T.C.
HARRAN ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

MAKİNE ÖĞRENMESİ REGRESYON YÖNTEMLERİNİN NESNELERİN İNTERNETİ VERİLERİNE UYGULANMASI

Muhammet Emre IRMAK

ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

ŞANLIURFA
2019

Doç. Dr. İbrahim Berkan AYDİLEK danışmanlığında Muhammet Emre IRMAK'ın hazırladığı "**Makine Öğrenmesi Regresyon Yöntemlerinin Nesnelerin İnterneti Verilerine Uygulanması**" konulu bu çalışma 18/06/2019 tarihinde aşağıdaki jüri tarafından oy birliği ile Harran Üniversitesi Fen Bilimleri Enstitüsü Elektrik Elektronik Mühendisliği Anabilim Dalı'nda YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

İmza

Danışman : Doç. Dr. İbrahim Berkan AYDİLEK .....

Üye : Dr. Öğr. Üyesi Abdülkadir GÜMÜŞÇÜ .....

Üye : Dr. Öğr. Üyesi Seydi KAÇMAZ .....

Bu Tezin Elektrik Elektronik Mühendisliği Anabilim Dalında Yapıldığını ve Enstitümüz Kurallarına Göre Düzenlendiğini Onaylarım.

Doç. Dr. İsmail HİLALİ

Enstitü Müdürü

Not: Bu tezde kullanılan özgün ve başka kaynaktan yapılan bildirişlerin, çizelge, şekil ve fotoğrafların kaynak gösterilmeden kullanımı 5846 sayılı Fikir ve Sanat Kanunundaki hükümlere tabidir.

İÇİNDEKİLER

Sayfa No

ÖZET

Yüksek Lisans Tezi

MAKİNE ÖĞRENMESİ REGRESYON YÖNTEMLERİNİN NESNELERİN İNTERNETİ VERİLERİNE UYGULANMASI

Muhammet Emre IRMAK

Harran Üniversitesi
Fen Bilimleri Enstitüsü
Elektrik Elektronik Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. İbrahim Berkan AYDİLEK
Yıl: 2019, Sayfa: 66

Akıllı şehirler, akıllı ulaşım, akıllı tarım, endüstri 4.0, nesnelerin interneti gibi kavramlar ve uygulamaları son yıllarda daha popüler hale gelmiştir. Günlük yaşantımızda yer alan ve her geçen daha fazla kullanacağımız bu sistemlerde birçok noktadan alınan büyük boyutlardaki anlık verinin gerçek zamanlı olarak işlenmesi ve elde edilen sonuçlara göre kararlar alınması gerekmektedir. Büyük boyutlu verinin analizi ve akıllı sonuçların çıkarılması aşamasında yapay zekâ yöntemleri araştırmacılara fayda ve kolaylıklar sağlamaktadır. Bu tez çalışmasında verilerin gönderiminde kullanılacak bir nesnelerin interneti donanım bileşeni tasarlanmış, gerçek zamanlı verilerin alınması ve makine öğrenme uygulamasını sağlayan bir yazılım geliştirilmiştir. Kullanılan makine öğrenmesi algoritmaları başarı oranları ve işlem süreleri bakımından incelenerek kıyaslanmıştır.

ANAHTAR KELİMELER: Nesnelerin interneti, makine öğrenmesi, regresyon algoritmaları

ABSTRACT

MSc Thesis

APPLYING MACHINE LEARNING REGRESSION METHODS TO INTERNET OF THINGS DATA

Muhammet Emre IRMAK

Harran University
Graduate School of Natural and Applied Sciences
Department of Electrical and Electronics Engineering

Supervisor: Assoc. Prof. Dr. İbrahim Berkan AYDİLEK
Year: 2019, Page: 66

Concepts and applications such as smart cities, smart transportation, smart agriculture, industry 4.0, internet of things have become more popular in recent years. In these systems, which are going to be used more and more in our daily life, it is necessary to process real-time instant data received from many points in real time and to make decisions based on the results obtained. In the stage of analysis of the large-scale data and the elaboration of intelligent results, artificial intelligence methods provide benefits and facilities to the researchers. In this thesis, the hardware component of the internet of things is designed to be used for the transmission of data, a software has been developed to receive real-time data and enable machine learning. The machine learning algorithms used were compared by examining the success rates and processing times.

KEY WORDS: Internet of things, machine learning, regression algorithms

TEŞEKKÜR

Aileme ve tez çalışmamın daha iyi bir noktaya gelmesindeki katkılarından dolayı danışmanım Doç. Dr. İbrahim Berkan AYDİLEK' e şükranlarımı sunarım.

ŞEKİLLER DİZİNİ

Sayfa No

ÇİZELGELER DİZİNİ

Sayfa No

SİMGELER ve KISALTMALAR DİZİNİ

IOT	Internet of Things
ITU	International Telecommunications Union
IEEE	Institute of Electrical and Electronics Engineers
ETSI	European Telecommunications Standards Institute
SMTP	Simple Mail Transfer Protocol
EPA	Unites States Environmental Protection Agency
MLR	Multiple Linear Regression- Çoklu Doğrusal Regresyon
DTR	Decision Tree Regression- Karar Ağacı Regresyon
KNNR	K-Nearest Neighbor Regression- K En Yakın Komşu Regresyonu
SVR	Support Vector Regression- Destek Vektör Regresyon
NNR	Neural Network Regression- Yapay Sinir Ağı Regresyon
RFR	Random Forest Regression- Rastgele Orman Regresyon
STCKR	Stacking Regression- Yığın Regresyon
ADBR	Adaboost Regression- Uyumlu Artırıcı Regresyon
GRBR	Gradient Boosting Regression- Eğimli Artırıcı Regresyon
BAGR	Bagging Regression- Örneklemeli Toplam Regresyon

1. GİRİŞ

Akıllı şehirler, akıllı evler, akıllı tarım, nesnelerin interneti, büyük veri gibi kavramları son zamanlarda sıkça duyulmaktadır. Günlük hayatımızın kolaylaşması, üretimde verimin artması gibi faydalar sağlayan bu sistemler özellikle son yüz yılda geliştirilen teknolojinin sayesinde mümkün olmaktadır. Bu sistemlerde binlerce belki yüzbinlerce noktaya yerleştirilen sensör, kamera gibi algılayıcı donanımlardan gerçek zamanlı olarak toplanan büyük boyutlardaki bilginin farklı algoritmalar tarafından işlenmesi ve sonuçlar üretilerek gerekli kararların alınması gerekmektedir. İşlenen verinin büyük boyutlarda olduğu ve sonuç üretme işleminin sade denklemlerle yapılamadığı durumlarda yapay zekâ algoritmaları başarılı sonuçlar üretebilmektedir.

Çalışmada geliştirilen sistem sayesinde gerçek zamanlı olarak sensör verileri toplanmaktadır ve bu verilere yapay zekâ algoritmalarından makine öğrenmesi regresyon algoritmaları uygulanmaktadır. Sistem çalışmaya devam ederken yeni gelen gerçek zamanlı sensör verileriyle algoritmalar eğitilmeye devam edilmekte ve bu bakımdan sistem çevrimiçi öğrenen bir sistem olarak tanımlanmaktadır. Çevirim dışı sistemlerde ise algoritmalar sistem henüz çalışmaz iken eğitilmekte ve sisteme yüklenmektedir.

Akıllı sistemlerde gerek donanımsal gerekse yazılımsal pek çok bileşen kullanılmaktadır. Donanımsal bileşenler arasında en önemli bileşen nesnelerin interneti bileşenidir. Nesnelerin interneti (internet of things) kavramı kısaca internete bağlanabilen herhangi bir nesneyi ifade etmektedir. Bu nesne bir sensör, bir lamba, bir buzdolabı olabileceği gibi bir saksı, bir akvaryum, bir hayvan ya da bir insan olabilmektedir. Nesneler üzerlerindeki donanım sayesinde bilgileri internet üzerinden gönderebilmekte ve alabilmektedir.

Nesnelerin internetinde en önemli bileşen internettir. İnternet doksanlı yılların başında Dünya'da kullanılmaya başlanmıştır. Günümüzde ise gerek kapsama alanı ve hız bakımından gerekse maliyet bakımından kullanımı kolaylaşmış ve yaygınlaşmıştır. Nesnelerin internetindeki diğer en önemli bileşenler ise nesnelerin internete

bağlanmasını sağlayan donanım ve yazılım bileşenleridir. Özellikle son 10 yılda çok daha hızlı çalışabilen, çok düşük enerji tüketimi ile uzun süreler çalışabilen, küçük boyutlarda, düşük maliyetlerle üretilebilen donanımlar nesnelerin internetini mümkün kılmıştır.

2005 yılında Uluslararası Telekomünikasyon Birliği tarafından nesnelerin internetiyle ilgili ilk rapor yayınlanmıştır. 2008 yılında ilk Avrupa Nesnelerin İnterneti konferansı düzenlenmiştir (Postscapes, 2018).

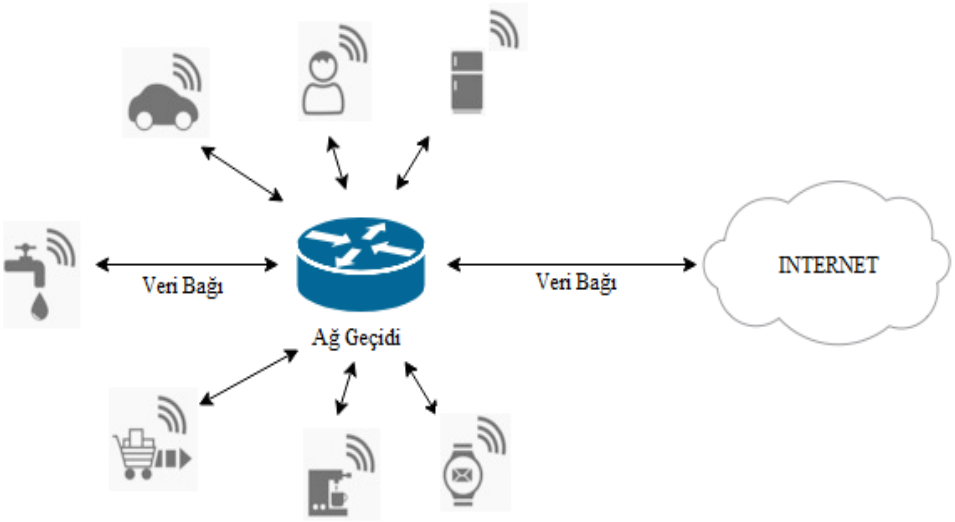
Cisco internet işleri çözümü grubuna göre 2008-2009 yılları arasında internete bağlanan nesne veya obje sayısı, internete bağlanan kişi sayısını geçmiştir ve bu nesnelerin internetinin doğuşu olarak kabul edilmiştir (Postscapes, 2018).

2008 yılında ABD Ulusal İstihbarat Konseyi, nesnelerin internetini 2025 yılına kadar ABD'nin çıkarları üzerinde potansiyel etkileri olan 6 yıkıcı sivil teknolojiden biri olarak sıralamıştır (Postscapes, 2018).

2011 yılında IPv6 kamuya açıklanmıştır. Yeni protokol sayesinde 2^{128} adet farklı ip numarası üretilebilecektir ve bu dünyada üretilebilecek tüm cihazlara yeterli gelebilecektir (Postscapes, 2018).

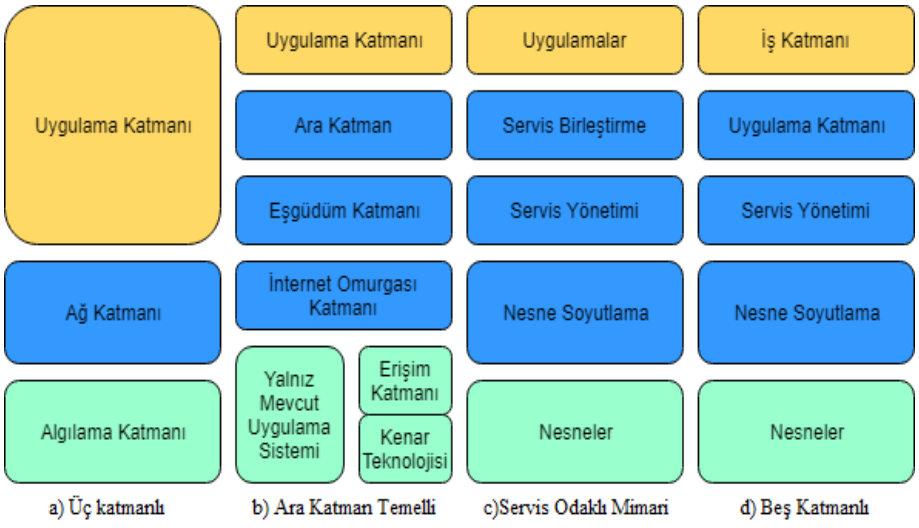
2025 yılında internete bağlı nesnelerin sayısının günümüzdekinin üç katına ulaşarak 25 milyar olması beklenmektedir. Cisco ve Intel şirketleri ise bunun 50 milyar olabileceğini tahmin etmektedir (Paxtechnica, 2018).

1.1. Nesnelerin İnterneti Temel Yapısı



Şekil 1.1. Nesnelerin interneti temel yapısı

Nesnelerin interneti için çeşitli sayıda ve yapıda katman içeren modeller önerilmektedir. Bu konuda belirlenmiş bir standart model yoktur. Probleme yaklaşım açısına göre farklı katmanlı modeller oluşturulmaktadır (Al-Fuqaha ve ark., 2018). Şekil 1.2.'de literatürdeki bazı modeller gösterilmiştir.



Şekil 1.2. Nesnelerin internetinde kullanılan farklı modeldeki katman yapıları

Çalışmada kullanılan model ise sırasıyla şu katmanları içermektedir; Algılama Katmanı, Veri Bağı Katmanı, İnternet Ağı Katmanı, Uygulama Katmanı, İş Katmanı

Şekil 1.3. Çalışmada kullanılan katmanlı model ve protokoller

1.2. Nesnelerin İnternetinde Katmanlar Standartlar ve Protokoller

1.2.1. Algılama katmanı

Nesne ile ilgili bilgilerin alındığı katmandır. Bu bilgiler sensör, rfid/nfc, kamera gibi donanımlar aracılığıyla alınmaktadır. Genellikle bu katman düşük özellikli mikro denetleyici içermektedir. Mikro denetleyici ile algılama bileşeni arasındaki bağlantı uart (rs-232, rs-485), spi, i2c, can, modbus ve usb gibi haberleştirme yöntemleriyle yapılmaktadır.

1.2.2. Veri bağı katmanı

Algılama katmanından alınan veriler bu katmandaki kurallara göre kodlanmaktadır. Kodlanan bilgi kablolu veya kablosuz bir fiziksel bağlantıyla diğer cihazlara veya internet ağına eriştirilmektedir (Engineersgarage, Wustl edu, 2018). Bu katmandaki haberleşme protokolleri kullanıldıkları alanlara göre Çizelge 1.1.'de verilmiştir. IEEE, ITU, ETSI topluluklarının yayınladığı bilgiler esas alınarak hazırlanan Çizelge 1.5.'de haberleşme protokollerinin frekans, menzil ve veri hızı değerleri verilmiştir.

Çizelge 1.1. Kullanıldıkları alanlara göre nesnelerin internet protokolleri

Kişisel Alan Ağı (Personal Area Network- PAN) & Ev Alan Ağı (Home Area Network- HAN) & Yerel Alan Ağı (Local Area Network- LAN)		
• Ethernet	• Wi-Fi	• Bluetooth 4.0 & BLE
• RFID & NFC	• HomePlug	• Mi-Wi
• 6LoWPAN	• WirelessHART	• Zigbee
• Thread	• Z-Wave	• DiGiMesh
Düşük Güç Uzun Menzilli Kablosuz Ağ (Low Power Long Range Wireless Network- LPWAN)		
• Sigfox	• LoRaWAN	• LTE M-1/ LTE- MTC
• Ingenu RPMA	• Weightless	• EC-GSM-IOT
• NBIOT	• Cellular (GPRS/2G/3G/4G/5G)	• DASH7

1.2.3. Ağ adresleme katmanı

Bu katman, internet üzerinden veri paketlerinin adreslemesinden sorumludur. Aktarım katmanından gelen veri gramları kaynak ve hedef adresleri içerir. Ağ katmanında, paketler IP adresleri olarak adlandırılan benzersiz adreslerle kapsüllenmektedir (Engineersgarage, Wustl edu, 2018). Bu katmanda kullanılan bazı protokoller Çizelge 1.2.'de verilmiştir.

Çizelge 1.2. Ağ adresleme katmanı protokolleri

• IPv4	• IPv6	• 6LoWPAN	• 6Lo
• 6TiSCH	• IPv6 over Bluetooth Low Energy		• IPv6 over G.9959

1.2.4. Ağ aktarım ve yönlendirme katmanı

Bu katman, veri paketlerinin yönlendirilmesinden sorumludur. Bu katmanda, paketlerin sıralanması, veri paketlerinin teslimatında hata tespiti ve düzeltmesi gerçekleştirilmektedir (Engineersgarage, Wustl edu, 2018). Bu katmanda kullanılan bazı protokoller Çizelge 1.3.'de verilmiştir.

Çizelge 1.3. Ağ aktarım ve yönlendirme katmanı protokolleri

• TCP	• UDP	• DTLS
• ROLL	• Aeron	• CCN
• CORPL	• QUIC	• uIP
• TLS	• RPL	• CARP
• NanoIP	• TSMP	

1.2.5. Uygulama katmanı

Bu iletişim ağı içindeki en yüksek tabakadır. IoT cihazlar ve ağ arasındaki arabirimdir. Bu katman, cihaz ucunda özel bir uygulama ile uygulanır (Engineersgarage, Wustl edu, 2018). Bu katmanda kullanılan bazı protokoller Çizelge 1.4.'te verilmiştir.

Çizelge 1.4. Uygulama katmanı protokolleri

• MQTT	• CoAP	• SMQTT
• STOMP	• XMPP & XMPP-IOT	• AMQP
• SMTP	• LWM2M	• M3DA
• DDS	• SMCP	• LLAP
• SSI	• RESTful HTTP	• MQTT-SN
• ONS 2.0	• HTTP/2	• SOAP
• Websocket	• Reactive Streams	• JavaScript IOT

Çizelge 1.5. Nesnelerin internetinde kullanılan haberleşme protokollerinin özellikleri

Haberleşme Türü	Frekans	Menzil	Veri Hızı (bps)
• Ethernet	100,500 MHz	100 m	10 M,100 M,1 G,10 G
• Wi-Fi	<1 GHz, 2.4 GHz, 5 GHz ISM	1000m	2 M,11 M,40 M,54 M,600 M
• Bluetooth	2.4 GHz ISM	100 m	1-3 M
• RFID	13.56 MHz	10 m	423 k
• NFC	13.56 MHz	10 cm	424 k
• Mi-Wi	<1 GHz ,2.4 GHz ISM	100m	250k
• 6-LoWPAN	<1 GHz ,2.4 GHz ISM	100m	250k
• WirelessHART	<1 GHz ,2.4 GHz ISM	100 m	250 k
• Zigbee	<1 GHz ,2.4 GHz ISM	100 m	250 k
• Thread	<1 GHz ,2.4 GHz ISM	100 m	250 k
• Z-Wave	< 1 GHz ISM	30 m	40 k, 100 k
• DiGiMesh	2.4 Ghz ISM	1.5 km	250 k
• Homeplug	1.8 – 30 Mhz	100 m	4-200 M
• Sigfox	< 1 GHz ISM	50 km	UL:100 DL:600
• LoraWAN	< 1 GHz ISM	15 km	50 k
• LTE M-1/MTC	1.4 MHz 20 Mhz	15 km	UL:1 M DL:1 M
• EC-GSM-IOT	0.2 MHz	15 km	UL:0.5 M, DL:0.5 M
• Cellular (1G,2G,3G,4G)	800 MHz,900 MHz 1800Mhz,2100Mhz	15 km	UL,DL: 2.4 k - 1 G
• NBIOT	200 kHz	15 km	UL:0.2 M, DL:0.2 M
• DASH7	< 1 GHz ISM	5 km	UL,DL:9.6k,55k 166 k
• Weightless	< 1 GHz ISM	5 km	UL,DL:1k-100k
• Ingenu RPMA	2.4 GHz ISM	50 km	UL:624 k DL:156 k

2. ÖNCEKİ ÇALIŞMALAR

Jamil (2018), yaptığı tez çalışmasında nesnelerin interneti sonucu oluşan büyük miktarda verinin gerçek zamanlı olarak depolanması ve işlenmesinin zor olduğunu ifade etmektedir. Bunun için veri tabanı yönetimi ve veri işlenmesinde apache, kafka, spark, cassandra, zeppelin programlarını kullanarak gerçek zamanlı tahminler elde etmişlerdir. Tahminlerin üretilmesinde spark programının makine öğrenmesi kütüphanesi yardımıyla lojistik regresyon modelini kullanmışlardır.

Gülaçar (2018), yaptığı tez çalışmasında Avrupa Birliği'nin akıllı şehirler için bir veri toplama ve yönetimi projesi olan VITAL kapsamında toplanan İstanbul şehri D100 karayolu trafik hız sensörü verileri üzerinde çalışmıştır. Python makine öğrenme kütüphanesi Scikit-learn ve MongoDB gibi programlar yardımıyla farklı makine öğrenmesi sınıflandırma yöntemlerini bu hız verilerine uygulayarak ileri bir zamandaki hız değerlerini tahmin etmeye çalışmıştır.

Tümer (2018), yaptığı tez çalışmasında sağlık alanındaki beldeki disk rahatsızlığı ile ilgili nesnelerin interneti verilerini kullanmıştır. Weka makine öğrenmesi ve Siddhi gerçek zamanlı veri işleme aracını kullanarak gerçek zamanlı sağlık verileri üzerinde makine öğrenmesi algoritmalarının ürettiği sonuçları incelemiştir. Bu çalışma sonunda, basit lojistik algoritmasının diğer algoritmalardan daha iyi sonuç verdiği gözlemlenmiştir.

Bozuklu (2018), yaptığı tez çalışmasında bir gps modülünden aldığı konum bilgilerini gsm modül ile internete bağlanarak bir tcp sunucuya json formatında gönderen bir nesnelerin interneti nesnesi geliştirmiştir. Bulut sunucu üzerinde çalışan ve ters-vintecy formüllerini kullanan bir bilinç kamanı oluşturmuş, toplanan konum bilgilerine göre hesaplanan hedef mesafelerine göre kişilere sms göndermiştir.

Ateş (2018), yaptığı tez çalışmasında merkezi olarak veri izlemesinin ve kontrolünün yapıldığı SCADA sistemlerinin yerini gelecekte nesnelerin interneti tabanlı sistemlerin alacağını düşünmektedir. Bu doğrultuda 50 VA gücünde bir trafonun bilgilerini arduino mikrodenetleyicisine bağlı sıcaklık ve akım-gerilim sensörleri yardımıyla almıştır. Ardından mikrodenetleyiciye bağlı esp8266 wi-fi modülü ile alınan bilgileri ubidots bulut platformuna aktarmıştır. Ubidots platformunun arayüzünü kullanarak wi-fi ile internete bağlı bir nodemcu cihazına komut göndererek röle çekme işlemini yapmıştır. Ubidots platformunu kullanarak gelen transformatör bilgilerinin grafiklerini incelemişlerdir.

Çağlar (2018), yaptığı tez çalışmasında nesnelerin internetini kullanarak akıllı bir sera geliştirmiştir. Sera ile ilgili verileri sensörler ve arduino mikrodenetleyici ile alarak esp8266 wi-fi modülü ile internete bağlanmış ve gerekli api keyleri kullanarak thingspeak.com isimli nesnelerin interneti platformuna aktarmıştır.

Yazıcı ve ark. (2018), yaptıkları çalışmada makine öğrenmesi algoritmalarını, bulut bilgisayarlarda değil bulut bilgisayarlara veri gönderen köprü bilgisayar üzerinde çalıştırmışlardır. Regresyon ve sınıflandırma yöntemlerinde kullanmak için hava kirliliği, enerji tüketimi, malzeme cinsi gibi 5 farklı veri seti seçmişlerdir. Sonuç olarak makine öğrenmesi algoritmalarının çalıştırıldıkları köprü bilgisayar üzerindeki enerji tüketimlerini, ürettikleri sonuçların doğruluk oranlarını ve hesaplama sürelerini göstermişlerdir.

Önal ve ark. (2017), yaptıkları çalışmada Amerika Birleşik Devletlerindeki 8000 farklı hava ölçüm istasyonundan toplanmış hava sıcaklığı, rüzgâr hızı, bağıl nem, görünürlük ve basınç verileri üzerinde k-ortalama kümeleme algoritmasını uygulamışlardır. Sonuç olarak verilerin coğrafi olarak anlamlı olarak dağıldığını belirlemişlerdir. Ayrıca hava ölçüm sensörlerindeki arızaların ve anormalliklerin ölçüm verilerinin harita üzerindeki dağılımlarından anlaşılabileceğini belirlemişlerdir.

Hromic ve ark. (2015), yaptıkları çalışmada Avrupa Birliği'nin geliştirdiği gerçek zamanlı bir veri toplama ve işleme platformu olan OpenIoT'yi kullanmışlardır. Geliştirdikleri taşınabilir hava ölçme cihazı ile Zagreb şehri içerisinde 3 gün boyunca dolaşarak havadaki karbon monoksit (CO) ve kükürt dioksit (SO_2) değerlerini ölçmüşlerdir. Ölçtükleri verileri ise akıllı bir telefona bağlanarak bir uygulama üzerinden gerçek zamanlı olarak OpenIoT platformuna aktarmışlardır. Sonrasında OpenIoT platformu üzerinde ölçtükleri değerlerin konumlarına k-ortalama kümeleme algoritmasını uygulayarak 4 tane alan merkezi oluşturmuşlardır. Bu merkezlerdeki CO ve SO_2 değerler ile Zagreb şehrinin aynı tarihlerdeki hava sıcaklık, nem ve basınç değeri verileri ile olan korelasyonunu incelemişlerdir.

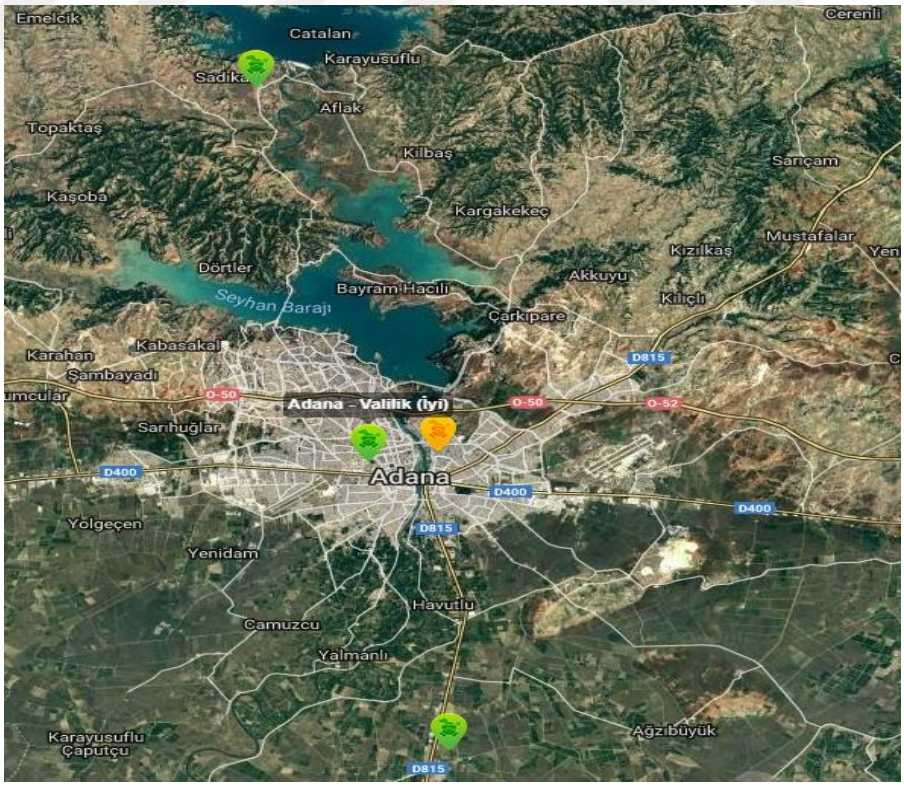
Kanawaday ve Sane (2017), yaptıkları çalışmada bir fabrikada kullanılan dilimleme makinasından alınan sensör verileri üzerinde otoregresif entegre hareketli ortalama (autoregressive integrated moving average- ARIMA) yöntemini uygulayarak gelecekteki kaliteyi artırmayı ve arızaları tahmin etmeyi hedeflemişlerdir. Makine üzerindeki sensör verilerini kablo üzerinden RS485 haberleştirme protokolü ile programlanabilir mantık denetleyicisine (programmable logic controller- PLC) göndermişlerdir. PLC'nin bağlı olduğu bilgisayar ile sensör bilgilerini alarak MQTT protokolüyle internet üzerinden bulut sunucuya göndermişlerdir. Bulut sunucuya erişerek veriler üzerinde ARIMA modelini uygulamışlar ve sonuçları incelemişlerdir.

Akbar ve ark. (2017), yaptıkları çalışmada gerçek zamanlı ve büyük boyutlardaki nesnelerin interneti verisinden kullanılabilir sonuçlar çıkarmak için karmaşık olay işleme ve makine öğrenmesi algoritmalarını kullanmıştır. İspanya şehrinin 4 farklı noktasından alınan trafik verilerinden hızlı olarak sonuçlar üretebilmek için karmaşık olay işlemeyle birlikte uyarlamalı hareketli pencere regresyonu, destek vektör regresyonu, doğrusal regresyon, karar ağacı regresyon ve rastgele orman regresyonunu kullanmıştır. Hibrit yapıların sonuç üretmede daha iyi sonuçlar verdiğini gözlemlemiştir.

3. MATERYAL ve YÖNTEM

3.1. Veri Seti

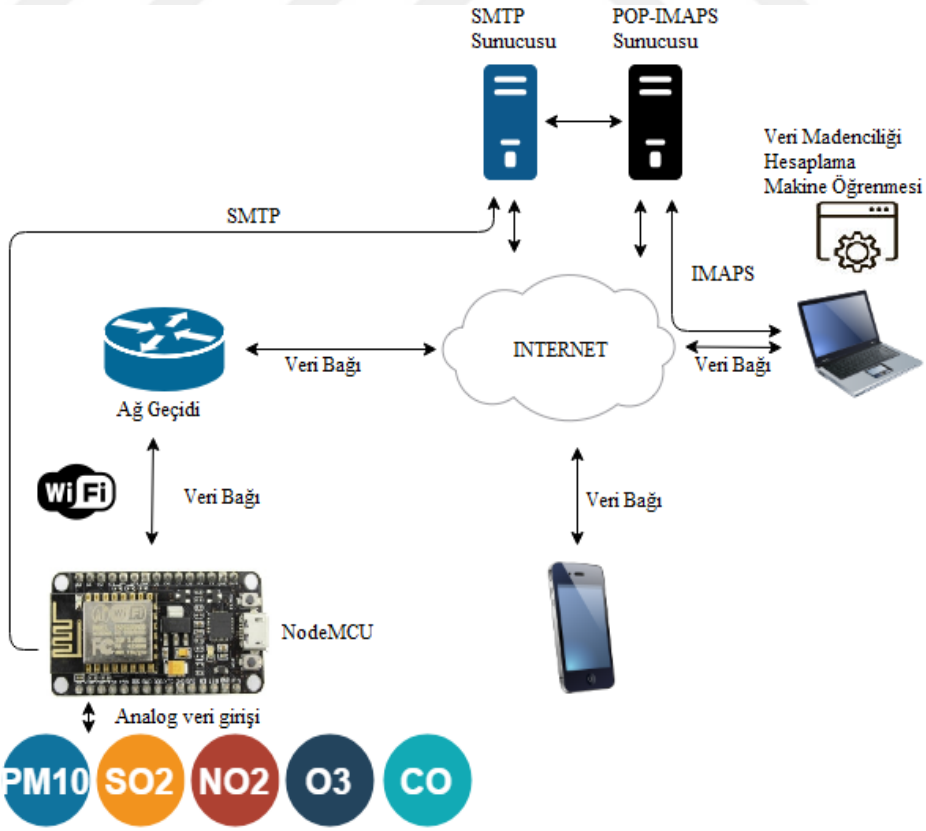
Çalışmada kullanılan veriler Adana ilindeki 4 hava ölçüm istasyonundan biri olan valilik istasyonuna ait hava kirliliği ölçüm verileridir. Şekil 3.1.'de Adana ilindeki 4 istasyonun yerleştirildikleri konumlar görülmektedir. Şehir merkezi olduğu için bu istasyon tercih edilmiştir. Veri seti, Türkiye Cumhuriyeti Çevre ve Şehircilik Bakanlığı'nın geliştirdiği ulusal hava kalite izleme ağının internet sitesinden elde edilmiştir (Havaizleme). Veriler 2013-2017 yılları arasındaki toz parçacıkları (PM10), azot dioksit (NO_2), kükürt dioksit (SO_2), karbon monoksit (CO), ozon (O_3) ve hava kirleticilerinin saatlik ölçüm değerlerinden oluşmaktadır. 2012 yılının son günü verileri de hesaplamada kullanılacağı için eklenmiştir.



Şekil 3.1. Adana ilideki hava ölçüm istasyonları ve konumları

3.2. Sistemin Çalışma Şekli

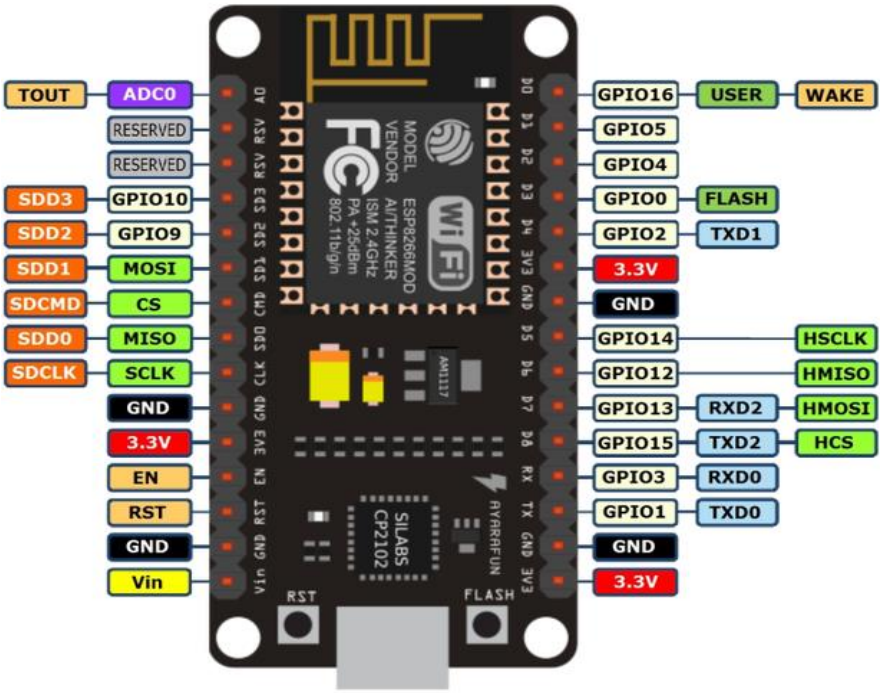
Gönderilecek olan gerçek zamanlı yeni veriler nesnelerin interneti bileşeni aracılığıyla gönderilir. Nesnelerin interneti bileşeni olarak nodeMCU kullanılmıştır. NodeMCU bir mikrodenetleyici ile bir wi-fi modülünün birleştirilmesiyle üretilmiş açık kaynak kodlu bir donanımdır. NodeMCU bağlı olduğu nesnedeki bilgileri sensörler aracılığıyla alabilmekte, kablosuz olarak bir modeme bağlanarak aldığı bu bilgileri internet üzerinden bir sunucuya gönderebilmektedir. Burada alışılagelmişin dışında olarak veriler smtp protokolü kullanılarak email olarak smtp sunucusuna gönderilmektedir. Hesaplama ve makine öğrenmesi işlemlerinin yapılacağı bilgisayar bu verileri alabilmek için internet üzerinden imaps protokolünü kullanarak imaps sunucusuna bağlanır ve emailleri okuyarak bir dosyaya kaydeder. Önceden kayıtlı olan veri setine yeni verileri ekleyerek gerçek zamanlı hesaplama işlemleri ve makine öğrenmesi uygulayarak sonuçları email olarak kullanıcıya gönderir (Şekil 3.2.).



Şekil 3.2. Sistemin çalışma şekli

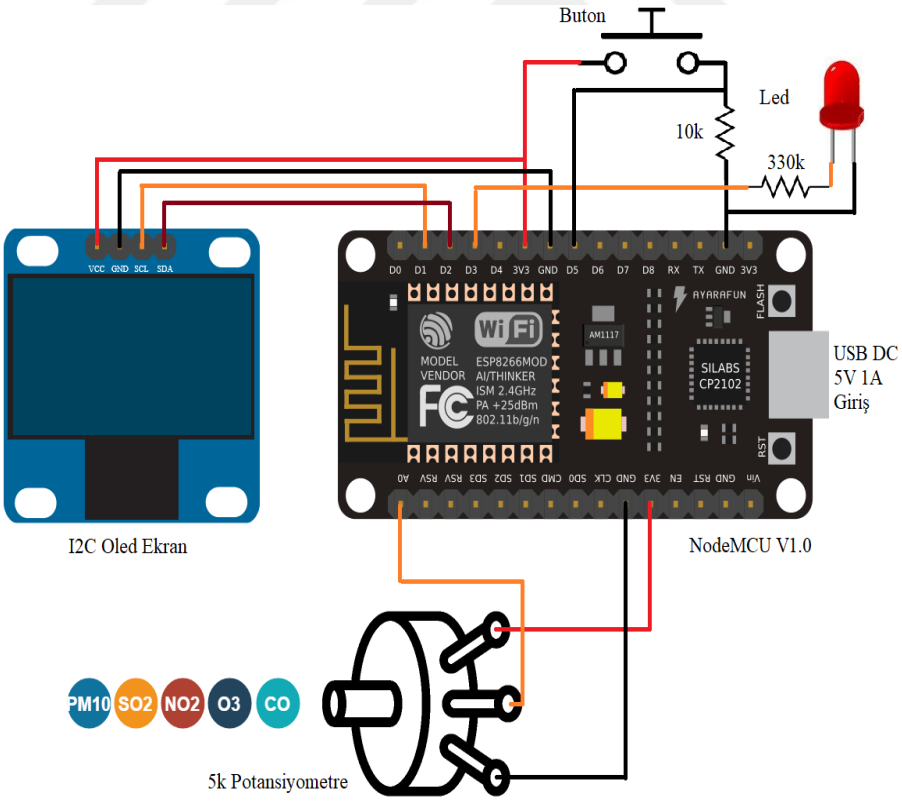
3.3. Nesnelerin İnterneti Bileşeni

Bu bileşen test verilerinin üretilmesinde ve email olarak gönderilmesinde kullanılmaktatır. NodeMCU'da kullanılan temel modül AI Thinker firması tarafından geliştirilmiş olan ESP8266-12E tümleşik modüldür. 32 bit ve 80 Mhz bir mikro denetleyiciyi ve kablosuz haberleşme (802.11 b/g/n) donanımını içerisinde barındırmaktadır. 4 MiB (mebibayt) dahili flash belleği, 10 bit analog- sayısal dönüştürücüsü, dahili TCP-IP protokol yığını vardır. Spi, i2c, uart haberleşme arayüzlerine sahiptir. Genel amaçlı giriş çıkış pinlerine (gpio) sahiptir. Ortalama çalışma akımı 80mA ve beklemede güç tüketimi 1mW kadardır. Silicon Labs firmasının ürettiği CP2102 usb – uart haberleşme dönüştürücüsü nodeMCU ve bilgisayar arasında veri iletimini mümkün kılmaktadır. Advenced Monolithic Systems firmasının ürettiği AMS1117 voltaj regülatörü usb'den alınan 5 voltun 3.3 volta dönüştürülmesinde kullanılmıştır. Bunların haricinde yüzeye monteli (surface mounted device-smd) transistör, diyot, direnç ve kapasitör gibi devre elemanları kullanılmıştır. Şekil 3.3.'te nodeMCU'nun bileşenleri ve pin özellikleri gösterilmiştir.

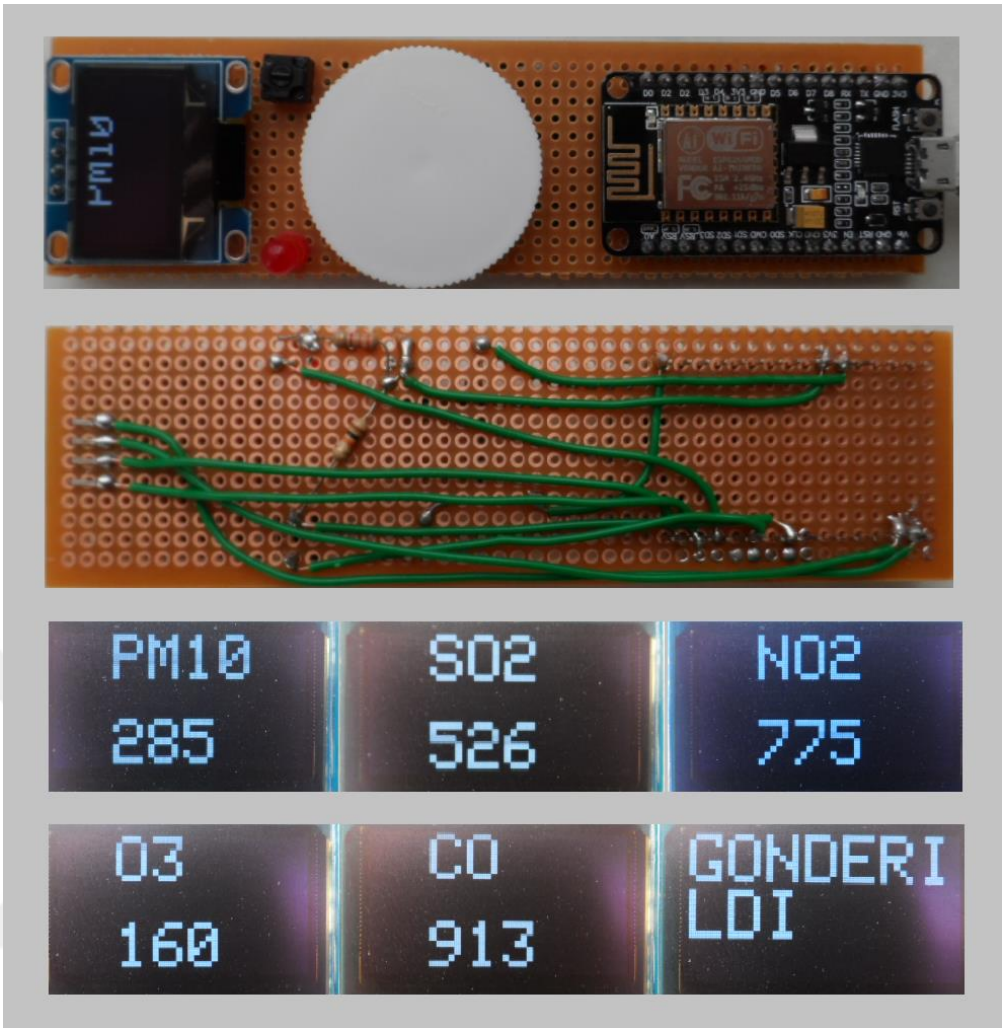


Şekil 3.3. NodeMCU bileşenleri ve pin özellikleri

Hava kirliliği verileri nodeMCU'ya bağlanan bir potansiyometre ile ayarlanmaktadır. Burada sensör yerine potansiyometrenin tercih edilmesinin sebebi istenilen değerde veriler üretebilmektir. Potansiyometre nodeMCU'nun A0 (analog-sayısal dönüştürücü) pinine bağlanmıştır. Kullanıcının potansiyometreden ayarladığı değerlerin türünü ve seviyesini görebilmesi için i2c haberleşme kullanan bir oled (128x64) ekran, verinin kaydedebilmesi için bir buton ve veri girişinin tamamladığını gösteren kırmızı bir led nodeMCU'ya bağlanmıştır. Kullanıcı bu bileşen ile sırasıyla PM10, SO_2, NO_2, O_3 ve CO seviye değerlerini nodeMCU'ya girmektedir. Son aşamada nodeMCU daha önce SSID (service set identifier) ve parolası kayıtlı olan bir kablosuz modeme bağlanmaktadır. Bağlantı kurulduktan sonra kayıtlı email adresine girilen verileri göndermektedir. Şekil 3.4.'te nesnelerin interneti bileşeninin devre tasarımı gösterilmiştir. Tasarlanan devre delikli pertinaks üzerinde lehimleme yöntemiyle uygulanmıştır (Şekil 3.5.).



Şekil 3.4. Nesnelerin interneti bileşeni devre tasarımı



Şekil 3.5. Tasarlanan donanıma ait görseller

NodeMCU için kod yazarken lua derleyicisi veya Arduino derleyicisi kullanılabilmektedir. Kütüphane desteği daha fazla olduğundan Arduino derleyicisi kullanılmıştır. Kablosuz bağlantı işlemleri için ESP8266WiFi.h kütüphanesi, email gönderim işlemleri için Gsender.h kütüphanesi kullanılmıştır. Email gönderilirken bir smtp sunucusu ve port numarası kullanılmaktadır. Emaili gönderen kullanıcının email adresi ve şifresi koda eklenmektedir. Ayrıca emailin gönderileceği kullanıcının email adresi ve emailin konusuda koda eklenmektedir. SPI.h ve Wire.h kütüphaneleri nodeMCU ve i2c oled ekran arasında veri iletiminde kullanılmaktadır. Adafruit_GFX.h ve Adafruit_SSD1306.h kütüphaneleri 128x64 piksel oled ekrandaki karakterlerin piksel konumu ve büyüklüklerinin ayarlanmasında kullanılmaktadır.

Email olarak gönderilen örnek test verisi ise şu formattadır;

SA-1B2 43848 1.01.2018 00:00 97 113 0 402 0 90 1005

SA-1B2 temsili sensör numarasıdır. Bu numara okunan epostanın hangi dosyaya kaydedileceğinin belirlenmesinde kullanılmaktadır. 43848 sayısı indeks numarasıdır. Kayıtlı veri setindeki son indeks numarasının bir fazlasıdır. Email olarak gönderilecek veriler 2018 yılından başlatılmıştır. 1.01.2018 verinin tarihini 00:00 ise saatini ifade etmektedir. Diğer sayılar ise sırasıyla PM10, SO_2, NO, NO_2, NOX, O_3, CO hava kirleticilerinin girilen değerleridir. NO ve NOX değerleri kullanılmayacak ve program tarafından silinecektir. Her döngüde indeks ve saat değeri bir birim artırılmaktadır. Hava kirletici değerleri ise yeniden girildikten sonra açıklanan formatta eposta olarak gönderilmektedir. Sonuç olarak epostalar okunduğunda ve bir klasöre kaydedildiğinde Şekil 3.6'daki gibi görünmektedir.

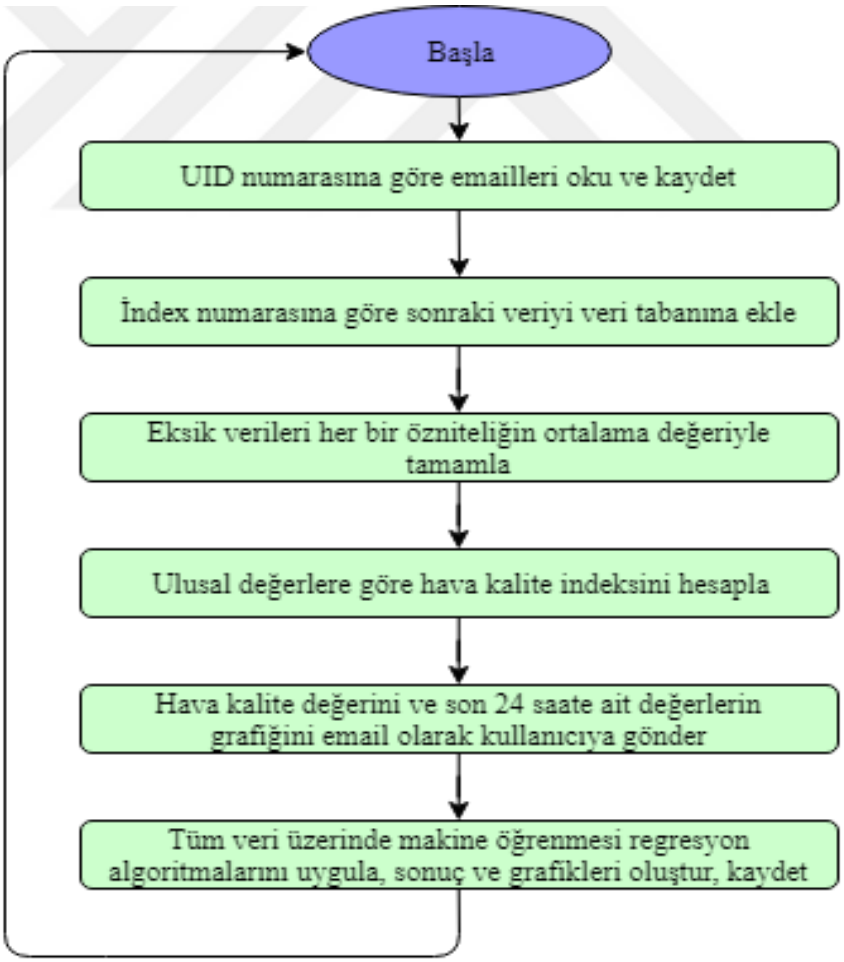
```
sensor index Tarih Zaman PM10 SO2 NO NO2 NOX O3 CO
SA-1B2 43848 1.01.2018 00:00 97 113 0 402 0 90 1005
SA-1B2 43849 1.01.2018 01:00 46 64 0 19 0 55 405
SA-1B2 43850 1.01.2018 02:00 27 25 0 8 0 62 358
SA-1B2 43851 1.01.2018 03:00 31 14 0 13 0 57 375
SA-1B2 43852 1.01.2018 04:00 38 9 0 21 0 27 428
```

Şekil 3.6. Sensör verilerinin kaydedildiği txt dosyasının görünümü

Burada ilk satırdaki başlıklar her bir sütundaki verinin türünü belirlemektedir. Aralarında bir boşluk olması önemlidir çünkü program verileri bu şekilde birbirinden ayırt edebilmektedir.

3.4. Gerçek Zamanlı İşlemler için Geliştirilen Program

Gerçek zamanlı olarak nesnelerin interneti bileşeni tarafından gönderilen verileri alan ve önceden kayıtlı veriler üzerine ekleyen, hava kalite indeksini hesaplayan ve tüm veri seti üzerinde makine öğrenmesini uygulayan bir program geliştirilmiştir. Programın yazılmasında Python programlama dili tercih edilmiştir. Python derleyicisi olarak ise görsel arayüz desteği olması, email okuma ve gönderme işlemleri için bir kütüphane içermesi ve pek çok makine öğrenmesi algoritması için kütüphane desteği sunmasından dolayı Spyder (the scientific python development environment-bilimsel python geliştirme ortamı) tercih edilmiştir (Spyder). Programın akış şeması Şekil 3.7.'de verilmiştir. Bu bölümde programın aşamaları açıklanacaktır.



Şekil 3.7. Programın akış şeması

Programda kullanılacak olan kütüphaneler programın başında eklenmelidir. Kullanılan kütüphaneler yeri geldikçe açıklanacaktır. Programın ilk aşamasında gelen eposta olup olmadığı kontrol edilmektedir. Eposta okuma ve göndermede kullanılan kütüphaneler şunlardır;

```
import email

import imaplib

import smtplib

from itertools import chain

from email.mime.text import MIMEText

from email.mime.multipart import MIMEMultipart

from email.mime.base import MIMEBase

from email import encoders
```

Programda epostaları okumak ve göndermek için bir eposta hesabına ihtiyaç vardır. Bu amaçla ada1a.project@gmail.com hesabı oluşturulmuştur. Program oluşturulan eposta hesabı ve şifresi ile imaps protokolünü kullanarak imap.gmail.com adresine imap ssl portu olan 993 numaralı port üzerinden bağlanır. Bağlantı kurulduktan sonra sadece ada1a.project@gmail.com adresinden gelen ve konusu ESP8266 olan epostalar okunur. Tüm epostalar bir uid numarasına sahiptir. En büyük uid numarası programın her çalıştırılmasında bir dosyaya kaydedilir ve daha sonraki döngülerde bu numaraya bakılır. Eğer en son uid numarasından büyük uid'ye sahip epostalar var ise sadece bu epostalar okunur ve belirlenen bir dosyaya kaydedilir. Epostanın hangi dosyaya kaydedileceği ise eposta içeriğindeki sensor numarasına bakılarak belirlenir. Eğer temsili sensör numarası olarak kullandığımız SA-1B2 kelimesi eposta içeriğinde var ise txt uzantılı belirli dosyaya eklenir. Diğer sensör numaraları belirlenen diğer dosyalara kaydedilebilir. Burada veri aktarımının eposta ile yapılmasının avantajları öne çıkmaktadır. Örneğin gönderilen veriler bir program tarafından okunmasa dahi süresiz olarak eposta sunucularında kayıtlı tutulmaktadır. Eposta gönderme ve okuma işlemlerinde yüksek seviyede şifreleme uygulanmaktadır ve bu ise kullanıcılara yüksek veri güvenliği sağlamaktadır. Ayrıca hücresel internet

şebekelerinde genellikle sabit (statik) ip numaraları kullanılmadığından sabit ip satın alınması ve port yönlendirme işlemleri yapılması gerekmektedir. Kullanılan yöntemde ise bu işlemlere gerek kalmamakta ve Spyder editörünün kurulu olduğu her bilgisayarda program eposta okuma ve gönderme yapabilmektedir. Programın devamında verilerin kayıtlı olduğu dosyalardan okunması işlemi yapılmaktadır. Verinin okunmasında ve üzerinde düzenleme işlemlerinin yapılmasında kullanılan kütüphaneler şunlardır;

import os , import pandas as pd, import datetime as dt

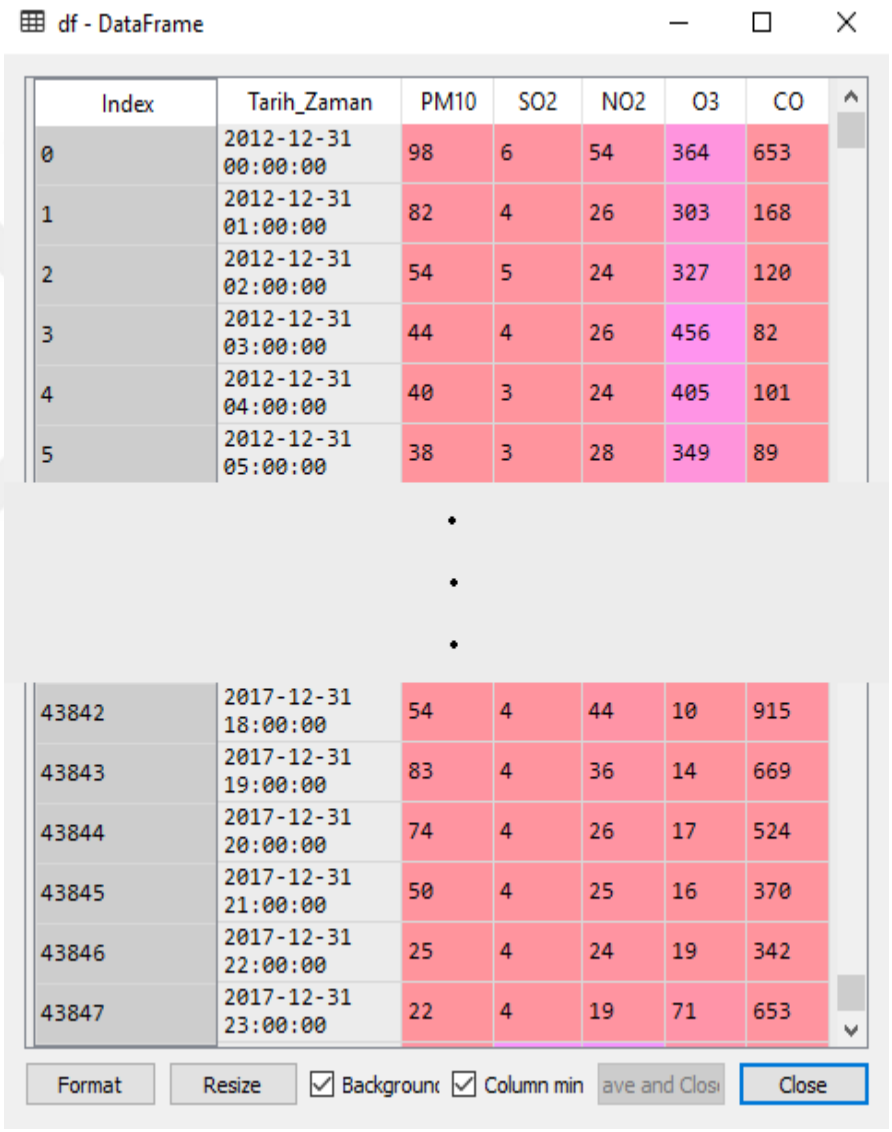
Pandas kütüphanesi veri madenciliği işlemleri için özel olarak geliştirilmiştir ve pek çok katılımcı tarafından gönüllü olarak geliştirilmeye devam edilmektedir. Bu kütüphanedeki pd.read_csv fonksiyonu ile kayıtlı dosyadan okunan veri bir veri çerçevesi (data frame) formatına dönüştürülmektedir. Kayıtlı hava kirliliği veri setinin txt dosyasındaki görünümü ve programdaki veri çerçevesi görünümü Şekil 3.8. ve Şekil 3.9'da gösterilmiştir.

```
,Tarih_Zaman,PM10,SO2,NO2,O3,CO
0,2012-12-31 00:00:00,98,6,54,364.0,653.0
1,2012-12-31 01:00:00,82,4,26,303.0,168.0
2,2012-12-31 02:00:00,54,5,24,327.0,120.0
3,2012-12-31 03:00:00,44,4,26,456.0,82.0
4,2012-12-31 04:00:00,40,3,24,405.0,101.0
5,2012-12-31 05:00:00,38,3,28,349.0,89.0
                    •
                    •
                    •
43843,2017-12-31 19:00:00,83,4,36,14.0,669.0
43844,2017-12-31 20:00:00,74,4,26,17.0,524.0
43845,2017-12-31 21:00:00,50,4,25,16.0,370.0
43846,2017-12-31 22:00:00,25,4,24,19.0,342.0
43847,2017-12-31 23:00:00,22,4,19,71.0,653.0
```

Şekil 3.8. Veri setinin txt dosyasındaki görünümü

Veri setindeki veriler birbirlerinden virgülle ayrılmışlardır. Program dosyayı okurken belirtilen ayırma karakterine göre verileri ayırt etmektedir.

Programda indeks sütunu ilk sütun (0.) olarak seçilmiştir. Tarih_Zaman isimli sütun programda tarih ve saat olarak tanımlanabilmesi için belirtilmiştir. Program tarafından bu bilgilerle oluşturulan veri çerçevesi yapısı Şekil 3.9.'da gösterilmiştir. Hava ölçüm istasyonundaki bakım çalışmaları ve sensör arızalarından kaynaklı olduğu düşünülen eksik veriler mevcuttur. Bu veriler her bir sütunun ortalama (mean) değeriyle tamamlanmıştır.



Şekil 3.9. Program ara yüzündeki veri çerçevesi yapısı

Bu işlemden sonra sensör verilerinin kaydedildiği dosya okunur ve veri çerçevesine dönüştürülür. Bu veri çerçevesinden sensor, NO, NOX değerleri

kullanılmayacağı için çıkarılmakta ve Şekil 3.10'da gösterilen veri çerçevesi oluşturulmaktadır.

Index	Tarih_Zaman	PM10	SO2	NO2	O3	CO
43848	2018-01-01 00:00:00	97	113	402	90	1005

Şekil 3.10. Sensör bilgilerine ait veri çerçevesi

Hava kirliliği veri setine ait veri çerçevesindeki Index sütunun son satırına ait indeks numarası alınarak bir değişkene aktarılır ve bir artırılır. Bu değer sensör verilerine ait veri çerçevesinin Index sütununda aranır ve var ise veri setine ait veri çerçevesinin sonuna eklenir. Değer yok ise veri gelene kadar program beklemektedir. Bu işlemden sonra hava kalite indeksi hesaplanmaktadır. Hava kalite indeksi, havadaki kirletici gazların insan ve diğer canlıların sağlıklarına etkisinin derecesini ifade etmektedir. Birleşik Devletlere bağlı çevre koruma ajansının (EPA- Unites States Environmental Protection Agency) yayımladığı değerler Çizelge 3.1.'de gösterilmiştir. Ülkemizde hava kalite indeksi hesaplamasında kullanılan değerler (Çizelge 3.2.) EPA değerlerinin ülkemiz mevzuatına ve sınır değerlerine göre yeniden hesaplanmasıyla belirlenmiştir (Havaizleme).

Çizelge 3.1. EPA hava kalite indeksi değerleri

Hava Kalitesi İndeksi (AQI) EPA Değerleri	Sağlık Endişe Seviyeleri	Renkler	Anlamı
0-50	İyi	Yeşil	Hava kalitesi iyidir ve herhangi bir risk söz konusu değildir
51-100	Orta	Sarı	Hava kalitesi iyi ancak bazı kirleticilere hassaslığı olan az sayıda insan için orta seviyede risk oluşturabilir
101-150	Hassas	Turuncu	Kamuyu etkileyecek bir durum yoktur ancak hassas gruptaki insanlar için risk söz konusudur
151-200	Sağlıksız	Kırmızı	İnsanların sağlığını olumsuz etkileyecek durumda kirli bir hava vardır. Hassas gruplar için ciddi etkiler oluşur
201-300	Kötü	Mor	Nüfusun tamamını ciddi derece olumsuz etkileyecek derecede kirli bir hava vardır
301-500	Tehlikeli	Kahverengi	Sağlık Alarmı: Nüfusun tamamında çok ciddi sağlık sorunlarına yol açacak hava kirliliği söz konusudur

Hava kirleticilerinin ölçüm değerleri µg/m^3 cinsinden ifade edilmektedir. Bu bir metre küp havada kaç mikrogram oldukları anlamına gelmektedir. Çizelge 3.2.'de verilen ulusal değerlere göre hava kalite indeksi hesaplanırken SO_2 ve NO_2 hava kirleticilerinin saatlik ortalama değerleri, CO ve O_3 hava kirleticilerinin son 8 saate ait ortalama değerleri, PM10 hava kirleticisinin ise son 24 saate ait ortalama değeri hesaba katılmaktadır. Bunun için veri setine ait veri çerçevesinin son 24 saatlik kısmı alınarak ayrı bir veri çerçevesi oluşturulmaktadır. Bu veri çerçevesinde 2018 yılının ilk saatine ait sensör değerinin eklenmiş olduğu görülmektedir (Şekil 3.11.).

Çizelge 3.2. Ulusal hava kalitesi indeksi kesme noktaları

Hava Kalite İndeksi	**SO_2 [µg/m^3]**	**NO_2 [µg/m^3]**	**CO [µg/m^3]**	**O_3 [µg/m^3]**	**PM10 [µg/m^3]**
	1 Sa. Ort.	1 Sa. Ort.	8 Sa. Ort.	8 Sa. Ort.	24 Sa. Ort.
İyi	0-100	0-100	0-5500	0-120	0-50
Orta	101-250	101-200	5501-10000	121-160	51-100
Hassas	251-500	201-500	10001-16000	161-180	101-260
Sağlıksız	501-850	501-1000	16001-24000	181-240	261-400
Kötü	851-1100	1001-2000	24001-32000	241-700	401-520
Tehlikeli	>1100	>2000	>32000	>700	>520

Programda hava kalite indeksleri için sayısal değer ataması yapılmıştır. İyi indeksi için 1, orta indeksi için 2, hassas indeksi için 3, sağlıksız indeksi için 4, kötü indeksi için 5 ve tehlikeli indeksi için 6 sayısal değerleri atanmıştır. Hava kalite indeksi hesaplaması işlem sonuçlarının yazılması amacıyla yeni bir veri çerçevesi oluşturulmuştur. 2018 yılının ilk saatine ait sonuçları içeren veri çerçevesi Şekil 3.12.'de gösterilmiştir. SO_2 ve NO_2 hava kirleticilerinin saatlik ortalama değerleri sırasıyla 113 ve 402 olarak hesaplanmıştır. Çizelge 3.2.'ye göre SO_2'nin hava kalite indeksi orta (2), NO_2'nin hava kalite indeksi hassas (3) seviyesindedir. CO ve O_3 hava kirleticilerinin son 8 saate ait ortalama değerleri sırasıyla 632 ve 34 olarak hesaplanmıştır. Çizelge 3.2.'ye göre CO'in hava kalite indeksi iyi (1), O_3'un hava kalite indeksi iyi (1) seviyesindedir. Son olarak PM10 hava kirleticisinin son 24 saate ait ortalama değeri 45 olarak hesaplanmıştır. Çizelge 3.2.'ye göre PM10'un hava kalite indeksi iyi (1)'dir. Bu değerlerden en büyüğü yani en kirlisi saatlik hava kalite indeksini belirlemektedir. Burada 3 değeri en yüksek olduğundan hava kalite indeksi 3 olarak hesaplanmakta ve HKI isimli sütuna yazılmaktadır (Şekil 3.12.).

df24 - DataFrame

Index	Tarih_Zaman	PM10	SO2	NO2	O3	CO
43825	2017-12-31 01:00:00	74	4	14	6	393
43826	2017-12-31 02:00:00	67	4	9	16	267
43827	2017-12-31 03:00:00	51	4	9	20	272
43828	2017-12-31 04:00:00	40	4	6	17	249
43829	2017-12-31 05:00:00	35	3	6	16	191
43830	2017-12-31 06:00:00	26	3	9	17	177
43831	2017-12-31 07:00:00	31	3	17	10	165
43832	2017-12-31 08:00:00	25	3	25	11	302
43833	2017-12-31 09:00:00	29	3	19	17	220
43834	2017-12-31 10:00:00	28	3	20	13	285
43835	2017-12-31 11:00:00	37	3	19	20	292
43836	2017-12-31 12:00:00	44	4	17	27	310
43837	2017-12-31 13:00:00	45	4	26	23	462
43838	2017-12-31 14:00:00	49	4	24	28	411
43839	2017-12-31 15:00:00	38	4	29	25	461
43840	2017-12-31 16:00:00	31	4	26	29	382
43841	2017-12-31 17:00:00	35	4	26	36	578
43842	2017-12-31 18:00:00	54	4	44	10	915
43843	2017-12-31 19:00:00	83	4	36	14	669
43844	2017-12-31 20:00:00	74	4	26	17	524
43845	2017-12-31 21:00:00	50	4	25	16	370
43846	2017-12-31 22:00:00	25	4	24	19	342
43847	2017-12-31 23:00:00	22	4	19	71	653
43848	2018-01-01 00:00:00	97	113	402	90	1005

Format Resize Background colo Column min/max Save and Close Close

Şekil 3.11. Son 24 saat verilerine ait veri çerçevesi

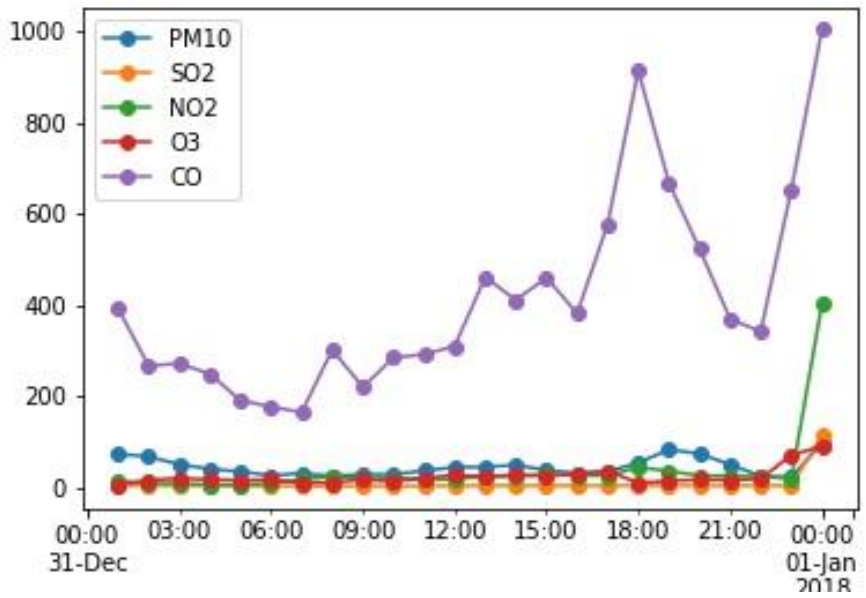
Açıklanan hava kalite indeksi hesaplama işlemleri kullanılan veri seti üzerinde daha öncesinde uygulanmış ve sonuçlar bir klasöre kaydedilmiştir. Bu değerler programın başında ayrı bir veri çerçevesine alınmıştır. Bu sayede program her seferinde tüm veri seti üzerinde değil sadece son saate ait değerler üzerinde hesaplama işlemi uygulamaktadır.

Tarih_Saat	SO2	NO2	CO_8	O3_8	PM10_24
2018-01-01 00:00:00	113	402	632	34	45

v1	v2	v3	v4	v5	HKI
2	3	1	1	1	3

Şekil 3.12. 2018 yılı ilk saat verisi için hesaplanan hava kalite indeksi

Şekil 3.12.'deki son 24 saate ait verileri içeren veri çerçevesi ve matplotlib kütüphanesi kullanılarak bir grafik oluşturulmaktadır (Şekil 3.13.). Şekil 3.12'de verilen veri çerçevesinden HKI değeri okunarak, kullanıcıya Adana valilik istasyonunun hava kalitesini ve son 24 saate ait grafiği ek olarak içeren bir bilgilendirme epostası gönderilmektedir.



Şekil 3.13. Kullanıcıya eposta olarak gönderilen grafik

Son gelen saate ait verilerinde eklenmiş olduğu veri seti ilgili dosyaya kaydedilmektedir. Ayrıca son saate ait hava kalite indeksi işlem sonuçları da ilgili veri çerçevesine eklenmiş ve ilgili dosyaya kaydedilmiştir. Bu işlemlerden sonra hava kalite indeksi sonuçlarını içeren dosyadaki veriler üzerinde makine öğrenmesi regresyon algoritmaları uygulanacak, sonuçlar incelenecektir.

3.5. Makine Öğrenmesi

3.5.1. Makine öğrenmesi uygulanacak veri seti

Program bu aşamada makine öğrenmesinin uygulanacağı veri setinin kayıtlı olduğu dosyayı okuyacaktır. Veri setinin kayıtlı olduğu txt dosyasının yapısı Şekil 3.14.'te verilmiştir.

```
,Tarih_Saat,SO2,NO2,CO_8,O3_8,PM10_24,v1,v2,v3,v4,v5,HKI
0,2013-01-01 00:00:00,7.0,56.0,2128.0,290.0,109.0,1.0,1.0,1.0,5.0,3.0,5.0
1,2013-01-01 01:00:00,6.0,54.0,1919.0,299.0,108.0,1.0,1.0,1.0,5.0,3.0,5.0
2,2013-01-01 02:00:00,5.0,41.0,1139.0,296.0,108.0,1.0,1.0,1.0,5.0,3.0,5.0
3,2013-01-01 03:00:00,5.0,28.0,531.0,295.0,109.0,1.0,1.0,1.0,5.0,3.0,5.0
4,2013-01-01 04:00:00,5.0,24.0,370.0,278.0,109.0,1.0,1.0,1.0,5.0,3.0,5.0
5,2013-01-01 05:00:00,4.0,23.0,258.0,279.0,109.0,1.0,1.0,1.0,5.0,3.0,5.0
                                    .
                                    .
                                    .
43820,2017-12-31 20:00:00,4.0,26.0,550.0,23.0,53.0,1.0,1.0,1.0,1.0,2.0,2.0
43821,2017-12-31 21:00:00,4.0,25.0,539.0,22.0,50.0,1.0,1.0,1.0,1.0,,1.0
43822,2017-12-31 22:00:00,4.0,24.0,530.0,21.0,47.0,1.0,1.0,1.0,1.0,1.0,1.0
43823,2017-12-31 23:00:00,4.0,27.0,529.0,20.0,45.0,1.0,1.0,1.0,1.0,1.0,1.0
43824,2018-01-01 00:00:00,113.0,402.0,632.0,34.0,45.0,2.0,3.0,1.0,1.0,1.0,3.0
```

Şekil 3.14. Makine öğrenmesi için kullanılan veri setinin txt formatı

Program veri setindeki SO2, NO2, CO_8, O3_8 ve PM10_24 isimli sütunlardaki verileri algoritmalar için girdi verileri, HKI isimli sütundaki verileri ise çıktı verileri olarak kullanmak üzere okumakta ve veri çerçevesi formatına dönüştürmektedir. SO2 ve NO2 değerleri hava kirleticilerinin saatlik ortalama değerlerini, CO_8 ve O3_8 değerleri hava kirleticilerinin son 8 saate ait ortalama değerlerini ve PM10_24 değeri hava kirleticinin son 24 saate ait ortalama değerini ifade etmektedir. HKI değeri hesaplanmış olan hava kalite indeksidir. Veri setine sklearn.preprocessing kütüphanesinin MinMaxScaler fonksiyonu kullanılarak asgari – azami (min-max) normalleştirme (Eşitlik (3.1)) uygulanmış ve veriler 0-1 aralığında ölçeklenmiştir. Veri setinin %75'i algoritmaların eğitiminde %25'i ise eğitilen algoritmaların test edilmesinde kullanılacaktır. Bu işlem için sklearn.cross_validation kütüphanesinin train_test_split fonksiyonu kullanılmıştır. Bu fonksiyon ile veri

setinden rastgele seçimler yapılarak verilerin %75'i eğitim %25'i test verileri olmak üzere 2 ayrı girdi-çıktı kümesi oluşturulmuştur.

$$v' = \frac{v - \min_A}{\max_A - \min_A}(\text{new\_max}_A - \text{new\_min}_A) + \text{new\_min}_A \quad (3.1)$$

3.5.2. Makine öğrenmesi algoritmaları

Çalışmada veri seti üzerinde denetimli öğrenme yöntemlerinden olan regresyon yöntemleri kullanılmıştır. Denetimli öğrenme işleminde algoritma girdilere karşılık çıktıları belli olan bir veri setiyle eğitilmektedir. Test verileri uygulanarak eğitilen algoritmanın tahmin başarısı incelenmektedir. Makine öğrenmesi regresyon algoritmaları Sklearn kütüphanesindeki (Scikit-learn) regresyon modelleri kullanılarak çalıştırılmıştır. Çalışmada kullanılan regresyon algoritmaları şunlardır;

3.5.2.1. Çoklu doğrusal regresyon (multiple linear regression- mlr)

Birden fazla sayıda bağımsız değişken girdiden bağımlı değişken olan çıktının doğrusal olarak hesaplandığı eşitliktir (Eşitlik (3.2))(Stanton ve ark., 2001). Çalışmadaki bağımsız değişkenler hava kirletici değerleri, bağımlı değişken hava kalite indeksi değeridir. Modelin geliştirilmesinde sklearn.linear_model kütüphanesinin LinearRegression modeli kullanılmıştır. Modele eğitim verisi uygulanarak değişken katsayıları belirlenmektedir**.**

$$Yi = \beta 0 + \beta 1\, Xi1 + \beta 2\, Xi2 + \cdots + \beta k\, Xik + \varepsilon i, \qquad i = 1, 2, \ldots, n \quad (3.2)$$

3.5.2.2. Karar ağacı regresyon (decision tree regression- dtr)

Breiman'ın yayınladığı sınıflandırma ve regresyon ağaçları (Breiman ve ark. 1984) ve Quinlan'ın yayınladığı karar ağaçları çalışmaları temel alınarak geliştirilen bir algoritmadır (Quinlan, 1986). Modelin geliştirilmesinde sklearn.tree kütüphanesinin DecisionTreeRegressor modeli kullanılmıştır. Bir karar ağacı yaprak düğümleri ve karar düğümlerinden oluşmaktadır. Karar ağacı regresyonunda sınıflandırmada kullanılan bilgi kazanımı yerine standart sapma (Eşitlik (3.3)) kullanılmaktadır. İlk olarak çıktı verisinin standart sapması hesaplanmaktadır. Sonrasında girdi verileriyle çıktı verisinin ikili standart sapma değerleri (Eşitlik (3.4)) hesaplanmaktadır. Her birisinin sonucu çıktı verisinin standart sapma değerinden (Eşitlik (3.5)) çıkarılmaktadır. En büyük SDR'ye sahip olan küme küme kök olarak seçilmektedir. Bu adımlar her düğümde tekrarlanarak karar ağacı oluşturulmaktadır.

$$S = \sqrt{\frac{\sum(x-\mu)^2}{n}} \tag{3.3}$$

$$S(T,X) = \sum_{c \in X} P(c)S(c) \tag{3.4}$$

$$SDR(T,X) = S(T) - S(T,X) \tag{3.5}$$

3.5.2.3. K-en yakın komşu regresyonu (k-nearest neighbor regression- k-nnr)

Modelin geliştirilmesinde sklearn.neighbors kütüphanesinin KNeighborsRegressor modeli kullanılmıştır. Her bir test verisinin girdi değerleri kullanılarak eğitim verileriyle uzaklıkları hesaplanmaktadır. Uzaklık mesafesi, öklid uzaklık fonksiyonu (Eşitlik (3.6)), manhatın uzaklık fonksiyonu (Eşitlik (3.7)) ya da minkowski uzaklık fonksiyonundan (Eşitlik (3.8)) birisi kullanılarak hesaplanmaktadır (Alkhatib ve ark., 2013). K tane en yakın uzaklıktaki verinin çıktı değerleri ortalaması test verisinin çıktısı olarak hesaplanmaktadır. Gerçek sonuç ile çıkan sonuç arasındaki fark regresyon algoritmasının başarı oranını belirleyecektir.

$$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2} \quad (3.6)$$

$$\sum_{i=1}^{k}|x_i - y_i| \quad (3.7)$$

$$\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{\frac{1}{q}} \quad (3.8)$$

3.5.2.4. Destek vektör regresyon (support vector regression- svr)

Vapnik tarafından ortaya atılan destek vektör makinelerinin regresyon modelini Smola geliştirmiştir (Vapnik 2000) (Smola ve ark., 2004). Destek vektör regresyonunda birbirlerinden farklı özelliklere sahip özniteliklerin, yakınlarından geçen bir doğru veya eğriyle birbirlerinden ayrılması amaçlanmaktadır (Eşitlik (3.9), Eşitlik (3.10), Eşitlik (3.11), Eşitlik (3.12)). Bir doğru ile ayrılamayan öznitelikler, çeşitli çekirdek fonksiyonları ile ayrılabilmektedir (Eşitlik (3.13), Eşitlik (3.14), Eşitlik (3.15), Eşitlik (3.16)).

$$y = wx + b \quad (3.9)$$

$$\min \frac{1}{2}\|w\|^2 \quad (3.10)$$

$$y_i - wx_i - b \leq \varepsilon \quad (3.11)$$

$$wx_i + b - y_i \leq \varepsilon \quad (3.12)$$

$$y = \sum_{i=1}^{N}(a_i - a_i^*)\, K(x_i, x) + b \quad (3.13)$$

$$k(x, x') = \langle x, x' \rangle^P \quad (3.14)$$

$$k(x, x') = (\langle x, x' \rangle + c)^P \quad (3.15)$$

$$k(x, x') = \tanh(\vartheta + \kappa\langle x, x' \rangle) \quad (3.16)$$

3.5.2.5. Yapay sinir ağı regresyon (neural network regression- nnr)

Farklı sayıda nöronlar içerebilen giriş, çıkış ve gizli katmanlardan oluşan bir modeldir. Katmanlar ve nöronlar birbirleriyle bağlantılıdır. Nöronlar ve bağlantılar ağırlık katsayılarına sahiptir. Nöronlar ayrıca bir aktivasyon fonksiyonuna (Eşitlik (3.18), Eşitlik

(3.19), Eşitlik (3.20)) sahiptir. Giriş katmanından yapay sinir ağına girilen veriler ağırlık değerleriyle çarpılarak ve nöronlardaki aktivasyon fonksiyonlarından geçerek çıkış katmanına ulaşır (Eşitlik (3.17)). Her eğitim verisi girişinde ağırlıklar güncellenir (Murtagh, 1991). Modelin geliştirilmesinde sklearn.neural_network kütüphanesinin MLPRegressor modeli kullanılmıştır.

$$y = f(w1.X1 + w2.X2 + b) \quad (3.17)$$

$$y = x \quad (3.18)$$

$$y = \frac{1}{1+e^{-x}} \quad (3.19)$$

$$y = \frac{1-e^{-2x}}{1+e^{2x}} \quad (3.20)$$

3.5.2.6. Rastgele orman regresyon (random forest regression- rfr)

Çok sayıda karar ağacı yapısı oluşturulmakta ve ürettikleri sonuçların ortalaması alınmaktadır (Breiman, 2001). Modelin geliştirilmesinde sklearn.ensemble kütüphanesinin RandomForestRegressor modeli kullanılmıştır. Oluşturulacak karar ağacı sayısı 30 olarak belirlenmiştir.

3.5.2.7. Yığın regresyon (stacking regression- stckr)

Çok sayıda regresyon modeli birlikte kullanılmakta ve daha iyi bir tahmin sonucu elde edilmeye çalışılmaktadır (Breiman, 1996). Her bir regresyon algoritması eğitim veri setiyle eğitilir. Test verisi eğitilen algoritmalara uygulanarak bir tahmin çıktısı elde edilir. Tahmin değerleri ile eğitimde kullanılan veriler yeni eğitim verisi olmaktadır. Bu eğitim verisi son olarak bir regresyon modelinin eğitilmesinde kullanılmakta elde edilen tahminler genellikle daha iyi olmaktadırlar. Modelin geliştirilmesinde mlxtend.regressor kütüphanesinin StackingRegressor modeli kullanılmıştır.

3.5.2.8. Uyumlu artırıcı regresyon (adaboost regression- adbr)

Tahmin işleminde kullanılacak regresyon modeli olan karar ağacı regresyonu veri seti ile eğitilmektedir. Eğitim sonucunda yanlış olarak tahmin edilen durumların göreceli ağırlıkları artırılarak bir sonraki eğitimde daha ön planda olmaları sağlanmaktadır (Freund ve Schapire, 1997). Gerekli durdurma şartları oluşana kadar regresyon modeli çalıştırılmaya devam edilmektedir. Modelin geliştirilmesinde sklearn.ensemble kütüphanesinin AdaBoostRegressor modeli kullanılmıştır.

3.5.2.9. Eğimli artırıcı regresyon (gradient boosting regression- grbr)

Tahmin işleminde kullanılacak regresyon modeli olan karar ağacı regresyonu veri seti ile eğitilmektedir. Tahmin sonucu oluşan hata bir sonraki modelin hedef çıktı kümesine eklenmektedir. Bu modelin ürettiği tahmin hataları ise yine bir sonraki modelin hedef kümesine eklenmektedir. Tüm modellerin ürettikleri tahminler toplanarak sonuç değeri elde edilecektir. (Freidman, 1999). Modelin geliştirilmesinde sklearn.ensemble kütüphanesinin GradientBoostingRegressor modeli kullanılmıştır.

3.5.2.10. Örneklemeli toplam regresyon (bagging regression- bagr)

Regresyon modeli veri setinden alınan farklı altkümeler ile eğitime tabi tutulmaktadır (Breiman, 1996). Altkümelerin oluşturulması yerine koyarak örnekleme yöntemiyle sağlanmaktadır ve bu sayede aynı veriler farklı altkümelerde bulunabilmektedir. Farklı altkümelerle eğitilen temel regresyon modelinin ürettiği tahmin değerlerinin ortalaması alınmaktadır. Modelin geliştirilmesinde sklearn.ensemble kütüphanesinin BaggingRegressor modeli kullanılmıştır.

3.6. Başarı Değerlerini Hesaplama

Regresyon algoritmalarının hava kalite indeksini tahmin başarıları değerlendirilirken kullanılan ölçütler; belirlilik katsayısı (r^2), ortalama mutlak hata (omh - mean absolute error (mae)) ve ortalama karesel hatadır (okh – mean squared error (mse)). Belirlilik katsayısı (r^2), regresyon karelerin toplamının genel karelerin toplamına bölünmesiyle hesaplanan istatistiksel bir değerdir (Eşitlik (3.21)). Regresyon algoritmasının ürettiği hava kalite indeksi tahmin değerlerinin gerçek indeks değerlerine yakınlığının ne oranda olduğunun bir göstergesidir (Irmak ve Aydilek, 2019). Değerin 1'e yakın olması algoritmanın başarısının yüksek olduğunu göstermektedir.

$$r^2 = \frac{\sum_i(\hat{y}_i-\bar{y})^2}{\sum_i(y_i-\bar{y})^2} = 1 - \frac{\sum_i(y_i-\hat{y}_i)^2}{\sum_i(y_i-\bar{y})^2} \quad (3.21)$$

Ortalama mutlak hata, hava kalite indeksinin gerçek değerleri ile regresyon algoritmasının ürettiği tahmin değerleri arasındaki farkların mutlak değerlerinin ortalamasının alınmasıyla hesaplanmaktadır (Eşitlik (3.22)). Değerin 0'a yakın olması hata oranın düşük olduğu anlamına gelmektedir.

$$\frac{1}{n}\sum_{j=1}^{n}\left|y_j - \hat{y}_j\right| \quad (3.22)$$

Ortalama karesel hata, hava kalite indeksinin gerçek değerleri ile regresyon algoritmasının ürettiği tahmin değerleri arasındaki farkların karelerinin ortalamasının alınmasıyla hesaplanmaktadır. (Eşitlik (3.23)). Değerin 0'a yakın olması hata oranın düşük olduğu anlamına gelmektedir.

$$\frac{1}{n}\sum_{j=1}^{n}\left(y_j - \hat{y}_i\right)^2 \quad (3.23)$$

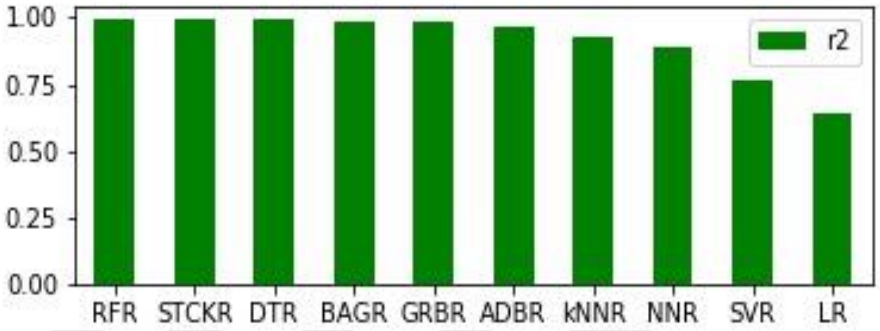
4. ARAŞTIRMA BULGULARI ve TARTIŞMA

Regresyon algoritmalarına ait belirlilik katsayısı (r^2), ortalama karesel hata (okh – mean squared error (mse)), ortalama mutlak hata (omh- mean absolute error (mae)) ve saniye cinsinden hesaplama süresi değerleri büyükten küçüğe sıralı olarak Çizelge 4.1.'de verilmiştir. Belirlilik katsayısı (r^2) ölçütüne göre en başarılı algoritmanın rastgele orman regresyonu olduğu görülmüştür. Topluluk regresyon algoritmaları olan rastgele orman, yığın, örneklemeli toplam, eğimli artırıcı ve uyumlu artırıcı regresyonlarının en iyi sonuçları ürettikleri görülmüştür. Belirlilik katsayısı(r^2) değerleri iyi olan regresyon algoritmalarının ortalama mutlak hata (omh) ve ortalama karesel hata (okh) değerlerinin düşük ve 0'a daha yakın olduğu görülmüştür. Rastgele orman regresyonun hata değerlerinin en iyi değerler olduğu görülmektedir. Hesaplama süresi en hızlı olan regresyon algoritması 0.00699 değeriyle doğrusal regresyonu olmuştur (Irmak ve Aydilek, 2019).

Çizelge 4.1. Regresyon algoritmalarının başarı değerleri

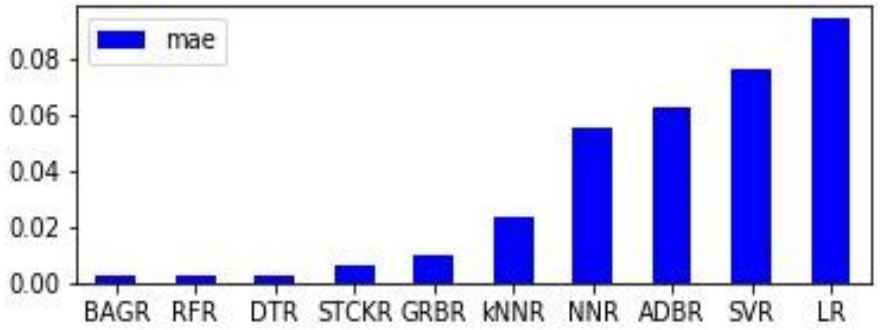
Regresyon Algoritması	r^2	mae	mse	süre
Rastgele Orman (RFR)	**0.99049**	**0.00275**	**0.00037**	1.12502
Yığın (STCKR)	0.99047	0.00442	**0.00037**	21.42568
Karar Ağacı (DTR)	0.99036	0.00293	0.00038	0.04088
Örneklemeli Toplam (BAGR)	0.99006	0.00283	0.00039	1.01229
Eğimli Artırıcı (GRBR)	0.98704	0.00967	0.00051	0.33809
Uyumlu Artırıcı (ADBR)	0.96364	0.02447	0.00142	0.57346
K-En Yakın Komşu (KNNR)	0.92685	0.02331	0.00286	0.29321
Yapay Sinir Ağı (NNR)	0.88783	0.05604	0.00439	1.08011
Destek Vektör (SVR)	0.76300	0.07609	0.00927	13.88685
Doğrusal (LR)	0.64496	0.09443	0.01389	**0.00699**

Şekil 4.1.'de belirlilik katsayıları (r^2) bakımından regresyon algoritmalarının başarı sıralamaları verilmiştir. Bu değerin 1'e yakın olması algoritmanın tahmin başarısının yüksek olduğunu ifade etmektedir. Belirlilik katsayısı bakımından en iyi regresyon algoritması 0.99049 değeriyle topluluk regresyon algoritmalarından olan rastgele orman regresyonu olmuştur.



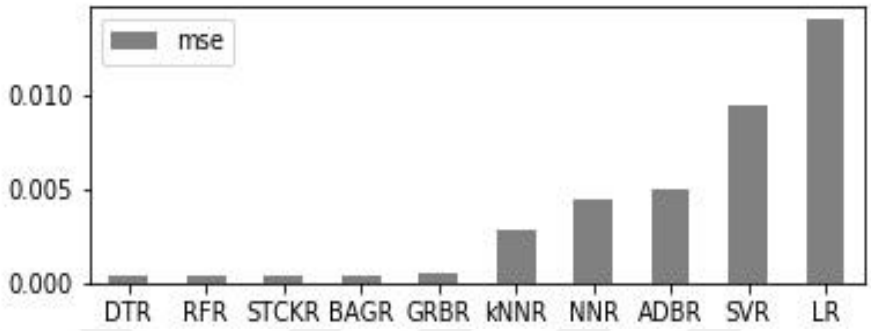
Şekil 4.1. Belirlilik katsayıları bakımından regresyon algoritmalarının sıralaması

Şekil 4.2.'de ortalama mutlak hata (OMH) değerleri bakımından regresyon algoritmalarının başarı sıralamaları verilmiştir. Bu değerin 0'a yakın olması algoritmanın tahmin başarısının yüksek olduğu anlamına gelmektedir.



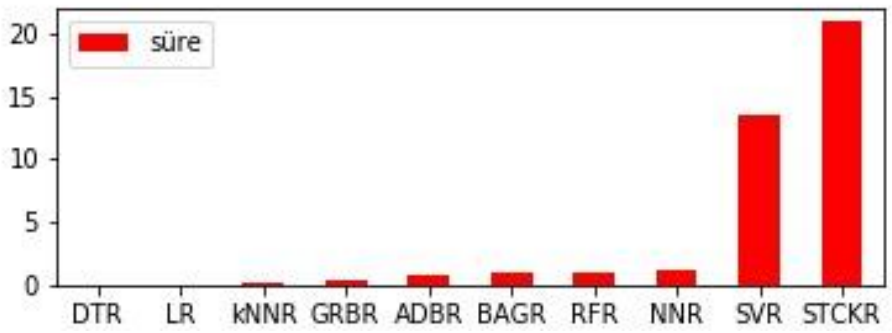
Şekil 4.2. Ortalama mutlak hata bakımından regresyon algoritmalarının sıralaması

Şekil 4.3.'te ortalama karesel hata (OKH) değerleri bakımından regresyon algoritmalarının başarı sıralamaları verilmiştir. Bu değerin 0'a yakın olması algoritmanın tahmin başarısının yüksek olduğu anlamına gelmektedir.



Şekil 4.3. Ortalama karesel hata bakımından regresyon algoritmalarının sıralaması

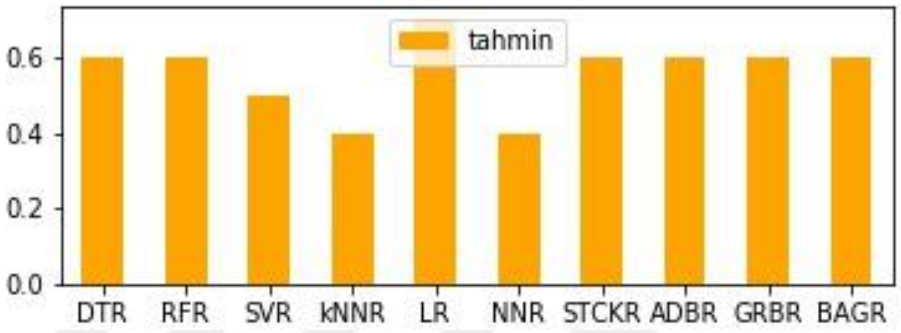
Şekil 4.4.'te hesaplama süreleri bakımından regresyon algoritmalarının başarı sıralamaları verilmiştir. Destek vektör ve yığın regresyon regresyonu algoritmalarının hesaplama sürelerinin diğer algoritmalara göre daha uzun olduğu görülmektedir.



Şekil 4.4. Hesaplama süreleri bakımından regresyon algoritmalarının sıralaması

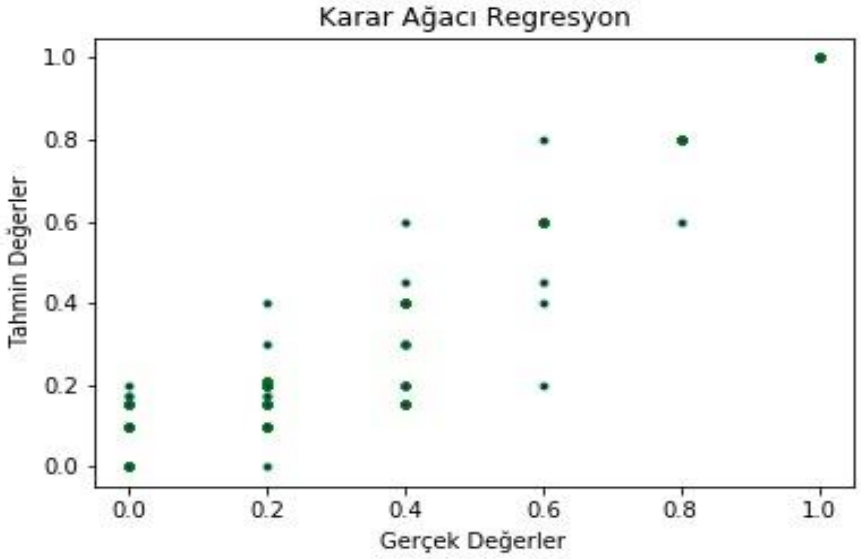
Grafiklerde görülen gerçek değerler 0, 0.2, 0.4, 0.6, 0.8 ve 1'dir. Bu değerler hava kalite indeks değerlerinin karşılığı olarak belirlediğimiz sayısal değerler olan 1,2,3,4,5 ve 6 değerlerinin 0-1 arasında olacak şekilde min-maks normalleştirmeye tabi tutulmalarıyla oluşturulmuştur.

Şekil 4.5.' te eğitilmiş olan algoritmaların örnek bir veri için tahmin ettikleri hava kalite indeks değerleri görülmektedir. Algoritmaların çoğunluğunun normalleştirilmiş değer olan 0.6 değerini yani 4. derece sağlıksız hava kalite indeksini buldukları görülmektedir. Farklı tahmin değerleri hatalı sonuçlar tahmin edildiğini göstermektedir.

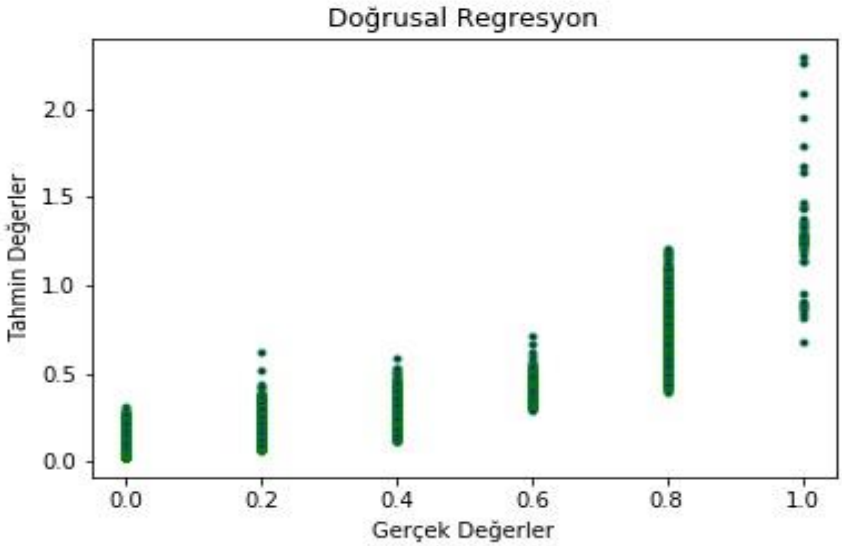


Şekil 4.5. Regresyon algoritmalarının ürettikleri tahmin değerleri

Regresyon algoritmalarının tahmin ettiği indeks değerleri ile test verisindeki gerçek indeks değerlerini bir arada gösteren grafikler sırasıyla verilmiştir (Şekil 4.6., Şekil 4.7., Şekil 4.8., Şekil 4.9., Şekil 4.10., Şekil 4.11., Şekil 4.12., Şekil 4.13., Şekil 4.14., Şekil 4.15.)

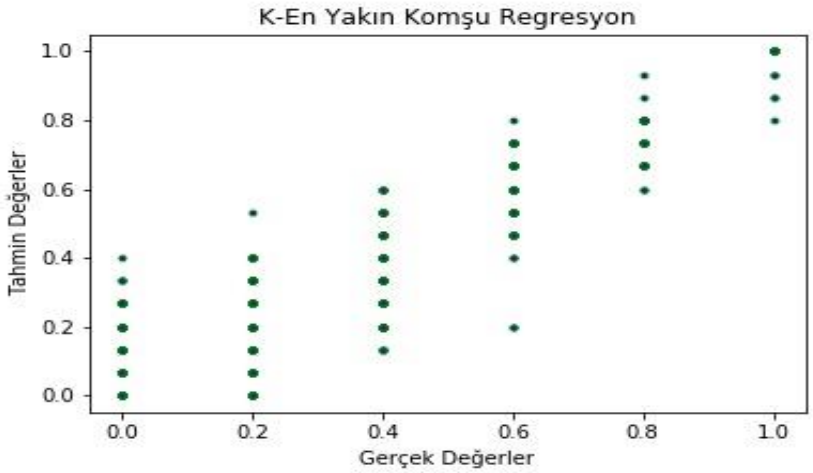


Şekil 4.6. Karar ağacı regresyonuna ait değerler

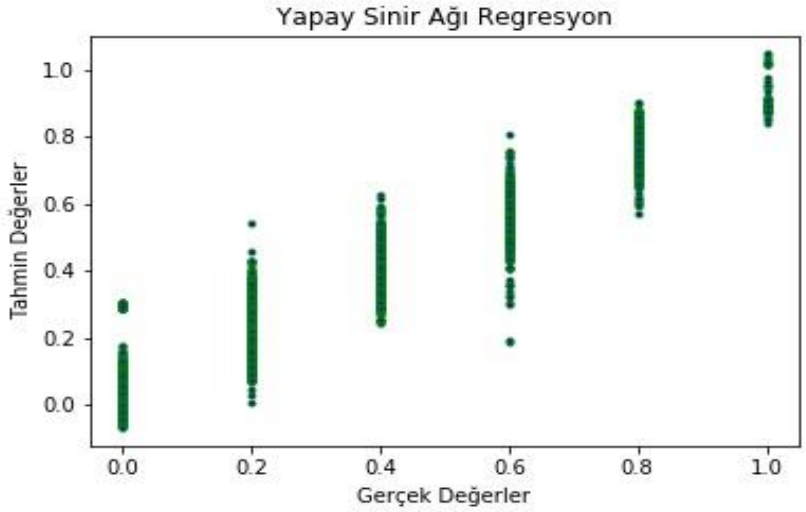


Şekil 4.7. Doğrusal regresyonuna ait değerler

Grafikler incelendiğinde veri setindeki gerçek hava kalite indeks değerleri ve regresyon algoritmalarının tahmin ettiği indeks değerlerin dağılımı görülmektedir. Örneğin gerçek değeri 0.6 olan indekslerin tahmin değerlerinin 0.6 ve etrafında dağılmış olduğu görülmektedir.

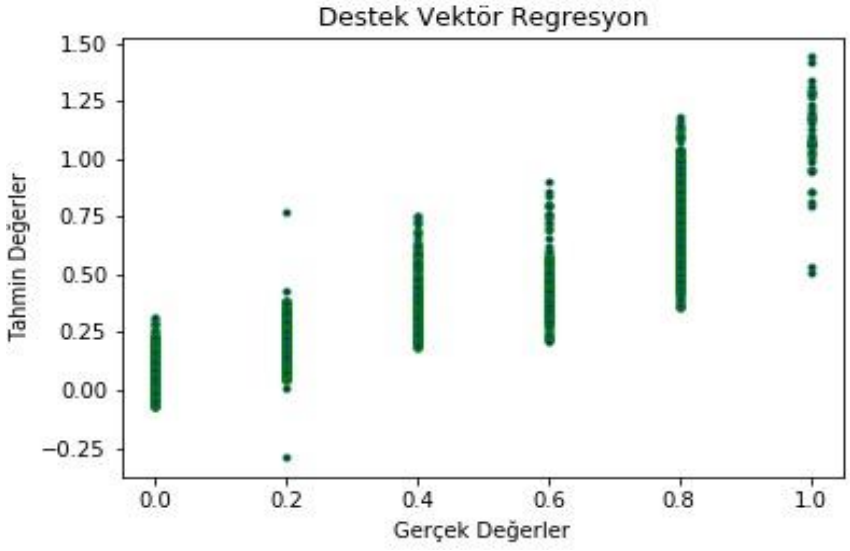


Şekil 4.8. K-en yakın komşu regresyonuna ait değerler

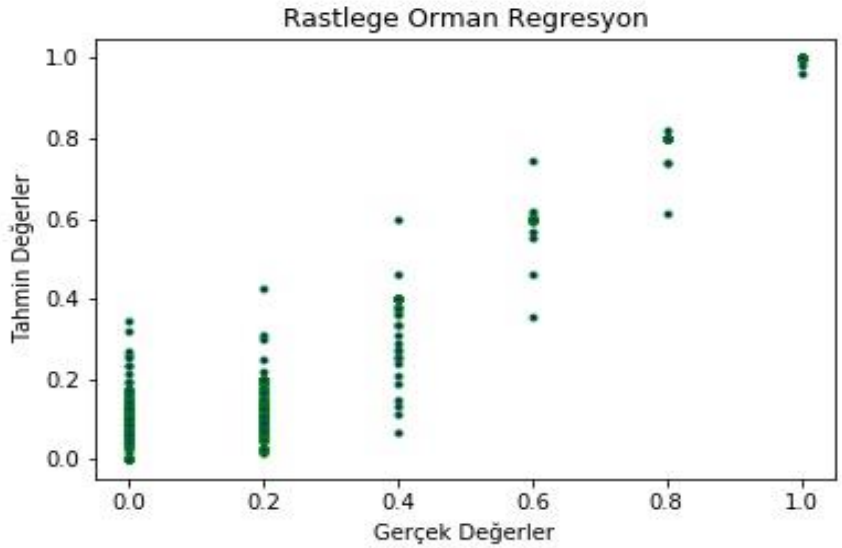


Şekil 4.9. Yapay sinir ağı regresyonuna ait değerler

Gerçek hava kalite indeks değerlerine karşılık algoritmaların tahmin ettikleri indeks değerinin dağılım göstermesi algoritmaların ürettiği sonuçlardaki hataları göstermektedir.

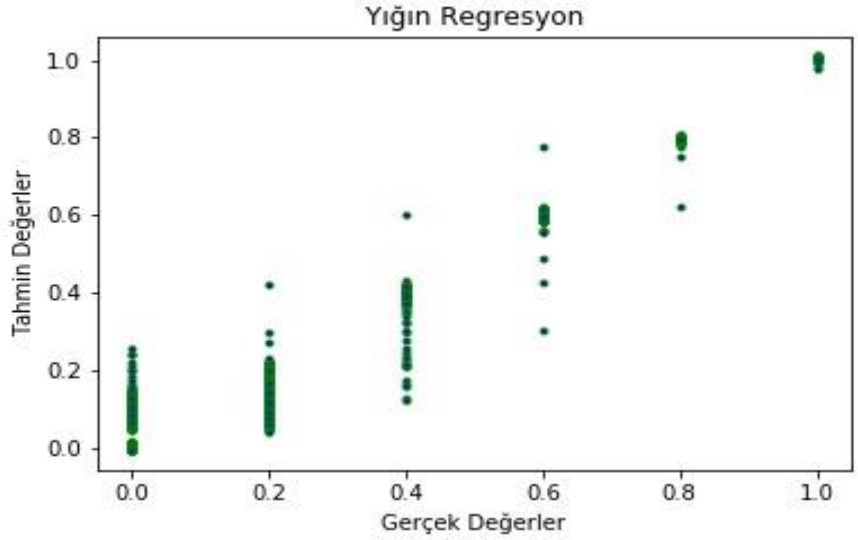


Şekil 4.10. Destek vektör regresyonuna ait değerler

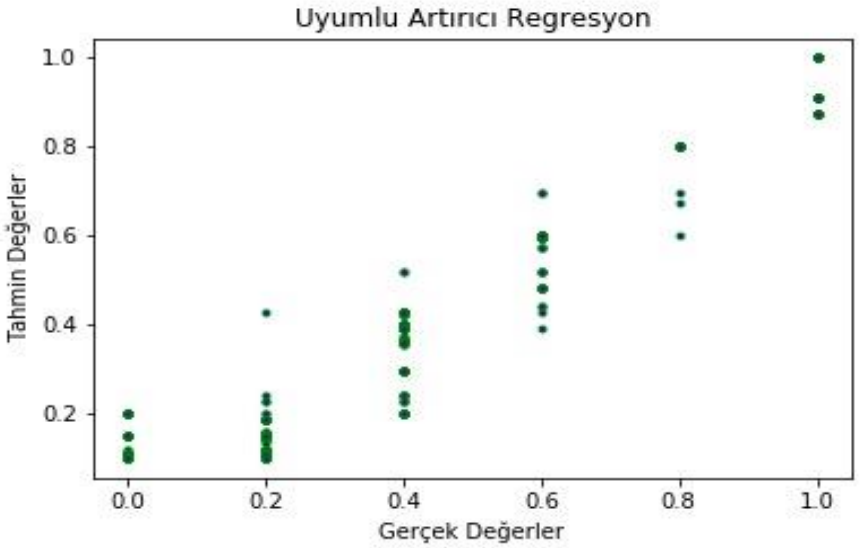


Şekil 4.11. Rastgele orman regresyonuna ait değerler

Algoritmalar tarafından üretilmiş olan hava kalite indeksi tahmin değerleri gerçek değerden daha uzak noktalara dağılmışsa bu algoritmanın daha hatalı sonuçlar ürettiği anlamına gelmektedir.

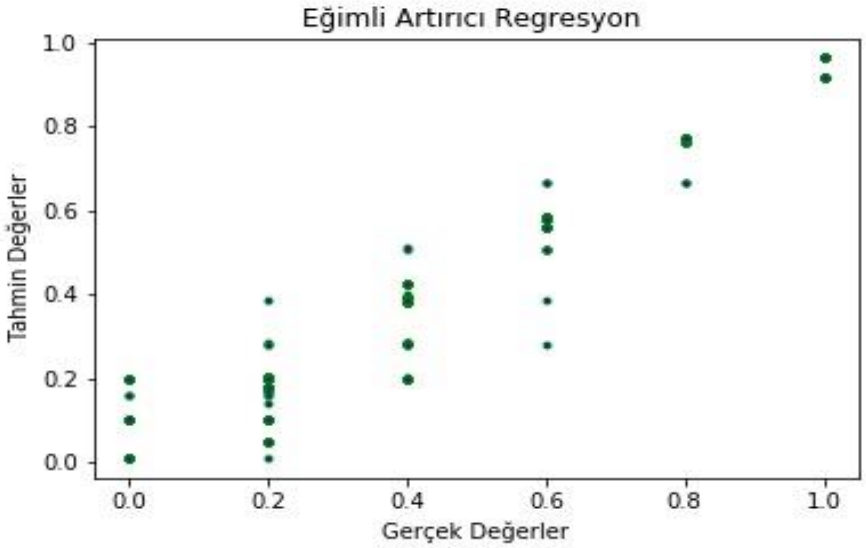


Şekil 4.12. Yığın regresyonuna ait değerler

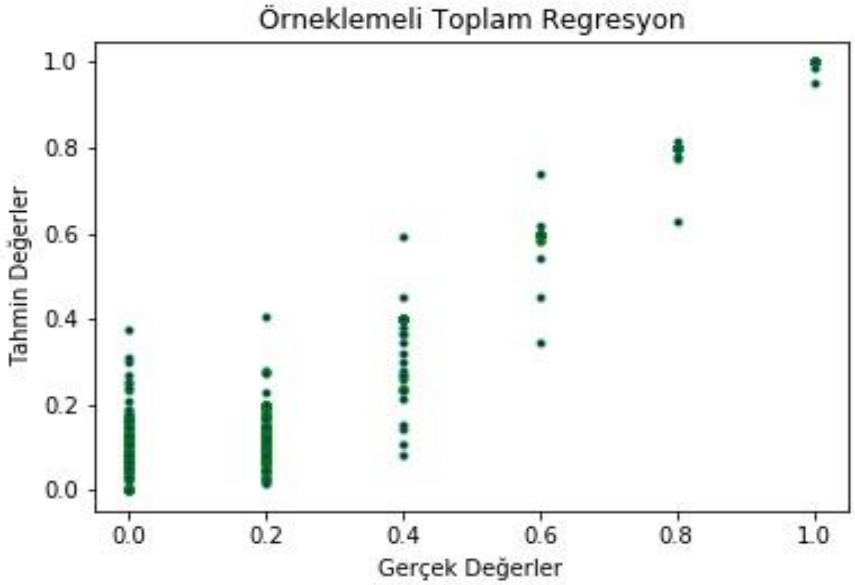


Şekil 4.13. Uyumlu artırıcı regresyonuna ait değerler

Regresyon algoritmalarına ait grafikler incelendiğinde gerçek hava kalite indeks değerlerine karşılık tahmin edilen indeks değerlerinin farklı dağılımlar gösterdiği görülmektedir. Bu algoritmaların yapısal farklılıklarından kaynaklanmaktadır.



Şekil 4.14. Eğimli artırıcı regresyonuna ait değerler



Şekil 4.15. Örneklemeli toplam regresyonuna ait değerler

5. SONUÇLAR ve ÖNERİLER

5.1. Sonuçlar

Belirlilik katsayısı (r^2) ölçütü ele alındığında hava kalite indeksinin tahmin edilmesinde topluluk regresyon algoritmalarının daha iyi sonuçlar ürettikleri görülmektedir. Başarı değeri en düşük olan algoritma doğrusal regresyon olmuştur. Belirlilik katsayısı (r^2) değeri yükseldikçe ortalama mutlak hata (omh) ve ortalama karesel hata (okh) değerlerinin düştüğü görülmektedir. Hesaplama sürelerinin destek vektör regresyonu ve yığın regresyonu hariç oldukça hızlı oldukları gözlenmektir. Hava kalite indeksinin tahmin edilmesinde başarı ölçütleri incelendiğinde rastgele orman regresyonun en iyi algoritma olduğu görülmektedir (Irmak ve Aydilek, 2019).

Regresyon algoritmalarının gerek hata değerleri gerekse hesaplama süreleri incelendiğinde gerçek zamanlı akıllı sistemlerde kullanılmaya uygun oldukları düşünülmektedir.

2. kısımda açıklanan önceki çalışmalardan farklı olarak hazır bir platform kullanılmamış ve özel olarak bir sistem yazılımı geliştirilmiştir. Bu ise gerek nesnelerin interneti bileşeni tarafında gerekse sistem tarafında daha özgürce değişiklikler yapma imkânı sağlamıştır. Ayrıca veri iletişiminde alışılagelmişin dışında smtp protokolü kullanılmıştır. Bu ise veri iletişimindeki güvenliğin artırılması ve veri kaybolma riskinin azaltılması yönlerinden büyük fayda sağlamıştır.

5.2. Öneriler

Çalışmada hava kalite ölçüm verileri kullanılmıştır. Bunun dışındaki veri türleri, farklı veri formatları ve veri boyutları kullanılabilir. Çalışmada saatlik veri gönderim periyodu kullanılmıştır. Farklı veri gönderme periyotlarıyla sistemin işleyişi incelenebilir.

Makine öğrenmesi algoritmalarından regresyon algoritmaları kullanılmıştır. Farklı olarak sınıflandırma ve kümeleme algoritmaları kullanılabilir.

Veri gönderilmesinde tek bir nesnelerin interneti bileşeni kullanılmıştır. Birden fazla sayıda ve farklı türlerde bileşen kullanılabilir. Ayrıca birden fazla sayıda sensör verisi kullanılarak farklı türlerdeki algoritmalar çalıştırılabilir.

KAYNAKLAR

AKBAR, A., KHAN, A., CARREZ, F., MOESSNER, K., 2017. Predictive Analytics for Complex IoT Data Streams. IEEE Internet of Things Journal, 4(5): 1571-1582.

AL-FUQAHA, ALA., GUIZANI, M., MOHAMMADI, M., ALEHADRI, M., AYYASH, M., 2015. Internet of Things: A Survey on Enabling Technologies, Protocols and Applications. IEEE Communications Surveys and Tutorials, 17(4): 2347-2376.

ALKHATIB, K., NAJADAT, H., HMEIDI, I., SHATNAWI, M. K. A., 2013. Stock Price Prediction Using K-Nearest Neighbor (k-NN) Algorithm. Int. J. Bus. Humanit. Technol., 3(3): 32–44.

ATEŞ, S. B., 2018. Akıllı Şebekeler Uygulamalarına Yönelik Nesnelerin İnterneti Tabanlı (IoT) Prototip Transformatör Merkezi Tasarımı ve Uygulaması. Necmettin Erbakan Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, Konya, 95s.

BOZUKLU, M., 2016. Çevresel Veriler ile Gerçek Zamanlı Nesnelerin İnterneti Uygulaması. Gaziosmanpaşa Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, Tokat, 132s.

BREIMAN, L., FRIEDMAN, J.H., OLSHEN, R.A., STONE, C.I., 1984. Classification and regression trees. Belmont, Calif.: Wadsworth, 368s.

BREIMAN, L. 1996. Stacked regressions. Machine learning, 24(1): 49-64.

BREIMAN, L., 1996. Bagging predictors. Machine Learning, 24(2): 123-140.

BREIMAN, L., 2001. Random forests. Machine Learning, 45(1): 5-32.

ÇAĞLAR, K., 2018. Nesnelerin İnterneti ile Akıllı Sera Uygulaması. Eskişehir Osmangazi Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, Eskişehir, 78s.

ENGINEERSGARAGE. (Erişim Tarihi: Nisan 2018). IOT Standards and Protocols: IOT Part 3. url: https://www.engineersgarage.com/Articles/IOT-Architecture-Standards-Protocols

EPA. (Erişim Tarihi: Nisan 2018). Air Quality Index. url: https://www3.epa.gov/airnow/aqi_brochure_02_14.pdf

ETSI. (Erişim Tarihi: Nisan 2018). European Telecommunications Standards Institute. url: https://www.etsi.org

FREIDMAN, J. H., 1999. Greedy Function Approximation: A Gradient Boosting Machine. The Annals of Statistics, 29(5): 1189-1232.

FREUND, Y., R. SCHAPIRE, R., 1997. A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting. Journal of Computer and System Sciences, 55(1): 119-139.

GÜLAÇAR, H., 2018. Nesnelerin İnterneti Platformları için Makine Öğrenmesi Tabanlı Bir Tahmin modülü. İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, İstanbul, 53s.

HAVAIZLEME. (Erişim Tarihi: Nisan 2018). Ulusal Hava Kalite İzleme Ağı. url: https://www.havaizleme.gov.tr

HROMIC, H., PHUOC, D. L., SERRANO, M., ANTONIC, A., ZARKO, I. P., HAYES, C., DECKER, S., 2015. Real Time Analysis of Sensor Data for the

Internet of Things by means of Clustering and Event Processing. IEEE International Conference. on Communications, 8-12 June, London, s.685-691.
IEEE. (Erişim Tarihi: Nisan 2018). Institute of Electrical and Electronics Engineers. url: https://www.ieee.org
IRMAK, M. E., AYDILEK I. B., (Basım Aşamasında), 2019. Hava Kalite İndeksinin Tahmin Başarısının Artırılması için Topluluk Regresyon Algoritmalarının Kullanılması. Academic Platform Journal of Engineering and Science
ITU. (Erişim Tarihi: Nisan 2018). International Telecommunications Union. url: https://www.itu.int
JAMIL, H., 2018. Real Time Stream Processing for Internet of Things. Sakarya Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, Sakarya, 23s.
KANAWADAY, A., SANE, A., 2017. Machine Learning for Predictive Maintenance of Industrial Machines using IoT Sensor Data. 8th IEEE International Conference on Software Engineering and Service Science (ICSESS), 24-26 November, Beijing China, s.87-90.
MURTAGH, F.,1991. Multilayer Perceptron for Classification and Regression. Neurocomputing, 2(5-6): 183-197.
ONAL, A. C., SEZER, O. B., OZBAYOGLU, M., DOGDU, E., 2017. Weather Data Analysis and Sensor Fault Detection Using an Extended IoT Framework with Semantics, Big Data, and Machine Learning. IEEE International Conference on Big Data, 11-14 December, Boston, MA, s.2037-2046.
PAXTECHNICA. (Erişim Tarihi: Nisan 2018). IOT Timeline. url: http://paxtechnica.org/?page_id=16
POSTSCAPES. (Erişim Tarihi: Nisan 2018) Internet of Things (IoT) History. url: https://www.postscapes.com/internet-of-things-history/
QUINLAN, J. R., 1986. Induction of Decision Trees. Machine Learning, 1(1): 81-106.
SCIKIT-LEARN. (Erişim Tarihi: Nisan 2018). Machine Learning in Python. url: https://scikit-learn.org/stable/
SMOLA, A. J., SCHÖLKOPF, B., 2004. A tutorial on support vector regression, Statistics and Computing, 14(3): 199-222.
SPYDER. (Erişim Tarihi: Nisan 2018). The Scientific Python Development Environment. url: https://www.spyder-ide.org
STANTON, J. M., 2001. Galton, Pearson, and the Peas: A Brief History of Linear Regression for Statistics Instructors. Journal of Statistics Education, 9(3): 1-13.
TÜMER, E. D., 2018. Nesnelerin İnterneti Ortamındaki Veriler Kullanılarak Makine Öğrenmesi Algoritmaları ile Tahminleme ve Gerçek Zamanlı Veriler Üzerinde Karmaşık Olay İşleme. Ege Üniversitesi, Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, İzmir, 62s.
VAPNIK, V., 2000. The nature of statistical learning theory. Springer-Verlag, New York, 122s.
WUSTL EDU. (Erişim Tarihi: Nisan 2018). Internet of Things Protocols and Standards. url:https://www.cse.wustl.edu/~jain/cse57015/ftp/iot_prot/index.html
YAZICI, M. T., BASURRA, S., GABER, M. M., 2018. Edge Machine Learning: Enabling Smart Internet of Things Applications. Big Data and Cognitive Computing, 2(3): 26-43.

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı : Muhammet Emre Irmak

Uyruğu : Türkiye Cumhuriyeti

Doğum Tarihi ve Yeri : 1990 – Seyhan /ADANA

Telefon : +90506 927 30 26

e-mail : memreirmak@hotmail.com

EĞİTİM

Derece	Adı, İlçe, İl	Bitirme Yılı
Lise	Seyhan Çukurova Lisesi, Adana	2007
Üniversite	İskenderun Teknik Üniversitesi, Hatay	2016
Yüksek Lisans	Harran Üniversitesi, Şanlıurfa	2019

UZMANLIK ALANI

Gömülü Sistemler, Kablosuz Haberleşme Sistemleri, Yapay Zeka

YABANCI DİLLER

İngilizce YDS: 48.75 (2017 İlkbahar Dönemi)

YAYINLAR

IRMAK, M. E., AYDILEK I. B., (Basım Aşamasında), 2019. Hava Kalite İndeksinin Tahmin Başarısının Artırılması için Topluluk Regresyon Algoritmalarının Kullanılması. Academic Platform Journal of Engineering and Science

EKLER

EK 1. Nesnelerin İnterneti Bileşeni Kaynak Kodları

```
#include <ESP8266WiFi.h>
#include "Gsender.h"
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define OLED_RESET LED_BUILTIN  //4
Adafruit_SSD1306 display(OLED_RESET);

#if (SSD1306_LCDHEIGHT != 64)
#error("Height incorrect, please fix Adafruit_SSD1306.h!");
#endif
const int buttonPin = 14;
const int ledPin =  0;
int buttonState = 0;
String mesaj;
int PM10;
int SO2;

int NO2;

int O3;
int CO;
int sayac = 1;
int x = 43848;
int saat =0;
String kod="SA-1B2";
String tarih="1.01.2018 0";

#pragma region Globals
//const char* ssid = "ADA-1A";                        // WIFI network name
//const char* password = "PASSWORD";                  // WIFI network password
uint8_t connection_state = 0;                    // Connected to WIFI or not
uint16_t reconnect_interval = 10000;             // If not connected wait time to try again
#pragma endregion Globals

uint8_t WiFiConnect(const char* nSSID = nullptr, const char* nPassword = nullptr)
{
   static uint16_t attempt = 0;
   Serial.print("Connecting to ");
   if(nSSID) {
      WiFi.begin(nSSID, nPassword);
```

```
      Serial.println(nSSID);
    } else {
      WiFi.begin(ssid, password);
      Serial.println(ssid);
    }

    uint8_t i = 0;
    while(WiFi.status()!= WL_CONNECTED && i++ < 50)
    {
      delay(200);
      Serial.print(".");
    }
    ++attempt;
    Serial.println("");
    if(i == 51) {
      Serial.print("Connection: TIMEOUT on attempt: ");
      Serial.println(attempt);
      if(attempt % 2 == 0)
        Serial.println("Check if access point available or SSID and Password\r\n");
      return false;
    }
    Serial.println("Connection: ESTABLISHED");
    Serial.print("Got IP address: ");
    Serial.println(WiFi.localIP());
    return true;
}

void Awaits()
{
    uint32_t ts = millis();
    while(!connection_state)
    {
      delay(50);
      if(millis() > (ts + reconnect_interval) && !connection_state){
        connection_state = WiFiConnect();
        ts = millis();
      }
    }
}

void setup()
{
   pinMode(ledPin, OUTPUT);
    pinMode(buttonPin, INPUT);
    digitalWrite(ledPin, LOW);
   display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
   display.setTextSize(3);
   display.setTextColor(WHITE);
```

```
   Serial.begin(115200);
}

void loop()
 {

digitalWrite(ledPin, LOW);
  while(sayac == 1){
  display.setCursor(30,0);
  display.print("PM10");
  display.setCursor(30,40);
  PM10 = analogRead(A0);
  display.println(PM10);
  display.display();
  display.clearDisplay();
  delay(30);
  buttonState = digitalRead(buttonPin);
  if (buttonState == HIGH){
    digitalWrite(ledPin, HIGH);
     delay(500);
    digitalWrite(ledPin, LOW);
  sayac++;
  delay(1000);
  break;}
  }
  while(sayac ==2){
  display.setCursor(30,0);
  display.print("SO2");
  display.setCursor(30,40);
  SO2 = analogRead(A0);
  display.println(SO2);
  display.display();
  display.clearDisplay();
  delay(30);
   buttonState = digitalRead(buttonPin);
  if (buttonState == HIGH){

    digitalWrite(ledPin, HIGH);
     delay(500);
    digitalWrite(ledPin, LOW);
  sayac++;
  delay(1000);
  break;}
  }
  while(sayac ==3){
  display.setCursor(30,0);
  display.print("NO2");
  display.setCursor(30,40);
```

```
NO2 = analogRead(A0);
display.println(NO2);
display.display();
display.clearDisplay();
delay(30);
 buttonState = digitalRead(buttonPin);
if (buttonState == HIGH){

  digitalWrite(ledPin, HIGH);
   delay(500);
  digitalWrite(ledPin, LOW);
sayac++;
delay(1000);
break;}
}
while(sayac ==4){
display.setCursor(30,0);
display.print("O3");
display.setCursor(30,40);
O3 = analogRead(A0);
display.println(O3);
display.display();
display.clearDisplay();
delay(30);
 buttonState = digitalRead(buttonPin);
if (buttonState == HIGH){

  digitalWrite(ledPin, HIGH);
   delay(500);
  digitalWrite(ledPin, LOW);
sayac++;
delay(1000);
break;}
}
while(sayac ==5){
display.setCursor(30,0);
display.print("CO");
display.setCursor(30,40);
CO = analogRead(A0);
display.println(CO);
display.display();
display.clearDisplay();
delay(30);
 buttonState = digitalRead(buttonPin);
if (buttonState == HIGH){

  digitalWrite(ledPin, HIGH);
   delay(500);
```

```
  digitalWrite(ledPin, LOW);
 sayac++;
 delay(1000);
 break;}
 }
 while(sayac ==6){

 digitalWrite(ledPin, HIGH);
   delay(500);
  digitalWrite(ledPin, LOW);
 mesaj=kod+" "+x+" "+tarih+saat+":00 "+PM10+" "+SO2+" 0 "+NO2+" 0 "+O3+"
"+CO+"\n";
 x++;
 saat++;
  Serial.print(mesaj);
  connection_state = WiFiConnect();
  if(!connection_state)  // if not connected to WIFI
     Awaits();          // constantly trying to connect

  Gsender *gsender = Gsender::Instance();    // Getting pointer to class instance


 String subject = "ESP8266";

 if (gsender->Subject(subject)->Send("ada1a.project@gmail.com", mesaj)) {
     Serial.println("Message send.");
  } else {
     Serial.print("Error sending message: ");
     Serial.println(gsender->getError());
  }

  display.setCursor(0,0);
  display.println("GONDERILDI");
   display.display();
  display.clearDisplay();


    if (sayac=6){


 sayac=1;
 delay(1000);
 break;}


 }

 }
```

EK 2. Gerçek Zamanlı İşlemler için Geliştirilen Program Kaynak Kodları

```
# -*- coding: utf-8 -*-
"""
Created on Sat Jul 14 11:25:00 2018

@author: memreirmak
"""
import time
import email
import imaplib
import smtplib
from itertools import chain
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
from email.mime.base import MIMEBase
from email import encoders
import os
import pandas as pd
import datetime as dt
import numpy as np
import matplotlib.pylab as plt


from sklearn.preprocessing import MinMaxScaler
from sklearn.cross_validation import train_test_split


from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor

from sklearn.linear_model import LinearRegression
from sklearn.neural_network import MLPRegressor
from mlxtend.regressor import StackingRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import BaggingRegressor

from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score



gmail_user = "ada1a.project@gmail.com"
```

```
gmail_pwd = "parola"

imap_ssl_host = 'imap.gmail.com'  # imap.mail.yahoo.com
imap_ssl_port = 993
username = 'ada1a.project@gmail.com'
password = 'parola'

# Restrict mail search. Be very specific.
# Machine should be very selective to receive messages.
criteria = {
    'FROM':    'ada1a.project@gmail.com',
    'SUBJECT': 'ESP8266',
    #'BODY':    'body',
}
uid_max = 0
mesaj= "SA-1B2"
mesaj2="SA-1D1"

def search_string(uid_max, criteria):
    c = list(map(lambda t: (t[0], '"'+str(t[1])+'"'), criteria.items())) + [('UID', '%d:*' %
(uid_max+1))]
    return '(%s)' % ' '.join(chain(*c))
    # Produce search string in IMAP format:
    #   e.g. (FROM "me@gmail.com" SUBJECT "abcde" BODY "123456789" UID
9999:*)


def get_first_text_block(msg):
    type = msg.get_content_maintype()

    if type == 'multipart':
        for part in msg.get_payload():
            if part.get_content_maintype() == 'text':
                return part.get_payload()
    elif type == 'text':
        return msg.get_payload()

server = imaplib.IMAP4_SSL(imap_ssl_host, imap_ssl_port)
server.login(username, password)
server.select('INBOX')

result, data = server.uid('search', None, search_string(uid_max, criteria))

uids = [int(s) for s in data[0].split()]
print (uids)
if uids:
    uid_max = max(uids)
```

```
    # Initialize `uid_max`. Any UID less than or equal to `uid_max` will be ignored
subsequently.
    print (uid_max)
    type(uid_max)

with open('uid.txt', 'r') as f:
    rd=f.readline()
    rd=int(rd)
    print (rd)
for uid in uids :
     if uid>rd:
       print(uid)

       result, data = server.uid('fetch',str(uid),'(RFC822)')  # fetch entire message
       msg = email.message_from_bytes(data[0][1])
       text = get_first_text_block(msg)

       print ('GELEN MESAJ ')
       print (text)

       if mesaj in text:
         with open('sensorveri.txt', 'a') as f:
          f.write(text)
         print ('veri yazildi')
       if mesaj2 in text:
         myfile = open('deneme2.txt', 'a')
         myfile.write(text)
         print  ('veri yazildi')
         myfile.close()
with open('uid.txt', 'w') as f:
 f.write('%d' % uid_max)
print ('uid_max yazildi')
server.logout()


def mailgonder(to, subject, text, attach):
    msg = MIMEMultipart()

    msg['From'] = gmail_user
    msg['To'] = to
    msg['Subject'] = subject

    msg.attach(MIMEText(text))

    part = MIMEBase('application', 'octet-stream')
    part.set_payload(open(attach, 'rb').read())
    encoders.encode_base64(part)
```

```
    part.add_header('Content-Disposition', "attachment; filename=%s" %
os.path.basename(attach))
    msg.attach(part)

    mailServer = smtplib.SMTP("smtp.gmail.com", 587)
    mailServer.ehlo()
    mailServer.starttls()
    mailServer.ehlo()
    mailServer.login(gmail_user, gmail_pwd)
    mailServer.sendmail(gmail_user, to, msg.as_string())
    print('mail gönderildi')
    # Should be mailServer.quit(), but that crashes...
    mailServer.close()



while True:
    try:

        print ('--------------------')
        server = imaplib.IMAP4_SSL(imap_ssl_host, imap_ssl_port)
        server.login(username, password)
        server.select('INBOX')

        result, data = server.uid('search', None, search_string(uid_max, criteria))

        uids = [int(s) for s in data[0].split()]
        for uid in uids:
            # Have to check again because Gmail sometimes does not obey UID
criterion.
            if uid > uid_max:
                result, data = server.uid('fetch',str(uid),'(RFC822)')  # fetch entire
message
                print (uid)
                msg = email.message_from_bytes(data[0][1])
                uid_max = uid
                with open('uid.txt', 'w') as f:
                    f.write('%d' % uid_max)
                print ('uid_max yazildi')
                text = get_first_text_block(msg)
                print ('GELEN MESAJ ')
                print (text)
                if mesaj in text:
                    with open('sensorveri.txt', 'a') as f:
                      f.write(text)
                    print ('veri yazildi')
                if mesaj2 in text:
```

```
                myfile = open('deneme2.txt', 'a')
                myfile.write(text)
                print ('veri yazildi')
                myfile.close()
```

```
        df =
pd.read_csv(r'C:\Users\hp\Desktop\hkiveri.txt',sep=',',index_col=0,parse_dates=['Tar
ih_Zaman'],dayfirst=False)
        dfo = pd.read_csv(r'C:\Users\hp\Desktop\sensorveri.txt',sep='
',index_col='index',decimal='','',parse_dates=[['Tarih', 'Zaman']],dayfirst=False)
        dfi0 =
pd.read_csv(r'C:\Users\hp\Desktop\dfi.txt',sep=',',index_col=0,parse_dates=['Tarih_S
aat'],dayfirst=False)


        del dfo['NO']
        del dfo['NOX']
        del dfo['sensor']

        son=df.tail(1).index.item()
        df=df.append(dfo.loc[son+1],ignore_index=True)
        sonindex=son+1
        with open('index.txt', 'w') as f:
         f.write('%d' % sonindex)
         print ('son index yazildi')

        with open('index.txt', 'r') as f:
           rd=f.readline()
           #rd=int(rd)
           print (rd)



        df=df.round(0)
        df=df.fillna(df.mean())
        df=df.round(0)


        cols = ['Tarih_Saat','SO2',
'NO2','CO_8','O3_8','PM10_24','v1','v2','v3','v4','v5','HKI']
        dfi = pd.DataFrame(columns=cols, index=range(1))

        i=0
```

```
al=son+1


dfi.loc[i].Tarih_Saat =df['Tarih_Zaman'][al]
dfi.loc[i].SO2        =df['SO2'][al]
dfi.loc[i].NO2        =df['NO2'][al]
dfi.loc[i].CO_8       =df['CO'][:al+1].tail(8).mean()
dfi.loc[i].O3_8       =df['O3'][:al+1].tail(8).mean()
dfi.loc[i].PM10_24    =df['PM10'][:al+1].tail(24).mean()

v1=dfi['SO2'][i]
v2=dfi['NO2'][i]
v3=dfi['CO_8'][i]
v4=dfi['O3_8'][i]
v5=dfi['PM10_24'][i]

if v1>=0 and v1<=100:
   dfi.loc[i].v1=1
elif v1>=101 and v1<=250:
   dfi.loc[i].v1=2
elif v1>=251 and v1<=500:
   dfi.loc[i].v1=3
elif v1>=501 and v1<=850:
   dfi.loc[i].v1=4
elif v1>=851 and v1<=1100:
   dfi.loc[i].v1=5
elif v1>1001:
   dfi.loc[i].v1=6

if v2>=0 and v2<=100:
   dfi.loc[i].v2=1
elif v2>=101 and v2<=250:
   dfi.loc[i].v2=2
elif v2>=251 and v2<=500:
   dfi.loc[i].v2=3
elif v2>=501 and v2<=1000:
   dfi.loc[i].v2=4
elif v2>=1001 and v2<=2000:
   dfi.loc[i].v2=5
elif v2>2001:
   dfi.loc[i].v2=6

if v3>=0 and v3<=5500:
   dfi.loc[i].v3=1
elif v3>=5501 and v3<10000:
   dfi.loc[i].v3=2
elif v3>=10001 and v2<=16000:
   dfi.loc[i].v3=3
```

```
        elif v3>=16001 and v3<=24000:
            dfi.loc[i].v3=4
        elif v3>=24001 and v3<=22000:
            dfi.loc[i].v3=5
        elif v3>32001:
            dfi.loc[i].v3=6

        if v4>=0 and v4<121:
            dfi.loc[i].v4=1
        elif v4>=121 and v4<161:
            dfi.loc[i].v4=2
        elif v4>=161 and v4<181:
            dfi.loc[i].v4=3
        elif v4>=181 and v4<241:
            dfi.loc[i].v4=4
        elif v4>=241 and v4<701:
            dfi.loc[i].v4=5
        elif v4>701:
            dfi.loc[i].v4=6

        if v5>=0 and v5<=50:
            dfi.loc[i].v5=1
        elif v5>=51 and v5<=100:
            dfi.loc[i].v5=2
        elif v5>=101 and v5<=260:
            dfi.loc[i].v5=3
        elif v5>=261 and v5<=400:
            dfi.loc[i].v5=4
        elif v5>=401 and v5<=520:
            dfi.loc[i].v5=5
        elif v5>521:
            dfi.loc[i].v5=6



        dfi['HKI'] = dfi[['v1','v2','v3','v4','v5']].apply(max, axis=1)
        dfi= dfi.convert_objects(convert_numeric=True)
        dfi=dfi.round(0)

        dfi0=dfi0.append(dfi.loc[i],ignore_index=True)
        df24=df.tail(24)

        df24.plot(x="Tarih_Zaman", y=["PM10", "SO2", "NO2","O3","CO"],
linestyle='-', marker='o' )
        plt.savefig(r"C:\Users\hp\Desktop\sonuclar\image.jpg")
        plt.clf()
        plt.cla()
        plt.close()
```

```
        hki=dfi['HKI'][i]
        if hki==1:
           mailgonder("memreirmak@hotmail.com", "Hava Kalitesi", "Adana Valilik
İstasyonu (İYİ  1)", r"C:\Users\hp\Desktop\sonuclar\image.jpg")
        elif hki==2:
            mailgonder("memreirmak@hotmail.com", "Hava Kalitesi", "Adana Valilik
İstasyonu (ORTA  2)", r"C:\Users\hp\Desktop\sonuclar\image.jpg")
        elif hki==3:
            mailgonder("memreirmak@hotmail.com", "Hava Kalitesi", "Adana Valilik
İstasyonu (HASSAS  3)", r"C:\Users\hp\Desktop\sonuclar\image.jpg")
        elif hki==4:
            mailgonder("memreirmak@hotmail.com", "Hava Kalitesi", "Adana Valilik
İstasyonu (SAĞLIKSIZ  4)", r"C:\Users\hp\Desktop\sonuclar\image.jpg")
        elif hki==5:
            mailgonder("memreirmak@hotmail.com", "Hava Kalitesi", "Adana Valilik
İstasyonu (KÖTÜ  5)", r"C:\Users\hp\Desktop\sonuclar\image.jpg")
        elif hki==6:
            mailgonder("memreirmak@hotmail.com", "Hava Kalitesi", "Adana Valilik
İstasyonu (TEHLİKELİ 6)", r"C:\Users\hp\Desktop\sonuclar\image.jpg")




        df.to_csv('hkiveri.txt',sep=',')
        dfi0.to_csv('dfi.txt',sep=',')


        X = pd.read_csv(r'C:\Users\hp\Desktop\dfi.txt', usecols=['SO2', 'NO2', 'CO_8',
'O3_8', 'PM10_24'])

        Y = pd.read_csv(r'C:\Users\hp\Desktop\dfi.txt', usecols=['HKI'])






        scaling = MinMaxScaler(feature_range=(0,1)).fit(X)
        X = scaling.transform(X)
        xt=X[-1:]
        scaling = MinMaxScaler(feature_range=(0,1)).fit(Y)
        Y = scaling.transform(Y)

        X1, X2, Y1, Y2 = train_test_split(X, Y, test_size=0.25, random_state=42)
```

```
start = time.time()
regr_1 = DecisionTreeRegressor(max_depth=5)
regr_1.fit(X1, Y1)
y_1 = regr_1.predict(X2)
yt1 = regr_1.predict(xt)
end = time.time()
t1=end - start
yt1=np.round(yt1,1)
plt.xlabel('Gerçek Değerler')
plt.ylabel('Tahmin Değerler')
plt.title('Karar Ağacı Regresyon')
plt.plot(Y2, y_1, color='green', linestyle=' ', marker='.',markerfacecolor='blue')
plt.savefig(r"C:\Users\hp\Desktop\sonuclar\DTR.jpg")
plt.show()

start = time.time()
abc = RandomForestRegressor(n_estimators=30)
abc.fit(X1, Y1)
y_2 = abc.predict(X2)
yt2 = abc.predict(xt)
end = time.time()
t2=end - start
yt2=np.round(yt2,1)
plt.xlabel('Gerçek Değerler')
plt.ylabel('Tahmin Değerler')
plt.title('Rastlege Orman Regresyon')
plt.plot(Y2, y_2, color='green', linestyle=' ', marker='.',markerfacecolor='blue')
plt.savefig(r"C:\Users\hp\Desktop\sonuclar\RFR.jpg")
plt.show()

start = time.time()
abc2 = SVR(C=5.0, epsilon=0.1)
abc2.fit(X1, Y1)
y_3 = abc2.predict(X2)
yt3 = abc2.predict(xt)
end = time.time()
t3=end - start
yt3=np.round(yt3,1)
plt.xlabel('Gerçek Değerler')
plt.ylabel('Tahmin Değerler')
plt.title('Destek Vektör Regresyon')
plt.plot(Y2, y_3, color='green', linestyle=' ', marker='.',markerfacecolor='blue')
plt.savefig(r"C:\Users\hp\Desktop\sonuclar\SVR.jpg")
plt.show()

start = time.time()
```

```
knn = KNeighborsRegressor(n_neighbors=3, algorithm='auto', leaf_size=30,
weights='uniform')
knn.fit(X1, Y1)
y_4 = knn.predict(X2)
yt4 = knn.predict(xt)
end = time.time()
t4=end - start
yt4=np.round(yt4,1)
plt.xlabel('Gerçek Değerler')
plt.ylabel('Tahmin Değerler')
plt.title('K-En Yakın Komşu Regresyon')
plt.plot(Y2, y_4, color='green', linestyle=' ', marker='.',markerfacecolor='blue')

plt.savefig(r"C:\Users\hp\Desktop\sonuclar\kNNR.jpg")
plt.show()

start = time.time()
lin = LinearRegression()
lin.fit(X1,Y1)
y_5 = lin.predict(X2)
yt5 = lin.predict(xt)
end = time.time()
t5=end - start
yt5=np.round(yt5,1)
plt.xlabel('Gerçek Değerler')
plt.ylabel('Tahmin Değerler')
plt.title('Doğrusal Regresyon')
plt.plot(Y2, y_5, color='green', linestyle=' ', marker='.',markerfacecolor='blue')
plt.savefig(r"C:\Users\hp\Desktop\sonuclar\LR.jpg")
plt.show()

start = time.time()
mlp = MLPRegressor()
mlp.fit(X1, Y1)
y_6 = mlp.predict(X2)
yt6 = mlp.predict(xt)
end = time.time()
t6=end - start
yt6=np.round(yt6,1)
plt.xlabel('Gerçek Değerler')
plt.ylabel('Tahmin Değerler')
plt.title('Yapay Sinir Ağı Regresyon')
plt.plot(Y2, y_6, color='green', linestyle=' ', marker='.',markerfacecolor='blue')
plt.savefig(r"C:\Users\hp\Desktop\sonuclar\NNR.jpg")
plt.show()


start = time.time()
```

```
stregr =
StackingRegressor(regressors=[abc,lin,abc2,knn,regr_1],meta_regressor=mlp)
stregr.fit(X1,Y1)
y_7=stregr.predict(X2)
yt7=stregr.predict(xt)
end = time.time()
t7=end - start
yt7=np.round(yt7,1)
plt.xlabel('Gerçek Değerler')
plt.ylabel('Tahmin Değerler')
plt.title('Yığın Regresyon')
plt.plot(Y2, y_7, color='green', linestyle=' ', marker='.',markerfacecolor='blue')
plt.savefig(r"C:\Users\hp\Desktop\sonuclar\stacked.jpg")
plt.show()



start = time.time()
adab= AdaBoostRegressor(n_estimators=30)
adab.fit(X1, Y1)
y_8= adab.predict(X2)
yt8= adab.predict(xt)
end = time.time()
t8=end - start
yt8=np.round(yt8,1)
plt.xlabel('Gerçek Değerler')
plt.ylabel('Tahmin Değerler')
plt.title('Uyumlu Artırıcı Regresyon')
plt.plot(Y2, y_8, color='green', linestyle=' ', marker='.',markerfacecolor='blue')
plt.savefig(r"C:\Users\hp\Desktop\sonuclar\ADABOOST.jpg")
plt.show()


start = time.time()
gbrt=GradientBoostingRegressor(n_estimators=30)
gbrt.fit(X1, Y1)
y_9= gbrt.predict(X2)
yt9= gbrt.predict(xt)
end = time.time()
t9=end - start
yt9=np.round(yt9,1)
plt.xlabel('Gerçek Değerler')
plt.ylabel('Tahmin Değerler')
plt.title('Eğimli Artırıcı Regresyon ')
plt.plot(Y2, y_9, color='green', linestyle=' ', marker='.',markerfacecolor='blue')
plt.savefig(r"C:\Users\hp\Desktop\sonuclar\grdBOOST.jpg")
plt.show()
```

```
start = time.time()
br=BaggingRegressor(n_estimators = 30)
br.fit(X1, Y1)
y_10= br.predict(X2)
yt10= br.predict(xt)
end = time.time()
t10=end - start
yt10=np.round(yt10,1)
plt.xlabel('Gerçek Değerler')
plt.ylabel('Tahmin Değerler')
plt.title('Örneklemeli Toplam Regresyon')
plt.plot(Y2, y_10, color='green', linestyle=' ', marker='.',markerfacecolor='blue')
plt.savefig(r"C:\Users\hp\Desktop\sonuclar\Bagging.jpg")
plt.show()



p1= r2_score(Y2, y_1)
p2= mean_absolute_error(Y2, y_1)
p3= mean_squared_error(Y2, y_1)
p1=p1.round(5)
p2=p2.round(5)
p3=p3.round(5)
print ('DecisionTreeRegressor\n')
print (p1)
print (p2)
print (p3)
print ('\n')
print (t1)
print ('\n')

p4= r2_score(Y2, y_2)
p5= mean_absolute_error(Y2, y_2)
p6= mean_squared_error(Y2, y_2)
p4=p4.round(5)
p5=p5.round(5)
p6=p6.round(5)

print ('RandomForestRegressor\n')
print (p4)
print (p5)
print (p6)
print ('\n')
print (t2)
print ('\n')
```

```
p7= r2_score(Y2, y_3)
p8= mean_absolute_error(Y2, y_3)
p9= mean_squared_error(Y2, y_3)
p7=p7.round(5)
p8=p8.round(5)
p9=p9.round(5)
print ('SVR regressor\n')
print (p7)
print (p8)
print (p9)
print ('\n')
print (t3)
print ('\n')

p10= r2_score(Y2, y_4)
p11= mean_absolute_error(Y2, y_4)
p12= mean_squared_error(Y2, y_4)
p10=p10.round(5)
p11=p11.round(5)
p12=p12.round(5)
print ('Knn regressor\n')
print (p10)
print (p11)
print (p12)
print ('\n')
print (t4)
print ('\n')

p13= r2_score(Y2, y_5)
p14= mean_absolute_error(Y2, y_5)
p15= mean_squared_error(Y2, y_5)
p13=p13.round(5)
p14=p14.round(5)
p15=p15.round(5)
print ('Linear regressor\n')
print (p13)
print (p14)
print (p15)
print ('\n')
print (t5)
print ('\n')

p16= r2_score(Y2, y_6)
p17= mean_absolute_error(Y2, y_6)
p18= mean_squared_error(Y2, y_6)
p16=p16.round(5)
p17=p17.round(5)
```

```
p18=p18.round(5)
print ('Neural Network Regressor\n')
print (p16)
print (p17)
print (p18)
print ('\n')
print (t6)
print ('\n')


psr= r2_score(Y2, y_7)
psr_mae= mean_absolute_error(Y2, y_7)
psr_mse= mean_squared_error(Y2, y_7)
psr=psr.round(5)
psr_mae=psr_mae.round(5)
psr_mse=psr_mse.round(5)
print ('stacking Regressor\n')
print (psr)
print (psr_mae)
print (psr_mse)
print ('\n')
print (t7)
print ('\n')

p_ada= r2_score(Y2, y_8)
p_ada_mae= mean_absolute_error(Y2, y_8)
p_ada_mse= mean_squared_error(Y2, y_8)
p_ada=p_ada.round(5)
p_ada_mae=p_ada_mae.round(5)
p_ada_mse=p_ada_mse.round(5)
print ('Adaboost \n')
print (p_ada)
print (p_ada_mae)
print (p_ada_mse)
print ('\n')
print (t8)
print ('\n')

p_gbrt= r2_score(Y2, y_9)
p_gbrt_mae= mean_absolute_error(Y2, y_9)
p_gbrt_mse= mean_squared_error(Y2, y_9)
p_gbrt=p_gbrt.round(5)
p_gbrt_mae=p_gbrt_mae.round(5)
p_gbrt_mse=p_gbrt_mse.round(5)
print ('gradboost \n')
print (p_gbrt)
print (p_gbrt_mae)
print (p_gbrt_mse)
```

```
print ('\n')
print (t9)
print ('\n')

p_br= r2_score(Y2, y_10)
p_br_mae= mean_absolute_error(Y2, y_10)
p_br_mse= mean_squared_error(Y2, y_10)
p_br=p_br.round(5)
p_br_mae=p_br_mae.round(5)
p_br_mse=p_br_mse.round(5)
print ('bagging \n')
print (p_br)
print (p_br_mae)
print (p_br_mse)
print ('\n')
print (t9)
print ('\n')

r2 =  [p1,p4,p7,p10,p13,p16,psr,p_ada,p_gbrt,p_br]
mae = [p2,p5,p8,p11,p14,p17,psr_mae,p_ada_mae,p_gbrt_mae,p_br_mae]
mse = [p3,p6,p9,p12,p15,p18,psr_mse,p_ada_mse,p_gbrt_mse,p_br_mse]
süre = [t1,t2,t3,t4,t5,t6,t7,t8,t9,t10]
ytg = [yt1,yt2,yt3,yt4,yt5,yt6,yt7,yt8,yt9,yt10]
index = ['DTR', 'RFR', 'SVR','kNNR', 'LR',
'NNR','STCKR','ADBR','GRBR','BAGR']



df1 = pd.DataFrame({'r2' :  r2}, index=index)
df2 = pd.DataFrame({'mae': mae}, index=index)
df3 = pd.DataFrame({'mse': mse}, index=index)
df4 = pd.DataFrame({'süre': süre}, index=index)
df5 = pd.DataFrame({'tahmin': ytg}, index=index)
df5=df5.astype(float)
df1 = df1.sort_values(by=['r2',], ascending=False)
df2 = df2.sort_values(by=['mae',], ascending=True)
df3 = df3.sort_values(by=['mse',], ascending=True)
df4 = df4.sort_values(by=['süre',],  ascending=True)

axes1 = df1.plot.bar(rot=0,figsize=(6,2),color='g')
plt.savefig(r"C:\Users\hp\Desktop\sonuclar\r2.jpg")
plt.show()

axes2 = df2.plot.bar(rot=0,figsize=(6,2),color='b')
plt.savefig(r"C:\Users\hp\Desktop\sonuclar\mae.jpg")
plt.show()

axes3 = df3.plot.bar(rot=0,figsize=(6,2),color='gray')
```

```
    plt.savefig(r"C:\Users\hp\Desktop\sonuclar\mse.jpg")
    plt.show()

    axes4 = df4.plot.bar(rot=0,figsize=(6,2),color='red')
    plt.savefig(r"C:\Users\hp\Desktop\sonuclar\süre.jpg")
    plt.show()

    axes5 = df5.plot.bar(rot=0,figsize=(6,2),color='orange')
    plt.savefig(r"C:\Users\hp\Desktop\sonuclar\tahmin.jpg")
    plt.show()



except Exception as ex:
    print (ex)
    print('--- VERI BEKLENIYOR ---')
    time.sleep(10)
```