T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BIG DATA YÖNTEMLERİ VE LOJİSTİK REGRESYON ANALİZİ İLE İNTERNET ALTYAPI KALİTE DEĞERLENDİRMESİ

YÜKSEK LİSANS TEZİ

Talha ALTUN

Enstitü Anabilim Dalı	**:**	**ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ**
Enstitü Bilim Dalı	**:**	**ELEKTRİK MÜHENDİSLİĞİ**
Tez Danışmanı	**:**	**Doç. Dr. Mehmet Recep BOZKURT**

Mart 2020

T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BIG DATA YÖNTEMLERİ VE LOJİSTİK REGRESYON ANALİZİ İLE İNTERNET ALTYAPI KALİTE DEĞERLENDİRMESİ

YÜKSEK LİSANS TEZİ

Talha ALTUN

Enstitü Anabilim Dalı : ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ

Bu tez 12.03.2020 tarihinde aşağıdaki jüri tarafından oybirliği / oyçokluğu ile kabul edilmiştir.

Doç. Dr. Mehmet R. BOZKURT Jüri Başkanı	Dr. Öğr. Üyesi Burhan BARAKLI Üye	Doç. Dr. Akif AKGÜL Üye

BEYAN

Yazılmış olan tez içerisinde var olan verilerin akademik kurallara uyularak tarafımca elde edilmiş olduğunu, kullanılmış görsellerin ve yazılmış her bilginin ve sonuçlarının akademik ve etik kurallar çerçevesine uyulmuş bir biçimde sunulmuş olduğunu, kullanılmış tüm verilerin hiçbir tahrifata uğratılmadığını, başka şahısların ortaya koymuş olduğu eserlerden faydalanılması söz konusu olduğunda bilimsel ve akademik kurallara uygun bir biçimde atıfta bulunulduğunu, tezde var olan veriler ve görsellerin yüksek lisansıma devam ettiğim Sakarya Üniversitesi veya herhangi bir üniversitede hiçbir çalışmada kullanılmadığını beyan ederim.

Talha ALTUN
12.03.2020

TEŞEKKÜR

Sakarya Üniversitesi Elektrik Elektronik Mühendisliği yüksek lisans programı süresince kıymetli tecrübe ve bilgilerinden faydalanmış olduğum, herhangi bir sorunumda tecrübe ve bilgisini almaktan geri durmadığım, üzerinde özenle çalışmış olduğum bu değerli araştırmanın planlanma sürecinden yazılı hale gelinceye kadar geçen tüm aşamalarda desteklerini gösteren, araştırma iştahımı kabartan ve aynı titizlikte şahsıma yön veren kıymetli danışman hocam Doç. Dr. Mehmet Recep BOZKURT' a teşekkürlerimi bir borç bilirim.

AR-GE imkanları ve sağlanılan olanaklar bakımından gösterdikleri anlayış, sabır ve yardımlarını hiçbir şekilde geri çekmeyen SKYZ TECH (eski adıyla SEKOM YAZILIM) ailesine teşekkür ederim.

İÇİNDEKİLER

SİMGELER VE KISALTMALAR LİSTESİ

HDFS	: Hadoop Distributed File System
VT	: Veri Tabanı
İAS	: İnternet Altyapı Sağlayıcıları
ADSL	: Asymmetric Digital Subscriber Line
GSM	: Global System for Mobile Communications
CPU	: Central Process Unit
RAM	: Random Access Memory
MB	: MegaByte
SQL	: Structured Query Language
CSV	: Comma-Separated Variables
FTP	: File Transfer Protocol
DSL	: Digital Subscriber Line
IPTV	: Internet Protocol Television
VDSL	: Very High Speed Digital Subscriber Line
ODBC	: Open DataBase Connectivity
UID	: Unique Identify
ARGE	: Araştırma ve Geliştirme
XML	: Extensible Markup Language

ŞEKİLLER LİSTESİ

ÖZET

Anahtar kelimeler: big data, hdfs, lojistik regresyon analizi, pig script

Bu çalışmada, internet altyapı sağlayıcılarının (İAS) yatırım yapılacak potansiyel lokasyonları ve arıza oranlarını ay ve/veya yıl düzeyinde belirlemesi amaçlanmıştır. Arıza oranları tekil ve toplam arıza olarak Big Data yöntemiyle belirlenmiş olup Lojistik Regresyon Analizi kullanılarak var olan arıza oranlarından potansiyel "stabile"/"unstabile" lokasyonlar belirlenmiştir.

Farklı veritabanlarından (VT) alınan anlamsız gözüken veriler, Big Data yöntemleri kullanılarak hadoop distributed file system (HDFS) üzerinde depolanmıştır. Depolanmış olan anlamsız verilerle pig script kullanılarak veri analizi yapılmıştır. Pig script, veri analizinde HDFS üzerindeki veri kümelerini ortak kolon isimlerine göre birleştirerek tek bir dosya üzerine kaydedebilmek için kullanılmış, komplike işlemlerin gerçekleştirilmesine olanak sağlamıştır.

Komplike işlemler, anlamsız görünen verilerin Big Data mantığı ile değerlendirilip anlamlandırılması ile tahminleme kısmına hazırlanması amaçlanmıştır.

Tahminlemenin ikili sonuç şeklinde oluşması hedef alınmış, bu sebeple lojistik regresyon analizi uygulanmaya el verişli son rapor elde edilmiştir.

Elde edilen ve HDFS üzerinde tutulan sonuç dosyası python programlama dili ve lojistik regresyon uygulaması ile potansiyel "stabile"/"unstabile" lokasyonların tespiti için kullanılmıştır. Lojistik regresyon ile daha önceden belirlenmiş arıza sayıları (tekil ve toplam arıza) ve var olan diğer kolonlar bir bütün halinde değerlendirilmiş olup, yeni verilerde potansiyel "stabile"/"unstabile" lokasyonları tespit edilmiştir.

Sonuç olarak yatırım yapılacak lokasyonların önceden tespit edilmesi; elde edilen sonuç verisi ve lokasyonların potansiyel stabilitesi ile çok daha anlaşılır bir şekilde gösterilmiştir. Bu çalışma, Big Data ve Lojistik Regresyon Analizi ortak kullanımı ile anlamlı veriler üzerinden yorum yapabilmeyi kolaylaştırmayı hedeflemiştir.

QUALITY ASSESMENT OF INTERNET INFRASTRUCTURE USING BIG DATA METHODS AND LOGISTIC REGRESSION ANALYSIS

SUMMARY

Keywords: big data, hdfs, logistic regression analysis, confusion matrix, pig script

In this study, it is aimed that Internet Infrastructure Providers (IIP) determine the potential locations to be invested and the failure rates are determined at the month and/or year level. The failure rates are determined by the Big Data method as single and total failures and the locations where potential failure will occur from the existing failure rates are determined by using Logistic Regression Analysis.

The meaningless data obtained from different databases were stored on Hadoop Distributed File System (HDFS) by using Big Data methods. Data analysis was performed by using pig script with stored meaningless data. Pig script has been used in data analysis to save the data sets on HDFS into a single file by combining them according to common column names, allowing complicated operations to be performed.

Complicated cleanup is goal-oriented for the forecasting department by evaluating and interpreting it with Big Data logic in a meaningless view. It is aimed to form a binary result of the estimation, so the last report, which can be applied Logistic Regression Analysis, was obtained.

The result file obtained on HDFS was used to determine potential fault locations with python programming language and Logistic Regression Analysis. With the Logistic Regression Analysis, the pre-determined number of failures (single and total failures) and other existing columns are evaluated as a whole and potential fault locations are determined in the new data.

As a result, predetermining the locations to be invested; generated result data and potential fault locations will be obtained more clearly. This study aims to facilitate the interpretation of meaningful data through the joint use of Big Data and Logistic Regression Analysis.

BÖLÜM 1. GİRİŞ

İnternet, birçok kişi tarafından kimi zaman çağımızın kimi zaman da tüm zamanların en büyük teknoloji atılımı olarak nitelendirilmiştir [1]. Öyle ki günümüzde internet getirdiği kolaylıklardan ve bilgiye çabuk ulaşmadaki rahatlığından ötürü elektrik, su veya doğalgaz gibi büyük önem taşımaktadır. İnternet kesintisi, bağlantının kopması ve hatta hızının düşmesi sadece bireysel değil bağlı olan kaynakların da işlerini aksatmasına ya da ertelemesine sebep olmaktadır.

İnternetin iletim hızı birçok sebebe bağlı olmak ile birlikte, iletim hızının maksimum kapasitesinin bağlı olduğu farklı türleri vardır: Adsl, Vdsl ve mobil.

Adsl ve Vdsl internet tipleri evlerde kullanılmakta, mobil internet ise gsm operatörlerinden satın alınmış hatlarda kullanılmaktadır.

İnternet altyapısı denildiği zaman İnternet Servis Sağlayıcıların ilk noktadan evlerimize kadar ulaştırdığı tüm hat boyu akla gelmektedir. Bir kullanıcının Adsl mi yoksa Vdsl mi olduğunu belirleyen etkenlerden en önemlisi kullanıcının lokasyonunda bulunan altyapısıdır. Eğer kullanıcının altyapısı eski tip bir mimariye sahip ise alabileceği hızın maksimum değeri bellidir.

Hızın maksimum değerinin belirli olmasına rağmen optimum düzeyde tutulamamasının da çeşitli sebepleri olabilmektedir. Tüm etmenleri tek çatı altında toplayıp çözmek için doğru yatırımlar yapmak gerekmektedir. Doğru yatırım yapabilmek için doğru bilgilere zamanında ulaşabilmek gerekmektedir.

Tüm internet servis sağlayıcılarının kendi veri tabanı bulunmakta ve bu veri tabanlarında hız, şikâyet, arıza kaydı ve daha birçok farklı kolon isimleriyle doldurulmaktadır. Uçtan uca tutulan tüm veriler ayrıştırılabilmekte ve anlamsız bilgilerden anlamlı bilgiler elde edilebilmektedir. "İnternet Servisi" sağlayan firmadaki son kullanıcı monitörize dahi edilebilmektedir. Bölge'den İl'e, Santral'den Binalara hatta dairelere kadar uzanan bu hatta birbirine bağlı olan lokasyonlardaki arızalar kümülatif olarak artmaktadır. Belirli bir lokasyonda var olan arıza yoğunluğu daha önceden yapılmış hataların habercisi olmaktadır.

Bu çalışmada Big Data yöntemleri kullanılarak anlamsız görünen verilerden anlamlı hale getirilmiş veriler türetilmiştir. HDFS üzerinde farklı lokasyonlardan çekilmiş olan veriler tek bir noktada toplanmış olup, pig script ile veri işleme gerçekleştirilmiştir. Son olarak oluşturulan anlamlı veriler üzerinde Lojistik Regresyon Analizi yapılmış ve lokasyonların potansiyel stabilite durumları elde edilmiştir.

BÖLÜM 2. KAYNAK ARAŞTIRMASI

2.1. Big Data ve Kullanım Alanları

Big Data terimi, son birkaç yılda hemen hemen bütün alanlarda çokça kullanılmaya ve uygulanmaya başlanmıştır. Hatta veri alanına uzak kişi ve kurumlar dahi bu terimden söz etmekte ve faydalanılması gereken bir oluşum olduğuna dikkat çekmektedir. Big Data, medya sektöründen tutun da iletişim alanlarından kara yolları planlanmasına kadar her türlü verinin dijital ortamlarda kaydının tutulması ile ortaya çıkmıştır ve teknoloji ile birlikte gelişimine devam etmektedir. Büyük çapta verilerin analizini daha önce sadece bilişim sektörü ile uğraşan kişiler yapabilirdi. Veri oluşumunda ise yediden yetmişe herkesin dijital ortama veri kaydedebilmesi beraberinde hızlı bir veri artışını getirmektedir. Kayıt altına alınan verilerin yoğunluğu ise oldukça fazladır; herhangi bir kişi bile bir günden daha kısa sürede megabaytlarca veriyi rahatlıkla dijital ortama kaydetmekte ve dahası binlerce makine ve kişinin kullanımına imkân vermektedir. Bu kadar çok veri oluşmasından dolayı büyük veri kavramının ortaya çıkması kaçınılmaz olmuştur. Ancak büyük veri sadece çok veri değildir. Çokluk veya miktar, büyük verinin sadece bir özelliğidir [2].

Big Data kavramının oluşması için Thomas H. Davenport 5V kavramını aşağıdaki gibi detaylandırmaktadır:

Miktar (Volume): Verinin büyüklüğü günümüzde terabyte seviyelerinde ve hatta petabyte seviyelerine ulaşmış durumda olduğundan, veriyi depolamak ve analiz sürecini yönetmek için bilinen yaklaşımlar yetersiz kalmak durumundadır.

Hız (Velocity): Çok hızlı çoğalan veriler, o verilere ihtiyaç duyan işlem sayısının ve buna bağlı çeşitliliğinin de benzer hızlarda yükselmesi sonucunu doğurmaktadır.

Çeşitlilik (Variety): Üretilmiş olan verilerin neredeyse yüzde 80'i yapısal durumda değildir ve yeni gelen her teknoloji, değişik formatlardaki verileri beraberinde getirmektedir. Tabletlerden, telefonlardan kısacası bütünleşik devrelerden gelebilecek olan çok çeşitli veri tipleri ile uğraşılması ve bu veri tiplerinin birbirlerine dönüşmeleri gerekmektedir.

Doğrulama (Verification): Var olan bilgi yoğunluğu içerisinde verinin bir noktadan başka bir noktaya akışı esnasında güvenli olarak süregelmesi de bir başka bileşendir. Akış esnasında verinin, doğru noktadan ve doğru güvenlik bütünlüğü çerçevesinde izlenmesi ve doğru şahıslarca görüntülenebilir olması veya tam tersi şekilde saklı tutulması gerekmektedir.

Değer (Value): Büyük verinin, üretim ve işleme aşamalarının ardından kurum için bir artı değer oluşturuyor olması, karar verme süreçlerine anlık ve doğrudan etki etmesi, doğru kararı vermede hemen el altında olması gerekmektedir [3].

Big Data yani büyük verinin kullanılmasının iki büyük nedeni var. Birincisi Maliyet tasarrufu, ikincisi ise zaman tasarrufudur [4].

Big Data' nın sektörlere olan katkıları ise aşağıdaki gibi özetlenebilir.

İşletme Sektörü: Müşteriyi kişiselleştirmek, müşteri kaybının sebeplerini belirlemek ve incelemek, dağıtım ve lojistik optimizasyonu yapabilmek.

Teknoloji Sektörü: İşlem sürelerini azaltabilmek, gerçek zamanlı analizler yapabilmek, kriz dönemlerinde sorunlara hızlı cevap üretebilmek, riskleri en aza indirebilmek için otomatik sistemler ile karar verebilmek.

Sağlık Sektörü: Hastalık tespitlerini hızlandırmak ve doğru teşhisi koyabilmek, hasta seyrinin takibi ve sağlığını güçlendirmek için kişisel DNA analizi yapabilmek.

Kamu Sektörü: Verilere erişilebilirlik oluşturarak şeffaflığı sağlayabilmek, doğru ürün ve hizmetler için gerekli yatırımların uyarlanmasını öngörebilmek.

Perakende Satış Sektörü: Mağaza davranış analizi yapabilmek, çeşitlilik ve fiyat optimizasyonu ile katma değer üretmek, ürün yerleştirme tasarımını yapabilmek, satış ve deneyimi geliştirebilmek, çalışan gelirlerini optimize edebilmek.

Kişisel Konum Verileri: Doğru yönlendirme yapabilmek, coğrafi hedefli reklamcılığa imkân vermek, acil müdahale gerektiren durumlarda zamanı iyi kullanabilmek.

Akıllı Şehirler: Doğal kaynakları iyi yönetebilmek, sürdürülebilir ekonomik gelişmeyi ve yüksek kaliteli yaşamı sağlayabilmek.

2.2. Hadoop

Geleneksel Big Data yöntemlerinin birçok problemi ortaya çıkmıştır. Kısaca bahsetmek gerekirse bunlar;

- Yapılandırılmamış veriye platformun uygun olmaması, ölçeklenememe problemi, lisans ücretlerinin yüksek olması,

- Cpu, ram, disk, network gibi bileşenlerin limitlerinin yetersizliği

- Yazılımsal karmaşıklıklar

Geleneksel yöntemlere yeni bir yaklaşım olarak Hadoop ortaya çıkmıştır.

Hadoop, basit programlama modellerini kullanarak büyük bir veriyi bilgisayar kümeleri arasında dağıtılmış bir ortamda saklamayı ve işlemeyi sağlayan açık kaynaklı bir platformdur. Lokal hesaplama ve depolama sağlayan tekil sunuculardan binlerce makineye kadar ölçeklendirmek için tasarlanmıştır [5].

Veri işleme amaçlı dağıtık uygulamalar yazılmasını sağlayan bir platform olmasının yanında açık kaynaklı bir Apache projesidir.

Getirdiği önemli yeni yaklaşımlar; ihtiyaç duyulduğunda sistemin genişleyebilir olması (ölçeklenebilirlik), veri saklanırken bozulmaması (tutarlılık), verilerin yedekli bir yapıda tutulması (yedeklilik), sistemin her durumda çalışabilir (erişilebilirlik) olmasıdır.

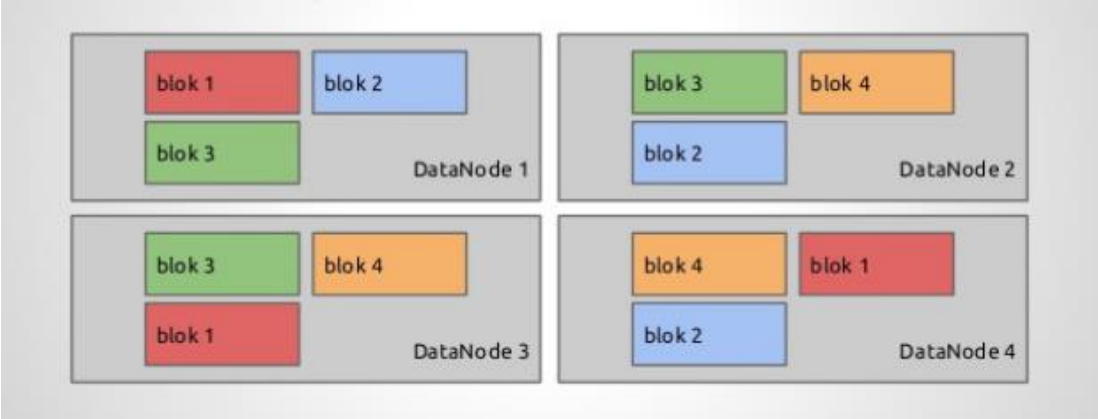
2.2.1. HDFS ve MapReduce

Hadoop'un gücü, işlenen dosyaların her zaman ilgili düğümün yerel diskinden okunması ile ağ trafiğini meşgul etmemesinden ve birden fazla işi aynı anda gerçekleştirerek doğrusal olarak ölçeklenmesinden gelmektedir [6].

Hadoop iki temel bileşenden oluşmaktadır. Bunlar Hadoop Distributed File System (HDFS) ve MapReduce şeklindedir.

HDFS çok büyük miktarda veri tutar ve daha kolay erişim sağlar. Bu kadar büyük veriyi muhafaza etmek için dosyalar birden fazla makinede saklanmaktadır. Bu dosyalar, arıza durumunda çalışmakta olan sistemi olası veri kayıplarından kurtarmak için yedekli şekilde tutulmaktadır. Ayrıca uygulamaları paralel işleme için uygun hale getirmektedir [7].

HDFS; Google File System baz alınarak oluşturulmuş, verinin saklanmasından sorumlu dağıtık dosya tutma düzenine sahip bir platformdur. Üzerinde birden çok "data-node" barındırır ve bunlara bağlı diskleri birleştirir. Verileri varsayılan 64MB veya 128MB bloklar halinde saklar ve bu bloklar varsayılan 3 kopya şeklinde tutulur. Pahalı donanım gereksinimleri gerektirmemektedir.



Şekil 2.1. HDFS Blokları

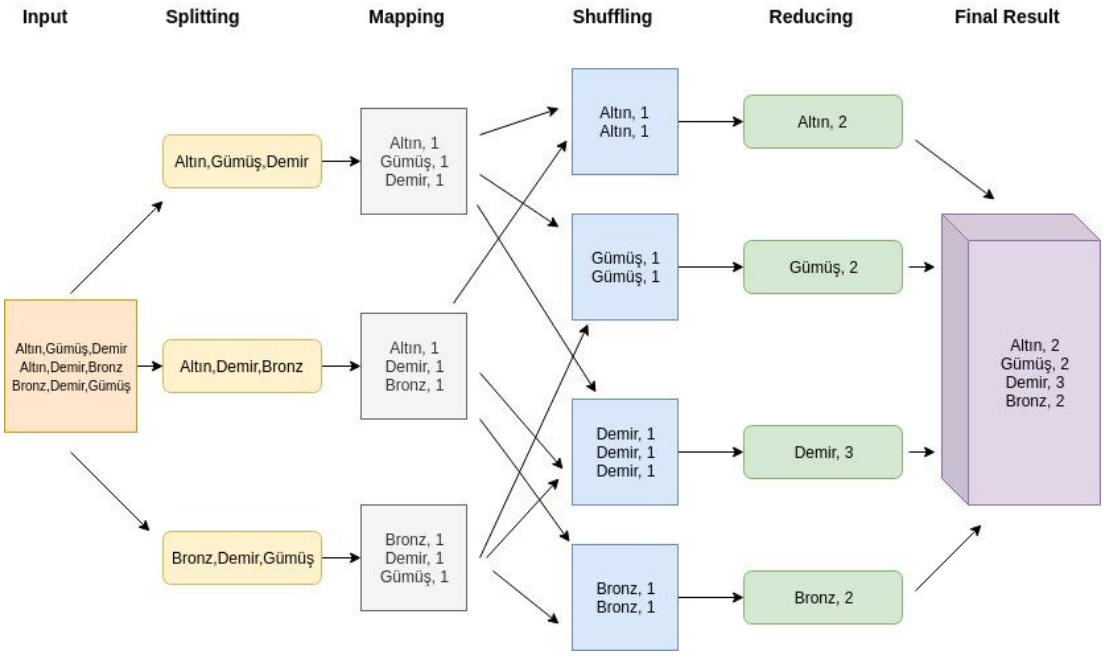
Normal dosya sistemleri üzerinde çalışabilmesi ve komut satırı üzerinden erişilebilmesi gibi özellikleri kolay ve esnek kullanıma örnek olarak gösterilebilir.

MapReduce; platform ve dilden bağımsız bir programlama modelidir. Kayıt tabanlı veri işleme yöntemine sahiptir. Anahtar-değer ikilisi yöntemini kullanır; bunun yanında oluşan görevlerin data-node yani düğümler üzerinde dağıtılmasını sağlar.

JobTracker ve TaskTracker süreçlerinden oluşmaktadır. JobTracker, yazılmış olan MapReduce programının küme üzerinde dağıtılarak çalıştırılmasından sorumlu kısımdır. Bunun yanında dağıtılan iş parçacıklarının çalışması sırasında oluşabilecek herhangi bir problemde o iş parçacığının sonlandırılması ya da yeniden başlatılması işlemi de JobTracker'ın görevleri arasında bulunmaktadır. TaskTracker, data-node'ların bulunduğu sunucularda çalışmaktadır ve JobTracker'dan tamamlanmak üzere iş parçacığı talep etmektedir. JobTracker, NameNode'un yardımı ile data-node'un lokal diskindeki veriye göre en uygun Map işini TaskTracker tarafına yönlendirmektedir. Bu şekilde verilen iş parçacıkları tamamlanmaktadır ve sonuç çıktısı yine HDFS üzerinde bir dosya olarak yazılmakta daha sonrasında program sonlanmaktadır [6].

Kısaca temelde Map ve Reduce olarak iki aşamadan oluşmaktadır. Map verilerin doğru bir şekilde düğümlere dağıtılmasını sağlarken, Reduce ise düğümdeki verilerin

tekilliğini oluşturmaya aynı zamanda da varolan değerlerin (sayıca) tutulmasını sağlamaktadır.



Şekil 2.2. MapReduce Süreci

2.2.2. Veri işleme yöntemleri

Veriyi tümüyle ve/veya yarı otomatik yöntemlerle ya da bunlardan bağımsız olarak otomatik olmayan başka yollara başvurarak ilk kez elde edilmesinden itibaren veri bütününde gerçekleştirilmiş işlemlerin hepsine veri işleme yöntemleri denmektedir.

Bu işlemlere kaydetmek, toplamak, paylaşmak, depolamak, değiştirmek, okumak, açıklamak ya da imha etmek gibi birçok parça örnek olarak gösterilmektedir.

2.2.2.1. Pig, Hive ve Impala

Apache Pig, Apache Hadoop üzerinde sistematik ve prosedüre dayalı bir şekide veri akışı yazılmasına imkân veren bir veri işleme platformudur. Veriyi kullanacak kişi

veya kurumlara Hadoop platformunun güçlü, esnek ve dağıtık yapısına üst seviye java kodları dahil edilmeden daha üst bir yapıdan erişim imkânı sağlayan bir script dilidir.

Java'da o kadar iyi olmayan programcılar normalde Hadoop'la çalışmak için, özellikle de MapReduce görevlerini yerine getirirken zorlanırlardı. Apache Pig, tüm bu programcılar için bir nimettir.

Pig script kullanarak, programcıların Java öğrenmesine ve karmaşık kodlar yazmasına gerek kalmadan veri işlemeyi kolayca gerçekleştirmeleri mümkün olmaktadır.

Apache Pig, tekil sorgulama değil çoklu sorgulama yaklaşımını kullanmaktadır, bu sayede kodların uzunluğu azalmaktadır. Java dilinde 200 satır kod yazılmasını gerektiren bir işlem bütününde Apache Pig kullanılarak en fazla 10 satır kod yazılarak aynı işlem bütünü kolayca gerçekleştirilebilmektedir. Apache Pig, geliştirme süresini neredeyse 16 kat azaltmaktadır [8].

Pig özünde SQL benzeri bir dildir. Bu sebeple SQL'e aşina olan kişilerin Apache Pig'i öğrenmesi çok hızlı ve kolay olmaktadır.

Apache Pig; birleştirme, filtreleme, çağırma vb. veri işlemlerini desteklemek için birçok yerleşik operatöre yani komut türüne sahiptir.

Apache Hive, dağıtık dosya sistemlerinde çok yüksek seviyelerde, örneğin petabayt seviyesinde var olan veri kümelerini yazmayı, okumayı ve yönetmeyi sağlayan SQL diline çok benzer bir biçimde interaktif sorgular atılabilen açık kaynak kod olan ve bu yolla veri işlemeyi sağlayan bir projedir.

Hive, Hadoop platformunda yapılandırılmış verileri işlemek için bir veri ambarı altyapı aracıdır. Büyük verileri özetlemek için Hadoop'un üzerinde yer almaktadır. Sorgulama yapmayı ve analiz etmeyi kolaylaştırmaktadır.

Başlangıçta Hive, Facebook tarafından geliştirilmiştir, daha sonra ise Apache Software Foundation bünyesine katılmış ve Apache Hive adı altında açık bir kaynak olarak geliştirilmeye devam edilmiştir. Farklı şirketler tarafından kullanılmaktadır. Örnek olarak Amazon Şirketi, Amazon Elastic MapReduce'da Hive kullanmaktadır [9].

Impala, Cloudera tarafından geliştirilmiş SQL sorgularına benzeyen sorgular yönlendirebileceğimiz bir veri işleme projesidir. HDFS veya HBase platformunda bulunan veri kümeleri üzerinde, JOIN, SELECT ve AGGREGATION gibi fonksiyonları ile gerçek zamanlı sorgulama yapılabilmektedir.

Impala, HDFS, HBase, Metastore, YARN ve Sentry gibi standart bileşenleri kullanarak, geleneksel bir analitik veritabanının SQL desteğini ve çok kullanıcılı performansını Apache Hadoop'un ölçeklenebilirliği ve esnekliği ile birleştirir.

Impala ile kullanıcılar, Hive gibi diğer SQL motorlarına kıyasla daha hızlı SQL sorguları kullanmakta, HDFS veya HBase ile iletişim kurabilmektedirler.

Impala, Hadoop tarafından kullanılan Parke ya da Avro gibi hemen hemen tüm dosya formatlarını okuyabilmektedir.

Impala, toplu odaklı veya gerçek zamanlı sorgular için birleştirilmiş bir platform sağlayan aynı meta verileri, SQL sözdizimini (Hive SQL), ODBC sürücüsünü ve Apache Hive ile kullanıcı arabirimini (Hue Beeswax) kullanmaktadır.

Apache Hive'den farklı olarak Impala, MapReduce algoritmalarına dayanmamaktadır. Aynı makinelerde çalışan sorgu yürütmenin tüm yönlerinden sorumlu olan daemon işlemlerine dayanan dağıtılmış bir mimari uygulamaktadır [10].

MapReduce kullanma gecikmesini azaltan tüm bu sebepler ise Impala’yı, Apache Hive'den daha hızlı hale getirmektedir.

2.2.2.2. ElasticSearch

ElasticSearch, bünyesinde büyük veriler barındıran ve bu veriler ile çalışan firmaların, isminden de anlaşılacağı üzere indekslenmiş veri üzerinde içerik araması, veri analizi, sorgulamalar ve/veya öneriler gibi işlemlerin yapılabildiği; özellikle performans kabiliyetlerinin güçlü ve bunun yanında esnek olmasından da ötürü tercih edildiği bir ilişkisiz veri tabanıdır. ElasticSearch kullanan büyük firmalara örnek vermek gerekirse bunlar LinkedIn, Stack Overflow, Foursquare, GitHub ve Amazon olacaktır. ElasticSearch yapısının geliştirildiği dil Java dilidir. Ayrıca "Lucene" altyapısı üzerine konuşlandırılmış açık kaynaklı bir projedir.

Gerçek zamanlı olarak dağıtılmış ve açık kaynaklı bir "tam metin arama" ve analiz motoru şeklinde çalışmaktadır. RESTful web servis arayüzünden erişilebilmektedir. Veri depolamak için kalıtsal bir şema ile daha az JSON objesi kullanmaktadır. Java dili üzerine kuruludur. Elasticsearch farklı platformlarda çalışabilmektedir. Kullanıcıların çok büyük miktarda veriyi çok yüksek hızda keşfetmelerini sağlamaktadır.

Elasticsearch'ün genel özellikleri aşağıdaki gibidir:

Elasticsearch ile yapılandırılmış ve/veya yapılandırılmamış veriler petabayta kadar ölçeklenebilmektedir.

Elasticsearch, MongoDB ve RavenDB gibi belge depoları yerine kullanılabilmektedir.

Arama performansını iyileştirmek için denormalizasyon kullanılmaktadır.

Popüler kurumsal arama motorlarından biridir ve hali hazırda Wikipedia, The Guardian, StackOverflow, GitHub, vs. gibi birçok büyük kuruluş tarafından kullanılmakta ve desteklenmektedir.

Elasticsearch açık bir kaynaktır ve Apache lisansı 2.0 sürümü altında mevcuttur.

Kurulumu ve genel konfigurasyon terimleri aşağıdaki gibidir.

Node, tek bir çalışan Elasticsearch örneğini ifade etmektedir. Tek bir fiziksel ve sanal sunucu, RAM, depolama ve işleme gücü gibi fiziksel kaynaklarının özelliklerine bağlı olarak birden çok düğümü barındırmaktadır.

Cluster, bir veya daha fazla düğümün bir koleksiyonu anlamına gelmektedir. Cluster, tüm veriler için tüm düğümlerde toplu dizin oluşturma ve arama yeteneklerini sağlamaktadır.

Index, farklı türde belge ve özelliklerinden oluşan bir koleksiyonu temsil etmektedir. Index, performansı artırmak için "shard" kavramını da kullanmaktadır. Örnek vermek gerekirse bir belge kümesi, bir sosyal ağ uygulamasının verilerini içerebilmektedir.

Document, JSON objesi şeklinde tanımlanmış olan belirli bir biçimde bir alan koleksiyonudur. Her belge bir türe aittir ve bir dizinde bulunmaktadır. Her belge, UID adı verilen benzersiz bir tanımlayıcı ile ilişkilendirilmektedir.

Shard, dizinlerin yatay olarak parçalara bölünmesi anlamındadır. Bu, her bir parçanın tüm özelliklerini içerdiği ancak dizinden daha az sayıda JSON objesi içerdiği anlamına gelmektedir. Yatay ayrılma, parçayı herhangi bir düğümde depolanabilen bağımsız bir düğüm haline getirmektedir. Birincil parça, bir index'in orijinal yatay kısmıdır ve daha sonra bu ana parçalar, kopya parçalar halinde çoğaltılmaktadır.

Replicas, bir kullanıcının indekslerinin ve kırıklarının kopyalarını oluşturmasına izin vermek anlamına gelmektedir. Çoğaltma, başarısızlık durumunda verilerin kullanılabilirliğinin arttırılmasına yardımcı olmakla beraber, aynı zamanda bu kopyalarda paralel bir arama işlemi gerçekleştirerek arama performansını da artırmaktadır [11].

Avantajları ve dezavantajları aşağıda belirtildiği gibidir.

Elasticsearch, Java üzerinde geliştirilmiştir; bu özelliği ile neredeyse her platforma uyumlu olmaktadır.

Elasticsearch gerçek zamanlıdır, başka bir mana ile, bir saniye sonra eklenen belge bu motorda aranabilmektedir

Elasticsearch, her büyük organizasyonu ölçeklendirebilen ve entegrasyonunu kolaylaştıran şekilde dağıtılmıştır.

Tam yedekleme yapmak, Elasticsearch'te bulunan ağ geçidi kavramını kullanarak kolay hale getirilmiştir.

Multi-tenancy kullanımı, Apache Solr'a kıyasla çok kolaydır [11].

Elasticsearch, JSON objelerini yanıt olarak kullanır, bu da Elasticsearch sunucusunu çok sayıda farklı programlama diliyle çağırmayı mümkün kılmaktadır.

CSV, XML ve JSON formatlarında mümkün olan Apache Solr'dan farklı olarak, istek ve yanıt verilerini kullanma konusunda çok dilli desteğe sahip değildir. Sadece JSON objesi olarak cevap dönmektedir.

Bazen, Elasticsearch'ün ayrıştırma ile ilgili bir sorunu olabilmektedir.

Elasticsearch, metin oluşturmayı desteklemeyen türlerin haricinde geriye kalan belge türlerinin neredeyse hepsini desteklemektedir.

Daha önce bahsedilen Pig, Hive ve Impala gibi veri işleme projelerinden ayrılan yönü ise veriyi kendi üzerinde barındıran bir arama motoru yapısına sahip olmasıdır. NoSql tabanlı olup aggregation sorgular ile veri üzerinde elde etmek, yani işlemek istediğiniz veri kümesinin sorgulanmasına izin veren bir projedir.

2.3. Regresyon Analizi ve Deep Learning Kavramı

Regresyon analizi, ilgilenilen iki veya daha fazla değişken arasındaki ilişkinin incelenmesini sağlayan güçlü bir istatistiksel yöntemdir.

Pek çok regresyon analizi türü varken, aslında hepsi bir veya daha fazla bağımsız değişkenin bağımlı değişken üzerindeki etkisini incelemektedir.

Regresyon analizi, ürün ve hizmetleri daha da iyileştirmek için uygulanabilecek ayrıntılı bilgiler sağlamaktadır.

Regresyon analizi, hangi değişkenlerin ilgi konusu üzerinde etkili olduğunu belirlemede güvenilir bir yöntemdir. Bir regresyon gerçekleştirme süreci, hangi faktörlerin en önemli olduğunu, hangi faktörlerin göz ardı edilebileceğini ve bu faktörlerin birbirlerini nasıl etkilediğini güvenle belirlemeyi sağlamaktadır.

Regresyon analizini tam olarak anlamak için aşağıdaki terimlerin anlaşılması önem arz etmektedir.

Bağımlı Değişken: Anlamaya veya tahmin edilmeye çalışılan ana faktördür.

Bağımsız Değişkenler: Varsayılan faktörlerin bağımlı değişkene etkisini temsil eden değerlerdir.

Bir regresyon analizi yapmak için, varsayılan bir ya da birkaç bağımsız değişkenden etkilendiği düşünülen bağımlı bir değişken tanımlanması gerekmektedir. Tüm bu veri kümesini veya setini oluşturabilmek için birçok yöntem bulunmaktadır. Bunlar anket yapmak, şikâyet bildirimleri ya da sosyal medya verileri toplamak şeklinde olabilmektedir.

Günümüzde Machine Learning, Deep Learning ve Yapay Zekâ gibi özelleştirilmiş üç ana başlık altında şekillendirilmiş yazılım teknolojilerinin hızlı bir şekilde artması beraberinde bu üç ana başlığı da ilginin odağı haline getirmiştir.

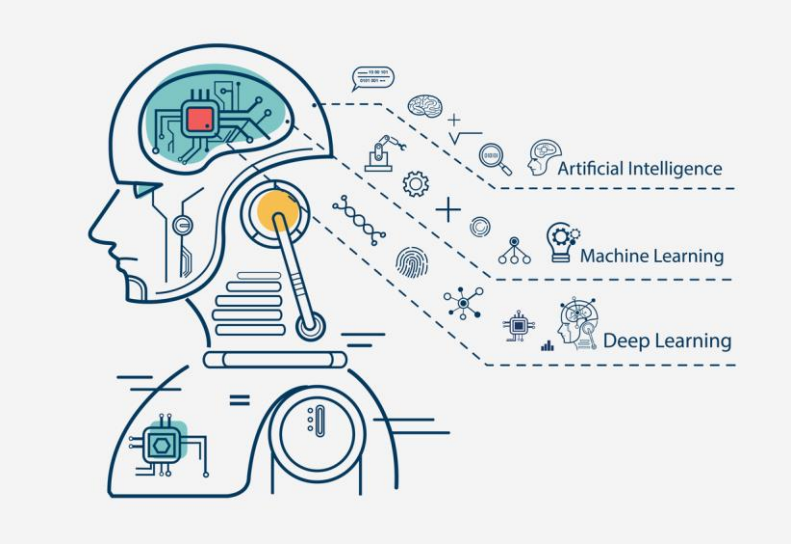
Bu konular üzerinde çalışma gerçekleştirenler, bu kelimelerle ilgili hali hazırda fikir ortaklığına varabilmiş olmamalarına rağmen bazen iç içe bazen ise apayrı anlamlarda kullanılmış olan bu üç ana başlık hakkında gün geçtikçe yeni kavramlar ortaya çıkmaktadır.

Gün geçtikçe ilerlemeye devam eden teknoloji ile birlikte hayatımızın merkezine doğrudan girmiş olan Yapay Zekâ, bilgisayarların karar mekanizmalarını düzenleyen Machine Learning ve bütün bunların altında var olan veri analizini ve algoritmalarını inceleyen Deep Learning, artık sadece mühendislik değil birçok disiplin için ortak olan "gelecek çalışma sahası" olma yolunda ilerlemektedir. Diğerlerine göre daha eski bir terim olduğu için Yapay Zekâ olarak geçen tüm çalışmalar teknolojinin gittiği yönü ve gidiş şeklini de doğrudan etkilemektedir.

Fakat bu kabiliyet ne kadar ilerlerse ilerlesin, Yapay Zekâ ne kadar hayatın içine girerse girsin insan beyni karşısında yenilgi almıştır.

Örnek olarak tekrarlamalardan, kontrol gruplarından veya matematiksel işlemlerden oluşan yapıları kolaylıkla kavrayabilip çözebilirken; mantıksal olarak bakılması gereken, kesin olarak tanımlanmayan ifadelerde mevcut teknoloji yetersiz kalabilmektedir. Bu sebeple bu sistemlere öğrenme yöntemlerini öğretmeyi esas alan Deep Learning (Derin Öğrenme) ortaya koyulmaktadır.

Yapay Zekâ öğrenimi model alınarak öncelikle veri bilimi altında algoritma ve veri kümelerinden Machine Learning'e kadar olan süreç sonrası Big Data olarak adlandırılan, Yapay Zeka'nın en üst noktası haline gelen görüntü işleme ya da analiz etme yöntemleri daha sonrasında ise hemen hemen hepsini bünyesinde barındıran Deep Learning de tüm bu ihtiyaçlardan dolayı son zamanlarda popüler bir başlık halini almıştır.

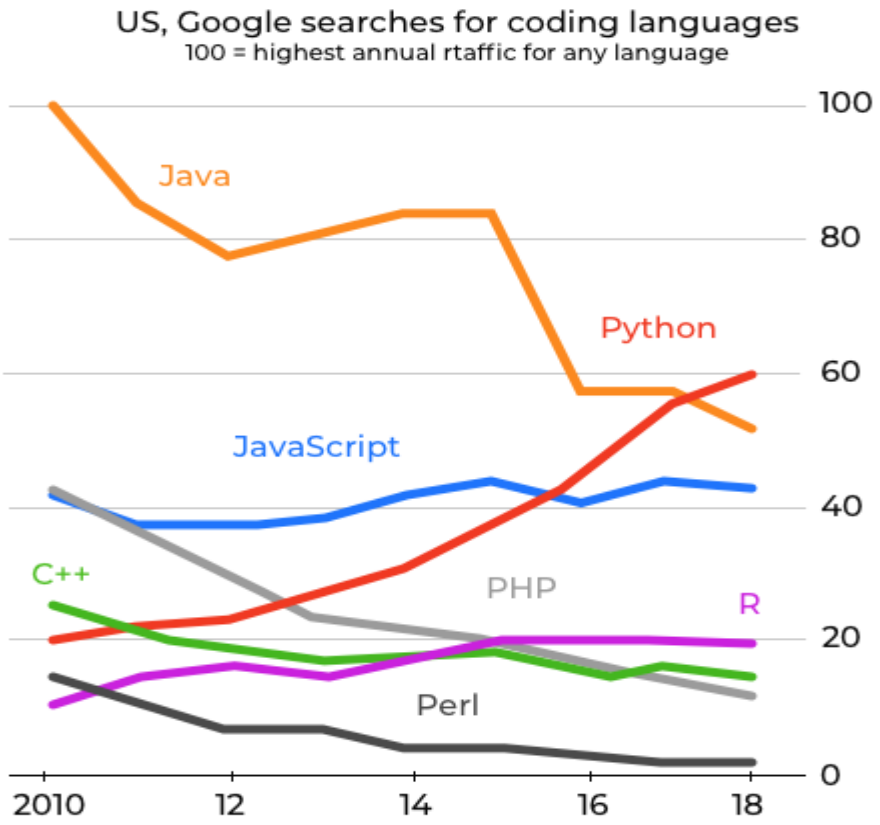


Şekil 2.3. Deep Learning, Machine Learning ve Yapay Zekâ [12].

Derin Öğrenme yani Deep Learning esasen bir makine öğrenme yöntemi olmakla beraber elde edilmiş veya var olan bir veri kümesi üzerinden çıktıları tahmin edebilecek yapay zekâ kavramını eğitmemize olanak sağlayan bir yapıdır. Yapay zekanın eğitilmesi için hem denetimli hem de denetimsiz öğrenme metotları kullanılabilmektedir.

Denetimli Öğrenme yöntemi, girdisi yapılmış verilere ve beklenen veri çıktılarına sahip etiketli veri setlerini dahil ederek yapılan bir öğrenme şeklidir.

Denetimsiz Öğrenme yöntemi ise, denetimli öğrenme yöntemine zıt olarak belirli bir yapıda olmayan veri kümelerini kullanması ile öğrenimini gerçekleştiren bir öğrenme yöntemidir.



Şekil 2.4. Programlama Dilleri Kullanımı [13].

Yazılımsal teknolojilerde kullanılan programlama dilleri barındırdıkları kütüphanelere göre bazı alanlarda öne çıkmaktadırlar. Şekil 2.4. te görüldüğü üzere 2018 yılında Python programlama dili yükselişe geçmektedir. Bu yükselişe sebep olan en büyük etken ise Python dilinin Machine Learning, Deep Learning ve Yapay Zekâ başlıklarında birçok kütüphaneyi kullanmaya olanak sağlamasıdır.

Regresyon analizi yapılırken veya Deep Learning uygulaması gerçekleştirilirken kullanılan programlama dilleri barındırdıkları kütüphaneler açısından büyük önem arz etmektedir.

Çalışma gerçekleştirilirken ihtiyaç duyulan algoritmaların kütüphane kullanımları da büyük ölçüde çalışmayı kolaylaştırmakta ve süreci kısaltmaktadırlar.

Var olan birçok regresyon analizinde, kullanılması gerekli olan kütüphaneler Python geliştiricileri tarafından oluşturulmuştur. Pandas, numpy ve sci-kit learn kütüphaneleri örnek olarak gösterilmektedir.

BÖLÜM 3. MATERYAL VE YÖNTEM

3.1. Materyal

Araştırmada, Sekom Yazılım firmasının proje geliştirme sürecinde kullandığı Karadeniz Bölgesine ait internet altyapı bilgilerinin ve arıza kayıtlarının barındığı test verileri kullanılmıştır.

Test verileri kısıtlı bir bölge için gerçek olmayan fakat gerçek veriler üzerinden yola çıkılarak oluşturulmuş veri setleridir.

Kullanılan veri setlerinde kullanıcıdan saha dolabına kadar olan veriler ilk etapta bina, daha sonra dağıtım kutusu ve en sonunda da saha dolabı şeklinde kümülatif büyümektedir.

Bir kullanıcı bir saha dolabına bağlanabilirken, saha dolabında birden fazla kullanıcı olabilmektedir. Kümülatif artan değerler bu esasa bağlıdır. Hesaplamalar yapılırken ve veriler anlamlı hale getirilirken bina üzerinde var olan kullanıcılardan yola çıkılarak dağıtım kutusu ve saha dolabı özelinde gruplandırılmış ve işlemler gerçekleştirilmiştir. Veri seti kullanıcının kabinet tipinden, kullanım tipine kadar her bilgiyi kullanıcı özelinde barındırmaktadır.

Tek satırda bulunan ve kullanıcıya özel olan kolonlar derin öğrenmeyi kolaylaştırmaktadır.

3.2. Yöntem

3.2.1. Kullanılan araç-gereçler

Çalışmada kullanılan başlıca programlar ve kod blokları, Cloudera Manager, HDFS, Hue arayüzü, Pig Script, Pycharm, Lojistik Regresyon ve Karmaşıklık Matrisi şeklindedir.

Cloudera Manager yüklenerek HDFS ve Hue arayüzü kullanılabilir hale getirilmiştir.

Pig Script ile hızlı ve kolay bir şekilde veri işleme basamakları gerçekleştirilmiştir.

Pycharm Python dili için oldukça kullanışlı bir kod yazma ve derleme platformudur.

Lojistik Regresyon kullanımı kolay ve ikili sonuç ifadelerinde hızlı sonuçlar vermektedir.

Karmaşıklık matrisi sonucun verimliliğini görmek için kullanılmaktadır.

3.2.2. Kullanılan veri işleme yöntemi

Veri işleme yöntemleri birçok dil ve platformda yapılırken bu çalışmada hakim olunan Pig Script kullanılmıştır. Java dilini hızlı bir şekilde öğrenmenin mümkün olmaması ve Big Data tarafında işleri ve süreçleri kolaylaştırmak için ortaya çıkmış ve kullanımı gittikçe yaygınlaşmış Java altyapısına dayanan bir script dili olması ayrıca kullanım gereksinimi oluşturmuştur.

Çalışmada kullanılan Pig Script ile HDFS üzerinde Hue arayüzünden de faydalınalarak veri işlenmesi sağlanmıştır.

HDFS üzerinde tutulan farklı lokasyonlardan elde edilmiş “csv” dosyaları Pig Script ile load, join, group by ve benzeri birçok komut kullanılarak işlenmiştir. Hue arayüzü ile kullanım kolaylığı sağlanmıştır.

3.3. Analizler

3.3.1. Sonuç dosyasının oluşumu ve kontrolü

Belirli bir bölgeye ait çalışmayı gerçekleştirebilmek için var olan test verileri ayrı ayrı sunucularda bulunmaktadır. Klasik ftp yöntemleri ile bu veriler her makinadan ayrı ayrı HDFS sunucusuna çekilmiştir. Burada birbirinden bağımsız halde duran dosyaların tek özelliği bir veya birden fazla kolonun diğer dosyalarda da mevcut olmasıdır. Bu çalışmada ortak kolon “müşteri numarası” olmuştur. Dosyalar birleştirildikten hemen sonra ise kolonlar “saha dolabı” adındaki diğer bir kolon üzerinden işlem görmüştür. Burada bir müşteri numarası sadece bir saha dolabına bağlı olabilirken, bir saha dolabının birden çok müşteri numarası mevcut olabilmektedir. Saha dolabı belirli bir alana/sahaya ait altyapı düğüm noktasıdır.

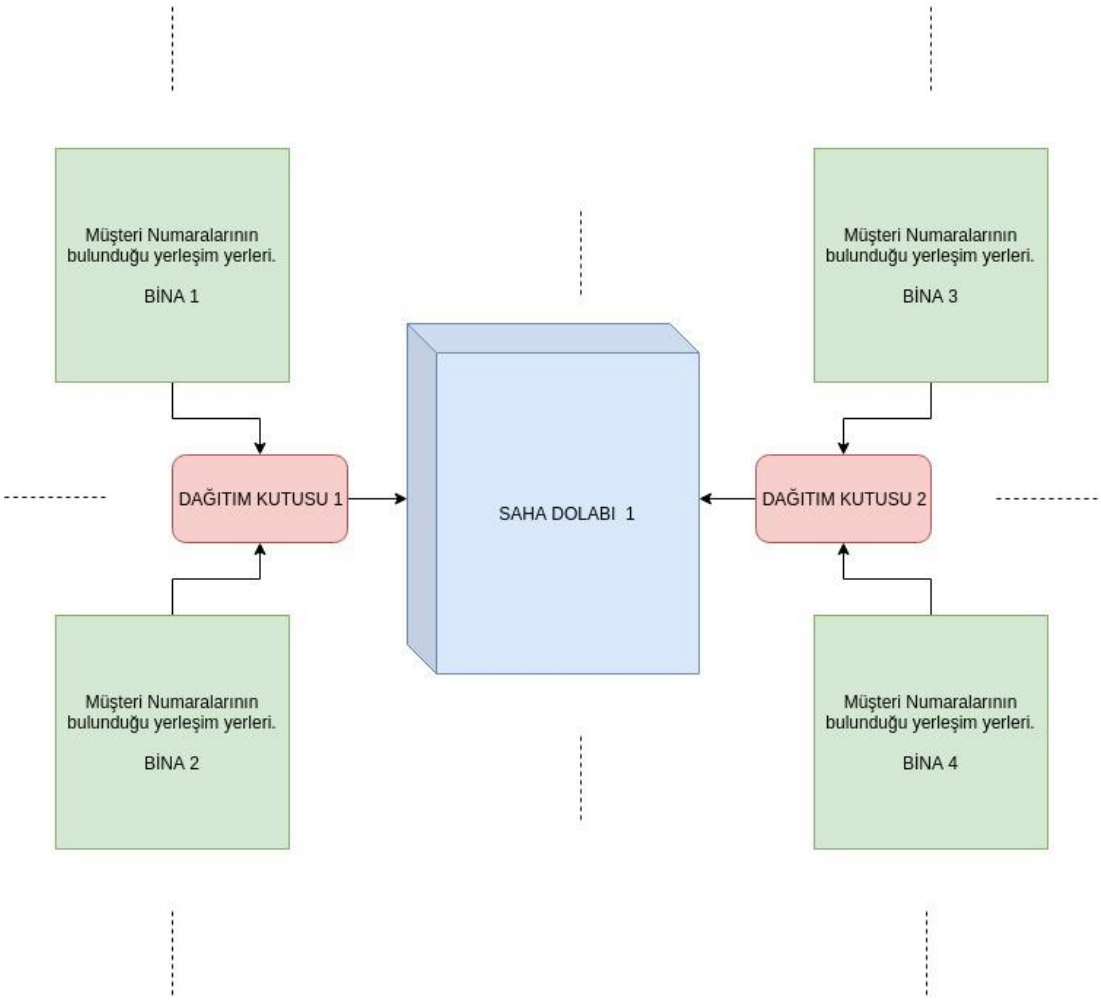
Üç farklı noktadan HDFS üzerine çekilmiş veriler Pig script ile temelde LOAD, JOIN, GROUP BY ve STORE komutları kullanılarak veri seti işlenmiştir.

LOAD komutu: HDFS üzerindeki herhangi bir lokasyondan dosyayı işleme dahil etmeyi sağlamaktadır.

JOIN komutu: İşleme dahil edilmiş farklı iki dosyayı ortak bir kolon aracılığı ile birleştirmeyi sağlamaktadır.

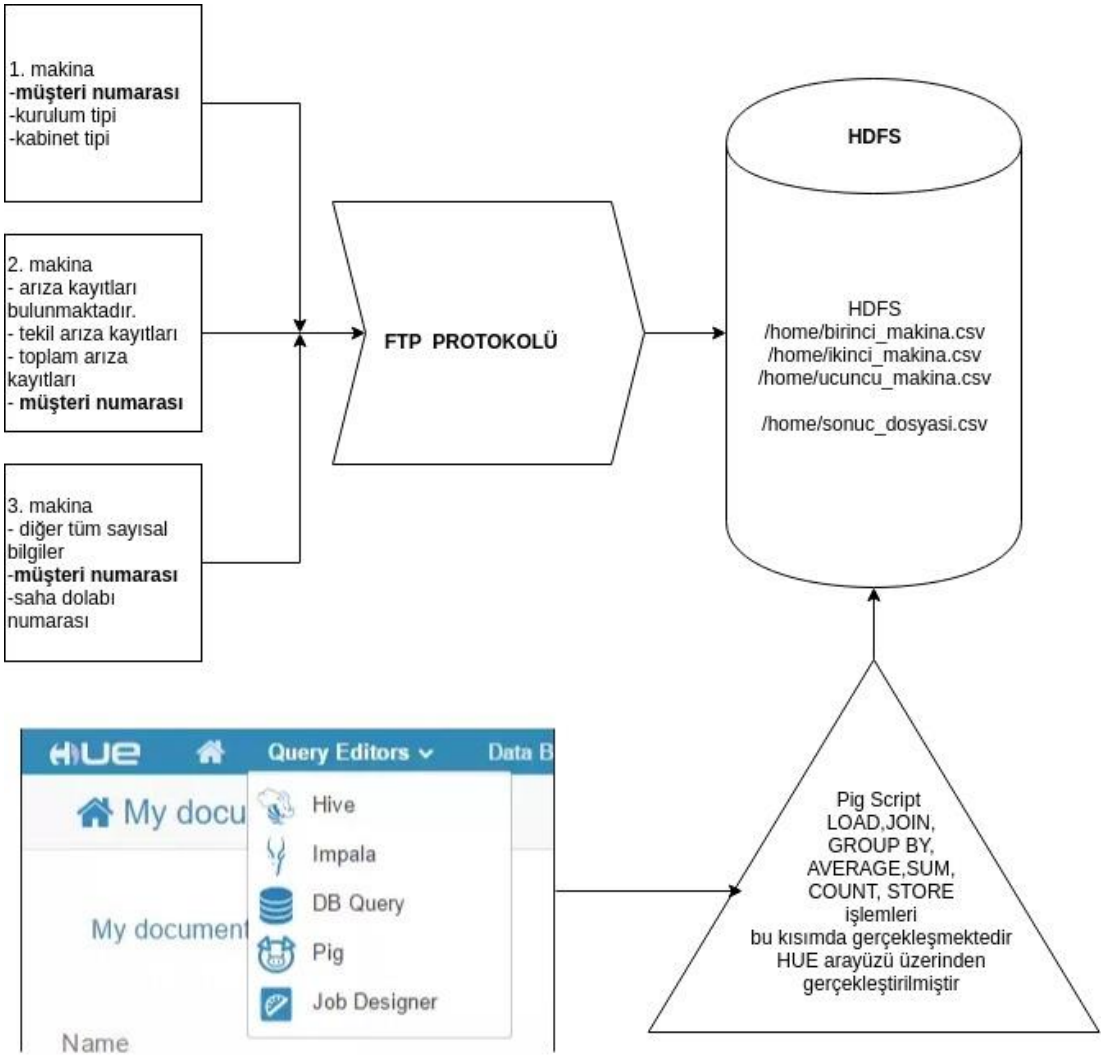
GROUP BY komutu: İşleme dahil edilmiş dosyanın kolonlarından bir veya birden fazla kolona göre işlem yapabilmeyi sağlamaktadır.

STORE komutu: İşlem bitiminden sonra dosyayı istenilen formatta ve istenilen yere kaydetmeyi sağlamaktadır.



Şekil 3.1. Lokasyon Tipleri

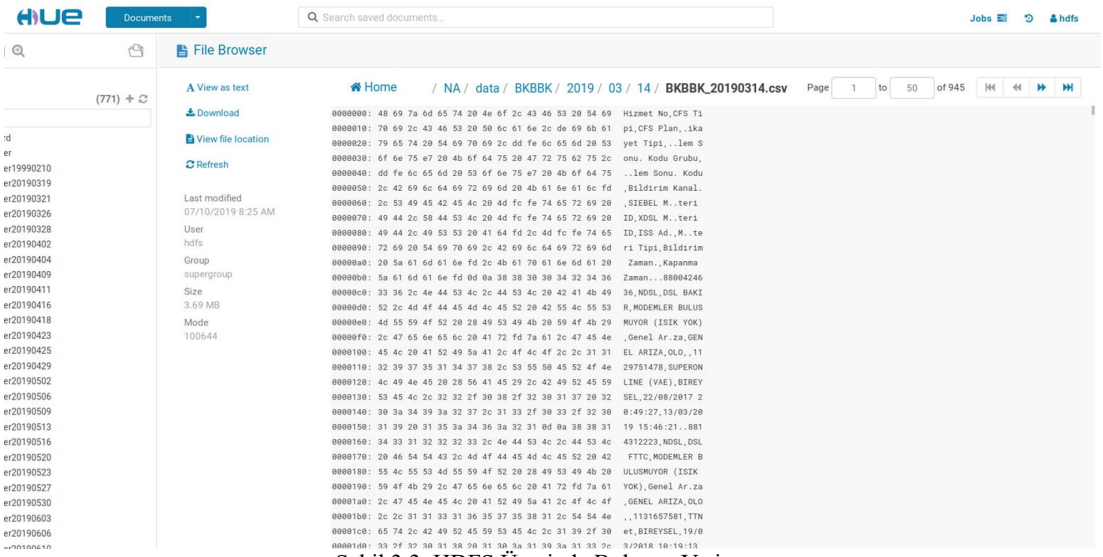
Veri setinin içerisinde bulunan kolonlar saha dolabı numarası, santral ismi, saha alt dolabı ismi, bölge ismi, şehir ismi, bulunduğu lokasyon ismi, kurulum tipi, kabinet tipi, toplam dsl sayısı, toplam sabit hat sayısı, toplam iptv sayısı, toplam adsl sayısı, adsl indirme hızı ortalaması, adsl genel indirme ortalaması, adsl kurulum santral mesafesi, toplam vdsl sayısı, vdsl indirme hızı ortalaması, vdsl genel indirme ortalaması, vdsl kurulum santral mesafesi, toplam arıza sayısı, toplam arıza oranı, tekil arıza sayısı, tekil arıza oranı şeklindedir.



Şekil 3.2. İşlemler Diyagramı

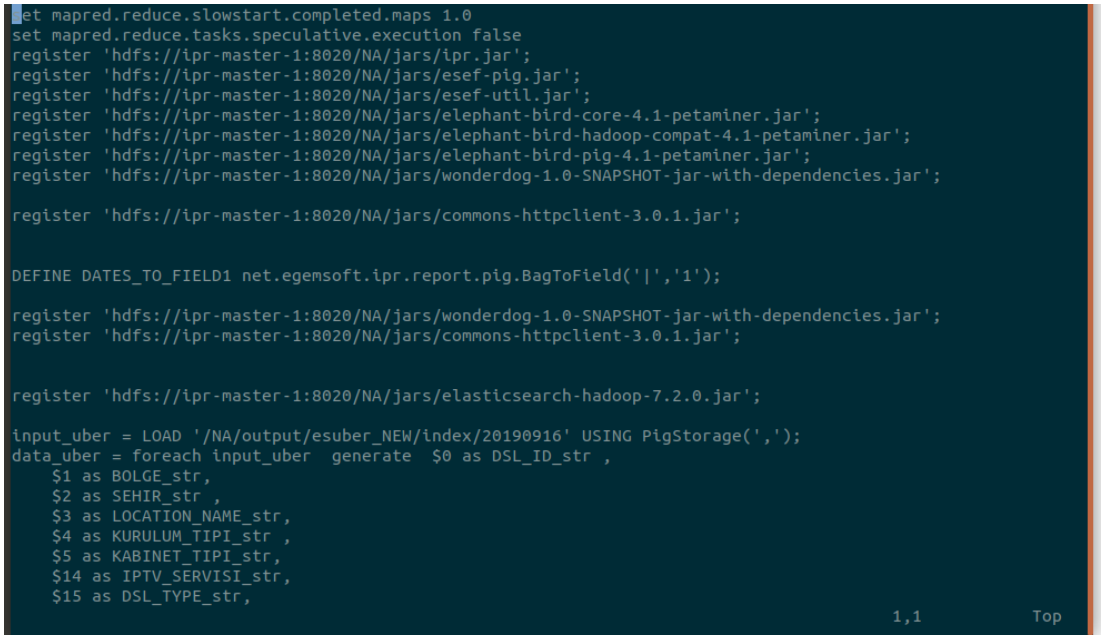
Sonuç dosyası oluştuğunda sadece saha dolabına bağlı müşterilerin sahip oldukları veriler ortaya çıkmaktadır. Artık müşteri numarası değil daha genel bir yaklaşım ile problem kaynağının olduğu noktaların tespiti için işlem yapılabilmektedir. İstenildiği takdirde saha dolabına bağlı dağıtım kutusu ona bağlı binalar ve içerisindeki müşteri numaralarına kadar kümulatif azalma veya tam tersi artma sağlanabilmektedir. Bu çalışmada saha dolabının kullanılmak istenilmesinin nedeni, müşteri numarasına bağlanana kadar arada 3 adet daha bileşen olması ve böylelikle bir sonraki aşama olan derin öğrenme kısmını kolaylaştırmasının planlanmasıdır. Bir düğümde ne kadar çok bağımlı veri oluşursa tahmin yüzdesinin değeri o kadar gerçeğe yakın olacaktır.

Veri işlemede kullanılan veri kümesinin bir kısmı Şekil 3.3 te gösterilmiştir.



Şekil 3.3. HDFS Üzerinde Bulunan Veri

Veri işlemede kullanılan kod bloğunun bir kısmı Şekil 3.4 te gösterilmiştir.



Şekil 3.4. Kullanılan Kod Bloğunun Bir Kısmı

Başlangıçta 3 farklı noktadan çekilen dosyanın son hali Şekil 3.5 de gösterilmiştir.

```
10000000000105,GÜMÜŞHACIKÖY-10,18,SAMSUN (KUZEY 1),AMASYA,05GHACIKOY,INDOOR,,48.0,53,3.0,47.0,10583.897,17965.615,1122.9149,1.0,24576.0,,980.0,,,0.0,0.0,8.0
10000000000106,GÜMÜŞHACIKÖY-10,19,SAMSUN (KUZEY 1),AMASYA,05GHACIKOY|05MERZIFON,INDOOR|OUTDOOR,T7,39.0,39,0.0,37.0,9440.613,15314.032,1346.919,0.0,,,,,0.0,0.0,21.0
10000000000107,GÜMÜŞHACIKÖY-10,21,SAMSUN (KUZEY 1),AMASYA,05GHACIKOY,INDOOR,,77.0,99,0.0,74.0,8724.518,12797.914,1833.6757,0.0,,,,1.0,1.2987013,1.0,1.2987013,38.0
10000000000108,HAMAMÖZÜ-10,1,SAMSUN (KUZEY 1),AMASYA,05HAMAMOZU,INDOOR,,21.0,23,0.0,18.0,11831.833,20610.834,820.7222,2.0,35838.0,,390.0,,,0.0,0.0,6.0
10000000000109,HAMAMÖZÜ-10,2,SAMSUN (KUZEY 1),AMASYA,05HAMAMOZU,INDOOR,,2.0,2,0.0,2.0,1447.5,1934.5,3664.0,0.0,,,,,0.0,0.0,
10000000000113,GÖLKÖY-10,1,SAMSUN (KUZEY 1),AMASYA,05GOLKOY,INDOOR,,7.0,29,0.0,0.0,,,,0.0,,,,,0.0,0.0,4.0
10000000000115,YUKARI OVACIK-10,1,SAMSUN (KUZEY 1),AMASYA,05YUKARIOVACIK,INDOOR,,1.0,2,0.0,0.0,,,,0.0,,,,,0.0,0.0,
10000000000117,YUKARI OVACIK-10,3,SAMSUN (KUZEY 1),AMASYA,05YUKARIOVACIK,INDOOR,,6.0,14,0.0,0.0,,,,0.0,,,,,0.0,0.0,9.0
10000000000118,MERZİFON 1B-10,90,SAMSUN (KUZEY 1),AMASYA,05MERZIFON,OUTDOOR,T7,102.0,107,3.0,0.0,,,,9.0,,,298.0,6.0,5.882353,6.0,5.882353,45.0
10000000000120,PUSACIK-10,2,SAMSUN (KUZEY 1),AMASYA,05PUSACIK,INDOOR,,1.0,5,0.0,0.0,,,,0.0,,,,,0.0,0.0,
10000000000121,KIRCA-10,1,SAMSUN (KUZEY 1),AMASYA,05KIRCA,INDOOR,,9.0,26,0.0,0.0,,,,0.0,,,,,0.0,0.0,4.0
10000000000122,KIRCA-10,3,SAMSUN (KUZEY 1),AMASYA,05KIRCA,INDOOR,,7.0,58,0.0,0.0,,,,0.0,,,,,0.0,0.0,4.0
10000000000123,KIRCA-10,4,SAMSUN (KUZEY 1),AMASYA,05KIRCA,INDOOR,,1.0,9,0.0,0.0,,,,0.0,,,,,0.0,0.0,
10000000000124,KIRCA-10,5,SAMSUN (KUZEY 1),AMASYA,05KIRCA,INDOOR,,1.0,12,0.0,0.0,,,,0.0,,,,,0.0,0.0,
10000000000126,KIRCA-10,9,SAMSUN (KUZEY 1),AMASYA,05KIRCA,INDOOR,,1.0,53,0.0,0.0,,,,0.0,,,,,0.0,0.0,
10000000000127,SARAYÖZÜ-10,1,SAMSUN (KUZEY 1),AMASYA,05SARAYOZU,INDOOR,,2.0,26,0.0,0.0,,,,0.0,,,,,0.0,0.0,
10000000000128,YAZIYERİ-10,1,SAMSUN (KUZEY 1),AMASYA,05YAZIYERI,INDOOR,,17.0,21,0.0,0.0,,,,0.0,,,,1.0,5.882353,1.0,5.882353,23.0
10000000000129,ÇAVUŞKÖY-10,1,SAMSUN (KUZEY 1),AMASYA,05CAVUSKOY,INDOOR,,13.0,31,0.0,0.0,,,,0.0,,,,,0.0,0.0,2.0
10000000000130,ÇAVUŞKÖY-10,2,SAMSUN (KUZEY 1),AMASYA,05CAVUSKOY,INDOOR,,6.0,14,0.0,0.0,,,,0.0,,,,,0.0,0.0,
10000000000131,ÇAVUŞKÖY-10,3,SAMSUN (KUZEY 1),AMASYA,05CAVUSKOY,INDOOR,,19.0,46,0.0,0.0,,,,0.0,,,,1.0,5.263158,1.0,5.263158,12.0
10000000000133,ÇİTLİBAĞLICA-10,1,SAMSUN (KUZEY 1),AMASYA,05CITLIBAGLICA,INDOOR,,7.0,28,0.0,0.0,,,,0.0,,,,,0.0,0.0,3.0
10000000000135,GÜMÜŞ-10,1,SAMSUN (KUZEY 1),AMASYA,05GUMUS,INDOOR,,1.0,4,0.0,0.0,,,,0.0,,,,,0.0,0.0,3.0
10000000000138,MERZİFON 1B-10,27,SAMSUN (KUZEY 1),AMASYA,05MERZIFON,INDOOR,,
```

Şekil 3.5. Sonuç Dosyası

Oluşturulan sonuç dosyası ElasticSearch üzerine de indekslenmiştir.

Şekil 3.6. ElasticSearch Üzerine İndekslenmiş Veri Görüntüsü

Sonuç dosyasının kontrolü saha dolabı numarası üzerinden müşteri numarasının bulunduğu klasöre gidilerek test edilebilir. Burada sayısal tüm veriler ister elle ister script yazarak otomatik bir şekilde kontrol edilebilmektedir. Sonuç dosyası kontrol edildiğinde tüm verilerin her bir dosyadan ayrı ayrı çekilip birleştirildiği, toplama ve

ortalama gibi matematiksel işlemlerin doğru bir şekilde yapıldığı ve sonunda hatasız/eksiksiz bir şekilde yeni dosyaya kaydedildiği görülmüştür.

Burada ayrıca belirtmek gerekirse sadece ftp yöntemleri ile değil shell script yazılarak ve içerisine sql sorgusu koyularak veri tabanlarından da verileri çekilmesi mümkün olacaktır. Hem ftp hem sql sorguları HDFS üzerine dosya çekilirken oozie workflow kullanımı gerçekleştirilerek kolaylıkla sistematik bir yönteme çevirilebilmektedir. Saatlik, günlük, haftalık veya istenilen yani kurgusu yapılmış herhangi bir zaman zarfında çalışması oozie ile mümkün kılınabilmektedir.

3.3.2. Lojistik Regresyon ve Karmaşıklık Matrisi

Lojistik regresyon yöntemi önceleri daha çok tıp alanındaki çalışmalarda kullanılan bir yöntem olmasına rağmen son yıllarda sosyal bilimler alanındaki araştırmalarda popülerlik kazanan Probit modele alternatif oluşturan ileri düzey bir regresyon yöntemidir. Lojistik regresyonun da temelde amacı, diğer regresyon yöntemleri gibi bir ya da birden çok bağımsız değişken ile bağımlı değişken arasındaki ilişkiyi modellemektir [14].

Lojistik Regresyon sınıflandırma işlemini gerçekleştirmeyi sağlayan bir regresyon yöntemidir. Kategorisel veya matematiksel verilerin sınıflandırılmasında kullanılmaktadır. Bağımlı olan değişkenin yani sonuç ifadesinin sadece 2 farklı değer alacağı durumlarda çalışmaktadır (Stabile / Unstabile, Kadın / Erkek, Pozitif / Negatif şeklinde). Doğrusal sınıflandırma yapılacak projelerde genellikle Lojistik Regresyon kullanılmaktadır.

Lojistik regresyon modelinde de kullanılan dil Python dili ve veri analizine yönelik python dili kütüphaneleridir. Lojistik regresyonu anlamak konusunda oldukça kolaylık sağlayan “İris Çiçeğinin Cinsiyet Tahminlemesi” ve kullanılan kütüphaneleri Şekil 3.7. te görülmektedir.

```
# csv dosyalarını okumak için

import pandas as pd
from sklearn.model_selection import train_test_split

# csv dosyamızı okuduk.
data = pd.read_csv('dataset.csv')

# Age kolonundaki değerleri bir değişkene atadık
kurulum = data.iloc[:, 25:26].values

# Bağımlı Değişkeni (sex) bir değişkene atadık
stabilite = data.iloc[:, 24:25].values
id = data.iloc[:, 0:1].values

# len(age)

# Age kolonundaki eksik değerlerin yerine tüm kolonun ortalaması yazıldı.
from sklearn.preprocessing import Imputer

imputer = Imputer(missing_values="NaN", strategy="mean", axis=0)
imputer = imputer.fit(kurulum[:, 0:1])
kurulum[:, 0:1] = imputer.transform(kurulum[:, 0:1])

# DataFrame'e dönüştürdük.
dfAge = pd.DataFrame(data=kurulum, index=range(len(kurulum)), columns=['age'])

# concat fonksiyonu ile bağımsız verileri birleştirdik.
concat = pd.concat([data.iloc[:, 8:12], dfAge, data.iloc[:, 21:24]], axis=1)

# Veri kümemizi test ve train şekinde bölüyoruz
# from sklearn.cross_validation import train_test_split
x_train, x_test, y_train, y_test = train_test_split(concat, stabilite, test_size=0.99, random_state=0)

# LogisticRegression sınıfını import ettik
```

Şekil 3.7. Örnek Lojistik Regresyon Kod Bloğu [15].

Lojistik Regresyon ikili sınıflandırma için en uygun teoremdir. (y = 0 ya da 1 değerlerinde 1 varsayılan sınıfı belirtir. Örnek: Bir olayın meydana gelip gelmeyeceğini tahmin ederken, meydana gelen olay 1 olarak sınıflandırılır. Bir kişinin hasta olup olmayacağını tahmin ederken, hasta örnekleri 1 olarak adlandırılır. Bu teoremde S biçimli bir eğri olan Lojistik Fonksiyon — h(x)= 1/ (1 + ex) — kullanıldığı için bu adı almıştır.) [16].

Karmaşıklık matrisi ise analiz verileri ile tahmin edilmiş olan değerleri karşılaştırabilmeyi ve oluşturulan modelin tahmin yüzdesinin öğrenilmesini sağlar. Çok kısa bir kod bloğu ile kolaylıkla sonuç öğrenilebilmekte ve tahmin yüzdesinin gerçeğe ne kadar yakın olduğu görülebilmektedir.

BÖLÜM 4. ARAŞTIRMA BULGULARI

4.1. Lojistik Regresyon Sonucunda Elde Edilen Tahmin Yüzdesi

HDFS üzerinde oluşturulmuş veri kümesini lojistik regresyona dahil etmeden önce "stabile" ve "unstabile" olarak her bir satır nitelendirilmiştir. Niteleme işlemi yapılırken dikkat edilen husus bir satırda toplam arıza oranının %5 'in üzerinde mi yoksa altında mı olduğudur. Eğer yüzde 5'in üzerinde ise "unstabile", altında ise "stabile" olarak değerlendirilmiştir. Ayrıca kurulum tipi "INDOOR" olanlar "1" diğer değerler "0" kabul edilmiştir.

Veri kümesini lojistik regresyona sokmadan önceki veri kümesi aşağıdaki gibidir.

```
50000000000040,ULUKAVAK,53,SAMSUN (KUZEY 1),CORUM,19ULUKAVAK,INDOOR,,
226.0,219,12.0,210.0,8328.2295,14485445,1063.8,1.0,16383.0,18984.0,861.0,8.0,3.539823,5.0,2.2123895,164.0,stabile,1
50000000000041,ULUKAVAK,54,SAMSUN (KUZEY 1),CORUM,19ULUKAVAK,INDOOR|
OUTDOOR,T7,151.0,140,12.0,1.0,4096.0,7912.0,91.0,23.0,23476.0,97870664,110.434784,13.0,8.609272,12.0,7.94702,85.0,unstabile,0
50000000000042,ULUKAVAK,74,SAMSUN (KUZEY 1),CORUM,
19ULUKAVAK,OUTDOOR,T1,324.0,299,28.0,159.0,12030007,20402852,201.69812,113.0,28510.21,75735.38,176.90265,2.0,0.61728394,2.0,0.61728394,104.0,stabile,0
50000000000043,ULUKAVAK,35,SAMSUN (KUZEY 1),CORUM,19ULUKAVAK,INDOOR,,
168.0,154,12.0,113.0,8983991,16708746,393.55753,34.0,25508258,44567684,385.85294,4.0,2.3809524,3.0,1.7857144,84.0,stabile,1
50000000000044,ULUKAVAK,55,SAMSUN (KUZEY 1),CORUM,19ULUKAVAK,INDOOR,,15.0,62,0.0,11.0,2664.5,3678.8,2951.3635,0.0,,,,5.0,33.333336,5.0,33.333336,22.0,unstabile,1
50000000000045,ULUKAVAK,36,SAMSUN (KUZEY 1),CORUM,19ULUKAVAK,INDOOR,,
102.0,88,4.0,78.0,8933986,17825666,522.88464,16.0,21533875,30755445,530.0,2.0,1.9607844,2.0,1.9607844,28.0,stabile,1
50000000000046,ULUKAVAK,48,SAMSUN (KUZEY 1),CORUM,
19ULUKAVAK,OUTDOOR,T1,82.0,113,10.0,48.0,10329.22,19947977,317.76596,25.0,29905.0,90654.79,203.08,1.0,1.2195121,1.0,1.2195121,47.0,stabile,0
50000000000048,ULUKAVAK,HAYAT,SAMSUN (KUZEY 1),CORUM,
19ULUKAVAK,OUTDOOR,T6,17.0,38,0.0,11.0,7167.1665,18239334,719.4545,2.0,24551.0,66865.0,333.5,4.0,23.529411,3.0,17.647058,18.0,unstabile,0
50000000000049,ULUKAVAK,38,SAMSUN (KUZEY 1),CORUM,19ULUKAVAK,OUTDOOR,T7,115.0,103,5.0,0.0,,,,4.0,,,206.5,3.0,2.6086955,3.0,2.6086955,157.0,stabile,0
50000000000050,ULUKAVAK,39,SAMSUN (KUZEY 1),CORUM,19ULUKAVAK,OUTDOOR,T7,128.0,108,2.0,0.0,,,,1.0,,,240.0,6.0,4.6875,6.0,4.6875,122.0,stabile,0
50000000000051,ULUKAVAK,42,SAMSUN (KUZEY 1),CORUM,19ULUKAVAK,OUTDOOR,T7,96.0,93,9.0,0.0,,,,1.0,,,241.0,8.0,8.333334,8.0,8.333334,58.0,unstabile,0
```

Şekil 4.1. Veri Kümesinin Regresyon Öncesi Hali

Python dilinde yazılmış olup sci-kit learn kütüphanesi kullanılmış ve Imputer fonksiyonu ön işlem sürecinde devreye alınmıştır.

Kod bloğu içerisinde önceden hazırlanmış olan veri seti, Imputer fonksiyonu ile tekrardan kullanıma hazır hale getirilmiştir.

```
def __init__(self, missing_values="NaN", strategy="mean", axis=0, verbose=0, copy=True):
        self.missing_values = missing_values
        self.strategy = strategy
        self.axis = axis
        self.verbose = verbose
        self.copy = copy
```

Şekil 4.2. Imputer Fonksiyonu

Şekil 4.2. de görüldüğü üzere Imputer fonksiyonu ile veri seti tekrar regresyona hazır hale getirilmek üzere işlenmiştir.

Fonksiyon içerisinde missing_values, strategy ve axis tanımlamaları veriyi önceden hazırlamaktadır.

Missing_values : Veri içerisinde ölçümü yapılmamış veya en başta veri tabanında kaydı olmayan değerleri göz ardı etmektedir.

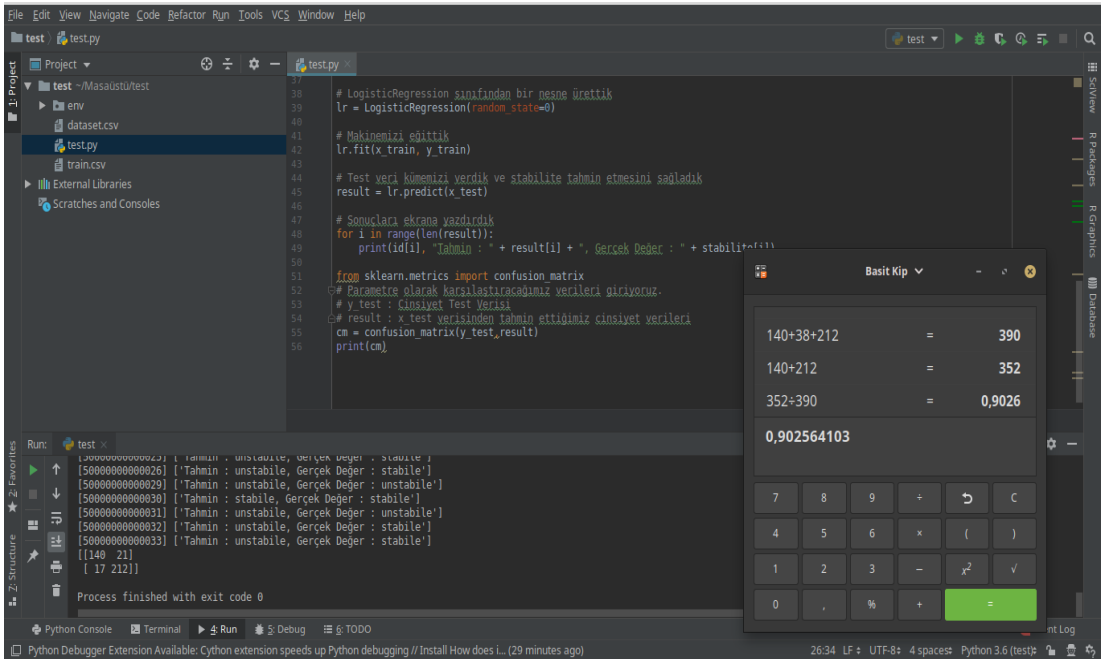
Strategy : Göz ardı edilecek değerlerin hangi stratejiye dayandığını gösteren kısımdır. Mean veya median yönelimleri ile göz ardı edilecek verilerin stratejisi belirlenmektedir.

Axis : Göz ardı edilecek verilerin hangi eksende olacağının belirtildiği tanımlamadır. 0 yatay ekseni, 1 dikey ekseni temsil etmektedir.

Imputer kelimesinin dilimizdeki karşılığı affetmek/göz ardı etmek şeklindedir. Lojistik regresyon analizi içerisinde Imputer fonksiyonu ile daha doğru sonuçlar almak için doğru stratejiler belirlemek gerekmektedir.

Çalışmada belirlenmiş strateji; kayıp değerleri yatay eksende yani satırda göz ardı edilecek ve mean yönelimi olacak şekilde kullanılmıştır. Bahsedilen yönelim matematiksel olarak ortalamaya etki edecek çok yüksek veya çok düşük değerleri göz ardı etmiştir.

Pycharm üzerinde yazılmış, lojistik regresyon uygulanmış veri setinin sonucu aşağıdaki şekilde elde edilmiştir. Karmaşıklık matrisi ile sonucun tahmin yüzdesi hesaplanmıştır.



Şekil 4.3. Lojistik Regresyon Sonucu

Regresyon içerisinde veri setinin tamamı öğrenmeye dahil edilip %33'ü üzerinden değerlendirilme yapılmıştır. Tamamı değerlendiridiğinde bu oran %90 verimliliğinin altına düşecektir.

Karmaşıklık matrisi sonuçlandırılması 3 başlık altında değerlendirilmektedir. Doğruluk, duyarlılık ve özgüllük.

Doğruluk, duyarlılık ve özgüllük arasındaki ilişkiyi tanımlamak için bazı rastgele çıktılar göz önüne alınarak, aynı veri sayısına ve farklı veri sayısına sahip iki veri setini örnekleyerek karşılaştırmalı bir çalışma yapılır. Sonuçlardan, doğruluk duyarlılık ve özgüllük'ten kaynaklandığı için tek başına doğruluğun güvenilir bir faktör olmadığı görülmektedir.

Sonuç olarak doğruluk = (duyarlılık + özgüllük) / 2 şeklinde hesaplamak gerekmektedir [17].
Duyarlılık = doğru pozitif değerler / (doğru pozitif değerler + yanlış negatif değerler)

Özgüllük = doğru negatif değerler / (doğru negatif değerler + yanlış pozitif değerler)

Yapılan çalışmaya uyarlandığında;

Duyarlılık = 140 / (140 + 21) = 0.87

Özgüllük = 212 / (212 + 17) = 0.93

Doğruluk = (0.87 + 0.93) / 2 = 0.90 şeklinde hesaplanmaktadır.

BÖLÜM 5. TARTIŞMA VE SONUÇ

Bu çalışmada farklı noktalarda bulunan ve tek başına anlam taşımayan veri kümeleri bir noktada toplanmıştır. Toplama işlemi ftp ve veri tabanlarına atılan SQL sorguları ile sağlanmıştır.

Toplanmış olan veri kümeleri belirli bir kolon ismine göre birleştirilmiştir. Bu işlem pig script ile gerçekleştirilmiştir. Bazı veri kümeleri işlem basamaklarını gerçekleştirebilmek için LOAD komutu ile script içerisinde tanımlanmıştır.

Veri kümelerininden birkaçı LOAD komutundan sonra kendi içerisinde işleme tabii tutulmuştur. Birleştirme işleminden önce işlem görmesi gereken kolonlar (hesaplama ya da işleme dahil etmeme) birleştirmenin yapılacağı kolona göre gruplanmıştır.

Birleştirilen bu veri tekrar birleştirilmiş kolona göre işlem görmüş ve sonuç dosyası elde edilmiştir. Daha sonrasında muhafaza ve kontrol için ElasticSearch' e indeksleme işlemi gerçekleştirilmiştir.

Sonuç dosyası üzerinde Lojistik Regresyon uygulanması için bazı değişiklikler yapılmıştır. String yapıda olan bazı kolonlar 1 veya 0 şeklinde tanımlanmıştır. Bu tanımlamanın sebebi ise hem bu yapıya yani 1 ve 0 mantığına uygun olmaları hem de regresyon içerisinde kullanabilmektir. Kurulum Tipi ve Kabinet Tipi değerlerinin dönüşüm işlemi bu yapıya örnek gösterilmiştir.

Yapılan değişikliklerin hemen ardından elde edilen son veri kümesi üzerinde Lojistik Regresyon uygulanmış ve bir tahmin yüzdesi elde edilmiştir. Karmaşıklık Matrisi uygulaması ile doğruluk yüzdesi hesaplanmıştır.

Karmaşıklık Matrisi sonucu ile görülmektedir ki yapılan regresyon analizi ile ortaya çıkan sonuç yol gösterici olmaktadır.
Doğruluk oranının yüksekliği, veri kümelerinin birleştirilerek doğru analizler ile doğru yatırımların yapılabileceğini ortaya koymaktadır. Bölge'den, son kullanıcıya kadar elde edilmiş verilerin Big Data olarak değerlendirilmesi ve işlemler bütünün sağlanabilmesi, Big Data yöntemlerinin farklı projelerde ya da arge çalışmalarında da verimi yüksek oranda arttırabileceğini kanıtlar niteliktedir.

Sonuç olarak anlamsız gözüken veri kümeleri birleştirilerek ve işleme tabii tutularak anlamlı veriler üretilmiş ve regresyon analizinden de faydalanılarak oluşturulan bu tahmin yüzdesi şahıs ve/veya kurumların ellerindeki veriler ile yeni ve yol gösterici veriler oluşturulabileceği gösterilmiştir.

KAYNAKLAR

[1] M. Güngör, G. Evren, İnternet Sektörü ve Türkiye İncelemeleri, Ankara, 13 Mayıs 2002.

[2] M. D. Assuno, R. N. Calheiros, S. Bianchi, M. A. S. Netto, and R. Buyya, "Big Data Computing and Clouds: Trends and Future Directions," J. Parallel Distrib. Comput., vol. 79–80, pp. 3–15, 2015.

[3] T. H. Davenport Big Data at Work 2014.

[4] https://alimutlu.com.tr/big-data-nedir-ne-ise-yarar-kullanim-alanlari-nelerdir, Erişim Tarihi: 27.11.2019.

[5] https://www.tutorialspoint.com/hadoop/index.htm, Erişim Tarihi: 22.11.2019.

[6] http://devveri.com/hadoop-nedir, Erişim Tarihi: 27.11.2019.

[7] https://www.tutorialspoint.com/hadoop/hadoop_hdfs_overview.htm, Erişim Tarihi: 27.11.2019.

[8] https://www.tutorialspoint.com/apache_pig/apache_pig_overview.htm, Erişim Tarihi: 16.11.2019.

[9] https://www.tutorialspoint.com/hive/hive_introduction.htm, Erişim Tarihi: 22.11.2019.

[10] https://www.tutorialspoint.com/impala/impala_overview.htm, Erişim Tarihi: 21.11.2019.

[11] https://www.tutorialspoint.com/elasticsearch/elasticsearch_basic_concepts.htm, Erişim Tarihi: 22.11.2019.

[12] https://ceotudent.com/deep-learning-derin-ogrenme-nedir, Erişim Tarihi: 5.11.2019.

[13] https://steelkiwi.com/blog/python-for-ai-and-machine-learning/, Erişim Tarihi: 11.11.2019.

[14] https://medium.com/@cerebro.tech/denetimli-makine-ogrenmesi-algoritmalari, Erişim Tarihi: 19.11.2019.

[15] https://medium.com/@ekrem.hatipoglu/, Erişim Tarihi: 17.11.2019.

[16] İ. Ege, A. Bayrakdaroğlu, ZKÜ Sosyal Bilimler Dergisi, Cilt 5, Sayı 10, 2019

[17] http://www.lifescience.com/bioinformatics/sensitivity-specificity-accuracy-and, Erişim Tarihi: 27.11.2019.

ÖZGEÇMİŞ

Talha ALTUN, 25.12.1991'de İstanbul'da dünyaya gözlerini açtı. İlk okul, orta okul ve lise öğrenimini burada tamamladı. 2010 senesinin Eylül ayında başladığı Karabük Üniversitesi Elektrik-Elektronik Mühendisliği Bölümü'nü ise 2015 senesinin Haziran ayında tamamladı. 2016 senesinde Sakarya Üniversitesi Elektrik Elektronik Mühendisliği Bölümü'nde yüksek lisans (master programı) eğitimine başlamış bulundu. 2019 senesinde Sekom Yazılım şirketinde Yazılım Uzmanı olarak çalışmaya başladı, ardından yüksek lisans eğitiminin tez aşamasını Sakarya Üniversitesi Elektrik Elektronik Mühendisliği Bölümü'nde devam ettirdi. Halen Sekom Yazılım şirketinde Yazılım Uzmanı olarak görev yapmaktadır.