Journal of the Faculty of Engineering and Architecture of Gazi University 40:3 (2025) 1647-1658

Mühendislik Mimarlık Fakültesi Dergisi
Journal of The Faculty of Engineering and Architecture of Gazi University
Elektronik / Online ISSN : 1304 - 4915
Basılı / Printed ISSN : 1300 - 1884

Anomaly detection model with deep learning methods from big data perspective in kubernetes architecture

Mehmet Ulvi Şimşek*, Nuh Ali Akgül, Murat Akın
[1]ASELSAN, Ankara, 06200, Türkiye
[2]Local US Informatics Inc., Ankara, 06530, Türkiye
[3]TUSAŞ Kazan Vocational School, Computer Technologies Department, Gazi University, Ankara, 06980, Türkiye

Highlights:

- A generalized framework for anomaly detection in big data using kubernetes
- Traffic anomaly detection in smart city
- A distributed and fault-tolerant big data architecture using kubernetes

Keywords:

- Traffic data
- Anomaly detection
- Big data
- Kubernetes
- Deep Learning

Article Info:

Research Article
Received: 26.07.2023
Accepted: 08.01.2025

DOI:

10.17341/gazimmfd.1333162

Correspondence:

Author: Murat Akın
e-mail: muratakin@gazi.edu.tr
phone: +90 505 664 3810

Graphical/Tabular Abstract

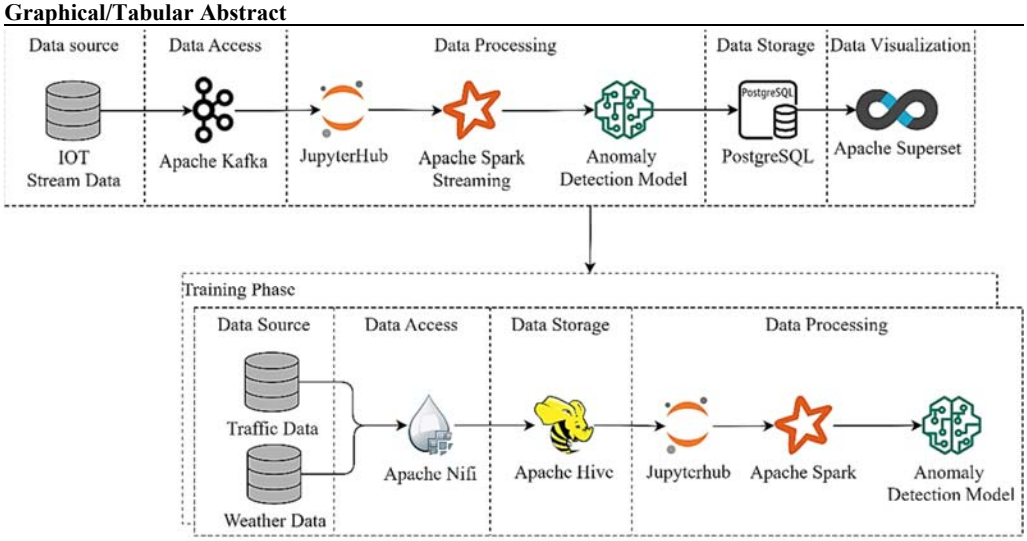


Figure A. General Flow of Data Anomaly Detection Model

Purpose: In the present days, with the increasing number of sensors, a vast amount of data is being generated at high speed, diverse sources, and large volume is crucial for analysing the jointly produced data from different sources. The different kind of sensors data are analysed together with the help of big data architecture. Big data architectures are preferred for the processing of large volumes of data flowing at a fast rate based on big data concepts for anomaly detection on big data. In this manner, the purpose of study is to develop an anomaly detection model in big data with using kubernetes architecture which enables us to fault tolerance and distributed system.

Theory and Methods: This study presents a novel framework for big data anomaly detection that combines the strength of the kubernetes and deep learning algorithms, which can be seen in Figure A. In this framework, different kind of open-source applications are used in big data layer architecture to gathering, filtering, altering, processing and visualizing data. In the architecture, kubernetes system is preferred to orchestrate In the processing stage, deep learning algorithms are used to detect anomalies in the processing stage that are processed distributed environment in kubernetes system.

Results: The proposed anomaly detection model using LSTM, GRU and CNN algorithms was tested on Kubernetes architecture with using traffic and weather dataset. The experimental results showed that the CNN model has the highest prediction success among the others models used in the system. On the other hand, the framework performance on the kubernetes system is examined in terms of CPU usage, it is seen that the operations are distributed to the worker nodes. As a result, the proposed kubernetes based big data anomaly detection model offers a powerful anomaly detection model to combine the different kinds of sensors values and process together from data accessing stage to presentation stage.

Conclusion: In the study, an anomaly detection model was proposed within the scope of kubernetes-based big data. The proposed model is a generalized model and anomaly detection process has been done by considering traffic data. The experimental results show that the outcomes of the models are satisfying and successful. The framework offers new and alternative solutions to data analytics in big data systems.

Journal of the Faculty of Engineering and Architecture of Gazi University 40:3 (2025) 1647-1658

Mühendislik Mimarlık Fakültesi Dergisi
Journal of The Faculty of Engineering and Architecture of Gazi University

Elektronik / Online ISSN : 1304 - 4915
Basılı / Printed ISSN : 1300 - 1884

Kubernetes mimarisinde büyük veri perspektifinden derin öğrenme yöntemleriyle anomali tespit modeli

Mehmet Ulvi Şimşek*, Nuh Ali Akgül, Murat Akın
[1]ASELSAN, Ankara, 06200, Türkiye,
[2]Local US Bilişim A.Ş., Ankara, 06530, Türkiye,
[3]Gazi Üniversitesi, Bilgisayar Teknolojileri Bölümü, TUSAŞ Kazan Meslek Yüksekokulu, Ankara, 06980, Türkiye

ÖNEÇIKANLAR

- Büyük veri yapısında kubernetes kullanımı ile anomali tespitinin genelleştirilmiş bir çerçevesi
- Akıllı şehirlerde trafik anomalisi tespiti
- Kubernetes kullanarak dağıtık ve hata toleranslı bir büyük veri mimarisi

Makale Bilgileri
Araştırma Makalesi
Geliş: 26.07.2023
Kabul: 08.01.2025

DOI:
10.17341/gazimmfd.1333162

Anahtar Kelimeler:
Trafik verisi,
anomali tespiti,
büyük veri,
kubernetes,
derin öğrenme

ÖZ

Günümüzde sensör sayısının artması ile birlikte yüksek hızda, çeşitlilikte ve hacimde veri üretilmektedir. Üretilen yüksek hızda ve farklı kaynaklardan gelen verinin birlikte analiz edilmesi önem arz etmektedir. Bu noktada büyük veri sistemleri katmanlı mimarisi ile birlikte çözümler sunmaktadır. Her bir katmanda farklı uygulamalar çalışmakta ve birbirleri ile iletişim kurarak çalışmaktadırlar. Bu çalışma, Kubernetes mimarisi kullanılarak sensör verilerinin birleştirilmesi, yapay zeka yöntemleri ile anomali tespiti ve anlık verilerin işlenmesine yönelik model sunmaktadır. Önerilen sistem modeli verilerin işlenmesi ve birleştirilmesi, yapay zekâ tabanlı model geliştirilmesi ve görselleştirme aşamalarından oluşmaktadır. Bu noktada Kubernetes mimarisi ile birlikte orkestrasyon işlemi sağlanarak açık kaynak kodlu uygulamalar aracılığı ile veri birleştirme, işleme ve görselleştirme işlemleri dağıtık, hata toleranslı ve etkin kaynak yönetimi sağlayacak şekilde oluşturulmuştur. Anomali tespit işlemi için ise yapay zekâ algoritmalarından LSTM, GRU ve Conv1d algoritmaları kullanılarak karşılaştırmalı analiz yapılmıştır. Sistemin normal durumuna, eğitim ve anlık veri analizi aşamasındaki durumlarına ilişkin işlemci kullanım oranları karşılaştırılarak sonuçlar sunulmuştur. Sonuç olarak önerilen Kubernetes tabanlı anomali tespit modeli ile baştan sona veri toplama aşamasından başlayarak veri işleme ve görselleştirme aşamalarına ilişkin bir sistem modeli gerçekleştirilmiştir. Bu bağlamda önerilen sistem modelinin akış verilerini birleştirerek analizinde dağıtık işlem gerçekleştirme ve hata toleranslı şekilde işlemleri gerçekleştirdiği görülmüştür.

Anomaly detection model with deep learning methods from big data perspective in kubernetes architecture

HIGHLIGHTS

- A generalized framework for anomaly detection in big data using kubernetes
- Traffic anomaly detection in smart city
- A distributed and fault-tolerant big data architecture using kubernetes

Article Info
Research Article
Received: 26.07.2023
Accepted: 08.01.2025

DOI:
10.17341/gazimmfd.1333162

Keywords:
Traffic data,
anomaly detection,
big data,
kubernetes,
deep learning

ABSTRACT

Today, with the increasing number of sensors, data is generated at high speed, in various formats, and in large volumes. Analyzing this rapidly produced data from different sources is crucial. To address this challenge, big data systems with layered architecture offer viable solutions. Each layer contains different applications that interact with one another. This study introduces a model for combining sensor data using the Kubernetes architecture, incorporating anomaly detection through artificial intelligence methods, and processing stream data. The proposed model comprises three stages: data processing and merging, an artificial intelligence-based model, and visualization. To facilitate these stages, the orchestration process utilizing the Kubernetes architecture enables the creation of data merging, processing, and visualization processes by leveraging open-source applications. This approach allows for the development of a distributed, fault-tolerant, and efficient resource management system. In the anomaly detection stage, a comparative analysis is conducted using artificial intelligence algorithms, namely LSTM, GRU, and Conv1d. Additionally, the results are presented by comparing processor usage rates during normal system operation and during the training and stream data analysis phases. Consequently, we propose a Kubernetes-based anomaly detection model that encompasses the stages of data collection, data processing, and visualization. The proposed system model successfully combines streaming data and executes operations in a distributed and fault-tolerant manner during the analysis process.

*Sorumlu Yazar/Yazarlar / *Corresponding Author/Authors: mehmetulvi@gmail.com, nuhaliakgul@gmail.com, *muratakin@gazi.edu.tr / Tel: +90 505 664 3810

1. Giriş (Introduction)

Sensörlerden alınan veriler, birbirine bağlı cihazlar, akıllı ev uygulamaları, akıllı şehir uygulamaları, iletişim, sağlık gibi uygulamaların artması ile birlikte gerçek zamanlı olarak akan veri miktarı günden güne artmaktadır [1]. Hızla gelişen teknolojilere paralel olarak günümüzde günlük olarak üretilen veri miktarı üssel bir oranda artarken üretilen büyük miktarda verinin depolanması ve yönetilmesi gerekmekte, anlamlı bilgiler çıkarılması önem arz etmektedir. Elde edilen farklı alanlara ait büyük veri araştırmacılar tarafından analiz edilerek çeşitli çözümler önerilmiş, verinin toplanması, depolanması, işlenmesi ve analizi konusunda birçok yaklaşım geliştirilmiştir. [2]. Yapılan çalışmalar incelendiğinde farklı alanlarda istenmeyen, aykırı verilerin temizlenmesi için anomali tespiti çalışmaları mevcuttur ve anomali tespit işlemi büyük veri uygulamaların çeşitli olması sebebi ile önemli bir araştırma olarak öne çıkmaktadır. [3].

Büyük veri konsepti farklı boyutlarda ele alınsa da temel olarak hacim, hız ve çeşitlilik kavramları ile incelenmektedir. [4, 5]. Büyük verinin temel olarak ele alınan boyutları incelendiğinde;

- Hacim: Büyük veri miktarının boyutsal ifadesidir. Organizasyonlar tarafından üretilen veri yıllar geçtikçe artmaktadır. Organizasyonların kullandığı kurumsal telefon uygulamalarından, web sitelerinden veya sosyal medya hesapları gibi dijital kaynaklardan veri akışları olabilmektedir. Böylelikle elde edilen ve üretilen büyük miktarda veri için miktar kavramı hacim olarak ifade edilmektedir.
- Hız: Verinin üretilme ve iletilme aşamasındaki süratini tanımlamaktadır. Bazı verilerin işlenmesi, çıktılarının oluşturulması ve sonrasında analizi için veri hızı oldukça önemlidir. Verinin hızına bağlı olarak veri akışlarının gerçek zamanlı değerlendirilmesi ve işlenmesi gerekmektedir.
- Çeşitlilik: Türdeş olmayan, farklı karakteristiğe sahip verileri karşılamaktadır. Geleneksel veri tabanı yapılarında ilişkisel olarak tutulan verilerden ziyade yapılandırılmamış, yeni veri türleri büyük veri kavramı içerisinde karşımıza çıkmaktadır ve bu yapılandırılmamış verilerin işlenebilecek bir formata dönüştürülmesi gerekmektedir.

Büyük veri üzerinde anomali tespiti için büyük veri kavramlarından yola çıkarak büyük hacimde ve hızlı oranda akan verinin işlenebilmesi için katmanlı mimariler tercih edilmektedir. Bu katmanlar genellikle alt katmandan üst katmana doğru ilerleyecek şekilde her katmanda yapılan işlemler nitelendirilerek tasarlanmaktadır. Katmanlı mimari için literatürde farklı şekilde yaklaşımlar bulunurken, çoğunlukla verinin en alt katmandan alınarak sunum aşamasına gelmesine kadar olan süreçte yapılan işlemler katmanlara ayrılarak sınıflandırılmıştır. Büyük veri mimarisinde genel olarak üç katmanlı yapı karşımıza çıkmaktadır. Bunlar veri toplama ve depolama, veri işleme ve uygulama katmanı olarak ifade edilmektedir. Wang vd. [5] çalışmasında bu katmanları aşağıdaki gibi tanımlamıştır.

- Veri Toplama ve Depolama Katmanı: Bu katmanda veriler statik veriler ve dinamik akış olarak iki türde ele alınmaktadır. Temel olarak veri toplama işlemi için tüm akış verisi yerine işlenecek olan ve ihtiyaç duyulan akış verisinin aktarılması, diğer akış verilerinin dikkate alınmaması prensibi geçerlidir. Depolama kısmında ise geleneksel ilişkisel veri tabanı ve yapılandırılmış veri yönetim tekniklerinden ziyade büyük veri karakteristiğinin yansıması olarak dağıtık dosya sistemleri, NoSQL, NewSQL ve diğer veri yönetim sistemlerinin entegre edilmesini sağlayacak işlemler yer almaktadır.
- Veri İşleme: Ham verinin temizlendiği, ayrıştırıldığı ve diğer büyük veri sistemleri için kullanılabilir hale getirildiği katmandır. Bu katmanda tarihsel veri analizi, akış veri analizi, çizge(graf) veri analizi ve hibrit veri analizi olarak işlemler yer almaktadır.
- Uygulama Katmanı: Büyük verinin tavsiye sistemleri, duygu analizi, arama motorları gibi uygulandığı çeşitli alanlar mevcuttur ve uygulamalar ilk iki katman sonucu elde edilen ve işleme tabi tutulan büyük verinin farklı tekniklerle işlenmesi ile gerçekleştirilmektedir.

Yukarıda belirtilen katmanların dışında çeşitli araştırmalarda geliştirilen büyük veri mimarilerinde katmanlar yapısal olarak farklı ele alınmış, bazı durumlarda arttırılmış/azaltılmış, yapılan işlemler tanımlanan katmanlar da gerçekleştirilmiştir. Benhlima vd. yaptıkları çalışmada büyük veri mimarisini veri depolama, büyük veri işleme ve analitik, görselleştirme katmanları ile ele almıştır. [6]. Benzer olarak Hajjaji vd. çalışmasında [7] nesnelerin interneti (IoT) konseptini dikkate alarak büyük veri mimarisini depolama, iletim, işleme ve analiz katmanlarına ayırarak gerçekleştirmiştir. Kune vd. çalışmasında [8] verinin katmanlar arasındaki işlemleri sınıflandırılmış, fiziksel veri, veri işleme ve analitik katmanı olarak farklı katmanlar belirlenmiştir. Fiziksel veri katmanında farklı veri türlerinin operasyonel veya tarihsel veri olarak ayrılmasıyla veri ambarına aktarılma aşamalarını içermektedir. Veri katmanında verinin bloklara bölünmesi ve düzenlenmesi sağlanmaktadır. İşleme katmanı NoSQL erişim, Map / Reduce, alan bazlı istatiksel işlemler ve makine öğrenmesi teknikleri gibi işlemlerin yapıldığı aşamadır. Analitik katmanı ise alana özgü kullanılan standartlar araçları ve analiz tekniklerini içermektedir. Büyük veri analitiği üzerine yapılan bir derleme çalışmasında [9] katmanlar veri katmanı, analitik katman, entegrasyon ve karar katmanları olarak tanımlanmıştır. Bu katmanlardan biri olan veri katmanında, veri ambarı, veri kaynakları ve Spark, Hadoop gibi büyük veri ürünleri yer almaktadır. Analitik katman, yerel verinin güncellenmesinin belirli aralıklarla yapıldığı ve analiz performansının artırılmaya çalışıldığı katmandır. Entegrasyon katmanında analitik katman için kullanıcı uygulamaları arası entegrasyonun yapılırken, karar katmanı ise son kullanıcılara yönelik masaüstü ve interaktif uygulamalarını içermektedir.

Literatürde uzun zamandır bir çalışma konusu olan anomali tespiti ile veri seti üzerinde beklenmeyen noktaların veya dizilerin tespit edilmesi amaçlanmaktadır. Bir başka ifade ile normal, standart ve beklenenden farklı nesneler olağan dışı anlamında "anomali" olarak işaretlenmektedir [10]. Büyük veri açısından bakıldığında anomali tespiti yapabilmek için bazı bileşenlerin probleme özgü olarak ele alınması ve geliştirilen mimariye eklenmesi gerekmektedir. Rettig vd. çalışmasında [11] bileşenlerin akan ve durağan veri analizi için eklendiği görülmektedir. Yapılan çalışmada elde edilen verilerin hem akan hem de durağan veri düzeyinde analizi yapılması için ilk olarak veri deposuna Spark aracılığıyla iletilmiş, depolama için HDFS kullanılmış, Spark Streaming ile veri analizi gerçekleştirilmiş ve akan veriden sorgu işlemleri Kafka ile elde edilerek Spark streaming aracılığıyla sonuç döndürülmüştür. Sonuç olarak çalışmada önerilen model sayesinde Spark Streaming ve Spark araçları kullanılarak YARN mimarisinde anomali tespit işlemi hem duran veri hem de akan veri üzerinde gerçekleştirilmiştir. Sistemin çıktıları hem nümerik hem de kategorik veriler göz önünde bulundurularak, göreceli entropi ve Pearson korelasyonu metrikleri ile değerlendirilmiştir. Stojanovic vd. [12] çalışmasında veri depolama, veri işleme, analitik ve sunum katmanlarını içeren bir mimari önermiştir. Veri depolama katmanında HDFS ve PostgreSQL kullanılmış, veri işleme katmanında gerçek

zamanlı işlem için Storm ve karmaşık olay işleme (CEP) teknolojilerinden faydalanılmış, tarihsel verilerin işlenmesi Spark ve Hadoop ekosistemiyle gerçekleştirilmiştir.

Veri görselleştirme katmanında ise REST servisler ve web portal aracılığıyla sonuçlar son kullanıcıya gösterilmiştir. Anomali tespit işlemi için KNN kümeleme yöntemi ile birlikte CEP yöntemi hibrit şekilde uygulanmıştır. Ancak çalışma sonunda değerlendirme için algoritma başarım oranlarına yer verilmemiştir. Jallad vd. [13] çalışmasında anomali tespiti için büyük veri içerisinde derin öğrenme ve geleneksel makine öğrenmesi modelleri kullanmış ve modelleri doğrulama değerleriyle karşılaştırmıştır. Çalışmada derin öğrenme yöntemlerinden LSTM tercih edilerek model hata oranı yaklaşık olarak 0,10 olarak bulunmuş, fakat çalışmada büyük veri mimarisi açısından değerlendirme yapılmamıştır. Zhou vd. [14] çalışmalarında endüstriyel büyük veri için LSTM yöntemi kullanarak anomali tespiti yapmışlardır. Parwez vd. [15] çalışmalarında mobil büyük veri üzerinde K-means ve hiyerarşik kümeleme yöntemleri kullanılarak kullanıcıların farklı yer ve zamandaki aktivitelerinde anomali tespiti yapmış, alışılmışın dışındaki kullanıcı aktiviteleri trafik isteğine sebep olması dolayısıyla anomali olarak etiketlenmiştir. Model değerlendirmelerinde Ortalama Kare Hatası (MSE) metriğini kullanarak model başarımını değerlendirmişlerdir. Karatepe vd. [16] çalışmalarında gerçek bir hücresel ağda kullanıcı aramalarını analiz ederek anomali tespiti yapmışlardır. Çalışmalarında HDFS Map Reduce ve Hadoop açık kaynak uygulamalarını kullanmış, geleneksel veri ambarı yaklaşımlarına kıyasla bulut hizmetleri tarafından sağlanan hesaplama kolaylığının veri işleme maliyetini azalttığı ifade edilmiştir. Buna ek olarak verilerin geleneksel işletmelerde tutularak işlenmesi operatör ve veri tabanı sağlayıcıları arasından lisans anlaşmalarına bağlı olarak daha pahalı olacağı ileri sürülmüştür.

Büyük veri mimarisinde işlemlerin yönetimi, koordinasyonu ve zamanlaması kritik bir öneme sahiptir. Kubernetes, mikro hizmet sağlayan uygulamaların konteynerler içerisinde çalışmasına olanak tanıyan bir platform olarak öne çıkmaktadır. Bu mimari, hızlı ve verimli kaynak kullanımı ile küme içinde dağıtım, ölçekleme ve yönetim süreçlerini otomatikleştirerek esnek bir yapı sunmaktadır. [17]. Kubernetes mimarisi, kontrol düzlemi, işçi düğümleri ve orkestrasyonun sağlandığı yapıdan oluşmaktadır. Kullanıcılar, Kubernetes'in orkestrasyon yeteneklerinden faydalanmak için kubectl komut satırı aracını kullanarak kontrol düzlemiyle etkileşimde bulunurlar. Kontrol düzlemi, küme durumunu izlemek ve çalışan düğümleri yönetmek amacıyla kritik bir rol üstlenmektedir. İşçi düğümleri ise, uygulamaların pod içerisinde çalıştırılması için gerekli altyapıyı sağlamaktadır. [18]. Kubernetes'in esnek ve kaynak verimli yönetim özellikleri, uygulamaların minimum kesinti ile hizmet sunmasını mümkün kılmakta ve bu durum, Kubernetes tabanlı uygulamaların artışına zemin hazırlamaktadır. Bu alanda yapılan çalışmalardan Xu vd. [19] çalışmasında görev planlama yönetimi problemi için Kubernetes mimarisi kullanarak derin öğrenme teknikleri ile bir model önermiş, çalışma sonucunda kaynak verimliliği ve sistem performansı açısından iyileşmeler olduğunu ifade etmişlerdir. Ancak kullanılan algoritma sonuçlarına ilişkin değerlendirme makale içerisinde sunulmamıştır. Dang-Quang vd. [20] çalışmalarında iş yükünü dinamik olarak yönetmek amacı ile Kubernetes mimarisi kullanmış, ARIMA ve BI-LSTM modellerini kullanarak karşılaştırmalı sonuçlarını sunmuşlardır. Jung vd. [21] çalışmalarında etkili operasyon ve tesis yönetimi için Kubernetes tabanlı bir anomali tespit modeli önermişlerdir. Model içerisinde kullanılan sensör verileri açık kaynak kodlu uygulamalardan, RabbitMQ, lostash ve Elastic Search uygulamaları ile elde edildiği ifade edilirken, LSTM ve SVM yöntemlerini kullanarak sınıflandırma başarım sonuçlarını sunmuşlardır. Hernandez vd. [22] çalışmalarında Kubernetes altyapısı kullanarak büyük veri içerisinde anomali problemini ele almışlardır. Veri depolama katmanında Hadoop kümesi, veri işleme katmanında Spark kütüphaneleri, veri motoru ve veri depoları gibi kaynaklara ulaşmak amacı ile Linux terminali kullanmışlardır. Önerilen küme kapsamında uygulamalar arası etkileşim ve altyapı önerilmiştir. Muralidharan vd. [23] çalışmalarında akıllı şehir ortamında güvenli, dağıtık ve güvenilir bir izleme sistemi önermişlerdir. Çalışmalarında Kubernetes kullanarak dağıtık mimarinin IoT ortamındaki etkinliğini göstermişlerdir. Chen vd. [24] pil izleme yönetimi amacı ile Kubernetes ve bulut-kenar teknolojilerini entegre eden bir platform önermişlerdir. Kubernetes mimarisi içerisinde termal kaçak problemi için K-Means algoritması kullanmış, çalışma altyapısında beş işçi düğüm ve bir kontrol düzlemi/master olmak üzere toplam 6 düğüm çalıştırılmıştır. Platform etkinliğini göstermek amacı ile K-Means algoritması ile önerdikleri sistemin gecikmesini azalttıklarını ifade etmişlerdir.

Bu çalışmada büyük veriden Kubernetes tabanlı anomali tespit modeli önerilmiştir. Önerilen model geliştirilirken derin öğrenme yöntemlerinin ve Kubernetes mimarisinin yeteneklerinden faydalanılmıştır. Sistem mimarisi katmanlarında farklı açık kaynak kodlu uygulamalar ve bileşenleri bir araya getirilmiş, derin öğrenme yöntemleri kullanılarak anomali tespit işlemi yapılmış, gerçekleştirilen tüm uygulamalar Kubernetes altyapısıyla büyük veri konseptine uygun olarak gerçekleştirilmiştir. Önerilen model ve yapılan çalışmanın yenilikçi yönleri ve katkıları aşağıda sunulmuştur:

- Yapılan çalışma ile büyük veri konseptine uygun olacak şekilde Kubernetes tabanlı anomali tespiti için genelleştirilmiş yeni bir çerçeve model önerilmiş ve akıllı şehir uygulamalarında kullanılan verilerden yararlanılarak model doğruluğu sağlanmıştır.
- Trafik verileri kullanılarak büyük veri kapsamında Kubernetes mimarisini içeren anomali tespiti bu alanda ilk defa yapılmıştır. Önerilen model içerisinde gerçekleştirilen Kubernetes mimarisi sayesinde dağıtık kaynak yönetimi yapabilen ve hataya toleranslı bir sistem tasarlanmıştır.
- Önerilen model üzerinde farklı derin öğrenme yöntemleri kullanılarak karşılaştırmalı sonuçlar elde edilmiştir. Sistem üzerindeki işlemci yük hesaplamaları dikkate alınarak sonuçlar sunulmuştur.

Çalışmanın organizasyonunda; 2. bölümde kullanılan veri kümesi ve uygulanan yöntemler açıklanmış, 3. bölümde önerilen model ve süreci anlatılmış, elde edilen sonuçlar 4. bölümde sunulmuş ve tartışılmıştır. Çalışma sonucu son bölümde verilerek kısa bir değerlendirme yapılmıştır.

2. MATERYAL VE METOT (MATERIAL AND METHOD)

Bu bölümde anomali belirleme aşaması için girdi olacak veri kümesi ve kullanılan yöntemler verilmiştir. Kubernetes tabanlı büyük veride anomali tespit edecek sistemde veri kümesi olarak 8 adet özniteliğe sahip trafik verisi kullanılmış, veriler ön temizleme işlemine tutularak hazırlanmış, sonrasında LSTM, GRU ve CNN algoritmaları kullanılarak sınıflandırma gerçekleştirilmiştir. Kullanılan veri kümesi ve yöntemlere ait detay bilgiler alt başlıklarda verilmiştir.

2.1. Veri Kümesi (Data Set)

Önerilen modelin doğruluğunu sağlamak ve deneysel sonuçları gerçekleştirmek için City Pulse EU projesi [25] bünyesinde toplanan trafik ve hava verileri kullanılmıştır. Trafik verilerinin detaylarında sabit bir noktalardan geçen araç yoğunluk verisi ve ortalama hız verileri altı ay süresince belli aralıklar ile toplandığı, hava durum verilerinin içeriğinde ise çiğ noktası, basınç, sıcaklık, rüzgâr şiddeti ve yönü verileri toplandığı ifade edilmiştir [26]. Modelin eğitimi ve test sonuçlarının elde edilmesi için kullanılan verinin nitelikleri açıklaması ile Tablo 1'de, 15559 kayıt içeren veri kümesinin niteliklerine ait istatistiki bilgiler Tablo 2'de verilmiştir.

Tablo 1. Kullanılan veri kümesine ait öznitelik bilgileri (Attribute information of the dataset used)

Nitelik	Açıklama
Ortalama hız (avgspeed)	Belirli bir süre boyunca iki nokta arasında gözlemlenen araç ortalama hızı
Araç sayısı (vehicle count)	Belirli bir süre boyunca iki nokta arasında gözlemlenen araç sayısı
Rüzgâr hızı (wspdm)	Saatlik bazda rüzgâr hızı
Nem(hum)	Ölçümlenen yüzdesel nem oranı
Rüzgâr yönü (wdird)	Bölgedeki anlık rüzgâr yönü
Basınç (pressurem)	Basınç miktarı
Sıcaklık (tempm)	Ölçümlenen sıcaklık değeri
Çiğ noktası (dewptm)	Su buharının sıvı duruma geçtiği sıcaklık değeri

Tablo 2. Veri kümesine ait istatiksel bilgiler (Statistical information about the dataset)

	Ortalama hız	Araç sayısı	Rüzgâr hızı	Nem	Rüzgâr yönü	Basınç	Sıcaklık	Çiğ noktası
Ortalama değer	93,43	5,93	12,76	70,13	185,79	1013,79	15,58	10,38
Standart Sapma	12,15	6,03	3,53	7,43	43,52	3,94	1,83	1,43
Minimum	6	0	0	16	0	992	2	0
25%	88	1	12,7	70,1	185,3	1013,8	15,6	10,4
50%	94	4	12,7	70,1	185,3	1013,8	15,6	10,4
75%	101	9	12,7	70,1	185,3	1013,8	15,6	10,4
Maksimum	141	40	38,9	94	360	1030	27	19

Bu çalışmada büyük veriden anomali tespiti için LSTM, GRU ve CNN yöntemleri kullanılmıştır. Bu yöntemlerden LSTM algoritması ilk olarak 1997 yılında önerilmiş olup basit bir RNN bloğuna hafıza bloğu ve kapılar eklenerek genişletilmiş [27] ve uzun vadeli bağımlılıkları aktarılması amacı ile tasarlanmıştır [28]. Hafıza bloğu hücreler arası bilgi akışını düzenleyen giriş, çıkış ve unutma kapıları ile desteklenmiştir [29]. LSTM metodunda ilgili hücreler zamana karşı mevcut durumlarını korurken uzun zaman bağımlıklarını da öğrenerek geliştirmektedirler. Örneğin giriş elemanları x= (x1, x2, x3,…,xn) şeklinde olan bir LSTM modelinde belirli bir zaman aralığında(t) input(i), forget(f) ve output(o) olmak üzere 3 kapıdan oluşmaktadır. LSTM modelini oluşturan kapılara ait matematiksel denklemler Eş. 1-3 ile ifade edilmiştir. Eşitliklerde aktivasyon fonksiyonu σ simgesi ile, ağırlık matrisleri W simgesiyle, bias değeri ise b harfi ile gösterilmiştir. Hücreye ait durum(ct) Eş. 4 ile hesaplanırken, gizli katman(h) Eş. 5'de tanımlanmıştır.

$$i_t = \sigma(W_{iX_t} + W_{ih_{t-1}} + b_i) \quad (1)$$

$$f_t = \sigma(W_{fX_t} + W_{fh_{t-1}} + b_f) \quad (2)$$

$$o_t = \sigma(W_{oX_t} + W_{oh_{t-1}} + b_o) \quad (3)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot tanh(W_{cX_t} + W_{ic_{t-1}} + b_c) \quad (4)$$

$$h_t = o_t \odot tanh(c_t) \quad (5)$$

GRU algoritması Cho vd. [30] tarafından 2014 yılında geliştirilmiş, LSTM'in daha basit bir uygulaması olarak uzun vadeli bağımlılıkları yakalamada iyi sonuçlar verecek şekilde tasarlanmıştır. GRU yapısında sıfırlama ve güncelleme kapıları kullanılmamaktadır [31]. CNN modeli ise daha çok görüntü verisinde sınıflandırma yöntemi olarak kullanılmaktadır. Tek boyutlu CNN giriş veri dizisine uygulanarak girdi dizisi verilerinden önemli özellikler çıkartmak için kullanılmaktadır [32]. Zaman serileri problemlerinde geleneksel CNN modelleri yerine tek boyutlu CNN modelleri kullanılması daha uygun görülmektedir [33]. Bunun sebebi zaman serilerinin vektör olarak tek boyutlu ifade edilmesi olarak söylenebilir. CNN yapısında havuzlama ve evrişimli katmanlar yer almaktadır. Havuzlama katmanında, önceki katman boyutunun boyutu azaltılır. Evrişimli katmanda, matrisin kaydırma işlemi yardımıyla girdi verilerinden daha küçük özelliklere sahip veriler çıkartılarak verinin haritaları çıkarılır [34]. Sadeleştirilmiş LSTM olarak ifade edilebilecek GRU yöntemi ise üç kapılı sistem yerine güncelleme ve sıfırlama olmak üzere 2 kapıya sahiptir. Eş. 6-9 ile GRU denklemleri verilmiş olup, eşitliklerde x girdi vektörü, h çıktı vektörü, z güncelleme vektörü, r sıfırlama vektörü, ağırlık matrisleri W simgesiyle, bias değeri ise b harfi ile gösterilmiştir.

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \underline{h_t} \quad (6)$$

$$\underline{h_t} = g(W_t\, x_t + U_t\, (r_t \odot h_{t-1}) + b_t) \quad (7)$$

$$z_t = g(W_z\, x_t + U_z\, h_{t-1} + b_z) \quad (8)$$

$$r_t = g(W_r\, x_t + U_r\, h_{t-1} + b_r) \quad (9)$$

Açıklanan LSTM, GRU ve CNN yöntemlerinde kullanılacak olan adım sayısı(epoch), sinir ağı katman sayısı, filtre boyutu gibi parametrelerin optimum değerlerini bulmak için ızgara arama tekniğini kullanılmıştır. Izgara yöntemi ile elde edilen ve Tablo 3'de gösterilen parametreler, en iyi yapıyı oluşturmak için çok sayıda deneyden elde edilmiştir.

Tablo 3. Izgara yöntemiyle elde edilen optimum model parametreleri (Optimum model parameters obtained by the grid method)

Model	Parametreler
CNN	Epoch: 100, Filter size: 256, Max pooling
GRU	Epoch: 100, Hidden Layers:1, Batch Size: 2, Units:128
LSTM	Epoch: 100, LSTM Layers:2, Hidden Layers:1, Batch Size: 2, Units: 128

Büyük veriden anomali tespiti için geliştirilen modelin sonuçlarını değerlendirmek için literatürde sıklıkla kullanılan hata göstergeleri olan RMSE ve MAPE hata ölçüm değerleri kullanılmıştır. Hata oranları, *n* elemanlı bir dizi için her tahmin değerinden ($\hat{y}_i$) ile gerçek değer (y_i) arasındaki fark değerlerini dikkate alınarak hesaplanmaktadır. RMSE ve MAPE hesaplama eşitlikleri sırasıyla Eş. 10 ve 11'de ifade edilmiştir.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\widehat{y_i} - y_i)^2} \quad (10)$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\widehat{y_i} - y_i|}{y_i} \quad (11)$$

Modelin anomali etiketleme aşaması yeniden çatılma hatası (reconstruction error) kullanılarak belirlenen eşik değeri doğrultusunda değerlendirilmiştir. Anomali değer, beklenen değer ve gerçek değer arası farkın eşik değerinden büyük olmasına göre karar verilmiştir. Yeniden çatılma hatası hesaplama eşitliği Eş. 12 ile ifade edilmiştir.

$$RE = (\bar{x} - x) \quad (12)$$

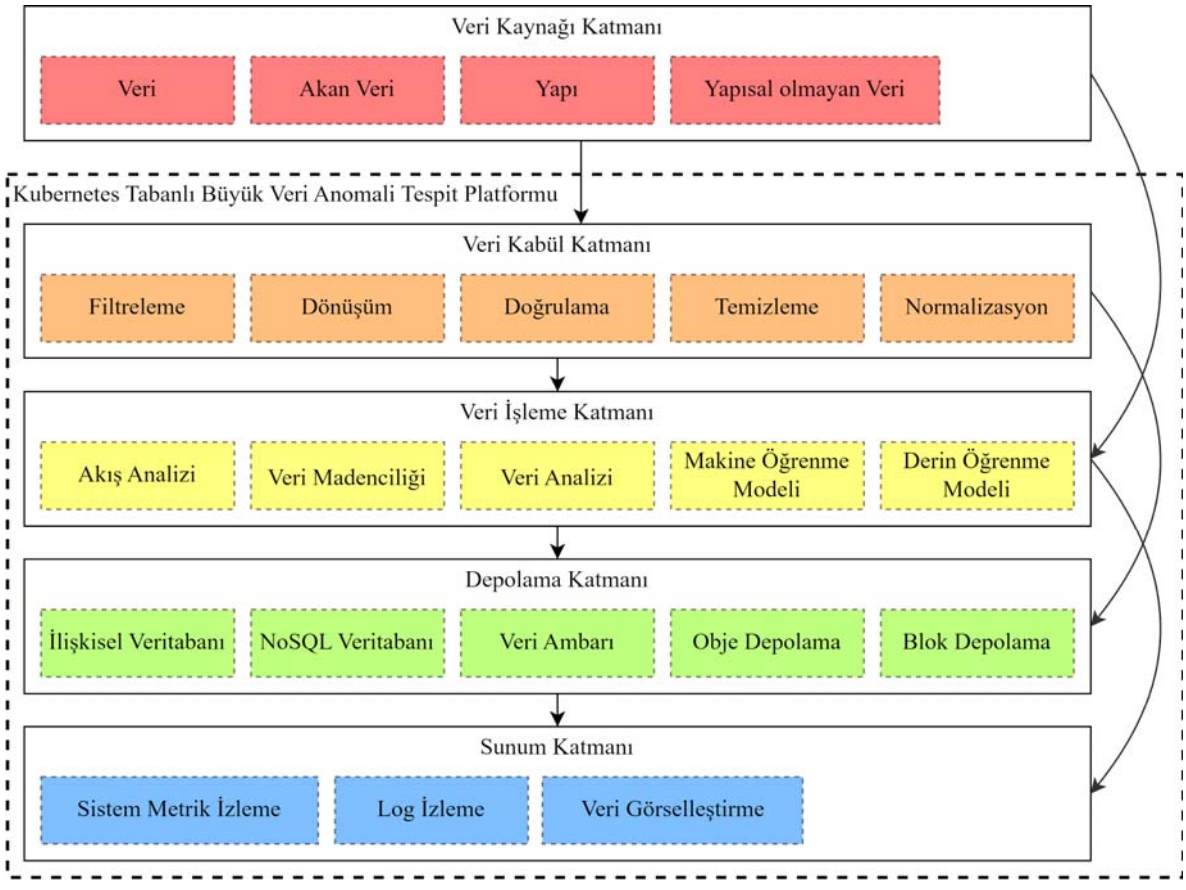
Zaman serisi S = (s_1, s_2, s_3, s_4, .., s_t) her nokta (s_t), t anındaki sensör kayıtlarını temsil eder. s_1'den s_t'ye verilen giriş zaman serisi dizisi için zaman serisinin bir sonraki değerlerini tahmin etmek üzere geçmiş ardışık değerlerle eğitilir. s'_t, geçmiş *t - n* beslenen değerden türetilen dizinin *t + 1* zamanını temsil eden tahmin değeridir. Tahmin edilen nokta s'_t için yeniden çatılma hatası (RE = (r_1, r_2, r_3, r, .., r_t)) Eş. 12'ye göre hesaplanmaktadır. Ortalama yeniden çatılma hatası vektörleri, belirli bir eşik değerine sahip normal veya anormal dizileri tanımlamak için kullanılır. Eşik değeri δ'nin seçimi, denetimsiz öğrenmedeki bir diğer sorundur. Bu çalışma kapsamında eşik değeri belirlenmesi için literatürde yaygın olarak kullanılan δ = 3α olarak seçilmiştir (δ eşik değeri, α standart sapma) seçilerek anomali tespiti yapılmıştır.

3. Önerilen Model (Proposed Model)

Çalışmanın bu bölümünde, anomali tespiti için önerilen Kubernetes tabanlı modelinin genel hatları açıklanmıştır. Veri kaynağı, veri kabul, veri işleme, depolama ve sunum katmanlarından olmak üzere 5 katman olarak tanımlanan model ile akıllı şehirler için elde edilen veri kümesi üzerinde anomali tespiti yapılmıştır. Önerilen model katmanlarına genel olarak bakıldığında;

- Akıllı şehir uygulamalarında yer alan sensörler dağıtık şekilde şehrin belirli bölümlerinde yer almaktadır ve sensörlerin bulunduğu katman önerdiğimiz modelde veri kaynağı katmanı olarak ifade edilmiştir.
- Veri kabul katmanında verilerin filtrelenmesi, dönüşümü, doğrulaması, temizlemesi ve normalizasyon işlemi yapılmaktadır.
- Sensör verilerinin işlenmesi amacıyla, akış analizi ve makine öğrenmesi modelleri veri işleme katmanında çalıştırılmaktadır.
- Veri işleme katmanından elde edilen değerli bilgiler ise depolama katmanında yer alan araçlar aracılığıyla saklanmaktadır.
- Saklanan bilgiler sunum katmanında görselleştirilerek son kullanıcıya sunulmaktadır.

Bu katmanlar ve işlevleri kapsayacak şekilde, akıllı şehir konsepti problemlerinde kullanılabilen ve dağıtık mimari üzerinde çalışabilen büyük veriden anomali tespiti gerçekleştirecek sistemin Kubernetes mimarisi Şekil 1'de gösterilmiştir.



Şekil 1. Katmanlı kubernetes mimarisi (Layered kubernetes architecture)

Akıllı şehirlerin için kullanılan sistemlerin genişleyen yapısı ve istemde kullanılan veriyi sağlayıcıyı sensörlerin dağıtık olarak yerleştirilmesinden dolayı verilerle birlikte kaynakların etkin olarak ele alınması gerekmektedir. Bu hususları dikkate alacak şekilde tasarlanan Kubernetes mimarisi ve katmanları, birbirine bağlı şekilde çalışarak kaynaklar verimli ve dağıtık şekilde kullanılmış ve büyük veriden anomali tespit işlemi gerçekleştirilmiştir. Mimaride yer alan beş katmanla ve büyük veri mimarisi tercih edilerek Kubernetes orkestrasyonunu sağlayacak bir yapı oluşturulmuştur. Modele ait katmanlar ve bu katmanların yapıdaki görevleri şu şekildedir.

- Veri kaynağı katmanında akıllı şehir konseptinde yer alan sensörler yer almaktadır. Sensörler trafik verilerine ilişkin verileri toplamakta ve veri kabul katmanına iletmektedir.
- Veri kabul katmanı, verilerin büyük veri platformuna ilk giriş yeri olarak ifade edilebilir. Bu katman içerisinde verilerin filtrelenmesi, dönüştürülmesi ve analiz için ihtiyaç duyulan bilgilerin alınması gibi ön işlemler yapılmaktadır. Bu katmanda sensörler tarafından gönderilen veriler ön işlemden geçirilerek analiz veya eğitim için hazır hale getirilmektedir.
- Verinin makine öğrenmesi teknikleri ile analizi veya akış tespit işlemleri veri işleme katmanında gerçekleştirilmektedir. Bu bağlı hem duran veri analizi hem de akış verisi analiz işlemleri için Apache Spark kullanılmıştır. Veriler, Apache Spark aracılığı ile eğitilerek anomali belirleme işlemi yapılmaktadır. Analiz sonucu oluşan değerli bilgiler büyük veri platformu içerisinde yer alan depolama katmanında saklanmaktadır. Depolama katmanında HIVE ve POSTGRE tercih edilmiştir.
- Hazırlanan veriler analiz edilmesi için depolama katmanına veya veri işleme katmanına aktarılmaktadır. Depolama katmanına aktarılan veriler eğitim için kullanılmaktayken analiz katmanına aktarılan veriler akış analizi ve anlık anomali tespit etme için kullanılmaktadır. Veri kaynaklarından verinin iletilmesi için Kafka mimarisi kullanılmaktadır. Veri kabul katmanında ise Apache NIFI [35] ile birlikte veri üzerinde filtreleme, birleştirme, dönüştürme gibi işlemler yapılmıştır. Aynı zamanda NIFI aracılığı ile verilere ilişkin olarak istenilen alanlar veya farklı alanlar kullanılarak analiz işlemi için hazır hale getirilmektedir. Eğitim için kullanılması muhtemel veriler depolama katmanına aktarılarak büyük veri platformu veri tabanında saklanmaktadır.
- Sunum katmanında ise sistem metrikleri, olay günlük verileri ve son kullanıcılar için verilerin görselleştirmesi işlemleri gerçekleştirilmekte, anomali olarak etiketlenen veriler görselleştirilerek sunum katmanında son kullanıcılara sunulmaktadır. Sunum katmanında açık kaynak kodlu uygulamalardan Grafana, Prometeus ve Superset kullanılırken sistem metriklerinin toplanması ve görselleştirilmesinde Grafana ve Prometeus'dan faydalanılmıştır. Bununla birlikte anomali olarak etiketlenen veriler Superset üzerinde görselleştirilerek son kullanıcıya sunulmaktadır. Önerilen model mimarisinin süreç akışının sözde kodla ifadesi Tablo 4'de gösterilmiştir.

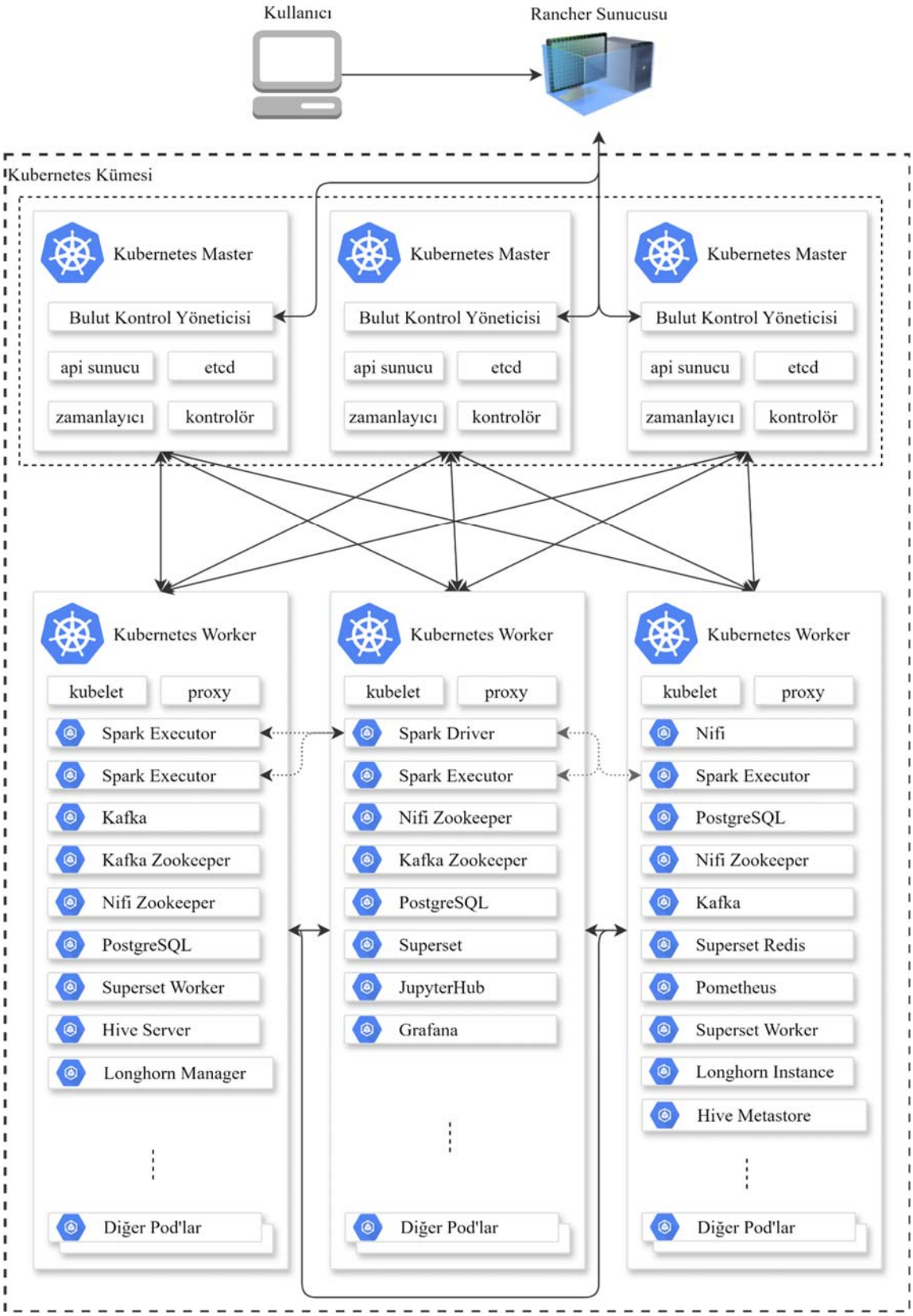
Tablo 4. Büyük Veri Anomali Belirleme modeli için geliştirilen algoritmanın sözde kodu
(Pseudocode of the algorithm developed for the Big Data Anomaly Detection model)

Algoritma: Büyük Veri Anomali Belirleme Modeli	
Girdi:	Sensor Verisi (S1.....Sn)
Çıktı:	Anomali Verisi
Adım 1:	Sensör verilerini kaynaktan okuma
Adım 2:	Verileri veri kabul katmanına aktar (Kafka - NIFI)
Adım 3:	Veri filtreleme ve ön analiz aşaması (NIFI-Depolama)
Adım 4:	Anomali etiketleme aşaması (Spark Analiz)
Adım 5:	Anomali verilerinin görselleştirilmesi (Superset-Sunum işlemi)

4. Deneysel Sonuçlar (Experimental Results)

Çalışmanın bu bölümünde, bir önceki bölümde genel hatları verilen mimarinin uygulama safhasında yapılan farklı konfigürasyon detayları ve elde edilen sonuçlar anlatılmıştır. Önerdiğimiz model mimarisi tamamı açık kaynak uygulamaların bir araya getirilmesiyle oluşturulmuş, sistem mimarisi Kubernetes tabanlı olarak tasarlanarak yatayda ölçeklenebilen, hataya karşı duyarlı ve paralel işlem yapabilme kapasitesine sahip yapıda olacak şekilde planlanmıştır. Kubernetes kümesinin yönetimi Rancher isimli açık kaynak uygulama tarafından yapılmıştır ve bu uygulama aracılığıyla fiziksel veya sanal olarak sisteme dahil edilen sunucuların kaynakları kullanılarak büyük veri anomali belirleme işlemi gerçekleştirilmiştir. Bu kapsamda Kubernetes kümesinin oluşturulması için 1 adet Rancher sunucusu, 3 adet Kubernetes ana sunucu(master) ve 3 adet Kubernetes işçi(worker) düğümü olacak şekilde toplamda yedi adet sanal makine kullanılmış, kullanılan sunucu ve düğümlere ilişkin test ortamı Şekil 2'de verilmiştir. Kurulan yapıda sunucular ve düğümler arası koordinasyon ve orkestrasyonun sağlanması için bir Kontrol Düzlemi (Control Plane) bulunmaktadır. Kontrol düzlemi bir başka deyişle Kubernetes kümesini yöneten ve idame edilmesini sağlayan yapıdır. İçerisinde API Server, Etcd, Planlayıcı, Controller ve Cloud Controller Manager bileşenlerini barındırmakta, bu bileşenlerin hepsi aynı düğümde çalışabildiği gibi her biri ayrı ayrı düğümlerde de çalışabilmektedir. Kontrol düzleminde yer alan bileşenlere ait açıklamalar aşağıda verilmiştir.

- API Sunucu (API Server): Kubernetes kümesinin merkez iletişim noktası olarak ifade edilmektedir. Kümede yer alan bileşenlerin bütün iç ve dış trafik akışı bu bileşen aracılığı ile gerçekleştirilmektedir. Ayrıca küme içerisindeki bileşenlerin hepsi API sunucu üzerinden iletişim kurmaktadır [36].
- Etcd: Kubernetes'in kendi konfigürasyonlarını ve kümenin anlık durum bilgisini anahtar-değer çifti şeklinde saklarken bu sayede yüksek düzeyde erişilebilir ve tutarlı bir veri tabanı özelliğindedir [37].
- Planlayıcı (Scheduler): Konteynırlar içerisine yüklenmiş uygulamaların hangi düğümde çalışacağını belirleyen bileşendir. Pod'ların durumlarını API sunucunun izleme mekanizmasıyla sürekli olarak takip eder ve pod'un hangi düğümde çalışacağına kullanıcın konfigürasyonlarına ve düğümlerin mevcut kaynaklarına (CPU, hafıza ve disk kapasitesi) göre karar verir. Temel anlamda filtreleme ve puanlama olmak üzere iki işlevi bulunmaktadır. Filtreleme aşamasında uygun düğümler bulunurken puanlama aşamasında kaynak dağılımına göre uygun olan Podlar sıralanarak seçim yapılmaktadır [36].
- Denetleyici (Controller): Kubernetes farklı amaçlar için kullanılan birden fazla denetleyiciye sahiptir. Denetleyiciler Kubernetes kümesinin durumunu anlık takip etmektedir ve gerekli aksiyonların alınmasını sağlamaktadır. Herhangi bir düğümün arızalanması veya benzeri durumlarda kümeyi istenen duruma tekrardan getirmek için gerekli denetleyiciyi tetikleyerek dinamik olarak küme değişikliklerini planlamaktadır [38].
- Bulut Denetleyici Yöneticisi (Cloud Controller Manager): Bu bileşen buluta özgü kontrol mantığını içermektedir. Kümeyi birbirine bulut sağlayıcı API sunucuları ile bağlanmasını sağlar ve bulut sağlayıcı bileşenlerinin arasından sadece kümenin kullanacağı bileşenleri seçer. Ayrıca Google Cloud Platform, AWS Azure ve Rancher gibi platformların sunduğu ekstra özellikleri Kubernetes içerisinde kullanmayı sağlar. [39].
- İşçi Düğümü Bileşenleri (Worker Node Componenets): Konteynırlara yüklenmiş uygulamaların çalıştırılması ve birbirleriyle haberleşerek son kullanıcıya gerekli servisleri sağlayan bileşendir. Tüm işçi düğümleri içerisinde çalışan kubelet ve kube proxy olmak üzere iki temel bileşeni bulunmaktadır [36].



Şekil 2. Sunucuları ve düğümleri içeren test ortamı (Test environment including servers and nodes)

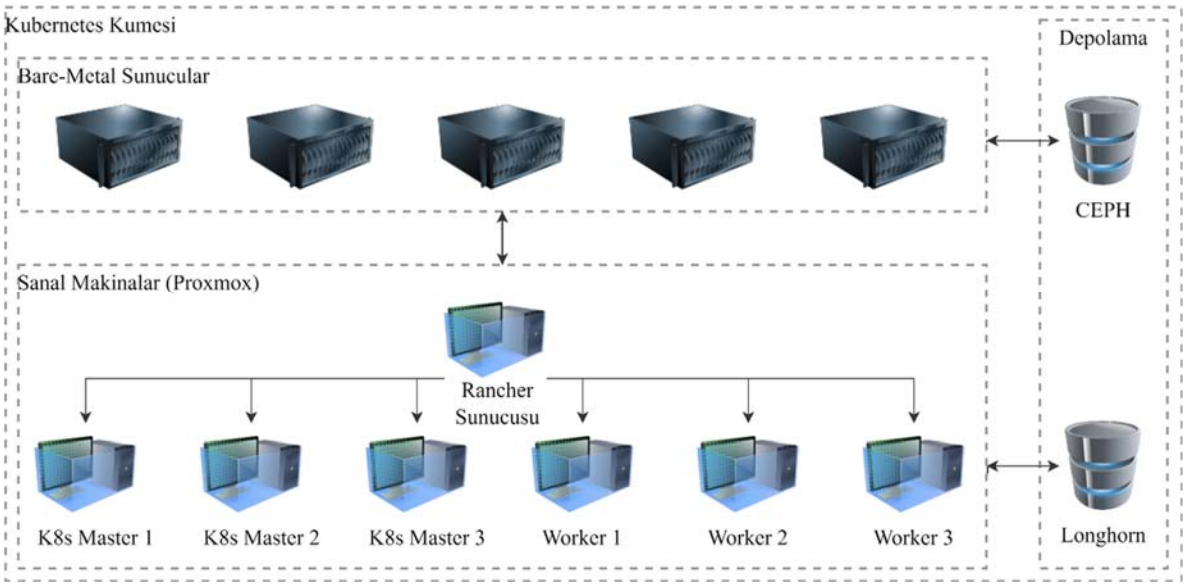
- Kube Proxy: Kümedeki her düğümde çalışan bir vekil sunucudur. Düğümlerdeki ağ kurallarını koruyarak Pod'ların küme içindeki veya dışındaki ağ iletişimine izin verir [37], [40].
- Kubelet: İşçi düğümü içerisinde çalışan bileşenlerin operasyonel işlemlerinden sorumludur. API sunucuda bir düğüm kaynağı oluşturarak üzerinde çalıştığı düğümü kaydettirir. Kendi düğümündeki tüm Pod'lar için API sunucuyu sürekli izler ve Pod bileşeni üzerinde yer alan konteynırları çalıştırır. API sunucudan gelen Pod konfigürasyonlarını ve değişim taleplerini uygular. Aynı zamanda düğümün anlık durumunu API sunucuya geri bildirerek nihai tutarlılığı sağlar [40].

Kubernetes bileşenleri yönetimi için 1 düğüm, Kubernetes ana sunucu işlemleri için 3 düğüm ve Kubernetes işçi işlemleri için 3 düğüm olmak üzere Şekil 3'te de görülebilen Kubernetes kümesi oluşturulmuştur. Kubernetes kümesinin yönetimi Rancher platformu ile gerçekleştirilmiş, kullanıcı istekleri Rancher üzerinden Kubernetes ana düğümlerine iletilmiştir. Benzer olarak ana düğümler de bulut yönetici bileşeni aracılığıyla API Server üzerinden diğer Kubernetes bileşenlerine istek yapmaktadır. Rancher uygulaması ara yüzlerinden küme bileşenleri bulut yöneticisi aracılığı ile yönetilebilmektedir. Kullanıcılar tarafından Kubernetes mimarisine eklenmek istenilen açık kaynak kodlu uygulamalar ve bileşenleri planlayıcı aracılığı ile işçi düğümlerine dağıtılmaktadır. Bu sayede dağıtık şekilde kaynak yönetimi sağlanmaktadır. Bileşenler, Kubernetes kümesi içerisinde birbirleriyle tanımlanan network konfigürasyonları aracılığıyla bağlantı kurmaktadır. Aynı zamanda Kubernetes kümesi içerisinde yer alan uygulamaların birden fazla kopyası(replikası) oluşturularak hataya karşı toleranslı hale getirmektedir. Bu sayede önerilen model hem dağıtık olarak çalışabilmekte hem de hataya karşı toleranslı olarak işlemlerini yürütmektedir. Yani her bir işçi üzerinde açık kaynak kodlu uygulamalardan NIFI, Kafka, Superset, PosgreSQL ve JupyterHub gibi uygulamalar çalışabilmektedir. Bu uygulamaların erişebileceği ortak saklama alanı depolama katmanında yer almaktadır. Kubernetes konteynır içerisinde oluşan veriler kalıcı olmayacak şekilde saklanmaktadır. Bu sebeple konteynırın ihtiyaç duyacağı veya işleme sonucu oluşan verilerin kalıcı olarak saklanması amacı ile volume yapısı kullanılmıştır. Böylelikle her konteynır için volume aracılığı ile kalıcı veri ortamı sağlanmıştır. Önerilen mimari model içerisinde volume yönetimi açık kaynak kodlu Longhorn uygulamasıyla sağlanmıştır. Longhorn uygulamasının blok depolama çözümü sayesinde volume'ler hataya karşı toleranslı olabilmektedir.

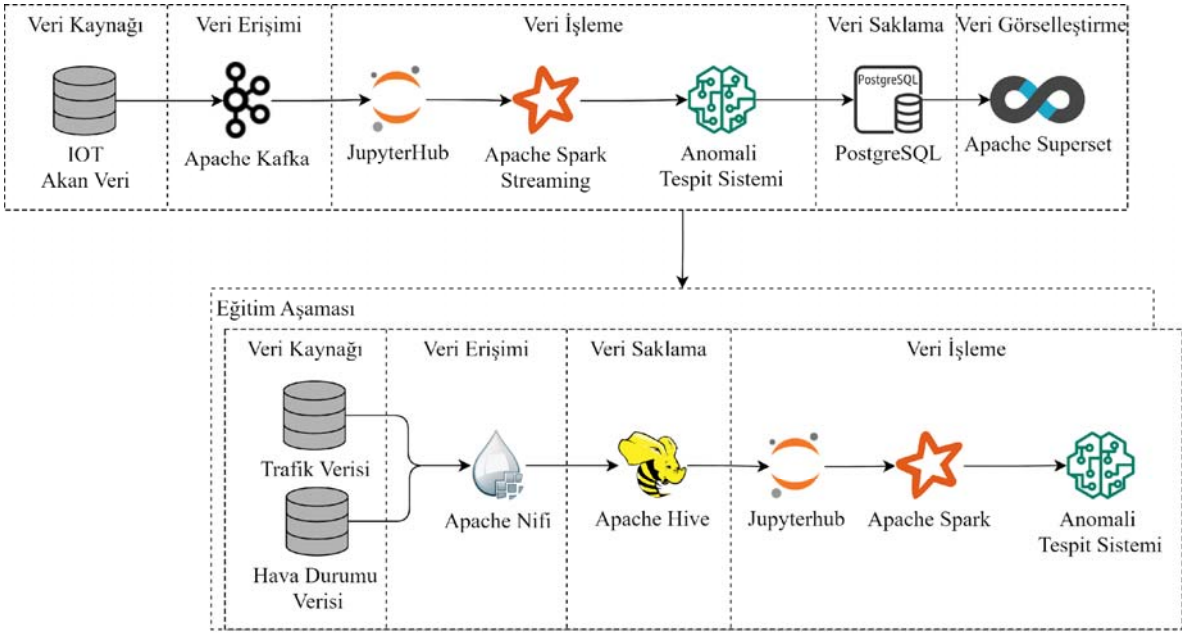
CEPH depolama sistemi ise yatayda ölçeklenebilir bir yapıya sahip olan ve birden fazla diski tek bir disk olarak kullanılmasını sağlayan bir çözümdür. İçerisinde veri tabanı, konteynır imajı, sanal makina, resim, video, kayıt dosyası gibi her türden veri sağlanabilmektedir. Bu sayede yapısal olmayan veriler, medya verisi, IOT telemetri verisi, yedekleme verisi, arşiv verileri depolama gibi konularda çözüm sunabilmektedir. Önerilen mimaride sisteme dâhil edilen sanal makinalar CEPH kullanılarak depolama hususunda yatayda ölçeklenebilir ve hataya karşı toleranslı bir hale getirilmiştir.

Önerilen model üzerinde veri toplama, analiz, kod geliştirme ve görselleştirme işlemleri dağıtık şekilde sağlanabilmektedir. Makine öğrenme algoritmalarının koşturulması amacı ile Apache Spark ortamı kullanılmıştır. Apache Spark uygulaması Kubernetes kümesi üzerinde dağıtık olarak çalışmaktadır. Apache Spark bileşenleri olan Spark Driver ve Spark Executor birden fazla işçi düğümler üzerinde dağıtık şekilde çalışmaktadır. Apache Spark, makine öğrenmesi kodlarının dağıtık olarak çalışması için Spark Driver bileşenini, yapılması istenilen işi birden çok parçaya ayırarak Spark Executor bileşenlerine dağıtır. Bu sayede Spark executor bileşeni kendi düğümü içerisinde verilen işi yerine getirerek dağıtık ve paralel şekilde işlemlerini gerçekleştirir. Spark executor bileşeninde herhangi bir hata oluşması durumunda Kubernetes kümesi içerisinde hatalı pod sonlandırılarak yerine yenisi oluşturulmaktadır. Yeni oluşan pod eskisinin görevlerini devralarak işlemlerine devam etmektedir. Bu bağlamda hatalı pod işlemleri sonucu oluşabilecek yeniden hesaplamaların önüne geçilerek sistem verimliliği artırılmaktadır. Aynı zamanda sistemin hataya karşı toleranslı hale getirilmektedir.

Farklı ortamlarda yer alan sensör verileri Kafka aracılığı ile verilerini yollamaktadır. Bu bağlamda anlık veri akışı işlemleri için Kafka üzerinden gelen veriler JupyterHub platformu aracılığı ile işlenmektedir. Çalışma kapsamında, trafik ve hava verileri kaynaklarının farklı olmaları sebebi ile birleştirme işlemi uygulanmış, NIFI ortamında farklı kaynaklardan gelen veriler okunarak birleştirilmiş ve veri işlenmeye hazır hale getirilmiştir. Birleştirilen veriler dönüştürülerek Hive aracılığı ile saklanmış, saklanan veriler üzerinde derin öğrenme modeli JupyterHub ve Spark aracılığı ile geliştirilerek kullanıma hazır hale geliştirilmiş ve Spark ortamında veriler eğitilerek model oluşturulmuştur. Şekil 4'de veri birleştirme ve akış için oluşturulan model sunulmuştur.



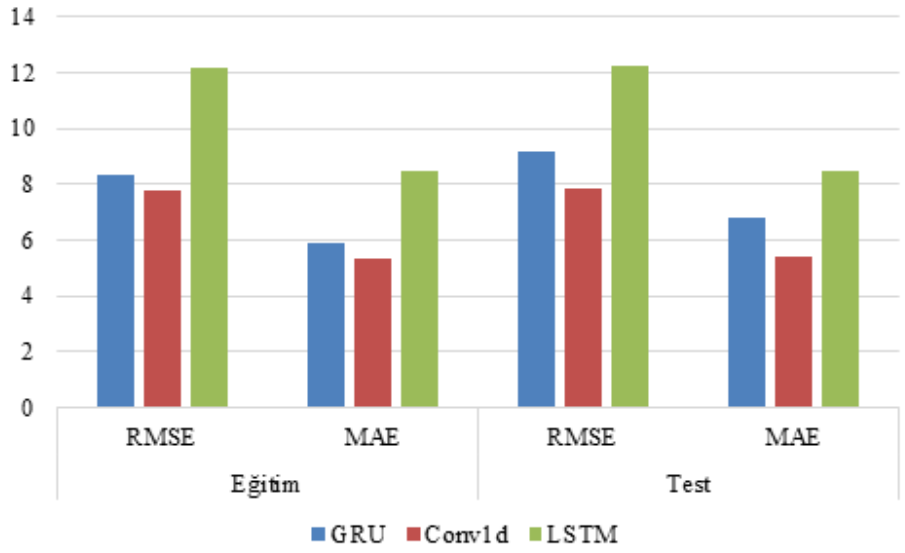
Şekil 3. Önerilen modelin fiziksel mimarisi (The physical architecture of proposed model)



Şekil 4. Veri akış modeli (Data flow model)

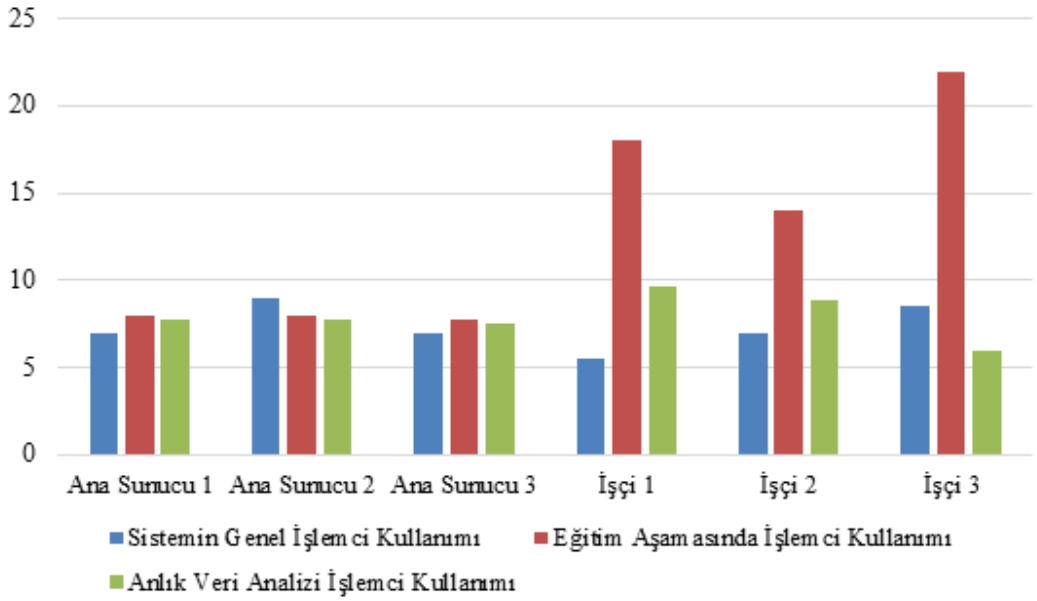
Tablo 5. Model hata değerleri (Model error values)

	Eğitim		Test	
Metot	RMSE	MAE	RMSE	MAE
GRU	8.30	5.87	9.14	6.79
Conv1d	7.80	5.30	7.85	5.37
LSTM	12.14	8.45	12.22	8.44



Şekil 5. Model hata sonuçları (Model error rates)

Önerilen model kapsamında kullanılan derin öğrenme modellerinin anomali tespitine ait sonuçların değerlendirilmesi RMSE ve MAE metrikleri kullanılmıştır. Modelin eğitim aşaması için farklı veri kaynakları olan trafik verileri ve hava verileri büyük veri mimarisi içerisinde NIFI açık kaynak kodlu uygulama aracılığı ile birleştirilmiş ve eğitime hazır hale getirilmiştir. Veri seti eğitim ve test verisi olarak sırasıyla yüzde 80 ve yüzde 20 oranında bölünmüş ve her model eğitildikten sonra hem eğitim hem de test veri seti üzerinden değerlendirilmiştir. Anomali tespiti için LSTM, GRU ve Conv1d algoritmaları kullanılmış, tahmin performansına ilişkin bilgiler Tablo 5'te ve Şekil 5'de sunulmuştur. Sonuçlar değerlendirildiğinde kullanılan algoritmaların tahmin performansı RMSE ve MAE metrikleri ile değerlendirildiğinde; Conv1d yöntemi, RMSE metriğiyle, eğitim ve test verisinde sırasıyla 7.80 ve 7.85 değerleri



Şekil 6. Önerilen kubernetes mimarisi ana ve işçi düğümlerin işlemci kullanım oranları
(CPU usage rates of master and worker nodes in the proposed kubernetes architecture)

üretirken diğer yöntemler arasında en başarılı yöntem olarak öne çıkmıştır. GRU yöntemi her iki hata metrik değerleri ile Conv1d yöntemine yaklaşırken, üç yöntem arasında başarımı en düşük LSTM yöntemi olmuştur.

Modelin sistem üzerinde çalışma etkilerini değerlendirmek amacı ile en iyi başarım sonucuna sahip Conv1d modeli kullanılarak anomali etiketleme işlemleri gerçekleştirilmiştir. Anomali belirleme işlemi için yeniden çatılma hatası kullanılarak belirlenen eşik değeri doğrultusunda değerlendirilmiştir. Anomali değeri, beklenen değer ve gerçek değer arası farkın eşik değerinden büyük olmasına göre karar verilmiştir. Bu çalışma kapsamında anomali eşik değeri, denetimsiz öğrenme yöntemlerinde yoğunluklu olarak kullanılan standart sapmanın 3 katı olarak belirlenmiştir [41]. Belirlenen anomali dikkate alınarak Spark streaming aracılığıyla Jupyter ortamında Kafka dan gelen akış verileri modelden geçirilerek anomali etiketleme işlemi gerçekleştirilmiştir. Anomali olarak etiketlenen veriler POSTGRE veri tabanına aktarılarak SUPERSET aracılığı görselleştirilmiştir.

Kubernetes mimarisinde orkestrasyon esnasında herhangi bir görev ana düğümler aracılığı ile alt işçi düğümlere dağıtılmış, ana ve işçi düğümler kullanılarak küme bileşenleri arası işlemler gerçekleştirilmiştir. Sistemin genel performansını incelemek amacıyla ortalama işlemci kullanım süreleri hesaplanmış, sistemin boş durumdayken işlemci kullanımı, eğitim aşaması işlemci kullanımı ve veri işleme durumundaki işlemci kullanımları incelenmiş ve bu değerler şekil 6'da verilmiştir. Sistemin boş durumdayken genel işlemci kullanımı ortalama %5 ile %10 arasındadır. Modelin eğitim aşamasında sadece ana düğüm 3'te küçük bir artış gözlemlenmektedir. Bu durumun ana düğümlerin sadece kullanıcı isteklerini işleyerek diğer Kubernetes bileşenlerine dağıtımı sağladığı ve bu şekilde işçi düğümleri arası kaynak yönetimini sağladığını göstermektedir. İşçi düğümlerin sistemin boş durumunda çalışma ortalamaların %7 ile %10 arasında olduğu ölçülmüştür. Modelin eğitimi sırasında işçi düğümlerin işlemci kullanım oranları artarak %14 ile %22 arasında değerler almıştır. Ana düğümlere göre işçi düğümlerin eğitim aşamasında işlemci kullanımının fazla olmasının nedeni ana düğümler tarafından işçi düğümlere işler dağıtılarak gerçekleştirilmesidir. Anlık veri analiz aşamasında modelin eğitimi sürecinde ana düğümlerin işlemci kullanım oranı ortalama %7 ile %10 arasında gerçekleşmiştir. İşçi düğümlerin sistemin boş durumundaki işlemci kullanımı daha yüksek olup, eğitim aşamasında bu kullanım oranı daha düşmüştür.

IoT kapsamında kullanılan sensörler ve veri miktarındaki artış, geliştirilen sistemlerin dağıtık ve hata toleranslı olacak şekilde tasarlanmasının gerekliliği ön plana çıkarmaktadır. Bu doğrultuda Kubernetes kümesi içerisinde yer alan uygulamalar replikaları sayesinde çalışmalarını hata toleranslı olarak gerçekleştirmektedir. Herhangi bir işçi düğümün podları işlerini tamamlayamayıp kapanması durumunda, işler diğer podlara devredilerek görevler yürütülmektedir. IoT sensörlerinin dağıtık yapısı ve hızlı veri akışı göz önüne alındığında, önerilen modelin hata bu alanda toleranslı bir şekilde çalışabilmesi ve dağıtık ortamlarda etkin bir şekilde uygulanması kritik bir önem taşımaktadır.

5. Sonuçlar (Conclusions)

Günümüzde sensör miktarının artması ile birlikte birbirine bağlı cihaz sayısı ve üretilen veri miktarı artmıştır. Üretilen büyük hacimde ve miktardaki verinin birlikte analiz edilmesi önem arz etmektedir. Bu nedenle büyük veri konsepti kapsamında geliştirilen çözümler artmaya başlamıştır. Verinin işleme kapasitesinin artırılması amacı ile dağıtık mimariler ön plana çıkmakta ve hata toleranslı sistemler oluşturulmaktadır. Yapılan çalışmada Kubernetes tabanlı büyük veri kapsamında anomali tespit modeli önerilmiştir. Önerilen model genelleştirilmiş bir model olup trafik verileri dikkate alınarak anomali belirleme işlemi yapılmıştır. Modelin ilk aşamasında farklı kaynaklardan gelen veriler birleştirilerek derin öğrenme algoritmaları kullanılarak model geliştirilmiştir. Oluşturulan model büyük veri mimarisi içerisinde veri işleme katmanında kullanılarak anomali belirleme işlemi gerçekleştirilmiştir. Veri işleme aşamasında derin öğrenme algoritmalarından LSTM, GRU ve Conv1d algoritması değerlendirilmiştir. Conv1d algoritması diğer algoritmalara göre daha iyi performans sonuçlarına sahiptir. Anomali belirleme işlemi sonrası anlamlı veriler kalıcı olarak saklanarak görselleştirme işlemleri Superset uygulaması üzerinden yapılmıştır. Önerilen modelin gerçekleştirilmesi amacıyla kullanılan Kubernetes kümesi

analizlerinde sistemin boş, eğitim ve akış verisi analizi durumlarında işlemci kullanımının ana düğümlere etkisinin az olduğu gözlemlenmiştir. Benzer olarak işçi düğümlerin ise işlemci kullanım oranlarının özellikle eğitim aşaması ve akış verisi analizi kısmında yükseldiği görülmektedir. Sistemin dağıtık yapısı sayesinde bu yüklerin işçi düğümler arası paylaşıldığı gözlemlenmiştir. Sonuç olarak önerilen model kapsamında Kubernetes mimarisi kaynakların dağıtık şekilde çalışması ve hataya karşı toleranslı bir anomali tespit sistemi oluşturulması sağlamıştır.

Kaynaklar (References)

1. Habeeb R.A.A., Nasaruddin F., Gani A., Hashem I.A.T., Ahmed E., Imran M., Real-time big data processing for anomaly detection: A survey, Int. J. Inf. Manage., 45, 289-307, 2019.
2. Bakshi K., Considerations for big data: Architecture and approach, 2012 IEEE Aerospace Conference, Big Sky - USA, 1-7, 2012.
3. Morales F.A., Ramírez J.M., Ramos E.A., A mathematical assessment of the isolation random forest method for anomaly detection in big data, Math. Methods Appl. Sci., 46 (1), 1156-1177, 2023.
4. Michalik P., Štofa J., Zolotova I., Concept definition for big data architecture in the education system, 2014 IEEE 12th International Symposium on Applied Machine Intelligence and Informatics (SAMI), Herl'any- Slovakia, 331-334, 2014.
5. Wang J., Yang Y., Wang T., Sherratt R.S., Zhang J., Big data service architecture: A survey, J. Internet Technol., 21 (2), 393-405, 2020.
6. Benhlima L. et al., Big data management for healthcare systems: Architecture, requirements and implementation, Adv. Bioinf., 2018, 2018.
7. Hajjaji Y., Boulila W., Farah I.R., Romdhani I., Hussain A., Big data and IoT-based applications in smart environments: A systematic review, Comput. Sci. Rev., 39, 100318, 2021.
8. Kune R., Konugurthi P.K., Agarwal A., Chillarige R.R., Buyya R., The anatomy of big data computing, Softw. Pract. Exp., 46 (1), 79-105, 2016.
9. Praveena M.A., Bharathi B., A survey paper on big data analytics, 2017 International Conference on Information Communication and Embedded Systems (ICICES), Chennai - India, 1–9, 2017.
10. Ma X., Wu J., Xue S., Yang J., Zhou C., Sheng Q.Z., Xiong H., Akoglu L., A comprehensive survey on graph anomaly detection with deep learning, IEEE Trans. Knowl. Data Eng., 2021.
11. Rettig L., Khayati M., Cudré-Mauroux P., Piórkowski M., Online anomaly detection over big data streams, Appl. Data Sci.: Lessons Learned Data-Driven Bus., 289-312, 2019.
12. Stojanovic L., Dinic M., Stojanovic N., Stojadinovic A., Big-data driven anomaly detection in industry (4.0): An approach and a case study, 2016 IEEE International Conference on Big Data (Big Data), Washington DC - USA, 1647-1652, 2016.
13. Al Jallad K., Aljnidi M., Desouki M.S., Anomaly detection optimization using big data and deep learning to reduce false-positive, J. Big Data, 7 (1), 1-12, 2020.
14. Zhou X., Hu Y., Liang W., Ma J., Jin Q., Variational LSTM enhanced anomaly detection for industrial big data, IEEE Trans. Ind. Inform., 17 (5), 3469-3477, 2020.
15. Parwez M.S., Rawat D.B., Garuba M., Big data analytics for user-activity analysis and user-anomaly detection in mobile wireless network, IEEE Trans. Ind. Inf., 13 (4), 2058–2065, 2017.
16. Karatepe I.A., Zeydan E., Anomaly detection in cellular network data using big data analytics European Wireless 2014; 20th European Wireless Conference, Barcelona - Spain, 1–5, 2014.
17. Senjab K., Abbas S., Ahmed N., Khan A.U.R., A survey of Kubernetes scheduling algorithms, J. Cloud Comput., 12 (1), 87, 2023.
18. Shamim S.I., Gibson J.A., Morrison P., Rahman A., Benefits, challenges, and research topics: A multi-vocal literature review of Kubernetes, arXiv preprint arXiv:2211.07032, 2022.
19. Xu Z., Gong Y., Zhou Y., Bao Q., Qian W., Enhancing Kubernetes automated scheduling with deep learning and reinforcement techniques for large-scale cloud computing optimization, arXiv preprint arXiv:2403.07905, 2024.
20. Dang-Quang N.M., Yoo M., Deep learning-based autoscaling using bidirectional long short-term memory for Kubernetes, Appl. Sci., 11 (9), 3835, 2021.
21. Jung G., Ha H., Lee S., Anomaly detection of facilities and non-disruptive operation of smart factory using Kubernetes, J. Inf. Process. Syst., 17 (6), 2021.
22. Hernández-Rivas A., Morales-Rocha V., Ruiz-Hernández O., Big data platform as a service for anomaly detection, Data Analytics and Computational Intelligence: Novel Models, Algorithms and Applications, Springer Nature, Switzerland, 2023.
23. Muralidharan S., Song G., Ko H., Monitoring and managing IoT applications in smart cities using Kubernetes, Cloud Comput., 11, 2019.
24. Chen Q., He Y., Yu G., Xu C., Liu M., Li Z., KBMP: Kubernetes-orchestrated IoT online battery monitoring platform, IEEE Internet Things J.,11 (14) 2024.
25. I.C.P. Dataset. http://iot.ee.surrey.ac.uk:8080/. Erişim tarihi Mayıs 17, 2023.
26. Ali M.I., Gao F., Mileo A., Citybench: A configurable benchmark to evaluate RSP engines using smart city datasets, The Semantic Web-ISWC 2015: 14th Int. Semantic Web Conf., Springer, 374-389, 2015.
27. Smagulova K., James A.P., A survey on LSTM memristive neural network architectures and applications, Eur. Phys. J. Spec. Top., 228 (10), 2313-2324, 2019.
28. Akin M., Sagiroglu S., Short term traffic speed prediction with RNN method for roads characterized by density based clustering method, Journal of the Faculty of Engineering and Architecture of Gazi University, 37 (2), 581-593, 2022.
29. Wang Y., A new concept using LSTM neural networks for dynamic system identification, 2017 American Control Conference (ACC), Seattle - USA, 5324–5329, 2017.
30. Cho K., Van Merriënboer B., Gulcehre C., Bahdanau D., Bougareş F., Schwenk H., Bengio Y., Learning phrase representations using RNN encoder-decoder for statistical machine translation, Empirical Methods Nat. Lang. Process. (EMNLP), 2014.
31. Chung J., Gulcehre C., Cho K., Bengio Y., Empirical evaluation of gated recurrent neural networks on sequence modeling, arXiv preprint arXiv:1412.3555, 2014.
32. Lawal A., Rehman S., Alhems L.M., Alam M.M., Wind speed prediction using hybrid 1D CNN and BLSTM network, IEEE Access, 9, 156672-156679, 2021.
33. Xiao L., Zhang L., Niu F., Su X., Song W., Remaining useful life prediction of wind turbine generator based on 1D-CNN and Bi-LSTM, Int. J. Fatigue, 163, 107051, 2022.
34. Kiranyaz S., Avci O., Abdeljaber O., Ince T., Gabbouj M., Inman D.J., 1D convolutional neural networks and applications: A survey, Mech. Syst. Signal Process., 151, 107398, 2021.
35. Wnęk K., Boryło P., A data processing and distribution system based on Apache Nifi, Photonics, 10 (2), 210, 2023.
36. Manaouil K., Lebre A., Kubernetes and the edge?, Doktora Tezi, Inria Rennes-Bretagne Atlantique, 2020.
37. Jeffery A., Howard H., Mortier R., Rearchitecting Kubernetes for the edge, Proceedings of the 4th International Workshop on Edge Systems, Analytics and Networking (EdgeSys '21), New York -USA, 7-12, 2021.
38. Vayghan L.A., Saied M.A., Toeroe M., Khendek F., A Kubernetes controller for managing the availability of elastic microservice-based stateful applications, J. Syst. Softw., 175, 110924, 2021.
39. Kubernetes Documentation-Cloud Controller Manager. http://web.archive.org/web/20230604021425/https://kubernetes.io/docs/concepts/architecture/cloud-controller/. Erişim tarihi Haziran 12, 2023.
40. Johansson B., Rågberger M., Nolte T., Papadopoulos A.V., Kubernetes orchestration of high availability distributed control system, 2022 IEEE International Conference on Industrial Technology (ICIT), Shanghai - China, 1-8, 2022.
41. Braei M., Wagner S., Anomaly detection in univariate time-series: A survey on the state-of-the-art, arXiv preprint arXiv:2004.00433, 2020.