en sık kullanılan açık kaynak kodlu yazılımlardan birisi olan R ile büyük veri araçlarının birlikte kullanımları gözlenmek istenmiştir. Bu kapsamda büyük veri üzerinde keşifçi veri analizi, büyük veri üzerinde görselleştirme yöntemleri ve büyük veri üzerinde makine öğrenmesi uygulamalarının uçtan uca R ile gerçekleştirilmesine çalışılmıştır. Özellikle büyük veri yazılımları ile R programının birlikte çalışmasının ele alınması tezi benzer çalışmalardan öne çıkmaktadır.

BÖLÜM 2

BÜYÜK VERİ

İnsanoğlunun temas halinde olduğu hemen hemen her şey veri üretmeye başlamıştır. Hangi sıcaklıkta kahve içeceğimizi soran kahve makineleri, eve yaklaştığımızda kendisini açan kapılar, sosyal medya kanalları ile üretilen veriler, ülkelerin savunma sanayisinde kullandıkları insansız hava araçları, restoranlarda servis yapan robotlar ve hatta tarlalarımızda yetişen bitkiler [7]. Her yıl Cumulus Medya'dan Lor Lewis ve Chadd Callahan'ın yayınladığı "Bir dakikada internette neler oluyor?" raporu ve görseli (Şekil 2.1) bir dakika da üretilen verinin ile ilgili fikir verici olmaktadır [8].



Şekil 2.1 Bir dakikada internette neler oluyor?

2.1 Büyük Veri Nedir?

Meydana gelen ve hızla artan verinin büyüklüğü veriyi saklama ve işleme konusunda yeni problemleri beraberinde getirmiştir. Oluşan bu büyük verinin saklanması, işlenmesi

ve kendisinden anlamlı sonuçlar çıkarılması mevcut sistemler ile oldukça zorlaşmıştır. Olağanüstü hızla üretilen verinin saklanması, gerekli olması halinde başvurulması, içerisinden değerli bilgilerin çıkarılması gibi ihtiyaçları sık kullanılan araçlarca kabul edilebilir zamanlarda giderilememeye başlamıştır. Bilinen bu yöntem ve araçlarla kontrol edilemez ve işlenemez boyutta olan verilere büyük veri denir [9].

Büyük veri kendisini oluşturan bileşenlerce de tanımlanabilmektedir. Büyük veriyi oluşturan bileşenler bazı kaynaklarda 3V bazı kaynaklarda ise 5V olarak anılmaktadır [10].

2.1.1 Büyük Verinin Bileşenleri

Hacim: Verinin boyutunun geleneksel yaklaşımlar ile saklanamayacak/işlenemeyecek kadar büyük olmasıdır.

Hız: Büyük hacimlerde olan verinin hacminin hızlı bir şekilde artmaya devam etmesidir.

Çeşitlilik: Büyük hacimde olan ve büyük bir hızla artmaya devam eden verinin, farklı kaynaklardan, yapısal ya da yapısal olmayan veri türlerinden ve farklı tiplerdeki verilerden oluşmasıdır.

Burada önemli olan nokta büyük verinin büyük veri olarak adlandırılabilmesi için bu bileşenlere sahip olması gerekmediğidir. Örneğin verinin boyutunun çok büyük olması durumu gerçekleşiyorken, hızla artması ve çok çeşitli olması durumları gerçekleşmiyorsa bu veriye büyük veri değildir denilemez. Benzer şekilde veri setinin boyutunun çok büyük olmasıyla beraber farklı türden verileri de barındırdığında diğer bir ifade ile ilişkisel veri tabanlarında tutulamayan türden veriler olduğunda bu iki durum gerçekleşiyorken büyük hız şartı gerçekleşmeyebilir, bu durumda da veri büyük değildir denilemez.

2.1.2 Büyük Veri Analitiği Türleri

Genel anlamda iş dünyasına hitap eden büyük veri için çeşitli analitik yaklaşım türleri oluşmuştur bu yaklaşımlar betimleyici analitik, tanı analitiği, tahminsel analitik ve reçeteli analitiktir [11].

2.1.2.1 Betimleyici Analitik

İncelenen veri ya da problem üzerinde "Ne oldu?" sorusuna cevap aranan analitik yaklaşımdır. Veri içerisindeki mevcut durumun istatistiksel teknikler ile anlaşılmaya çalışılmasıdır.

2.1.2.2 Tanı Analitiği

Veri içerisindeki yapılarda ne olduğunun açıklanmasından farklı olarak veri içerisinde meydana gelen yapılar için "Neden oldu?" sorusunu araştıran analitik yaklaşımdır. Örneğin bir ülkede meydana gelen bir salgının hangi şehirde kaç kişide gözlemlendiği, ne kadar insanın tedaviye yanıt verdiği ve ilk olarak hangi şehirde gözlendiği gibi bilgilerin tespit edilmesi tanımlayıcı analitik kapsamına girerken, salgının ilk gözlendiği şehirde neden gözlendiği ya da şehirlerdeki hava kirliliği ile şehirdeki salgın sayısı arasında bir ilişki olup olmadığı gibi bilgilerin araştırılması teşhis analitiği kapsamına girer.

2.1.2.3 Tahminsel Analitik

Mevcut verilerin analiz edilmesi ile geleceğe yönelik çıkarımlarda bulunmak için kullanılan tahminsel analitik yaklaşımları "Ne olacak?" sorusuna cevap arar. Bir şirketin önümüzdeki ay ne kadar satış yapılacağının tahmin edilmesi, ülke çapındaki mağazaların hangi üründen ne kadar talepte bulunacağı, bir evin ya da aracın fiyatının ne kadar olabileceği, 10 yıl sonra insan nüfusunun ne olacağı, bir fabrikanın üretim bandından çıkan ürünlerin ne kadarının bozuk olabileceği gibi sorulara cevap aranır.

2.1.2.4 Reçeteli Analitik

Reçeteli analitik yaklaşımlar "Ne yapılmalı?" sorusuna cevap aramaktadır. Gerekli tahminsel modeller ile elde edilen sonuçların amaçlar doğrultusunda kullanılarak eylemlere dönüştürülmesidir. Tahminsel modeller ile elde edilen sonuçlara "bu sonuçlara göre hangi eylemler yapılmalı" sorusu yöneltilir. Örneğin terk edecek müşterinin tahmin edilmesi tahminsel modeller ile gerçekleştirildikten sonra müşterinin terk etmesini engellemek için hangi eylemlerin yapılması gerektiği sorusuna cevap aranır.

2.2 Apache Hadoop

Büyük veri araçları arasında ilk akla gelen ve bütün büyük veri dünyasının temelini oluşturan yazılım çerçevesi Hadoop'un çıkış noktası Google tarafından yayınlanan bazı makalelere dayanmaktadır. Big Table [12], Google File System [13] ve MapReduce [14] isimli bu makaleler, büyük boyuttaki; farklı türden verileri hataya karşı dayanaklı, ölçeklenebilir ve hızlı bir şekilde işleyebilme konuları etrafında şekillenmiştir. Büyük boyuttaki verilerin işlenmesi ile alakalı benzer problemler ile uğraşan Doug Cutting ve Mike Cafarella, Google tarafından yayınlanan bu makalelerden esinlenerek Hadoop'u icat etmişlerdir.

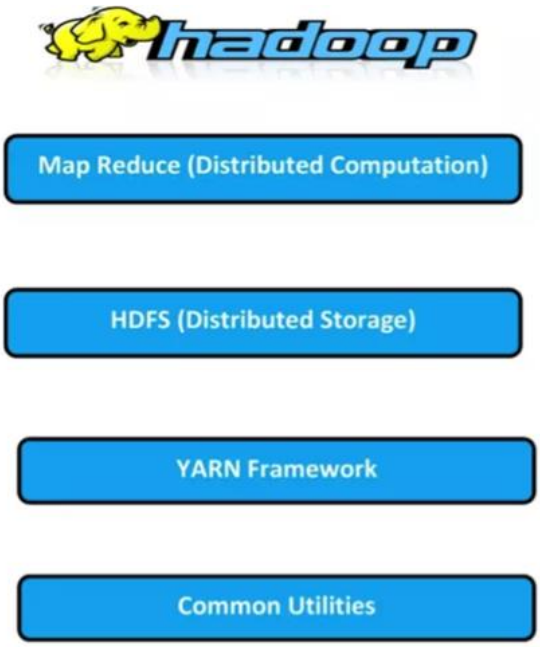
Hadoop'un doğmasının temel sebebi ilişkisel veri tabanlarının büyük hacim, farklı tür ve boyutu hızla artan verilerin saklanması ve işlenmesi ile alakalı ihtiyaçları karşılayamamasıdır [9].

Apache Hadoop bu ihtiyacı karşılamak için basit bir programlama modeli kullanarak bilgisayarlardan oluşan kümelerdeki büyük veri setlerini dağıtık olarak işlemeye olanak sağlayan bir kütüphanedir. Tek bir bilgisayardan oluşan sunucularda kullanılabileceği gibi, her birisi veri saklama ve hesaplama işlemleri yapabilen binlerce bilgisayardan oluşan sunucular içinde de kullanılabilecek şekilde tasarlanmıştır. Apache Hadoop küme içerisindeki bilgisayarlarda meydana gelebilecek hataları belirleyebildiği ve bu hatalarla başa çıkabildiği için yüksek kullanılabilirlik sağlamaktadır [15].

Apache Hadoop'un işlevselliği dağıtık hesaplama mimarisi sayesinde gerçekleşmektedir. Dağıtık hesaplama; problemlerin çözümü için, birden çok bilgisayarın tek bir bilgisayar gibi davranması yaklaşımına dayanmaktadır. Bir iş tüm bilgisayarlar arasında iş bölümü yapılarak ele alınır ve tek bir bilgisayar gibi davranılarak iş sonuçlandırılır [16].

2.2.1 Apache Hadoop Temel Bileşenleri

Büyük veri setlerini aynı ağda bulunan bilgisayarlar üzerinde dağıtık olarak işlemeye olanak sağlayan Apache Hadoop diğer bütün büyük veri teknolojilerine temel oluşturmaktadır. Büyük veri ile ilgili kullanılan tüm teknolojiler Apache Hadoop üzerindedir. Apache Hadoop temel olarak Hadoop Common, Hadoop Distributed File System(HDFS), Hadoop YARN ve Hadoop MapReduce bileşenlerinden oluşur [15].



Şekil 2.2 Apache hadoop temel bileşenleri [17]

2.2.1.1 Hadoop Ortak Gereksinimler (Hadoop Common)

Büyük veri teknolojileri, başka bir ifade ile diğer tüm Hadoop modüllerini destekleyen ortak gereksinimlerdir.

2.2.1.2 Hadoop Dağıtık Dosya Sistemi (Hadoop Distributed File System)

Hadoop Dağıtık Dosya Sistemi, yerel dosya sistemleri olan FAT32 ve NTFS dosya sistemleri gibi bir dosya sistemidir. Bu sistemlerden farkı büyük boyutlardaki veriyi dağıtık şekilde depolamaya ve kontrol etmeye olanak sağlamasıdır.

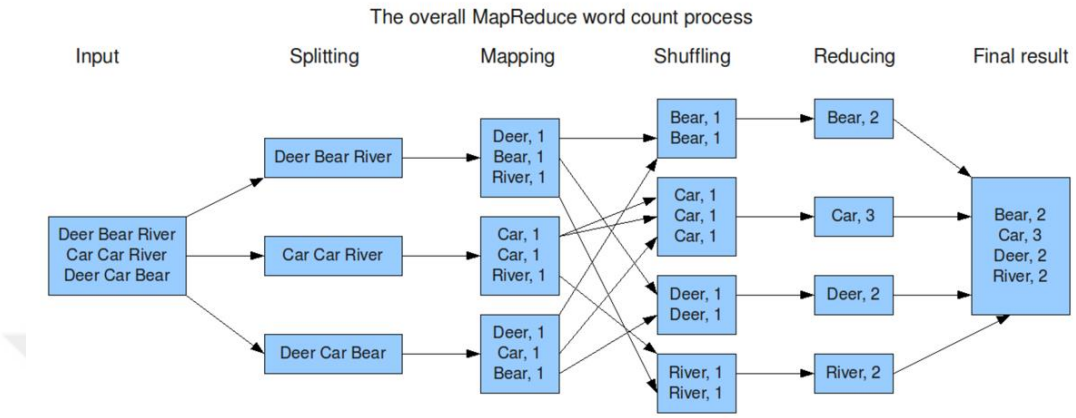
2.2.1.3 Hadoop Kaynak Yönetimi (Yet Another Resource Negotiator)

Kaynak yönetimi ve iş planlaması için kullanılan YARN, Apache Hadoop'un daha etkili kullanılmasına olanak sağlamaktadır.

2.2.1.4 Hadoop MapReduce

Aynı ağdaki dağıtık bilgisayar kümeleri üzerinde büyük veri analizi yapılabilmesi için geliştirilmiş bir programlama modelidir. Dağıtık dosya sisteminde tutulan veriler map, shuffle ve reduce ismi verilen üç aşamadan geçer. HDFS üzerinde dağıtık olarak tutulan veriler map aşamasında paralel olarak işlenerek veri işleme yükü dengeli olarak aynı ağdaki tüm fiziksel makinalara dağıtılır. Shuffle ve Reduce aşamasında map aşamasında

dağıtık olarak yapılan tüm işlemler ağ üzerinden toplanarak son çıktı haline getirilir. Son kullanıcı tek bir bilgisayar üzerinde işlemler yaparken arka planda bir bilgisayar kümesi çalışır. Bu işlemin yazılım anlamında çözümü MapReduce [14] sayesinde gerçekleşir.



Şekil 2.3 MapReduce kelime sayma örnek uygulaması [18]

Map Aşaması:

Map aşamasının görevi HDFS üzerindeki girdi verilerini işlemektir. Girdi verisi satır satır map fonksiyonundan geçirilir. Map fonksiyonu veriyi işler ve yeni veri parçacıkları oluşturur.

Reduce Aşaması:

Bu aşama shuffle ve reduce aşamalarının birleşiminden oluşur. Reduce aşamasının görevi map aşamasından gelen veriyi işlemektir. Map aşamasında tüm makinalardan gelen veri işlenip indirgenerek istenilen görev sonuçlandırılmış ve çıktılar elde edilmiş olur.

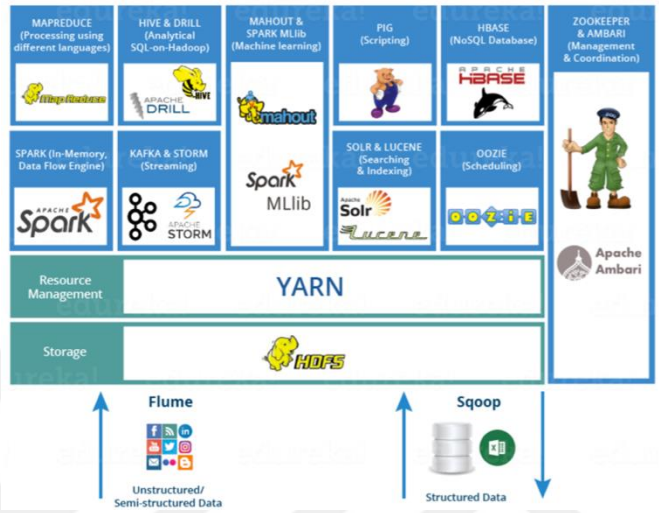
MapReduce görevleri boyunca Hadoop Map ve Reduce görevlerini küme üzerindeki tüm makinalara gönderir. Tüm bu görevlerin başarılı olup olmadığı, verilerin taşınıp taşınmadığı gibi işler kontrol edilir.

Input ile ifade edilen HDFS üzerinde bulunan veridir. Bu veri Splitting olarak ifade edilen birbirinden farklı fiziksel makinalara dağıtılmıştır. Her bir makinada bu verinin hangi parçasının olduğu bilgisi bilinmektedir. Veri üzerinde yapılması istenilen görev mapping aşaması ile her bir makina üzerinde ayrı ayrı uygulanır. Örnekte amaç kelime saymaktır. Bu sebeple her bir makina kendisinde kayıtlı olan kelimeleri sayar. Shuffling

ve Reducing aşamasında her bir makinadan gelen sonuçlar birleştirilerek tüm küme üzerinde hangi kelimeden kaçar adet olduğu belirlenmiş olur.

2.2.2 Hadoop Ekosistemi

Apache Hadoop üzerinde çeşitli görevler için kullanılan birçok farklı yazılım mevcuttur.



Şekil 2.4 Apache Hadoop ekosistemi [19]

- *Apache Ambari*

Apache Hadoop kümelerinin kurulumu, yönetimi ve gözetimi için geliştirilmiş web tabanlı bir araçtır [20]. Kümelerin durumunu ve performanslarını takip edebilecek bir gösterge paneli sağlar. Panel sayesinde küme genelindeki Hadoop servislerini kurmak, yönetmek ve izlemek oldukça kolaydır.

- *Apache Cassandra*

Apache Cassandra açık kaynaklı bir NoSQL veri tabanıdır. Performanstan ödün vermeden ölçeklenebilirlik ve yüksek erişilebilirlik hedeflendiğinde kullanılabilecek bir veritabanıdır [21].

- *Apache Hive*

Apache Hive'da arka planda SQL komutlarını MapReduce'e çevirerek büyük veri üzerinde SQL sorguları atılabilmesini sağlar [22].

- *Apache HBase*

HBase büyük boyutlu verilere gerçek zamanlı okuma-yazma erişimi yapmak gerektiği zamanlarda kullanılan açık kaynak kodlu sütun bazlı bir veritabanıdır [23].

- *Apache Sqoop*

Klasik veritabanları ile HDFS arasında veri transferleri yapılmasını sağlayan açık kaynak kodlu bir araçtır. İsmini Sql-to-Hadoop ifadesinden almıştır [24].

- *Apache Oozie*

Hadoop ekosistemi üzerindeki çeşitli işlerin iş akışını planlamak ve işleri zamanlamak için kullanılan açık kaynak kodlu bir araçtır [25].

- *Apache Impala*

Cloudera tarafından geliştirilen açık kaynak kodlu impala SQL arayüzü ile gerçek zamanlı sorgular yapılabilmesine olanak sağlamasıyla hızlı analizler ve keşifçi çalışmalar için uygundur [26].

- *Apache Mahout*

Apache Mahout makine öğrenmesi algoritmalarını barındıran bir açık kaynak kodlu kütüphanedir. Hadoop üzerinde ölçeklenebilir makine öğrenmesi projeleri için kullanılır. Fakat Apache Spark sonrasında Apache Mahout projesi kendisini Spark odaklı olarak geliştirmeye devam etmektedir [27].

- *Azkaban*

Azkaban açık kaynak kodlu görev planlayıcısıdır [28].

- *HUE*

HUE web tabanlı bir sorgu editörüdür. Sağladığı arayüz ile interaktif sorgular, görselleştirme ve keşifçi veri incelemeleri yapılabilir [29].

- *Apache Kafka*

Büyük veri akışını düşük bir gecikme ile gerçek zamanlı sağlayabilen Apache Kafka Linkedin bünyesinde geliştirilmiş açık kaynak kodlu bir projedir [30].

2.2.3 Apache Hadoop'un Değerlendirilmesi

Veri Saklama ve İşleme Gücü: İlişkisel veri tabanlarında meydana gelen zorlukları ortadan kaldırıp daha büyük boyuttaki ve çeşitteki verileri saklama ve işleme imkanı sunması.

Açık Kaynak: Apache Hadoop'un açık kaynak olması ve geleneksel yöntemlere göre işletme maliyetlerinin çok çok daha düşük olması.

Hız: Hızlı olması.

Esneklik: Farklı tür ve farklı kaynaklardan gelen her tipteki veriyi bir platformda birleştirebilmesi ve gerektiğinde bu verileri işleyerek anlamlı hale getirebilmeye olanak sağlaması.

Ölçeklenebilirlik: Mimari yapının, ihtiyaç olması halinde çalışan sistemler etkinlenmeden genişletilebilmesi ya da düzenlenebilmesi.

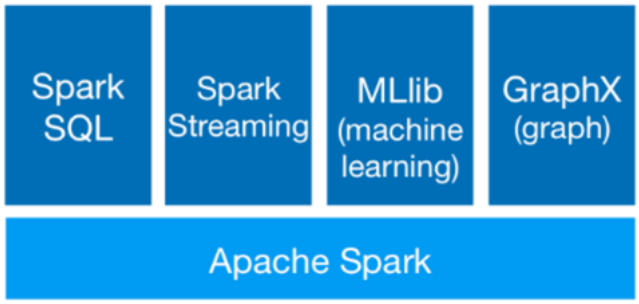
Hata Toleransı: Kümeyi oluşturan düğümlerin herhangi birisinde meydana gelebilecek hatalarla sistemin genel çalışmasını etkilemeyecek şekilde mücadele edilebilmesi.

Hadoop'un sağladığı bunca faydanın [31] yanında MapReduce programlama modelinin eksik kaldığı önemli bir yön vardır. MapReduce disk tabanlı çalışan bir model olduğu için her MapReduce görevinde diskten okuma ve diske yazma işlemi yapılır. Bu sebeple iterasyon barındıran işlemler zaman almakta ve kaynakları meşgul etmektedir [32]. MapReduce'un eksik kaldığı bu yön Apache Spark'ın bellek içi çalışma prensibiyle giderilmiştir.

2.3 Apache Spark

Apache Spark küme üzerinde hızlı ve genel amaçlı bilgi işleme sistemidir [33]. 2009 yılında Berkeley Üniversitesi AMPLab'da bir proje olarak geliştirilmeye başlanan Apache Spark, Hadoop'un MapReduce yapısına bir alternatif olarak geliştirilmiştir. MapReduce'un disk bazlı modelinden farklı olarak dağıtık bellek içi veri işleyebilme yeteneğine sahiptir. Apache Spark bu özelliği ile Apache Hadoop'un değil MapReduce programlama modelinin bir alternatifidir [34]. Bellek içi veri işleyebilme özelliğinin disk tabanlı veri işleme modeli ile kıyaslandığı çalışmalarda Apache Spark'ın Apache Hadoop'a göre 100 kat daha hızlı olduğu gözlenmektedir. R, Python, Scala ve Java

dillerine verdiği destek yazılım geliştiriciler, veri analistleri, veri bilimciler ve veri mühendisleri için Apache Spark'ın kullanımını kolaylaştırmıştır.

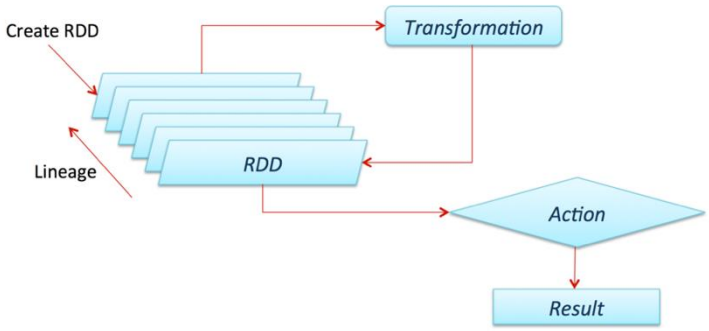


Şekil 2.5 Apache Spark bileşenleri [33]

2.3.1 Apache Spark Temel Bileşenleri ve RDD (Spark Core ve Resilient Distributed Datasets)

Hadoop Common'a benzer olarak Spark Core Apache Spark'ın temel yapısını oluşuran bir kütüphanedir. Hafıza yönetimi, görevlerin dağıtılması, hata kurtarma, saklama ve dosya sistemlerine erişim gibi temel bileşenler Spark Core içerisindedir. Apache Spark'ın tüm bileşenleri Spark Core baz alınarak geliştirilmiştir.

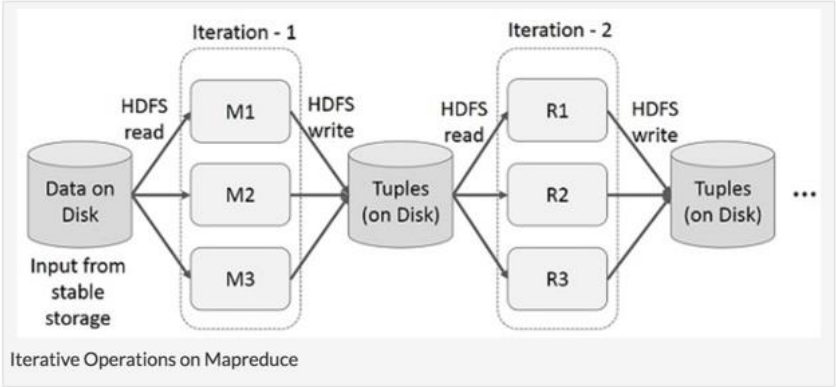
Apache Spark'ın programlama modeli, verinin bellek içi tutularak paralel işlenmesini ifade eden RDDs'dir. Apache Spark'ın çalışması 3 aşamadan oluşur: RDDs, transformasyon ve aksiyon. RDDs üretilen bir işlemi askıda bekletir, yapılmak istenen işlemi anlar ve kendisinde bekletir, transformasyon tutulan RDDs'ler üzerinden yeni RDDs'ler üretir, aksiyon işleminde ise tranformasyon işlemi sonrasında üretilen RDDs'ler işlenir ve toplanarak sonuçlar elde edilir. Aksiyon basamağı tetiklenmeden uygulama (execution) işlemi gerçekleşmemektedir. Bu yapısıyla tembel çalışma (Lazy Evaluation) olarak adlandırılan Spark'ın çalışma mantığı (Şekil 2.6) gereksiz hesaplamalardan kaçınarak hesaplama gücü ve hız sağlamaktadır [34].



Şekil 2.6 Spark tembel (lazy) çalışma yapısı

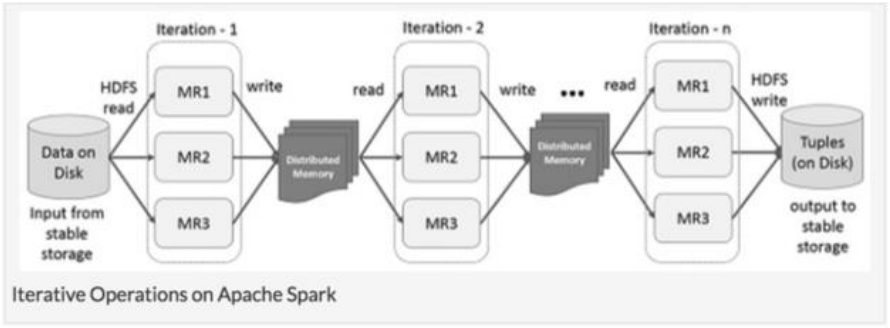
Apache Spark'ın transformasyon işlemi Apache Hadoop üzerindeki Map işlemi, actions işlemi de Reduce işlemi olarak ifade edilebilir.

MapReduce disk tabanlı çalışan bir model olduğu için her MapReduce görevinde diskten okuma ve diske yazma işlemi yapılır (Şekil 2.7). Bu sebeple iterasyon barındıran işlemler zaman almakta ve kaynakları meşgul etmektedir [32].



Şekil 2.7 MapReduce'ün iteratif işlemlerde çalışma yapısı [35]

MapReduce programlama modelinde her iterasyonda diske yazılıp diskten okunma işlemleri yer alırken Apache Spark programlama modelinde (RDD) (Şekil 2.8) diske yazım yapılmamakta bunun yerine verinin bellek içi tutulması sağlanmaktadır. Bu sayede Apache Spark MapReduce'e göre 100 kata kadar daha hızlı çalışmaktadır [34].



Şekil 2.8 RDD'nin iteratif çalışma yapısı [35]

2.3.2 Spark SQL

Spark SQL, SQL veya bir DataFrame API'si kullanarak Spark programlarındaki yapılandırılmış veriyi sorgulama imkanı sağlar [36].

2.3.3 Spark MLlib (Spark Makine Öğrenmesi Kütüphanesi)

Spark MLlib Apache Spark'ın ölçeklenebilir makine öğrenmesi kütüphanesidir. Apache Spark bellek içi çalışma mantığından dolayı iteratif işlemler barındıran veri madenciliği ve makine öğrenmesi uygulamaları için uygun bir çözüm sağlamaktadır [37].

R, Python, Scala ve Java ile kolayca kullanılabilen Spark MLlib, makine öğrenmesi algoritmalarının dağıtık şekilde uygulanabilmesi için optimize edilmiştir. Spark MLlib içerisinde kullanılabilecek testler ve algoritmalar aşağıda listelenmiştir.

Temel İstatistik:

- Korelasyon
- Hipotez Testleri

Sınıflandırma:

- Binomial Lojistik Regresyon
- Multinominal Lojistik Regresyon
- Karara Ağaçları
- Rastgele Ormanlar
- Gradient-Boosted Ağaç Sınıflandırıcı
- Çok Katmanlı Algılayıcı
- Doğrusal Destek Vektör Makineleri
- One-vs-All Sınıflandırıcı
- Naive Bayes Sınıflandırıcı

Regresyon:

- Doğrusal Regresyon
- Genelleştirilmiş Doğrusal Regresyon
- Regresyon Ağaçları
- Rastgele Orman Regresyonu
- Gradient-Boosted Ağaç Regresyonu
- Sağkalım Regresyon

- İzotonik Regresyon

Kümeleme:

- K-Means

- Latent Dirichlet Allocation

- Bisecting K-Means

- Gaussian Mixture Model

Yukarıda sıralanan genel başlıklar dışında, değişken çıkarımı-dönüşümü-seçimi, tavsiye sistemleri, model seçimi ve parametre optimizasyonu gibi işlemler içinde gerekli fonksiyonlar bulunmaktadır.

2.3.4 Spark Akan Veri İşleme (Spark Streaming)

Spark Streaming akan verinin ölçeklenebilir, yüksek hacimli ve hata toleranslı olarak işlenmesine imkan sağlayan temel Spark API'sinin bir uzantısıdır. Birçok farklı kaynaktan gelen veri Spark Streaming'e entegre edilebilir ve map, reduce, join, windows gibi yüksek seviye fonksiyonlarla işlenebilir. İşlenen veri dosya sistemleri, veri tabanları ve canlı gösterge ekranlarına gönderilebilir. Hatta Spark MLlib içerisindeki makine öğrenmesi algoritmaları ve graf işlemleri akan veri üzerinde uygulanabilir [38].

2.3.5 GraphX

GraphX grafikler ve paralel-grafik tabanlı hesaplamalar için kullanılan bir kütüphanedir [39].

BÖLÜM 3

MAKİNE ÖĞRENMESİ

3.1 Makine Öğrenmesi Nedir

Yapay zekanın bir alt dalı olan makine öğrenmesi, öğrenme metodları ve bu metotların performansı ile ilgilenir. Bilgisayarların insanlara benzer şekilde öğrenmesini sağlamak için çeşitli algoritma ve tekniklerin geliştirilesi için çalışılan bilimsel bir çalışma alanıdır [40].

Makine öğrenmesi, büyük veri yığınları içerisinde fark edilmesi zor olan yapıları bulmak ve ortaya çıkarmak ile ilgilenir [40].

Makine öğrenmesi algoritmaları kendilerine verilen örnek olayları inceleyerek bu olayların meydana geliş biçimlerini öğrenir ve örnek olaylar üzerinden genelleme yapma yeteneği kazanır [41].

Gerçekleşen olayların meydana gelmesini sağlayan etkenleri ve bu etkenlerin etki şekillerini öğrenen algoritmalar herhangi bir yeni olay meydana gelmeden önce bu olayı tahmin edebilirler. Bir uçağın ne kadar gecikeceği, gün içinde yağmur yağışı olup olmayacağı, şirketin ay sonu gelirlerinin ne kadar olacağı, bir müşterinin markayı terk edip etmeyeceği, insansız hava aracının tespit ettiği nesnenin ne olduğu gibi durumlar ifade edilen olaylara örnek olarak verilebilir.

3.2 Öğrenme Çeşitleri

3.2.1 Gözetimli Öğrenme

Gözetimli öğrenme girdi ve çıktı değerlerinin birlikte yer aldığı veri setleri için gerçekleştirilebilen öğrenme biçimidir. Hangi girdi değerlerinin hangi çıktı değerine karşılık geldiğinin bilinmesi veri setinin etiketli olduğu anlamına gelir. Bu öğrenme

tekniğinde algoritmalar girdi ve çıktı değerleri arasındaki ilişkiyi ifade etmesi beklenen fonksiyonlar üretmeye çalışır. Bu fonksiyon yardımıyla yeni bir örnek ve örneği oluşturan girdiler verildiğinde hangi çıktı değerinin oluşacağının tahmini yapılır. Üretilen fonksiyonun çıktıları eğri uydurmak için ya da girdilerin sınıflarının tahmin edilmesi için kullanılabilir [42]. Doğrusal regresyon modeli, lojistik regresyon, destek vektör makinaları, karar ağaçları, topluluk öğrenme yöntemleri ve yapay sinir ağları gözetimli öğrenme algoritmalarına örnek olarak verilebilir [43].

3.2.2 Yarı-Gözetimli Öğrenme

Yarı-gözetimli öğrenmede yeni örneklerin etiketlerinin tahmin edilmesi için hem etiketlenmiş hem de bir miktarda etiketlenmemiş veri kullanılır [40].

3.2.3 Gözetimsiz Öğrenme

Sadece girdi değerlerinden oluşan veri setleri için söz konusu olan öğrenme biçimidir. Etiketlenmiş bir veri olmadığı için gözetimsiz öğrenmede değerlendirme yapmak zordur. Temel bileşen analizi, çok boyutlu ölçekleme ve kümeleme analizi gözetimsiz öğrenme metodlarına örnek olarak verilebilir [40].

3.2.4 Pekiştirmeli Öğrenme

Pekiştirmeli öğrenmede öğrenme sistemine yerine getirmesi için bir hedef verilmekte ve bu hedefi deneme-yanılma yoluyla gerçekleştirmesi beklenmektedir. Sistem deneme-yanılma işlemlerini en çok ödül sağlayan eylemlerine göre düzenlemekte ve verilen hedefe nasıl ulaşacağını kendisi öğrenmektedir [42].

3.3 Gözetimli Öğrenme Yöntemleri

3.3.1 Çoklu Doğrusal Regresyon Modeli

Veri bilimi ve makine öğrenmesi alanlarında en sık kullanılan algoritmalardan birisi olan doğrusal regresyon modelleri, bağımlı ve bağımsız degişkenler arasındaki ilişkiyi en iyi ifade eden denklemi oluşturmayı amaçlar. Çoklu doğrusal regresyonda araştırmacının iki genel amacı vardır. Bunlardan birisi bağımlı değişkeni etkilediği belirlenen değişkenler vasıtasıyla bağımlı değişkenin değerini tahmin etmek, bir diğeri; bağımlı değişkeni etkilediği düşünülen bağımsız değişkenlerden hangisinin veya

hangilerinin bağımlı değişkeni daha çok etkilediğini tespit etmek ve aralarındaki ilişkiyi tanımlamaktır [44].

Çoklu doğrusal regresyon modelinde Y bağımlı değişkeni ile $X_1, X_2,\ldots, X_p$ bağımsız değişkenleri arasındaki ilişki aşağıdaki şekilde ifade edilir:

$$Y_i = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \cdots + \beta_j X_{ij} + \cdots + \beta_p X_{ip} + \varepsilon_i \tag{3.1}$$

Y_i bağımlı değişkenin değerlerini, $X_1, X_2,\ldots, X_p$ bağımsız değişkenlerin değerlerini ifade eder. $\beta_0, \beta_1, \beta_2, \cdots, \beta_p$ bilinmeyenlerine regresyon katsayıları denir. β_0 sabit terimdir. Tüm diğer bağımsız değişkenlerin değerleri sıfır olduğunda bağımlı değişkenin değerini gösterir. ε hata terimidir. Amaç gerçek değerler ile tahmin edilen değerler arasındaki farkların kareleri toplamını minimum yapmaya çalışmaktır. Bu durumu ifade eden aşağıdaki eşitlik açılıp tahmin edilmek istenen parametrelere göre kısmi türevler alındığında erişilecek olan normal denklemlerin ortak çözümü ile parametreler elde edilebilecektir [44].

$$\sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} \left(y_i - \hat{y}_i\right)^2 \tag{3.2}$$

Elde edilen normal denklemler aşağıdaki matris formunda ifade edilerek problem (3.3) denklemini çözmeye indirgenir, çözüm sonrasında elde edilen vektör (b), tahmin edilmek istenen parametrelerden oluşur [44].

$$b = [X'X]^{-1}X'Y \tag{3.3}$$

Doğrusal Regresyon Modellerinde çalışmaların güvenilirliği için bazı varsayımların yerine gelmesi gerekmektedir [45]. Bu varsayımlar:

- Hatalar normal dağılır, birbirinden bağımsızdır ve aralarında otokorelasyon yoktur.

- Her bir gözlem için hata terimleri varyansları sabittir.

- Değişkenler ile hata terimi arasında ilişki yoktur.

- Bağımsız değişkenler arasında çoklu doğrusal ilişki problemi yoktur.

Çoklu doğrusal regresyon analizinde modelin tahmin başarısı ve bağımsız değişkenlerin bağımlı değişkendeki varyasyonu açıklama konusundaki başarısı ile ilgili çıktılar elde edilebilir [44].

3.3.2 Lojistik Regresyon

Lojistik regresyon modeli bağımlı değişkeninin kategorik olması ve sınıflandırma işlemleri için kullanılması ile doğrusal regresyon modellerinden ayrılır. Lojistik regresyon analizi adını, bağımlı değişkene uygulanan logit dönüştürmeden (logit transformation) almaktadır. Doğrusal regresyon modellerinde olduğu gibi amaç bağımlı ve bağımsız değişkenler arasındaki ilişkiyi tanımlayabilecek bir model kurmaktır. Lojistik regresyonda doğrusal regresyonda aranan varsayımların hiç birisi aranmadığı için daha esnek kullanılabilirliği vardır [46].

Lojistik regresyonda bağımlı değişkenin 1 olarak tanımlanan değerinin gerçekleşme olasılığı hesaplanır. Dolayısıyla bağımlı değişkenin alacağı değer ile ilgilenilmez. Bağımsız değişkenler ile π olasılık değeri arasındaki ilişki lojistik fonksiyonu ile gösterilir. Böylece üretilen değerlerin 0 ile 1 arasında olması sağlanır.

Lojistik dağılım fonksiyonu:

$$\pi(x) = P(Y = 1 \mid X = x) = \frac{e^{\beta_0+\beta_1 x_1+\beta_2 x_2+\cdots+\beta_p x_p}}{1 + e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2+\cdots+\beta_p x_p)}} \tag{3.4}$$

π(x), X=x olduğunda Y'nin 1 olma olasılığıdır. Doğrusal olmayan π(x) lojistik regresyon fonksiyonu logit dönüşüm uygulanarak doğrusallaştırılır. Doğrusallaştırma sayesinde doğrusal regresyon modeline benzer yorumlamalar yapma olanağı sağlanır. Logit dönüşüm uygulayabilmek için odds oranına ihtiyaç duyulmaktadır. Bir olayın gerçekleşmesi olasılığının gerçekleşmemesi olasılığına olan oranı odds oranı olarak adlandırılır.

Lojistik regresyon fonksiyonunun Odds'u:

$$\frac{\pi(x)}{1-\pi(x)} \tag{3.5}$$

Odds oranının doğal logaritmasının alınmasıyla lojistik regresyon modeli elde edilir (logit model):

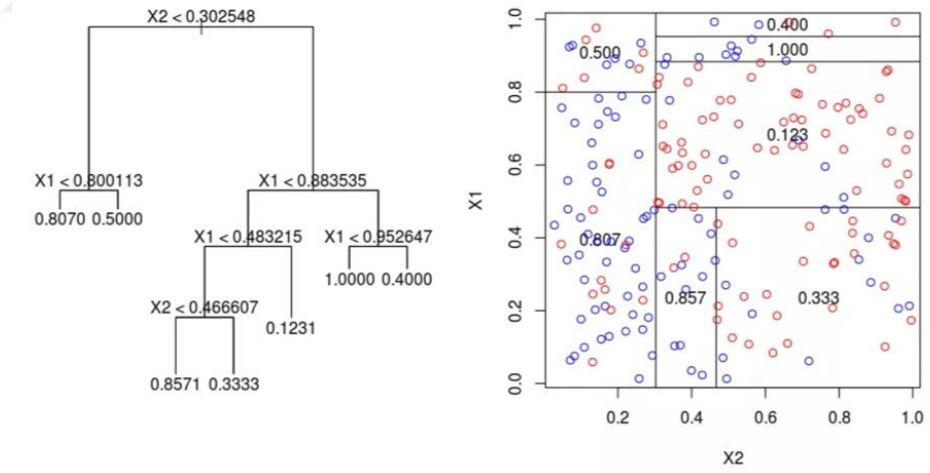
$$g(x) = ln\ \frac{\pi(x)}{1-\pi(x)} = \ln e^{(\beta_0+\beta_1 x_1+\beta_2 x_2+\cdots+\ \beta_p x_p)} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p \quad (3.6)$$

Yukarıda denklemi verilen lojistik regresyon modelinde model katsayılarının tahminlenebilmesi analitik olarak gerçekleştirilemediği için iteratif parametre tahmin yöntemleri kullanılır. Maksimum olabilirlik tekniği bu yöntemlerden birisidir.

Parametreler için belirlenen başlangıç değerleri ile örnek veri setinin bu parametreler ile oluşma olasılıkları hesaplanır. İteratif olarak maksimum olasılık değerleri elde edilinceye kadar parametrelere ait tahmini değerler hesaplanır. Böylece gözlenme olasılığı en yüksek olan parametreler elde edilmiş olur [44].

3.3.3 Karar Ağaçları

Karar ağaçları sınıflandırma ya da regresyon problemlerinde kullanılabilen gözetimli öğrenme yöntemlerindendir. Karar ağaçları veri içerisindeki karmaşık yapıları basit karar yapılarına dönüştürür. Diğer yöntemlerle kıyaslandığında karar ağaçlarının yapılandırılması, anlaşılması ve yorumlanması daha kolaydır. Bunların yanında karar ağaçlarının diğer bir avantajı ise başarılı modeller üretmesidir.



Şekil 3.1 Örnek karar ağacı [47]

Karar ağaçları heterojen veri setlerini belirlenmiş bir hedef değişkene göre homojen alt gruplara ayırır. Alt gruplara ayırma işlemi bir takım karar verme kuralları ile gerçekleştirilir [48]. Karar ağacı süreçlerinde algoritmaların temel adımları hemen hemen birbirleriyle aynıdır. Öncelikle bir kök düğüm oluşturulur. Oluşan bu düğüm içerisindeki örnekler homojen ise kök yaprağa dönüşür değil ise karar kuralları uygulanarak dallanma başlar. Dallanma(bölme) işlemine bir düğüme düşen örneklerin saflığına göre karar verilir. Saflığı ölçmek için Kategorik Ki-Kare, Gini, Entropi gibi kriterler kullanılır. Bu şekilde kökten yapraklara kadar ağaç oluşturma işlemi gerçekleştirilir ve test işlemi yapraklarda biter [49].

3.3.4 Topluluk Öğrenme Yöntemleri (Ensemble Learning)

Ağaca dayalı yöntemlerden olan topluluk öğrenme yöntemlerinin temeli birden çok karar ağacının ürettiği tahminlerin biraraya getirilerek değerlendirilmesine dayanır. Topluluk içindeki ağaçların her birisinin yaptığı sınıf tahminleri oylamaya tabi tutulur. En çok oyu alan sınıf diğer bir ifade ile ağaçların üzerinde daha fazla mutabık kaldığı sınıf topluluğun sınıf tahmini olarak açıklanır. Tahminler çok sayıda sınıflandırıcının oylamasından gelen sonuçlardan oluştuğu için topluluk öğrenme yöntemleri daha güvenilirdir. Bu durum tekil öğrenme algoritmalarına göre daha başarılı olmalarının temel sebebidir. Bagging, Boosting ve Random Forrests en çok üzerinde çalışılan ve bilinen topluluk öğrenme algoritmalarıdır [50].

3.3.4.1 Torbalama Yöntemi (Bagging)

Bagging (Bootstrap Aggregating) bir topluluk öğrenme yöntemi olup sınıflama ve regresyon modelleri için uygulanabilmektedir. Bagging yönteminde veri seti içerisinden çok sayıda karar ağaçı oluşturulur. Oluşturulan her bir ağaç eğitilir ve her bir ağaç bir tahmin sonucu üretir. Çözülmek istenen probleme göre her bir ağacın ürettiği tahmin sonuçlarının ortalaması alınarak ya da sınıf oylaması yapılarak nihai sonuç topluluğun sonucu olan ilan edilir [50].

Bagging yönteminde ağaçlar veri setindeki sınıf yapısı korunacak şekilde rastgele örneklem yöntemi bootstrap ile oluşturulur. Oluşturulan ağaçların daha önce oluşturulan ağaçlar ile bağımlılıkları yoktur. Oluşturulan her bir ağaç veri setinin ilk halinden oluşturulur. Bu yaklaşımı sağlayan bootstrap rastgele örnekleme yöntemidir. Bir ağaç

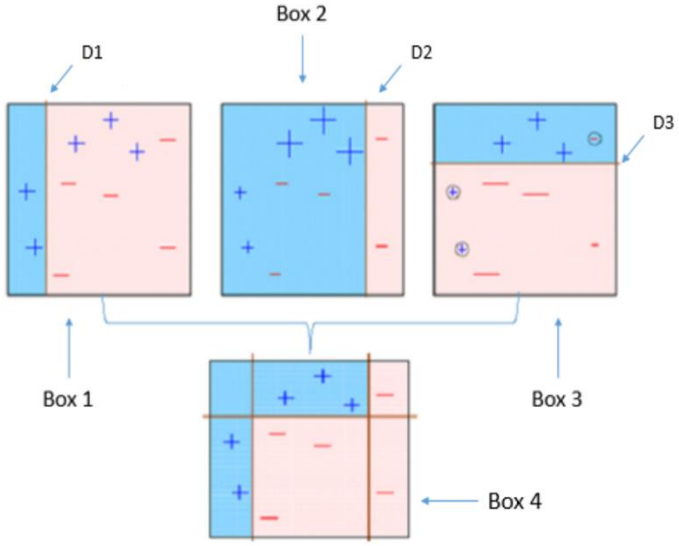
oluşturulması için kullanılan gözlemler bir dahaki ağaç oluşturması sırasında tekrar kullanılabilirdir [51].

Bagging yöntemi veri seti üzerinden bootstrap rastgele örnekleme yöntemi ile çalışıyor olmasıyla birçok ağaç üretebilmekte ve her bir ağacı bir karar verici olarak kullanabilmektedir. Bu sayede bagging yöntemi doğru sınıflandırma oranını arttıran, varyansı düşüren ve ezberlemeye karşı dayanıklı olan bir yöntemdir [50].

Bagging yöntemi eğitim setinde meydana gelebilecek küçük değişikliklerin model üzerinde önemli değişikliklere sebep olabileceği doğrusal olmayan kararsız modellerin kullanımında etkilidir [52].

3.3.4.2 Boosting Yöntemi

Topluluk öğrenme yöntemlerinden olan Boosting yöntemleri topluluk içerisindeki zayıf sınıflandırıcıları biraraya getirerek güçlü bir sınıflandırıcı oluşturulmasını sağlamaya çalışmaktadır [53]. Boosting yöntemlerinden en tanınmış olanı Adaboost yöntemidir [54]. AdaBoost yöntemi çoğu zayıf sınıflandırıcılar tarafından yanlış olarak tahmin edilen örneklere odaklanarak çalışır. Zayıf sınıflandırıcıların sonuçlarını tekrarlı olarak toplayarak en güçlü sınıflandırıcıyı oluşturmaya çalışır. Yanlış sınıflandırılan örneklere daha fazla ağırlık vererek (Şekil 3.2) [55] iteratif olarak yeni ağaçlar oluşturulur. Her iterasyonda arttırılarak gerçekleştirilen ağırlık güncelleştirilmesi belirlenen bir öğrenme tur sayısı kadar tekrar edilir. En son tahmin için ağırlıklı ortalama alınır ya da sınıf oylaması yapılır [53].



Şekil 3.2 Yanlış sınıflandırılan örneklerin ağırlıklandırılması ve iteratif ele alınması

3.3.4.3 Rastgele Ormanlar (Random Forests)

Topluluk öğrenme yöntemlerinden birisi olan Random Forests [50] diğer yöntemlere benzer olarak karar ormanındaki her bir ağacı veri seti içerisinden bootstrap tekniği ile rastgele örnekler seçerek oluşturur. Bagging yönteminden farklı olarak örnek seçimi ile birlikte değişkenlerin seçimi içinde rastgelelik söz konusudur. Her düğüm noktasında bütün değişkenler arasından belirlenen bir sayıda rastgele şekilde değişken seçilerek en iyi bölünme sağlayacak olan değişken belirlenip dallara ayırma işlemi belirlenen bu değişken ile başlar. Değişken seçiminde kullanılan random subspace yöntemi, veri setindeki tüm değişkenler içerisinden rastgele seçilen az sayıdaki değişken içerisinden değişken seçilmesi şeklinde uygulanır. Amaç ağacın her bir düğümünde en iyi dal bölümlenmesini sağlayacak değişkenlere ulaşmaktır [56].

Random Forests'ta karar ormanını oluşturan ağaçlar gözlemler ve değişkenler seçildikten sonra CART algoritması kullanılarak oluşturulur. CART algoritması dallara ayırma işleminin hangi değişken ile başlayacağına karar vermek için bilgi kazancını kullanır. Değişken seçimi sonrasında seçilen değişkenin cut-off değeri gini katsayısı ile belirlenir [50].

Gözlem ve değişken seçimi sonrasında eğitim seti ile eğitilen ağaçlardan oluşan karar ormanı tahmin işlemini şu şekilde gerçekleştirir:

Test veri seti içerisindeki her bir gözlem (satır elemanı, deney elemanı) oluşturulan her bir ağaca yerleştirilir. Her bir ağaç her bir gözlem için bir tahmini değer üretir. Bu işlem sonucunda her bir gözlem için her bir ağacın ürettiği tahmin sonuçları oylamaya tabi tutularak en fazla oy olan tahmini değer karar ormanın nihai tahmini değeri olarak sunulur. Sınıflandırma problemlerinde tüm ağaçlarca en fazla tahmin edilen sınıf karar ormanının nihai sınıf tahmini olarak sunulurken, regresyon problemlerinde tüm ağaçlarca tahmin edilen sürekli değerlerin ortalaması karar ormanının nihai tahmin değeri olarak sunulur. Nihai tahmin için ağaçlardan tahmin değerleri talep edilirken her bir ağacın daha önce hesaplanan hata oranları göz önüne alınarak ağaçlara ağırlık verilir. Düşük hata oranına sahip olan ağaçların ağırlıkları daha fazla olurken yüksek hata oranına sahip olan ağaçların ağırlıkları daha az olur. Böylece tahmin doğruluğu yüksek olan ağaçlara daha fazla söz hakkı tanınmış olur [50].

3.3.5 Naive Bayes

Bayes teoremine dayanan Naive Bayes sınıflandırıcısı, örnek veri içerisindeki sınıfların yapısını öğrenerek yeni elde edilen bir gözlemin hangi olasılıkla hangi sınıfa dahil olabileceğini gösteren bir yöntemdir [57]. Bayes Teoremi bir olayın gerçekleşme olasılığını bu olaydan daha önce gerçekleşen bir olayın olasılığını göz önüne alarak hesaplar. Netice olarak gerçekleşen olay B olmak üzere ve B'nin gerçekleşmesi A olayına bağımlı ise -B bağımlı olay ise- A olayının gerçekleştiği durumda B olayının gerçekleşmesi olasılığı ile ilgilenilir [58].

Bayes Teoremi:

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)} \tag{3.7}$$

P(B|A) sonsal olasılığı, P(A|B) koşullu olasılığı, P(A) A olayının bağımsız olasılığını, P(B) B olayının bağımsız olasılığını temsil etmektedir. Naive Bayes Sınıflandırıcısı Bayes Kuralını kullanarak her sınıf için incelenen örneklerin var olan sınıflara ait olma olasılıklarını tahminler [59].

$x_1, x_2, \dots, x_n$ gözlemleri C bağımlı değişkendeki sınıfları temsil etmek üzere sınıf tahmini aşağıdaki denklem ile yapılır:

$$P(C|\, x_1, \ldots, x_n) = P(C) \sum_{i=1}^{n} P(x_i|\, C) \tag{3.8}$$

Olasılık değerleri sınıf bilgisi verildiğinde her öznitelik için bireysel koşullu olasılıkların çarpımı ile elde edilir. Bu işlem sınıf bilgisi verildiğinde özniteliklerin bağımsız olduğu varsayımı ile yapılır [60].

Her bir sınıf için hesaplanan sonsal olasılıklar sonrasında sonsal olasılığı en yüksek olan sınıf tahmin sınıfı olur. Diğer bir ifade ile Naive Bayes sınıflandırıcı olasılığı en yüksek olan sınıfı, kazanan sınıf olarak açıklar [61].

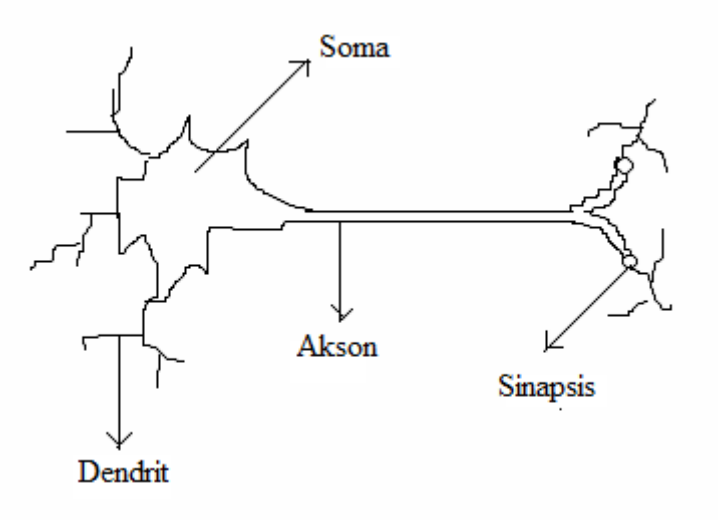
Naive Bayes Sınıflandırıcısı az sayıda örnek kullanarak gerekli parametreleri tahmin edebilmesi, ikili ve çoklu sınıflandırmalar için kullanılabilmesi ve girdi değişkenlerin sınıf sayılarının fazla olması durumlarında da başarılı çalışması sebebiyle avantaj sağlamaktadır [62].

3.3.6 Yapay Sinir Ağları

Yapay sinir ağları insan beyninin öğrenme yapısına benzer şekilde çalışan gözetimli öğrenme yaklaşımlarından birisidir. Yapay sinir ağları birbiri ile ilişkilendirilmiş birimlerden oluşur. Ağı oluşturan bu birimlere nöron adı verilir. Bu nöronlar (yapay sinir hücreleri) ağ içerisinde birbirleri ile ilişkilendirilmişlerdir. Yapay sinir ağlarına eğitim sırasında girdi ve çıktı değerleri bir arada verilerek verilebilecek girdi değerlerine karşılık oluşabilecek çıktı değerlerini tahmin etme yetisi kazandırılmaya çalışılır [63].

3.3.6.1 Biyolojik Sinir Hücresi Yapısı

Biyolojik sinir sisteminin temel yapı taşı olan nöronlar (sinir hücreleri) (Şekil 3.3) dendrit, soma, akson ve sinapsis yapılarından meydana gelir [64].



Şekil 3.3 Biyolojik sinir hücresi

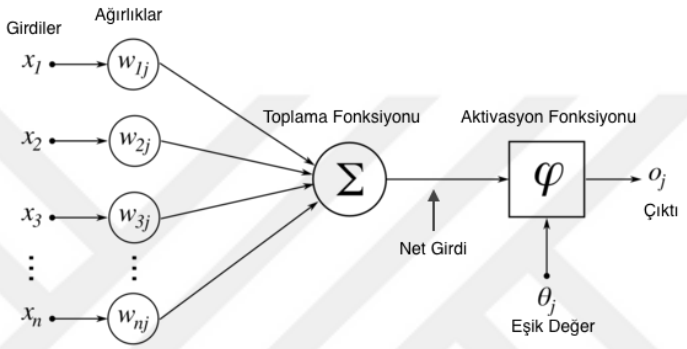
Dendritler alınan sinyalleri çekirdeğe (soma) iletmek ile görevlidirler. Hücre çekirdeği ve dendritler arasında gerçekleşen iletişim dendritlere göre farklılık gösterebilmekte ve hücre çekirdeği alınan sinyaller konusunda seçicilik yapabilmektedir. Bu durum dendritlerin hücre çekirdeğine ilettiği bilgilerin hücre çekirdeği tarafından farklı ağırlıklarla dikkate alınabildiğini ifade etmektedir. Somaların (çekirdeklerin) görevi dendritler tarafından iletilen sinyalleri bir merkezde toplamaktır. Soma kendisinde topladığı sinyalleri aksona iletir. Aksonun görevi çekirdekten gelen sinyalleri diğer hücrelere iletmektir. Sinyaller diğer hücrelere aktarılmadan önce snapsislerde ön işlemden geçirilir. Sinapsislerin görevi gelen sinyalleri belirli bir eşik değerine denk getiricek şekilde değiştirmek ve sonrasında diğer sinir hücrelerine aktarmaktır [63].

3.3.6.2 Yapay Sinir Hücresi Yapısı

Yapay sinir hücreleri biyolojik yapay sinir hücreleri taklit edilerek oluşturulmuştur ve benzer şekilde girdi değişkenleri (dendritler), snapsis ve dendritler arasında gerçekleşen etkileşim ile oluşan ve güncellenen ağırlıklar, toplama fonksiyonu (soma), aktivasyon fonksiyonu (akson ve sinapsis), ve çıktı (snapsisin ürettiği ve ilettiği sonuç) bileşenlerinden oluşur (Şekil 3.4) [65].

Girdi değişkenleri modelleme için kullanılacak olan bağımsız değişkenleri ifade etmektedir. Ağırlıklar yapay sinir ağlarının öğrenme yapısını ifade eden snapsis ve dendritler arasındaki etkileşim ile oluşup güncellenen değerlerdir. Güncellenen ağırlıklar öğrenme eylemi gerçekleştiğini ifade etmektedir [63]. Bir sinir hücresinden gelen bilgiler toplam fonksiyonuna gönderilmeden önce geldikleri bağlantıların

ağırlıkları ile çarpılır ve sonrasında çekirdeğe gönderilirler. Böylece girdilerin oluşacak olan çıktı değeri üzerinde olan etkileri ayarlanabilmektedir. Toplam fonksiyonu ağırlıklarla çarpıldıktan sonra gelen girdileri bir araya getirerek hücrenin net girdisini elde eder. Aktivasyon fonksiyonu hücreye gelen net girdiyi değerlendirerek bu hücrenin bu girdiye karşılık hangi çıktı değerini oluşturacağını belirler. Birçok aktivasyon fonksiyonu olmakla birlikte çok katmanlı algılayıcılarda en sık kullanılan sigmoid fonksiyonudur. Çıktı hücrenin çıktısını ifade etmektedir. Aktivasyon fonksiyonundan geçen net girdi ile üretilen değerdir. Bu çıktı değeri yapının nihai çıktısı ya da başka bir hücrenin girdisi olabilir [63].

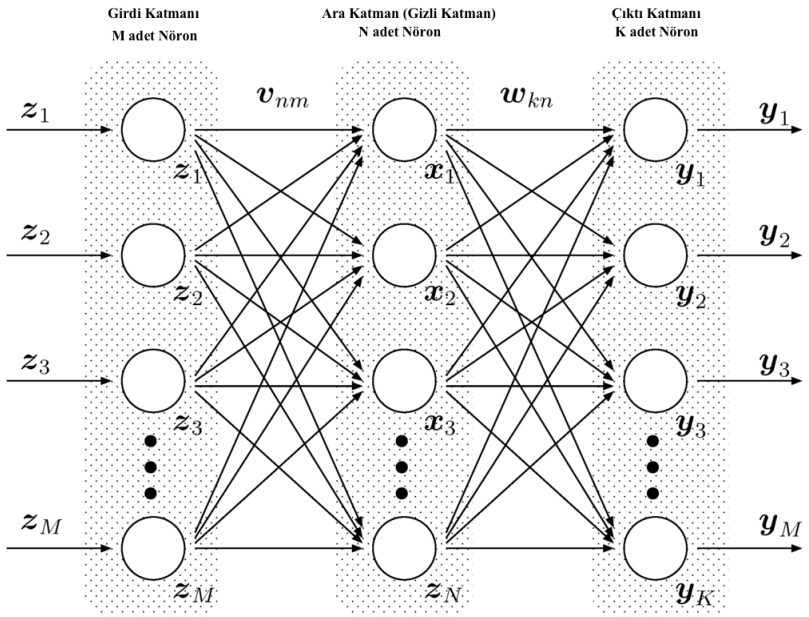


Şekil 3.4 Yapay sinir hücresi

Şekildeki tek bir yapay sinir hücresi yapısındaki x_n değerleri girdi değişkenlerini (bağımsız değişkenleri), w_{nj} değerleri girdi değişkenlerinin ağırlık katsayılarını ifade etmektedir. Transfer fonksiyonu tüm girdilerin ağırlıklı toplamlarını hesaplamaktadır. Net girdi ile ifade edilen bu toplam, aktivasyon (eşikleme) fonksiyonuna giriş bilgisi olarak iletilir. Aktivasyon fonksiyonunun ürettiği çıktı yapı tek katmanlı yapı ise tahmin değeri çok katmanlı yapı ise bir sonraki hücre için giriş değeri olarak üretilir.

3.3.6.3 Çok Katmanlı Yapay Sinir Ağları

Çok katmanlı yapay sinir ağları diğer bir ifade ile çok katmanlı algılayıcılar temelde üç yapıdan meydana gelir: giriş katmanı, ara katman ve çıkış katmanı. Giriş katmanı girdi değerlerini (bağımsız değişkenler) ifade eden katmandır. Ara katman giriş katmanından gelen bilgileri işler ve çıkışları bir sonraki katmana iletir. Ara katmanda birden fazla katman olabilir. Ara katmanlarda da birden fazla yapay sinir hücresi olabilir. Bir katmandaki tüm hücreler diğer katmandaki tüm hücreler ile bağlantılıdır [63].



Şekil 3.5 Çok katmanlı yapay sinir ağı [66]

Çok Katmanlı Algılayıcılar içerisinde en sık kullanılanı geriye yayınımlı (back propagation) ağlardır. Bu ağlarda öğrenme kuralı olarak Delta Öğrenme kuralı kullanılır. Delta Öğrenme Kuralının çalışma prensibi kabul edilebillir bir hata miktarı elde edinceye kadar ağ çıktısı ile gerçek çıktı arasındaki farkları minimize etmek için girdilerin ağırlıklarını değiştirmeye dayanmaktadır [63].

Çok katmanlı yapay sinir ağı uygulaması; örnek veri setinin toplanması, ağın topolojik yapısına karar verilmesi, öğrenme parametrelerine karar verilmesi, ağdaki ağırlıklara başlangıç değeri atamasının yapılması, örnek veri setinin ağa sunulması, ileri hesaplamaların yapılması, gerçek çıktılar ile tahmin edilen çıktıların karşılaştırılması, karşılaştırmaya göre ağırlıkların güncelleştirilmesi ve öğrenmenin tamamlanması basamanlarından oluşmaktadır [63].

3.4 Gözetimsiz Öğrenme Yöntemleri

3.4.1 Kümeleme Analizi

Kümeleme yöntemlerinin amacı; benzerlik matrislerini kullanarak gözlemleri ya da değişkenleri kümelemeye çalışmaktır. Oluşturulmaya çalışılan kümelerin kendi içinde homojen olması, birbirilerine göre heterojen olması istenir. Kümeleme yöntemleri; hiyerarşik ve hiyerarşik olmayan yöntemler başlıkları altında incelenir. İki yaklaşımda da amaç kümeler içi benzerliği yüksek kümeler arası benzerliği düşük yapmaya

çalışmaktır [44]. Çok değişkenli istatistiksel yöntemler altında istatistik literatüründe yer alan Kümeleme Analizi makine öğrenmesi alanında denetimsiz öğrenme başlığı altına karşılık gelmektedir.

3.4.1.1 Hiyerarşik Olmayan Kümeleme Yöntemleri

Hiyerarşik olmayan kümeleme yöntemlerinde veri setine göre belirlenen sayıda küme oluşturulur, oluşturulan kümelerin merkezleri hesaplanır ve bu işlem bütün gözlemlerin kümelere atanmasına kadar devam ettirilir. Dolayısıyla hiyerarşik olmayan yöntemlerde gözlemler kümeleme işlemi bitene kadar farklı kümelere girip çıkabilir [44].

3.4.1.2 Hiyerarşik Kümeleme Yöntemleri

Hiyerarşik kümeleme yöntemleri gözlemlerin fazla sayıda alt gruba ayrılmak istendiği durumlarda kullanılır. Kümeleme işlemi birleştirici ve bölümleyici şekilde yapılır.

Birleştirici kümeleme yöntemi şu şekilde uygulanır: Birer gözleme sahip gözlem sayısı kadar küme kendisine en fazla benzeyen kümeler ile birleştirilir. İki kümenin birleşiminden oluşan bu yeni kümeler aynı şekilde birbirlerine benzerliklerine göre tekrar birleştirilir. Tüm gözlemler tek bir küme de toplanana kadar bu işlemler tekrar edilir [44].

Bölünebilir hiyerarşik yöntem birleştirici hiyerarşik yöntemin tersten işleyen halidir. Tüm gözlemlerin bir arada olduğu küme iki alt kümeye ayrılır daha sonra oluşan bu kümelerde birbirlerine benzemeyen alt kümelere bölünür ve bu işlem gözlem sayısınca alt küme elde edilinceye kadar devam eder [44].

3.4.1.3 Hiyerarşik ve Hiyerarşik Olmayan Yöntemler Arasındaki Farklar

Hiyerarşik yöntemlerde küme sayısına dendogram sonuçlarına bakılarak karar verilirken, hiyerarşik olmayan yöntemlerde küme sayısı uygulama yapılmadan önce belirlenir [67].

Hiyerarşik kümeleme yöntemlerinde veri seti gözlemler ya da değişkenler bazında kümeleme işlemine sokulabilirken hiyerarşik olmayan yöntemlerde sadece gözlemlerin kümelenmesi mümkündür.

Hiyerarşik yöntemlerde gözlem birleştirmeleri uzaklık ve benzerlik matrisleri yardımıyla gerçekleştirilirken hiyerarşik olmayan kümeleme yöntemlerinde birleştirmeler ham veri matrisi ya da standart değerler matisi aracılığı ile gerçekleştirilir.

3.4.2 Temel Bileşen Analizi

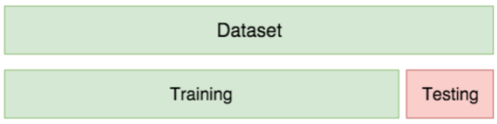
Temel bileşen analizi bir boyut indirgeme yaklaşımıdır. Temel bileşenler analizinin temel fikri, çok değişkenli verinin ana özelliklerini az sayıda değişkenle temsil etmektir. Boyut indirgeme gürültü etkisini ortadan kaldırmaktadır fakat bir miktar bilgi kaybı da söz konusu olmaktadır. Oluşabilecek bilgi kaybının görece hata ve gürültü ile kıyaslanabilir düzeyde küçük olması beklenir [44].

3.5 Model Doğrulama Yöntemleri

Regresyon ya da sınıflandırma problemlerinde eğitim verisi üzerinde kullanılan algoritmaların ürettiği sonuçların doğruluğunun değerlendirilmesi gerekmektedir. Bu yöntemlerin kullanılma gerekçesi temelde model başarı sonuçlarının daha doğru değerlendirilmesi çabasına dayanır. Değerlendirme yöntemlerinden en sık kullanılanlar Holdout, K-Katlı Çapraz Doğrulama, Bootstrap ve leave-one-out olarak sıralanabilir.

3.5.1 Sınama Seti Yaklaşımı (Holdout)

Holdout yönteminde veri seti eğitim ve test seti olarak iki parçaya ayrılır. Eğitim seti ile kurulan modelin performansı test seti ile ölçülür. Genellikle veri setinin 2/3'ü eğitim, geriye kalan 1/3'lük kısım ise test seti olarak ayrılır (Şekil 3.6).

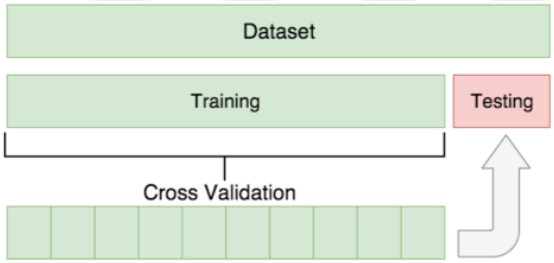


Şekil 3.6 Holdout yöntemi [68]

Gözlem sayısı az olduğunda test için yeteri kadar gözlem ayrılamayacak olması, model kurmak için ve modeli test etmek kullanılacak olan verinin miktari için ikilimde kalınması ve veri ayrımının isabetli gerçekleştirilememe ihtimali başka yöntemlerin gelişirilmesini zorunlu kılmıştır. Holdout yönteminde gözlem sayısı az olduğunda yeteri kadar test verisi ayrılamamaktadır. Bununla birlikte test ve eğitim veri setleri ayrımının olmaması durumunda hata oranları yanıltıcı olabilecektir. Bu iki problemden dolayı diğer yöntemlere yönelim olmuştur [69].

3.5.2 K-Katlı Çapraz Doğrulama

K-katlı çapraz doğrulama yönteminde veri seti k adet parçaya ayrılır. Belirlenen alt kümelerden birisi dışarıda bırakılır ve elde kalan diğer kümeler ile oluşturulan model dışarıda bırakılan küme ile test edilerek test hatası oluşturulur. Aynı işlem bütün parçaların test seti olarak dışarıda bırakılması şeklinde uygulanır ve k adet test hatası elde edilir. Kurulan modelin test hatası elde edilen k adet test hatasının ortalamasıdır. Örneğin 5 parçaya bölünmüş bir veri setinde birinci grup test için diğer 4 grup model kurmak için kullanılır. 4 grup ile kurulan modelin performansı birinci grup ile test edilir. Bu işlem her seferinde test setinin değiştirilmesiyle 5 kez tekrar edilir. Elde edilen 5 adet test hatasının ortalaması alınarak modelin test hatası belirlenmiş olur. 10'lu çapraz doğrulama sık kullanılan bir yaklaşım olsada oluşturulacak grup sayısı araştırmacı tarafından belirlenir [61].



Şekil 3.7 K-Katlı çapraz doğrulama yöntemi [68]

3.5.3 Leave-one-out

K-katlı çapraz doğrulama yönteminin özel bir hali olan Leave-one-out yönteminde k küme sayısı veri setindeki örnek sayısı n'e eşittir. Veri setinde örnek sayısı kadar küme sayısı vardır. Çalışma prensibi k-katlı çapraz doğrulama ile aynıdır. Model n-1 adet örnek ile eğitilir ve dışarda kalan 1 adet örnek ile test edilir. Bu işlem her bir örnek bir kere sınama için kullanılacak şekilde gerçekleştirilir. Model test hatası elde edilen tüm test hatalarının ortalaması alınarak elde edilir [61]. Büyük miktarda verinin eğitim için kullanılması doğru sınıflandırma yapma olasılığını arttırmaktadır fakat örnek sayısı kadar test yapılması gerektiği için çok büyük veri setlerinde kullanımı zorlaşmaktadır [69].

3.5.4 Yeniden Örnekleme Yöntemi (Bootstrap)

Veri seti içerisinden her defasında yerine koyarak farklı örnek seçip yeni bir veri seti oluşturabilir. Bu şekilde yeni veri setleri oluşturmak bootstrap metodu olarak adlandırılır. Genellikle küçük veri setleri için uygulanır. Yerine koymalı (bootstrap) şekilde birçok örnek veri seti içerisinden çekilir. Bir gözlem birden fazla örnekte yer alabilir. Elde edilen örneklemlerin her birisinin hatalarının ortalaması modelin test hatası olarak belirlenir [61].

3.6 Model Başarı Değerlendirme Yöntemleri

Sınıflandırma ya da regresyon problemleri için kurulan modellerin başarılarının test edilmesi gerekmektedir. Örneğin sınıflandırma için oluşturulan birden fazla sınıflandırıcının hangisinin seçilmesi gerektiği söz konusu olduğunda sınıflandırıcılardan daha iyi olanı seçmek için karışıklık matrisi(confusion matrix) ile doğruluk(accuracy), hata oranı(error rate), anma(recall), kesinlik(precision), F-ölçütü gibi değerler ile ROC eğrisi gibi yöntemlerden birisi kullanılabilir [69].

3.6.1 Karışıklık Matrisi

Sınıflandırma problemlerinde model başarısı model tarafından yapılan sınıf tahminlerinin doğru ve yanlış sınıflandırmasının değerlendirilmesi ile gerçekleştirilir. Karışıklık matrisi yardımıyla doğruluk, hata oranı, anma, kesinlik ve F-ölçütü hesaplanabilir.

Çizelge 3.2 Karışıklık Matrisi

		Tahmin Edilen Sınıf	
		Sınıf= 1	Sınıf= 0
Gerçek Sınıf	Sınıf= 1	a	b
	Sınıf= 0	c	d

a: True Pozitif (TP)

d: True Negatif (TN)

c: False Pozitif (FP)

b: False Negatif (FN)

Doğruluk: (TP+TN)/Hepsi

Doğruluk bir sınıfın gerçekte 1 iken model ile 1 olarak tahmin edilmesi ve bir sınıf gerçekte 0 iken model ile 0 olarak tahmin edilmesi durumlarının birlikte değerlendirilmesini ifade etmektedir.

Hata Oranı: (FN+FP)/Hepsi

Hata Oranı bir sınıfın gerçekte 1 iken model ile 0 tahmin edilmesi ve gerçekte 0 iken model ile 1 olarak tahmin edimesi durumlarının birlikte değerlendirilmesidir. Doğruluk ve hata oranı ile modelin sınıflandırma başarısı incelenir. Bu değerler sınıflandırma başarısını ifade eder. Fakat mutlak anlamda bir modelin daha iyi olduğunu göstermez.

Kesinlik: TP/(TP+FP)

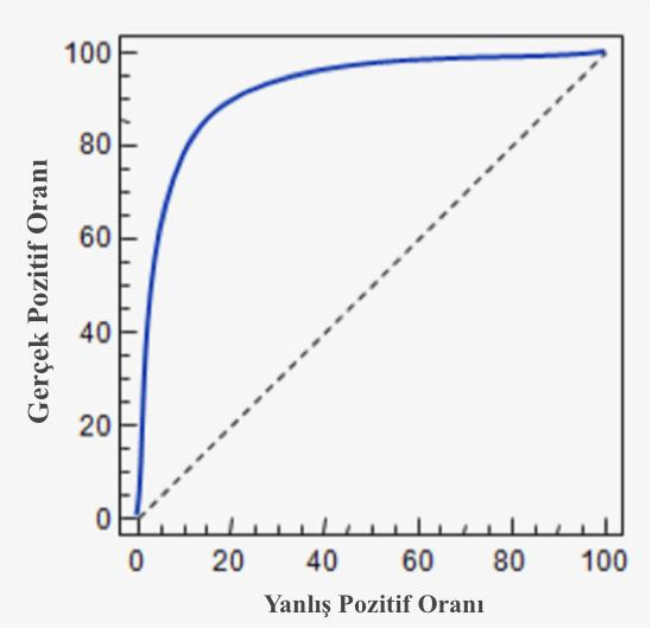
Anma: TP/(TP + FN)

Aralarında ters orantı olan kesinlik ve anma ölçütleri model karşılaştırmalarında kullanılır. Daha yüksek kesinlik ve anma değerine sahip modeller daha başarılıdır. Model karşılaştırma için anma ve kesinlik değerlerinin harmonik ortalaması da kullanılabilir. Bu hesaplamayı ifade eden F-Ölçütü:

F-Ölçütü = (2*kesinlik*anma)/(kesinlik + anma)

3.6.2 Alıcı İşlem Karakteristiği Eğrisi

Alıcı işlem karakteristiği eğrileri (ROC) oluşturulan birbirinden farkı sınıflandırıcıların karşılaştırılmasına olanak sağlar. Gerçek pozitif ve yanlış pozitif oranları kullanılarak oluşturulan istatistiksel bir karşılaştırma yöntemidir.



Şekil 3.8 ROC eğrisi [70]

İdeal durum için dikey olarak (0,0) noktasından (0,1) noktasına ve buradan grafiğin en sağ üstündeki (1,1) noktasına doğru çizilir. Bir çizgide (0,0) noktasından (1,1) noktasına doğru çizilir, bu çizgi rastgele yapılacak bir sınıflandırma tahmini ile sınıflandırma modeli ile yapılacak olan sınıflandırma tahmini arasında bir fark olup olmadığını ölçmek için kullanılır. ROC eğrisinin altındaki alan(AUC) hesaplanarak modelin tahmin başarısı değerlendirilir. ROC değerinin 1'e yakın olması modelin sınıflandırma başarısının iyi olduğunu gösterir, 0.5'e yakın olması modelin sınıflandırma başarısının rastgele alınan bir karardan daha iyi olamadığını gösterir, 0'a yakın bir değer modelin tüm örnekleri yanlış sınıflandırdığını ifade eder [69].

BÖLÜM 4

BÜYÜK VERİ UYGULAMALARI

Büyük veriden anlamlı bilginin hızlı ve ölçeklenebilir şekilde çıkarılması ihtiyacı yapay öğrenme algoritmalarının büyük veri üzerinde uygulanabilmesi çabasını doğurmuştur. Yapy öğrenme algoritmaları tıptan, hukuka, sağlık sektöründen telekominikasyona kadar birçok alanda insanlığa yüksek katma değerler sağlamıştır. Son yıllarda büyük verinin katkısıyla yapay öğrenme algoritmaları daha büyük ve daha çeşitli veri setleri üzerinde daha hızlı çalışarak geçmişe göre daha faydalı keşifler ve daha doğru tahminler yapılmasına olanak sağlamıştır [71].

4.1 Programlama Dilleri ve R Kütüphaneleri

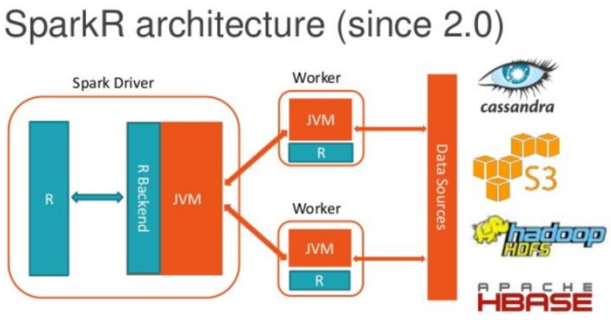
Büyük veri üzerinde uygulama geliştirebilmek için Java, Scala, Python ya da R dillerinden herhangi birisi kullanılabilir. Tez kapsamında R dili kullanılacağı için, R dili ile Spark arasında arayüz sağlayan kütüphaneler tanıtılacaktır. Büyük veri üzerinde makine öğrenmesi algoritmaları çalıştırmanın temel mantığı R içerisinde çalıştırılan fonksiyonların Spark ile dağıtık olarak işlenmesine dayanmaktadır. Local R fonksiyonlarından farklı olarak, dağıtık hesaplama için yazılmış olan fonksiyonlar ile bu işlemler yapılmaktadır. R için sağlanan arayüzden bir fonksiyon çalıştırıldığında arka planda bu fonksiyon dağıtık data frame işlemleri gerçekleştirmektedir. Böylece istatistiksel modellemeler, veri görselleştirmeleri ve veri manipülasyonu gibi konularda çok güçlü olan R ve büyük veriyi etkili bir şekilde kullanmaya olanak sağlayan Spark birbirini çok etkin bir şekilde tamamlamıştır.

4.1.1 SparkR

Apache Spark topluluğu tarafından büyük veri analitiği yapmak üzere geliştirilen SparkR, R üzerinden Apache Spark'ı kullanmak için arayüz sağlayan bir kütüphanedir.

SparkR büyük veri setleri üzerinde dağıtık data frame entegrasyonu sağlayarak veri manipülasyonu ve yapay öğrenme algoritmalarının kullanılmasına olanak sağlamaktadır [72].

SparkDataFrame Apache Spark'ın veri tutma türüdür. Birçok farklı veri kaynağını destekleyen SparkDataFrame, isimlendirilmiş kolonlara göre organize edilmiş dağıtık veri koleksiyonudur. Konsept olarak ilişkisel veri tabanlarındaki tablolara ya da R'daki data frame yapısına benzerdir fakat daha iyi optimize edilmiştir. SparkR ana düğüm noktası ve diğer işçi düğüm noktalarında kurulu olan (Şekil 4.1) [73] R'ın birlikte kullanılmasına olanak sağlayarak R ile büyük veri üzerinde analitik işlemler yapılabilmesine imkan sağlar.



Şekil 4.1 SparkR Mimarisi

4.1.2 DistributedR

Distributed R, büyük veri setleri üzerinde R kullanımı için açık kaynaklı yüksek performanslı bir kütüphanedir. Uygulama zamanını kısaltmak ve büyük veri setleri üzerinde analitik işlemler yapabilmek için düğümler (nodes) arasında iş paylaştırmaları yapar. Distributed R, R'ı dağıtık veri yapıları üzerinde paralel fonksiyonlar çalıştırarak, iş planlaması yaparak, farklı kaynaklardan veri yüklemesine olanak tanıyarak daha kuvvetli bir hale getirir. Daha çok dağıtık makine öğrenmesi görevlerinin entegre edilmesi için kullanılır. Şubat 2015'ten bu yana, Hewlett-Packard (HP) kullanıcılarına Distrbuted R için ticari anlamda destek sağlamaktadır [74].

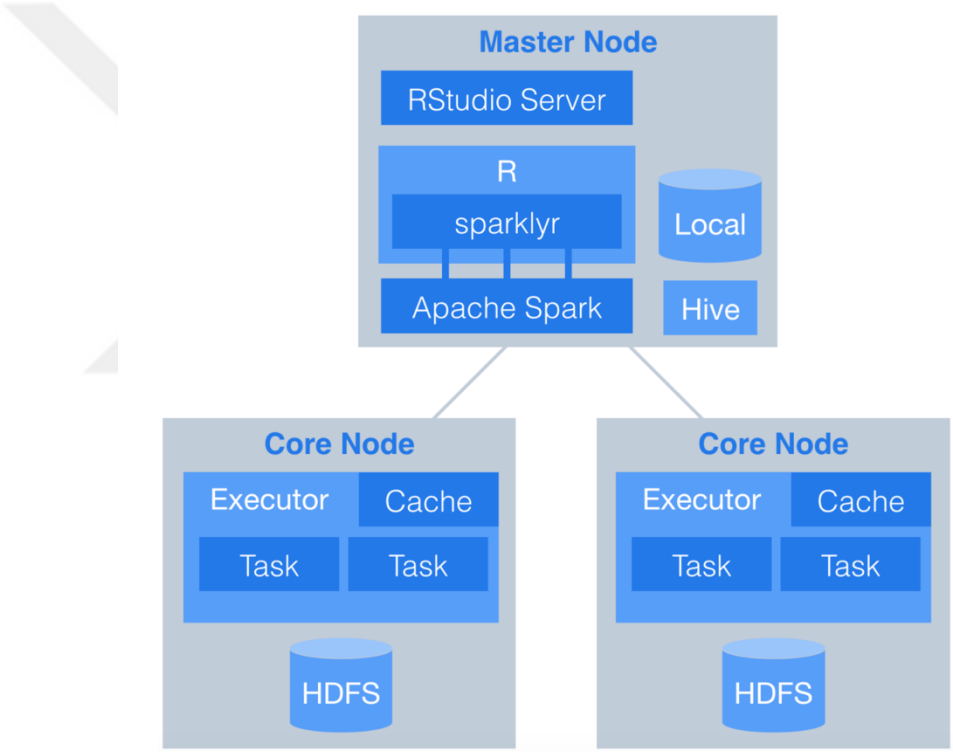
4.1.3 sparklyr

Rstudio tarafından büyük veri analitiği için geliştirilen sparklyr tıpkı SparkR gibi R üzerinden Apache Spark'ı kullanmak için bir arayüz sağlayan, büyük veri süreçleri için dağıtık hesaplamalar ve modellemeler yapmaya olanak sağlayan bir kütüphanedir [75].

sparklyr sağladığı dplyr desteği yerel hesaplamalarda kullanılan fonksiyonlar ile neredeyse aynı olan fonksiyonlar yardımıyla Spark kümesi üzerinde hesaplama yapma gücünden faydalanma imkanı sunmaktadır.

sparklyr interaktif şekilde dplyr ile veri manipülasyonu yapılmasına olanak sağlamasıyla birlikte Spark MLlib ve H20 kütüphanelerinde bulunan makine öğrenmesi algoritmalarının da kullanımı mümkün kılmaktadır.

Ana düğüm noktasına (master node) kurulu olan R sparklyr aracılığı ile Spark ile bağlanarak sağlanan arayüz yardımıyla büyük veri ile çalışma olanağı sağlar (Şekil 4.2) [75]. Spark'a ait API'ler ile iletişim kurularak ön yüzde R Studio ekranı ile çalışıyor olmasına rağmen arka tarafta hesaplamalar Spark ile yapılır.



Şekil 4.2 sparklyr mimarisi

4.2 Çalışma Ortamının Hazırlanması

Bu bölümde Amazon Web Servisi (AWS) üzerinde Elastic Map Reduce (EMR) hizmeti ile Hadoop, Spark, Hive, R Studio Server ve sparklyr ile büyük veri modellemesi yapılacaktır. İlgili araçların kurulumları yapılıp, analiz için veri setleri indirilip Hadoop dağıtık dosya sistemine aktarılacak sonrasında Hive tabloları oluşturulup uzaktan erişim ayarları ile kişisel bilgisayardan erişim sağlandıktan sonra Spark üzerinde R ile veri

manipülasyonu, veri görselleştirmesi ve makine öğrenmesi uygulaması yapılacaktır [76].

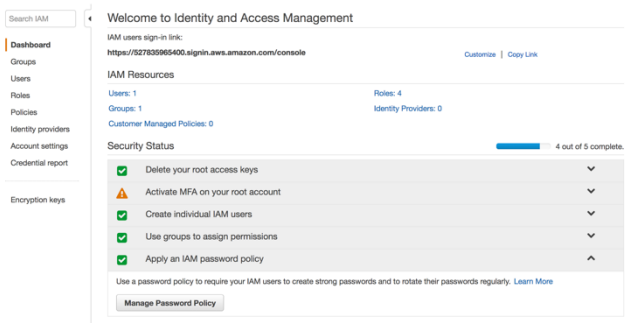
AWS üzerinde gerçekleştirilecek olan bu uygulama Microsoft, Google ya da diğer bulut hizmeti sağlayıcıları ile de gerçekleştirilebilir. AWS'in sunduğu Elastic Map Reduce (EMR) servisi büyük veri platform kurulum işlemlerini oldukça kolay ve hızlı hale getirmiştir [77]. Fakat EMR hizmeti ücret ile çalışmaktadır. Alternatif olarak bulut üzerinde ücretsiz çalışabilmek adına yine AWS'in free tier olarak adlandırılan hizmetlerinden faydalanıp büyük veri analitiği çalışmaları yapılabilir. EMR kurulum işlemleri daha hızlı ve kolay olduğu için tercih edilmiştir.

4.2.1 Amazon Web Servisi Kullanıcı Hesabı Oluşturma

aws.amazon.com adresinden bir AWS kullanıcı hesabı açılması ve kullandıkça ödeneceği taahhüdü ile kredi kartı bilgilerinin girilmesi gerekmektedir. Bilgiler girildikten sonra kayıt doğrulaması için amazon tarafından otomatik olarak gerçekleştirilen bir çağrı yapılacak bu çağrı ile verilecek olan doğrulama kodunun kayıt ekranına girilmesi gerekecektir. Sonrasında kayıt için girilen mail adresine gelecek olan doğrulama maili ile kayıt işlemi tamamlanacaktır.

4.2.2 Güvenlik Kimliği Ayarlamaları

Kullanıcı profili -> My Security Credentials -> Dashboard sırasıyla seçimler yapıldıktan sonra aşağıdaki "Security Status" ekranında gerekli ayarlamalar yapılır.

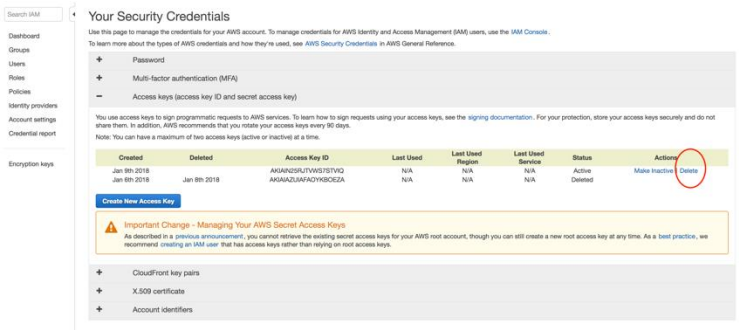


Şekil 4.3 Security Status ekranı

4.2.2.1 Erişim Anahtarının Silinmesi

AWS hesabı oluşturulduktan sonra otomatik olarak oluşturulan root_key'in silinmesi gerekiyor. Bu key ile root yetkisine sahip olunduğu için ortaya çıkabilecek birtakım

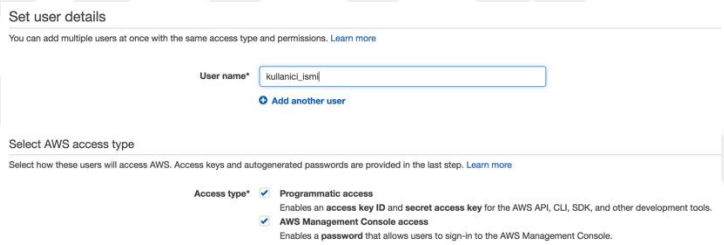
sorunların önüne geçmek adına “Delete your root access key” bölümü tıklanarak aşağıdaki şeklide silme işlemi yapılır.



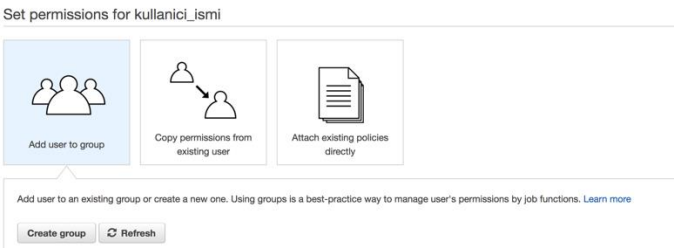
Şekil 4.4 Erişim anahtarının silinmesi

4.2.2.2 Bireysel Kullanıcı Oluşturma

Kullanıcı adı ve erişim bilgileri şekildeki gibi girildikten sonra “next” ile grup oluşturma sayfası açılır.



Şekil 4.5 Kullanıcı oluşturma



Şekil 4.6 Grup oluşturma

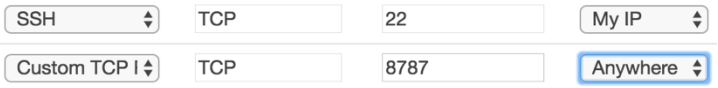
Burada da “create group” seçilerek grup oluşturma basamağına geçilir. Bu bölümde grup için hangi yetkilerin tanımlanacağı söz konusudur. Bunun için arama kısmına “elasticmapreduce” yazarak gelen sonuçların hepsi seçilir sonrasında grup ismi belirlenerek “create group” seçilerek “next”, “create user” ile kullanıcı ve grup oluşturma işlemi tamamlanır. Böylece Security Status ile ilgili 3 madde tamamlanmış

olur. Son olarak “Apply an IAM password policy” kısmından şifre kullanım politikası onaylanarak Güvenlik Kimliği ayarlamaları tamamlanır.

4.2.2.3 EC2 Ayarları

Amazon Elastik Bulut Hesaplama Platformu (Amazon Elastic Compute Cloud) EC2 kurulacak olan kümenin bileşenlerini üzerinde bulunduracaktır. Bu sebeple EC2 ile uzaktan erişim ile konuşabilmek adına bir key dosyası alınması gerekmektedir. Bu key dosyası oluşturulacak olan EMR ile otomatik olarakta oluşturulabilir. Sol üst tarafta bulunan “Services” kısmı tıklandıktan sonra “NETWORK & SECURITY” sekmesinin altından “Key Pairs” tıklanır, key pair ismi verilerek kaydedilir.

Takip eden bölüm EMR oluşturulduktan sonra yapılacaktır. Dolayısıyla EMR kurulumundan sonra bu sekmeye geri gelip aşağıdaki gerekli ayarlamalar yapılmalıdır. EMR kurulumundan sonra yine bu sekmeye gelip şu ayarlar yapılır: “NETWORK & SECURITY” sekmesinin altındaki “Security Groups” tıklanarak “default VPC security group” işaretlenir ve sayfanın sol alt tarafındaki “Inbounds” sonrasında “edit” tıklanır. Bu bölümde uzaktan terminal erişimi için gerekli SSH bağlantısı ayarı ve uzaktan web arayüzü ile R Studio Server bağlantısı için gerekli olan bağlantı ayarları eklenir. Eklemeler aşağıdaki şekilde yapıldıktan sonra “save” ile kaydedilerek çıkılır.

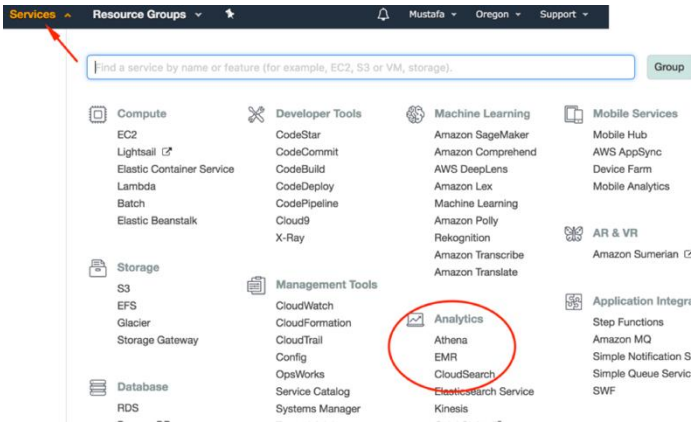


Şekil 4.7 Inbound ayarları

Uzaktan erişim sağlanmak istendiğinde eğer ip değişikliği söz konusu oldu ise bu durumda ssh bağlantısı yapılmadan önce “inbound” ayarlarından ssh kısmı tekrar “myip” olarak işaretlenmelidir. Bu durumda mevcut oturumdaki ip tanıtılmış olacak ve böylece ssh bağlantısı sağlanmış olacaktır. Aksi durumda bağlantı yapılamayacak ve “Operation timed out” hatası alınacaktır.

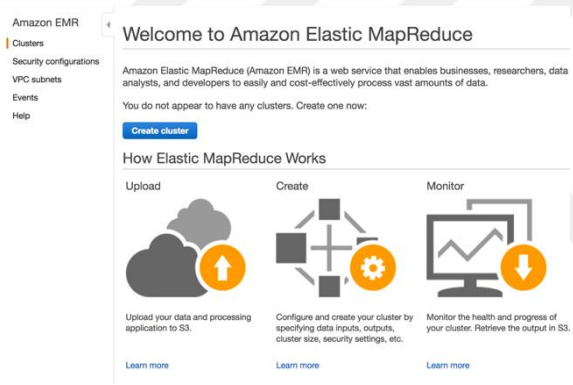
4.2.3 EMR ile Hadoop Kümesinin Oluşturulması

Sol üst köşedeki “Services” sonrasında “Analytics” bölümünden “EMR” seçilir. İlgili seçim aşağıdaki görselde ifade edilmiştir.



Şekil 4.8 EMR servisinin başlatılması

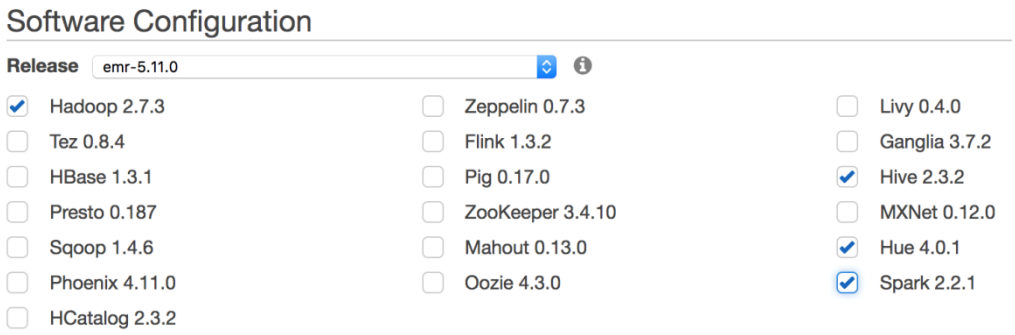
“create cluster” ile Elastic Map Reduce servisi ile Hadoop küme oluşturma işlemi başlatılır.



Şekil 4.9 EMR servisi ile küme kurulumunun başlatılması

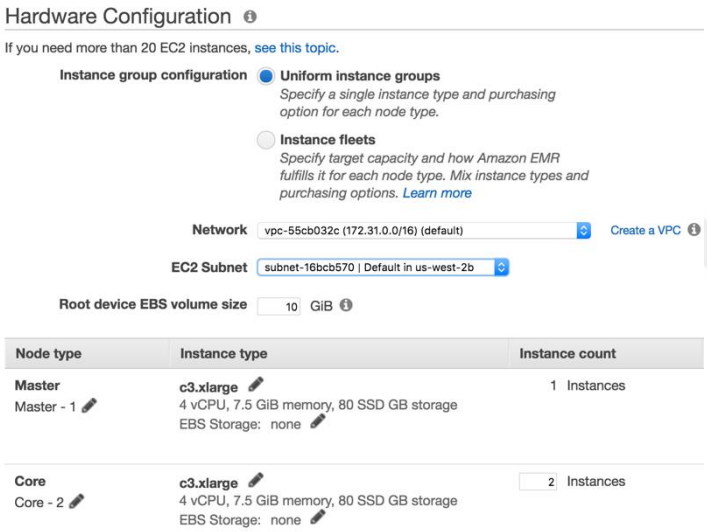
“Go to advanced options” seçilerek ayarlamalara devam edilir. Kümeyi ayağa kaldırmak için yazılım konfigürasyonu, donanım konfigürasyonu ve güvenlik ayarları yapılması gerekmektedir. Yazılım konfigürasyonu bölümünde kurulacak olan Hadoop kümesinin içine yüklenmesi istenilen bileşenler seçilir. Uygulama kapsamında Hadoop dağıtık dosyalama sisteminde bulunan verileri Hive tablo formatında işleyecek olduğumuz için Hive’ı, hesaplama ve makine öğrenmesi uygulamaları için Spark’ı ve gerekli olması durumunda arayüz ile interaktif hadoop yönetimi sağlayabilmek için HUE (Hadoop User Experience) bileşenleri seçilmiştir.

Not: spark’ın kurulmasıyla birlikte R otomatik olarak kurulmaktadır. R büyük veri dünyasında spark’ın içinde kurulu olarak gelmesiyle kalıcı olarak yer edinmiştir.



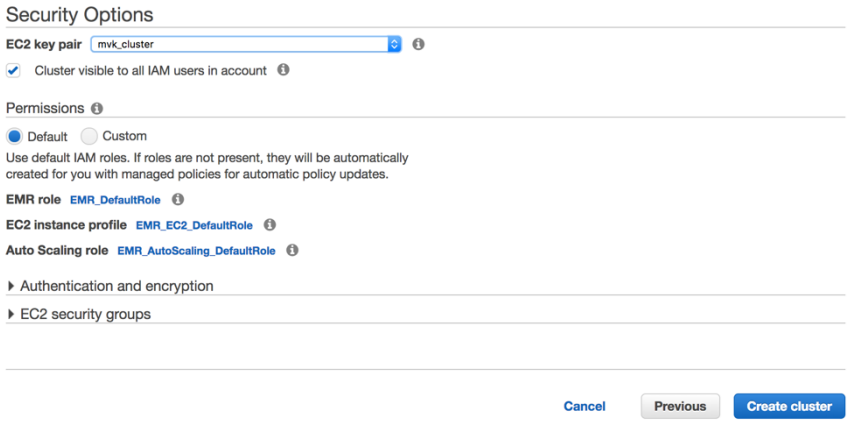
Şekil 4.10 Gerekli yazılımların seçilmesi

Donanım konfigürasyonu bölümü kurulacak olan fiziki sistemin özelliklerini barındırmaktadır. Uygulama kapsamında bir ana düğüm (name nod) ve iki işçi düğüm (worker node) kurulacaktır. Örnek uygulama için tüm makinalar c3.xlarge olarak seçilmiştir. Makinaların özellikleri ve gerekli ayarlamalar görselde ifade edilmiştir.



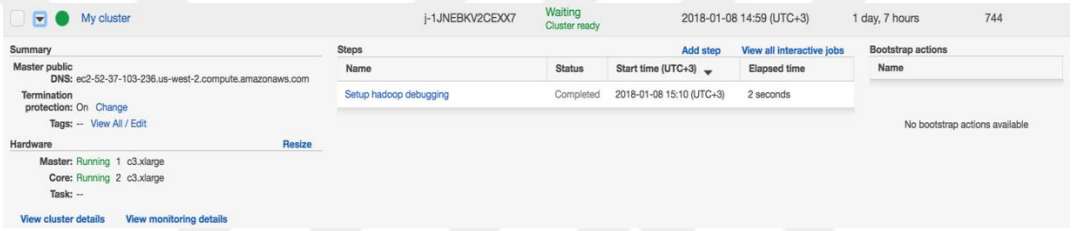
Şekil 4.11 Donanım ayarları

Güvenlik ayarlarında uzaktan erişim için kullanılacak olan EC2'ye key pair dosyası işaretlenmiştir.



Şekil 4.12 Güvenlik Ayarları

Diğer ayarlarda değişiklik yapılmayarak “Create Cluster” tıklanarak küme oluşturulur.



Şekil 4.13 Küme genel görünüm

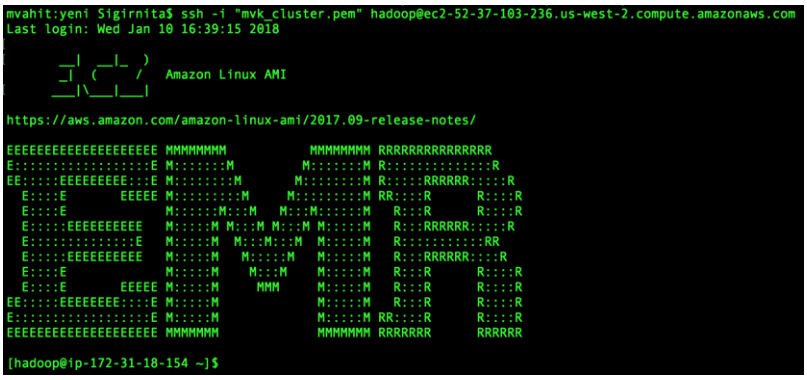
Bu işlemlerden sonra “My cluster” isimli Hadoop kümesi oluşturulmuştur. Ana ve işçi düğüm noktalarına yönelik bilgiler ve uzaktan erişim için gerekli olan “Master Public DNS” bilgileri verilmiştir.

Uzaktan Erişim Ayarları

Bulut üzerinde kurulan sisteme kişisel bilgisayar üzerinden uzaktan erişim sağlayabilmek için önceki bölümde yer alan “Master public DNS” kopyalanır. Terminal ile daha önce elde edilen EC2 key pair (mvk_cluster.pem) dosyasının bulunduğu klasöre gidilerek “Master Public DNS” ve key pair dosyası kullanılarak uzaktan bağlantı oluşturulur.

ssh -i "mvk_cluster.pem" hadoop@ec2-52-37-103-236.us-west-2.compute.amazonaws.com

ssh bağlantısı sonrasında aşağıdaki gibi bir ekran görüntüsü oluşacaktır:



Şekil 4.14 Uzaktan erişim sonrası ekran görüntüsü

4.2.5 R Studio Server Kurulumu ve Kullanıcı Oluşturma

SSH bağlantısı sonrasında EMR tarafından otomatik olarak oluşturulan hadoop kullanıcısı ile ana düğüm noktası üzerinde işlemlere devam edilecektir. Linux CentOS işletim sistemi kurulu olan EMR üzerinde aşağıdaki şekilde önce gerekli paketler için güncellemeler yapılır ve sonrasında R Studio Server kurulur.

CentOS için güncelleme ve yüklemeler:

sudo yum update

sudo yum install libcurl-devel openssl-devel

R Studio server kurulumu ve yüklenmesi:

wget https://download2.rstudio.org/rstudio-server-rhel-1.1.383-x86_64.rpm

sudo yum install --nogpgcheck rstudio-server-rhel-1.1.383-x86_64.rpm

Kullanıcı Oluşturma:

R Studio'ya erişim ve analizleri ayrık ve belirli bir şekilde yapabilmek adına R Studio kullanıcısı ve dizinleri oluşturulur.

Kullanıcı oluşturulması ve şifresinin belirlenmesi:

sudo useradd -m rstudio-user

sudo passwd rstudio-user

Hadoop dağıtık dosya sistemi üzerinde R Studio için dizin oluşturulup bu dizine gerekli yetkilerin verilmesi gerekmektedir.

Dizin oluşturulması:

hadoop fs -mkdir /user/rstudio-user

Dizine gerekli izinlerin verilmesi:

hadoop fs -chmod 777 /user/rstudio-user

rstudio-user kullanıcısının açılmasından sonra rstudio-user kullanıcısına geçilip veri indirme ve verinin HDFS üzerine dağıtılması işlemleri yapılacaktır.

Kullanıcı değiştirme:

su rstudio-user

4.2.6 Verinin HDFS Üzerine Dağıtılması

Veri setinin indirileceği dizinin oluşturulması:

mkdir /tmp/flights

Dizin oluşturma işleminden sonra veri setinin bu dizine indirilmesi gerekmektedir. Aşağıdaki döngü yardımıyla indirilecek olan verinin boyutu yaklaşık 25 GB'dır. R içerisinde bulunan flights isimli veri seti bu veri setinin örneklemi durumundadır.

Flights veri setinin bir for döngüsü aracılığı ile indirilmesi:

```
for i in {1987..2008}
  do
    echo "$(date) $i Download"
    fnam=$i.csv.bz2
    wget -O /tmp/flights/$fnam http://stat-computing.org/dataexpo/2009/$fnam
    echo "$(date) $i Unzip"
    bunzip2 /tmp/flights/$fnam
  done
```

Havayolu şirketlerine ait veri setinin indirilmesi:

wget -O /tmp/airlines.csv

http://www.transtats.bts.gov/Download_Lookup.asp?Lookup=L_UNIQUE_CARRIERS

Havaalanlarına ait veri setinin indirilmesi:

wget -O /tmp/airports.csv

https://raw.githubusercontent.com/jpatokal/openflights/master/data/airports.dat

Verilerin HDFS'e dağıtılması

CSV formatında ana düğüm noktasına indirilen verilerin hadoop dağıtık dosyalama sistemi üzerine aktarılması için önce dizinler oluşturulur sonrasında indirilen veriler aktarılır.

Dizinlerin oluşturulması:

hadoop fs -mkdir /user/rstudio-user/flights/

hadoop fs -mkdir /user/rstudio-user/airlines/

hadoop fs -mkdir /user/rstudio-user/airports/

Verilerin taşınması:

hadoop fs -put /tmp/flights /user/rstudio-user/

hadoop fs -put /tmp/airlines.csv /user/rstudio-user/airlines

hadoop fs -put /tmp/airports.csv /user/rstudio-user/airports

Hive Tablolarının Oluşturulması:

Bu kısımda HDFS üzerine dağıtılan csv formatındaki dosyalar SQL arayüzü ile okunabilirlik ve işleyebilirlik kazanabilmesi adına Hive tabloları oluşturularak kullanılacaktır. Terminal üzerinden "hive" yazıldığında Hive çalışacaktır sonrasında kullanmak istediğimiz verilere ait meta data özellikleri tablo oluşturma komutu ile oluşturulacak ve veriler oluşturulan bu meta data özelliklerine göre Hive tablolarına yüklenecek/dönüştürülecektir. Tablo oluşturma ile ilgili ek A,B,C incelenebilir.

4.2.7 R Studio ve Spark Bağlantısı

R Studio Server Arayüz Erişimi:

Önceki bölümlerde almış olduğumuz ana düğüm noktası DNS bilgisi aşağıdaki şekilde yazılarak sonuna 8787 portu girildiğinde R Studio arayüz giriş ekranı gelecektir. Önceki bölümlerde oluşturulan kullanıcı adı ve şifre girildiğinde web üzerinden R Studio'ya erişim sağlanmış olacaktır.

username: rstudio-user

password: rstudio-user

Spark ve R Bağlantısının Yapılması:

R Studio içerisinde yeni bir çalışma dosyası ve proje oluşturularak çalışmaya başlanır. Bunu için sağ üst taraftaki proje kısmından dizin ve isim bilgileri girilerek yeni proje oluşturulur ve sol üst tarafta yer alan sekmelerden R Script dosyası oluşturulur.

Ana düğüm noktası üzerinde kurulu olan R Studio Server ile Spark'ın birbirine bağlanması gerekmektedir. Bu işlem için kullanılacak olan sparklyr bize spark'ın fonksiyonlarını R arayüzü ile kullanma imkanı sağlayacaktır. sparklyr kütüphanesinin indirilmesi ve kurulması:

install.packages("sparklyr")

library(sparklyr)

Spark Home dizinin tanıtılması ve sonrasında spark bağlantısını sağlanması:

Sys.setenv(SPARK_HOME="/usr/lib/spark")

sc <- spark_connect(master = "yarn-client")

Bağlantı sonrasında "Connection" sekmesinde Hive tabloları görünür hale gelecektir. Tablolara tıklanarak incelemeler yapılabilir. Hive tablolarının 1000'er adet gözlemi görülebilecektir.

Tabloların geçici belleğe yüklenmesi:

HDFS üzerinde duran ve Hive tabloları şeklinde görüntülediğimiz veri setlerini Spark ile kullanmak için tabloların geçici belleğe yüklenmesi gerekmektedir. Yükleme sonrası tabloları R aracılığı ile okuyabilmek için spark bağlantı bilgisi ve tablo isimlerinin R ile ilişkilendirilmesi gerekir.

Tabloların geçici belleğe yüklenmesi:

tbl_cache(sc, 'flights')

tbl_cache(sc, 'airlines')

tbl_cache(sc, 'airports')

Tabloların R ile etkileşiminin sağlanması:

flights_tbl <- tbl(sc, 'flights')

airlines_tbl <- tbl(sc, 'airlines')

airports_tbl <- tbl(sc, 'airports')

4.3 Büyük Veride Keşifçi Veri Analizi Uygulamaları

R ve Spark'ın entegrasyonuna imkan sağlayan sparklyr kütüphanesi bu entegrasyon ile birlikte sağladığı dplyr [78] desteği ile de büyük veri üzerinde veri manipülasyonunu da kolay hale getirmiştir. Böylece R'da kullanılan dplyr fonksiyonlarının bir çoğu büyük veri üzerinde kullanılabilmektedir. dplyr ile verilen komutlar Spark SQL API'si aracılığı ile büyük veri üzerinde sorgular ve manipülasyonlar yapma imkanı sağlamaktadır.

Havayolu şirketlerine göre uçuş sayıları:

```
flights_tbl %>%
  group_by(uniquecarrier) %>%
  summarise(count=n()) %>%
  arrange(desc(count))
```

Şekil 4.15 Uçuş sayıları R kodu

```
# Source:     lazy query [?? x 2]
# Database:   spark_connection
# Ordered by: desc(count)
   uniquecarrier    count
   <chr>            <dbl>
 1 DL            16547870
 2 WN            15976022
 3 AA            14984647
 4 US            14075530
 5 UA            13299817
 6 NW            10292627
 7 CO             8145788
 8 MQ             3954895
 9 TW             3757747
10 HP             3636682
# ... with more rows
```

Şekil 4.16 Uçuş sayıları R çıktısı

Havayolu şirketlerinin tam isimleri ve uçuş sayıları:

```
flights_tbl %>%
  group_by(uniquecarrier) %>%
  summarise(count=n()) %>%
  arrange(desc(count)) %>%
  left_join(carriers_tbl,
  by = c("uniquecarrier" = "code"))
```

Şekil 4.17 Uçuş sayıları ve tam isimler R kodu

```
# Source:     lazy query [?? x 3]
# Database:   spark_connection
# Ordered by: desc(count)
   uniquecarrier    count description
   <chr>            <dbl> <chr>
 1 DL            16547870 Delta Air Lines Inc.
 2 WN            15976022 Southwest Airlines Co.
 3 AA            14984647 American Airlines Inc.
 4 US            14075530 US Airways Inc.
 5 UA            13299817 United Air Lines Inc.
 6 NW            10292627 Northwest Airlines Inc.
 7 CO             8145788 Continental Air Lines Inc.
 8 MQ             3954895 Envoy Air
 9 TW             3757747 Trans World Airways LLC
10 HP             3636682 America West Airlines Inc.
# ... with more rows
```

Şekil 4.18 Uçuş sayıları ve tam isimler R çıktısı

Kalkış havaalanlarına göre uçuş sayıları:

```
flights_tbl %>%
  group_by(origin) %>%
  summarise(count=n())
```

Şekil 4.19 Havaalanlarına göre uçuş sayıları R kodu

```
# Source:   lazy query [?? x 2]
# Database: spark_connection
   origin  count
   <chr>   <dbl>
 1 BGM     26486
 2 DLG      4924
 3 PSE      2883
 4 INL       290
 5 MSY    955377
 6 GEG    263177
 7 BUR    581007
 8 SNA    822099
 9 GTF     61067
10 GRB     87128
# ... with more rows
```

Şekil 4.20 Havaalanlarına göre uçuş sayıları R çıktısı

Havayolu şirketlerinin gecikme ortalamlarına göre sıralanması:

```
flights_tbl %>%
  group_by(uniquecarrier) %>%
  summarise(n=n(),
  mean = mean(depdelay)) %>%
  arrange(desc(mean)) %>%
  left_join(carriers_tbl,
  by = c("uniquecarrier" = "code"))
```

Şekil 4.21 Gecikme ortalamalarına göre sıralama R kodu

```
# Source:     lazy query [?? x 4]
# Database:   spark_connection
# Ordered by: desc(mean)
   uniquecarrier       n  mean description
   <chr>           <dbl> <dbl> <chr>
 1 EV            1697172 13.5  ExpressJet Airlines Inc.
 2 YV             854056 12.9  Mesa Airlines Inc.
 3 B6             811341 11.3  JetBlue Airways
 4 FL            1265138 10.3  AirTran Airways Corporation
 5 UA           13299817  9.67 United Air Lines Inc.
 6 DH             693047  9.61 Independence Air
 7 PI             873957  9.56 Piedmont Aviation Inc.
 8 OH            1464176  9.31 PSA Airlines Inc.
 9 MQ            3954895  9.22 Envoy Air
10 WN           15976022  9.08 Southwest Airlines Co.
# ... with more rows
```

Şekil 4.22 Gecikme ortalamalarına göre sıralama R çıktısı

En fazla gecikme yapan havayolu şirketine ait uçaklar ve gecikme süreleri:

```
flights_tbl %>%
  filter(uniquecarrier == "EV") %>%
  group_by(tailnum) %>%
  summarise(n=n(), mean=mean(depdelay)) %>%
  arrange(desc(mean))
```

Şekil 4.23 Uçaklar ve gecikme süreleri R kodu

```
# Source:     lazy query [?? x 3]
# Database:   spark_connection
# Ordered by: desc(mean)
   tailnum     n  mean
   <chr>   <dbl> <dbl>
 1 N673BR    690  19.9
 2 N687BR    561  19.7
 3 N659CA   3160  18.9
 4 N698BR   2270  18.2
 5 N761EV   5399  18.1
 6 N355CA   3076  17.4
 7 N694BR    589  17.3
 8 N707EV   9711  17.1
 9 N713EV   9590  17.1
10 N390CA   2863  17.1
# ... with more rows
```

Şekil 4.24 Uçaklar ve gecikme süreleri R çıktısı

En fazla geciken havayolları için ikili karşılaştırmalar:

```
data <- flights_tbl %>%
  filter(uniquecarrier %in% c("EV","YV","B6","UA")) %>%
  filter(month == 12 & depdelay > 300) %>%
  mutate(deppdelay = depdelay/60) %>%
  select(uniquecarrier, deppdelay) %>%
  collect()

with(data, pairwise.t.test(deppdelay, uniquecarrier))
```

Şekil 4.25 İkili karşılaştırmalar R kodu

```
        Pairwise comparisons using t tests with pooled SD

data:  deppdelay and uniquecarrier

   B6      EV      UA
EV < 2e-16 -       -
UA 0.00383 < 2e-16 -
YV 0.35380 < 2e-16 0.00031

P value adjustment method: holm
```

Şekil 4.26 İkili karşılaştırmalar R çıktısı

Büyük veri üzerinde indirgemesi sağlanabilen her veri R ortamına çekilerek R'da var olan tüm fonksiyonlardan yararlanılabilir.

4.4 Büyük Veri Görselleştirme

Spark üzerinde işlenen verilerin yine R aracılığı ile görselleştirilmesi mümkündür. Bunun için çok büyük miktarda olan veri öncelikle spark üzerinde toplulaştırılarak diğer bir ifade ile özet istatistikleri oluşturularak indirgenir ve sonrasında R ortamına bu indirgenmiş veri taşınarak görselleştirme işlemleri sağlanır. R ortamına verinin indirgenmesi için collect() fonksiyonu kullanılır.

En fazla gecikme ortalamasına sahip şirketlerin barplot ile görselleştirilmesi

Gerekli verinin oluşturulması:

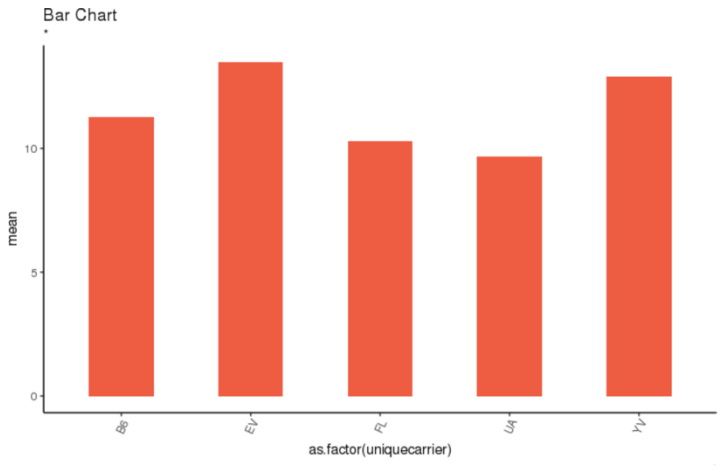
```
flights_tbl %>%
  group_by(uniquecarrier) %>%
  summarise(n=n(),
  mean = mean(depdelay)) %>%
  arrange(desc(mean)) %>%
  left_join(carriers_tbl,
  by = c("uniquecarrier" = "code"))

theme_set(theme_classic())
data <- flights_tbl %>%
  filter(uniquecarrier %in%
  c("EV","YV","B6","UA","AA")) %>%
  group_by(uniquecarrier) %>%
  summarise(mean = mean(depdelay)) %>%
  collect()
```

Şekil 4.27 Verinin oluşturulması verinin oluşturulması

Verinin görselleştirilmesi:

```
library(ggplot2)
g <- ggplot(data, aes(as.factor(uniquecarrier),mean))
g + geom_bar(stat="identity", width = 0.5, fill="tomato2") +
  labs(title="Bar Chart",
       subtitle="*",
       caption="*") +
  theme(axis.text.x = element_text(angle=65, vjust=0.6))
```

Şekil 4.28 Verinin görselleştirilmesi



Şekil 4.29 Gecikmelerin barplot ile görselleştirilmesi

Havayolu şirketlerinin saat cinsinden geç kalma sürelerinin boxplot ile incelenmesi

Gerekli verinin oluşturulması:

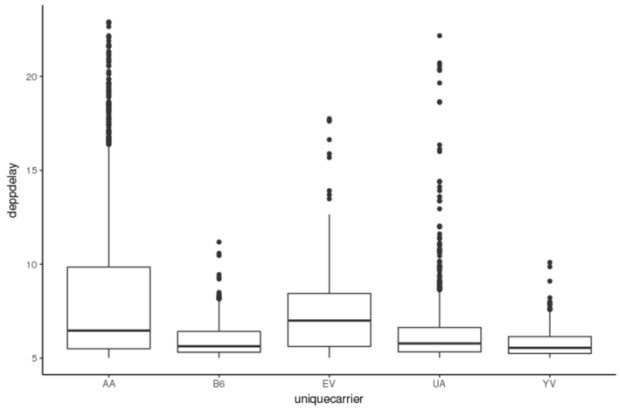
```
data <- flights_tbl %>%
  filter(uniquecarrier %in% c("EV","YV","B6","UA","AA")) %>%
  filter(month == 12 & depdelay > 300) %>%
  mutate(deppdelay = depdelay/60) %>%
  select(uniquecarrier, deppdelay) %>%
  collect()
```

Şekil 4.30 Gerekli verinin oluşturulması

Verinin ggplot ile görselleştirilmesi:

```
ggplot(data, aes(uniquecarrier, deppdelay)) + geom_boxplot()
```

Şekil 4.31 Verinin görselleştirilmesi



Şekil 4.32 Geç kalma sürelerinn boxplot ile görselleştirilmesi

Şubat aylarına dair saat cinsinden gecikme süreleri histogramı

Gerekli verinin oluşturulması:

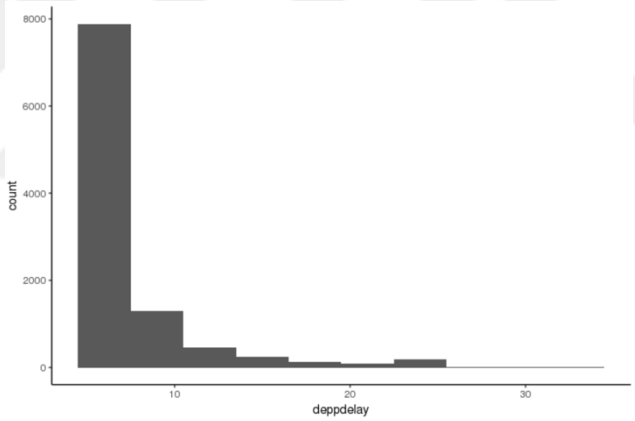
```
data <- flights_tbl %>%
  filter(month == 12 & depdelay > 300) %>%
  mutate(deppdelay = depdelay/60) %>%
  select(uniquecarrier, deppdelay) %>%
  collect()
```

Şekil 4.33 Gecikme süreleri verisinin oluşturulması

Verinin görselleştirilmesi:

```
g <- ggplot(data, aes(deppdelay))
g + geom_histogram(binwidth = 3)
```

Şekil 4.34 Gecikme süreleri verisinin görselleştirilmesi



Şekil 4.35 Gecikme süreleri histogramı

4.5 Büyük Veride Makine Öğrenmesi Uygulaması

Bu bölümde titanik veri seti ile titanik gemisinde bulunan yolcuların ölme ya da ölmeme durumlarını ifade eden değişken için bir sınıflandırma problemi ele alınacaktır. Değişken dönüşümü, veri setinin test ve eğitim olarak ayrılması, modellerin kurulması, modellerin tahmin performansı karşılaştırılmasının yapılması ve modellerin sistem üzerinde çalışma performanslarının karşılaştırılması yapılacaktır.

Şu adresten indirilen veri seti R Studio arayüzü üzerinden files -> upload ile çalışma dizini içerisine yüklenir. Gerekli kütüphaneler yüklenmesi ve çağırılması:

Kütüphanelerin kurulması:

```
kutuphaneler<-c("dplyr","tidyr","sparklyr","ggplot2","purrr","readr")
install.packages(kutuphaneler)
lapply(kutuphaneler, library, character.only = TRUE)
```

Şekil 4.36 Gerekli kütüphanelerin kurulması

Veri setinin okutulması ve değişken isimlerinin küçük harfe çevrilmesi:

```
titanik <- read_csv("train.csv")
names(titanik)<-tolower(names(titanik))
```

Şekil 4.37 Veri setinin okutulması ve değişken isimlerinin düzenlenmesi

Spark bağlantısının yapılması ve verinin geçici belleğe taşınması:

```
sc <- spark_connect(master = "local")
titanik_tbl <- copy_to(sc, titanik, overwrite = TRUE)
```

Şekil 4.38 Spark bağlantısının yapılması ve verinin geçici belleğe taşınması

4.5.1 Veri Düzenleme ve Değişken Dönüşümleri

dplyr, SPARK SQL API'sini kullanarak veri üzerinde manipülatif işlemler ve keşifçi veri analizine yönelik işlemler yapılmasına olanak sağlar. Değişken dönüştürme, yeni değişken oluşturma, değişken ya da gözlem seçimi gibi işlemler dplyr arayüzü ile kolayca yapılabilir. Örnek olması açısından "sibsp" ve "parch" değişkenleri toplanarak "family_size" isimli yeni bir değişken oluşturulmuştur. "embarked" değişkeni içerisindeki kayıp gözlemler silinmiş "age" değişkeninde var olan kayıp gözlemler için age değişkeninin ortalaması kullanılarak doldurma işlemi yapılmıştır.

dplyr kullanarak Spark SQL API'si aracılığı ile değişken dönüşümü:

```
titanik2_tbl <- titanik_tbl %>%
  mutate(family_size = sibsp + parch + 1L) %>%
  filter(!is.na(embarked)) %>%
  mutate(age = if_else(is.na(age), mean(age), age))
```

Şekil 4.39 Spark SQL API'si ile değişken dönüşümü

Spark ML API'si ile değişken dönüşümü:

```
titanik_final_tbl <- titanik2_tbl %>%
  mutate(family_size = as.numeric(family_size)) %>%
  sdf_mutate(
    family_sizes = ft_bucketizer(family_size, splits =
c(1,2,5,12)))
```

Şekil 4.40 Spark MLLib API'si ile değişken dönüşümü

4.5.2 Modellerin Eğitilmesi

Eğitim ve test veri setlerinin oluşturulması:

```
bolumler <- titanik_final_tbl %>%
  mutate(survived = as.numeric(survived), sibsp =
as.numeric(sibsp), parch = as.numeric(parch)) %>%
  select(survived, pclass, sex, age, sibsp, parch, fare,
embarked, family_sizes) %>%
  sdf_partition(egitim = 0.75, test = 0.25, seed = 8585)
```

Şekil 4.41 Eğitim ve test veri setlerinin oluşturulması

Tablo referanslarının oluşturulması:

```
egitim_tbl <- bolumler$egitim
test_tbl <- bolumler$test
```

Şekil 4.42 Tablo referanslarının oluşturulması

Bağımlı ve bağımsız değişkenlerin tanımlanması:

```
model <- formula(survived ~ pclass + sex + age + sibsp + parch + fare +
embarked + family_sizes)
```

Şekil 4.43 Bağımlı ve bağımsız değişkenlerin tanımlanması

Modellerin eğitilmesi:

```
ml_log <- ml_logistic_regression(egitim_tbl, model) #Lojistik regreson
ml_dt <- ml_decision_tree(egitim_tbl, model) #karar agaci
ml_rf <- ml_random_forest(egitim_tbl, model) #random forests
ml_gbt <- ml_gradient_boosted_trees(egitim_tbl, model) #gdt
ml_nb <- ml_naive_bayes(egitim_tbl, model) #naive bayes
ml_nn <- ml_multilayer_perceptron(egitim_tbl, model, layers = c(10,15,2)) #ysa
```

Şekil 4.44 Modellerin eğitilmesi

Modellerin bir araya toplanması:

```
ml_modeller <- list(
  "Logistic" = ml_log,
  "Decision Tree" = ml_dt,
  "Random Forest" = ml_rf,
  "Gradient Boosted Trees" = ml_gbt,
  "Naive Bayes" = ml_nb,
  "Neural Net" = ml_nn
)
```

Şekil 4.45 Modellerin bir araya toplanması

4.5.3 Modellere Göre Değişken Önem Düzeylerinin İncelenmesi

Önem düzeylerinin hesaplanması:

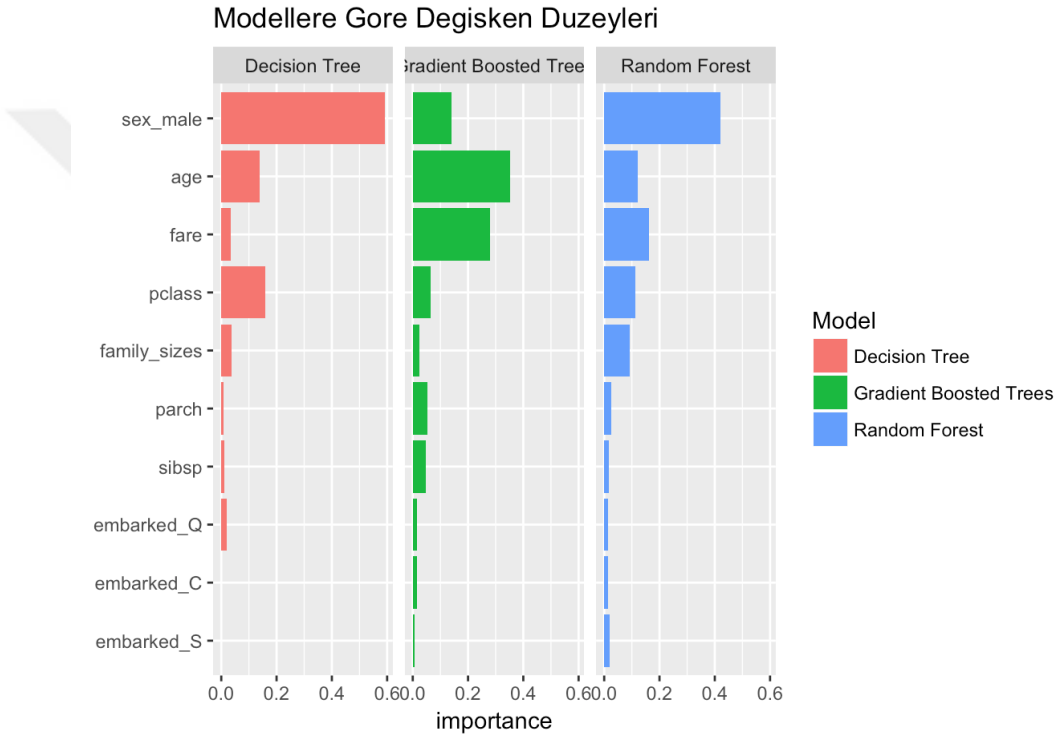
```
for(i in c("Decision Tree", "Random Forest", "Gradient Boosted Trees")){
  deg_onem <- ml_tree_feature_importance(sc, ml_modeller[[i]]) %>%
    mutate(Model = i) %>%
    mutate(importance = as.numeric(levels(importance))[importance]) %>%
    mutate(feature = as.character(feature)) %>%
    rbind(deg_onem, .)
}
```

Şekil 4.46 Modellere göre önem düzeylerinin hesaplanması

Sonuçların görselleştirilmesi:

```
deg_onem %>%
  ggplot(aes(reorder(feature, importance), importance, fill = Model)) +
  facet_wrap(~Model) +
  geom_bar(stat = "identity") +
  coord_flip() +
  xlab("") +
  ggtitle("Modellere Gore Degisken Duzeyleri")
```

Şekil 4.47 Modellere göre önem düzeylerinin görselleştirilmesi



Şekil 4.48 Modellere göre değişkenlerin önem düzeyleri

4.5.4 Model Tahmin Başarılarının Karşılaştırılması

Eğitim seti ile eğitilen modellerin test seti ile sınanması ile oluşacak olan başarı durumları karşılaştırılacaktır. Bunun için aşağıda ifade edilecek olan bazı fonksiyonlar yazılacak ve karşılaştırmalar görselleştirilecektir.

Test için fonksiyon yazılması:

```
skor_test_data <- function(model, data=test_tbl){
  pred <- sdf_predict(model, data)
  select(pred, survived, prediction)
}
```

Şekil 4.49 Test fonksiyonu

Tüm modellerin test seti ile sınanması:

```
ml_skor <- lapply(ml_modeller, skor_test_data)
```

Şekil 4.50 Modellerin skorlanması

AUC ve Doğruluk:

Spark ile ROC eğrileri oluşturulamadığı için model performans karşılaştırmaları için AUC ve doğruluk değerleri kullanılacaktır. AUC ve Doğruluk değerlerinin hesaplanması için SPARK ML'de bulunan ml_classification_eval ve ml_binary_classification_eval fonksiyonları kullanılacaktır.

Doğruluk (accuracy) hesaplaması için fonksiyon yazılması:

```
dogruluk_hesap <- function(data, kesim_noktasi = 0.5){
  data %>%
    mutate(prediction = if_else(prediction > kesim_noktasi, 1.0, 0.0)) %>%
    ml_classification_eval("prediction", "survived", "accuracy")
}
```

Şekil 4.51 Doğruluk hesaplaması için fonksiyon yazılması

AUC ve doğruluk değerlerinin hesaplanması:

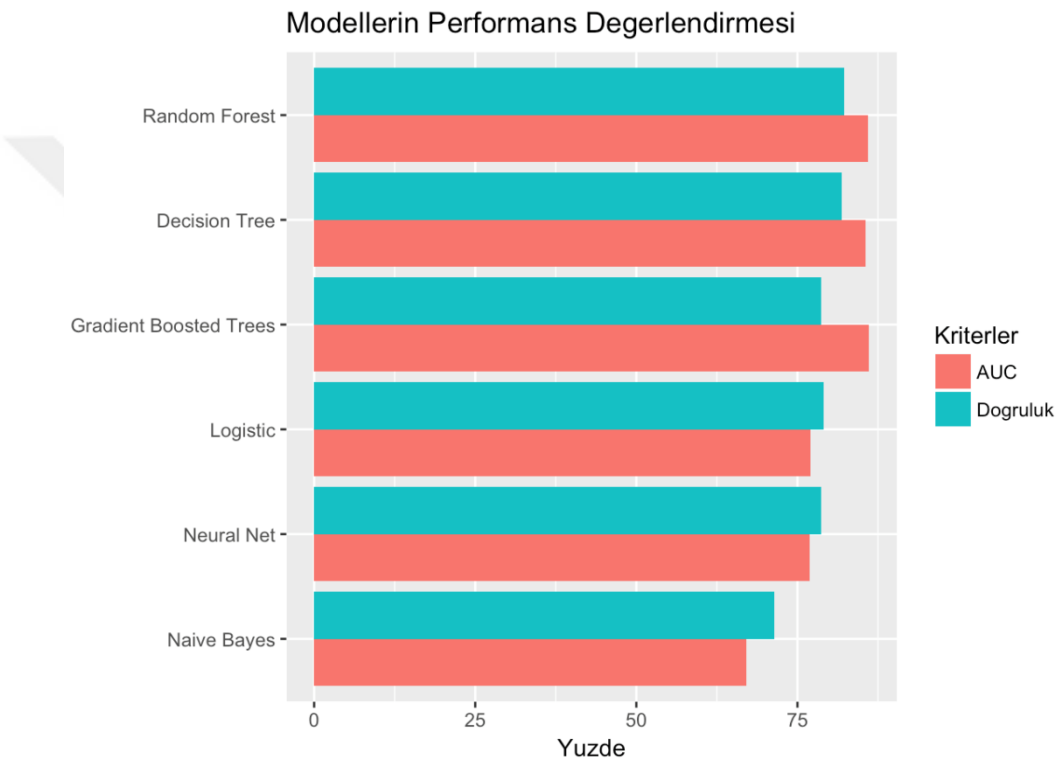
```
kriterler <- data.frame(
  model = names(ml_skor),
  AUC = 100 * sapply(ml_skor, ml_binary_classification_eval, "survived", "prediction"),
  Dogruluk = 100 * sapply(ml_skor, dogruluk_hesap),
  row.names = NULL, stringsAsFactors = FALSE)
```

Şekil 4.52 AUC ve Doğruluk değerlerinin hesaplanması

Sonuçların görselleştirilmesi:

```
gather(kriterler, Kriterler, value, AUC, Dogruluk) %>%
  ggplot(aes(reorder(model, value), value, fill = Kriterler)) +
  geom_bar(stat = "identity", position = "dodge") +
  coord_flip() +
  xlab("") +
  ylab("Yuzde") +
  ggtitle("Modellerin Performans Degerlendirmesi")
```

Şekil 4.53 Sonuçların görselleştirilmesi



Şekil 4.54 Model performans karşılaştırması

BÖLÜM 5

SONUÇ VE ÖNERİLER

Uygulama kapsamında R ve büyük veri araçlarının birlikte kullanılması ile uçtan uca büyük veride makine öğrenmesi uygulaması yapılmıştır. R ve büyük veri araçlarının birlikte kullanımı ile büyük veri yığınları içerisinde interaktif şekilde veri analitiği yapılabilmektedir. R dili kullanan İstatistikçilerin, Veri Analistlerinin ve Veri Bilimcilerin sparklyr gibi kütüphaneler aracılığı ile büyük veride R programlama dilinin esnekliği ile çalışabileceği gösterilmiştir. Apache Spark'ın sunduğu Spark SQL ve MLlib kütüphanesinin sparklyr ile efektif olarak kullanılabildiği görülmüştür. Spark MLlib'in sahip olduğu tüm fonksiyonlar sparklyr üzerinden de kullanılabilmektedir. İstendiği taktirde küme üzerinde yerel R fonksiyonlarınında kullanılabiliyor olması kullanıcıya çok büyük esneklikler sağlamaktadır. Sonraki çalışmalarda python ile büyük veri araçları arasındaki entegrasyonlar incelenebilir, R ve Python arasındaki farklılıklar ifade edilebilir, makine öğrenmesi araçları sunan h20 kütüphanesi, Scala, python ve R'ın makine öğrenmesi modelleri konusundaki esneklikleri, kısıtları ve sundukları özellikler karşılaştırılabilir.

KAYNAKLAR

[1] Yengi, Y., (2016). Büyük Veride Duygu Analizine Dayalı Öneri Sistemleri, Yüksek Lisans Tezi, Kocaeli Üniversitesi Fen Bilimleri Enstitüsü, Kocaeli.

[2] Salur, M.U., (2016). Büyük Veri Araçlarından Hadoop Kullanarak Veri Madenciliği, Yüksek Lisans Tezi, Pamukkale Üniversitesi Fen Bilimleri Enstitüsü, Denizli.

[3] Özdeş, M., (2017). Büyük Veri Araçları Kullanarak Duygu Analizi Gerçekleştirimi, Yüksek Lisans Tezi, Pamukkale Üniversitesi Fen Bilimleri Enstitüsü, Denizli.

[4] Akgün, B., (2016). Apache Spark Tabanlı Destek Vektör Makineleri ile Akan Büyük Veri Sınıflandırma, Yüksek Lisans Tezi, İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü, İstanbul.

[5] Çetinkaya, S., (2016). Hadoop/MapReduce Teknolojisi Kullanılarak Hızlı Tüketim Sektöründe Büyük Veri Analizi, Yüksek Lisans Tezi, İstanbul Üniversitesi Fen Bilimleri Enstitüsü, İstanbul.

[6] Hallaç, İ.R., (2014). Büyük Veri Analizinde Dağıtık Makine Öğrenmesi Algoritmalarının Kullanılması, Yüksek Lisans Tezi, Fırat Üniversitesi Fen Bilimleri Enstitüsü, Elazığ.

[7] Tarla.io, Akıllı Tarım Uygulamaları Girişimi, Tarla.io, 3 Şubat 2018.

[8] Lorilewis Twitter Profili, Bir Dakikada İnternette Neler Oluyor?, https://twitter.com/lorilewis, 3 Şubat 2018

[9] De, M., Girmaldi, M., Greco, A. (2015). "What is big data? A consensual definition and a review of key research topics", AIP Conference Proceedings, 5–8 September 2014, Madrid, 97-104.

[10] SAS, What is big data, http://www.sas.com/en_us/insights/big-data/what-is-big-data.html, 20 Kasım 2015.

[11] IBM, Büyük Veri Analitiği Türleri, http://www-01.ibm.com/common/ssi/cgi-bin/ssialias?infotype=SA%26subtype=WH%26htmlfid=TIW14162USEN, 03 Şubat 2018.

[12] Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., Gruber, R. E. (2008). "Bigtable", ACM Transactions on Computer Systems, 26(2): 1-26.

[13] Ghemawat, S., Gobioff, H., Leung, S. (2003). “The Google File System”, SOSP '03 Proceedings of the nineteenth ACM symposium on Operating systems principles, 19–22 October 2003, NY, 29-43.

[14] Dean, J., Ghemawat, S., (2008). “Mapreduce: simplified data processing on large clusters”, ACM SIGMOD Record, 51: 107–113.

[15] Hadoop Offical Web Site, Apache Hadoop, http://hadoop.apache.org, 08 Şubat 2018.

[16] Gonzolez, J., (2012). Parallel and Distributed Systems for Probabilistic Reasoning, Doctoral thesis, Machine Learning Department School of Computer Science Carnegie Mellon University, Pittsburgh.

[17] Devops.com, Hadoop Common Figure, https://devops.com/bigdata-understanding-hadoop-ecosystem, 08 Şubat 2018.

[18] Stackoverflow.com, MapReduce Word Count Proccess Figure, https://stackoverflow.com/questions/20317152/how-shuffling-is-done-in-mapreduce, 08 Şubat 2018.

[19] Edureka.com, Hadoop Ecosystem Figure, https://www.edureka.co/blog/hadoop-ecosystem, 08 Şubat 2018.

[20] Apache Ambari Official Web Site, Apache Ambari, https://ambari.apache.org/, 08 Şubat 2018.

[21] Apache Cassandra Official Web Site, Apache Cassandra, http://cassandra.apache.org/, 08 Şubat 2018.

[22] Apache Hive Official Web Site, Apache Hive, http://hive.apache.org/, 08 Şubat 2018.

[23] Apache HBase Official Web Site, Apache HBase, http://hbase.apache.org/, 08 Şubat 2018.

[24] Apache Sqoop Official Web Site, Apache Sqoop, http://sqoop.apache.org/, 08 Şubat 2018.

[25] Apache Oozie Official Web Site, Apache Oozie, http://oozie.apache.org/, 08 Şubat 2018.

[26] Apache Impala Official Web Site, Apache Impala, http://impala.apache.org/, 08 Şubat 2018.

[27] Apache Mahout Official Web Site, Apache Mahout, http://mahout.apache.org/, 08 Şubat 2018.

[28] Azkaban Official Web Site, Azkaban, https://azkaban.github.io/, 08 Şubat 2018.

[29] HUE Official Web Site, HUE, http://gethue.com/,08 Şubat 2018.

[30] Apache Kafka Official Web Site, Apache Kafka, http://kafka.apache.org/, 08 Şubat 2018.

[31] itproportal.com, 5 Major Advantages of Hadoop, https://www.itproportal.com/2013/12/20/big-data-5-major-advantages-of-hadoop/, 08 Şubat 2018.

[32] SAS, What is it and Why does it matter, http://www.sas.com/en_us/insights/big-data/hadoop.html, 08 Şubat 2018.

[33] Apache Spark Offical Web Site, Apache Spark, http://spark.apache.org/, 08 Şubat 2018.

[34] Zaharia, M., (2013). An Architecture for Fast and General Data Processing on Large Clusters, Doctoral thesis, Graduate Division in Computer Science University of California, Berkeley.

[35] cloudxloab.com, Apache Spark Components, https://cloudxlab.com/blog/spark-interview-questions/, 08 Şubat 2018.

[36] Apache Spark Offical Web Site, Spark SQL, https://spark.apache.org/sql/, 08 Şubat 2018.

[37] Apache Spark Offical Web Site, Spark MLlib, https://spark.apache.org/sql, 08 Şubat 2018.

[38] Apache Spark Offical Web Site, Spark Streaming, https://spark.apache.org/streaming, 08 Şubat 2018.

[39] Apache Spark Offical Web Site, GraphX, https://spark.apache.org/graphx, 08 Şubat 2018.

[40] Alpaydın, E., (2010). Introduction to Machine Learning, Massachusetts Institute of Technology, United States of America

[41] Shalev-Shwartz, S., and Ben-David, S., (2014). Understanding Machine Learning: From Theory to Algorithms, Cambridge University Press, Cam

[42] Mohri, M., Rostamizadeh, A., & Talwalkar, A., (2012). Foundations of machine learning, The MIT Press Cambridge, London.

[43] Niculescu-Mizil, A., & Caruana, R., (2005). "Predicting good probabilities with supervised learning", Proceedings of the 22nd international conference on Machine learning, 07–11 August 2005, Bonn, 625-632.

[44] Alpar, R., (2013). Uygulamalı Çok Değişkenli İstatistiksel Yöntemler, 4. Baskı, Detay Yayıncılık, Ankara.

[45] Güriş, S., Çağlayan, E., (2013). Ekonometri: Temel kavramlar, 4.Baskı, Der Yayınevi, İstanbul.

[46] Mertler, C. A., & Vannatta, R. A., (2005). Advanced and multivariate statistical methods: practical application and interpretation, CA: Pyrczak, Glendale.

[47] www.r-bloggers.com, Decision Tree Figure, https://www.r-bloggers.com/regression-tree-using-ginis-index/, 08 Şubat 2018.

[48] Linoff, G. S., & Berry, M. J., (2011). Data mining techniques: for marketing, sales, and customer relationship management, IN: Wiley Publishing, Indianapolis.

[49] Murphy, K. P. (2012). Machine Learning A Probabilistic Perspective, The MIT Press, Cambridge, Massachusetts, London.

[50] Breiman, L., (2001). "Machine Learning", Kluwer Academic Publishers Hingham, 45(1), 5-32

[51] Singh, K., & Xie, M., (2010). “Bootstrap Method”, International Encyclopedia of Education, 46-51.

[52] Breiman, L. (1996). “Bagging predictors”, Machine Learning, 24(2), 123-140.

[53] Schapire, R. E., (1990). “The strength of weak learnability”, Machine Learning, 5(2), 197-227.

[54] Freund, Y., Schapire, R. E., Abe, N., (1999). “A Short Introduction to Boosting”, Japanese Society For Artificial Intelligence, 37(3), 277-296.

[55] www.towardsdatascience.com, Boosting Figure, https://towardsdatascience.com/boosting-in-machine-learning-and-the-implementation-of-xgboost-in-python-fb5365e9f2a0, 08 Şubat 2018.

[56] Ho, T. K., (1998). “The Random Subspace Method for Constructing Decision Forests”, IEEE Transaction on Pattern Analysis and Machine Intelligence, 20(8), 832-844.

[57] Silahtaroğlu, G., (2013). Kavram ve Algoritmalarıyla Veri Madenciliği Kavram ve Algoritmaları, 2.Basım, Papatya Yayıncılık Eğitim, İstanbul.

[58] Stephen, M. S., (1982). “Thomas Bayes' Bayesian Inference”, Journal of the Royal Statistical Society, Series A, 250–258.

[59] Duda, R. O., Hart, P. E., Stark, D. G., (2001). Pattern Classification, 2nd ed., John Wiley & Sons, New York.

[60] Lewis, D. D., (1998), “The independence assumption in information retrieval”, Springer Berlin Heidelberg, 4-15.

[61] Hastie, T., Tibshirani, R., and Friedman, J. (2001). The elements of statistical learning: data mining, inference, and prediction, Second Edition, Springer, New York.

[62] Domingos P., Pazzani M., (1997). “On the Optimality of the Simple Bayesian Classifier under Zero-One Loss”, Machine Learning, 29, 103-130.

[63] Öztemel, E., (2012). Yapay Sinir Ağları, 2. Basım, Papatya Yayıncılık Eğitim, İstanbul.

[64] Fausett, L., (1994). Fundamentals of neural networks: architectures, algorithms, and applications, Prentice-Hall, NJ, US.

[65] wikimedia.org, Perceptron Figure, https://commons.wikimedia.org/wiki/File:Rosenblattperceptron.png, 08 Şubat 2018.

[66] mdpi.com, Multilayer Perceptron Figure, http://www.mdpi.com/2078-2489/3/4/756/htm, 08 Şubat 2018.

[67] Altaş,D., (2006). Türk Bankacılık Sektörünün İstatistik Tekniklerle Analizi (2000 Krizi Öncesi ve Sonrası), 1. Baskı, Derin Yayınları, İstanbul.

[68] towardsdatacience.com, Train Test Split Figure, https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6, 08 Şubat 2018.

[69] Witten, I. H., Frank, E., & Hall, M. A., (2011). Data Mining: Practical Machine Learning Tools and Techniques, Second Edition, Elsevier, San Francisco.

[70] medcalc.org, ROC Curve Figure, https://www.medcalc.org/manual/roc-curves.php, 08 Şubat 2018.

[71] A. Rajaraman and J. Ullman, (2011). Mining of Massive Data Sets, Cambridge Univ. Press, Cambridge.

[72] Apache Spark Official Page, SparkR , https://spark.apache.org/docs/latest/sparkr.html, 08 Şubat 2018.

[73] Slideshare.net, SparkR Architecture , https://www.slideshare.net/databricks/parallelizing-existing-r-packages-with-sparkr, 08 Şubat 2018.

[74] github.com, DistributedR, https://github.com/vertica/DistributedR, 08 Şubat 2018.

[75] rstudio.com, sparklyr, http://spark.rstudio.com/, 08 Şubat 2018.

[76] rstudio.com, Yarn Cluster EMR, http://spark.rstudio.com/examples/yarn-cluster-emr/, 08 Şubat 2018.

[77] amazon.com, EMR, https://aws.amazon.com/emr/, 08 Şubat 2018.

[78] tidyverse.org, dplyr, http://dplyr.tidyverse.org/, 08 Şubat 2018.

EK- A

FLIGHTS VERİSİ İÇİN HIVE TABLOSU OLUŞTURMA KODU

```
CREATE EXTERNAL TABLE IF NOT EXISTS flights
(
  year int, month int, dayofmonth int, dayofweek int, deptime int,
  crsdeptime int, arrtime int, crsarrtime int, uniquecarrier string,
  flightnum int, tailnum string, actualelapsedtime int,
  crselapsedtime int, airtime string, arrdelay int, depdelay int,
  origin string, dest string, distance int, taxiin string,
  taxiout string, cancelled int, cancellationcode string,
  diverted int, carrierdelay string, weatherdelay string,
  nasdelay string, securitydelay string, lateaircraftdelay string
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
TBLPROPERTIES("skip.header.line.count"="1");


#Verinin tabloya yüklenmesi:
LOAD DATA INPATH '/user/rstudio-user/flights' INTO TABLE flights;
```

EK – B

AIRLINES HIVE TABLOSU OLUŞTURMA KODU

```
CREATE EXTERNAL TABLE IF NOT EXISTS airlines
(
  Code string,
  Description string
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
WITH SERDEPROPERTIES
(
  "separatorChar" = '\,',
  "quoteChar"     = '\"'
)
STORED AS TEXTFILE
tblproperties("skip.header.line.count"="1");

# Verinin tabloya yüklenmesi
LOAD DATA INPATH '/user/rstudio-user/airlines' INTO TABLE airlines;
```

EK – C

AIRPORTS HIVE TABLOSU OLUŞTURMA KODU

```
CREATE EXTERNAL TABLE IF NOT EXISTS airports
(
  id string,
  name string,
  city string,
  country string,
  faa string,
  icao string,
  lat double,
  lon double,
  alt int,
  tz_offset double,
  dst string,
  tz_name string
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
WITH SERDEPROPERTIES
(
  "separatorChar" = '\,',
  "quoteChar"     = '\"'
)
STORED AS TEXTFILE;

# Verinin tabloya yüklenmesi
LOAD DATA INPATH '/user/rstudio-user/airports' INTO TABLE airports;
```

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı : Mustafa Vahit KESKİN

Doğum Tarihi ve Yeri : 18.02.1989, Eskişehir

Yabancı Dili : İngilizce

E-posta : m.v.k@msn.com

ÖĞRENİM DURUMU

Derece	Alan	Okul/Üniversite	Mezuniyet Yılı
Lisans	İstatistik	Marmara Üniversitesi	2015
Lise	Bilişim Teknolojileri	Sultanahmet Anadolu Meslek	2010

İŞ TECRÜBESİ

Yıl	Firma/Kurum	Görevi
2016	sahibinden.com	Uzman Veri Bilimci

YAYINLARI

Bildiri

1. Yıldız D., Keskin M.V., Çakır Yıldız N. ve Evren A., (2016). "İstatistik Bölümü Öğrencilerinin İstatistiksel Düşünme Becerileri Üzerine Bir Değerlendirme", Xth International Statistics Days Conference, 07-09 October 2016, Giresun.