T.C.
İSTANBUL ÜNİVERSİTESİ-CERRAHPAŞA
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

BÜYÜK VERİ ARAÇLARI İLE COVİD-19 PANDEMİSİNİN İSTANBUL TRAFİĞİNE ETKİSİNİN ANALİZİ

Uğur ALCAN

DANIŞMAN
Prof. Dr. Fırat KAÇAR

Elektrik-Elektronik Mühendisliği Anabilim Dalı

Elektrik-Elektronik Mühendisliği Programı

İSTANBUL-2022

Bu çalışma, 11.01.2022 tarihinde aşağıdaki jüri tarafından Elektrik-Elektronik Mühendisliği Anabilim Dalı, Elektrik-Elektronik Mühendisliği Programında Yüksek Lisans tezi olarak kabul edilmiştir.

Tez Jürisi

Prof. Dr. Fırat KAÇAR(Danışman)
İstanbul Üniversitesi-Cerrahpaşa
Mühendislik Fakültesi

Doç. Dr. Abdurrahim AKGÜNDOĞDU
İstanbul Üniversitesi-Cerrahpaşa
Mühendislik Fakültesi

Doç. Dr. Muhammed Emin BAŞAK
Yıldız Teknik Üniversitesi
Gemi İnşaatı ve Denizcilik Fakültesi

20.04.2016 tarihli Resmi Gazete'de yayımlanan Lisansüstü Eğitim ve Öğretim Yönetmeliğinin 9/2 ve 22/2 maddeleri gereğince; Bu Lisansüstü teze, İstanbul Üniversitesi-Cerrahpaşa'nın abonesi olduğu intihal yazılım programı kullanılarak Lisansüstü Eğitim Enstitüsü'nün belirlemiş olduğu ölçütlere uygun rapor alınmıştır.

ÖNSÖZ

Tezimin hazırlanma süresince benden yardımlarını, bilimsel katkı ve desteğini esirgemeyen; hem Lisans hem de Yüksek Lisans dönemlerimde bilgi ve tecrübesiyle kendisine hayranlık duyup, şahsi olarak örnek aldığım değerli hocam Prof. Dr. Fırat Kaçar'a, İstanbul Üniversitesi-Cerrahpaşa Mühendislik Fakültesi Elektrik-Elektronik Mühendisliği bölümü öğretim üyelerimize ve destekleriyle her zaman yanımda olan aileme ve arkadaşlarıma çok teşekkür ederim.

Ocak 2022 Uğur ALCAN

İÇİNDEKİLER

Sayfa No

ÖNSÖZ ..... **iv**
İÇİNDEKİLER ..... **v**
ŞEKİL LİSTESİ ..... **viii**
TABLO LİSTESİ ..... **xi**
SİMGE VE KISALTMA LİSTESİ ..... **xiii**
ÖZET ..... **xiv**
SUMMARY ..... **xvi**
1. GİRİŞ **1**
2. GENEL KISIMLAR **3**
2.1. BÜYÜK VERİ 3
2.2. BÜYÜK VERİ BİLEŞENLERİ 4
2.2.1. Variety (Çeşitlilik) 4
2.2.2. Volume (Hacim/Büyüklük) 4
2.2.3. Velocity (Hız) 4
2.2.4. Verification (Doğruluk) 5
2.2.5. Value (Değer) 5
2.3. BÜYÜK VERİNİN KULLANIM ALANLARI 5
2.4. BÜYÜK VERİ ARAÇLARI 7
2.4.1. Apache Hadoop 7
2.4.1.1. Hdfs (Hadoop Distributed File System) 8
2.4.1.2. Yarn (Yet Another Resource Negotiator) 8
2.4.1.3. MapReduce 8
2.4.1.4. Hadoop Common 8
2.4.2. Apache Kafka 8
2.4.3. Apache Storm 8
2.4.4. Apache Zookeeper 9
2.4.5. Apache Spark 9
2.4.6. Apache Nifi 9
2.4.7. Apache Pig 9
2.4.8. Apache Hive 10

2.4.9. Apache Cassandra 10
2.4.10. Apache Flume 10
2.4.11. Apache HBase 10
2.4.12. Apache Flink 10
2.4.13. Sqoop 10
2.4.14. Apache Solr 10
2.4.15. Elasticsearch 11
2.4.16. Apache Mahout 11
2.5. LİTERATÜR ÖZETİ 11
3. MALZEME VE YÖNTEM 14
3.1. VERİ SETİ 14
3.2. ÇALIŞMADA KULLANILAN YÖNTEM VE ARAÇLAR 17
3.2.1. Intellij Idea 17
3.2.2. Java 17
3.2.3. Apache Spark 18
3.2.3.1. Spark Core 19
3.2.3.2. Spark SQL 21
3.2.3.3. Spark MLlib 30
3.2.3.4. Spark Streaming 41
3.2.3.5. GraphX 42
3.2.4. Elasticsearch 42
3.2.5. Kibana 44
4. BULGULAR 46
4.1. 2020 YILI İSTANBUL TRAFİK DURUMU ANALİZ SONUÇLARI 46
4.2. TRAFİK DURUMU TAHMİNİ MAKİNE ÖĞRENMESİ UYGULAMALARI 56
4.2.1. Lojistik Regresyon Yöntemi İle Trafik Durumu Tahmini 57
4.2.2. Karar Ağaçları Yöntemi İle Trafik Durumu Tahmini 58
4.2.3. Rastgele Orman Yöntemi İle Trafik Durumu Tahmini 60
4.2.4. Gradyan Yükseltme Ağacı Tabanlı OneVsRest Yöntemi İle Trafik Durumu Tahmini 62
4.2.5. Doğrusal Destek Vektör Makinesi Tabanlı OneVsRest Yöntemi İle Trafik Durumu Tahmini 64
5. TARTIŞMA VE SONUÇ 67
KAYNAKLAR 71
EKLER 75

EK 1. Düzenlenmiş trafik indeksi veri seti.....75
ÖZGEÇMİŞ 76

ŞEKİL LİSTESİ

Sayfa No

Şekil 2.1: Büyük verinin 5V'si [6]. 4

Şekil 2.2: 2020 yılında her 1 dakikada internette veri kullanımı [8]. 5

Şekil 2.3: Hadoop ekosistemi [11]. 7

Şekil 3.1: Düzenlenmiş veri setinden örnek. 16

Şekil 3.2: Intellij IDEA ara yüzü 17

Şekil 3.3: Apache Spark bileşenleri [39] 19

Şekil 3.4: JavaRDD örneği. 21

Şekil 3.5: Müşteriler.csv dosyasının içeriği. 22

Şekil 3.6: Müşteriler.csv dosyasının RDD ile analizi. 22

Şekil 3.7: Müşteriler.csv dosyasının Dataset ile analizi. 23

Şekil 3.8: Müşteriler.csv dosyasının RDD ile analizi sonucu. 23

Şekil 3.9: Müşteriler.csv dosyasının Dataset ile analizi sonucu. 24

Şekil 3.10: Müşteriler.csv dosyasının Spark SQL yapısı ile analizi. 24

Şekil 3.11: 2020-saatlik.csv dosyasının Spark SQL yapısı ile okunması. 25

Şekil 3.12: Haftanın günlerine göre İstanbul 2020 yılı trafik indeksi analizi uygulaması. 27

Şekil 3.13: Haftanın günlerine göre İstanbul 2020 yılı ortalama trafik indeksi sıralaması. 28

Şekil 3.14: Gunluk_ortalama_index Dataset'inin elasticsearch'e yazılması. 29

Şekil 3.15: Tez kapsamında yapılan analizler için genel süreç. 29

Şekil 3.16: Analizlerde yapılan temel işlemlerin blok diyagramı ile ifade edilmesi. 29

Şekil 3.17: Makine öğrenmesi için veri setinde yapılan işlemler. 31

Şekil 3.18: Makine öğrenmesi için hazırlanan modellerin doğruluk oranlarının kontrol edilmesi 31

Şekil 3.19: Normal düzey trafik için başarı metriklerin oluşturulması. 32

Şekil 3.20: Makine öğrenmesi uygulamalarında yapılan temel işlemlerin süreci. 33

Şekil 3.21: Lojistik regresyon modelinin oluşturulması. 34

Şekil 3.22: Örnek bir karar ağacı algoritması [48]. 36

Şekil 3.23: Karar ağacı modelinin oluşturulması. 36

Şekil 3.24: Rastgele orman modelinin oluşturulması. 37

Şekil 3.25: Gradyan yükseltme ağacı tabanlı OneVsRest modelinin oluşturulması. 39

Şekil 3.26: Destek vektör makineleri için iki sınıflı problem örneği [52]. 40

Şekil 3.27: Doğrusal destek vektör makineleri tabanlı OneVsRest modelinin oluşturulması. 40

Şekil 3.28: Spark Streaming için temel kaynak ve hedef örnekleri [53]. 41

Şekil 3.29: Spark Streaming temel çalışma prensibi [53]. 42

Şekil 3.30: Elasticsearch indexleme örneği [54]. 43

Şekil 3.31: Günlük_trafik_index type ve index isimli verinin Elasticsearch'den okunması 44

Şekil 3.32: Kibana ara yüzü. 45

Şekil 3.33: Kibana ile haftanın günlerinin ortalama trafik indeksi değerlerinin sütun grafiği ve tablosu. 45

Şekil 4.1: İstanbul 2020 yılı saatlik bazda ortalama trafik indeksi grafiği. 47

Şekil 4.2: İstanbul 2020 yılı sokağa çıkma yasaklarının saatlik bazda ortalama trafik indeksine etkisi grafiği. 47

Şekil 4.3: İstanbul 2020 yılı vakitlere göre ortalama trafik indeksi alan grafiği. 48

Şekil 4.4: İstanbul 2020 yılı haftanın günlerine göre ortalama trafik indeksi grafiği. 49

Şekil 4.5: İstanbul 2020 yılı sokağa çıkma yasaklarının, haftanın günlerine göre ortalama trafik indeksine etkisi grafiği. 51

Şekil 4.6: İstanbul 2020 yılı gün–saat bazda ortalama trafik indeksi ısı haritası. 51

Şekil 4.7: İstanbul 2020 yılı sokağa çıkma yasaklarının gün–saat bazda ortalama trafik indeksine etkisi grafiği. 52

Şekil 4.8: İstanbul 2020 yılı aylara göre ortalama trafik indeksi grafiği. 53

Şekil 4.9: İstanbul 2020 yılı ay–saat bazlı ortalama trafik indeksi grafiği.....54

Şekil 4.10: İstanbul 2020 yılı ilk 5 ay için ay–saat bazlı ortalama trafik indeksi grafiği.....55

Şekil 4.11: İstanbul 2020 yılı ay–gün bazlı ortalama trafik indeksi grafiği.....55

Şekil 4.12: İstanbul 2020 yılı ortalama trafik indeksi ısı haritası. 56

Şekil 4.13: Trafik durumunu lojistik regresyon yöntemi ile tahmin eden modelin örnek çıktısı ve doğruluk oranı.....57

Şekil 4.14: Lojistik regresyon yöntemiyle yapılan modelin özet sonuçları. 58

Şekil 4.15: Trafik durumunu karar ağaçları ile tahmin eden modelin örnek çıktısı ve doğruluk oranı. 59

Şekil 4.16: Karar ağacı yöntemiyle yapılan modelin özet sonuçları.....59

Şekil 4.17: Trafik durumu tahmininde farklı ağaç sayıları için rastgele orman metodunun doğruluk oranı. 60

Şekil 4.18: Trafik durumunu rastgele orman yöntemiyle tahmin eden modelin örnek çıktısı ve doğruluk oranı.....61

Şekil 4.19: Rastgele orman yöntemiyle yapılan modelin özet sonuçları. 61

Şekil 4.20: Trafik durumunu GBT tabanlı OneVsRest yöntemiyle tahmin eden modelin örnek çıktısı ve doğruluk oranı.....63

Şekil 4.21: GBT tabanlı OneVsRest yöntemiyle yapılan modelin özet sonuçları. 63

Şekil 4.22: Trafik durumunu SVM tabanlı OneVsRest yöntemiyle tahmin eden modelin örnek çıktısı ve doğruluk oranı.....65

Şekil 4.23: SVM tabanlı OneVsRest yöntemiyle yapılan modelin özet sonuçları. 65

Şekil 5.1: Trafik Durumu tahmininde kullanılan yöntemlerin doğruluk oranları karşılaştırılması.....68

Şekil 5.2: Trafik Durumu tahmininde kullanılan yöntemlerin ortalama çalışma sürelerinin karşılaştırılması. 69

TABLO LİSTESİ

Sayfa No

Tablo 3.1: Düzenlenmiş veri setinin nitelikleri ve açıklamaları. 15

Tablo 3.2: Trafik indeksi ile ilgili özet durum. 16

Tablo 3.3: Apache Spark örnek transformation metodları ve açıklamaları. 19

Tablo 3.4: Apache Spark örnek action metodları ve açıklamaları. 20

Tablo 3.5: Analiz çalışmasında kullanılan örnek fonksiyonlar ve açıklamaları. 25

Tablo 3.6: Analiz çalışmasında kullanılan örnek tarih fonksiyonları ve açıklamaları. 26

Tablo 3.7: Karışıklık matrisi şeması. 32

Tablo 3.8: Lojistik regresyon yönteminin temel parametreleri ve varsayılan değerleri. 34

Tablo 3.9: Karar ağacı yönteminin temel parametreleri ve varsayılan değerleri. 36

Tablo 3.10: Rastgele orman yönteminin temel parametreleri ve varsayılan değerleri. 37

Tablo 3.11: Gradyan yükseltme ağacı yönteminin temel parametreleri ve varsayılan değerleri. 39

Tablo 3.12: Doğrusal destek vektör makineleri yönteminin temel parametreleri ve varsayılan değerleri. 41

Tablo 3.13: Elasticsearch ve RDBMS'in birbirlerine benzerlik gösteren temel bileşenleri. 42

Tablo 3.14: Apache Spark ile Elasticsearch'den veri okuma ve yazma işlemleri için gerekli kod parçacıkları. 43

Tablo 4.1: İstanbul 2020 yılı saatlik bazda ortalama trafik indeksi sıralaması. 46

Tablo 4.2: İstanbul 2020 yılı vakitlere göre ortalama trafik indeksi sıralaması. 48

Tablo 4.3: İstanbul 2020 yılı haftanın günlerine göre ortalama trafik indeksi sıralaması. 49

Tablo 4.4: İstanbul 2020 yılı sokağa çıkma yasaklarının gün bazında ortalama trafik indeksine etkisi. 50

Tablo 4.5: İstanbul 2020 yılı aylara göre ortalama trafik indeksi sıralaması. 53

Tablo 4.6: Makine öğrenmesi uygulamalarında kullanılan eğitim ve test verisinin dağılımı. 57

Tablo 4.7: Lojistik regresyon yöntemiyle yapılan modelin karışıklık matrisi. 58

Tablo 4.8: Lojistik regresyon yöntemiyle yapılan modelin başarı metrikleri. 58

Tablo 4.9: Karar ağaçları yöntemiyle yapılan modelin karışıklık matrisi. 60

Tablo 4.10: Karar ağaçları yöntemiyle yapılan modelin başarı metrikleri. 60

Tablo 4.11: Rastgele orman yöntemiyle yapılan modelin karışıklık matrisi. 62

Tablo 4.12: Rastgele orman yöntemiyle yapılan modelin başarı metrikleri. 62

Tablo 4.13: GBT tabanlı OneVsRest yöntemiyle yapılan modelin karışıklık matrisi. 64

Tablo 4.14: GBT tabanlı OneVsRest yöntemiyle yapılan modelin başarı metrikleri. 64

Tablo 4.15: SVM tabanlı OneVsRest yöntemiyle yapılan modelin karışıklık matrisi. 66

Tablo 4.16: SVM tabanlı OneVsRest yöntemiyle yapılan modelin başarı metrikleri. 66

SİMGE VE KISALTMA LİSTESİ

Simgeler **Açıklama**

P : Olasılık

Kısaltmalar **Açıklama**

API : Application programming interface
CSV : Comma separated values
FN : False negative
FP : False positive
GBT : Gradient-boosted tree
GSM : Global System for Mobile Communications
HDFS : Hadoop distributed file system
HTML : HyperText markup language
IoT : Internet of Things
İBB : İstanbul Büyükşehir Belediyesi
JSON : Java script object notation
MLlib : Machine learning library
NoSQL : Not only structured query language
RAM : Random access memory
RDBMS : Relational database management system
RDD : Resilient distributed dataset
SQL : Structured query language
SVM : Support vector machine
TN : True negative
TP : True positive
XML : Extensible markup language
YARN : Yet another resource negotiator

ÖZET

YÜKSEK LİSANS TEZİ

BÜYÜK VERİ ARAÇLARI İLE COVİD-19 PANDEMİSİNİN İSTANBUL TRAFİĞİNE ETKİSİNİN ANALİZİ

Uğur ALCAN

İstanbul Üniversitesi-Cerrahpaşa

Lisansüstü Eğitim Enstitüsü

Elektrik-Elektronik Mühendisliği Anabilim Dalı

Danışman : Prof. Dr. Fırat KAÇAR

Teknolojinin beraberinde getirdiği internet ile birlikte; insanlar, yaptığı hemen hemen tüm işlerde veri üretir hale geldi. Sıradan bir günde yaptığımız Whatsapp'tan mesaj atma, Instagramda fotoğraf paylaşma, Google'dan arama yapma, e-posta (elektronik posta) gönderme ve bunlar gibi sayamayacağımız birçok faaliyetimizle devasa boyutlarda bir veri kaynağı oluşturmaktayız. Bu kadar yoğun ve farklı veri aynı zamanda bilgi çöplüğüne neden olmaktadır. Bu çöplüğü geleneksel teknolojilerle analiz etmek ise ayrı bir problemdir. Bu bilgi yığınını kendi müşterilerinin davranışlarını analiz edip, elde edilen sonuçlara göre stratejilerini belirlemek isteyen büyük firmalar büyük veri kavramını ortaya çıkardılar. Büyük veri, sosyal medya paylaşımları, sensör verileri, fotoğraf arşivlerimiz, GSM (Global System for Mobile Communications) operatörlerinden elde edilen arama kayıtları, arama motorları istatistikleri, log dosyaları gibi farklı kaynaklardan elde ettiğimiz verilerin anlamlı ve işlenebilir hale dönüştürülmüş biçimidir [1].

Bu çalışmada büyük veri teknolojilerinin gücü kullanılarak günümüzün önemli bir problemi olan Covid-19 pandemisinin İstanbul trafiğine etkisi incelenmiştir. Bu kapsamda İstanbul Büyük Şehir Belediyesi'nin açık olarak yayınladığı 2020 yılına ait saatlik trafik indeksi veri seti [2] ile aynı yıla ait sokağa çıkma kısıtlaması zamanları veri seti ele alınmıştır. Bu veri setlerinin analizinde yeni nesil veri işleme aracı olan Apache Spark kullanılmıştır. Apache Spark ile 2020 yılına ait İstanbul trafik indeksi verilerinin öncelikle genel analizi yapılmış olup, daha sonra elde edilen veriler sokağa çıkma kısıtlaması zamanları veri seti ile ilişkilendirilerek

etki analizi yapılmıştır. İşlenen verilerin tutulmasında Elasticsearch kullanılmış olup, veri görselleştirmesinde ise Kibana kullanılmıştır. Çalışmanın sonunda ise Apache Spark'ın MLlib (Machine learning library) kütüphanesi kullanılarak lojistik regresyon, karar ağaçları, rastgele orman, gradyan yükseltme ağacı tabanlı OneVsRest ve doğrusal destek vektör makinesi tabanlı OneVsRest yöntemleriyle trafik indeksi konusunda makine öğrenmesi uygulamaları geliştirilmiştir. Çalışmanın esas amacı büyük veri algoritmaları ile Covid-19 pandemisinin İstanbul ulaşımına etkisini analiz etmek gibi gözükse de yavaş yavaş akıllı şehirlere dönüşmeye başladığımız bu dönemde, trafik analiz ve tahminin bu sürecin önemli bir parçası olması sebebiyle; yapılan bu çalışma, akıllı şehirlere dönüşme sürecine de hizmet etmektedir.

Ocak 2022, 93 sayfa.

Anahtar kelimeler: Büyük veri, trafik analizi, Apache Spark, Elasticsearch, Kibana

SUMMARY

M.Sc. THESIS

IMPACT ANALYSIS OF COVID-19 PANDEMİC ON ISTANBUL TRAFFIC WITH BIG DATA TOOLS

Uğur ALCAN

Istanbul University-Cerrahpasa

Institute of Graduate Studies

Department of Electrical and Electronic Engineering

Supervisor : Prof. Dr. Fırat KAÇAR

With the internet brought along by technology, people have started to produce data in almost all their jobs. We create a huge data source with many activities we cannot count, such as sending messages on Whatsapp, sharing photos on Instagram, searching from Google, sending electronic mails (email) and these, which we do on an ordinary day. Such dense and different data also leads to information garbage. Analyzing this dump with traditional technologies has been another problem. Big companies that want to analyze this mass of information, analyze the behavior of their customers and determine their strategies according to the results obtained, have come up with the concept of big data. Big data is the form of the data we obtain from different sources such as social media shares, sensor data, photo archives, call records obtained from GSM (Global System for Mobile Communications) operators, and search engine statistics, into a meaningful and processable form [1].

In this study, the effect of the Covid-19 pandemic, which is an important problem of today, on Istanbul traffic has been examined by using the power of big data technologies. In this context, the hourly traffic index of 2020 dataset which has openly been published by Istanbul Metropolitan Municipality [2] and curfew time dataset are discussed. Apache Spark, a new generation data processing tool, has been used in the analysis of these data sets. With Apache Spark, firstly a general analysis of the Istanbul traffic index data for 2020 has been carried out, and then the data obtained have been associated with the curfew time dataset and impact

analysis has been performed. Elasticsearch has been utilized to be able to keep the processed data, and Kibana has been used for data visualization. At the end of the study, machine learning applications on traffic density have been enhanced using Apache Spark's MLlib (Machine learning library) API with logistic regression, decision trees, random forest, gradient-boosted tree-based OneVsRest and linear support vector machine-based OneVsRest methods. Although the main purpose of the study seems to be to analyze the impact of the Covid-19 pandemic on Istanbul transportation with big data algorithms, in this period when ordinary cities are slowly turning into smart cities and since traffic analysis and forecasting are an significant part of this process, this study also contributes to the process of transforming into smart cities.

January 2022, 93 pages.

Keywords: Big data, traffic analysis, Apache Spark, Elasticsearch, Kibana

1. GİRİŞ

COVID-19 virüsü Aralık 2019'da Çin'de ilk ortaya çıkışından bu yana küresel olarak yayılım göstermesiyle, tüm ülkelere benzeri görülmemiş bir etkiye neden olan pandemi haline geldi. Virüsün bulaşma mekanizması başlangıç dönemlerinde büyük ölçüde bilinmediğinden, hükümetler farklı tedbirler alarak pandemiyle mücadele etmeye çalıştılar. Bu tedbirlerin en önemlisi halka çeşitli konularda kısıtlama getirmekti. Dünya nüfusunun çoğunluğu bu kısıtlamalardan etkilenip, günlük yaptıkları aktivitelerin bir çoğunluğunda doğrudan ve ya dolaylı olarak değişime mecbur kaldı. COVID-19 pandemisinin zirve yaptığı dönemlerde ise birçok ülke belirli zaman aralıklarıyla sokağa çıkma kısıtlaması uyguladı. Bu kısıtlamalar zaman içerisinde değişiklik gösterse de ülke içinde seyahatler çeşitli şekillerde devam etti [3].

Bu çalışmada Covid-19 virüsünün ve bu virüsün ilerlemesini yavaşlatıp ölümleri azaltmak için yapılan sokağa çıkma kısıtlamalarının, İstanbul trafiği üzerindeki etkisi incelenmiştir. Çalışmada İBB'den alınan 2020 yılı İstanbul saatlik trafik indeksi [2] ve 2020 yılı sokağa çıkma kısıtlaması zamanları veri setleri kullanılmıştır.

Öncelikle bu veri setlerinde büyük veri işleme araçlarından olan Apache Spark ile 2020 yılı özelinde genel analizler yapılmıştır. Saatlik trafik indeksi veri setinde gün, ay, saat gibi kriterlerde ayrı ayrı 1 senenin analizi yapılıp, NoSQL (Not only structured query language) veri tabanından olan Elasticsearch'e işlenmiş veriler atılmıştır. Buradaki işlenmiş verileri görselleştirmek için ise Kibana kullanılmıştır.

2020 yılının genel trafik indeksi analizi yapıldıktan sonra trafik indeksi veri seti sokağa çıkma kısıtlaması zamanları veri seti ile ilişkilendirilip, birbirilerine göre durumları Kibana ile görselleştirilmiştir. Analiz ve grafik sonuçları detaylı bir şekilde değerlendirilip, hem koranavirüsün hem de beraberinde getirdiği sokağa çıkma kısıtlamalarının İstanbul trafiğinde nasıl bir değişime neden olduğu yorumlanmıştır.

Çalışmanın son kısmında ise Apache Spark'ın API'larından (Application programming interface) olan MLlib kütüphanesi kullanılarak trafik indeksi konusunda lojistik regresyon, karar ağaçları, rastgele orman, gradyan yükseltme ağacı tabanlı OneVsRest ve doğrusal destek vektör makinesi tabanlı OneVsRest yöntemleriyle makine öğrenmesi uygulamaları yapılmıştır.

Bunun için 2020 trafik indeksi verilerinin %70'lik kısmı eğitim verisi, %30'luk kısmı ise test verisi olarak kullanılmıştır. Burada modeli etkileyen parametreler ay, ayın günü, haftanın günü, saat, sokağa çıkma kısıtlamasının ve resmi tatil olup olmaması olarak alınmıştır. Uygulama kapsamında kullanılan yöntemler sonucunda elde edilen doğruluk değerleriyle; yöntemlerin performansı değerlendirilip, çalışma sonlandırılmıştır.

Bu çalışmanın amacı Covid-19 pandemisinin ilk döneminde, insanların yolculuk konusunda eğilimlerinin değişimini incelemenin yanı sıra çalışma, modern şehirlerin yavaş yavaş akıllı şehre dönüşmeye başladığı günümüzde, akıllı şehirlerin önemli bir parçası olan Akıllı Ulaşım Sistemlerine de hizmet etmektedir. Kentleşmenin hızlanması ve kentsel nüfusun hızlı büyümesi, kentsel trafik yönetimine büyük zorluk getirmektedir. Akıllı Ulaşım Sistemi, akıllı şehrin vazgeçilmez bir parçasıdır ve trafik tahmin analizi Akıllı Ulaşım Sistemi'nin önemli bir bileşenidir. Birçok gerçek dünya uygulaması için doğru trafik tahmini gereklidir. Örneğin, trafik akışı tahmini şehrin tıkanıklığının hafifletmesine yardımcı olabilir. Fakat trafik analizi ve tahmini, trafiğin zaman ve mekân ile sürekli değişen karmaşık ve dinamik yapısı olması sebebiyle oldukça zordur. Bu çalışma bu analiz ve tahmin zorluğunun büyük veri ile minimumlara inebileceğini de gösterecektir.

2. GENEL KISIMLAR

2.1. BÜYÜK VERİ

Günümüzde teknolojinin gelişmesiyle birlikte insanlar birçok cihazla etkileşimli hale gelmiştir. Bugün kullanılan akıllı telefonlar, bilgisayarlar vb. cihazlar ile her geçen gün üretilen veri sayısı artmaktadır. Artan bu veriyi geleneksel teknolojilerle analiz etmek ise oldukça zor olmaktadır. Bu zorluk büyük veri kavramını beraberinde getirdi. Sosyal medya paylaşımları, sensörlerden gelen veriler, log dosyaları, IOT (Internet of Things) cihazları gibi kaynaklarla beslenerek gün geçtikçe devasa boyutlara yaklaşan büyük veri; değişik formatlarda olabilen ve birçok kaynaktan elde edilen verilerin anlamlı ve işlenebilir hale getirilmiş biçimidir. Büyük veri, düzensiz yapılı, hızla büyüyen ve yüksek hacimli olan verileri işleyip, anlamlı hale getirerek birçok sektöre değer katar. Kurumlar büyük veri sayesinde müşterilerini daha rahat analiz edip yeni bakış açıları elde ederler. Büyük veri doğru kullanıldığı takdirde birçok kurumun karar mekanizmasında oluşabilecek riskleri en aza indirip stratejik yönden doğru kararlar almasını sağlar [1, 4].

Büyük veri kavramının ortaya çıkmasının temel sebebi teknolojinin beraberinde getirdiği devasa boyutlardaki veriyi saklamada ve işlemede geleneksel teknolojilerin yetersiz kalmasıdır. Günümüzde oluşan birçok veri mevcut veri tabanlarının kapasitesini aşmaktadır. Geçmiş yıllarda veriler disk, CD, DVD gibi depolama araçlarında depolanırken, daha büyük veriler ise yakın zamana kadar elektronik tablolar veya veri tabanlarında tutuluyordu ve hepsi çok düzenliydi. Ancak günümüzde veri çok karmaşık hale geldi. Veri artık ses, video, resim, log, sensör gibi birçok kaynaktan beslenmektedir. Bu karmaşık yapılı verilere karşı ise geleneksel teknolojiler boyun eğmek zorunda kalıyordu. Büyük veri teknolojileri ile bu büyük hacimli ve düzensiz yapılı verileri işleyen sektördeki birçok firma, kar marjlarını arttırmıştır. Büyük verinin 3V ile başlayan ve günümüzde 5V olarak kabul edilen hacim, hız, çeşitlilik, değer ve doğruluk kapsamına giren verileri en anlamlı şekilde işlemesiyle telekomünikasyon, bankacılık, eğitim, kamu, enerji, sağlık, perakende, ulaşım, sosyal medya ve daha birçok alanda büyük veri başköşeye oturmuştur [1, 5].

2.2. BÜYÜK VERİ BİLEŞENLERİ

Genel anlamda büyük veriyi oluşturan 5 bileşenden bahsedebiliriz. Bunlar; çeşitlilik (variety), hacim (volume), hız (velocity), doğruluk (verification) ve değer (value) olarak adlandırılır. Bu bileşenler büyük verinin 5 V'si olarak bilinir.



Şekil 2.1: Büyük verinin 5V'si [6].

2.2.1. Variety (Çeşitlilik)

Variety, büyük veri kaynaklarının değişik formatlarda olabileceğini ifade eder. Hem insanlar hem de makineler tarafından üretilen veriler yapısal, yarı yapısal ya da yapısal olmayan formatta olabilir. Yapısal veriler, bir kayıt veya dosya içindeki sabit bir alanda tutulan herhangi bir veriyi ifade eder. Bu tür veriler ilişkisel veri tabanları ya da elektronik tablolarda tutulur. Fotoğraf, video, ses, blog yazıları gibi kolayca sınıflandırılamayan ve belirli bir formatı olmayan verilere ise yapısal olmayan veri denir. Yarı yapısal veri ise verileri ayırmak için etiket ve ya benzeri işaretler kullanan veri çeşididir. XML (Extensible markup language), HTML (HyperText markup language) ve JSON (Java script object notation) veri kaynakları yarı yapısal verilere örnek olarak gösterilebilir [1, 7].

2.2.2. Volume (Hacim/Büyüklük)

Volume, Gigabayt ile Zettabayt arasında değişen boyutlardaki yapısal ve yapısal olmayan verilerin miktarını açıklamak için kullanılır. Sosyal medyadan sensörlere kadar birçok veri kaynağı her dakika büyük miktarlarda veri üretir. Gün geçtikçe de bu büyük veri daha devasa ölçeklere ulaşmaktadır. Şirketlerin devamlılıklarını sağlama ve rekabet edebilme adına gelecekte bu büyüklükte verilerle nasıl başa çıkacaklarını erkenden planlamaları gerekmektedir.

2.2.3. Velocity (Hız)

Velocity, veri hacminin hızlı bir şekilde artmasını ifade etmektedir. İnternette dakikada üretilen veri miktarı hayal edilemeyecek noktalara ulaştı. Örneğin; 2020 yılında Google'da dakikada 4.1

milyon arama yapılmış, Youtube'da dakikada 4.7 milyon video izlenmiş, Instagramda dakikada yaklaşık 695 bin kaydırma (scroll) yapılmış, Whatsapp ve Messenger'da dakikada 59 milyon mesaj gönderilmiştir [8].



Şekil 2.2: 2020 yılında her 1 dakikada internette veri kullanımı [8].

2.2.4. Verification (Doğruluk)

Verification, gürültü, eksiklik, gecikme gibi nedenlerden dolayı büyük verinin doğruluğundan kesin emin olunması gerektiğini ifade eder. Örneğin sensör verilerine gürültü binmesi ya da erişilen her herhangi bir web adresinin yanlış bilgiler içermesi bizim de yanlış bilgiler elde edip, yanlış kararlar almamıza sebep olur. Bundan dolayı kuruluşlar hem verinin doğruluğundan hem de verinin doğru bir şekilde analiz edildiğinden emin olmalıdır.

2.2.5. Value (Değer)

Value, büyük verinin kuruluşlara değer katmasını ifade eder. Veriler işlenip, kurumlar ya da kişiler için anlamlı hale getirilmediği sürece hiçbir fayda sağlamaz. Bundan dolayı value, büyük verinin en önemli bileşenidir.

2.3. BÜYÜK VERİNİN KULLANIM ALANLARI

Firmalar için müşteri profillerini anlayıp, bu profillere göre müşterilerine önerilerde bulunmak, müşteri sadakatini arttırmak açısından oldukça kilit rol oynamaktadır. Bu durum sadık müşteri kitlesi oluşturmak açısından önem arz etmektedir. Büyük veri bu konuda firmalara yol gösterici olarak karşımıza çıkmaktadır. Büyük veri ile firmalar, müşterilerini kişisel profillerine kadar

analiz edip onların ihtiyaçlarına göre satış stratejileri yapabilmektedir. Firmalar açısından risk analizi de oldukça önemlidir. Büyük veri, firmalara risk azaltma stratejilerinde de yol gösterici olmaktadır.

Büyük veri, müşteri davranışları analizinin önem arz ettiği hemen hemen tüm sektörde kullanılmaktadır. Özellikle bankacılık sektörü büyük veri ile büyük kazançlar elde etmiştir. Son dönemlerde büyük verinin gücü hükümetlerce de anlanıp, devlet hizmetlerinde kullanılmaya başlanmıştır.

Büyük veriyi kullanan başlıca sektörler ve ilgili sektördeki kullanım alanları aşağıdaki şekilde sıralanabilir.

Hastaneler; hastalar, ilaçlar, hastalıklar ve bunların belirtileri ile ilgili sahip oldukları büyük verileri kullanarak detaylı hastalık analizleri yapmada, hastalıklara dair yeni tespitler elde etmede büyük veriden faydalanmaktadır. Büyük veri ile hastalıkların erken teşhis imkânı olabilmektedir [9].

Hükümetler; tehditleri tahmin edip önlem alma, vergi kaçakçılığı ve sahtecilik tespiti, suç tahmin edip önleme gibi birçok konuda büyük veriden faydalanabilmektedir.

Ulaşım alanında; akıllı ulaşıma geçişin başladığı şu zamanlarda büyük veri önem arz etmektedir. Trafiği kontrol etmek, en iyi rotayı belirlemek ya da trafik koşullarını tahmin edip yakıt ve zamandan tasarruf etmede büyük veriden yararlanılır [1].

Bankalar; kredi ve banka kartlarının kötüye kullanımının tespiti, kredi risk tespiti, müşteri istatistiklerindeki değişiklikleri analiz edip müşterileri modelleme gibi birçok konuda büyük veriden faydalanmaktadır [10].

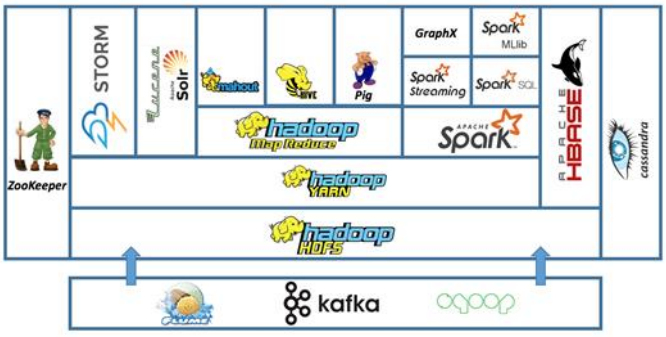
Enerji firmaları; daha iyi kaynak yönetimi sağlama, problemleri önceden tespit etme, müşterilerinin tüketim bilgilerini daha etkin şekilde kontrol etmede, şebeke ve sayaçlarda akıllı cihazlar kullanılarak elde edilen büyük verilerin analiz edilmesinden faydalanırlar.

Perakende şirketleri; büyük veri analiz yöntemlerini kullanarak müşterilerini daha rahat analiz edip; kampanyalarını, reklamlarını hatta web sitelerini müşteri özelinde düzenleme imkânı buldular.

2.4. BÜYÜK VERİ ARAÇLARI

Günümüzün madeni veri, etkili şekilde analiz edildiği takdirde kurum ve kuruluşlara büyük faydalar sağlayabilmektedir. Bugün ürettiğimiz verilerden en anlamlı sonuçlar elde etmek için büyük veri araçları kullanılmaktadır. Büyük veri dünyası, şirketlerin birçok değişik ihtiyaçlarına cevap verebilecek araçlara sahiptir.

Büyük veri gücünü dağıtık çalışma mimarisinden alır. Birçok sunucunun kaynakları bir araya gelip, tek bir sunucu gibi davranırlar. Gelen her veri sunucular üzerinde bloklar halinde dağıtılır, verinin analiz edilme durumunda her sunucu bu analizde görev alır. Bundan dolayı sistem performansı doğrudan sunucu sayısıyla ilişkilidir. Büyük veri mimarisinde sunucu sayısı istenildiği gibi arttırılıp, azaltılabilir yani sistem ölçeklenebilirdir. Büyük veri mimarisinin diğer bir önemli özelliği ise sistemin yüksek erişilebilir olmasıdır. Bu durum cluster'da yer alan sunuculardan bir ve bir kaçında problem olması durumunda, kullanıcıların bu problem veya problemlerden dolayı kesinti yaşamamasıdır. Büyük veri mimarisinin temelinde Hadoop bulunmaktadır.



Şekil 2.3: Hadoop ekosistemi [11].

Büyük veri teknolojileri aşağıdaki şekilde listelenebilir.

2.4.1. Apache Hadoop

Hadoop, büyük verileri paralel olarak saklayıp, işlemeyi sağlayan açık kaynak kodlu ve Java tabanlı bir platformdur. Hadoop mimarisi sayesinde büyük veriler cluster denilen bilgisayar kümelerince bölüştürülüp veri analizleri kısa sürede halledilebilir. Ölçeklenebilir olması, hatalara karşı dayanaklı olması, marka bağımsız olup özel donanım istememesi sayesinde düşük

maliyetli olması Hadoop'un öne çıkan özelliklerindendir. Apache Hadoop 4 temel bileşenden oluşur [12].

2.4.1.1. Hdfs (Hadoop Distributed File System)

HDFS (Hadoop Distributed File System) Hadoop'un dağıtık yazılım tabanlı depolama katmanıdır. HDFS, hataya karşı son derece dayanıklıdır ve düşük maliyetli donanımlara dağıtılmak üzere tasarlanmıştır. Hadoop cluster'ında; HDFS, datanode ve namenode olmak üzere iki temel bileşenden oluşur. Datanode verileri tutar, namenode ise tutulan bu verilerin metadata bilgilerinden sorumludur.

2.4.1.2. Yarn (Yet Another Resource Negotiator)

YARN, hadoop mimarisi gibi dağıtık sistemler için kaynak (cpu, ram) yönetimini sağlar. Hadoop cluster'ında; yarn, Resource manager ve node manager olmak üzere iki temel bileşenden oluşur. Node manager her bir node için durum bilgisini resource manager'a iletir, resource manager ise bu durum bilgisine göre kaynak planlamasını sağlar.

2.4.1.3. MapReduce

MapReduce, Hadoop mimarisinin büyük verileri işleyen kısmıdır. Verileri işlerken map ve reduce adı verilen iki aşamadan geçirir. Map verileri analiz edip, filtrelediği; reduce ise her bir sunucuda elde edilen analiz sonuçlarının toplandığı aşamadır.

2.4.1.4. Hadoop Common

Tüm Hadoop modüllerini destekleyen kütüphanedir.

2.4.2. Apache Kafka

Apache Kafka, bir mesajlaşma kuyruk sistemidir. Yani veriyi sıralı bir şekilde saklayıp, değiştirmeden olduğu gibi iletimini sağlar. Apache Kafka ilk olarak Linkedin tarafından geliştirilmiş daha sonrasında Apache bünyesine katılmıştır.

Apache kafka mimarisinde; veriyi kafkaya gönderen bileşene producer, verilerin depolandığı bileşene topic ve verileri okuyan bileşene ise consumer denmektedir [13].

2.4.3. Apache Storm

Apache Storm, dağıtılmış gerçek zamanlı akış işleme sistemidir. Apache Storm, Hadoop'un durağan veri için yaptığı veri işlemeyi akan veriler üzerinden yapmaktadır. Storm herhangi bir

programlama dili ile kullanılabilir. Storm, özellikle yüksek veri hızı ile dağıtılmış makine öğrenimi, gerçek zamanlı analiz, ETL ve diğer birçok durum için kullanılabilir. Apache Storm oldukça hızlı, ölçeklenebilir ve hataya karşı dayanaklı olup, kurulumu ve çalıştırması basittir. Storm bugün kullanılan bir çok veritabanı teknolojisiyle entegre olabilmektedir. Günümüzde gerçek zamanlı veri işlemede, Apache Flink ve Spark Streaming Storm'a göre daha fazla tercih edilmektedir [14].

2.4.4. Apache Zookeeper

ZooKeeper, dağıtık yapıda olan servislerin metadata bilgilerini tutup, koordineli şekilde çalıştırılması için geliştirilmiş bir servistir.

2.4.5. Apache Spark

Apache Spark yeni nesil veri işleme motorudur. Dağıtık çalışan birden fazla adımlı işlerde; veri transferini bellek içi sağlaması sebebiyle yüksek performanslı veri işlemeye olanak vermektedir. Apache Spark; Scala, Python, Java ve R dilleri ile geliştirilebilmektedir. Kendi yapısında bir çok kütüphane bulundurmaktadır. Örnek olarak makine öğrenmesi yöntemleri için MLlib, SQL (structured query language) sorguları için Spark SQL, graf ve ağ analizi geliştirmeleri için GraphX, gerçek zamanlı veri akışı için Spark Streaming kütüphaneleri gösterilebilir.

2.4.6. Apache Nifi

NiFi, farklı veri kaynakları ve sistemleri arasındaki veri hareketini otomatikleştiren, verilerin gerçek zamanlı olarak izlenmesini sağlayabilen bir veri akış platformudur. Apache NIFI birçok kaynak ve hedef arasında veri aktarımını desteklemektedir.

2.4.7. Apache Pig

Apache Pig, HDFS'de bulunan büyük veri setlerini analiz etmek için kullanılan platform ve map reduce için bir derleyicidir. Pig, Hadoop kullanan veri bilimcileri için Yahoo tarafından geliştirildi. Pig sayesinde veri bilimcileri mapreduce işleri yazmanın zorluğundan kurtulup, daha çok veri analizlerine odaklanma fırsatı buldular. Apache Pig ile Java, Python gibi dillerde komut yazmak da mümkündür. Günümüzde önemini yitirmiştir [15].

2.4.8. Apache Hive

SQL'e çok benzer bir dil ile Hadoop sistemlerinde bulunan büyük verilerin okunmasını, yazılmasını ve yönetilmesini sağlar. Durağan veriler üzerinde sorgulama imkânı tanımakta olup, Apache Spark ve MapReduce ile entegreli çalışabilir.

2.4.9. Apache Cassandra

Apache Cassandra, wide column tipinde, açık kaynaklı, Java ile geliştirilmiş bir NoSQL veritabanıdır. Wide column tipindeki NoSQL veritabanları, mantıksal olarak ilişkisel veri tabanlarına en çok benzeyen NoSQL veritabanlarıdır [16].

2.4.10. Apache Flume

Log tarzında yapısal olmayan verilerin etkili bir şekilde toplanması, birleştirilmesi ve HDFS'e taşınması için kullanılan bir araçtır.

2.4.11. Apache HBase

Apache HBase, açık kaynaklı bir NoSQL veritabanıdır. Petabaytlarca veriye yüksek tutarlı ve gerçek zamanlı erişim sağlar. Genel olarak belirli büyüklükteki verilerin, ilişkisel veritabanlarına benzer mantıkla tutulması istenen yerlerde Hbase kullanılır. HBase, tablolarındaki veriye SQL benzeri sorguları atmaya imkan sağlamak için Apache Phoenix ile birlikte çalışır [17].

2.4.12. Apache Flink

Apache Flink, çekirdeği Java ve Scala dili ile yazılmış bir veri akış motorudur. Flink hem durağan hem de gerçek zamanlı veri işlemeyi etkili şekilde yapabilmektedir. Flink de Spark gibi kendi makine öğrenmesi kütüphanesine ve SQL sorgusu kütüphanesine sahiptir.

2.4.13. Sqoop

Hadoop ile ilişkisel veritabanları gibi yapılandırılmış veri içeren depolama birimleri arasında veri aktarımını sağlayan bir araçtır. İsmini "SQL to Hadoop" kısaltmasından almıştır.

2.4.14. Apache Solr

Temeli indeksleme mekanizmasına dayanan, Lucene arama kütüphanesine kurulmuş açık kaynaklı kodlu arama platformudur. Java programlama dili ile geliştirilmiştir.

2.4.15. Elasticsearch

Elasticsearch; Java ile geliştirilmiş, genellikle metin arama işlemlerinde kullanılan açık kaynaklı bir NoSQL veri tabanıdır. Veriler Elasticsearch'de json formatında tutulur. Elasticsearch çoğu zaman tek başına tercih edilmeyip, Elastic Stack (ELK) denilen yazılımların bir araya gelmesiyle kullanılır. ELK; Elasticsearch, Logstash ve Kibana yazılımlarının baş harflerinin kısaltmasından gelmektedir. ELK ile yapılan projelerde Elasticsearch; verileri tutan ve arama işlemlerinin yapılmasını sağlayan, Logstash ise verileri toplamayı, işlemeyi ve taşınmasını sağlayan bileşendir. Kibana da işlenen verinin görselleştirilmesinde kullanılır.

2.4.16. Apache Mahout

Makine öğrenimi algoritmalarında kullanılmak için tasarlanmış fakat beklenilen ilgiyi görmemiştir. Günümüzde büyük veri alanında Apache Spark Mahout'a göre makine öğrenimi için daha çok tercih edilmektedir.

2.5. LİTERATÜR ÖZETİ

Gerek ülkemizde gerekse dünyada trafik problemi ile ilgili birçok çalışma yapılmıştır ve yapılmaya da devam etmektedir. Bu çalışmalar günümüzde daha da önem arz etmekte olup, trafik problemine bu çalışmalarla çözüm aranmaktadır. Trafik birçok parametreye bağlı olmakla birlikte bugüne kadar bu konuda yapılan çalışmaların; sıcaklık, nem, rüzgâr gibi hava koşulları ve gün, saat gibi zaman dilimli parametrelere ağırlık verdiği görülmektedir.

Trafik durumuna; ağırlıklı olarak hava koşulları etkisi dikkate alınarak yapılan çalışmalara bakıldığında, Datla ve Sharma [18] soğuk hava koşullarının trafik üzerindeki etkisinin zaman dilimi (gün, saat), yolun türü ve soğuğun şiddetine göre farklılık gösterdiğini belirtmişlerdir. Cools ve diğ. [19] rüzgâr hızı, bulutluluk ve yağışın trafik yoğunluğunu azalttığını, yüksek sıcaklığın ise tam tersi bir etki gösterip, trafik yoğunluğunu arttırdığını belirtip, hava koşullarının gelecekteki trafik güvenliği araştırmalarında yer alması gerektiğinin altını çizmiştir. Yan ve diğ. [20] sisli koşulların sürücülerin hızlarına etkisini inceleyip, sürücülerin sisli havalardaki sürüş riskini en aza indirmek için hızlarını düşürdüklerini tespit etmiştir.

Trafik tahmini konusunda yapılan çalışmalara bakıldığında daha çok derin öğrenme teknikleri ile modeller oluşturulduğu göze çarpmaktadır. Aşağıda bu kapsamda yapılmış birkaç örnek verilmiştir.

Zhu ve diğ. [21] evrişimli sinir ağı mimarisini kullanarak, trafik sıkışıklığını otomatik algılayan bir model tasarlamışlardır. Peppa ve diğ. [22] düşük kaliteli kamera sistemi görüntüleri kullanarak değişik mevsimlerde trafik düzenini tespit eden gene evrişimli sinir ağı temelli bir model tasarlamışlardır. Polson ve Sokolov [23] ise sokaktaki sensör verilerini kullanarak derin öğrenme teknikleriyle trafik akışının tahmini için model oluşturmuştur. Raj ve diğ. [24] trafik tahmini ve analizi için veri madenciliği tekniklerini (K en yakın komşu, yapay sinir ağı) kullanmıştır. Taş ve Sezen [25] ise gün, saat ve hava koşullarına göre yapay sinir ağları yardımıyla araçların hızlarını tahmin edip, trafik durumunu belirlemeye çalışmışlardır. Özdağ [26], İngiltere'deki bir otoyol ve bu yola bağlı birkaç kavşaktan aldığı trafik verileri ile derin öğrenme teknikli ağırlıklı, trafik analizi ve araç sayısı tahmini gerçekleştirmiştir.

Bu tez çalışmasındaki trafik analiz ve tahminlerinde önemli bir parametre olan Covid-19'un ulaşıma etkisi ile ilgili de çeşitli çalışmalar yapılmıştır. Bu çalışmalarda da beklenildiği gibi birçok ülke ve şehirde genel olarak araç kullanımının düştüğü sonucuna varılmıştır. Beck ve Hensher Avustralya da Mart 2020 sonu ve Nisan 2020 ortasında yaptığı çalışmada [27] Covid-19 ile birlikte, araç kullanımında %35 düşüş yaşandığını belirtti. Mayıs sonu haziran ortası yaptığı çalışmada [28] ise ulaşımın artmaya başladığını fakat hala Covid-19 öncesine göre ciddi derecede düşük seviyelerde olduğunu belirtmiştir. Shakibaei ve diğ. [29] Covid-19'un İstanbulluların seyahat davranışlarına etkisini virüsün Çin'de yeni yeni görülmeye başladığı ilk dönem, virüsün komşu ülkelerimizde görüldüğü ikinci dönem ve virüsün İstanbul'da görüldüğünün açıklandığı üçüncü dönem olmak üzere 3 fazda incelemiştir. İkinci ve üçüncü fazda genel olarak toplu ulaşım yerine özel araç kullanmaya başlayanların sayısında artış tespit etmişlerdir. Birçok ülke Covid-19'un bu ve bunlara benzer çalışmalar ile etkilerini incelese de, sonuçlar genel olarak benzerlik göstermiştir.

Bu çalışmalar dışında ülkemiz içinde büyük veri konulu yapılan literatür taramasında, çalışmaların daha çok çeşitli yöntemlerle makine öğrenmesi uygulamaları, mapreduce ve Apache Spark ile veri analizlerine dayandığı tespit edilmiştir. Bu tez çalışmasındaki kapsama konu olarak benzerlik göstermese de yapılan analiz ve makine öğrenmesi uygulamalarına benzerlik gösteren çalışmalar aşağıda açıklanmıştır.

Küçükayan [30] yüksek lisans tez çalışmasında büyük veri yöntemleriyle trafik kazaları, trafik sıkışıklıkları, yol yapım çalışmaları, araç bozulması gibi durumları tespit edip, erken müdahale ile oluşabilecek durumların önüne geçmeyi amaçlamıştır. Tez çalışmasında pilot şehir olarak

İstanbulu seçmiş olup, verilerin işlenmesinde Apache Spark kullanmıştır. Bu çalışmada kullanılan yapay sinir ağları ile yaklaşık %97 doğruluk oranıyla, oranların tespit edilebileceğini saptamıştır.

Can [31] yüksek lisans tez çalışmasında Türkiye Cumhuriyeti Resmi gazete ve Türkiye Radyo Televizyon haber sitesinde yayımlanan haber arşivlerindeki kelimeleri Apache Spark ile analiz edip, kullanılan iki veri setinin benzerlik ilişkisi tespitini yapmıştır.

Keskin [32] yüksek lisans tez çalışmasında bulut servisi üzerinde büyük veri araçları ile veri analizi, veri görselleştirmesi ve makine öğrenmesi uygulamalarını gerçekleştirmiştir. Amazon Web servisi Elastic Map reduce, Apache Hive, Apache Hadoop, Apache Spark ve R Studio araçlarını tez kapsamında kullanmıştır.

Zaralı [33] yüksek lisans tezi kapsamında Apache Spark ile akıllı çiftliklerdeki alarmlardan elde ettiği verileri analiz edip, bu analiz sonucunda alarmlar arasındaki temel ilişki ve örüntüleri bulmuştur.

Karabay [34] yüksek lisans tezi kapsamında dünyada yaşanmış terör olaylarını içeren veri kümesinde Apache Spark ile analiz ve sınıflandırma işlemlerini yapmıştır. Yaşanan terör olaylarının hangi gruba ait olduğunu tahmin eden bir de makine uygulaması gerçekleştirmiştir. Makine uygulamasında K en yakın komşu, Naive Bayes, Rastgele orman, karar ağacı, destek vektör makineleri ve lojistik regresyon yöntemlerini kullanmıştır.

Bayraktar [35] yüksek lisans tezi kapsamında en büyük piyasa değerine sahip 3 kripto para (Bitcoin, Ethereum, XRP) için K en yakın komşu, Destek vektör regresyonu ve rastgele orman yöntemleriyle fiyat tahmini için makine uygulaması geliştirmiştir.

Bugüne kadar yapılan çalışmalarda, trafik tahmin ve analizlerinde derin öğrenme yöntemleri ağır basmaktadır, bu çalışmada makine öğrenmesi uygulamaları kullanılmıştır. İlave olarak incelenen benzer çalışmalardan farklı olarak, Apache Spark ile yapılan analiz ve makine öğrenmesi geliştirmeleri Intellij IDEA gibi bir geliştirme ortamında yapılmış ve programlama için de Java dili kullanılmıştır. Aynı zamanda işlenen verilerin Elasticsearch'e atılması ve bu verilerin görselleştirilmesinde Kibana kullanılması da farklar arasındadır.

3. MALZEME VE YÖNTEM

Günümüzün madeni olan veri, etkili kullanıldığı takdirde ilgili kurum ya da kişileri sektörde öne çıkarabilmektedir. Fakat teknolojinin gelişmesiyle birlikte günümüzde veriler birçok kaynaktan farklı formatlarda gelebilmektedir. Bundan dolayı bu verileri depolamak ve analiz etmek geleneksel teknolojilerle oldukça zor olmaktadır. Bu zorluk ise büyük veri kavramını beraberinde getirdi. Büyük veri teknolojileri ile bu zorluk minimize edilmiş, birçok büyük firma en büyük yatırımlarını büyük veri alanında yapmıştır. Günümüzde ise birçok alanda büyük veri kullanılmaya başlanılmıştır.

Bu çalışmanın amacı büyük verinin veri işlemedeki gücü kullanılarak 2020 yılı için İstanbul'un trafik analizini yapmaktır. Bu analiz çalışmasında COVID-19 pandemisinin ve COVID-19 kaynaklı sokağa çıkma yasaklarının İstanbul trafiğine etkileri de tespit edilmiştir. Çalışma sonunda ise 5 farklı yöntemle trafik durumu tahmini yapılmıştır. Veriler Apache Spark ile işlenip, Kibana ile görselleştirmesi yapılmıştır. İşlenmiş verilerin tutulmasında ise Elasticsearch kullanılmıştır. Trafik tahmini bölümünde ise Apache Spark'ın MLlib kütüphanesi kullanmıştır. Kod geliştirme ortamı olarak Intellij IDEA tercih edilmiştir. Günümüzde nüfusun artması ve maddi olanakların iyileşmesiyle birlikte; birçok kişinin düşük ya da yüksek bütçeli bir arabaya sahip olma isteğiyle, her geçen gün büyüyen trafik sorununa çare olarak görülen akıllı ulaşım sistemlerine geçtiğimiz bu zamanlarda, trafik analiz ve tahminleri bu kapsamda önem arz etmektedir. Tez çalışması bu alana da hizmet etmektedir.

3.1. VERİ SETİ

Bu çalışma için kullanılacak olan 2020 yılı İstanbul trafik indeksi veri seti İBB'nin açık veri olarak yayınladığı "https://data.ibb.gov.tr/dataset/trafik-indeks-degeri-web-servisi" adresinden alınmıştır. Veri, 1'er saatlik ortalama İstanbul trafik indeksi değerlerini içermektedir. Trafik indeksi 1-99 arasında değişen değerler almaktadır. İndeks değeri büyüdükçe trafiğin yoğunlaştığı anlamına gelmektedir. Bundan dolayı, 1 açık trafik, 99 çok yoğun trafik olduğunu ifade etmektedir. Trafik indeksi veri setinin ham hali XML formatında olup, trafik indeks tarihi ve o tarihteki 1'er saatlik ortalama trafik indeksi değerini içermektedir. Veri ön işleme aşamasında veri setine sokağa çıkma yasağının ve resmi tatil olup olmaması nitelikleri eklenip,

veri formatı csv'e (comma separated values) çevrilmiştir. Veri setindeki niteliklerin açıklamaları tablo 3.1'de verilmiştir.

Tablo 3.1: Düzenlenmiş veri setinin nitelikleri ve açıklamaları.

Nitelik	Açıklama
TrafikIndeksTarihi	Tarih bilgisini içerir. YYYY-MM-DD HH:mm:ss formatındadır.
SokagaCikmaYasagi	İlgili tarihte sokağa çıkma yasağının olup olmadığını belirtir. 1 sokağa çıkma yasağının olduğunu, 0 ise olmadığını belirtir.
ResmiTatil	İlgili tarihte resmi tatil olup olmadığını belirtir. 1 resmi tatil olduğunu, 0 ise olmadığını belirtir.
TrafikIndeksi	İlgili tarihe ait ortalama trafik indeksini içerir. 1-99 arasında değer alır.

Veri setinde toplamda 2020 yılındaki 366 güne ait trafik indeksi değerleri bulunmaktadır. Saatlik bazda toplam 52 saate ait trafik indeksi değeri veri setinde eksiktir. Bu eksik değerlerin çalışmayı etkilememesi adına ilgili saatler, dolu verilerin ay, gün, saat, sokağa çıkma kısıtlamasının ve resmi tatil olup olmasına göre gruplandırılıp, ilgili zaman dilimlerinin trafik indeksi değerlerinin ortalaması ile doldurulmuştur. Doldurulmuş saatlik ortalama trafik indeksi veri setinin Şekil 3.1'de örnek bir kısmı, tablo 3.2'de ise trafik indeksi değerlerinin özeti verilmiştir.

1	TrafikIndeksTarihi	SokagaCikmaYasagi	ResmiTatil	TrafikIndeksi
2	2020-12-31 23:00:00	1	0	2.0
3	2020-12-31 22:00:00	1	0	2.0
4	2020-12-31 21:00:00	1	0	3.0
5	2020-12-31 20:00:00	0	0	16.0
6	2020-12-31 19:00:00	0	0	36.0
7	2020-12-31 18:00:00	0	0	54.0
8	2020-12-31 17:00:00	0	0	60.0
9	2020-12-31 16:00:00	0	0	61.0
10	2020-12-31 15:00:00	0	0	59.0
11	2020-12-31 14:00:00	0	0	54.0
12	2020-12-31 13:00:00	0	0	47.0
13	2020-12-31 12:00:00	0	0	40.0
14	2020-12-31 11:00:00	0	0	33.0
15	2020-12-31 10:00:00	0	0	30.0
16	2020-12-31 09:00:00	0	0	33.0
17	2020-12-31 08:00:00	0	0	31.0
18	2020-12-31 07:00:00	0	0	22.0
19	2020-12-31 06:00:00	0	0	10.0
20	2020-12-31 05:00:00	0	0	6.0

Şekil 3.1: Düzenlenmiş veri setinden örnek.

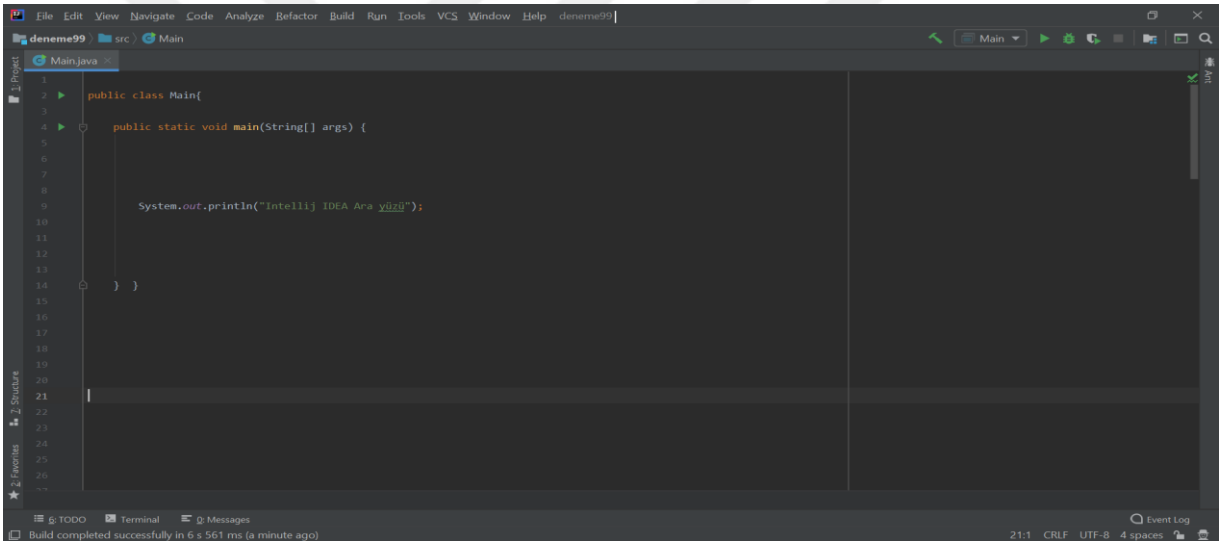
Tablo 3.2: Trafik indeksi ile ilgili özet durum.

Özet	Trafik Indeksi
Sayı	8784
Ortalama	23.61
Standart Sapma	17.93
Minimum	1
Maksimum	83

3.2. ÇALIŞMADA KULLANILAN YÖNTEM VE ARAÇLAR

3.2.1. Intellij Idea

Intellij idea, JetBrains şirketi tarafından piyasaya sürülmüş yazılım geliştirme ortamıdır. JVM dilleri (java, kotlin, scala ve groovy) için tasarlanmasına rağmen, çeşitli eklentilerle birlikte Python, Ruby ve Javascript gibi birçok dili de desteklemektedir [36]. Ücretli ve ücretsiz sürümleri bulunmaktadır. Intellij IDEA kendine ait birçok özelliği ve kolay ara yüzüyle oldukça öne çıkmaktadır. Kullanımı her geçen gün artmakta olup, Google bile Android Studio'yu Intellij IDEA üzerinde geliştirmiştir. Şekil 3.2'de Intellij IDEA'nın ara yüzü gösterilmiştir. Bu çalışmada açık kaynağa dayalı community edition sürümü kullanılıp, geliştirmeler Java'yla yapılmıştır.



Şekil 3.2: Intellij IDEA ara yüzü.

3.2.2. Java

Java, nesne yönelimli uygulamalar geliştirmek için Sun Microsytems tarafından geliştirilmiş sınıf tabanlı, platform bağımsız ve anlaşılması basit dillerden birisidir. Java'nın platform bağımsız olması Java ile yazılan uygulamaların yazıldığı ortam dışında da çalıştırılabileceği anlamına gelmektedir. Bu durum Java'nın aynı zamanda platform olmasından kaynaklanmaktadır. Java temelde üç bileşenden oluşur [37].

JVM (Java Virtual Machine), Java'da yazılan uygulamaların çalışması için gerekli olan sanal makinedir. Derleyici tarafından üretilen bayt kodunu işletim sistemlerine göre makine diline çevirmek temel görevidir. Java'nın platform bağımsız olmasını sağlayan bileşenidir [37, 38].

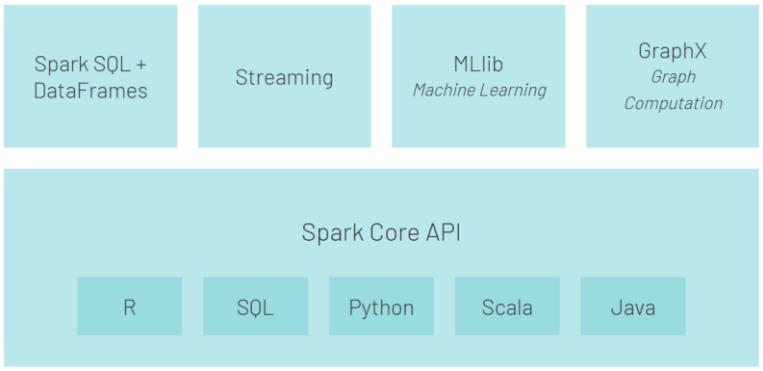
JRE (Java Runtime Environment), Java uygulamasını çalıştırmak için gerekli alt yapıyı barındıran yapıdır. JRE içerisinde, JVM'de bulunmaktadır.

JDK (Java Devolepment Kit), içerisinde Java ile geliştirme yapmak için bütün gereçleri içeren geliştirme paketidir. İçerisinde jre, jvm, derleyici ve geliştirme için gerekli kütüphaneleri içerir. Java uygulamalarını çalıştırmak için JDK zorunlu değildir, jre uygulamaların çalışması için yeterlidir. Fakat bir geliştirici Java diliyle uygulama yapmak istiyorsa, jre bu durumda yetersiz kalıp, jdk yüklenmesi gerekmektedir [37, 38].

3.2.3. Apache Spark

Apache Spark; büyük veriler üzerinde bellek içi işlem yapabilen, yeni nesil bir veri işleme motorudur. Apache Spark Scala ile geliştirilmesine rağmen, Java, Python gibi dillerle de geliştirme yapmaya imkân vermektedir. Spark'ı öne çıkaran en önemli özellik bellek içi çalışmasından kaynaklanan veri işlemedeki hızıdır. Hatta bu özellik Databricks sitesinde de [39] öne çıkarılmış ve hadoop'dan 100 kat daha hızlı olduklarını belirtmişlerdir. Apache Spark Hadoop mimarisindeki hdfs gibi bir depolama birimine sahip değildir. Spark işleyeceği veriyi hdfs, kafka ya da rdbms gibi yerlerden alıp, veride istenilen işlemleri yaptıktan sonra gene benzer yerlere yazabilmektedir.

Apache Spark yapısında Spark SQL, Spark Core, Spark Streaming, MLlib ve Graph X olmak üzeri 5 farklı bileşen bulundurur [39, 40].



Şekil 3.3: Apache Spark bileşenleri [39].

3.2.3.1. Spark Core

Spark Core, sparkın tüm bileşenlerinin temelini oluşturan kısımdır. Spark işlerindeki bellek yönetimi, hata yönetimi, görev dağılımları gibi faaliyetlerden sorumludur. Spark'ın en temel veri kümesi yapısı olan Resillient Distrubuted Dataset (RDD) Spark Core'da bulunur. RDD'de tutulan veriyi değiştirmek için yeni bir RDD oluşturmak gereklidir çünkü RDD'ler üzerinde işlem yapılabilir fakat RDD'ler veriyi değiştirmeye izin vermez [41, 42].

RDD'ler üzerinde yapılan işlemler transformation ve action metodları olarak ayrılır. Transformation metodları RDD'den yeni bir RDD'in elde edildiği yani RDD'lerde dönüştürmenin yapıldığı metodlardır. Tablo 3.3'de bazı transformation metodları ve açıklamaları verilmiştir [41].

Tablo 3.3: Apache Spark örnek transformation metodları ve açıklamaları.

Fonksiyon	Açıklama
map()	Her bir elemana uygulanan fonksiyona göre yeni bir RDD oluşturur.
filter()	Her bir elemanı verilen koşula göre kontrol edip, koşulu sağlayanlardan yeni bir RDD yapar.
flatMap()	Map fonksiyonuna benzer fakat map fonksiyonun yaptığı işi bir eleman için ayrı ayrı yapar.
sample()	Örnek RDD döndürür.
union()	İki RDD'yi birleştirip, yeni bir RDD döndürür.

Tablo 3.3 (devam): Apache Spark örnek transformation metodları ve açıklamaları.

intersection()	İki RDD'nin kesişimini, yeni bir RDD olarak döndürür.
distinct()	RDD'nin farklı elementlerini, yeni bir RDD olarak döndürür.
mapPartitions()	Map fonksiyonun yaptığı işi, her bir partition için yapar.
groupByKey()	Anahtar değer çiftli bir veri setinde, anahtara göre gruplama işini yapar.
coalasce()	RDD'deki partition sayısını azaltmak için kullanılır.

Action metodları ise RDD'de gerçekleşen transformation işlemlerini sonuca dönüştüren metodlardır. Action metodları olmadan transformation metodları çalışmamaktadır. Tablo 3.4'de bazı action metodları ve açıklamaları verilmiştir [41].

Tablo 3.4: Apache Spark örnek action metodları ve açıklamaları.

Fonksiyon	Açıklama
count()	RDD'deki eleman sayısını verir.
first()	RDD'deki ilk elemanı döndürür.
take()	RDD'den belirtilen sayı kadar elemanı liste şeklinde döndür.
foreach()	RDD'deki her bir elaman üzerinde istenilen fonksiyonu çalıştırır.
takeSample()	RDD içerisinden istenilen sayı kadar rastgele elemanı liste şeklinde döndürür.
collect()	RDD'deki tüm elemanları dizi olarak döndürür.
reduce()	Uygulanan fonksiyona göre, RDD elemanlarında eş zamanlı işlem uygular.

RDD'de uygulamanın konfigürasyonlarını yapmak için SparkConf yapısı kullanılır. SparkContext yapısı ise RDD'ye giriş kapısıdır. Şekil 3.4'de C dizini altındaki "çalışanlar" dosyasını okuyup, ekrana yazdıran JavaRDD örneği gösterilmiştir.

```
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.api.java.JavaSparkContext;

public class Deneme {
public static void main(String[] args) {

   SparkConf sparkConf = new SparkConf().setMaster("local").setAppName("Spark RDD");
   JavaSparkContext sparkContext = new JavaSparkContext(sparkConf);
   JavaRDD<String> stringRDD = sparkContext.textFile( path: "C:/çalışanlar.txt");
   stringRDD.foreach(line-> System.out.println(line));

}
}
```

Şekil 3.4: JavaRDD örneği.

3.2.3.2. Spark SQL

Spark SQL, Spark'ın yapılandırılmış veriler üzerine odaklanan kütüphanesidir. Spark SQL json, csv gibi yarı yapısal formattaki verileri DataFrame ya da Dataset halinde yapısal hale getirip, üzerlerinde kendisine özgü SQL sorguları atmaya imkân vermektedir. Dataset ve DataFrame de RDD gibi dağıtılmış veri kümeleridir. RDD'den en önemli farkları ise Dataset ve DataFrame yapılarının klasik ilişkisel veri tabanına benzer şekilde satır ve sütun halinde düzenlenmiş olmalarıdır. Dataset ve DataFrame yapılarının çıkma sebeplerinden bir tanesi RDD'de kodların optimizasyonunun zor olmasından kaynaklı olarak, performans sağlamanın kolay olmamasıdır. RDD kullanımı DataFrame ve Dataset yapılarına göre oldukça efor gerektirmektedir. Apache Spark, Spark SQL bileşeniyle birlikte Dataset, DataFrame yapılarına doğrudan klasik SQL sorguları atmaya imkân sağlayan bir yapı da sunup, bu zorluğu ortadan kaldırmıştır [40, 42, 43].

Aşağıda Dataset yapısının RDD'ye göre kullanım kolaylığını göstermek için bir örnek gösterilmiştir. Bu örnek için müşteriler isimli bir dosya oluşturulmuştur. Csv formatındaki bu müşteri dosyasında müşteri ismi, meslek ve maaş bilgisi bulunmaktadır. Şekil 3.5'de dosya içeriği gösterilmiştir.

```
isim,meslek,maas
Ahmet,Isci,3500
Ceren,Ogretmen,4200
Talat,Reklamci,9000
Banu,Asci,6000
Yasemin,Asci,5500
Ali,Ogretmen,5000
Damla,Asci,7300
Murat,Reklamci,12000
Ahmet,Ogretmen,5000
Mehmet,Isci,4000
```

Şekil 3.5: Müşteriler.csv dosyasının içeriği.

Müşteri veri setinden mesleklere göre ortalama maaşlar bulunup, ortalama maaşlara göre sıralanmış halinin RDD ile tasarlanmış hali şekil 3.6, Dataset ile tasarlanmış hali ise 3.7'de gösterilmiştir. Burada Dataset yazımının kolaylığı oldukça göze çarpmaktadır.

```
public class SparkRDD {
public static void main(String[] args) {
    SparkConf sparkConf = new SparkConf().setMaster("local").setAppName("Spark RDD");
   JavaSparkContext sparkContext = new JavaSparkContext(sparkConf);
   JavaRDD<String> stringRDD = sparkContext.textFile( path: "C:/müsteriler.csv");
    JavaPairRDD<String, Double> data = stringRDD.filter(line -> !(line.contains("isim,meslek,maas"))).
            map(line -> line.split( regex: ",")).
            mapToPair(line -> new Tuple2<String, Double>(line[1], Double.parseDouble(line[2])));

    JavaRDD<Tuple2<String, Double>> data2= data.groupByKey().map(line -> {
        double sum = 0;
        Double[] dizi = Iterators.toArray(line._2.iterator(), Double.class);
        int sayi = dizi.length;
        for (int i = 0; i < dizi.length; i++)
            sum += dizi[i];
        return new Tuple2<>(line._1, sum / sayi);
    }).sortBy(line -> line._2, ascending: false, numPartitions: 1);

    data2.foreach(line-> System.out.println(line));

}
}
```

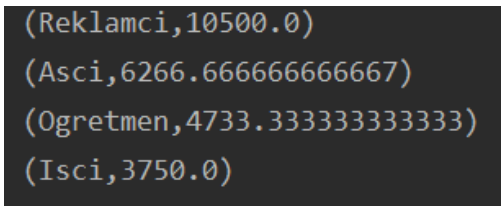
Şekil 3.6: Müşteriler.csv dosyasının RDD ile analizi.



Şekil 3.7: Müşteriler.csv dosyasının Dataset ile analizi.

Şekil 3.7'deki Dataset ile yapılan analize detaylı olarak bakıldığında, Dataset oluşumunun SparkSession oluşturulduktan sonra meydana geldiği gözükmektedir. SparkSession yapısı, Dataset'e açılan kapıdır. Yanına eklenen "appName" ile uygulama ismi, "master" ile de ana sunucunun yeri belirtilmiştir. SparkSession yapısına, "config" parametresi ekleyerek uygulamanın hafıza veya çekirdek miktarı gibi özellikleri de ayarlanabilir. Spark.read.csv yapısıyla csv dosyası okunmuştur. Okuma sırasında belirtilen "optionlar" ile de dosyanın başlığının olduğu ve Spark'ın kolon türlerini otomatik çıkarması gerektiği belirtilmiştir. Ardından groupBy fonksiyonuyla mesleğe göre gruplandırma yapılıp, avg ile de mesleklere göre ortalama maaş bilgisi oluşturulmuştur. ToDF yapısı burada kolon isimlerini vermek, orderBy yapısı ise ortalama maaşa göre sıralama işini yapmak için kullanılmıştır. Desc yapısı sıralamanın büyükten küçüğe olmasını sağlamaktadır. Son olarak görülen "show" fonksiyonu ile de analiz çıktısı ekrana yansıtılmıştır.

RDD ve Dataset ile yapılan analizin sonuçları da şekil 3.8 ve 3.9'da verilmiştir.



Şekil 3.8: Müşteriler.csv dosyasının RDD ile analizi sonucu.

```
+--------+-----------------+
|  meslek|    ortalama maas|
+--------+-----------------+
|Reklamci|          10500.0|
|    Asci|6266.666666666667|
|Ogretmen|4733.333333333333|
|    Isci|           3750.0|
+--------+-----------------+
```

Şekil 3.9: Müşteriler.csv dosyasının Dataset ile analizi sonucu.

Apache Spark'ın, Spark SQL katmanının doğrudan klasik SQL sorguları atmaya da imkân verdiği belirtilmişti. Müşteriler dosyasına, aynı işlemin doğrudan Spark'ın SQL katmanıyla yapılmış hali, şekil 3.10'da gözükmektedir. Yazılan uygulamanın çıktısı da şekil 3.9 ile aynı olmaktadır.

```
public class Sparksql {
    public static void main(String[] args) {

        SparkSession spark= SparkSession.builder().appName("mesleklere göre ortalama maas").master("local").getOrCreate();

        Dataset<Row> data = spark.read().option("header", true).option("inferSchema", true).csv( path: "C:/logs/müşteriler.csv");
        data.createOrReplaceTempView( viewName: "musteriler");

        Dataset<Row> ortalama_maas = spark.sql( sqlText: "select meslek,avg(maas)  ortalama_maas from musteriler group by meslek order by ortalama_maas desc");
        ortalama_maas.show();
    }}
```

Şekil 3.10: Müşteriler.csv dosyasının Spark SQL yapısı ile analizi.

Bu bölümün geri kalanında, Dataset ile ilgili bilgilendirmeler; doğrudan, tez çalışması kapsamında yapılan haftanın gününe göre trafik indeksi analizi kodu üzerinden yapılmıştır. Yapılan diğer tüm analizlerde, bu örneğe benzer adımlar uygulanmıştır.

RDD ile ilgili bilgilendirmede, SparkContext yapısının RDD'e açılan kapı olduğu belirtilmişti. SparkSession yapısı ise Dataset'e açılan kapıdır. Yapılan tüm analizlerde oluşturulan SparkSession yapısı ve temizlenmiş saatlik trafik indeksi verisinin okunma aşaması, şekil 3.11'de verilmiştir. SparkSession yapısındaki builder fonksiyonu yeni bir SparkSession oluştururken kullanılır ve SparkSession.Builder yapısını döndürür, appName yapısı ise yazılan uygulamaya isim verilen kısım olup, herhangi bir isimlendirme yapılabilmektedir. Master kısmı ise ana sunucunun ismini parametre olarak almakta olup, cluster olan bir sistemde kuruluma bağlı olarak yarn, mesos ya da kubernetes olabilir. Tez kapsamında yazılan tüm uygulamalara isim olarak "tez", master olarak ise kodun üzerinde çalıştığı lokal (local) bilgisayar verilmiştir.

SparkSession'daki getOrCreate ise SparkSession mevcutsa oluşturulmuş olanı, eğer mevcut değilse yeni bir SparkSession yapısını döndürür.

```
SparkSession spark = SparkSession.builder().appName("tez").master("local").getOrCreate();
Dataset<Row> traffic_index_data = spark.read().option("inferSchema", true).option("header",true).csv( path: "C:/2020-saatlik-index.csv");
```

Şekil 3.11: 2020-saatlik.csv dosyasının Spark SQL yapısı ile okunması.

Spark isminde SparkSession nesnesi oluşturulduktan sonra, spark.read.csv yapısıyla C dizini altında 2020-saatlik-index.csv dosyası okunmuştur. Okuma sırasında belirtilen "optionlar" ile de dosyanın başlığının olduğu (option("header",true)) ve Spark'ın kolon türlerini otomatik çıkarması (option("inferSchema",true)) gerektiği bilgileri verilmiştir.

Spark Dataset yapısı; select, where, groupBy gibi klasik SQL yapısına benzer fonksiyonlar da içermektedir. Tablo 3.5'de, tez kapsamında yapılan analiz çalışmalarında kullanılan temel fonksiyonlar yer almaktadır [43].

Tablo 3.5: Analiz çalışmasında kullanılan örnek fonksiyonlar ve açıklamaları.

Fonksiyon	Açıklama
select()	Kullanıldığı Dataset'in belirli kolonlarını getirmek için kullanılır. İstenilen kolonlar, virgüllerle ayrılmış halde select fonksiyonu içine yazılır.
groupBy()	Paramatre olarak aldığı kolonlara göre, gruplama işlemi yapar.
agg()	Datasetlerde min, max gibi fonksiyonlar için kullanılır.
withColumn()	Aldığı 2.parametresine göre, yeni bir kolon üreten ya da var olan kolonda değişiklik yapan fonksiyondur.
min()	Parametre olarak aldığı kolonun en küçük değerini döndürür.
max()	Parametre olarak aldığı kolonun en büyük değerini döndürür.

Tablo 3.5 (devam): Analiz çalışmasında kullanılan örnek fonksiyonlar ve açıklamaları.

mean()	Parametre olarak aldığı kolonun ortalama değerini döndürür.
stdev()	Parametre olarak aldığı kolonun standart sapma değerini döndürür.
round()	Ondalıklı sayı içeren kolonların değerlerini, yuvarlamak için kullanılır. Virgülden sonra kaç basamak olacağı ikinci parametre olarak verilebilir.
when()	Verilen koşula göre, istenilen değeri döndürmek için kullanılır.
join()	Verilen kritere göre (left, right, inner vb.) Datasetleri birleştirir.
show()	Dataset'in elemanlarını, ekranda göstermek için kullanılır. Parametre olarak kaç elaman gösterilmesi gerektiği verilmezse, ilk 20 elamanı gösterir.
toDF()	Bir veri listesinden DataFrame elde etmek için kullanılır.

Spark'ın Dataset bileşeninde tarihsel işlemlerle ilgili de birçok fonksiyon bulunmaktadır. Tablo 3.6'da, yapılan analizlerde kullanılan temel tarihsel fonksiyonlar verilmiştir [43]. Bu fonksiyonlar yyyy-MM-dd HH:mm:ss formatındaki trafik indeksi tarihi ya da sokağa çıkma yasağı tarihi gibi kolonlarda kullanılmıştır.

Tablo 3.6: Analiz çalışmasında kullanılan örnek tarih fonksiyonları ve açıklamaları.

Tarihsel Fonksiyon	Açıklama
date_format()	İki parametre alır. İlk parametre işlem yapılacak kolon ismini, diğer parametre ise döndürülecek biçimi içerir. Date_format'ın işlevi; date, timestamp ya da bu formatlara çevrilebilen string kolonları, 2.parametrede verilen formata çevirmektir.

Tablo 3.6 (devam): Analiz çalışmasında kullanılan örnek tarih fonksiyonları ve açıklamaları.

hour()	Date, timestamp ya da bu formatlara çevrilebilen string kolonlarda saat bilgisini int türünde döner.
month()	Date, timestamp ya da bu formatlara çevrilebilen string kolonlarda, ay bilgisini int türünde döner.
dayofmonth()	Date, timestamp ya da bu formatlara çevrilebilen string kolonlarda, ayın kaçıncı günü olduğu bilgisini int türünde döner.

Şekil 3.12'de haftanın günleri bakımından trafik indeksi analizini gerçekleştirmek için tasarlanan kod verilmiştir. Şekilden de görüldüğü gibi basit temel fonksiyonlarla Apache Spark kullanılarak, rahatça analizler gerçekleştirilebilmektedir.

```
public class GünlükTrafikIndex {
    public static void main(String[] args) {

        System.setProperty("hadoop.home.dir", "T:\\hadoop-common-2.2.0-bin-master");

        SparkSession spark = SparkSession.builder().appName("tez").master("local").getOrCreate();
        Dataset<Row> traffic_index_data = spark.read().option("inferSchema", true).option("header",true).csv( path: "C:/2020-saatlik-index.csv");
        Dataset<Row> gunluk_ortalama_index_ham = traffic_index_data
                .withColumn( colName: "gün", functions.date_format(traffic_index_data.col( colName: "TrafikIndeksTarihi"), format: "EEEE")) Dataset<Row>
                .groupBy( col1: "gün") RelationalGroupedDataset
                .agg(functions.min( columnName: "TrafikIndeksi"),functions.max( columnName: "TrafikIndeksi"),functions.mean( columnName: "TrafikIndeksi")
                        ,functions.stddev( columnName: "TrafikIndeksi")) Dataset<Row>
                .toDF("gün","Minimum TrafikIndeksi","Maksimum TrafikIndeksi","Ortalama TrafikIndeksi","StandartSapma TrafikIndeksi");

        Dataset<Row> gunluk_ortalama_index  = gunluk_ortalama_index_ham
                .select(functions.when(gunluk_ortalama_index_ham.col( colName: "gün").equalTo("Monday"), value: "Pazartesi")
                                .when(gunluk_ortalama_index_ham.col( colName: "gün").equalTo("Tuesday"), value: "Salı")
                                .when(gunluk_ortalama_index_ham.col( colName: "gün").equalTo("Wednesday"), value: "Çarşamba")
                        .when(gunluk_ortalama_index_ham.col( colName: "gün").equalTo("Thursday"), value: "Perşembe")
                                .when(gunluk_ortalama_index_ham.col( colName: "gün").equalTo("Friday"), value: "Cuma")
                                .when(gunluk_ortalama_index_ham.col( colName: "gün").equalTo("Saturday"), value: "Cumartesi")
                                .when(gunluk_ortalama_index_ham.col( colName: "gün").equalTo("Sunday"), value: "Pazar")
                ,functions.round(gunluk_ortalama_index_ham.col( colName: "Minimum TrafikIndeksi"), scale: 2)
                                ,functions.round(gunluk_ortalama_index_ham.col( colName: "Maksimum TrafikIndeksi"), scale: 2)
                                ,functions.round(gunluk_ortalama_index_ham.col( colName: "Ortalama TrafikIndeksi"), scale: 2)
                                ,functions.round(gunluk_ortalama_index_ham.col( colName: "StandartSapma TrafikIndeksi"), scale: 2))
                        .toDF("Gün","Minimum Trafik Indeksi","Maksimum Trafik Indeksi","Ortalama Trafik Indeksi","Standart Sapma")
                        .orderBy(functions.desc( columnName: "Ortalama Trafik Indeksi"));
            gunluk_ortalama_index.show();


    }
}
```

Şekil 3.12: Haftanın günlerine göre İstanbul 2020 yılı trafik indeksi analizi uygulaması.

Şekil 3.12'e bakıldığında SparkSession oluşturulduktan sonra, C dizini altındaki 2020-saatlik-index.csv dosyası okunup, traffic_index_data adında bir Dataset'e atandığı gözükmektedir. Ardından date_format ile TrafikIndeksTarihi kolonunun gün bilgisi bu Dataset'e withColumn

ile eklenip, oluşturulan gün isimli kolona göre gruplama yapılmıştır. Gruplama işleminden sonra gün bazında trafik indeksi kolonunun minimum, maksimum, ortalama ve standart sapma değerlerini almak için agg fonksiyonuna sırasıyla min, max, mean ve stdev fonksiyonları kullanılmıştır. İfadenin en sonunda yer alan toDF fonksiyonu ile yeni oluşan kolonlara isimlendirme yapılıp, gunluk_ortalama_index_ham isimli yeni bir Dataset oluşturulmuştur. Yeni oluşturulan Dataset'deki gün kolonunda, gün isimleri date_format fonksiyonundaki 2.parametre olan "EEEE" ile Sunday, Monday gibi İngilizce olarak gelmektedir. Bundan dolayı gün isimleri when fonksiyonu ile Türkçe haline getirilmiştir. Ardından round ile Minimum_Trafik_Indeksi, Maksimum_Trafik_Indeksi, Ortalama_Trafik_Indeksi, StandartSapma_TrafikIndeksi kolonlarındaki değerler virgülden sonra 2 basamak olacak şekilde yuvarlanmıştır. Düzenleme yapılan kolonlara toDF ile yeni isimlendirme yapılıp, yeni oluşan Dataset'i Ortalama Trafik Indeksi değerine göre sıralanmış halde elde etmek için orderBy fonksiyonu kullanılmıştır. Sıralamanın büyükten küçüğe olması için orderBy fonksiyonuna desc fonksiyonu parametre olarak verilmiştir. Oluşan yeni Dataset'e günlük_ortalama_index ismi verilmiştir. Gunluk_ortalama_index Dataset'inin değerlerini, ekranda görmek için show fonksiyonu gösterilmiştir. Uygulamanın ekran çıktısı şekil 3.13'de gözükmektedir.

Gün	Minimum Trafik Indeksi	Maksimum Trafik Indeksi	Ortalama Trafik Indeksi	Standart Sapma
Cuma	1.0	83.0	27.41	19.1
Çarşamba	1.0	81.0	26.63	18.1
Salı	1.0	80.0	26.59	18.81
Perşembe	1.0	74.0	26.59	18.05
Pazartesi	1.0	77.0	25.13	18.23
Cumartesi	1.0	63.0	19.22	15.66
Pazar	1.0	50.0	13.59	11.8

Şekil 3.13: Haftanın günlerine göre İstanbul 2020 yılı ortalama trafik indeksi sıralaması.

Tez çalışmasında; görselleştirmenin, Kibana da yapılacağı belirtilmişti. Bundan dolayı şekil 3.12'deki günlük_ortalama_index Dataseti şekil 3.14'deki komutla Elasticsearch'e yazdırılıp, Kibana'nın ise doğrudan Elasticsearch'deki veriyi kullanması sağlanmıştır.

```
gunluk_ortalama_index.write().format("es").option("es.nodes", "localhost").option("es.port", "9200")
        .mode("overwrite").save( path: "günlük_trafik_index/günlük_trafik_index");
```

Şekil 3.14: Gunluk_ortalama_index Dataset'inin elasticsearch'e yazılması.

Şekil 3.14'de write() fonksiyonu Dataset'i yazdırmak, format("es") verinin Elasticsearch'e yazdırılacağını ifade etmek, optionlar da Elasticsearch port ve node bilgisini vermek için kullanılmıştır. Mode fonksiyonu yazdırma modunu belirtmek için kullanılmış olup, save(günlük_trafik_index/günlük_trafik_index) ile de günlük_trafik_index indeksli ve type'li Elasticsearch'e yazdırma işlemi tamamlanmıştır. Yapılan tüm analizlerde kodlamalar haftanın günlerine göre trafik indeksi analizine benzer olarak tasarlanmıştır. Tez çalışması kapsamında yapılan tüm analiz ve görselleştirmedeki genel süreç şekil 3.15'de, analizler sırasında yapılan temel işlemlerin blok diyagramı ile ifade edilmiş hali ise şekil 3.16'da verilmiştir.



Şekil 3.15: Tez kapsamında yapılan analizler için genel süreç.



Şekil 3.16: Analizlerde yapılan temel işlemlerin blok diyagramı ile ifade edilmesi.

3.2.3.3. Spark MLlib

Veriler üzerinden yola çıkarak, tahminleme yapmak için yapılan modellemeler makine öğrenmesi olarak adlandırılır. Spark'ın makine öğrenmesi için gerekli kütüphaneleri içeren bileşeni, Spark MLlib olarak adlandırılmaktadır. Apache Spark, MLlib kütüphanesiyle birlikte makine öğrenmesini de büyük veri mimarisine uyarlayıp, ölçeklenebilir halde makine öğrenmesi uygulamaları yapabilmeyi sağlamıştır. Apache Spark MLlib kütüphanesi; sınıflandırma, regresyon, kümeleme, boyut azaltma gibi makine öğrenmesi algoritmaları için birçok yönteme sahiptir [44].

Tez kapsamında trafik yoğunluğu tahmini için kolay uygulanabilirliği ve yaygın kullanımı sebebiyle lojistik regresyon, başarılı genel amaçlı algoritmalardan biri olması sebebiyle karar ağaçları, birden fazla karar ağacıyla model oluşturarak aşırı uyum (overfitting) problemi riskini minimuma indirmek ve yüksek başarı oranına sahip olma ihtimalini arttırmak için rastgele orman yöntemi kullanılmıştır. İkili sınıflandırma algoritmalarının uygulamada nasıl bir sonuç göstereceğini görmek için bu 3 algoritmaya ilave olarak gradyan yükseltme ağacı ve doğrusal destek vektör makinesi tabanlı OneVsRest yöntemi kullanılmıştır. Çıkan sonuçlara göre kullanılan bu modellerin doğruluk oranları bulunup, yapılan tahminlere göre karşılaştırma yapılmıştır. Sınıflandırma algoritmaları kullanılacağı için tüm yöntemlerde trafik yoğunluğu aşağıdaki şekilde 3 kategoriye ayrılmıştır. Bu ayırmaya, veri setindeki trafik indeksi değerlerinin dağılımına göre karar verilmiştir. Trafik indeksi için öz nitelikler ise ay, ayın günü, gün, saat, sokağa çıkma yasağının ve resmi tatilin olup olmaması alınmıştır.

0<trafik indeksi<=20 →1

20<trafik indeksi<=50 →2

50<trafik indeksi<=100 →3

Veri setleri, makine öğrenmesi algoritmalarına girmeden önce çeşitli işlemlerden geçirilmelidir. Bu işlemlerden ilki veri setindeki tüm kolonların nümerik yapılmasıdır. Makine öğrenmesi uygulamalarında, kolonları nümerik yapmak için genellikle StringIndexer sınıfı kullanılmaktadır. Tez kapsamında yapılan makine öğrenmesi uygulamalarında kolonlar doğrudan withColumn fonksiyonu ile nümerik değerlere atandığı için StringIndexer sınıfı kullanılmamıştır. Kullanılan özellikler arasında büyük bir ölçek farkı varsa StandartScaler sınıfı ile ölçekleme yapılabilmektedir. Tez kapsamında ölçekleme yapmaya gerek duyulmamıştır.

Makine öğrenmesi uygulaması yapmak için veride yapılması gereken diğer bir işlem ise etiketin öz niteliklerinin vektör haline getirilmesidir. Kolonlardan vektör elde etmek için daha çok VectorAssembler sınıfı tercih edilmektedir. Bu yapı setInputCols yapısıyla dizi olarak aldığı kolonları, setOutputCol ile verilen isme göre vektör haline dönüştürmektedir. Tez kapsamında makine Öğrenmesi uygulamalarını gerçekleştirmek için veri setinde yapılan işlemler şekil 3.17'de verilmiştir.

```
public class Machine_Learning {
    public static void main(String[] args) {


        SparkSession spark = SparkSession.builder().appName("tez").master("local").getOrCreate();
        Dataset<Row> traffic_index_data = spark.read().option("inferSchema", true).option("header", true).csv( path: "C:\\2020-saatlik-index.csv");

        Dataset<Row> data1 = traffic_index_data.withColumn( colName: "ay", month(col( colName: "TrafikIndeksTarihi")))
                .withColumn( colName: "AyinGünü", dayofmonth(col( colName: "TrafikIndeksTarihi")))
                .withColumn( colName: "gün", date_format(traffic_index_data.col( colName: "TrafikIndeksTarihi"), format: "EEEE"))
                .withColumn( colName: "saat", hour(col( colName: "TrafikIndeksTarihi")))
                .withColumn( colName: "gün", when(col( colName: "gün").equalTo("Monday"), value: 1).when(col( colName: "gün").equalTo("Tuesday"), value: 2)
                        .when(col( colName: "gün").equalTo("Wednesday"), value: 3)
                        .when(col( colName: "gün").equalTo("Thursday"), value: 4)
                        .when(col( colName: "gün").equalTo("Friday"), value: 5)
                        .when(col( colName: "gün").equalTo("Saturday"), value: 6)
                        .when(col( colName: "gün").equalTo("Sunday"), value: 7)).drop("TrafikIndeksTarihi");

        Dataset<Row> data2 = data1.withColumn( colName: "TrafikIndeksi", when(col( colName: "TrafikIndeksi").leq(20), value: 1)
                .when(col( colName: "TrafikIndeksi").gt(20).and(col( colName: "TrafikIndeksi").leq(50)), value: 2)
                .when(col( colName: "TrafikIndeksi").gt(50), value: 3))
                .withColumn( colName: "label", col( colName: "TrafikIndeksi").cast("double")).drop("TrafikIndeksi");


        VectorAssembler features = new VectorAssembler().setInputCols(new String[]{"ay", "AyinGünü", "gün", "saat", "SokagaCikmaYasagi", "ResmiTatil"})
                .setOutputCol("features");

        Dataset<Row> data3 = features.transform(data2);


        Dataset<Row>[] datasets = data3.randomSplit(new double[]{0.7, 0.3}, seed: 100);
        Dataset<Row> traindata = datasets[0];

        Dataset<Row> testdata = datasets[1];
```

Şekil 3.17: Makine öğrenmesi için veri setinde yapılan işlemler.

Makine öğrenmesi modellerinin doğruluğu ise şekil 3.18'de görülen MulticlassClassificationEvaluator sınıfıyla kontrol edilmiştir. Modelin performansı doğru tahminlerin tüm tahminlere oranını veren "accuracy" ile hesaplanmıştır. Şekil 3.18'de yer alan transform_data, test datasına modellerin uygulanması sonucu elde edilen veri setidir.

```
MulticlassClassificationEvaluator evaluator = new MulticlassClassificationEvaluator().setLabelCol("label").setMetricName("accuracy").setPredictionCol("prediction");
System.out.println("Dogruluk orani:"+evaluator.evaluate(transform_data));
```

Şekil 3.18: Makine öğrenmesi için hazırlanan modellerin doğruluk oranlarının kontrol edilmesi.

Tez kapsamında modellerin sonuçları değerlendirilirken modellerin doğruluk oranı dışında, etiket bazında kesinlik, hassasiyet ve f1 skoru değerleri de hesaplanmıştır. Bu değerlerin

hesaplanmasında, karışıklık matrisinden yararlanılmaktadır. Karışıklık matrisi, modelde doğru ve yanlış tahmin edilen sınıfların sayılarını gösteren bir yapıdır. Apache Spark'da karışıklık matrisi, kesinlik, hassasiyet ve f1 skoru gibi metrikleri elde etmek için MulticlassMetrics sınıfı kullanılmaktadır. Şekil 3.19'da label=1 (normal düzey trafik) sınıfı için başarı metriklerinin elde edilmesi gösterilmiştir. Şekilde yer alan transform_data, sınıflandırıcı tarafından dönüşüme uğramış test verisidir.

```
MulticlassMetrics multiclassMetrics = new MulticlassMetrics(transform_data.select( col: "prediction", ...cols: "label"));
System.out.println("Karışıklık Matrisi: \n"+multiclassMetrics.confusionMatrix());
System.out.println("Kesinlik: " + multiclassMetrics.precision( label: 1));
System.out.println("Anma: " + multiclassMetrics.recall( label: 1));
System.out.println("F ölçütü: " + multiclassMetrics.fMeasure( label: 1));
```

Şekil 3.19: Normal düzey trafik için başarı metriklerin oluşturulması.

Tablo 3.7: Karışıklık matrisi şeması.

		Tahmin edilen Sınıf		
	Sınıflar	$c_0 \ldots c_{k-1}$	c_k	$c_{k+1} \ldots c_n$
Gerçek Sınıf	$c_0 \ldots c_{k-1}$	TN	FP	TN
	c_k	FN	TP	FN
	$c_{k+1} \ldots c_n$	TN	FP	TN

Tablo 3.7'de karışıklık matrisi verilmiştir. $c_{ij,}$ i sınıfının j sınıfı olarak tahmin edilme sıklığını ifade etmektedir. Bir karışıklık matrisi, hedef sınıf k'ya göre 4 çeşit sınıflandırma sonucu içerir. Bu sonuçlar TP, TN, FP ve FN olarak ifade edilir [45].

TP (True Positive) : Pozitif sınıfın doğru tahmini= $c_{k,k}$ (3.1)

TN (True Negative) : Negatif sınıfın doğru tahmini = $\sum_{i,j \in N\setminus\{k\}} c_{ij}$ (3.2)

FP (False Positive) : Pozitif sınıfın yanlış tahmini = $\sum_{i \in N\setminus\{k\}} c_{ik}$ (3.3)

FN (False Negative) : Negatif sınıfın yanlış tahmini = $\sum_{i \in N\setminus\{k\}} c_{ki}$ (3.4)

Makine öğrenmesi modellerini sınıflara göre değerlendirirken kullanılan kesinlik, hassasiyet ve f1 skoru değerlerinin açıklamaları ve formülleri ise şöyledir:

Kesinlik: Sınıflandırıcının, belirli bir sınıfı doğru tahmin etme başarısını gösteren bir ölçüttür.

$$Kesinlik_{sınıf} = \frac{TP_{sınıf}}{TP_{sınıf}+FP_{sınıf}} \quad (3.5)$$

Hassasiyet: Sınıflandırıcının, doğru bir sınıfı tahmin etme yeteneğini gösterir.

$$Hassasiyet_{sınıf} = \frac{TP_{sınıf}}{TP_{sınıf}+FN_{sınıf}} \quad (3.6)$$

F1 skoru: Kesinlik ve hassasiyetin harmonik ortalamasından elde edilen ölçüttür. Sınıflandırıcının ne kadar kesin ve güçlü olduğunu gösterir.

$$F1\ skoru_{sınıf} = \frac{2.TP_{sınıf}}{2.TP_{sınıf} + FN_{sınıf} + FP_{sınıf}} \quad (3.7)$$

Tez kapsamında Spark'ın MLlib kütüphanesinde bulunan lojistik regresyon, karar ağaçları, rastgele orman, gradyan yükseltme ağacı tabanlı OneVsRest ve doğrusal destek vektör makinesi tabanlı OneVsRest yöntemleri kullanılmıştır. Makine öğrenmesi uygulamalarında yapılan temel işlemlerin özetlenmiş hali, blok diyagramı halinde şekil 3.20'de verilmiştir.



Şekil 3.20: Makine öğrenmesi uygulamalarında yapılan temel işlemlerin süreci.

Lojistik Regresyon (Logistic regression) Yöntemi: Lojistik regresyon daha çok iki sınıflı sınıflandırma algoritmalarında kullanılan, fakat ikiden fazla sınıf içeren problemleri tahmin etmek için de genelleştirilebilen makine öğrenmesi algoritmasıdır. Bu algoritmada, K sınıflı bir sistemde K-1 lojistik model oluşturulmaktadır. Çok sınıflı sınıflandırma problemleri için, algoritma K kat sayısı kümesi ya da KxJ boyutlu bir matris üretir. Burada K sınıf sayısını, J ise öznitelik sayısını ifade etmektedir. [46].

Denklem 3.8 ve 3.9'da sırasıyla iki değişkenli ve çok değişkenli lojistik regresyon modeli formülü verilmiştir. Denklemlerde P(Y) hedef değişkenini ifade eder. Bağımsız değişkenler X, katsayılar ise β ile ifade edilmiştir.

$$P(Y) = \frac{e^{\beta_0+\beta_1 X_1}}{1+e^{\beta_0+\beta_1 X_1}} \quad (3.8)$$

$$P(Y) = \frac{e^{Z}}{1+e^{Z}} \quad Z = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \ldots \beta_p X_p \quad (3.9)$$

Çok sınıflı sınıflandırma problemleri için k∈1,2,3….,K sınıflarının koşullu olasılıkları denklem 3.10'daki softmax fonksiyonu ile modellenir.

$$P(Y=k|\mathbf{X},\beta_k,\beta_{0k}) = \frac{e^{\beta_k.\mathbf{X}+\beta_{0k}}}{\sum_{k'=0}^{K-1} \beta_{k'}.\mathbf{X}+\beta_{0k'}} \quad (3.10)$$

Tez kapsamında oluşturulan Lojistik regresyon nesnesi ve modeli şekil 3.21'de verilmiştir.

```
LogisticRegression logisticRegression = new LogisticRegression().setFeaturesCol("features").setLabelCol("label").setFamily("multinomial");

LogisticRegressionModel logisticRegressionModel = logisticRegression.fit(traindata);

Dataset<Row> transform_data = logisticRegressionModel.transform(testdata);
```

Şekil 3.21: Lojistik regresyon modelinin oluşturulması.

Apache Spark MLlib kütüphanesinde bulunan lojistik regresyon yönteminin temel parametreleri ve varsayılan (default) değerleri tablo 3.8'de verilmiştir.

Tablo 3.8: Lojistik regresyon yönteminin temel parametreleri ve varsayılan değerleri.

Parametre	Varsayılan Değeri
aggregationDepth	2
elasticNetParam	0.0
family	auto
fitIntercept	true
maxIter	100
regParam	0.0
standardization	true
threshold	0.5
tol	1.0E-6

Karar Ağaçları (Decision tree) Yöntemi: Karar ağaçları, sınıflandırma ve regresyon amaçlı kullanılan makine öğrenmesi yöntemlerindendir. Bir karar ağacının temel birimi kök olarak adlandırılır. Her bir gözlemin sınıflandırılmasını sağlayan ve kök hücrelerinin altında bulunan noktalar ise düğüm olarak adlandırılır. Karar ağacının en altında bulunan ve varılan sonucu gösteren kısım da yaprak olarak ifade edilmektedir. Bu algoritma ile eğitim için verilen verilere çeşitli sorular sorulmakta ve alınan cevaplara göre model oluşturulmaktadır. Diğer yöntemlere göre anlaşılmasının ve yorumlanmasının daha kolay olmasından dolayı makine öğrenmesi

uygulamalarında sık sık tercih edilmektedir. Şekil 3.22'de örnek bir karar ağacı verilmiştir. Karar ağaçlarının algoritması büyük verileri çeşitli alt gruplara bölmeye dayandığı için, burada bölmenin nasıl gerçekleşeceği önem arz etmektedir. Bu bölünme için karar ağaçları birden fazla algoritma kullanmaktadır. Bu algoritmalar arasında en çok kullanılanları gini ve entropidir [47, 48].

Entropi belirsizliğin ya da rastgeleliğin ölçüsü olarak ifade edilir. Denklem 3.11'de ifadesi verilmiştir. Burada f_i i.etiketin frekansını, C ise farklı etiket sayısını ifade etmektedir. Etiketler düzenliyse entropinin düşük olduğu söylenebilir. Bundan dolayı bölünmeler entropiyi ne kadar azaltırsa, modelin başarısı da o kadar yüksek olur.

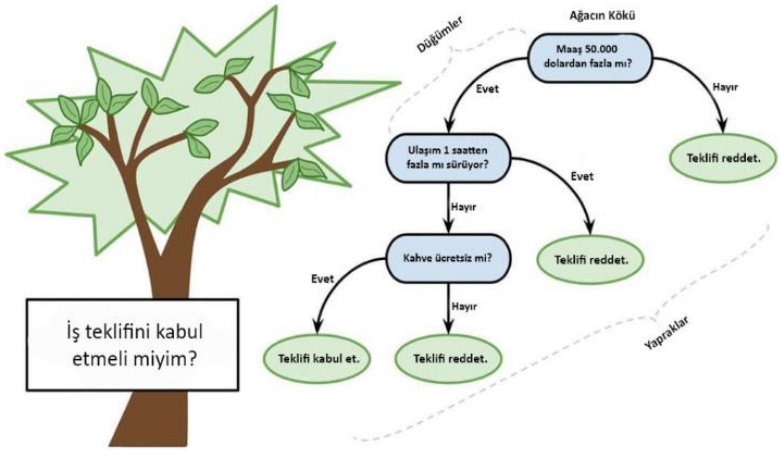
$$\sum_{i=0}^{C} -f_i \log(f_i) \tag{3.11}$$

Gini'nin ise entropi ile arasında çok büyük fark yoktur. Gini daha çok frekansı fazla olan sınıfı ayrıştırmaya meyillidir. Denklem 3.12'de gini denklemi verilmiştir.

$$\sum_{i=0}^{C} f_i(1 - f_i) \tag{3.12}$$

Karar ağaçlarında en iyi bölünmeyi belirlemek için "bilgi kazancı" ifadesini kullanırız. Bilgi kazancı; ana düğüm safsızlığından (impurity), iki alt düğüm safsızlığının ağırlıkları toplamı arasındaki farktır [47]. Denklem 3.13 de bilgi kazancı denklemi verilmiştir. Denklemde, IG(D,s) bölünmüş s'nin N boyutlu bir D veri kümesindeki bilgi kazancını, D_{left}, D_{right} ise sırasıyla N_{left} ve N_{right} boyutlarındaki bölünmüş veri kümelerini ifade eder.

$$\text{IG(D,s)=Impurity(D)}-(N_{left}/\text{N})\text{Impurity}(D_{left})-(N_{right}/\text{N})\text{Impurity}(D_{right}) \tag{3.13}$$



Şekil 3.22: Örnek bir karar ağacı algoritması [48].

Tez kapsamında oluşturulan Karar ağacı nesnesi ve modeli şekil 3.23'de verilmiştir.

```
DecisionTreeClassifier decisionTreeClassifier = new DecisionTreeClassifier().setFeaturesCol("features").setLabelCol("label");

DecisionTreeClassificationModel  decisionTreeClassificationModel = decisionTreeClassifier.fit(traindata);

Dataset<Row> transform_data = decisionTreeClassificationModel.transform(testdata);
```

Şekil 3.23: Karar ağacı modelinin oluşturulması.

Apache Spark MLlib kütüphanesinde bulunan karar ağacı yönteminin temel parametreleri ve varsayılan (default) değerleri tablo 3.9'da verilmiştir.

Tablo 3.9: Karar ağacı yönteminin temel parametreleri ve varsayılan değerleri.

Parametre	Varsayılan Değeri
impurity	gini
maxBins	32
maxDepth	5
maxMemoryInMB	256
minInfoGain	0.0
minInstancesPerNode	1
seed	159147643

Rastgele Orman (Random forest) Yöntemi: Rastgele orman algoritması, hem regresyon hem de sınıflandırma uygulamaları için kullanılabilen, temeli karar ağacı yöntemine dayanan makine öğrenmesi algoritmalarından birisidir. Rastgele orman yöntemi, karar ağaçları

algoritmasındaki veriyi ezberleme probleminin üstesinden gelebilmek için veri setlerini rastgele alt kümelere bölüp, bunları eğitmektedir. Böylelikle birden fazla karar ağacını bir araya getirip, her birinin tahminde bulunmasını sağlamaktadır. Rastgele Orman algoritması; tahmin işlemini, sınıflandırmada en çok oy alan tahmine göre, regresyonda ise her bir ağacın döndürdüğü reel sayının ortalamasını alıp yapmaktadır [49]. Rastgele orman yöntemi, birçok parametreye sahiptir. Aşağıda performansı iyileştirmek 2 önemli parametre verilmiştir.

- **numTrees:** Ağaç sayısının girilmesini sağlayan parametredir. Ağaç sayısının arttırılması tahminlerdeki varyansı azaltıp, modelin test zamanı doğruluğunu geliştirir.
- **maxDepth:** Her bir ağacın derinliğinin girilmesini sağlayan parametredir. Derinliği arttırmak modeli daha güçlü kılar, fakat derin ağaçların eğitilmesi uzun sürmektedir.

Tez kapsamında oluşturulan rastgele orman nesnesi ve modeli şekil 3.24'de verilmiştir. Modelin başarı oranı ağaç sayısı 1- 10 arasında kontrol edilmiş ve ağaç sayısı 2 iken en yüksek doğruluk oranı yakalanmıştır.

```
RandomForestClassifier randomForestClassifier = new RandomForestClassifier().setFeaturesCol("features").setLabelCol("label").setNumTrees(2);
RandomForestClassificationModel randomForestClassificationModel = randomForestClassifier.fit(traindata);
Dataset<Row> transform_data = randomForestClassificationModel.transform(testdata);
```

Şekil 3.24: Rastgele orman modelinin oluşturulması.

Apache Spark MLlib kütüphanesinde bulunan rastgele orman yönteminin temel parametreleri ve varsayılan (default) değerleri tablo 3.10'da verilmiştir.

Tablo 3.10: Rastgele orman yönteminin temel parametreleri ve varsayılan değerleri.

Parametre	Varsayılan Değeri
featureSubsetStrategy	auto
impurity	gini
maxBins	32
maxDepth	5
maxMemoryInMB	256
minInfoGain	0.0
minInstancesPerNode	1
numTrees	20
seed	207336481
subsamplingRate	1.0

OneVsRest Sınıflandırma Yöntemi: Makine öğrenmesinde kullanılan bazı popüler sınıflandırma yöntemleri, sadece ikili sınıflandırma durumlarında kullanılmaktadır. Dolayısıyla bu yöntemler doğrudan çok sınıflı problemlere uygulanamamaktadır. Bu durumun üstesinden gelmek için çoklu sınıfları birçok ikili sınıflara ayıran bazı yöntemler geliştirilmiştir. Bu yöntemlerden biri OneVsRest sınıflandırma yöntemidir. Bire karşı hepsi olarak da bilinen bu yöntem ikili sınıflandırmayı etkili bir şekilde yapabilen bir yöntemin temel sınıflandırıcı olarak verilmesiyle, çok sınıflı sınıflandırmayı sağlayan denetimli bir sınıflandırma yöntemidir. OneVsRest yöntemi, temel sınıflandırıcı için sınıflandırmanın durumlarını alıp, her bir sınıf özelinde ikili sınıflandırma problemi oluşturur. Herhangi bir sınıfı, diğer tüm sınıflardan ayırarak, etiket kolonun o sınıf olup olmadığını tahmin eder [50]. Örneğin verilen özelliklere göre canlının insan, hayvan ve ya bitki olarak tahmin etmek için oluşturulan makine öğrenmesi uygulamasını destek vektör makineleri gibi ikili sınıflandırmada kullanılan yöntemlerle doğrudan gerçekleştirilemez. Fakat OneVsRest yöntemiyle problem aşağıdaki şekilde 3 ikili sınıflandırma durumuna ayrılıp, destek vektör makinelerinin kullanılması sağlanabilir.

Durum 1: İnsan x [Hayvan, Bitki]

Durum 2: Hayvan x [İnsan, Bitki]

Durum 3: Bitki x [Hayvan, İnsan]

Örnek verilen problem doğrudan canlının insan, hayvan ve ya bitki olup olmadığını tahmin etmek yerine yukarıdaki 3 tane ikili sınıflandırma problemine dönüşür. Bu yöntem, etiket kolonunda çok fazla sınıfın bulunduğu durumlar için, süreci yavaşlatabilir. Fakat az sınıflı problemlere ikili sınıflandırma yöntemlerini uygulamak için oldukça idealdir. Tez kapsamında Gradyan yükseltme ağacı ve destek vektör makineleri tabanlı OneVsRest yöntemleri kullanılmıştır.

Gradyan Yükseltme Ağacı (Gradient–boosted tree) Yöntemi: Gradyan yükseltme ağacı, sınıflandırma ve regresyon problemlerinde kullanılan topluluk algoritmalarından biridir. Karar ağaçlarını yinelemeli şekilde eğiterek, hataları en aza indirmeyi amaçlar. Bu yöntemde, her yinelemede önceki yinelemelerde yapılan hataları düzeltmek için yeni modeller oluşturulur. Yani sonradan oluşan modellerin önceki oluşanlara bağımlı olduğu, sıralı bir işlem yapılır. Gradyan yükseltme ağacı 3 temel bileşene sahiptir. Bunlar verilen veriye göre modelin doğruluğunu kestirmek için kullanılan kayıp fonksiyonu (Loss Function), yüksek hata oranına sahip öğrenimleri ifade etmek için kullanılan zayıf öğrenici (Weak Learner) ve zayıf

öğrenicileri her seferinde sıralı bir şekilde ekleme yaklaşımını ifade eden katkı modeli (Additive Model) yaklaşımıdır [51].

Gradyan yükseltme Ağacı, Spark MLLIB kütüphanesinde ikili sınıflandırmayı ve regresyonu desteklemektedir. Tez kapsamında kullanılan Gradyan Yükseltme ağacı tabanlı OneVsRest algoritma nesnesinin ve modelinin oluşturulması şekil 3.25'de gösterilmiştir.

```
GBTClassifier gbtClassifier = new GBTClassifier().setFeaturesCol("featues").setLabelCol("label").setPredictionCol("prediction");
OneVsRest oneVsRest = new OneVsRest().setClassifier(gbtClassifier);
OneVsRestModel oneVsRestModel = oneVsRest.fit(traindata);
```

Şekil 3.25: Gradyan yükseltme ağacı tabanlı OneVsRest modelinin oluşturulması.

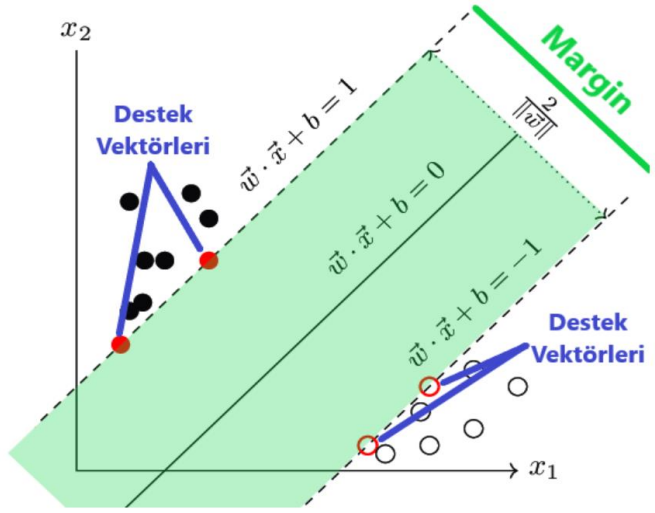
Apache Spark MLlib kütüphanesinde bulunan gradyan yükseltme ağacı yönteminin temel parametreleri ve varsayılan (default) değerleri tablo 3.11'de verilmiştir.

Tablo 3.11: Gradyan yükseltme ağacı yönteminin temel parametreleri ve varsayılan değerleri.

Parametre	Varsayılan Değeri
impurity	gini
lossType	logistic
maxBins	32
maxDepth	5
maxIter	20
maxMemoryInMB	256
minInfoGain	0.0
minInstancesPerNode	1
seed	-1287390502
stepSize	0.1
subsamplingRate	1.0

Doğrusal Destek Vektör Makineleri (Linear support vector machine) Yöntemi: Destek Vektör makineleri regresyon ve sınıflandırma problemlerinde kullanılan denetimli bir makine öğrenmesi algoritmasıdır. Genellikle sınıflandırma problemlerinde tercih edilmektedir. Destek vektör makineleri algoritmasının temel amacı tahmini yapılacak veriyi doğru kategorize etmek için, n boyutlu uzayı eğitim verisine göre en iyi şekilde ayıracak sınırı belirlemektedir. Bu sınıra, hiper-düzlem denir. SVM hiper-düzlemi belirlemek için sınır noktaları/vektörleri seçer, bu noktalara destek vektörleri denir. Destek vektörleri arasındaki margin (sınır) ne kadar geniş ise sınıf o kadar iyi ayrıştırılmış olur [52].

Spark MLLIB kütüphanesinde; doğrusal vektör makineleri, sınıflandırmalarda sadece ikili sınıflandırmayı desteklemektedir.



Şekil 3.26: Destek vektör makineleri için iki sınıflı problem örneği [52].

Şekil 3.26'da siyah ve beyaz olmak üzere iki sınıf verilmiştir. Formüle bakacak olursak aşağıdaki şekilde ifade edilebilir.

Y = 0 w^T.x>0 +b <0 (3.14)

Y = 1 w^T.x>0 +b ≥0 (3.15)

Verilen denklemde w ağırlık vektörü, x girdi vektörü b ise sapmadır. Yeni değer için çıkan sonuç 0'dan küçükse beyaz, 0'a büyük veya eşitse siyah noktaların sınıfına dahil olacaktır. Tez kapsamında kullanılan SVM tabanlı OneVsRest yönteminin nesnesinin ve modelinin oluşturulması şekil 3.27'de verilmiştir.

```
LinearSVC linearSVC = new LinearSVC().setFeaturesCol("featues").setLabelCol("label").setPredictionCol("prediction");
OneVsRest oneVsRest = new OneVsRest().setClassifier(linearSVC);
OneVsRestModel oneVsRestModel = oneVsRest.fit(traindata);
```

Şekil 3.27: Doğrusal destek vektör makineleri tabanlı OneVsRest modelinin oluşturulması.

Apache Spark MLlib kütüphanesinde bulunan doğrusal destek vektör makineleri yönteminin temel parametreleri ve varsayılan (default) değerleri tablo 3.12'de verilmiştir.

Tablo 3.12: Doğrusal destek vektör makineleri yönteminin temel parametreleri ve varsayılan değerleri.

Parametre	Varsayılan Değeri
aggregationDepth	2
fitIntercept	true
maxIter	100
regParam	0.0
standardization	true
threshold	0.0
tol	1.0E-6

3.2.3.4. Spark Streaming

Apache Spark'ın akan verileri üzerinde analiz yapmayı sağlayan bileşenidir. Flume, Nifi, Kafka, Hdfs gibi kaynaklardan Spark Streaming ile anlık veri analizi yapılabilmektedir. Şekil 3.28'de Spark Streaming için temel kaynak ve hedef örnekleri verilmiştir. Spark Streaming, akan veriler üzerinde makine öğrenmesi ve GraphX algoritmalarının uygulanmasına da izin vermektedir [53].



Şekil 3.28: Spark Streaming için temel kaynak ve hedef örnekleri [53].

Spark Streaming veri analizini akan verileri batchlere (yığın) bölüp yapmaktadır. Oluşturulan her yığın Spark motoru tarafından işlenip, elde edilen sonuçlar gene yığınlar halinde hedef sisteme aktarılır. Spark Streaming'in temel çalışma prensibi şekil 3.29'da verilmiştir.



Şekil 3.29: Spark Streaming temel çalışma prensibi [53].

3.2.3.5. GraphX

GraphX**,** Apache Spark'ın graph analizi için oluşturulan bileşenidir. RDD'leri yönlü graphlara dönüştürüp, işlenmesini sağlamaktadır. Sosyal medyadaki arkadaş analizleri ya da bu videoyu beğenenler bunları da beğendi gibi öneriler graph x analizlerine örnek olarak verilebilir.

3.2.4. Elasticsearch

Elasticsearch, temeli indeksleme mekanizmasına dayanan, Apache Lucene üzerinde java ile geliştirilmiş, açık kaynaklı kodlu arama ve analiz platformudur. Elasticsearch dağıtık mimaride çalışabilmektedir. Verileri json formatında tutmaktadır. Özellikle sorgulama ve metin arama gibi işlerde Elasticsearch oldukça fazla tercih edilmektedir. Elasticsearch'de klasik ilişkisel veri tabanlarındaki veri tabanlarına benzer bileşenine "index", gene ilişkisel veri tabanlarındaki tabloya karşılık gelen bileşenine ise "type" ismi verilmektedir. Elasticsearch'de her bir kayıt document olarak adlandırılmaktadır [54].

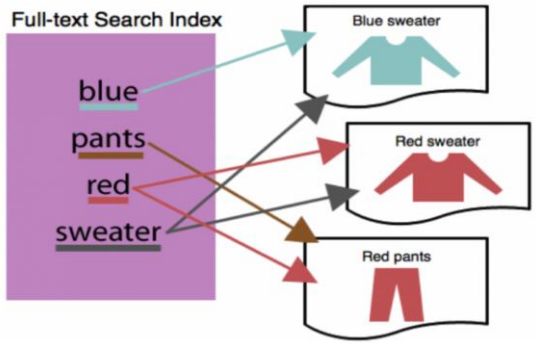
Tablo 3.13'de Elasticsearch'un RDBMS'lerin temel birimlerine karşılık gelen bileşenlerinin isimleri verilmiştir.

Tablo 3.13: Elasticsearch ve RDBMS'in birbirlerine benzerlik gösteren temel bileşenleri.

RDBMS	Elasticsearch
Veri tabanı (Database)	Index
Tablo (Table)	Type
Satır (Row)	Document

Elasticsearch'e bir veri kaydedildiğinde, Elasticsearch bu verinin hangi documentlerde geçtiğini daha hızlı bulmak için veriyi kelime bazlı indeksler. Bundan dolayı herhangi bir arama işlemi yapıldığında aranan kelime tüm kayıtlar içinden değil, kelimenin geçtiği documentler'de aranır.

Bu durum da arama işleminin hızlı yapılmasını sağlamaktadır. Şekil 3.30'a bakıldığında kullanıcının blue kelimesini aradığında blue ile ilgili tek kayıt olan blue sweater'ı, red'i aradığında ise red içeren iki kayıt olduğu için Red sweater ve Red pants sonuçlarını döndürdüğü gözükmektedir.



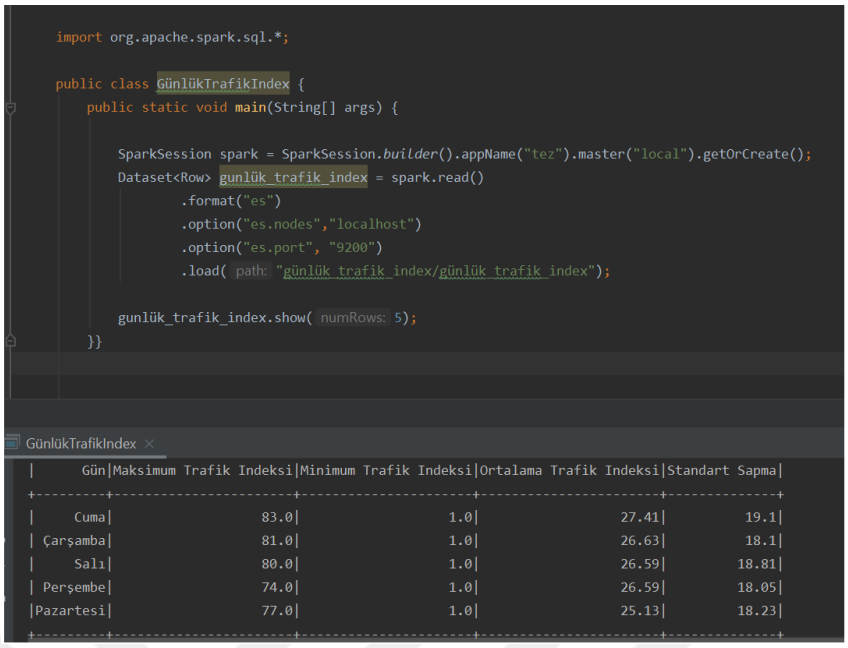
Şekil 3.30: Elasticsearch indexleme örneği [54].

Tez kapsamında işlenmiş veriler Elasticsearch'e atılıp, Kibana'da da Elasticsearch'e atılan bu verilerin tablo ve grafikleri yapılıp, verilerin görüntülenmesi sağlanmıştır. Java kullanarak, Apache Spark ile Elasticsearch'den veri okuma ve yazma işlemleri için gerekli kod parçacıkları tablo 3.14'de verilmiştir.

Tablo 3.14: Apache Spark ile Elasticsearch'den veri okuma ve yazma işlemleri için gerekli kod parçacıkları.

Apache Spark ile Elasticsearch'e veri yazma	Dataset_ismi. write(). format("es") .option("es.nodes","sunucu/sunucuların_ip_adresleri"). option("es.port","elasticsearch_port"). mode(yazma_modu).save(index_ismi/type_ismi)
Apache Spark ile Elasticsearch'den veri okuma	SparkSession_nesnesi_ismi. read(). format("es") .option("es.nodes","sunucu/sunucuların_ip_adresleri"). option("es.port","elasticsearch_port"). load(index_ismi/type_ismi)

Apache Spark ile Elasticsearch'den veri okuyup, yazmak tablo 3.14'deki gibi olmaktadır. Şekil 3.31'de haftanın günlerine göre ortalama trafik indeksi değerlerini içeren verinin Elasticsearch'den Apache Spark ile okunup, ilk 5 sonucunun ekrana yansıtılması gösterilmiştir.



```
import org.apache.spark.sql.*;

public class GünlükTrafikIndex {
    public static void main(String[] args) {

        SparkSession spark = SparkSession.builder().appName("tez").master("local").getOrCreate();
        Dataset<Row> gunluk_trafik_index = spark.read()
                .format("es")
                .option("es.nodes","localhost")
                .option("es.port", "9200")
                .load( path: "günlük_trafik_index/günlük_trafik_index");

        gunluk_trafik_index.show( numRows: 5);
    }}
```

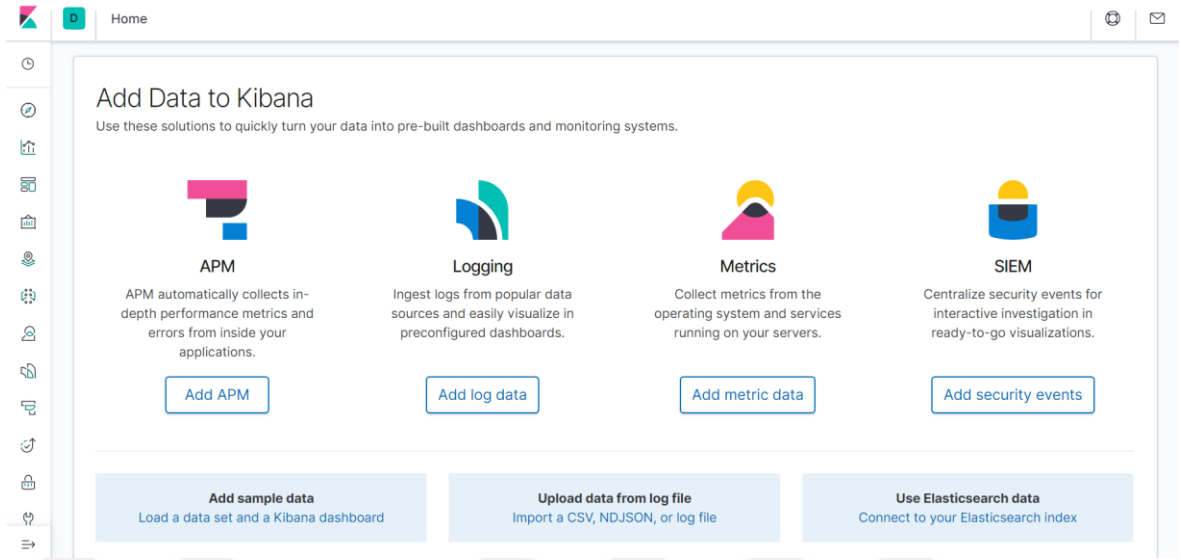
```
GünlükTrafikIndex
|      Gün|Maksimum Trafik Indeksi|Minimum Trafik Indeksi|Ortalama Trafik Indeksi|Standart Sapma|
+---------+-----------------------+----------------------+-----------------------+--------------+
|     Cuma|                   83.0|                   1.0|                  27.41|          19.1|
| Çarşamba|                   81.0|                   1.0|                  26.63|          18.1|
|     Salı|                   80.0|                   1.0|                  26.59|         18.81|
| Perşembe|                   74.0|                   1.0|                  26.59|         18.05|
|Pazartesi|                   77.0|                   1.0|                  25.13|         18.23|
+---------+-----------------------+----------------------+-----------------------+--------------+
```

Şekil 3.31: Günlük_trafik_index type ve index isimli verinin Elasticsearch'den okunması.

3.2.5. Kibana

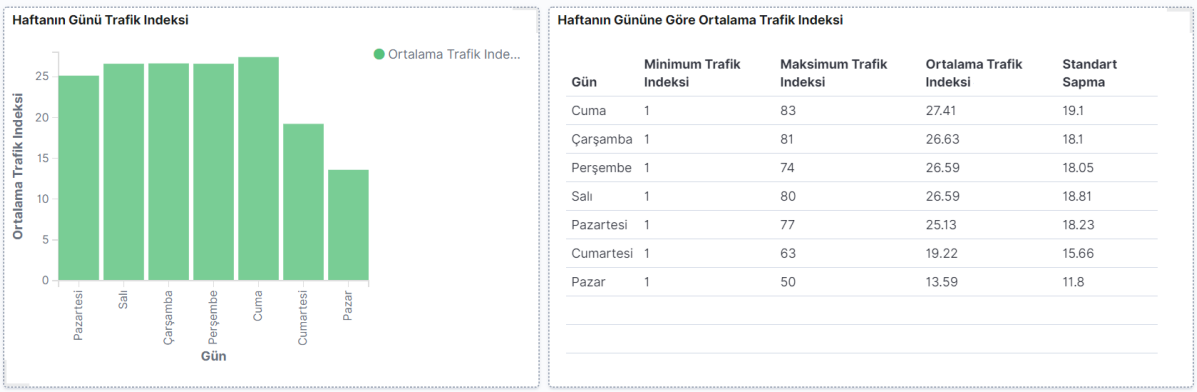
Kibana, Elasticsearch'de tutulan veriler için görselleştirme ve analiz yapılmasına imkân veren bir uygulamadır. Kibana, Elastic Stack denilen ELK yazılımlarıyla yapılan projelerin son kullanıcılar tarafından görüntülenmesini sağlayan bileşendir. ELK ismi Elasticsearch, Logstash ve Kibana uygulamalarının baş harflerinden gelmektedir. ELK ile yapılan projelerde Elasticsearch verileri tutan ve arama işlemlerinin yapılmasını sağlayan, Logstash ise verileri toplamayı, işlemeyi ve taşınmasını sağlayan bileşendir. Kibana da işlenen verinin görselleştirilmesinde kullanılır [55].

Şekil 3.32'de Kibana ara yüzü gözükmektedir.



Şekil 3.32: Kibana ara yüzü.

Tez kapsamında tüm görselleştirmeler Kibana ile yapılmıştır. Bunun için işlenmiş veriler doğrudan Elasticsearch'den Kibana ile okunmuştur. Şekil 3.33'de Haftanın günlerine Ortalama Trafik indeksi değerlerinin Kibana ile oluşturulmuş sütun grafiği örneği ve değerleri gösterilmiştir.



Haftanın Gününe Göre Ortalama Trafik Indeksi

Gün	Minimum Trafik Indeksi	Maksimum Trafik Indeksi	Ortalama Trafik Indeksi	Standart Sapma
Cuma	1	83	27.41	19.1
Çarşamba	1	81	26.63	18.1
Perşembe	1	74	26.59	18.05
Salı	1	80	26.59	18.81
Pazartesi	1	77	25.13	18.23
Cumartesi	1	63	19.22	15.66
Pazar	1	50	13.59	11.8

Şekil 3.33: Kibana ile haftanın günlerinin ortalama trafik indeksi değerlerinin sütun grafiği ve tablosu.

4. BULGULAR

Bu bölümde, öncelikle 2020 yılı için genel trafik analizi sonuçları ve sokağa çıkma yasaklarının trafik durumuna etkisi ile ilgili sonuçlar paylaşılmıştır. Bölümün son kısmında, trafik durumu tahmini için lojistik regresyon, rastgele orman, karar ağaçları, gradyan yükseltme ağacı tabanlı OneVsRest ve doğrusal destek vektör makinesi tabanlı OneVsRest ile tasarlanan makine öğrenmesi uygulamalarının sonuçlarına yer verilmiştir.

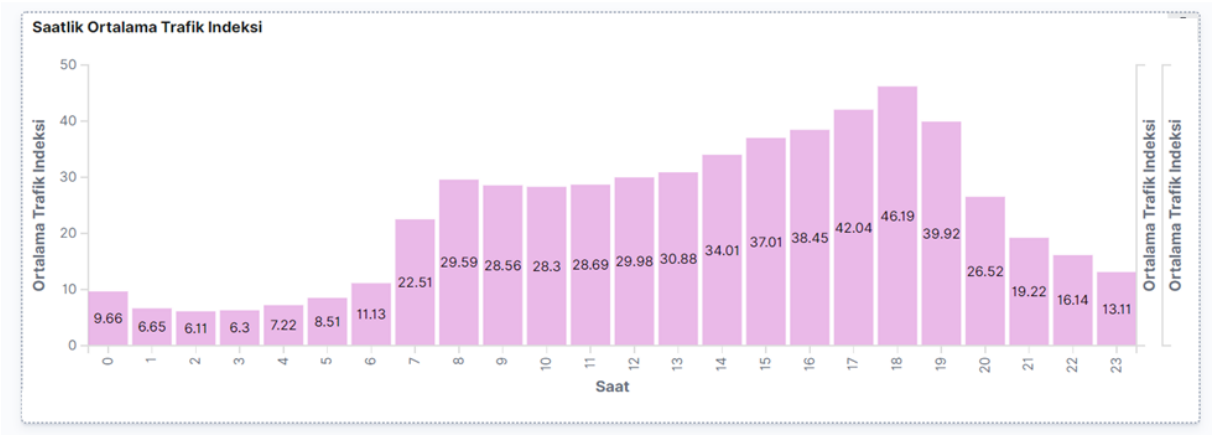
4.1. 2020 YILI İSTANBUL TRAFİK DURUMU ANALİZ SONUÇLARI

2020 yılı için saat bazında yapılan analiz sonucu tablo 4.1'de, grafiği ise şekil 4.1'de gösterilmiştir.

Tablo 4.1: İstanbul 2020 yılı saatlik bazda ortalama trafik indeksi sıralaması.

Saatlik Ortalama Trafik Indeksi Tablosu

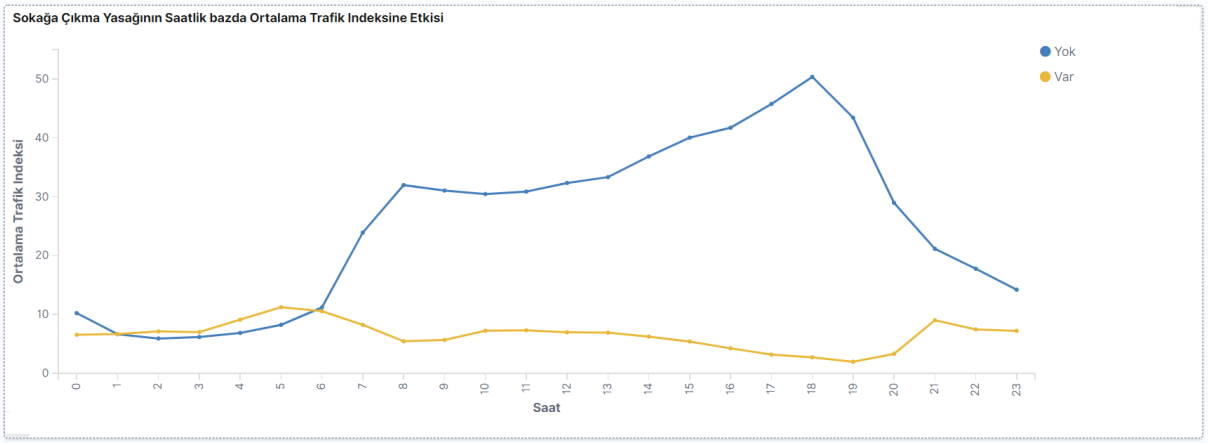
Saat	Minimum Trafik Indeksi	Maksimum Trafik Indeksi	Ortalama Trafik Indeksi	Standart Sapma
18	1	83	46.19	21.5
17	1	77	42.04	18.79
19	1	75	39.92	20.06
16	1	69	38.45	16.49
15	1	66	37.01	15.24
14	1	65	34.01	14.13
13	1	63	30.88	13.3
12	1	56	29.98	13.47
8	1	74	29.59	18.71
11	1	56	28.69	13.85
9	1	67	28.56	16.83
10	1	62	28.3	14.74
20	1	57	26.52	13.9
7	1	57	22.51	12.82
21	1	74	19.22	9.26
22	1	39	16.14	7.82
23	1	35	13.11	6.43
6	1	27	11.13	4.16
0	1	29	9.66	4.7
5	1	22	8.51	3.88
4	1	18	7.22	3.65
1	1	28	6.65	3.27
3	1	19	6.3	3.53
2	1	19	6.11	3.29



Şekil 4.1: İstanbul 2020 yılı saatlik bazda ortalama trafik indeksi grafiği.

Tablo 4.1'e bakıldığında, trafiğin özellikle eve dönüş saatlerinde olduğu göze çarpmaktadır. 2020 yılı için trafiğin en yoğun olduğu saat ise 18 olarak gözükmektedir.

Saatlik bazda trafik analizini biraz daha detaylandırıp, sokağa çıkma yasağının olup olmaması durumuna göre ayırdığımızda ise elde edilen grafik şekil 4.2'de görüldüğü gibi olmaktadır.



Şekil 4.2: İstanbul 2020 yılı sokağa çıkma yasaklarının saatlik bazda ortalama trafik indeksine etkisi grafiği.

Şekil 4.2'e baktığımızda sokağa çıkma yasağının olmadığı saatlerdeki durum, saatlik genel analizdeki ortalama trafik indeksi durumuna oldukça benzemektedir. Gene eve dönüş saatleri, trafik indeksinin en yoğun olduğu saatler olmuştur. Sokağa çıkma yasağının olduğu saatlerde beklenildiği gibi tüm saatler, oldukça düşük trafik indeksi değerlerine sahiptir. Saat 0-6 arası, sokağa çıkma yasağının olmadığı durumda trafik indeksi değerleri düşük olduğu için, bu saat aralığında sokağa çıkma yasağının olduğu ve olmadığı durumlarda ortalama trafik indeksi değerleri oldukça yakın seyretmiştir.

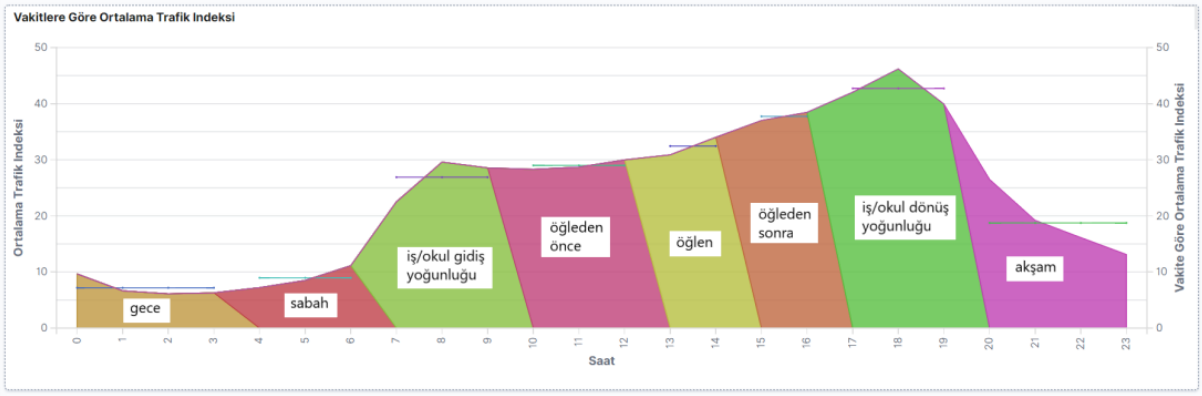
Yapılan saatlik analizi, zaman dilimlerine göre ayırdığımızda tablo 4.2'deki sonuçlar elde edilmiştir. Tabloya bakıldığında, trafiğin en yoğun olduğu zaman diliminin iş/okul çıkış vakitlerinde yaşandığı görülmektedir. Gece vakti ise trafik yoğunluğunun en az olduğu zaman dilimi olarak göze çarpmaktadır.

Tablo 4.2: İstanbul 2020 yılı vakitlere göre ortalama trafik indeksi sıralaması.

Vakitlere Göre Ortalama Trafik Indeksi Tablosu

Vakit	Minimum Trafik Indeksi	Maksimum Trafik Indeksi	Ortalama Trafik Indeksi	Standart Sapma
iş/okul dönüş yoğunluğu(17:00-19:59)	1	83	42.71	20.3
öğleden sonra(15:00-16:59)	1	69	37.73	15.88
öğlen(13:00-14:59)	1	65	32.44	13.8
öğleden önce(10:00-12:59)	1	62	28.99	14.04
iş/okul gidiş yoğunluğu(7:00-9:59)	1	74	26.89	16.59
akşam(20:00-23:59)	1	74	18.75	10.96
sabah(4:00-6:59)	1	27	8.96	4.23
gece(00:00-03:59)	1	29	7.18	4.01

Zaman dilimlerine göre yapılan analiz, alan grafiği ile ifade edildiğinde şekil 4.3'deki sonuç elde edilmektedir.



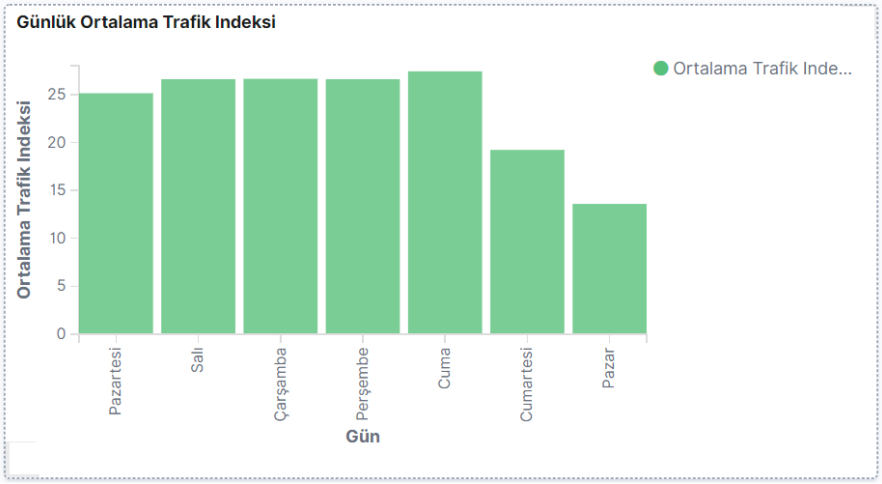
Şekil 4.3: İstanbul 2020 yılı vakitlere göre ortalama trafik indeksi alan grafiği.

2020 yılı için haftanın günleri bakımından ortalama trafik indeksi tablosu tablo 4.3'de, grafiği ise şekil 4.4'de verilmiştir.

Tablo 4.3: İstanbul 2020 yılı haftanın günlerine göre ortalama trafik indeksi sıralaması.

Günlük Ortalama Trafik Indeksi Tablosu

Gün	Minimum Trafik Indeksi	Maksimum Trafik Indeksi	Ortalama Trafik Indeksi	Standart Sapma
Cuma	1	83	27.41	19.1
Çarşamba	1	81	26.63	18.1
Perşembe	1	74	26.59	18.05
Salı	1	80	26.59	18.81
Pazartesi	1	77	25.13	18.23
Cumartesi	1	63	19.22	15.66
Pazar	1	50	13.59	11.8



Şekil 4.4: İstanbul 2020 yılı haftanın günlerine göre ortalama trafik indeksi grafiği.

Tablo 4.3'e bakıldığında, hafta içi günlerin ortalama trafik indeksi değerleri birbirine yakın olsa da mesai bitim günü olan Cuma gününün zirvede yer aldığı, hafta sonunun ise düşük değerlere sahip olduğu görülmektedir.

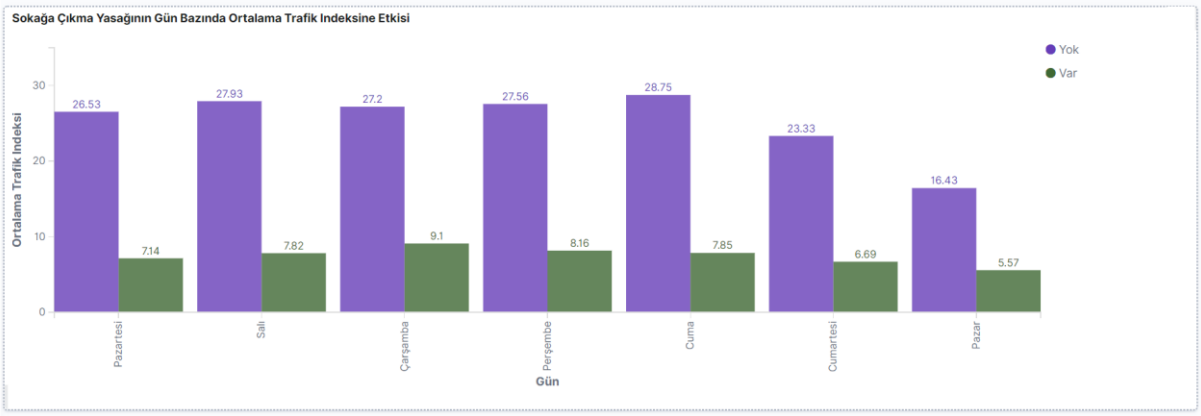
Haftanın günleri bakımından analizimizi de sokağa çıkma yasağının olup olmamasına göre ayırdığımızda tablo 4.4'deki sonuçlar edilmiştir.

Tablo 4.4: İstanbul 2020 yılı sokağa çıkma yasaklarının gün bazında ortalama trafik indeksine etkisi.

Sokağa Çıkma Yasağı - Gün Tablosu

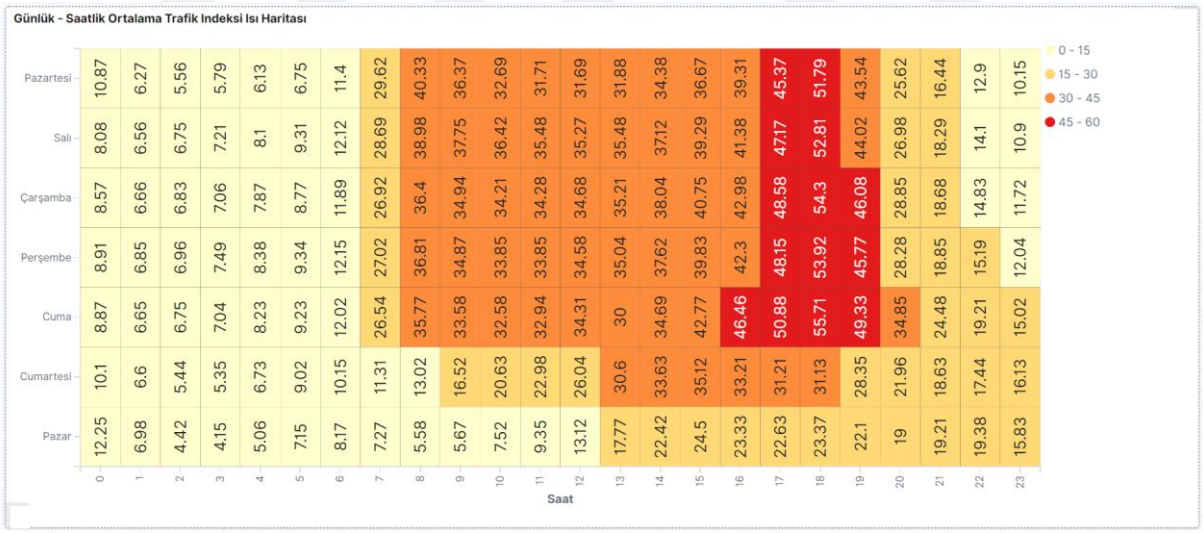
Sokağa Çıkma Yasağı	Gün	Toplam Saat	Minimum Trafik Indeksi	Maksimum Trafik Indeksi	Ortalama Trafik Indeksi	Standart Sapma
Yok	Cuma	1,168	1	83	28.75	18.99
Yok	Salı	1,165	1	80	27.93	18.73
Yok	Perşembe	1,208	1	74	27.56	17.99
Yok	Çarşamba	1,232	1	81	27.2	18.11
Yok	Pazartesi	1,158	1	77	26.53	18.17
Yok	Cumartesi	940	1	63	23.33	15.93
Yok	Pazar	922	1	50	16.43	12.32
Var	Çarşamba	40	5	16	9.1	2.85
Var	Perşembe	64	2	14	8.16	2.78
Var	Cuma	80	1	26	7.85	4.25
Var	Salı	83	1	17	7.82	4.28
Var	Pazartesi	90	1	20	7.14	3.65
Var	Cumartesi	308	1	17	6.69	3.34
Var	Pazar	326	1	22	5.57	4.14

Sokağa çıkma yasakları sadece tam gün olarak değil, bazı günlerde saatlik olarak yapıldığı için tablo 4.4'deki sonuçlara toplam saat kolonu da eklenmiştir. Saatlik olarak sokağa çıkma yasağına sahip olmayan hafta içi günler için ortalama trafik indeksi değerleri, günlük genel ortalama trafik indeksi analizinde elde edilen değerlere oldukça yakın olduğu gözükmektedir. Fakat cumartesi ve pazar günleri için aynı durum geçerli olmamıştır. Saatlik olarak sokağa çıkma yasağına sahip olmayan cumartesi ve pazar günlerinin trafik indeksi değerleri, tablo 4.3'deki değerlere göre %20'den daha fazla yüksek değerlere sahip olmuştur. Bunun en önemli sebebi, sokağa çıkma yasaklarının özellikle cumartesi ve pazar günü olmasından kaynaklanmaktadır. Bu durum tablo 4.4'de gözükmektedir. 2020 yılı için cumartesi günlerine toplam 308 saat, pazar günlerine ise 326 saat sokağa çıkma yasağı uygulanmıştır. Hafta içi ise toplam 90 saat ile en fazla sokağa çıkma yasağı pazartesi günlerine uygulanmıştır ki bu değer cumartesi ve pazar günlerine uygulan sokağa çıkma yasaklarının 3,5 da biridir. Hafta sonlarının, sokağa çıkma yasağının olup olmadığı iki durum için de en düşük trafik indeksi değerlerine sahip olduğu gözükmektedir. Sokağa çıkma yasağının olup olmamasına göre, haftanın günleri bazında ortalama trafik indeksi için sütun grafiği şekil 4.5'de verilmiştir.



Şekil 4.5: İstanbul 2020 yılı sokağa çıkma yasaklarının, haftanın günlerine göre ortalama trafik indeksine etkisi grafiği.

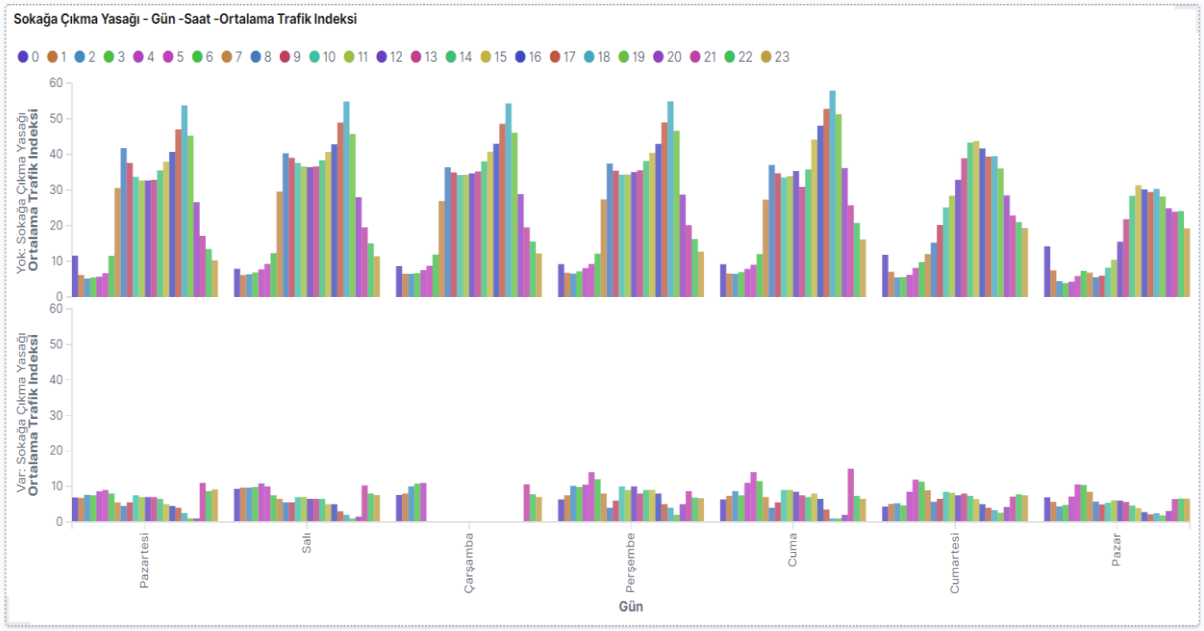
Sokağa çıkma yasaklarının farklı saat dilimlerine yığılmasından dolayı; yasakların, gün bazında ortalama trafik indeksi etkisine doğrudan bu açıdan bakmak karşılaştırma açısından pek uygun olmamaktadır. Yukarıda yapılan analizi gün - saat olarak yapmak karşılaştırma açısından daha uygun olmaktadır. Şekil 4.6'de 2020 yılı için gün-saat bazında ortalama trafik indeksi için ısı haritası verilmiştir.



Şekil 4.6: İstanbul 2020 yılı gün–saat bazda ortalama trafik indeksi ısı haritası.

Şekil 4.6'a bakıldığında, hafta içi tüm günlerde ortalama trafik indeksinin en yüksek olduğu saatin, 18 olduğu gözükmektedir. Ayrıca hafta içi günlerde trafiğin en fazla eve dönüş saatlerinde olduğu sonucuna varılabilir. Hafta sonları ise trafiğin en yoğun olduğu saat 15 olarak görülmektedir.

Ortalama trafik indeksinin gün–saat bazında, sokağa çıkma yasağının olup olmamasına göre durumu ise şekil 4.7'de görülmektedir.



Şekil 4.7: İstanbul 2020 yılı sokağa çıkma yasaklarının gün–saat bazda ortalama trafik indeksine etkisi grafiği.

Şekil 4.7'e bakıldığında, sokağa çıkma yasağının olmadığı durumda, hafta içindeki gün–saat trafik durumlarının oldukça benzerlik gösterdiği görülmektedir. Sokağa çıkma yasağının olduğu hafta içi günlerde ise en yüksek trafiğin 21'de olduğu görülmektedir. Burada göze çarpan başka detay ise 2020 yılında, sokağa çıkma yasağına tüm saat dilimlerinde sahip olmayan tek günün çarşamba günü olmasıdır.

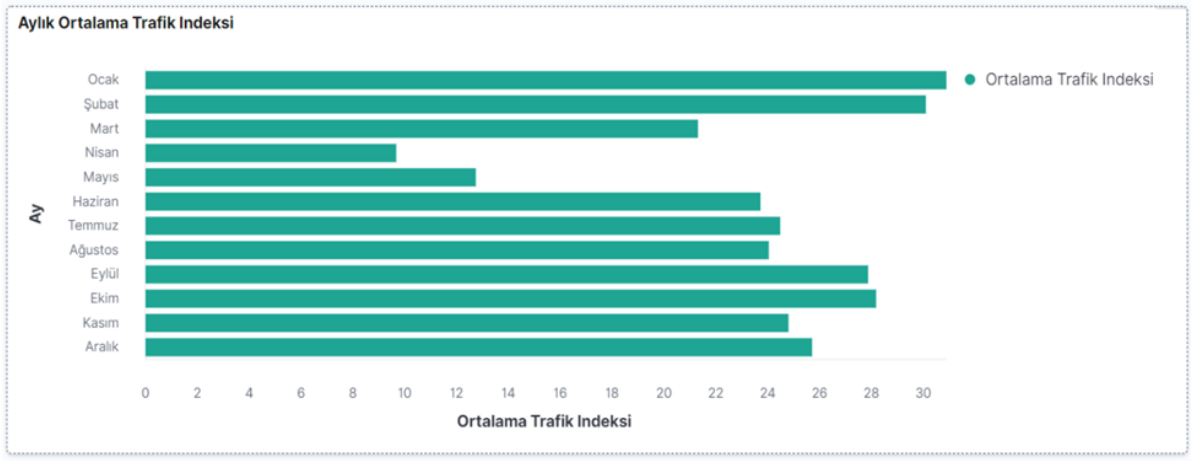
Buraya kadar 2020 yılı için gün ve saat bazında genel sonuçlar ile sokağa çıkma yasaklarının bu genel duruma etkisi gösterilmeye çalışılmıştır. Şimdi ise Covid- 19'un etkilerini görmek için ay bazlı analiz sonuçları gösterilecektir.

2020 yılı için ay bazında ortalama trafik indeksi tablosu, tablo 4.5'de ve grafiği şekil 4.8'de verilmiştir.

Tablo 4.5: İstanbul 2020 yılı aylara göre ortalama trafik indeksi sıralaması.

Aylık Ortalama Trafik Indeksi Tablosu

Ay	Minimum Trafik Indeksi	Maksimum Trafik Indeksi	Ortalama Trafik Indeksi	Standart Sapma
Ocak	2	80	30.89	19.03
Şubat	2	81	30.1	18.42
Ekim	1	83	28.18	19.24
Eylül	1	64	27.87	17.96
Aralık	1	79	25.71	21.83
Kasım	1	69	24.8	17.38
Temmuz	1	61	24.48	15.28
Ağustos	1	63	24.04	16.1
Haziran	1	65	23.72	16.65
Mart	1	74	21.31	17.02
Mayıs	1	63	12.74	11.87
Nisan	1	36	9.67	5.62

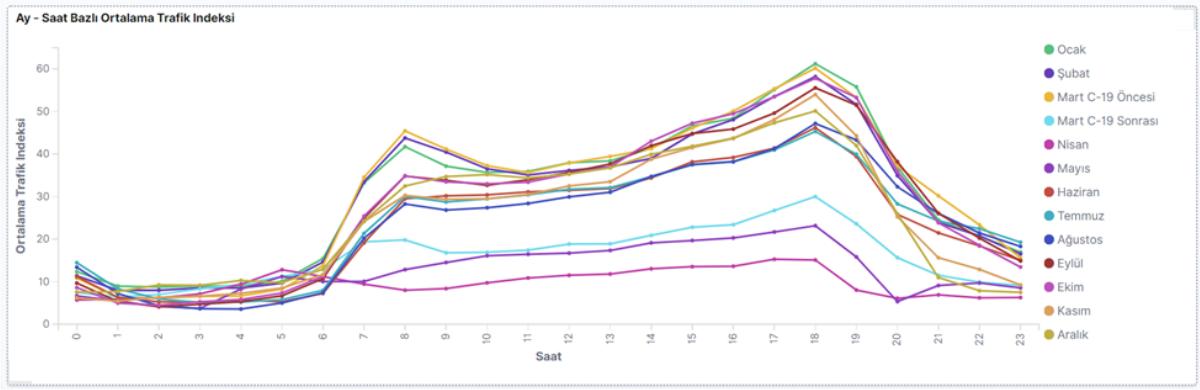


Şekil 4.8: İstanbul 2020 yılı aylara göre ortalama trafik indeksi grafiği.

Tablo 4.5'e bakıldığında ocak ve şubat aylarının ortalama trafik indeksi değerlerinin birbirlerine yakın ve 30 civarında olduğu gözükmektedir. Mart ayında ise bu değer 21 olarak ölçülmüştür. Mart ayını koronavürüs öncesi ve sonrası olarak ayırırsak (ilk vakanın açıklandığı 11 Mart ve öncesi koranavirüs öncesi, 11 Marttan ay sonuna kadar olan durum ise koronavirüs sonrası olarak isimlendirilmiştir.) koronavirüs öncesi trafik indeksi 31, koronavirüs sonrası ise 16 olarak tespit edilmiştir. Nisan ayında ise okulların kapatalı tutulması, çoğu şirketin evden çalışmaya geçmesi ve sokağa çıkma yasaklarıyla beraber ortalama trafik indeksi değerinin 10'a

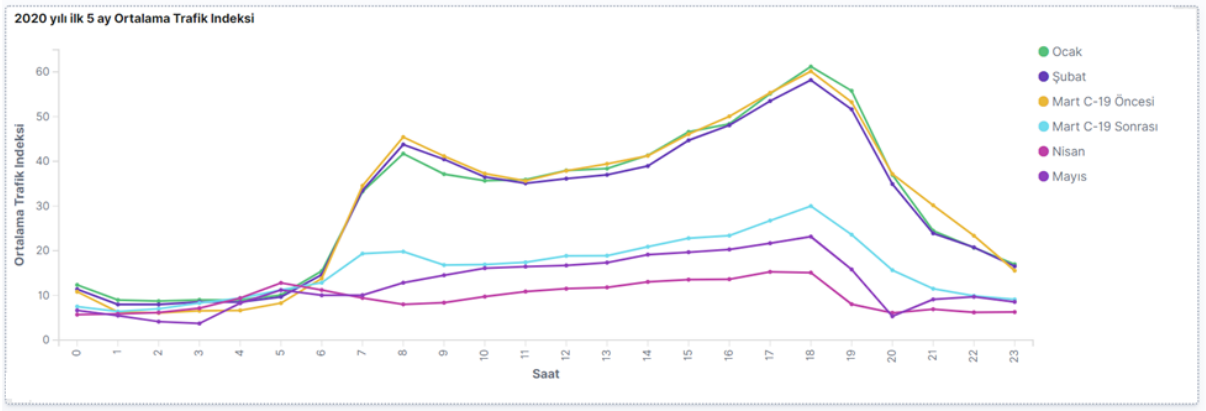
kadar düştüğü gözükmektedir. 2020 yılında en düşük ortalama trafik indeksi de nisan ayına aittir.

Bu analizimizi biraz daha detaylandırıp, ay–saat bazında yaptığımızda şekil 4.9'da görülen grafik elde edilmektedir.(ilk vakanın açıklandığı 11 Mart ve öncesi Mart C-19 öncesi, 11 Marttan ay sonuna kadar olan durum ise Mart C-19 sonrası olarak isimlendirilmiştir.)



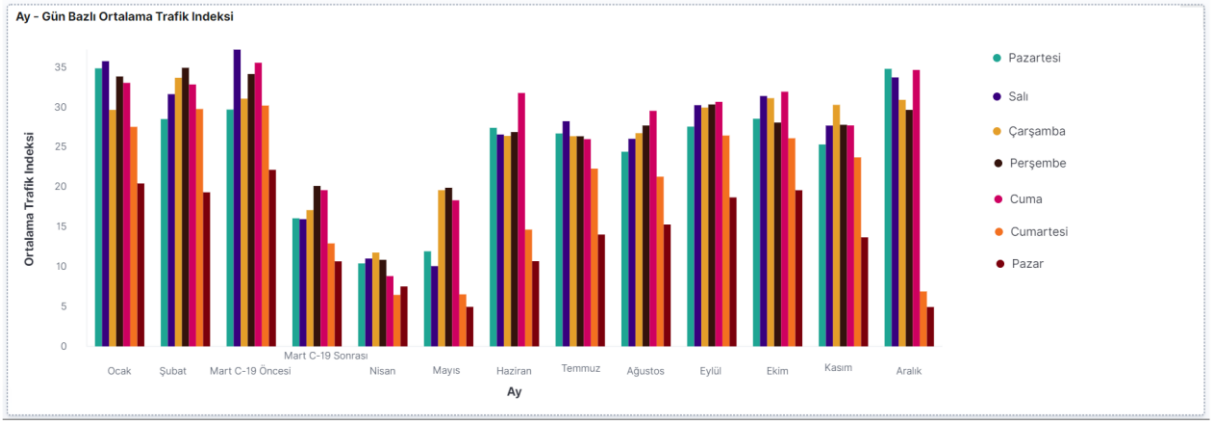
Şekil 4.9: İstanbul 2020 yılı ay–saat bazlı ortalama trafik indeksi grafiği.

Şekil 4.9'a baktığımızda, tüm aylarda trafiğin en yoğun olduğu saatin, 18 olduğu görülmektedir. Özellikle Mart C-19 öncesinden; Mart C-19 sonrasına geçişte, hemen hemen tüm saat dilimlerinde trafik ciddi derece düşüş yaşamıştır. Şekil 4.8'de Nisan ayının 2020 senesi için en düşük ortalama trafik indeksine sahip olduğu belirtilmişti, şekil 4.9'a bakıldığında Nisan ayı tüm saat dilimlerinde de düşük ortalama trafik indeksi değerlerine sahip olmuştur. Haziran ayından itibaren artış gösteren trafik indeksi değerleri, eylül ve ekim aylarındaki 28 trafik indeksi değeriyle; Covid-19'un etkisini gösterdiği aylar içerisinde trafiğin en yüksek ortalama trafik indeksi değerine sahip olduğu duruma ulaşmıştır. Şekil 4.10'da Covid-19'un ilk etkilerini daha detaylı görmek için ilk 5 ay özelinde ay–saat bazlı çizgi grafiği verilmiştir



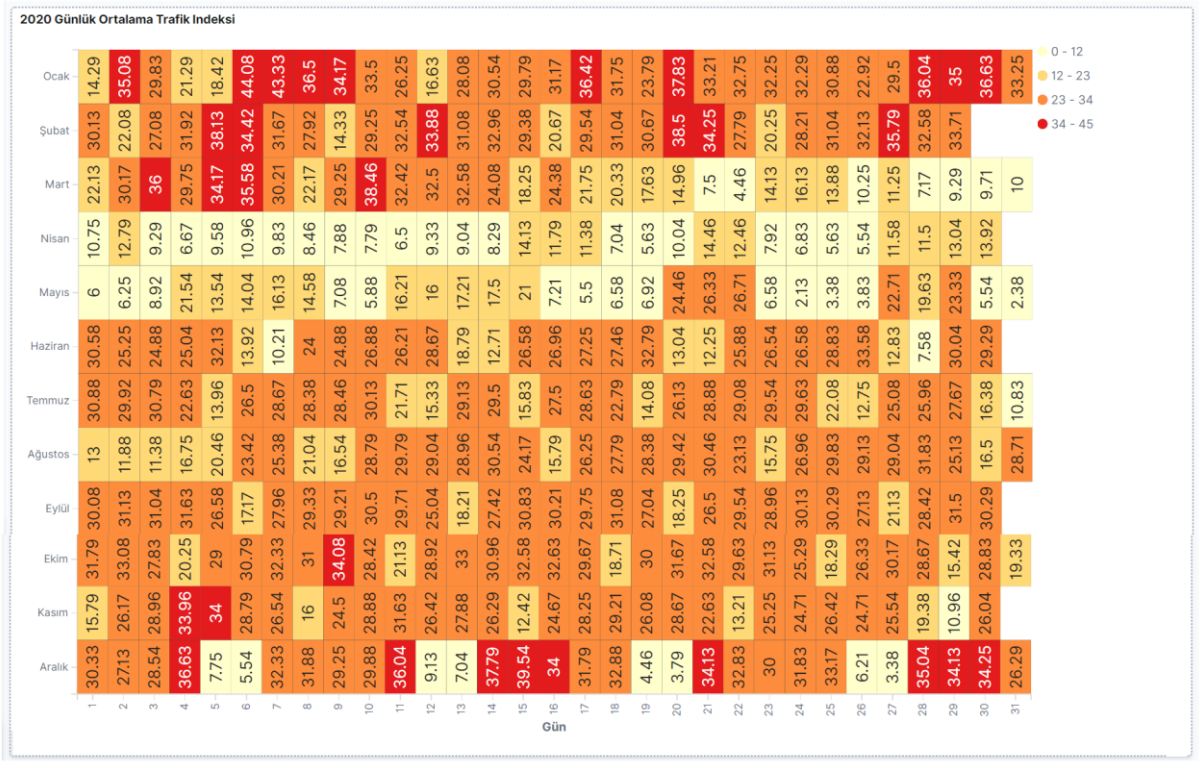
Şekil 4.10: İstanbul 2020 yılı ilk 5 ay için ay–saat bazlı ortalama trafik indeksi grafiği.

Covid-19'un etkilerini gün bazında da görmek için şekil 4.11'de ay–gün bazlı ortalama trafik indeksi sütün grafiği verilmiştir. Bu grafikte de Mart C-19 sonrası, Nisan ve Mayıs aylarındaki düşüş gün bazlı da görülebilmektedir.



Şekil 4.11: İstanbul 2020 yılı ay–gün bazlı ortalama trafik indeksi grafiği.

Şekil 4.12'de 2020 yılı için İstanbul geneli ortalama trafik indeksine ait ısı haritası verilmiştir.



Şekil 4.12: İstanbul 2020 yılı ortalama trafik indeksi ısı haritası.

4.2. TRAFİK DURUMU TAHMİNİ MAKİNE ÖĞRENMESİ UYGULAMALARI

Çalışmanın bundan sonraki kısmı trafik durumu tahmini için tasarlanan makine öğrenmesi uygulamalarının sonuçları ile ilgilidir. Makine öğrenmesi uygulamalarında en çok kullanılan sınıflandırma metotlarından olan lojistik regresyon, karar ağaçları, rastgele orman, gradyan yükseltme ağacı tabanlı OneVsRest ve doğrusal destek vektör makinesi tabanlı OneVsRest metotları kullanılmıştır. Özellik kolonları olarak ay, ayın günü, gün, saat, sokağa çıkma yasağının ve resmi tatil olup olmaması alınmıştır. Sınıflandırma metodu kullanılacağı için; etiket kolonu, trafik indeksinin aşağıdaki şekilde 3 kategoriye ayrılmasıyla oluşturulmuştur.

0<trafik indeksi<=20 →1: Normal düzeyde trafik olduğunu ifade eder.

20<trafik indeksi<=50 →2: Trafiğin orta düzeyde olduğunu ifade eder.

50<trafik indeksi<=100 →3: Trafiğin yoğun olduğunu ifade eder.

Tablo 4.6'da eğitim ve test verisinin dağılımı gözükmektedir.

Tablo 4.6: Makine öğrenmesi uygulamalarında kullanılan eğitim ve test verisinin dağılımı.

Toplam Veri	8784
Eğitim Verisi	6193
Test Verisi	2591

4.2.1. Lojistik Regresyon Yöntemi İle Trafik Durumu Tahmini

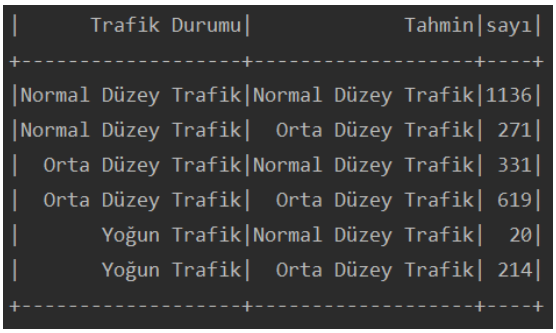
Trafik durumu tahmini için lojistik regresyon yöntemi kullanıldığında modelin doğruluk oranı %67 gelmiştir. Şekil 4.13'de lojistik regresyon yöntemiyle elde edilen örnek bir sonuç gösterilmiştir. Açıklayıcı olması adına kolonlarda düzenlemeler yapılmıştır.

ay	AyinGünü	gün	saat	SokagaCikmaYasagi	ResmiTatil	Trafik Durumu	Tahmin
1	11	6	5	0	0	Normal Düzey Trafik	Normal Düzey Trafik
1	18	6	7	0	0	Normal Düzey Trafik	Normal Düzey Trafik
1	22	3	21	0	0	Orta Düzey Trafik	Orta Düzey Trafik
1	22	3	23	0	0	Normal Düzey Trafik	Orta Düzey Trafik
1	27	1	3	0	0	Normal Düzey Trafik	Normal Düzey Trafik
1	27	1	8	0	0	Orta Düzey Trafik	Normal Düzey Trafik
2	9	7	16	0	0	Orta Düzey Trafik	Orta Düzey Trafik
2	19	3	0	0	0	Normal Düzey Trafik	Normal Düzey Trafik
3	5	4	19	0	0	Yoğun Trafik	Orta Düzey Trafik
3	8	7	21	0	0	Orta Düzey Trafik	Orta Düzey Trafik
3	9	1	20	0	0	Orta Düzey Trafik	Orta Düzey Trafik
3	14	6	4	0	0	Normal Düzey Trafik	Normal Düzey Trafik
4	22	3	23	0	0	Normal Düzey Trafik	Orta Düzey Trafik
5	5	2	19	0	0	Normal Düzey Trafik	Orta Düzey Trafik
6	2	2	15	0	0	Orta Düzey Trafik	Orta Düzey Trafik
6	5	5	18	0	0	Yoğun Trafik	Orta Düzey Trafik
6	12	5	3	0	0	Normal Düzey Trafik	Normal Düzey Trafik
6	25	4	9	0	0	Orta Düzey Trafik	Normal Düzey Trafik
7	3	5	1	0	0	Normal Düzey Trafik	Normal Düzey Trafik
7	5	7	18	0	0	Orta Düzey Trafik	Orta Düzey Trafik

Lojistik Regresyon Doğruluk oranı: 0.6773446545735238

Şekil 4.13: Trafik durumunu lojistik regresyon yöntemi ile tahmin eden modelin örnek çıktısı ve doğruluk oranı.

Şekil 4.14'de lojistik regresyon yöntemiyle oluşturulan modelin sonuçlarının genel özeti verilmiştir.



Trafik Durumu	Tahmin	sayı
Normal Düzey Trafik	Normal Düzey Trafik	1136
Normal Düzey Trafik	Orta Düzey Trafik	271
Orta Düzey Trafik	Normal Düzey Trafik	331
Orta Düzey Trafik	Orta Düzey Trafik	619
Yoğun Trafik	Normal Düzey Trafik	20
Yoğun Trafik	Orta Düzey Trafik	214

Şekil 4.14: Lojistik regresyon yöntemiyle yapılan modelin özet sonuçları.

Şekil 4.14'deki lojistik regresyon modelinin sonuçlarının, karışıklık matrisi ile gösterimi aşağıda gözükmektedir.

Tablo 4.7: Lojistik regresyon yöntemiyle yapılan modelin karışıklık matrisi.

	Normal Düzey Trafik	Orta Düzey Trafik	Yoğun Trafik
Normal Düzey Trafik	1136	271	0
Orta Düzey Trafik	331	619	0
Yoğun Trafik	20	214	0

Lojistik regresyon modelinin; 3 kategoriye ayırdığımız trafik durumunun tahmini konusunda, yoğun trafik durumu tahmininde başarısız olduğu göze çarpmaktadır. Bu yöntem yoğun trafik durumu tahmininde hiçbir doğru sonuçta bulunamamıştır. Modelimiz orta düzey trafik tahmini konusunda %56'lık kesinlik ve %65 hassasiyet değerleriyle bu düzeydeki trafik tahmini konusunda da başarılı sonuçlar elde edememiştir. Lojistik regresyon yöntemiyle oluşturulan modelin trafik tahmini konusunda başarı metrikleri aşağıda verilmiştir.

Tablo 4.8: Lojistik regresyon yöntemiyle yapılan modelin başarı metrikleri.

	Toplam Sayı	Doğru Sayısı	Yanlış Sayısı	Kesinlik	Hassasiyet	F1 Skoru
Normal Düzey Trafik	1407	1136	271	0.7639	0.8073	0.7850
Orta Düzey Trafik	950	619	331	0.5606	0.6515	0.6027
Yoğun Trafik	234	0	234	0	0	0

4.2.2. Karar Ağaçları Yöntemi İle Trafik Durumu Tahmini

İlgili makine öğrenmesi uygulaması, karar ağaçları yöntemiyle yapıldığında doğruluk oranı %81 gelmektedir. Şekil 4.15'de karar ağaçları yöntemiyle elde edilen örnek bir sonuç

gösterilmiştir. Açıklayıcı olması adına doğrudan modelden elde edilen veri setinde, kolon isimlerini değiştirme ve bazı kolonları silme işlemi yapılmıştır.

ay	AyinGünü	gün	saat	SokagaCikmaYasagi	ResmiTatil	Trafik Durumu	Tahmin
1	11	6	5	0	0	Normal Düzey Trafik	Normal Düzey Trafik
1	18	6	7	0	0	Normal Düzey Trafik	Orta Düzey Trafik
1	22	3	21	0	0	Orta Düzey Trafik	Normal Düzey Trafik
1	22	3	23	0	0	Normal Düzey Trafik	Normal Düzey Trafik
1	27	1	3	0	0	Normal Düzey Trafik	Normal Düzey Trafik
1	27	1	8	0	0	Orta Düzey Trafik	Orta Düzey Trafik
2	9	7	16	0	0	Orta Düzey Trafik	Normal Düzey Trafik
2	19	3	0	0	0	Normal Düzey Trafik	Normal Düzey Trafik
3	5	4	19	0	0	Yoğun Trafik	Yoğun Trafik
3	8	7	21	0	0	Orta Düzey Trafik	Orta Düzey Trafik
3	9	1	20	0	0	Orta Düzey Trafik	Orta Düzey Trafik
3	14	6	4	0	0	Normal Düzey Trafik	Normal Düzey Trafik
4	22	3	23	0	0	Normal Düzey Trafik	Normal Düzey Trafik
5	5	2	19	0	0	Normal Düzey Trafik	Yoğun Trafik
6	2	2	15	0	0	Orta Düzey Trafik	Orta Düzey Trafik
6	5	5	18	0	0	Yoğun Trafik	Yoğun Trafik
6	12	5	3	0	0	Normal Düzey Trafik	Normal Düzey Trafik
6	25	4	9	0	0	Orta Düzey Trafik	Orta Düzey Trafik
7	3	5	1	0	0	Normal Düzey Trafik	Normal Düzey Trafik
7	5	7	18	0	0	Orta Düzey Trafik	Yoğun Trafik

Karar Ağacı Yöntemi Doğruluk oranı: 0.8101119258973369

Şekil 4.15: Trafik durumunu karar ağaçları ile tahmin eden modelin örnek çıktısı ve doğruluk oranı.

Şekil 4.16'da karar ağacıyla oluşturulan modelin sonuçlarının genel özeti verilmiştir.

Trafik Durumu	Tahmin	sayı
Normal Düzey Trafik	Orta Düzey Trafik	188
Normal Düzey Trafik	Normal Düzey Trafik	1179
Normal Düzey Trafik	Yoğun Trafik	40
Orta Düzey Trafik	Normal Düzey Trafik	72
Orta Düzey Trafik	Orta Düzey Trafik	767
Orta Düzey Trafik	Yoğun Trafik	111
Yoğun Trafik	Normal Düzey Trafik	1
Yoğun Trafik	Orta Düzey Trafik	80
Yoğun Trafik	Yoğun Trafik	153

Şekil 4.16: Karar ağacı yöntemiyle yapılan modelin özet sonuçları.

Karar ağacı yöntemiyle oluşturulan modelin karışıklık matrisi aşağıda gözükmektedir.

Tablo 4.9: Karar ağaçları yöntemiyle yapılan modelin karışıklık matrisi.

	Normal Düzey Trafik	Orta Düzey Trafik	Yoğun Trafik
Normal Düzey Trafik	1179	188	40
Orta Düzey Trafik	72	767	111
Yoğun Trafik	1	80	153

Karar ağacı yöntemiyle oluşturulan modelin; yoğun trafik durumu tahmininde %50 kesinlik ve %65 hassasiyet değerleriyle fena olmayan bir sonuç elde ettiği gözükmektedir. Bu yöntem orta düzey trafik tahmini konusunda ise %74 kesinlik ve %80 hassasiyet değerleriyle iyi bir sonuç elde etmiştir. Karar ağacı yöntemiyle oluşturulan modelin başarı metrikleri aşağıda verilmiştir.

Tablo 4.10: Karar ağaçları yöntemiyle yapılan modelin başarı metrikleri.

	Toplam Sayı	Doğru Sayısı	Yanlış Sayısı	Kesinlik	Hassasiyet	F1 Skoru
Normal Düzey Trafik	1407	1179	228	0.9416	0.8379	0.8867
Orta Düzey Trafik	950	767	183	0.7410	0.8073	0.7727
Yoğun Trafik	234	153	81	0.5032	0.6538	0.5687

4.2.3. Rastgele Orman Yöntemi İle Trafik Durumu Tahmini

Trafik durumu tahmini için rastgele orman yöntemi kullanıldığında değişik ağaç sayıları için şekil 4.17'deki doğruluk oranları elde edilmiştir.

```
k:1-->0.8101119258973369
k:2-->0.8309532998842146
k:3-->0.8108838286375917
k:4-->0.807024314936318
k:5-->0.8108838286375917
k:6-->0.8058664608259359
k:7-->0.802778849864917
k:8-->0.807024314936318
k:9-->0.8120416827479737
k:10-->0.8016209957545349
```

Şekil 4.17: Trafik durumu tahmininde farklı ağaç sayıları için rastgele orman metodunun doğruluk oranı.

Şekil 4.17'den de görüleceği gibi ağaç sayısı 10'a kadar kontrol edildiğinde; model, ağaç sayısı 2 iken en yüksek doğruluk oranına sahip olmaktadır. Şekil 4.18'de rastgele orman metoduyla oluşturulan modelin örnek çıktısı ve doğruluk oranı verilmiştir.

ay	AyinGünü	gün	saat	SokagaCikmaYasagi	ResmiTatil	Trafik Durumu	Tahmin
1	11	6	5	0	0	Normal Düzey Trafik	Normal Düzey Trafik
1	18	6	7	0	0	Normal Düzey Trafik	Normal Düzey Trafik
1	22	3	21	0	0	Orta Düzey Trafik	Normal Düzey Trafik
1	22	3	23	0	0	Normal Düzey Trafik	Normal Düzey Trafik
1	27	1	3	0	0	Normal Düzey Trafik	Normal Düzey Trafik
1	27	1	8	0	0	Orta Düzey Trafik	Orta Düzey Trafik
2	9	7	16	0	0	Orta Düzey Trafik	Orta Düzey Trafik
2	19	3	0	0	0	Normal Düzey Trafik	Normal Düzey Trafik
3	5	4	19	0	0	Yoğun Trafik	Yoğun Trafik
3	8	7	21	0	0	Orta Düzey Trafik	Orta Düzey Trafik
3	9	1	20	0	0	Orta Düzey Trafik	Orta Düzey Trafik
3	14	6	4	0	0	Normal Düzey Trafik	Normal Düzey Trafik
4	22	3	23	0	0	Normal Düzey Trafik	Normal Düzey Trafik
5	5	2	19	0	0	Normal Düzey Trafik	Yoğun Trafik
6	2	2	15	0	0	Orta Düzey Trafik	Orta Düzey Trafik
6	5	5	18	0	0	Yoğun Trafik	Yoğun Trafik
6	12	5	3	0	0	Normal Düzey Trafik	Normal Düzey Trafik
6	25	4	9	0	0	Orta Düzey Trafik	Orta Düzey Trafik
7	3	5	1	0	0	Normal Düzey Trafik	Normal Düzey Trafik
7	5	7	18	0	0	Orta Düzey Trafik	Orta Düzey Trafik

Rastgele Orman Yöntemi Doğruluk oranı: 0.8309532998842146

Şekil 4.18: Trafik durumunu rastgele orman yöntemiyle tahmin eden modelin örnek çıktısı ve doğruluk oranı.

Şekil 4.19'da rastgele orman yöntemiyle oluşturulan modelin sonuçlarının genel özeti verilmiştir.

Trafik Durumu	Tahmin	sayı
Normal Düzey Trafik	Orta Düzey Trafik	182
Normal Düzey Trafik	Normal Düzey Trafik	1190
Normal Düzey Trafik	Yoğun Trafik	35
Orta Düzey Trafik	Normal Düzey Trafik	77
Orta Düzey Trafik	Orta Düzey Trafik	817
Orta Düzey Trafik	Yoğun Trafik	56
Yoğun Trafik	Normal Düzey Trafik	1
Yoğun Trafik	Orta Düzey Trafik	87
Yoğun Trafik	Yoğun Trafik	146

Şekil 4.19: Rastgele orman yöntemiyle yapılan modelin özet sonuçları.

Rastgele orman yöntemiyle oluşturulan modelin karışıklık matrisi aşağıda gözükmektedir.

Tablo 4.11: Rastgele orman yöntemiyle yapılan modelin karışıklık matrisi.

	Normal Düzey Trafik	Orta Düzey Trafik	Yoğun Trafik
Normal Düzey Trafik	1190	182	35
Orta Düzey Trafik	77	817	56
Yoğun Trafik	1	87	146

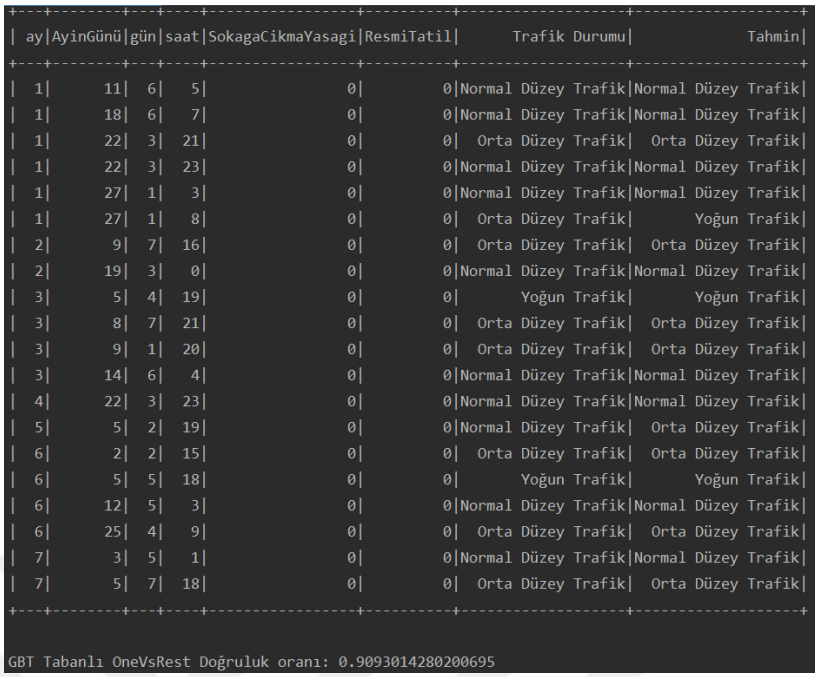
Tablo 4.11'den de görüldüğü gibi rastgele orman yöntemiyle oluşturulan modelin genel olarak iyi sonuçlar elde ettiği gözükmektedir. Bu yöntemle oluşturulan model, yoğun trafik tahmininde %61 kesinlik ve %62 hassasiyet değerleri elde etmiştir. Rastgele orman yöntemiyle oluşturulan modelin başarı metrikleri aşağıda verilmiştir.

Tablo 4.12: Rastgele orman yöntemiyle yapılan modelin başarı metrikleri.

	Toplam Sayı	Doğru Sayısı	Yanlış Sayısı	Kesinlik	Hassasiyet	F1 Skoru
Normal Düzey Trafik	1407	1190	217	0.9384	0.8457	0.8897
Orta Düzey Trafik	950	817	133	0.7523	0.86	0.8025
Yoğun Trafik	234	146	88	0.6160	0.6239	0.6199

4.2.4. Gradyan Yükseltme Ağacı Tabanlı OneVsRest Yöntemi İle Trafik Durumu Tahmini

Trafik durumu tahmini, bundan önceki yöntemlerde doğrudan çoklu sınıfları sınıflandırmayı destekleyen algoritmalarla yapılmıştı. Bu yöntem ise ikili sınıflandırmada kullanılan gradyan yükseltme ağacı yönteminin, OneVsRest yöntemiyle çoklu sınıfları sınıflandırması sağlanarak yapılmıştır. Bu yöntemle yapılan trafik durumu tahminin doğruluk oranı %90 gelmiştir. Şekil 4.20'de bu yöntemle oluşturulan modelin örnek çıktısı ve doğruluk oranı verilmiştir.



ay	AyinGünü	gün	saat	SokagaCikmaYasagi	ResmiTatil	Trafik Durumu	Tahmin
1	11	6	5	0	0	Normal Düzey Trafik	Normal Düzey Trafik
1	18	6	7	0	0	Normal Düzey Trafik	Normal Düzey Trafik
1	22	3	21	0	0	Orta Düzey Trafik	Orta Düzey Trafik
1	22	3	23	0	0	Normal Düzey Trafik	Normal Düzey Trafik
1	27	1	3	0	0	Normal Düzey Trafik	Normal Düzey Trafik
1	27	1	8	0	0	Orta Düzey Trafik	Yoğun Trafik
2	9	7	16	0	0	Orta Düzey Trafik	Orta Düzey Trafik
2	19	3	0	0	0	Normal Düzey Trafik	Normal Düzey Trafik
3	5	4	19	0	0	Yoğun Trafik	Yoğun Trafik
3	8	7	21	0	0	Orta Düzey Trafik	Orta Düzey Trafik
3	9	1	20	0	0	Orta Düzey Trafik	Orta Düzey Trafik
3	14	6	4	0	0	Normal Düzey Trafik	Normal Düzey Trafik
4	22	3	23	0	0	Normal Düzey Trafik	Normal Düzey Trafik
5	5	2	19	0	0	Normal Düzey Trafik	Orta Düzey Trafik
6	2	2	15	0	0	Orta Düzey Trafik	Orta Düzey Trafik
6	5	5	18	0	0	Yoğun Trafik	Yoğun Trafik
6	12	5	3	0	0	Normal Düzey Trafik	Normal Düzey Trafik
6	25	4	9	0	0	Orta Düzey Trafik	Orta Düzey Trafik
7	3	5	1	0	0	Normal Düzey Trafik	Normal Düzey Trafik
7	5	7	18	0	0	Orta Düzey Trafik	Orta Düzey Trafik

GBT Tabanlı OneVsRest Doğruluk oranı: 0.9093014280200695

Şekil 4.20: Trafik durumunu GBT tabanlı OneVsRest yöntemiyle tahmin eden modelin örnek çıktısı ve doğruluk oranı.

GBT tabanlı OneRest yöntemiyle oluşturulan modelin, test verileri tahmini konusundaki sonuçlarının genel özeti şekil 4.21'de gösterilmiştir.

Trafik Durumu	Tahmin	sayı
Normal Düzey Trafik	Orta Düzey Trafik	74
Normal Düzey Trafik	Normal Düzey Trafik	1329
Normal Düzey Trafik	Yoğun Trafik	4
Orta Düzey Trafik	Normal Düzey Trafik	56
Orta Düzey Trafik	Yoğun Trafik	31
Orta Düzey Trafik	Orta Düzey Trafik	863
Yoğun Trafik	Orta Düzey Trafik	70
Yoğun Trafik	Yoğun Trafik	164

Şekil 4.21: GBT tabanlı OneVsRest yöntemiyle yapılan modelin özet sonuçları.

GBT tabanlı OneVsRest yöntemiyle oluşturulan modelin elde ettiği sonuçların karışıklık matrisiyle gösterimi Tablo 4.13'de gözükmektedir.

Tablo 4.13: GBT tabanlı OneVsRest yöntemiyle yapılan modelin karışıklık matrisi.

	Normal Düzey Trafik	Orta Düzey Trafik	Yoğun Trafik
Normal Düzey Trafik	1329	74	4
Orta Düzey Trafik	56	863	31
Yoğun Trafik	0	70	164

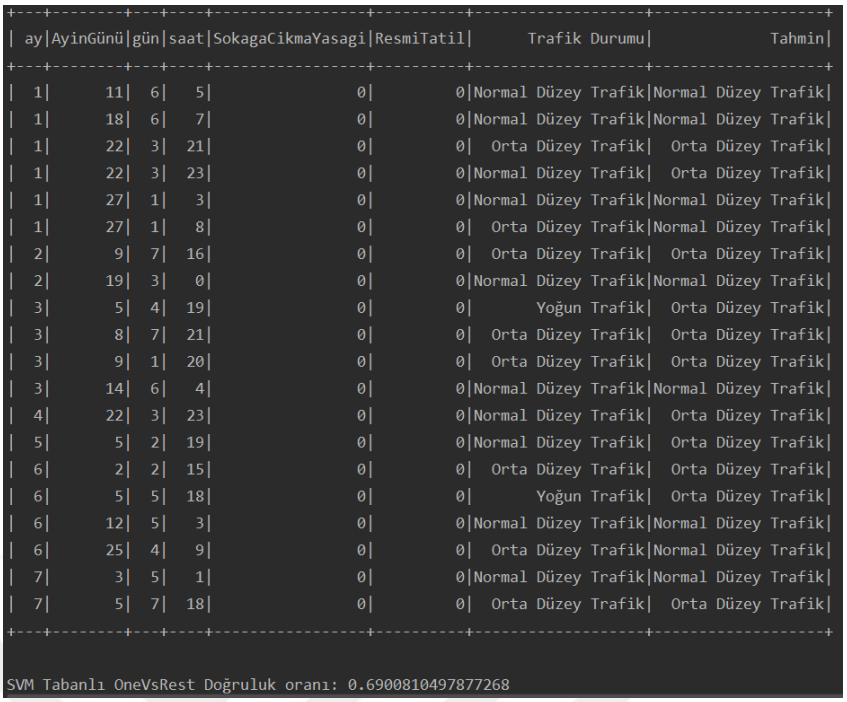
Tablo 4.13'e bakıldığında GBT tabanlı OneVsRest yöntemiyle oluşturulan modelin; tüm trafik durumu sınıflarının tahmininde oldukça iyi sonuçlar elde ettiği gözükmektedir. Yoğun trafik durumu tahminin; diğer durumların tahmininden daha zor olduğu düşünülürse, bu yöntem yoğun trafik tahmini konusunda %82 kesinlik ve %70 hassasiyet değerleriyle öne çıkmaktadır. GBT tabanlı OneVsRest yöntemiyle oluşturulan modelin başarı metrikleri aşağıda verilmiştir.

Tablo 4.14: GBT tabanlı OneVsRest yöntemiyle yapılan modelin başarı metrikleri.

	Toplam Sayı	Doğru Sayısı	Yanlış Sayısı	Kesinlik	Hassasiyet	F1 Skoru
Normal Düzey Trafik	1407	1329	78	0.9595	0.9445	0.9520
Orta Düzey Trafik	950	863	87	0.8570	0.9084	0.8819
Yoğun Trafik	234	164	70	0.8241	0.7008	0.7575

4.2.5. Doğrusal Destek Vektör Makinesi Tabanlı OneVsRest Yöntemi İle Trafik Durumu Tahmini

Trafik durumu tahmininde uzaklık algoritmalarıyla çalışan yöntemlerin nasıl bir sonuç göstereceğini görmek için destek vektör makinesi yöntemi de kullanılmıştır. Bu yöntem Spark MLLIB kütüphanesinde sınıflandırma yöntemi olarak kullanılmak istendiğinde, sadece ikili sınıflandırmayı desteklediği için SVM taban alınıp, OneVsRest yöntemi kullanılmıştır. SVM tabanlı OneVsRest yöntemi yüksek başarı gösterememiş ve doğruluk oranı %69'da kalmıştır. SVM tabanlı OneVsRest yöntemiyle elde edilen modelin örnek çıktıları ve doğruluk oranı şekil 4.22'de verilmiştir.



ay	AyinGünü	gün	saat	SokagaCikmaYasagi	ResmiTatil	Trafik Durumu	Tahmin
1	11	6	5	0	0	Normal Düzey Trafik	Normal Düzey Trafik
1	18	6	7	0	0	Normal Düzey Trafik	Normal Düzey Trafik
1	22	3	21	0	0	Orta Düzey Trafik	Orta Düzey Trafik
1	22	3	23	0	0	Normal Düzey Trafik	Orta Düzey Trafik
1	27	1	3	0	0	Normal Düzey Trafik	Normal Düzey Trafik
1	27	1	8	0	0	Orta Düzey Trafik	Normal Düzey Trafik
2	9	7	16	0	0	Orta Düzey Trafik	Orta Düzey Trafik
2	19	3	0	0	0	Normal Düzey Trafik	Normal Düzey Trafik
3	5	4	19	0	0	Yoğun Trafik	Orta Düzey Trafik
3	8	7	21	0	0	Orta Düzey Trafik	Orta Düzey Trafik
3	9	1	20	0	0	Orta Düzey Trafik	Orta Düzey Trafik
3	14	6	4	0	0	Normal Düzey Trafik	Normal Düzey Trafik
4	22	3	23	0	0	Normal Düzey Trafik	Orta Düzey Trafik
5	5	2	19	0	0	Normal Düzey Trafik	Orta Düzey Trafik
6	2	2	15	0	0	Orta Düzey Trafik	Orta Düzey Trafik
6	5	5	18	0	0	Yoğun Trafik	Orta Düzey Trafik
6	12	5	3	0	0	Normal Düzey Trafik	Normal Düzey Trafik
6	25	4	9	0	0	Orta Düzey Trafik	Normal Düzey Trafik
7	3	5	1	0	0	Normal Düzey Trafik	Normal Düzey Trafik
7	5	7	18	0	0	Orta Düzey Trafik	Orta Düzey Trafik

SVM Tabanlı OneVsRest Doğruluk oranı: 0.6900810497877268

Şekil 4.22: Trafik durumunu SVM tabanlı OneVsRest yöntemiyle tahmin eden modelin örnek çıktısı ve doğruluk oranı.

SVM tabanlı OneVsRest yöntemiyle oluşturulan modelin trafik durumu tahmini için verdiği sonuçların genel özeti şekil 4.23'de verilmiştir.

Trafik Durumu	Tahmin	sayı
Normal Düzey Trafik	Normal Düzey Trafik	1119
Normal Düzey Trafik	Orta Düzey Trafik	288
Orta Düzey Trafik	Normal Düzey Trafik	281
Orta Düzey Trafik	Orta Düzey Trafik	669
Yoğun Trafik	Normal Düzey Trafik	17
Yoğun Trafik	Orta Düzey Trafik	217

Şekil 4.23: SVM tabanlı OneVsRest yöntemiyle yapılan modelin özet sonuçları.

SVM tabanlı OneVsRest yöntemiyle oluşturulan modelin sonuçlarının karışıklık matrisi tablo 4.15'de verilmiştir.

Tablo 4.15: SVM tabanlı OneVsRest yöntemiyle yapılan modelin karışıklık matrisi.

	Normal Düzey Trafik	Orta Düzey Trafik	Yoğun Trafik
Normal Düzey Trafik	1119	288	0
Orta Düzey Trafik	281	669	0
Yoğun Trafik	17	217	0

Tablo 4.15'e bakıldığında, SVM tabanlı OneVsRest yöntemiyle oluşturulan modelin sonuçlarının lojistik regresyon yöntemiyle oluşturulan modelin sonuçlarına oldukça benzediği gözükmektedir. SVM tabanlı OneVsRest yöntemi de lojistik regresyon yönteminde olduğu gibi yoğun trafik tahmini konusunda hiçbir doğru sonuç elde edememiştir. Aşağıda SVM tabanlı OneVsRest ile oluşturulan modelin başarı metrikleri verilmiştir.

Tablo 4.16: SVM tabanlı OneVsRest yöntemiyle yapılan modelin başarı metrikleri.

	Toplam Sayı	Doğru Sayısı	Yanlış Sayısı	Kesinlik	Hassasiyet	F1 Skoru
Normal Düzey Trafik	1407	1119	288	0.7896	0.7953	0.7924
Orta Düzey Trafik	950	669	281	0.5698	0.7042	0.6299
Yoğun Trafik	234	0	234	0	0	0

Modellerin doğruluk oranları karşılaştırıldığında, GBT tabanlı OneVsRest yönteminin diğer modellere göre oldukça başarılı olduğu görülmektedir. Bu yöntemle oluşturulan model, özellikle yoğun trafik durumu tahmini konusunda diğer yöntemlere göre öne çıkmaktadır. Karar ağaçları yöntemiyle oluşturulan model ise tatmin edici başarı metrikleri elde etmiştir. Karar ağacı yöntemiyle elde edilen sonucun doğruluk oranını daha iyi değerlere çıkarmak için 2 ağaçla oluşturulan rasgele orman yöntemi, yoğun trafiğin tahmini konusunda kesinlik değeri bakımından karar ağacı yönteminden iyi olsa da hassasiyet bakımından az da olsa karar ağacına göre düşük sonuç elde etmiştir. Genel olarak rastgele orman yöntemi daha büyük veriler için karar ağacı yöntemine göre daha iyi sonuçlar vermektedir. Lojistik regresyon ve SVM tabanlı OneVsRest yöntemleri ise birbirine oldukça benzer sonuçlar elde etmiştir. Bu modellerin ikisi de hiçbir yoğun trafik durumunu doğru tahmin edememiştir. Trafik tahmini gibi birçok parametreye bağlı olan bir durum için GBT tabanlı OneVsRest yönteminin oldukça iyi sonuç verdiği, karar ağaçları ve rastgele orman yöntemlerinin ise geçer not aldığı söylenebilir.

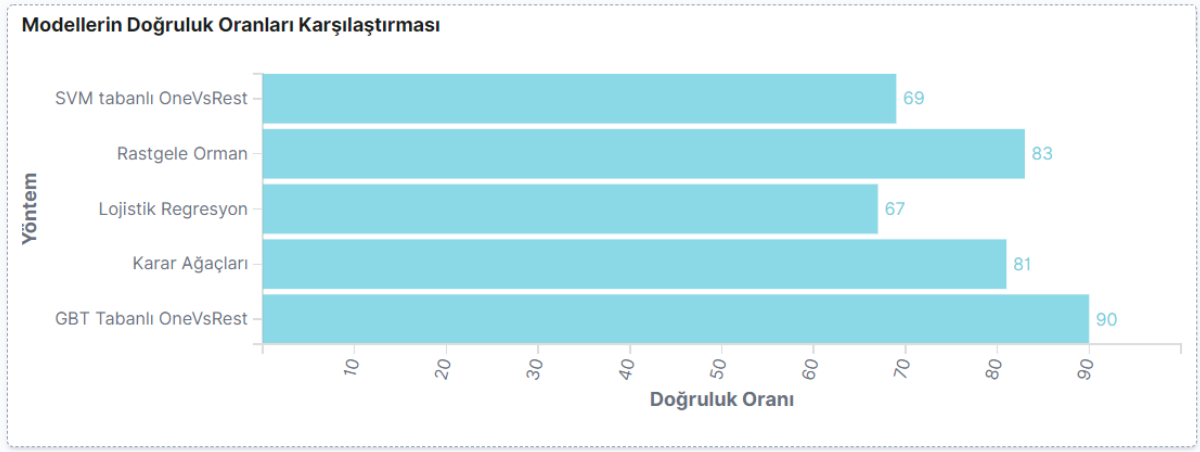
5. TARTIŞMA VE SONUÇ

Günümüzün önemli bir problemi olan Covid-19 pandemisi, tüm dünyada ciddi problemlere sebep oldu. Ülke liderleri bu problemlerin etkilerini minimum seviyelere indirmek için birçok konuda kısıtlama getirdi. Bu kısıtlamalarla, insanlar birçok alışkanlıklarını değiştirmek zorunda kaldı. Bu değişimlerden biri de seyahat alışkanlığı konusunda oldu. Seyahat alışkanlığındaki değişim de doğal olarak kentlerdeki trafik durumunu etkiledi. Özellikle Covis-19 pandemisinin zirve yaptığı dönemlerde, aramıza giren sokağa çıkma yasakları ile ana yol ve caddeler araçsız kaldı.

Bu tez çalışmasında asıl amaçlanan, İstanbul için 2020 yılı trafik analizini yapıp, Covid-19 pandemisinin etkilerini göstermektir. Bununla beraber yavaş yavaş akıllı şehirlere dönüşmeye başladığımız şu zamanlarda, akıllı ulaşım sistemlerinin bu dönüşümün önemli bir parçası olması ve bu önemli parça için trafik analiz ve tahminleri önem arz ettiğinden dolayı, yapılan çalışma bu alana da ışık tutmaktadır. Büyük veri teknolojileri ile yapılacak durağan ve gerçek zamanlı analizlerle trafik koşulları kontrol altına alınıp, trafiğin etkileri azaltılabilir.

Tez kapsamında, 2020 yılı İstanbul trafik indeksi ve sokağa çıkma yasağı veri setleri kullanılarak, Apache Spark ile 2020 yılı İstanbul trafik analizi yapılmıştır. Analizler gün, ay, saat, sokağa çıkma yasağının olup olmamasına göre yapılmıştır. 2020 yılında trafiğin en yoğun olduğu ayın ocak, günün cuma, saatin ise 18 olduğu tespit edilmiştir. Sokağa çıkma yasağının olduğu zaman dilimlerinde beklenildiği gibi düşük trafik yoğunlukları elde edilmiştir. Yapılan aylık bazlı analizlerde, Covid-19'un İstanbul trafiğinde nasıl bir değişime neden olduğu belirlenmiştir. Ocak ve şubat aylarının trafik yoğunluklarının birbirlerine yakın olduğu, Mart ayında ise ülkemizde ilk koronavirüs vakasından itibaren trafik yoğunluğunun düştüğü tespit edilmiştir. Nisan ayında okulların kapatalı tutulması, çoğu şirketin evden çalışmaya geçmesi ve sokağa çıkma yasaklarıyla beraber trafik yoğunluğu dip noktaya inmiş ve nisan ayı 2020 yılındaki en düşük trafik yoğunluğuna sahip ay olarak tespit edilmiştir. Haziran ayından itibaren artış gösteren trafik yoğunluğu, eylül ve ekim aylarında Covid-19'un etkisini gösterdiği aylar içerisinde en yüksek değerlere ulaşmıştır. Elde edilen analiz sonuçları NoSQL veri tabanına atılıp, kibana ile atılan bu verilerin görselleştirmesi yapılmıştır. 2020 yılı trafik durumu analiz işlemi yapılıp, sonuçlar elde edildikten sonra 2020 yılına ait trafik indeksi veri seti kullanılarak lojistik regresyon, karar ağaçları, rastgele orman, GBT tabanlı OneVsRest ve SVM tabanlı

OneVsRest yöntemleriyle trafik durumu tahmini için makine öğrenmesi uygulamaları yapılmıştır. Yapılan uygulamalarda ay, gün, ayın günü, saat, sokağa çıkma ve resmi tatil olup olmaması durumları öz nitelik kolonları olarak alınmıştır. Tasarlanan 5 modelin doğruluk oranları karşılaştırıldığında en iyi sonucu GBT tabanlı OneVsRest almıştır. GBT tabanlı OneVsRest yöntemi özellikle yoğun trafik tahmini konusunda diğer yöntemlerin önüne geçmiştir. Rastgele orman ve karar ağaçları yöntemleri ise gerek doğruluk oranında gerekse yoğun trafik tahmini konusunda GBT tabanlı OneVsRest yöntemi kadar başarılı olamasalar da tatmin edici sonuçlar vermiştir. Lojistik regresyon ve SVM tabanlı OneVsRest yöntemlerine gelecek olursak, bu yöntemler birbirlerine oldukça benzer sonuçlar vermiştir. İki yöntem de yoğun trafik durumunu tahmin etmede başarısız olmuştur.



Şekil 5.1: Trafik Durumu tahmininde kullanılan yöntemlerin doğruluk oranları karşılaştırılması.

Şekil 5.2'den modellerin ortalama çalışma sürelerine bakıldığında ise Rastgele orman, Lojistik regresyon ve Karar ağaçları yöntemlerinin ortalama çalışma sürelerinin birbirlerine yakın olduğu görülmektedir. SVM ve GBT tabanlı OneVsRest modellerinin ortalama çalışma süreleri ise diğer modellere göre fazladır. Bu durumun en önemli nedeni bu modellerin çözüm sırasında birden fazla ikili sınıflandırma problemlerine ayrıştırılmasıyla ilgilidir.



Şekil 5.2: Trafik Durumu tahmininde kullanılan yöntemlerin ortalama çalışma sürelerinin karşılaştırılması.

Bu çalışmada trafik analizi ve tahmini için 2020 yılına ait saatlik trafik indeksi veri seti kullanılmış olup, bu veri seti büyük veri teknolojileri için oldukça küçük bir veridir. Kullanılacak veri setinde bulunan verilerin gelme aralığı ne kadar sık olursa yapılan trafik analizleri ve tahminleri de o kadar detaylı olacaktır. Aynı zamanda çalışma kapsamında yapılan analiz ve tahminler durağan veriler üzerinde analiz edilmiştir, gerçek zamanlı akan veriler üzerinden yapılan analiz ve tahminler ile çalışma daha kapsamlı hale getirilebilir. Fakat çalışma kapsamı ne kadar arttırılırsa artırılsın yapılan aşamalar tez kapsamında yapılan aşamalardan çok farklı olmayacaktır.

Bugüne kadar; bu çalışmanın literatür özeti kısmında da bir kaçına yer verilen çalışmalarda olduğu gibi, trafik konusunda yapılan çalışmalarda ağırlıklı olarak derin öğrenme yöntemlerinin kullanıldığı gözükmektedir. Bu çalışmada sınırlı sayıda bir veri ve hava durumu, araçların hızları gibi önemli özellik kolonları olmadan sadece tarih ve resmi tatil ve ya sokağa çıkma yasağı olup olmaması durumları dikkate alınarak yüksek başarı oranları elde edilmiştir. Bu durum bize derin öğrenme tekniklerinin büyük miktarlarda eğitim verisine ihtiyaç duyması, uzun eğitim süresine sahip olması, makine öğrenmesine göre büyük miktarlarda hesaplama gücüne sahip olması sebebiyle üst düzey makinelere ihtiyacının olması gibi dezavantajlarına karşın, trafik konusunda yapılacak analiz ve tahminlerde doğrudan makine öğrenmesi algoritmalarının da iyi sonuçlar verebileceğini göstermektedir.

İlerleyen zamanlarda trafik analizi için yapılacak çalışmalarda, hem veri sayısı hem de veri setindeki kolon sayısı arttırılıp daha detaylı analizler yapılabilir. Veri setine anlık hava durumu, anlık bölgedeki araçların ortalama hızları gibi öznitelikler eklenip, bu parametrelerin de trafiğe etkisi analiz edilebilir ve daha yüksek doğrulukta trafik durumu tahmini için makine öğrenmesi

uygulamaları geliştirilebilir. Sosyal medyanın yaygın olarak kullanıldığı günümüzde, bu durumdan da faydalanmak için yol durumu verileri içeren sosyal medya verilerinin de analiz işlemlerine katılıp değişen çevre koşullarına ve insana bağlı durumlara karşı daha kararlı modeller elde edilebilir.

KAYNAKLAR

[1]. Aktan, R., 2018, "Büyük veri: uygulama alanları, analitiği ve güvenlik boyutu", Bilgi Yönetimi, 1(1), 1-22.

[2]. https://data.ibb.gov.tr/dataset/trafik-indeks-degeri-web-servisi, [Ziyaret Tarihi: 20 Mart 2021].

[3]. Parr, S., Wolshon, B., Renne, J., Tuite, P., Kim, K., 2020, Traffic impacts of the COVID-19 pandemic: statewide analysis of social separation and activity restriction, Natural Hazards Review, 21(3).

[4]. Doğan, K. ve Arslantekin, S., 2016, "Büyük veri: önemi, yapısı ve günümüzdeki durum", DTCF, 56(1), 15-36.

[5]. 2019, Big data (büyük veri) nedir?, https://medium.com/dusunenbeyinler/big-data-b%C3%BCy%C3%BCk-veri-analizi-d53d8f8ab52b, [Ziyaret Tarihi: 10 Nisan 2021].

[6]. https://www.fonepartner.com/blog/big-data-big-data-nedir-bu-big-data, [Ziyaret Tarihi: 10 Nisan 2021].

[7]. 2019, 5 V's of big data, https://www.geeksforgeeks.org/5-vs-of-big-data/, [Ziyaret Tarihi: 11 Nisan 2021].

[8]. Bekdemir,K., 2020, İnternette 1 Dakikada Neler Oluyor?(2020), https://www.brandingturkiye.com/internette-1-dakikada-neler-oluyor-2020/, [Ziyaret Tarihi: 11 Nisan 2021].

[9]. 2020, Dijital sağlık hizmetlerinde büyük veri (big data) nedir?, http://www.saglikteknoloji.com/dijital-saglik-hizmetlerinde-buyuk-veri-big-data-nedir/, [Ziyaret Tarihi: 24 Nisan 2021].

[10]. Haznedaroğlu, E., Bankacılık sektöründe big data analizi, https://www.pem360.com/blog/BusinessAnalysis/Bankacilik-Sektorunde-Big-Data-Analizi/496, [Ziyaret Tarihi: 24 Nisan 2021].

[11]. Ekmekçi, U., Hadoop ecosystem, https://techbros.com.tr/bigdata/hadoop-ecosystem/, [Ziyaret Tarihi: 25 Nisan 2021].

[12]. https://hadoop.apache.org/, [Ziyaret Tarihi: 25 Nisan 2021].

[13]. Eren, S., 2019, Apache kafka'ya giriş, https://medium.com/devopsturkiye/apache-kafkaya-giri%C5%9F-3399e5f33f8e, [Ziyaret Tarihi: 7 Mayıs 2021].

[14]. https://storm.apache.org/, [Ziyaret Tarihi: 7 Mayıs 2021].

[15]. https://www.tutorialspoint.com/apache_pig/apache_pig_overview.htm, [Ziyaret Tarihi: 7 Mayıs 2021].

[16]. Hammink, j., 2019, An introduction to Apache Cassandra, https://aiven.io/blog/an-introduction-to-apache-cassandra, [Ziyaret Tarihi: 7 Mayıs 2021].

[17]. https://www.tutorialspoint.com/hbase/hbase_overview.htm, [Ziyaret Tarihi: 7 Mayıs 2021].

[18]. Datla, S. and Sharma, S., 2008, Impact of cold and snow on temporal and spatial variations of highway traffic volumes, Journal of Transport Geography, 16(5), 358-372.

[19]. Cools, M., Moons, E., Wets, G., 2010, Assessing the ımpact of weather on traffic ıntensity, Weather Climate and Society, 2(1), 60-68.

[20]. Yan, X., Li, X., Liu, Y., Zhao, J., 2014, Effects of foggy conditions on drivers' speed control behaviors at different risk levels, Safety Science, 68, 275-287.

[21]. Zhu, L., Guo, F., Krishnan, R., Polak, J.W., 2018, A deep learning approach for traffic incident detection in urban networks, 21st International Conference on Intelligent Transportation Systems (ITSC), IEEE, 1011–1016.

[22]. Peppa, M.V., Bell, D., Komar, T., Xiao, W., 2018, Urban traffic flow analysis based on deep learning car detection from CCTV image series, The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 499–506.

[23]. Polson, N.G. and Sokolov, V.O., 2017, Deep learning for short-term traffic flow prediction, Transportation Research Part C: Emerging Technologies, 79, 1-17.

[24]. Raj, J., Bahuleyan, H., Vanajakshi, L.D., 2016, Application of data mining techniques for traffic density estimation and prediction, Transportation Research Procedia, 17, 321-330.

[25]. Tas, N. ve Sezen, B., 2020, Yapay sinir ağları ile trafik yoğunluğu tahmini, Afyon Kocatepe Üniversitesi Sosyal Bilimler Dergisi, *22*(4), 1020-1034.

[26]. Özdağ, M.E., 2019, Derin öğrenme teknikleri kullanılarak anayol trafik analizi, Yüksek Lisans Tezi, Karabük Üniversitesi.

[27]. Beck, M.J. and Hensher, D.A., 2020, Insights into the impact of COVID-19 on household travel and activities in Australia – The early days under restrictions, *Transport Policy*, 96, 76–93.

[28]. Beck, M.J. and Hensher, D.A., 2020, Insights into the impact of COVID-19 on household travel and activities in Australia – The early days of easing restrictions, *Transport Policy*, 99, 95–119.

[29]. Shakibaei, S., Jong, G.C., Alpkökin, P., Rashidi, T.H., 2021, Impact of the COVID-19 pandemic on travel behavior in Istanbul: A panel data analysis, Sustainable Cities and Society, 65.

[30]. Küçükayan, Y.G., 2016, Büyük veri algoritmalarıyla büyük şehirlerin trafik problemlerinin incelenmesi, Yüksek Lisans Tezi, Tobb Ekonomi Ve Teknoloji Üniversitesi.

[31]. Can, Y., 2018, Analytics of Turkish official gazette archives using big data mining techniques, Yüksek Lisans Tezi, Ankara Yıldırım Beyazıt Üniversitesi.

[32]. Keskin, M.V., 2018, Büyük veride makine öğrenmesi uygulaması, Yüksek Lisans Tezi, Yıldız Teknik Üniversitesi.

[33]. Zaralı, D.N., 2018, Akıllı çiftlikler için büyük veri analizi, Yüksek Lisans Tezi, Gazi Üniversitesi.

[34]. Karabay, B., 2019, Yaşanan terör olaylarını içeren büyük verinin makine öğrenmesi teknikleri ile analizi, Yüksek Lisans Tezi, Fırat Üniversitesi.

[35]. Bayraktar, F., 2020, Kripto para fiyat değişimlerinin büyük veri araçları ile analizi, Yüksek Lisans Tezi, Eskişehir Teknik Üniversitesi.

[36]. https://www.jetbrains.com/help/idea/discover-intellij-idea.html#IntelliJ-IDEA-editions, [Ziyaret Tarihi: 10 Temmuz 2021].

[37]. Hartman, J., 2021, What is Java? Definition, Meaning & Features of Java Platforms, https://www.guru99.com/java-platform.html, [Ziyaret Tarihi: 10 Temmuz 2021].

[38]. 2021, https://www.geeksforgeeks.org/introduction-to-java/, [Ziyaret Tarihi: 10 Temmuz 2021].

[39]. https://databricks.com/spark/about, [Ziyaret Tarihi: 21 Temmuz 2021].

[40]. Bilgin, F., 2020, Apache Spark'a Giriş, https://www.veribilimiokulu.com/apache-sparka-giris/, [Ziyaret tarihi: 21 Temmuz 2021].

[41]. https://spark.apache.org/docs/2.2.0/rdd-programming-guide.html, [Ziyaret Tarihi: 22 Temmuz 2021].

[42]. Damji, J., 2016, A Tale of Three Apache Spark APIs: RDDs vs DataFrames and Datasets, https://databricks.com/blog/2016/07/14/a-tale-of-three-apache-spark-apis-rdds-dataframes-and-datasets.html, [Ziyaret Tarihi: 23 Temmuz 2021].

[43]. https://spark.apache.org/docs/2.2.0/sql-programming-guide.html#overview, [Ziyaret Tarihi: 23 Temmuz 2021].

[44]. https://spark.apache.org/docs/2.2.0/ml-classification-regression.html, [Ziyaret Tarihi: 23 Temmuz 2021].

[45]. Abraham, S., Huynh, C., Vu, H., 2020, Classification of Soils into Hydrologic Groups Using Machine Learning, Data, 5(1), 8-9.

[46]. https://spark.apache.org/docs/2.2.0/ml-classification-regression.html#logistic-regression, [Ziyaret Tarihi: 7 Ağustos 2021].

[47]. https://spark.apache.org/docs/2.2.0/mllib-decision-tree.html, [Ziyaret Tarihi: 7 Ağustos 2021].

[48]. Akca, M.F., 2020, Karar Ağaçları, https://medium.com/deep-learning-turkiye/karar-a%C4%9Fa%C3%A7lar%C4%B1-makine-%C3%B6%C4%9Frenmesi-serisi-3-a03f3ff00ba5, [Ziyaret Tarihi: 7 Ağustos 2021].

[49]. https://spark.apache.org/docs/2.2.0/mllib-ensembles.html#random-forests, [Ziyaret Tarihi: 14 Ağustos 2021].

[50]. 2020, One-vs-Rest strategy for Multi-Class Classification, https://www.geeksforgeeks.org/one-vs-rest-strategy-for-multi-class-classification/, [Ziyaret Tarihi: 14 Ağustos 2021].

[51]. Kurama, V., 2019, Gradient Boosting In Classification: Not a Black Box Anymore!, https://blog.paperspace.com/gradient-boosting-for-classification/, [Ziyaret Tarihi: 14 Ağustos 2021].

[52]. Akca, M.F., 2020, Nedir Bu Destek Vektör Makineleri?, https://medium.com/deep-learning-turkiye/nedir-bu-destek-vekt%C3%B6r-makineleri-makine-%C3%B6%C4%9Frenmesi-serisi-2-94e576e4223e, [Ziyaret Tarihi: 15 Ağustos 2021].

[53]. https://spark.apache.org/docs/2.2.0/streaming-programming-guide.html, [Ziyaret Tarihi: 15 Ağustos 2021].

[54]. Doğan, K., 2019, Elasticsearch Nedir, https://medium.com/@kdrcandogan/elasticsearch-nedir-45d237c29b26, [Ziyaret Tarihi: 15 Ağustos 2021].

[55]. https://www.elastic.co/what-is/kibana, [Ziyaret Tarihi: 15 Ağustos 2021].

EKLER

EK 1. Düzenlenmiş trafik indeksi veri seti

1	TrafikIndeksTarihi	SokagaCikmaYasagi	ResmiTatil	TrafikIndeksi
2	2020-12-31 23:00:00	1	0	2.0
3	2020-12-31 22:00:00	1	0	2.0
4	2020-12-31 21:00:00	1	0	3.0
5	2020-12-31 20:00:00	0	0	16.0
6	2020-12-31 19:00:00	0	0	36.0
7	2020-12-31 18:00:00	0	0	54.0
8	2020-12-31 17:00:00	0	0	60.0
9	2020-12-31 16:00:00	0	0	61.0
10	2020-12-31 15:00:00	0	0	59.0
11	2020-12-31 14:00:00	0	0	54.0
12	2020-12-31 13:00:00	0	0	47.0
13	2020-12-31 12:00:00	0	0	40.0
14	2020-12-31 11:00:00	0	0	33.0
15	2020-12-31 10:00:00	0	0	30.0
16	2020-12-31 09:00:00	0	0	33.0
17	2020-12-31 08:00:00	0	0	31.0
18	2020-12-31 07:00:00	0	0	22.0
19	2020-12-31 06:00:00	0	0	10.0
20	2020-12-31 05:00:00	0	0	6.0
21	2020-12-31 04:00:00	1	0	8.0
22	2020-12-31 03:00:00	1	0	7.0
23	2020-12-31 02:00:00	1	0	7.0
24	2020-12-31 01:00:00	1	0	5.0
25	2020-12-31 00:00:00	1	0	5.0

.

.

.

8779	2020-01-01 06:00:00	0	1	6.0
8780	2020-01-01 05:00:00	0	1	4.0
8781	2020-01-01 04:00:00	0	1	2.0
8782	2020-01-01 03:00:00	0	1	4.0
8783	2020-01-01 02:00:00	0	1	8.0
8784	2020-01-01 01:00:00	0	1	10.0
8785	2020-01-01 00:00:00	0	1	9.0