KAPALI DÖNGÜ DAĞITIM SİSTEMLERİNDE KESTİRİMCİ BAKIM VE HATA ANALİZİ

Necati Barış TAŞ

T.C.
BURSA ULUDAĞ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

KAPALI DÖNGÜ DAĞITIM SİSTEMLERİNDE KESTİRİMCİ BAKIM VE HATA ANALİZİ

Necati Barış TAŞ

502205031

0009-0001-1833-8048

Doç. Dr. Arif BAŞGÜMÜŞ
(Danışman)
Doç. Dr. Mustafa Namdar
(Danışman)

YÜKSEK LİSANS TEZİ
ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

BURSA – 2025
Her Hakkı Saklıdır

ÖZET

Yüksek Lisans Tezi

KAPALI DÖNGÜ DAĞITIM SİSTEMLERİNDE KESTİRİMCİ BAKIM VE HATA ANALİZİ

Necati Barış TAŞ

Bursa Uludağ Üniversitesi
Fen Bilimleri Enstitüsü
Elektronik Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Arif BAŞGÜMÜŞ
Eş Danışman: Doç. Dr. Mustafa NAMDAR

Bu tez çalışmasında gelişen e-ticaret sektörü ile dağıtım merkezlerinde ilerleyen otomasyon teknolojileri ve getirdiği büyük veri üzerinde istatistiksel metotlar, makine öğrenmesi ve derin öğrenme algoritmaları ile kestirimci bakım yöntemi çalışılmıştır. Yüksek verimlilik hedefleri, gelişen teknoloji ile kompleks hale gelen sistemler ve artan ekipman sayısı sebebiyle bakım süreçleri de verimliliğini arttırmak zorunda kalmıştır. Bu sebeple veri biliminin bakım süreçlerine entegre edilmesi zorunlu hale gelmiştir. Ancak sistemlerin ürettiği verilerin analize hazır olması için ara sistemler ve yazılımların devreye alınması gerekmektedir.

Bu çalışmada verinin toplanması ve analiz için hazırlanması adımlarında uygulamaya özel yazılımlar geliştirilmiştir. Toplanan veriler içinde motorların hedeflenen ve anlık hızları arasındaki sapma değerleri ile oluşturulan zaman serilerinde SARIMA, Gradient Boosting, XGBoost, LSTM ve Prophet algoritmaları ile gelecek değer tahmini yapılarak gerçekleşen değer ile karşılaştırılmıştır. Sonuçlarda RMSE ve MAE parametreleri kullanılarak algoritmaların performans karşılaştırılması yapılmış, algoritmaların avantaj ve dezavantajlarına değinilmiştir. Sonuçlar ile faydalı olabilecek algoritmalar ve olası senaryolar belirlenmiştir.

Anahtar Kelimeler: Kestirimci Bakım, Makine Öğrenmesi, Derin Öğrenme, Ayrıştırma/Dağıtım Sistemleri

2025, xv + 51 sayfa.

ABSTRACT

MSc Thesis

PREDICTIVE MAINTENANCE AND FAULT ANALYSIS IN CLOSED-LOOP SORTATION SYSTEMS

Necati Barış TAŞ

Bursa Uludağ University
Graduate School of Natural and Applied Sciences
Department of Electronics Engineering

Supervisor: Assoc. Prof. Dr. Arif BAŞGÜMÜŞ
Co-Supervisor: Assoc. Prof. Dr. Mustafa NAMDAR

In this thesis study, predictive maintenance methods are explored using statistical methods, machine learning, and deep learning algorithms on the large-scale data generated by evolving automation technologies in distribution centers and the growing e-commerce sector. Due to increasing efficiency goals, the complexity of systems driven by advancing technology, and the rising number of equipment, maintenance processes are also required to improve their effectiveness. As a result, integrating data science into maintenance operations has become essential. However, for system-generated data to be ready for analysis, intermediate systems and software tools must be implemented.

In this study, custom software applications were developed for data collection and preprocessing specific to the application context. Using the time series generated from the deviation between the target and actual speeds of motors, future values were predicted using SARIMA, Gradient Boosting, XGBoost, LSTM, and Prophet algorithms, and these predictions were compared with the actual values. The performance of the algorithms was evaluated using RMSE and MAE metrics, and the advantages and disadvantages of each algorithm were discussed. Based on the results, potentially useful algorithms and applicable scenarios were identified.

Key words: Predictive Maintenance, Machine Learning, Deep Learning, Loop Sorter

2025, xv + 51 pages.

TEŞEKKÜR

Bu tez çalışması için hiçbir desteğini esirgemeyen tez danışmanlarım Doç. Dr. Arif BAŞGÜMÜŞ ve Doç. Dr. Mustafa NAMDAR'a teşekkürlerimi sunarım.

Ayrıca tez çalışmalarım sırasında her zaman arkamda olan ve manevi destek sağlayan eşim Ezgi İrem Bektaş Taş'a teşekkürlerimi sunarım.

Tez çalışmam için beni destekleyen ve veri kaynaklarına erişimimi sağlayan işverenim Vanderlande Industries BV'e teşekkürlerimi sunarım.

Necati Barış TAŞ
11/06/2025

İÇİNDEKİLER

Sayfa

ÖZET viii
ABSTRACT ix
TEŞEKKÜR x
SİMGELER ve KISALTMALAR DİZİNİ xii
ŞEKİLLER DİZİNİ xiv
ÇİZELGELER DİZİNİ xv
1. GİRİŞ 1
1.1. Sistem Bilgileri 4
2. KURAMSAL TEMELLER ve KAYNAK ARAŞTIRMASI 6
2.1. Bakım Türleri ve Özellikleri 6
2.1.1. Düzeltici Bakım (Corrective Maintenance) 6
2.1.2. Önleyici Bakım 6
2.1.3. Kestirimci Bakım (Predictive Maintenance) 7
2.1.4. Karşılaştırmalı Analiz 8
2.1.5. Kestirimci Bakım Makine Öğrenmesi Algoritmaları 9
2.2. Zaman Serileri 11
2.2.1. SARIMA 11
2.2.2. Gradient Boosting Regressor (GBR) 12
2.2.3. XGBoost (Extreme Gradient Boosting) 13
2.2.4. LSTM (Long Short-Term Memory) 15
2.2.5. Prophet 16
3. MATERYAL ve YÖNTEM 18
3.1. Sistem Verilerinin İncelenmesi 19
3.2. Verilerin Hazırlanması 19
3.3. Verilerin Analizi 23
3.3.1. Analiz Sonuçları 31
4. BULGULAR 37
4.1. Genel Performans Analizi 37
5. TARTIŞMA ve SONUÇ 47
KAYNAKLAR 49
ÖZGEÇMİŞ 51

SİMGELER ve KISALTMALAR DİZİNİ

Simgeler	Açıklama
y_t	t anındaki gözlenen değer (zaman serisi değeri)
$\hat{y}_t$	t anı için tahmin edilen değer
e_t	Hata terimi ($e_t = y_t - \hat{y}_t$)
ϕ	Otoregresif (AR) katsayısı
θ	Hareketli Ortalama (MA) katsayısı
d	Fark alma derecesi (differencing order)
s	Mevsimsel periyot (seasonal period)
p	AR teriminin sırası
q	MA teriminin sırası
P	Mevsimsel AR sırası
D	Mevsimsel fark alma derecesi
Q	Mevsimsel MA sırası
δ	Değişim noktalarında eğim farkı (trend değişimi)
ν	Öğrenme oranı (learning rate)
h_t	LSTM hücre durumundaki bilgi
f_t	Forget gate (Unutma kapısı) çıktısı
i_t	Input gate (Girdi kapısı) çıktısı
o_t	Output gate (Çıktı kapısı) çıktısı
k	Lineer trend eğimi (Prophet modeli)
m	Lineer trend başlangıç seviyesi

Kısaltmalar	Açıklama
RMFS	Robotik Mobil Sipariş Karşılama Sistemi
AS/RS	Otomatik Depolama ve Geri Alma Sistemi
AGV	Otomatik Yönlendirmeli Araç
AMR	Otonom Mobil Robot
WMS	Depo Yönetim Sistemi
MTBF	Arızalar Arası Ortalama Süre
RUL	Kalan Faydalı Ömür
IoT	Nesnelerin İnterneti
PdM	Kestirimci (Tahmine Dayalı) Bakım
SVM	Destek Vektör Makineleri
ANN	Yapay Sinir Ağı
KNN	K-En Yakın Komşu Algoritması
ARIMA	Otoregresif Tümleşik Hareketli Ortalama (Zaman Serisi Modeli)
LSTM	Uzun-Kısa Süreli Bellek
AR	Otoregresif
MA	Hareketli Ortalama
GBR	Gradyan Artırma Regresyonu
RNN	Tekrarlayan Yapay Sinir Ağı
PPI	Darbe Konum Göstergesi
RMSE	Kök Ortalama Kare Hata
MAE	Ortalama Mutlak Hata

ŞEKİLLER DİZİNİ

		Sayfa
Şekil 1.1.	Kapalı Döngü Dağıtım Sistemi	5
Şekil 3.1.	Darbe Konum Göstergesi Genel Yapısı	18
Şekil 3.2.	Motor Hareketi ile Üretilen Sinyal Yapısı	18
Şekil 3.3.	Motor Verisi İçin Filtrelenmiş Ham Veri	19
Şekil 3.4.	Veri Tabanı Tablosu	20
Şekil 3.5.	Veri Hazırlığı Ana Algoritma Kısım 1	21
Şekil 3.6.	Veri Hazırlığı Ana Algoritma Kısım 2	21
Şekil 3.7.	SQL Veri Tabanında Hazırlanmış Veri	22
Şekil 3.8.	Veri Satır Sayısının Motorlara Dağılımı	23
Şekil 3.9.	Colab Dosya Yükleme	25
Şekil 3.10.	Motor 8800.83.06 Analiz Sonuçları	32
Şekil 3.11.	Motor 8800.85.04 Analiz Sonuçları	33
Şekil 3.12.	Motor 8800.87.06 Analiz Sonuçları	34
Şekil 3.13.	Motor 8800.85.07 Analiz Sonuçları	35
Şekil 3.14.	Motor 8800.83.05 Analiz Sonuçları	36
Şekil 4.1.	Motor 8800.83.06 Tahmin Verileri	39
Şekil 4.2.	Motor 8800.85.04 Tahmin Verileri	40
Şekil 4.3.	Motor 8800.87.06 Tahmin Verileri	42
Şekil 4.4.	Motor 8800.85.07 Tahmin Verileri	43
Şekil 4.5.	Motor 8800.83.05 Tahmin Verileri	45

ÇİZELGELER DİZİNİ

		Sayfa
Çizelge 2.1.	Bakım Türleri Karşılaştırması	8
Çizelge 2.2.	Kestirimci Bakım Amaçlı Kullanılan Algoritmalar	10
Çizelge 2.3.	Algoritma Karşılaştırma Tablosu	17
Çizelge 3.1.	Motor 8800.83.06 RMSE ve MAE Değerleri	32
Çizelge 3.2.	Motor 8800.85.04 RMSE ve MAE Değerleri	33
Çizelge 3.3.	Motor 8800.87.06 RMSE ve MAE Değerleri	34
Çizelge 3.4.	Motor 8800.85.07 RMSE ve MAE Değerleri	35
Çizelge 3.5.	Motor 8800.83.05 RMSE ve MAE Değerleri	36
Çizelge 4.1.	Algoritma Performans Karşılaştırması	37
Çizelge 4.2.	Algoritma Tahminleri	39
Çizelge 4.3.	Algoritma Süre Karşılaştırması	40
Çizelge 4.4.	Algoritma Tahminleri	41
Çizelge 4.5.	Algoritma Süre Karşılaştırması	41
Çizelge 4.6.	Algoritma Tahminleri	42
Çizelge 4.7.	Algoritma Süre Karşılaştırması	43
Çizelge 4.8.	Algoritma Tahminleri	44
Çizelge 4.9.	Algoritma Süre Karşılaştırması	44
Çizelge 4.10.	Algoritma Tahminleri	45
Çizelge 4.11.	Algoritma Süre Karşılaştırması	46

1. GİRİŞ

Son yıllarda küresel ölçekte e-ticarette yaşanan hızlı büyüme, sipariş karşılama (fulfillment) süreçlerinin etkinliğini kritik hale getirmiştir. Özellikle 2020 sonrası dönemde artan çevrimiçi talep, depo operasyonlarında hız ve doğruluk beklentilerini yükseltmiştir. Bu durum, işletmeleri rekabetçi kalabilmek için otomasyon ve ileri teknolojilere yöneltmiştir. Nitekim akademik literatürde de e-ticaret lojistiğinde verimliliği artırmak amacıyla depo otomasyonu ve dijital dönüşüm konularına yoğun ilgi gösterilmektedir (Li vd., 2022; Zhen ve Li, 2022).

E-ticaret sipariş karşılama süreçlerinde otomasyonun en belirgin katkılarından biri operasyonel verimlilik artışıdır. Depo içi otomasyon, sipariş işleme hızını yükseltmekte ve hata oranlarını azaltmaktadır. Örneğin, robotik depo sistemleri sayesinde insan iş gücünün yürüme mesafesi ve elleçleme süreleri önemli ölçüde kısalmakta, böylece birim zamanda daha fazla sipariş doğru şekilde işlenebilmektedir. "Parçadan insana" prensibiyle çalışan otonom mobil robotlar (Robotic Mobile Fulfillment Systems, RMFS) ve otomatik depolama/geri çağırma sistemleri (AS/RS), depo işlemlerinde kesintisiz bir akış sağlayarak verimliliği maksimize etmektedir. Robot destekli sipariş toplama sistemlerinin uygun planlama ve koordinasyon algoritmalarıyla sipariş karşılama sürelerini belirgin biçimde kısalttığı vurgulanmıştır (Li vd., 2022). Otomasyonun sağladığı hız ve doğruluk artışı, operasyonel maliyetleri düşürürken aynı zamanda çalışanların iş yükünü de hafifletmektedir. Bunun yanında dijital teknolojilerin entegrasyonu sayesinde gerçek zamanlı envanter takibi ve şeffaf bilgi akışı mümkün olmakta, tedarik zinciri genelinde süreçler daha sıkı koordine edilebilmektedir (Tubis ve Rohman, 2023). Bütün bu iyileştirmeler, sonuç olarak müşteriye daha hızlı, hatasız ve güvenilir hizmet sunulmasını sağlamaktadır.

Otomasyonun müşteri deneyimine etkileri de en az operasyonel kazanımlar kadar önemlidir. E-ticaret müşterileri, siparişlerinin hızla hazırlanmasını ve vaat edilen sürelerde eksiksiz teslim edilmesini beklemektedir. Depo otomasyonu, siparişlerin daha kısa sürede kargoya verilmesine olanak tanıyarak teslimat sürelerini kısaltır ve bu da müşteri memnuniyetini artırır (Li vd., 2022). Örneğin, büyük e-ticaret şirketleri (Amazon,

JD.com vb.), yüksek otomasyon düzeyine sahip akıllı depolar kurarak hem operasyonel mükemmeliyet sağlamakta hem de müşteri beklentilerini karşılamaktadır. Otomasyon sayesinde sipariş hata oranlarının düşmesi de müşteri deneyimini iyileştiren bir unsurdur; yanlış veya eksik gönderimlerin asgariye inmesi, iade ve şikâyetlerin azalmasını sağlar. Ayrıca, bütünleşmiş bilgi sistemleri aracılığıyla müşterilere siparişlerinin durumu hakkında anlık bildirimler (ör. paketleme, kargoya verilme, teslimat takibi) iletilebilmekte, bu da güven duygusunu pekiştirmektedir. Kısacası, otomasyon sadece işletme içi verimliliği değil, dış müşteri memnuniyetini ve sadakatini de olumlu yönde etkilemektedir (Li vd., 2022; Tubis ve Rohman, 2023).

E-ticaret fulfillment alanında kullanılan otomasyon teknolojileri hem fiziksel otomasyon ekipmanlarını hem de dijital bilgi sistemlerini kapsamaktadır. Fiziksel otomasyonda öne çıkan teknolojiler arasında robotik sistemler ilk sıradadır. Otomatik yönlendirilen araçlar ve otonom mobil robotlar (AGV/AMR), depolardaki rafları veya ürün konteynerlerini alıp toplayıcı istasyonlara taşıyarak "üründen insana" (parts-to-picker) iş akışını gerçekleştirmektedir. Bu yaklaşım, Amazon'un Kiva robotları ile endüstride yaygınlaşmış ve manüel yürüyüş mesafesini ortadan kaldırarak sipariş toplama hızını dramatik şekilde artırmıştır (Aldarondo ve Bozer, 2022). Benzer şekilde, AS/RS sistemleri dikey depolama raflarında robotik asansörler kullanarak ürünleri otomatik olarak depolamakta ve gerektiğinde hızlıca getirmektedir. Bu tip robotik çözümler, depo alanının etkin kullanımını sağlarken insan hatalarını da en aza indirmektedir (Zhen ve Li, 2022). Konveyör bantları ve otomatik ayırma (sortation) sistemleri de yüksek hacimli siparişlerde ürünlerin ayrıştırılması ve paketleme istasyonlarına yönlendirilmesinde yaygın biçimde kullanılmaktadır. Bunların yanı sıra bazı depolar, envanter sayımı veya acil küçük paket ulaştırma gibi görevler için drone teknolojisini denemektedir (deneysel uygulamalar olarak). Genel olarak, fiziksel otomasyon teknolojileri depo içi işlemlerde hız, kapasite ve tutarlılık sağlama hedefiyle devreye alınmaktadır (Tubis ve Rohman, 2023).

Dijital teknolojiler cephesinde, Nesnelerin İnterneti (IoT) ve sensör ağları, e-ticaret depolarının vazgeçilmez unsurları haline gelmiştir. RFID etiketleri, barkodlar ve sensörler aracılığıyla ürünler ve siparişler gerçek zamanlı olarak izlenebilmekte, böylece

envanter görünürlüğü artmakta ve sistem genelinde anlık veri akışı sağlanmaktadır. Bu veriler, gelişmiş Depo Yönetim Sistemleri (WMS) ve diğer bilgi sistemleri aracılığıyla toplanarak büyük veri analitiği ve yapay zekâ uygulamalarına girdi oluşturmaktadır (Tubis ve Rohman, 2023; Zheng vd., 2020).

Gelişen otomasyon sistemleri ekipman sayısını, çeşidini ve karmaşıklığını artırmıştır. Yeni nesil sistemler, sensörlerden sağlanan verilerle büyük ölçekli veri setleri üretilmesine ve bakım süreçlerine yapay zekâ, makine öğrenmesi ve derin öğrenme tabanlı yaklaşımların entegrasyonuna zemin hazırlamaktadır. Bu dijitalleşme süreci, bakım operasyonlarının proaktif hale getirilmesine ve sistem güvenilirliğinin artırılmasına yönelik üst düzey içgörüler sağlamaktadır. Özellikle Nesnelerin İnterneti (IoT) altyapıları ve bağlı sensör sistemleri, gerçek zamanlı veri akışını mümkün kılmakta; böylece çevre koşullarındaki değişimler, titreşim, sıcaklık ve enerji tüketimi gibi parametrelerdeki anomaliler anlık olarak izlenebilmekte ve analiz edilebilmektedir (Ucar vd., 2024; Kvak ve Straka, 2024).

IoT sensörleri ve diğer kaynaklardan elde edilen büyük veri setleri, gelişmiş veri analitiği ve yapay zekâ yöntemleriyle işlenerek arıza öngörüsü modellerinde kullanılmaktadır. Bu veriler üzerinden makine öğrenmesi ve derin öğrenme modelleri eğitilerek kalan ömür tahminleri (RUL) ve anomali tespitleri gerçekleştirilmektedir; bu da ekipman arızalarının önceden tahmin edilerek plansız duruşların önemli ölçüde azaltılmasını sağlamaktadır. Bu sayede bakım stratejileri daha etkin planlanmakta, birim zaman başına operasyonel güvenilirlik yükselmekte ve hatalı bakım uygulamalarından kaynaklanan maliyetler azaltılmaktadır (Ucar vd., 2024; Kvak ve Straka, 2024).

1.1. Sistem Bilgileri

Bu bölümde üzerinde çalıştığımız kapalı döngü dağıtım/ayrıştırma (Loop Sorter) tanıtılacaktır.

Dağıtım/ayrıştırma sistemi temel olarak 3 bölümden oluşmaktadır;

- **Paket Besleme (Infeed)**

  Bu bölüm sistemi diğer sistemlerle bağlayan sabit hızlı giriş konveyörleri ve sonrasında paket boyutlarının ölçümünü sağlayan sensör bariyeri ve devamında değişken hızlı konveyörlerden oluşmaktadır. Bu bölümün temel amacı paketi döngü içerisindeki müsait alana aktarmaktır. Bu bölümde sabit hızlı 6, değişken hızlı 5 adet konveyör bulunmaktadır.

  Alt Komponentler;

  - Motor & Sürücü
  - Encoder
  - Reflektörlü sensörler
  - Işık bariyeri

- **Ana Konveyör (Loop)**

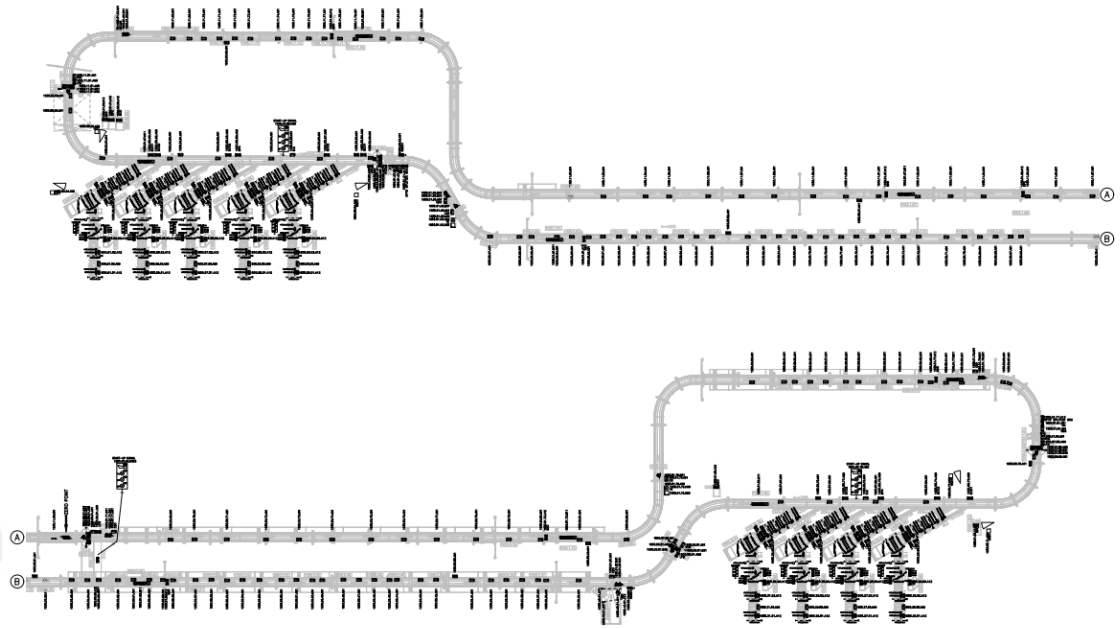
  Ana konveyör 8-10 arası sürücü ile kontrol edilen lineer motorlardan oluşur. Maksimum kapasitede 1000-1400 arası paket taşıyabilir ve 15000 paket/saat kapasiteye ulaşabilir.

  Alt Komponentler;

  - Motor & Sürücü
  - Encoder
  - Reflektörlü Sensörler
  - Boyut ölçüm kamerası
  - Lazer
  - Barkod okuyucu

- Çıkış Konveyörü (Chute)

  Paketlerin ayrıştırıldığı ve sistemden çıktığı bölümdür.



Şekil 1.1. Kapalı Döngü Dağıtım Sistemi

2. KURAMSAL TEMELLER ve KAYNAK ARAŞTIRMASI

2.1. Bakım Türleri ve Özellikleri

2.1.1. Düzeltici Bakım (Corrective Maintenance)

Düzeltici bakım (reaktif bakım) bir arıza meydana geldikten sonra yapılan onarım ve bakım faaliyetlerini ifade eder. Bu yaklaşımda ekipman, bozulana kadar işletilmeye devam eder ve bakım ihtiyacı ortaya çıktığında müdahale edilir. *Run-to-failure* olarak da bilinen bu strateji, kısa vadede bakım maliyetlerini en aza indirse de beklenmedik arızalar nedeniyle uzun duruş sürelerine ve buna bağlı üretim kayıplarına yol açabilir. Bu nedenle düzeltici bakım genellikle kritik olmayan ve kolay tamir edilebilen ekipmanlar için uygun görülür; hayati sistemlerde kullanımı ise plansız duruşlar ve ikincil hasarlar nedeniyle risklidir. Örneğin, bir aydınlatma ampulünün ancak patladıktan sonra değiştirilmesi, kritik olmadığı için düzeltici bakıma uygun bir yaklaşımdır (Molęda vd., 2023; Fausing Olesen ve Shaker ,2020).

2.1.2. Önleyici Bakım

Önleyici bakım (proaktif bakım), arızaları henüz meydana gelmeden önlemek amacıyla ekipmanların belirli periyotlarla planlı bakım ve kontrollerinin yapılmasıdır. Bu yöntemde bakım zamanları üretici tavsiyeleri, tecrübe veya MTBF (Arızalar Arası Ortalama Süre) gibi istatistiksel veriler temel alınarak önceden belirlenir. Örneğin, bir otomobilin her 10.000 km'de bir periyodik bakıma alınması önleyici bakım stratejisine tipik bir örnektir. Düzenli yağlama, ayarlama, parça değişimi ve muayene gibi işlemlerle ekipmanların daima iyi durumda kalması ve beklenmedik duruşların azaltılması hedeflenir. Planlı bakım sayesinde ekipmanın ömrü uzatılabilir ve arıza riski düşürülebilir; ancak her türlü arızayı bütünüyle engellemek mümkün değildir. Kapsam dışı kalan bileşenler yine arızalanabilir ve gereğinden sık parça değişimi, yeni parçaların gizli kusurları veya montaj hataları sebebiyle beklenmedik sorunlara yol açabilir. Ayrıca, önleyici bakımın bir dezavantajı, bakım sürelerinin üretim programına göre planlanması gerekliliği ve bunun getirdiği maliyetlerdir; bu nedenle bakım aralıklarını üretim

takvimiyle optimize etmek önem kazanır (Molęda vd., 2023; Fausing Olesen ve Shaker ,2020).

2.1.3. Kestirimci Bakım (Predictive Maintenance)

Kestirimci bakım (tahmine dayalı bakım), ekipmanların durumunu sürekli izleyip veri analizi yaparak arızalar meydana gelmeden hemen önce bakımın gerçekleştirilmesini amaçlar. Bu yaklaşım, koşul tabanlı bakımın bir alt türü olarak, makine sağlığının gerçek zamanlı izlenmesi ve geçmiş performans verilerine dayanarak gelecekteki arızaların oluşacağı zamanın öngörülmesi prensibine dayanır. Böylece bakım faaliyetleri "ihtiyaç anında" en uygun zamanda yapılır ve bu sayede hem planlı duruş süreleri hem de beklenmedik arıza duruşları en aza indirilebilir (Molęda vd., 2023; Zonta vd., 2020).

Kestirimci bakım uygulamalarında titreşim analizi, termal kamera ile sıcaklık izleme (termografi), yağ analizi, akustik sensörlerle ses takibi ve elektrik motoru akımının izlenmesi gibi zarar vermeyen durum izleme tekniklerinden yararlanılır. Bu sensör verileri, istatistiksel yöntemler veya yapay zekâ destekli algoritmalar ile işlenerek ekipmanın kalan kullanım ömrü (Remaining Useful Life, RUL) tahmin edilir ve bakım ihtiyacının ne zaman kritik hale geleceği belirlenir (Rigas vd., 2024). Örneğin, bir makinenin titreşim verilerindeki normalin üzerindeki artış eğilimi sayesinde rulman arızası oluşmadan önce tespit edilip ilgili parça planlı olarak değiştirilebilir. Kestirimci bakım, enerji üretimi, havacılık ve imalat gibi güvenilirliğin kritik olduğu sektörlerde de yaygın olarak uygulanmaya başlamıştır (Ersöz vd., 2022; Molęda vd., 2023; Fausing Olesen ve Shaker ,2020).

Son yıllarda Endüstri 4.0 ile nesnelerin interneti (IoT) cihazları ve makine öğrenimi algoritmaları, kestirimci bakımın merkezinde yer almaya başlamıştır. Büyük veri analitiği ve derin öğrenme teknikleri, geçmiş arıza kayıtlarından öğrenerek gelecekteki arızaları öngörmeyi ve ekipmanlardaki anormallikleri gerçek zamanlı tespit etmeyi mümkün kılmaktadır. Literatürde kestirimci bakım için yapay sinir ağları, karar ağaçları ve diğer makine öğrenimi modellerinin arıza teşhisi ile kalan ömür tahmininde başarıyla uygulandığı bildirilmektedir (Carvalho vd., 2019; Ersöz vd., 2022).

Kestirimci bakımın sağladığı başlıca kazanımlar arasında beklenmedik duruşların en aza indirilmesi, ekipman verimliliğinin artırılması, gereksiz bakım maliyetlerinin düşürülmesi, üretim süresinin maksimize edilmesi ve iş güvenliğinin iyileştirilmesi bulunmaktadır (Ersöz vd., 2022; Vita vd., 2023).

2.1.4. Karşılaştırmalı Analiz

Aşağıdaki tablo, bakım türlerinin temel kriterler üzerinden karşılaştırmasını sunmaktadır.

Çizelge 2.1. Bakım Türleri Karşılaştırması

Kriter	Düzeltici Bakım	Önleyici Bakım	Kestirimci Bakım
Tanım	Arıza sonrası müdahale	Zaman temelli planlı bakım	Veri temelli bakım (durum izleme)
Veri Gereksinimi	Düşük	Orta	Yüksek (sensör, analiz)
Maliyet (ilk yatırım)	Düşük	Orta	Yüksek
Uzun Vadeli Maliyet	Yüksek	Orta	Düşük
Sistem Kullanılabilirliği	Düşük	Orta	Yüksek
Arıza Önleme Etkisi	Düşük	Orta	Yüksek
Planlama Gereksinimi	Yok	Takvim tabanlı	Dinamik, veri tabanlı

Bu karşılaştırma, kestirimci bakımın uzun vadede daha sürdürülebilir ve verimli bir yaklaşım sunduğunu ortaya koymaktadır. Ancak bu yöntemin uygulanabilirliği; veri altyapısı, kurumsal yetkinlik ve ilk yatırım kapasitesine bağlıdır.

Düzeltici, önleyici ve kestirimci bakım türleri, farklı operasyonel ihtiyaçlara ve risk profillerine göre seçilmektedir. Modern endüstride, özellikle veri analitiği ve yapay zekâ teknolojilerinin gelişmesiyle birlikte, kestirimci bakım stratejileri ön plana çıkmaktadır. Bununla birlikte, her sektör ve sistem için en uygun bakım stratejisinin, ekipman türü,

üretim hacmi, maliyet yapısı ve risk toleransı gibi faktörler göz önünde bulundurularak belirlenmesi kritik öneme sahiptir (Serradilla vd., 2022; Rana, 2025).

2.1.5. Kestirimci Bakım Makine Öğrenmesi Algoritmaları

Kestirimci bakım (Predictive Maintenance-PdM), endüstriyel ekipmanların arızalanmadan önce durumunu izleyerek bakım ihtiyaçlarını tahmin etmeyi amaçlayan bir stratejidir. Bu yaklaşım, bakım maliyetlerini azaltırken operasyonel verimliliği artırır. PdM'nin başarısı, kullanılan makine öğrenmesi algoritmalarının etkinliğine bağlıdır. Bu bölümde, PdM uygulamalarında sıkça tercih edilen makine öğrenmesi algoritmaları teknik parametreleriyle birlikte incelenmiştir (Lekidis vd., 2024; de Andrade Melani vd., 2021).

Çizelge 2.2. Kestirimci Bakım Amaçlı Kullanılan Algoritmalar

Algoritma	Temel Parametreler	Avantajlar	Dezavantajlar	Kullanım Alanları
Karar Ağaçları (Decision Trees)	max_depth, min_samples_split, criterion	Kolay yorumlanabilir, hızlı çalışır	Aşırı öğrenme riski yüksektir	Arıza teşhis modelleri
Rastgele Ormanlar (Random Forest)	n_estimators, max_depth, bootstrap	Aşırı öğrenmeye karşı dirençli, yüksek doğruluk	Eğitim süresi uzundur, bellek kullanımı fazladır	Çoklu arıza türlerinin sınıflandırılması
Destek Vektör Makineleri (SVM)	C, kernel, gamma	Karmaşık sınıflarda güçlü performans, küçük veri setlerinde etkilidir	Büyük veri setlerinde yavaş çalışır, parametre ayarı zordur	Anomali tespiti ve sınıflandırma
Yapay Sinir Ağları (ANN)	hidden_layers, activation, solver, alpha	Karmaşık ilişkileri öğrenebilir, yüksek doğruluk sağlar	Büyük veri ve işlem gücü gerektirir, aşırı öğrenme riski vardır	Sensör verilerinin analizi, arıza tahmini
K-En Yakın Komşu (KNN)	n_neighbors, weights, metric	Basit ve sezgiseldir, parametrik olmayan yaklaşım	Büyük veri setlerinde yavaş çalışır, bellek kullanımı fazladır	Basit kestirimci bakım uygulamaları
XGBoost	n_estimators, learning_rate, max_depth	Hızlı ve yüksek doğruluk, aşırı öğrenmeye karşı dirençli	Parametre ayarı karmaşıktır, hesaplama maliyeti yüksektir	Kalan kullanım ömrü tahmini, arıza sınıflandırması
Zaman Serisi Modelleri (ARIMA/LSTM)	p, d, q (ARIMA); units, activation, optimizer (LSTM)	Zaman bağımlılıklarını modellemede etkilidir	LSTM'ler büyük veri ve işlem gücü gerektirir, ARIMA lineer varsayımlar yapar	Süreç trend tahmini, zaman serisi analizi

2.2. Zaman Serileri

Çalışmada kullanılan makine ve derin öğrenme algoritmaları zaman serileri analizi için kullanılan yöntemlerdir.

2.2.1. SARIMA

SARIMA (Seasonal AutoRegressive Integrated Moving Average) zaman serisi verileri üzerinde çalışan klasik istatistiksel bir modeldir.
Temel bileşenleri şunlardır:

- Seasonal (Mevsimsel): Yılın belirli dönemlerinde tekrarlayan desenlerin (pattern) modellenmesi.
- AR (AutoRegressive): Geçmiş değerlerin şimdiki değeri etkilediği kısım (örnek: y(t-1), y(t-2) vs.)
- I (Integrated): Serideki trendi ortadan kaldırmak için fark alma (differencing) işlemi.
- MA (Moving Average): Önceki hata terimlerinin şimdiki değeri etkilediği kısım.

SARIMA (p, d, q) (P, D, Q) s parametreleriyle yapılandırılır:

- p, d, q → normal ARIMA parametreleri
- P, D, Q → mevsimsel ARIMA parametreleri
- s → mevsim uzunluğu (örnek: saatlik veri için 24, günlük veri için 7 gibi)

SARIMA modeli şu şekilde ifade edilir:

$$SARIMA(p,d,q)(P,D,Q)_s \tag{2.1}$$

Genel formül:

$$\Phi_p(B^s)\phi_p(B)(1-B)^d(1-B^s)^D y_t = \Theta_Q(B^s)\theta_q(B)\epsilon_t \tag{2.2}$$

Buradaki terimler:

y_t : Zaman serisindeki gözlem

B : Gecikme operatörü

$\phi_p(B)$: p dereceli AR (Auto Regressive) polinomu

$\theta_q(B)$: q dereceli MA (Moving Average) polinomu

$\Phi_p(B^s)$: P dereceli mevsimsel AR polinomu

$\Theta_Q(B^s)$: Q dereceli mevsimsel MA polinomu

d : Fark alma derecesi

D : Mevsimsel fark alma derecesi

s : Mevsimsel periyot

Olarak ifade edilmiştir (Bontempi vd., 2020; Kumar Dubey vd., 2021).

2.2.2. Gradient Boosting Regressor (GBR)

Gradient Boosting, zayıf tahmincilerin (genellikle küçük karar ağaçları) ardışık olarak birbirlerinin hatalarını düzelttiği bir yöntemdir.

- Model her adımda mevcut hata (residual) üzerine odaklanır.
- Kayıp fonksiyonun (loss function) negatif gradyanını takip ederek yeni modeller üretilir.

Ana akış:

- İlk basit model oluşturulur.
- Kalan hatalar hesaplanır.
- Yeni model bu hataları azaltmaya çalışır.
- Adım adım toplanarak güçlü bir ansambl model elde edilir.

Gradient Boosting modeli, kayıp fonksiyonunun negatif gradyanını minimize ederek ardışık modeller inşa eder.

Başlangıç:

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma) \tag{2.3}$$

Her iterasyonda:

1. Gradyan (residual) hesaplanır:

$$r_{im} = -\left[\frac{\partial L\big(y_i, F(x_i)\big)}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \tag{2.4}$$

2. Yeni bir model $h_m(x)$ residual'lere fit edilir.
3. Model güncellenir:

$$F_m(x) = F_{m-1}(x) + \nu\gamma_m h_m(x) \tag{2.5}$$

Burada:

ν : öğrenme oranı (learning rate)

γ_m : adım büyüklüğü (line search ile bulunur)

Olarak ifade edilmiştir (Bontempi vd., 2020).

2.2.3. XGBoost (Extreme Gradient Boosting)

XGBoost, Gradient Boosting'in optimize edilmiş ve güçlendirilmiş bir versiyonudur. Öne çıkan özellikleri:

- Regularization (düzenlileştirme) içerir → aşırı öğrenmeyi azaltır.
- Sparsity-aware split yapabilir → eksik veri ile bile etkili çalışır.
- Tree pruning (ağaç budama) ile daha kompakt modeller oluşturur.

- Paralel işlem destekler (multi-threaded).

Matematiksel olarak optimizasyon, ikinci dereceden Taylor serisi açılımı kullanarak yapılır (loss fonksiyonunun hem birinci hem ikinci türevlerini kullanır).

XGBoost modeli, ikinci dereceden Taylor serisi açılımı ile optimizasyon yapar.

Amaç fonksiyonu:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t) \tag{2.6}$$

Burada:

l : Kayıp fonksiyonu

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

T : Ağaçtaki yaprak sayısı

w_j: Yaprak skorları

γ, λ : Düzenlileştirme parametreleri

Taylor ikinci dereceden açılımı ile:

$$\mathcal{L}^{(t)} \approx \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)\right] + \Omega(f_t) \tag{2.7}$$

g_i: birinci türev (gradient)

h_i: ikinci türev (Hessian)

Her split'te split gain şu şekilde hesaplanır:

$$\text{Gain} = \frac{1}{2}\left(\frac{\mathrm{G_L^2}}{\mathrm{H_L} + \lambda} + \frac{\mathrm{G_R^2}}{\mathrm{H_R} + \lambda} - \frac{(\mathrm{G_L} + \mathrm{G_R})^2}{\mathrm{H_L} + \mathrm{H_R} + \lambda}\right) - \gamma \tag{2.8}$$

- G_L, G_R: sol ve sağ alt gradient toplamı
- H_L, H_R: sol ve sağ alt Hessian toplamı

(Chen ve Guestrin, 2016; Li vd., 2022).

2.2.4. LSTM (Long Short-Term Memory)

LSTM, özellikle sıralı veriler için geliştirilmiş bir tür RNN (Recurrent Neural Network) mimarisidir.

RNN'lerin uzun zaman aralıklarındaki bilgi taşımakta yaşadığı vanishing gradient problemi çözmek için tasarlanmıştır. (Eang ve Lee, 2024)

Temel yapı taşları:

- Forget Gate: Hangi bilgiyi unutmamız gerektiğine karar verir.
- Input Gate: Hangi yeni bilgiyi hücreye ekleyeceğimize karar verir.
- Output Gate: Hücre durumunun hangi kısmının çıktıya aktarılacağını belirler.

LSTM hücresi şöyle çalışır:

- Bilgi taşıma kapasitesi uzun sekanslar boyunca korunur.
- Geçmiş saatlerde/dakikalarda/saniyelerde olan bir olayı hatırlamak için idealdir.

Bir LSTM hücresinde aşağıdaki denklemler çalışır:

Forget Gate:

$$f_t = \sigma\big(W_f \cdot [h_{t-1}, x_t] + b_f\big) \tag{2.9}$$

Input Gate:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.10}$$

$$\widetilde{C_t} = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{2.11}$$

Cell State Update:

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C_t} \tag{2.12}$$

Output Gate:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.13}$$

$$h_t = o_t * \tanh(C_t) \tag{2.14}$$

Burada:

σ : sigmoid aktivasyon fonksiyonu

$\tanh$: tanh aktivasyon fonksiyonu

W ve b : Ağırlık ve biaslar

x_t : Giriş verisi

h_{t-1} : Bir önceki gizli durum

(Hochreiter ve Schmidhuber, 1997; Shiri vd., 2023).

2.2.5. Prophet

Prophet, Facebook tarafından geliştirilmiş, kullanıcı dostu bir zaman serisi modelidir. Model, üç ana bileşenin toplamı olarak kurulmuştur:

- Trend: Veri setinin genel yükseliş veya düşüş yönü.
- Seasonality: Günlük, haftalık, yıllık gibi periyodik değişimler.
- Holidays/Events: Anomali yaratabilecek özel günler.

Prophet modeli:

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t \quad (2.15)$$

Bileşenler:

$g(t)$: Trend fonksiyonu (Lineer veya lojistik büyüme)

$s(t)$: Sezonsallık (Fourier serileri ile hesaplanan)

$h(t)$: Tatillerin ve özel günlerin etkisi

ϵ_t : Hata terimi (beyaz gürültü)

Trend Modeli (Piecewise Linear):

$$g(t) = (k + a(t)^T\delta)t + (m + a(t)^T\gamma) \quad (2.16)$$

- k: başlangıç eğimi
- m: başlangıç sapması
- δ: kırılma noktalarındaki değişim
- a(t): kırılma noktaları sonrasında 1 olan binary vektör

Sezonsallık (Fourier Serisi Yaklaşımı):

$$s(t) = \sum_{n=1}^{N} \left(a_n \cos\left(\frac{2\pi nt}{P}\right) + b_n \sin\left(\frac{2\pi nt}{P}\right) \right) \quad (2.17)$$

- P: periyot

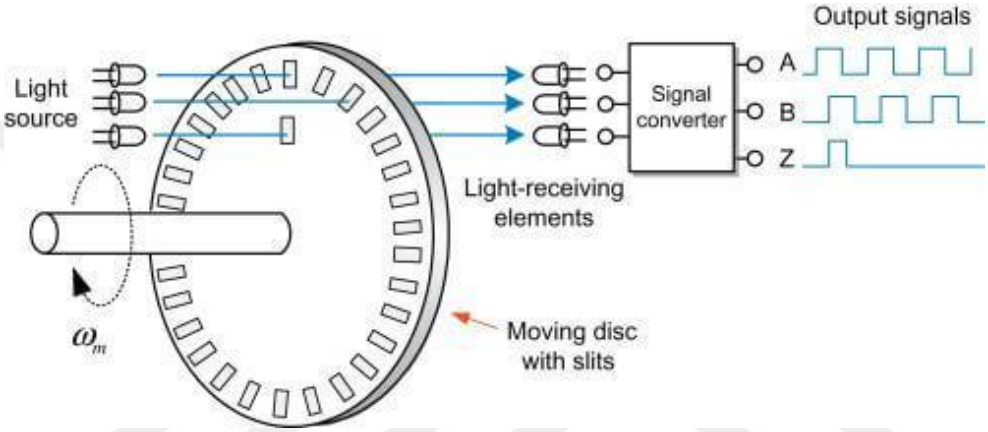
(Taylor ve Letham, 2021; Parizad ve Hatziadoniu, 2021).

Çizelge 2.3. Algoritma Karşılaştırma Tablosu

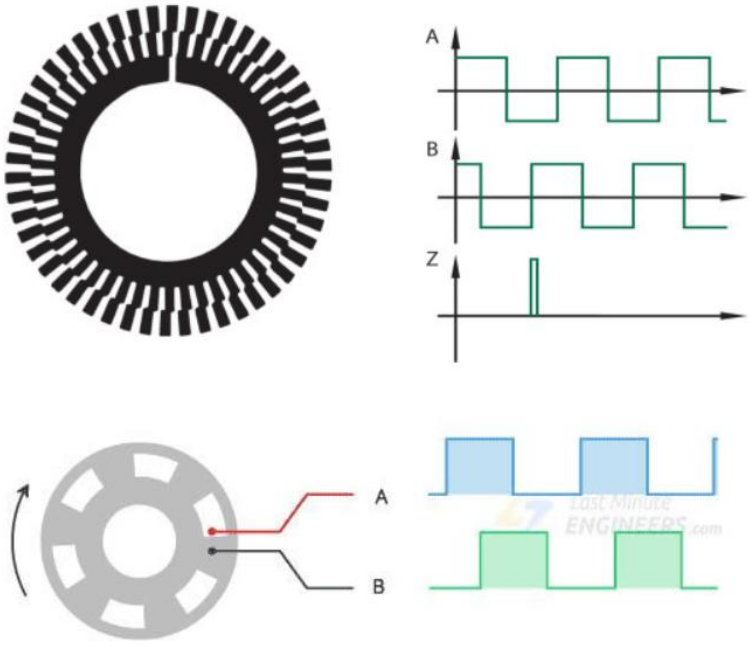
Özellik	**SARIMA**	**Gradient Boosting**	**XGBoost**	**LSTM**	**Prophet**
Model Tipi	Lineer, sezonsal	Ensemble, sequential	Ensemble, optimized	Derin öğrenme, hafızalı	Bileşen temelli
Hız	Orta	Yavaş	Hızlı	Yavaş	Hızlı
Sezonsallık Yönetimi	Parametrik	Özellik mühendisliği	Özellik mühendisliği	Öğrenme ile	Fourier serisi ile
Overfitting Riski	Düşük	Orta	Orta	Yüksek	Düşük
Veri Miktarı İhtiyacı	Orta	Yüksek	Orta/Yüksek	Yüksek	Düşük/Orta

3. MATERYAL ve YÖNTEM

Bu çalışmada lojistik amaçlı bir dağıtım/ayrıştırma tesisinin çıktıları analiz için düzenlenmiş ve makine öğrenmesi destekli kestirimci bakım yapılmıştır. Bu kısımda verinin hazırlanması için geliştirilen yazılımın detayları ve birden fazla makine öğrenmesi algoritması ile elde edilen sonuçların karşılaştırması yapılmıştır. Kestirimci bakım çalışması için seçilen alt komponent konveyör motorlarıdır. Enkoder verisi PPI (pulse position indicator) adında sensör tabanlı bir mekanizma ile ölçülmektedir.



Şekil 3.1. Darbe Konum Göstergesi Genel Yapısı



Şekil 3.2. Motor Hareketi ile Üretilen Sinyal Yapısı

3.1. Sistem Verilerinin İncelenmesi

Sistem verileri .txt formatında anlık olarak alınmaktadır. Dosya boyutu 10 mb olarak sınırlandırılmıştır, her 10 mb veri bitiminde yeni dosya oluşmaktadır ve veriler ftp sunucuda saklanmaktadır. Her dosya yaklaşık olarak 3 dakikalık veri içermektedir.

Örnek Veri:

```
00:00:02.028 [785][01] W>{PPICheckLeading.cpp:154}[8800.84.01.PPI21.PPIO] Deviation=-27393 PPI=482600 TimerPPI=509993
00:00:02.422 [785][01] W>{PPICheckLeading.cpp:154}[8800.83.04.PPI21.PPIO] Deviation=-25004 PPI=452200 TimerPPI=477204
00:00:02.608 [785][01] W>{PPICheckLeading.cpp:154}[8800.88.01.PPI21.PPIO] Deviation=-27374 PPI=482600 TimerPPI=509974
00:00:03.554 [785][01] W>{PPICheckLeading.cpp:154}[8800.89.06.PPI21.PPIO] Deviation=-23805 PPI=452200 TimerPPI=476005
00:00:03.728 [785][01] W>{PPICheckLeading.cpp:154}[8800.84.01.PPI21.PPIO] Deviation=-27398 PPI=482600 TimerPPI=509998
00:00:03.837 [785][01] W>{PPICheckLeading.cpp:154}[8800.89.07.PPI21.PPIO] Deviation=-23838 PPI=452200 TimerPPI=476038
00:00:03.945 [785][01] W>{PPICheckLeading.cpp:154}[8800.86.01.PPI21.PPIO] Deviation=-27453 PPI=482600 TimerPPI=510053
00:00:04.508 [785][01] W>{PPICheckLeading.cpp:154}[8800.83.04.PPI21.PPIO] Deviation=-23807 PPI=452200 TimerPPI=476007
00:00:05.645 [785][01] W>{PPICheckLeading.cpp:154}[8800.86.01.PPI21.PPIO] Deviation=-27395 PPI=482600 TimerPPI=509995
00:00:05.661 [785][01] W>{PPICheckLeading.cpp:154}[8800.84.01.PPI21.PPIO] Deviation=-27397 PPI=482600 TimerPPI=509997
00:00:05.668 [785][01] W>{PPICheckLeading.cpp:154}[8800.88.01.PPI21.PPIO] Deviation=-27406 PPI=482600 TimerPPI=510006
00:00:05.796 [785][01] W>{PPICheckLeading.cpp:154}[8800.87.06.PPI21.PPIO] Deviation=-24064 PPI=452200 TimerPPI=476264
00:00:06.540 [785][01] W>{PPICheckLeading.cpp:154}[8800.87.04.PPI21.PPIO] Deviation=-23807 PPI=452200 TimerPPI=476007
00:00:07.580 [785][01] W>{PPICheckLeading.cpp:154}[8800.83.04.PPI21.PPIO] Deviation=-23803 PPI=452200 TimerPPI=476003
00:00:07.685 [785][01] W>{PPICheckLeading.cpp:154}[8800.86.01.PPI21.PPIO] Deviation=-27390 PPI=482600 TimerPPI=509990
00:00:07.701 [785][01] W>{PPICheckLeading.cpp:154}[8800.84.01.PPI21.PPIO] Deviation=-27368 PPI=482600 TimerPPI=509968
00:00:08.728 [785][01] W>{PPICheckLeading.cpp:154}[8800.88.01.PPI21.PPIO] Deviation=-27362 PPI=482600 TimerPPI=509962
00:00:09.147 [785][01] W>{PPICheckLeading.cpp:154}[8800.89.05.PPI21.PPIO] Deviation=-23805 PPI=452200 TimerPPI=476005
00:00:09.284 [785][01] W>{PPICheckLeading.cpp:154}[8800.83.07.PPI21.PPIO] Deviation=-24229 PPI=452200 TimerPPI=476429
00:00:09.401 [785][01] W>{PPICheckLeading.cpp:154}[8800.84.01.PPI21.PPIO] Deviation=-27347 PPI=482600 TimerPPI=509947
00:00:09.610 [785][01] W>{PPICheckLeading.cpp:154}[8800.87.04.PPI21.PPIO] Deviation=-23832 PPI=452200 TimerPPI=476032
00:00:09.725 [785][01] W>{PPICheckLeading.cpp:154}[8800.86.01.PPI21.PPIO] Deviation=-27322 PPI=482600 TimerPPI=509922
00:00:10.589 [785][01] W>{PPICheckLeading.cpp:154}[8800.89.06.PPI21.PPIO] Deviation=-23838 PPI=452200 TimerPPI=476038
00:00:11.441 [785][01] W>{PPICheckLeading.cpp:154}[8800.84.01.PPI21.PPIO] Deviation=-27399 PPI=482600 TimerPPI=509999
```

Şekil 3.3. Motor Verisi İçin Filtrelenmiş Ham Veri

Kaynak verilerinden motorun anlık hızı (PPI), olması gereken hız (TimerPPI) ve sapma bilgileri alınmaktadır. (Veriler sadece motor bilgileri için filtrelenmiştir.)

3.2. Verilerin Hazırlanması

Analiz için 106 günlük veri incelenmiştir. Toplam veri 2140 text dosyası (177 Gb) oluşmaktadır. Tüm konveyörlerin sapma verileri yaklaşık 13,5 milyon satırdan oluşmaktadır. Bu sebeple düzenlenen veriler için SQL Database kullanılmıştır.

Column Name	Data Type	Allow Nulls
DateTime	datetime2(7)	☐
Section	nchar(10)	☑
Deviation	numeric(7, 0)	☑
PPIEncoder	numeric(8, 0)	☑
PPISet	numeric(8, 0)	☑
PrimaryKey1	bigint	☐
[Dev%]	numeric(4, 2)	☑

Şekil 3.4. Veri Tabanı Tablosu

DateTime: Tarih ve saat bilgisi

Section: Motor Numarası

Deviation: Sapma Değeri

PPI Encoder: Motorun Anlık Hız Bilgisi

PPISet: Hedef Hız Bilgisi

Dev%: Sapma Yüzdesi

Ham verinin filtrelenip tablo haline getirilmesi için Visual Studio programında C# dilinde çalışan bir yazılım geliştirilmiştir. Veri ayıklama ve SQL Database için .NET Framework üzerinde standart kütüphaneler kullanılmıştır.

```
else if (rbDebugDisplacement.Checked || rbDebugAllPPIData.Checked) //DEBUG Displacement Prep
{
    OpenFileDialog openFileDialog = new OpenFileDialog();
    openFileDialog.Multiselect = true;
    if (openFileDialog.ShowDialog() == DialogResult.OK)
    {
        query = "";
        finallist.Clear();
        string tablename;

        if (rbDebugDisplacement.Checked)
        {
            tablename = "DebugLoggingDisplacementData_" + DateTime.Now.ToString("yyyyMMddHHmmss");

            if (CreateSQLTable(tablename))
            {
                for (int i = 0; i < openFileDialog.FileNames.Length; i++)
                {
                    textBox3.Text = (i + 1).ToString() + "/" + openFileDialog.FileNames.Length.ToString();
                    textBox3.Refresh();
                    list.Clear();
                    list.AddRange(File.ReadAllLines(openFileDialog.FileNames[i]));

                    for (int j = 0; j < list.Count; j++)
                    {
                        if (list[j].Contains("PPICheckLeading.cpp"))
                        {
                            finallist.Add(list[j]);

                            if (rbToSQL.Checked)
                            {
                                string a = openFileDialog.SafeFileNames[i].Substring(0, 10);
                                string[] b = a.Split("-");
                                a = b[0] + "/" + b[1] + "/" + b[2];

                                string[] d = (a + " " + list[j]).Split(new char[] { ' ', '=', '[', ']' });

                                if (d.Length >= 26)
                                {
                                    string ddddd = Math.Abs(Math.Round((Convert.ToDouble(d[21]) /
                                        Convert.ToDouble(d[25])) * 100, 2)).ToString().Replace(",", ".");

                                    query += "INSERT INTO [dbo].[" + tablename + "]([DateTime],[Section],[Deviation],"
                                        +"[PPISet],[PPIEncoder],[Dev%]) VALUES("
                                        + "'" + (a + " " + list[j]).Substring(0, 19) + "','" + d[8].Substring(0, 10)
                                        + "','" + d[21] + "','" + d[23] + "','" + d[25] + "','" + ddddd + "')\n";

                                }
                            }
                        }
                    }
                }
```

Şekil 3.5. Veri Hazırlığı Ana Algoritma Kısım 1

```
if (rbToSQL.Checked)
{

    SqlConnection con = new("Server=VI-5CD141JML0\\SQLEXPRESS;Database=PM_ML;Trusted_Connection=True;");
    SqlCommand command; /*new SqlCommand(createTable, con);*/


    command = new SqlCommand(query, con);

    con.Open();
    try
    {
        command.ExecuteNonQuery();
    }
    catch (Exception)
    {
    }
    finally { con.Close(); }
}
else if (rbToText.Checked)
{
    File.WriteAllLines("C:\\Private\\YL2\\Tool_Output\\DebugDisplacement_"
        + DateTime.Now.ToString("ddMMyyyy HHmmss") + ".txt", finallist.ToArray());
}
query = "";
```

Şekil 3.6. Veri Hazırlığı Ana Algoritma Kısım 2

Geliştirilen yazılım ile Şekil 3.3.'te gösterilen ham veri tablo haline getirilmiş ve analiz için hazırlanmıştır.

SQLQuery2.sql - VI-..._ML (VI\nlbtas (68))* | SQLQuery1.sql - VI-..._ML (VI\nlbtas (65))*

```
SELECT  [DateTime]
      ,[Section]
      ,[Deviation]
      ,[PPISet]
      ,[Dev%]
  FROM [PM_ML].[dbo].[DebugLoggingDisplacementData_202504231942_PSR2]
```

100 %

Results | Messages

	DateTime	Section	Deviation	PPISet	Dev%
1	2025-01-02 14:47:48.0000000	8800.83.07	25651	523600	5.15
2	2025-01-02 14:47:48.0000000	8800.82.02	-39401	482600	7.55
3	2025-01-02 14:47:48.0000000	8800.85.05	-23801	452200	5.00
4	2025-01-02 14:47:48.0000000	8800.88.01	-27412	482600	5.37
5	2025-01-02 14:47:49.0000000	8800.84.01	-27610	482600	5.41
6	2025-01-02 14:47:49.0000000	8800.83.04	-23816	452200	5.00
7	2025-01-02 14:47:50.0000000	8800.89.01	-42421	482600	8.08
8	2025-01-02 14:47:50.0000000	8800.82.01	-27407	482600	5.37
9	2025-01-02 14:47:50.0000000	8800.84.01	-27417	482600	5.38
10	2025-01-02 14:47:51.0000000	8800.88.01	-27445	482600	5.38
11	2025-01-02 14:47:51.0000000	8800.82.01	-27372	482600	5.37
12	2025-01-02 14:47:52.0000000	8800.84.01	-27288	482600	5.35
13	2025-01-02 14:47:53.0000000	8800.89.05	-23826	452200	5.01
14	2025-01-02 14:47:54.0000000	8800.89.06	-23808	452200	5.00
15	2025-01-02 14:47:54.0000000	8800.82.01	-27385	482600	5.37
16	2025-01-02 14:47:54.0000000	8800.84.01	-27435	482600	5.38
17	2025-01-02 14:47:55.0000000	8800.88.01	-27136	482600	5.32
18	2025-01-02 14:47:55.0000000	8800.85.04	-23929	452200	5.03
19	2025-01-02 14:47:55.0000000	8800.89.04	-23821	452200	5.00
20	2025-01-02 14:47:56.0000000	8800.82.01	-27344	482600	5.36
21	2025-01-02 14:47:56.0000000	8800.84.01	-27423	482600	5.38
22	2025-01-02 14:47:57.0000000	8800.88.01	-27388	482600	5.37
23	2025-01-02 14:47:58.0000000	8800.83.06	-30098	452200	6.24
24	2025-01-02 14:47:58.0000000	8800.82.01	-27372	482600	5.37
25	2025-01-02 14:47:58.0000000	8800.84.01	-27437	482600	5.38
26	2025-01-02 14:47:58.0000000	8800.85.06	-23890	452200	5.02
27	2025-01-02 14:47:59.0000000	8800.89.05	-23812	452200	5.00
28	2025-01-02 14:47:59.0000000	8800.89.07	-23828	452200	5.01
29	2025-01-02 14:47:59.0000000	8800.82.01	-27376	482600	5.37

Quer... | VI-5CD141JML0\SQLEXPRESS (1... | VI\nlbtas (68) | PM_ML | 00:01:28 | 13.625.515 rows

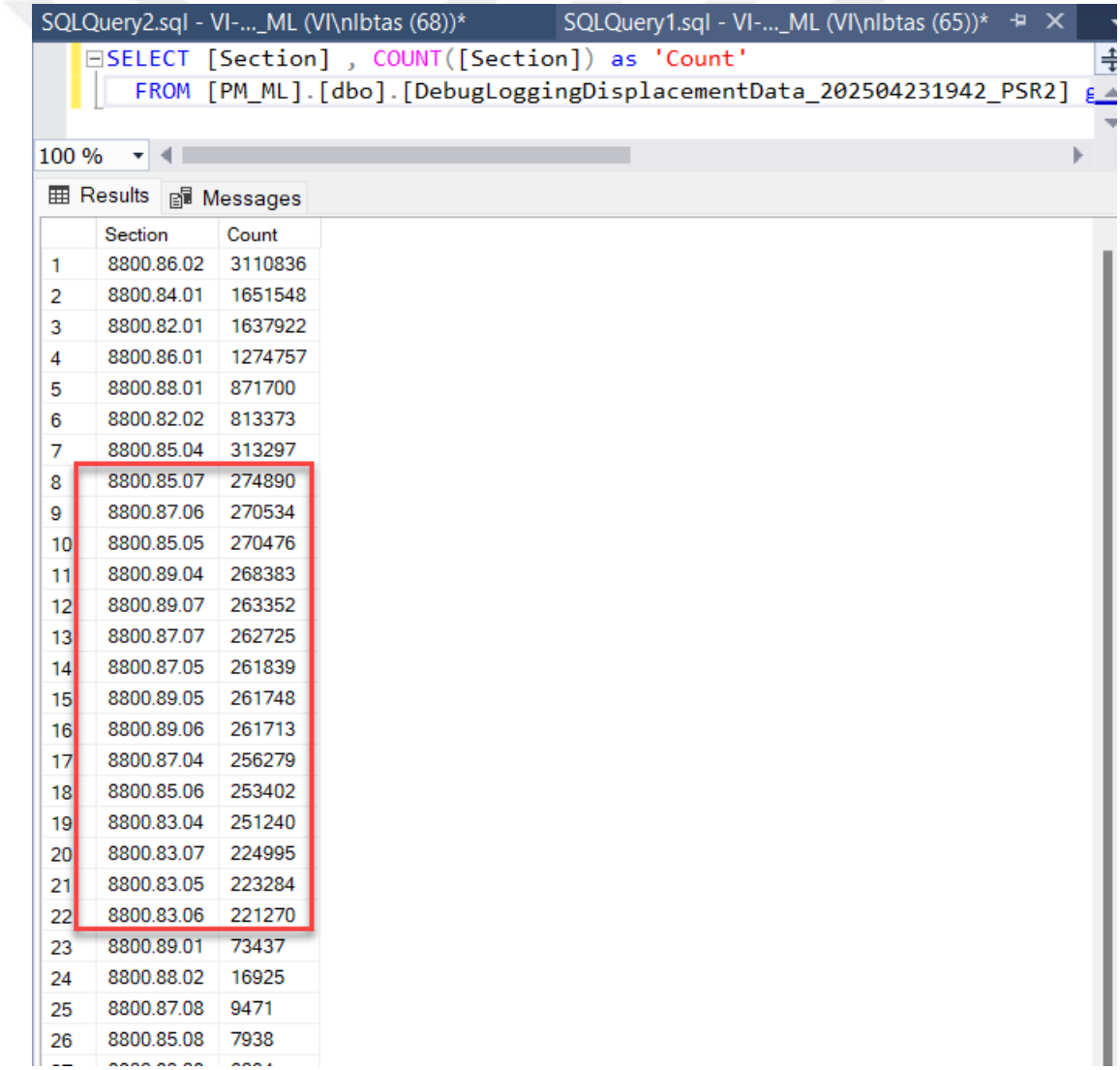
Şekil 3.7. SQL Veri Tabanında Hazırlanmış Veri

3.3. Verilerin Analizi

Çalışmada zaman serileri analizi için aşağıdaki algoritmalar kullanılmıştır:

- SARIMA
- Gradient Boosting
- XGBoost
- LSTM
- Prophet

Toplanan veriler içinde ortalama satır sayısına sahip motorlar kullanılmıştır.



	Section	Count
1	8800.86.02	3110836
2	8800.84.01	1651548
3	8800.82.01	1637922
4	8800.86.01	1274757
5	8800.88.01	871700
6	8800.82.02	813373
7	8800.85.04	313297
8	8800.85.07	274890
9	8800.87.06	270534
10	8800.85.05	270476
11	8800.89.04	268383
12	8800.89.07	263352
13	8800.87.07	262725
14	8800.87.05	261839
15	8800.89.05	261748
16	8800.89.06	261713
17	8800.87.04	256279
18	8800.85.06	253402
19	8800.83.04	251240
20	8800.83.07	224995
21	8800.83.05	223284
22	8800.83.06	221270
23	8800.89.01	73437
24	8800.88.02	16925
25	8800.87.08	9471
26	8800.85.08	7938

Şekil 3.8. Veri Satır Sayısının Motorlara Dağılımı

Kestirimci bakım yazılımı için Google Colab platformu kullanılmıştır (https://colab.research.google.com/) ve Python dilinde geliştirme yapılmıştır. Colab kullanımı tüm kütüphane ve sürücülere erişimi olduğundan birden fazla kütüphanenin kullanıldığı ortamlarda kolaylık sağlamaktadır. Doksan günlük veri test olarak yüklenmiş, sonraki 16 gün algoritmalar ile tahmin edilerek gerçekleşen veri ile karşılaştırılmıştır.

Kod (Python):

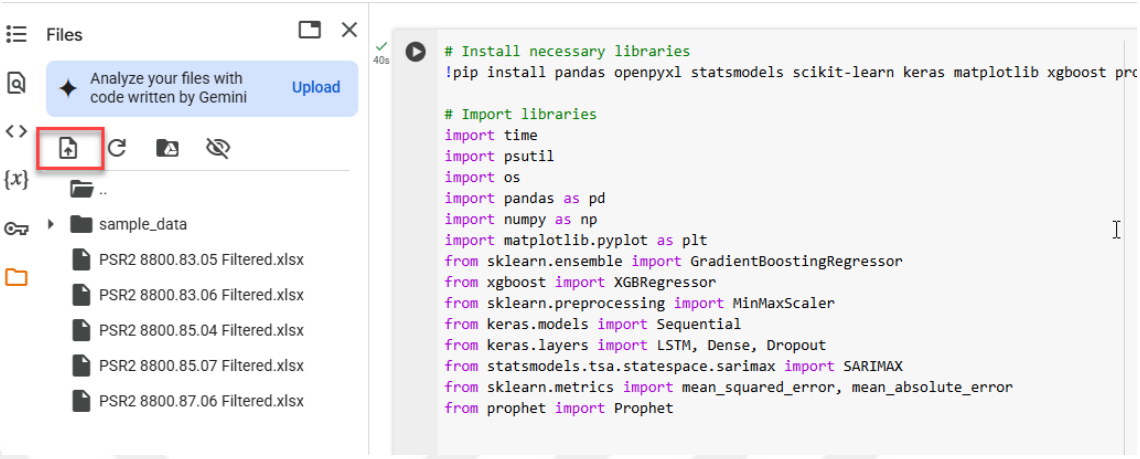
#Gerekli kütüphanelerin yüklenmesi

```
!pip install pandas openpyxl statsmodels scikit-learn keras matplotlib xgboost prophet
```

#Kütüphanelerin Import Edilmesi

```
import time
import psutil
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.ensemble import GradientBoostingRegressor
from xgboost import XGBRegressor
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import LSTM, Dense, Dropout
from statsmodels.tsa.statespace.sarimax import SARIMAX
from sklearn.metrics import mean_squared_error, mean_absolute_error
from prophet import Prophet
```

#Verinin Yüklenmesi (kodun veriyi ‘Content’ içinden okuması için dosyanın colab platformuna yüklenmesi gerekmektedir.)



Şekil 3.9. Colab Dosya Yükleme

file_path = ‘/content/PSR2 8800.85.07 Filtered.xlsx’

df = pd.read_excel(file_path)

#Günlük Ortalama Değerin Hazırlanması

df[‘Date’] = pd.to_datetime(df[‘DateTime’])

25igür_df = df.groupby(df[‘Date’].dt.date)[‘Dev%’].mean().reset_index()

25igür_df.columns = [‘ds’, ‘y’]

#Test ve Tahmin edilecek verilerin ayrılması

train_df = 25igür_df.iloc[:90]

test_df = 25igür_df.iloc[90:106]

print(f”Training from {train_df[‘ds’].iloc[0]} to {train_df[‘ds’].iloc[-1]}”)

print(f”Testing from {test_df[‘ds’].iloc[0]} to {test_df[‘ds’].iloc[-1]}”)

#Gerçekleşen verilerin tanımlanması

actual_values = test_df[‘y’].values

#Model 1: SARIMA Algoritması

```
start_time = time.time()
sarima_model = SARIMAX(train_df[‘y’], order=(1,1,1), seasonal_order=(1,1,1,7))
sarima_result = sarima_model.fit(disp=False)
sarima_forecast = sarima_result.forecast(steps=16)
sarima_time = time.time() — start_time
```

 # Model 2: Gradient Boosting Algoritması

```
start_time = time.time()

# Gecikme Özelliği Tanımı
train_df[‘y_shifted’] = train_df[‘y’].shift(1)
train_df = train_df.dropna()

X_train = train_df[[‘y_shifted’]]
y_train = train_df[‘y’]

gb_model = GradientBoostingRegressor()
gb_model.fit(X_train, y_train)

# Tekrarlı Tahmin
gb_preds = []
last_value = X_train.iloc[-1][‘y_shifted’]

for _ in range(16):
    pred = gb_model.predict([[last_value]])[0]
    gb_preds.append(pred)
    last_value = pred

gb_time = time.time() — start_time
```

#Model 3: XGBoost Algoritması

```
start_time = time.time()

xgb_model = XGBRegressor(objective=”reg:squarederror”)
xgb_model.fit(X_train, y_train)

# Tekrarlı Tahmin
xgb_preds = []
last_value = X_train.iloc[-1][‘y_shifted’]

for _ in range(16):
    pred = xgb_model.predict([[last_value]])[0]
    xgb_preds.append(pred)
    last_value = pred

xgb_time = time.time() — start_time
```

 # Model 4: LSTM Algoritması

```
start_time = time.time()

# Normalize y tanımı
scaler = MinMaxScaler()
train_scaled = scaler.fit_transform(train_df[[‘y’]])

# Sekansların Üretilmesi
seq_len = 10
X_lstm, y_lstm = [], []

for i in range(len(train_scaled) — seq_len):
    X_lstm.append(train_scaled[i:i + seq_len])
    y_lstm.append(train_scaled[i + seq_len])
```

```
X_lstm = np.array(X_lstm)
y_lstm = np.array(y_lstm)

# LSTM Model Yapılandırması
lstm_model = Sequential([
   LSTM(50, input_shape=(seq_len, 1)), Dropout(0.2), Dense(1)])

lstm_model.compile(optimizer='adam', loss='mse')
lstm_model.fit(X_lstm, y_lstm, epochs=50, batch_size=8, verbose=1)

# 16 Günlük Tahmin
last_seq = train_scaled[-seq_len:].reshape(1, seq_len, 1)
lstm_preds_scaled = []

for _ in range(16):
   pred = lstm_model.predict(last_seq, verbose=0)[0][0]
   lstm_preds_scaled.append(pred)
   last_seq = np.append(last_seq[:, 1:, :], [[[pred]]], axis=1)

lstm_preds = scaler.inverse_transform(np.array(lstm_preds_scaled).reshape(-
1,1)).flatten()

lstm_time = time.time() — start_time
```

 # Model 5: Prophet Algoritması

```
start_time = time.time()

prophet_df = train_df[['ds', 'y']].copy()
prophet_model = Prophet(28igür_seasonality=True)
prophet_model.fit(prophet_df)

future = prophet_model.make_future_dataframe(periods=16)
```

```
forecast = prophet_model.predict(future)

prophet_preds = forecast.iloc[-16:][‘yhat’].values

prophet_time = time.time() — start_time

# Değerlendirme RMSE ve MAE
def evaluate_model(name, preds):
    rmse = np.sqrt(mean_squared_error(actual_values, preds))
    Mae = mean_absolute_error(actual_values, preds)
    print(f”{name} -> RMSE: {rmse:.5f}, MAE: {mae:.5f}”)

print(“\n=== Model Evaluation ===”)
evaluate_model(“SARIMA”, sarima_forecast)
evaluate_model(“Gradient Boosting”, gb_preds)
evaluate_model(“XGBoost”, xgb_preds)
evaluate_model(“LSTM”, lstm_preds)
evaluate_model(“Prophet”, prophet_preds)

# Grafik Çizimi
plt.figure(figsize=(16,7))
plt.plot(train_df[‘ds’], train_df[‘y’], label=”Training Data”, color=”gray”)
plt.plot(test_df[‘ds’], actual_values, label=”Actual Dev% (Test)”, color=”black”, marker=’o’)

plt.plot(test_df[‘ds’], sarima_forecast, label=”SARIMA Forecast”, linestyle=”—“, marker=’x’, color=”red”)
plt.plot(test_df[‘ds’], gb_preds, label=”Gradient Boost Forecast”, linestyle=”—“, marker=’^’, color=”green”)
plt.plot(test_df[‘ds’], xgb_preds, label=”XGBoost Forecast”, linestyle=”—“, marker=’s’, color=”orange”)
```

```
plt.plot(test_df[‘ds’], lstm_preds, label=”LSTM Forecast”, linestyle=”—“, marker=’v’, color=”blue”)
plt.plot(test_df[‘ds’], prophet_preds, label=”Prophet Forecast”, linestyle=”—“, marker=’d’, color=”purple”)

plt.title(“Forecast Comparison (Train: First 90 Days, Test: Next 16 Days)”)
plt.xlabel(“Date”)
plt.ylabel(“Dev%”)
plt.legend()
plt.grid(True)
plt.show()
```

#Tahmin Edilen Değerlerin Yazdırılması

```
results_df = pd.DataFrame({
    ‘Date’: test_df[‘ds’],
    ‘Actual’: actual_values,
    ‘SARIMA’: sarima_forecast.values,
    ‘Gradient Boosting’: gb_preds,
    ‘XGBoost’: xgb_preds,
    ‘LSTM’: lstm_preds,
    ‘Prophet’: prophet_preds
})
print(“\n===Tahmin Sonuçları Tablosu ===”)
print(results_df)
```

 # Performans Özeti Tablosu

```
performance_summary = pd.DataFrame({
    ‘Model’: [‘SARIMA’, ‘Gradient Boosting’, ‘XGBoost’, ‘LSTM’, ‘Prophet’],
    ‘Time (s)’: [sarima_time, gb_time, xgb_time, lstm_time, prophet_time],
})
print(“\n=== Performans Özeti ===”)
print(performance_summary)
```

3.3.1. Analiz Sonuçları

Analiz sonuçları değerlendirmesi için kullanılan parametreler:

- MAE (Mean Absolute Error)

MAE, modelimizin yaptığı hataların ortalama büyüklüğünü ölçer. Hata değerlerinin (gerçek- tahmin) mutlak değerini alır, toplamını bulur ve veri sayısına böler.

Matematiksel tanımı:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \widehat{y_i}| \tag{3.1}$$

y_i: Gerçek değer

$\widehat{y_i}$: Tahmin edilen değer

n : Toplam veri sayısı

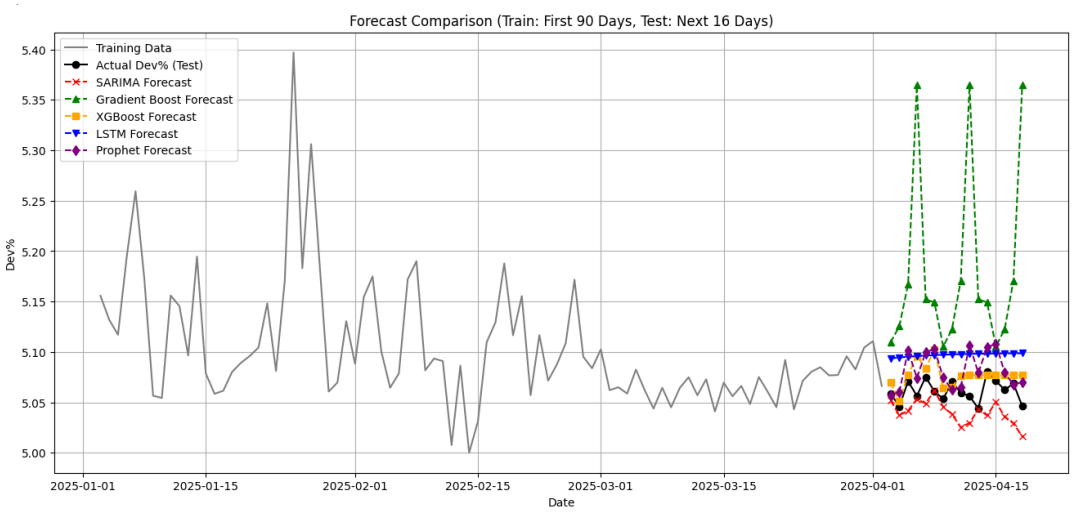
- RMSE (Root Mean Squared Error)

RMSE, hataların karelerinin ortalamasının karekökü dür. Büyük hataları karelediği için, birkaç büyük hataya karşı daha duyarlıdır.

Matematiksel tanımı:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y_i})^2} \tag{3.2}$$

Motor 8800.83.06 Analizi:



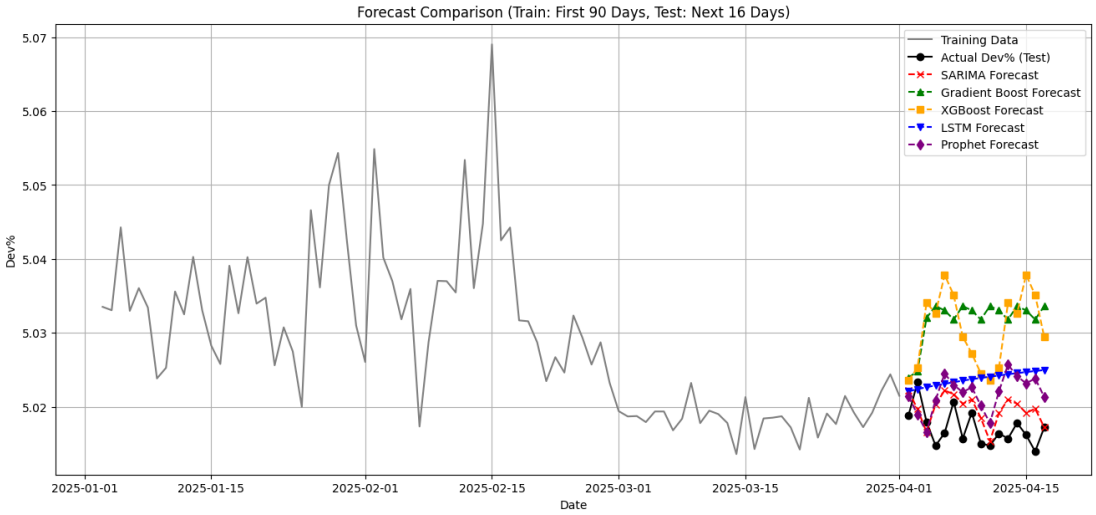
Şekil 3.10. Motor 8800.83.06 Analiz Sonuçları

Çizelge 3.1. Motor 8800.83.06 RMSE ve MAE Değerleri

	RMSE	**MAE**
SARIMA	0,0251	0,0211
Gradient Boost	0,1526	0,1197
XGBoost	0,0206	0,0163
LSTM	0,0373	0,0358
Prophet	0,0262	0,0221

Bu analizde en başarılı algoritmalar SARIMA, XGBoost ve Prophet olmuştur.

Motor 8800.85.04 Analizi:



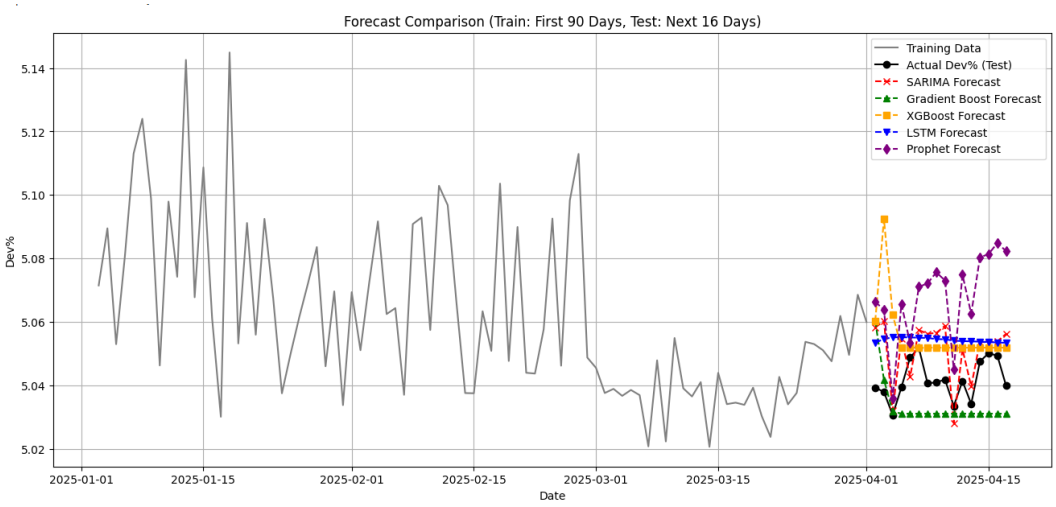
Şekil 3.11. Motor 8800.85.04 Analiz Sonuçları

Çizelge 3.2. Motor 8800.85.04 RMSE ve MAE Değerleri

	RMSE	**MAE**
SARIMA	0,0036	0,0031
Gradient Boost	0,0153	0,0146
XGBoost	0,0145	0,0133
LSTM	0,0072	0,0067
Prophet	0,0059	0,0053

Bu analizde en başarılı algoritmalar SARIMA ve Prophet olmuştur.

Motor 8800.87.06 Analizi:



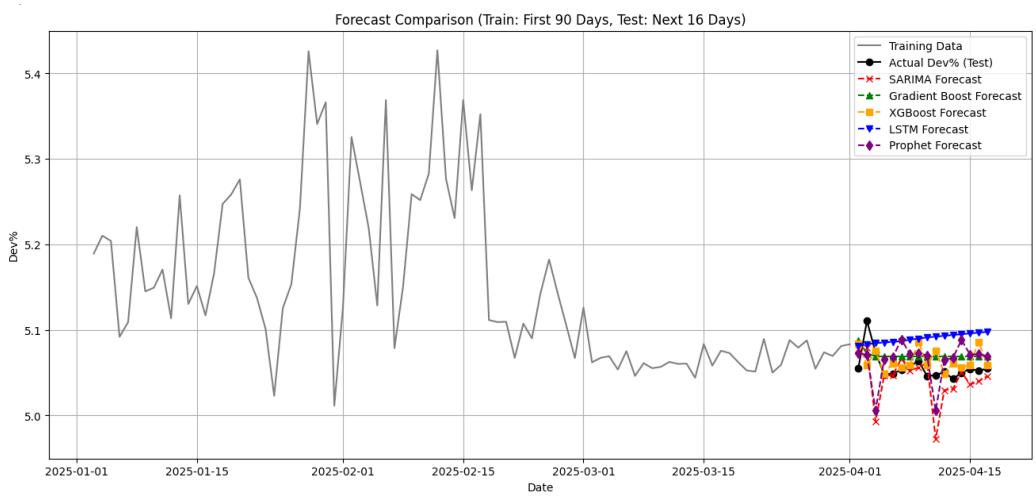
Şekil 3.12. Motor 8800.87.06 Analiz Sonuçları

Çizelge 3.3. Motor 8800.87.06 RMSE ve MAE Değerleri

	RMSE	**MAE**
SARIMA	0,0123	0,0105
Gradient Boost	0,0130	0,0113
XGBoost	0,0191	0,0138
LSTM	0,0140	0,0126
Prophet	0,0283	0,0263

Bu analizde en başarılı algoritmalar SARIMA, Gradient Boost ve LSTM olmuştur.

Motor 8800.85.07 Analizi



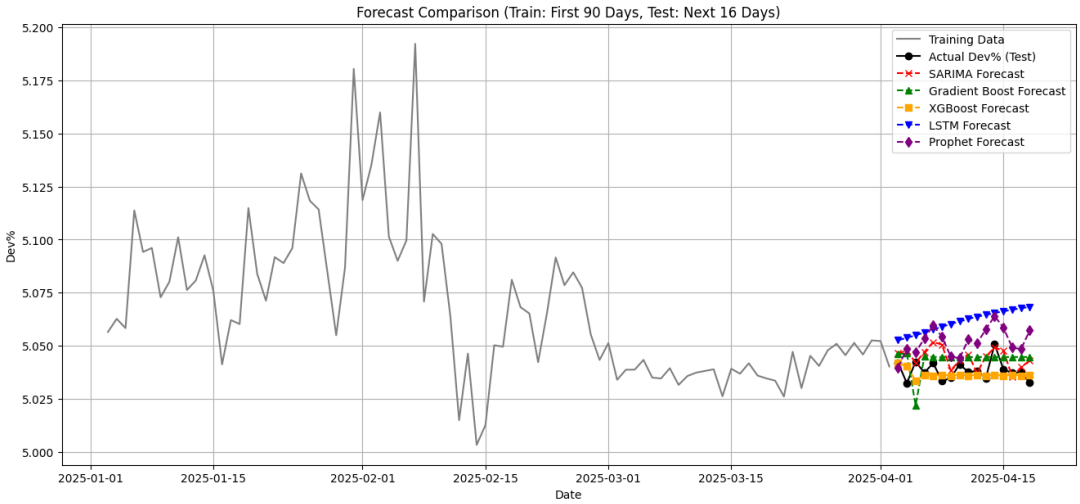
Şekil 3.13. Motor 8800.85.07 Analiz Sonuçları

Çizelge 3.4. Motor 8800.85.07 RMSE ve MAE Değerleri

	RMSE	**MAE**
SARIMA	0,0308	0,0204
Gradient Boost	0,0206	0,0188
XGBoost	0,0203	0,0144
LSTM	0,0380	0,0367
Prophet	0,0296	0,0258

Bu analizde en başarılı algoritmalar Gradient Boost ve XGBBoost olmuştur.

Motor 8800.83.05 Analizi



Şekil 3.14. Motor 8800.83.05 Analiz Sonuçları

Çizelge 3.5. Motor 8800.83.05 RMSE ve MAE Değerleri

	RMSE	**MAE**
SARIMA	0,0084	0,0066
Gradient Boost	0,0095	0,0085
XGBoost	0,0053	0,0038
LSTM	0,0240	0,0230
Prophet	0,0153	0,0139

Bu analizde en başarılı algoritmalar XGBBoost ve SARIMA olmuştur.

4. BULGULAR

4.1. Genel Performans Analizi

Bu çalışmada SARIMA, Gradient Boosting, XGBoost, LSTM ve Prophet algoritmaları kullanılarak dört farklı motor veri seti üzerinde tahminleme çalışmaları yapılmıştır. Modellerin performansı, RMSE (Root Mean Squared Error) ve MAE (Mean Absolute Error) hata metrikleri kullanılarak ölçülmüştür.

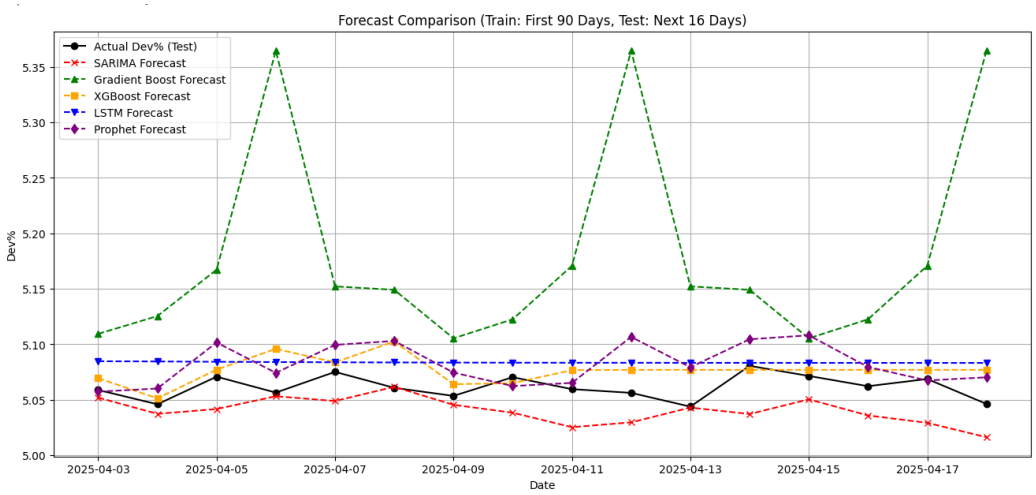
Çizelge 4.1. Algoritma Performans Karşılaştırması

		Algoritma				
Motor	**Parametre**	**SARIMA**	**Gradient Boost**	**XGBoost**	**LSTM**	**Prophet**
8800.83.06	RMSE	0,0251	0,1526	0,0206	0,0373	0,0262
	MAE	0,0211	0,1197	0,0163	0,0358	0,0221
8800.85.04	RMSE	0,0036	0,0153	0,0145	0,0072	0,0059
	MAE	0,0031	0,0146	0,0133	0,0067	0,0053
8800.87.06	RMSE	0,0123	0,013	0,0191	0,014	0,0283
	MAE	0,0105	0,0113	0,0138	0,0126	0,0263
8800.85.07	RMSE	0,0308	0,0206	0,0203	0,038	0,0296
	MAE	0,0204	0,0188	0,0144	0,0367	0,0258

Çizelge 4.1.'de görüleceği üzere SARIMA ve XGBoost algoritmaları yapılan analizde en iyi sonuçları vermiştir. SARIMA algoritması tekrarlayan desenleri daha iyi tahmin etmiştir. Tesisin haftanın belirli günleri daha yoğun olması ve bakım günlerinin düzenli aralıklarla belirlenmesi SARIMA algoritmasının mevsimsel etkileri analiz etme özelliğinin bu çalışmada faydalı olduğunu göstermiştir. XGBoost düzenlileştirme özelliği sayesinde desenleri yakalayamamış olsa da yakın değerler tahmin edebilmiştir. XGBoost algoritmasının avantajlarından biri eksik ve uzun aralıklı verilerle bile çalışabilmesidir, çalışmada veri yapısı oldukça sık ve düzenli olduğu için algoritma bu avantajını kullanamamıştır. Gradient Boost genel olarak iyi performans gösterse de motor

8800.83.06 analizinde (Şekil 3.9.) aşırı öğrenme sebebiyle yüksek RMSE ve MAE değerleri elde edilmiştir. Gradient Boosting, özellikle yüksek doğruluk isteyen problemler için güçlü bir yöntemdir. Ana fikri, her yeni eklenen zayıf modelin (genellikle karar ağaçları) önceki modellerin yapamadığı hataları düzeltmeye çalışmasıdır. Bu şekilde adım adım daha iyi sonuçlar elde edilir. Ancak bu güçlü yapı, aynı zamanda aşırı öğrenme (overfitting) riskini de beraberinde getirir. LSTM algoritması test edilen tüm motorlarda tutarlı sonuçlar vermiştir ama algoritması gereği fazla işlemci gücü tüketse de hiçbir motorda en iyi sonucu verememiştir. Prophet algoritması yaptığımız analiz çok değişkenli olmadığı için iyi sonuçlar vermiştir. Kestirimci bakımda ekipman üzerinde bakım yapıldıktan sonraki veriler kullanılacağından dolayı uzun yıllara ulaşan veriler kullanılmayacaktır bu da Prophet algoritmasının zayıf yönlerinden birini kapatmaktadır.

8800.83.06:



Şekil 4.1. Motor 8800.83.06 Tahmin Verileri

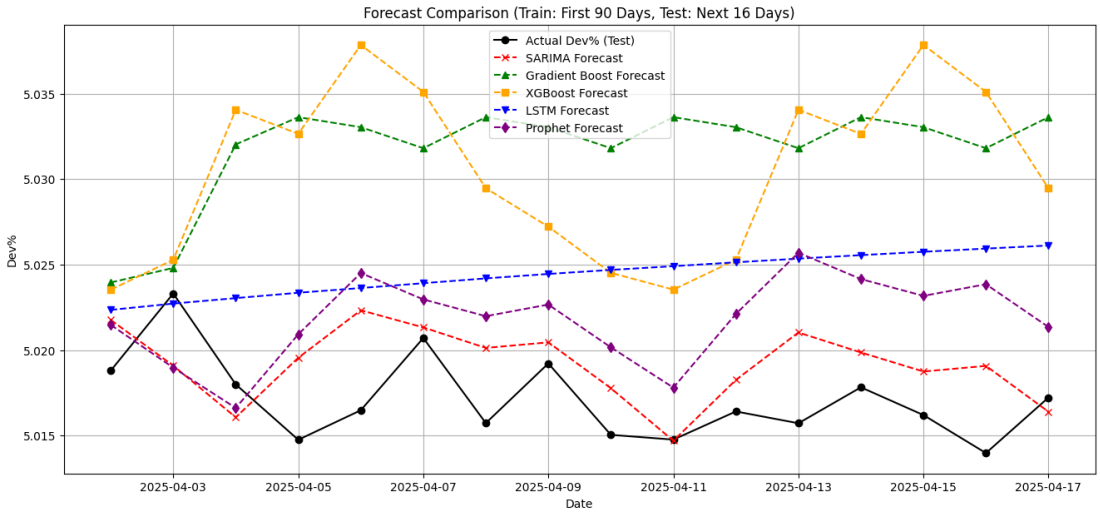
Çizelge 4.2. Algoritma Tahminleri

Date	Actual	SARIMA	Gradient Boosting	XGBoost	LSTM	Prophet
2025-04-03	5,058811	5,05195	5,109384	5,069475	5,084706	5,05728
2025-04-04	5,045905	5,037289	5,125393	5,051239	5,084494	5,060132
2025-04-05	5,070785	5,041631	5,167199	5,077166	5,084203	5,101586
2025-04-06	5,056441	5,053168	5,364616	5,095863	5,083966	5,074248
2025-04-07	5,075114	5,048853	5,152168	5,083737	5,083771	5,099466
2025-04-08	5,060743	5,061641	5,149077	5,102497	5,083611	5,103016
2025-04-09	5,053457	5,045417	5,105348	5,064007	5,08349	5,074467
2025-04-10	5,070305	5,038319	5,122532	5,064942	5,083397	5,062331
2025-04-11	5,05955	5,025271	5,170737	5,076714	5,083336	5,065182
2025-04-12	5,056198	5,029602	5,364616	5,076926	5,083307	5,106637
2025-04-13	5,043788	5,042925	5,152168	5,076926	5,083278	5,079298
2025-04-14	5,08054	5,03712	5,149077	5,076926	5,083252	5,104516
2025-04-15	5,071457	5,050337	5,105348	5,076926	5,083232	5,108067
2025-04-16	5,062213	5,035846	5,122532	5,076926	5,083217	5,079518
2025-04-17	5,068709	5,029187	5,170737	5,076926	5,083205	5,067381
2025-04-18	5,046219	5,01624	5,364616	5,076926	5,083197	5,070233

Çizelge 4.3. Algoritma Süre Karşılaştırması

Model	Süre (s)
SARIMA	0,754265
Gradient Boosting	0,098349
XGBoost	0,042341
LSTM	9,815178
Prophet	0,506953

8800.85.04:



Şekil 4.2. Motor 8800.85.04 Tahmin Verileri

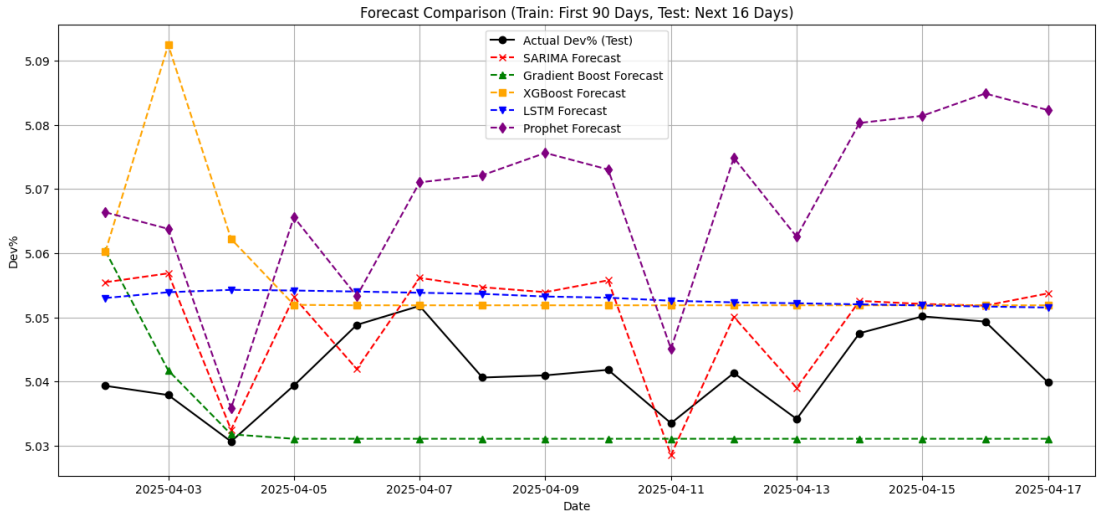
Çizelge 4.4. Algoritma Tahminleri

Date	Actual	SARIMA	Gradient Boosting	XGBoost	LSTM	Prophet
2025-04-02	5,018796	5,021763	5,023954	5,023537	5,022351	5,021473
2025-04-03	5,02331	5,019085	5,024809	5,025282	5,022719	5,018971
2025-04-04	5,01799	5,016064	5,032036	5,034061	5,02304	5,016612
2025-04-05	5,01475	5,019549	5,03363	5,032651	5,023354	5,020922
2025-04-06	5,016478	5,022328	5,033043	5,037853	5,023631	5,024508
2025-04-07	5,020706	5,021325	5,031813	5,035107	5,023912	5,022965
2025-04-08	5,015726	5,020125	5,03363	5,029468	5,024195	5,021982
2025-04-09	5,019218	5,020451	5,033043	5,027221	5,024455	5,022662
2025-04-10	5,015038	5,017769	5,031813	5,024519	5,024691	5,02016
2025-04-11	5,01476	5,014692	5,03363	5,023537	5,024909	5,017801
2025-04-12	5,016408	5,018255	5,033043	5,025282	5,025136	5,022112
2025-04-13	5,015717	5,021028	5,031813	5,034061	5,025351	5,025697
2025-04-14	5,01782	5,019858	5,03363	5,032651	5,025556	5,024154
2025-04-15	5,016197	5,018745	5,033043	5,037853	5,025752	5,023172
2025-04-16	5,013973	5,019074	5,031813	5,035107	5,025939	5,023851
2025-04-17	5,017221	5,016391	5,03363	5,029468	5,026117	5,02135

Çizelge 4.5. Algoritma Süre Karşılaştırması

Model	Süre (s)
SARIMA	0,754265
Gradient Boosting	0,098349
XGBoost	0,042341
LSTM	9,815178
Prophet	0,506953

8800.87.06:



Şekil 4.3. Motor 8800.83.06 Tahmin Verileri

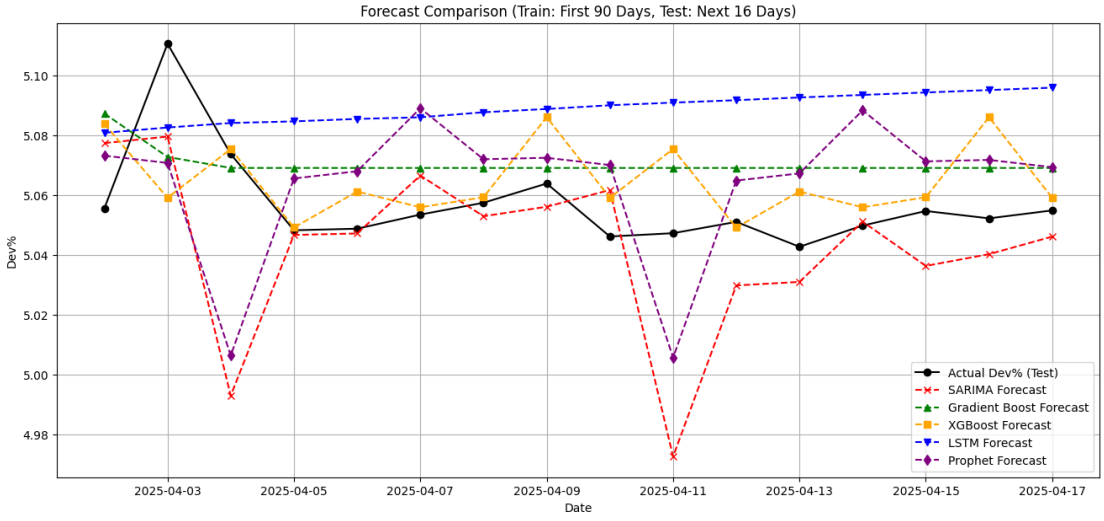
Çizelge 4.6. Algoritma Tahminleri

Date	Actual	SARIMA	Gradient Boosting	XGBoost	LSTM	Prophet
2025-04-02	5,039318	5,055459	5,060333	5,060207	5,053	5,066363
2025-04-03	5,037885	5,056852	5,041726	5,092449	5,053905	5,063777
2025-04-04	5,030641	5,032471	5,031756	5,062213	5,054276	5,035828
2025-04-05	5,039364	5,053154	5,031067	5,051939	5,054174	5,065606
2025-04-06	5,048816	5,041929	5,031067	5,051868	5,054009	5,053286
2025-04-07	5,051775	5,056141	5,031067	5,051868	5,053822	5,07103
2025-04-08	5,040609	5,054682	5,031067	5,051868	5,053648	5,072146
2025-04-09	5,040952	5,053916	5,031067	5,051868	5,053235	5,075618
2025-04-10	5,04181	5,05576	5,031067	5,051868	5,053054	5,073032
2025-04-11	5,03349	5,028509	5,031067	5,051868	5,052582	5,045083
2025-04-12	5,041314	5,050071	5,031067	5,051868	5,052324	5,074861
2025-04-13	5,034148	5,038993	5,031067	5,051868	5,052187	5,06254
2025-04-14	5,047519	5,052552	5,031067	5,051868	5,05202	5,080285
2025-04-15	5,050141	5,052075	5,031067	5,051868	5,051846	5,0814
2025-04-16	5,049329	5,051819	5,031067	5,051868	5,051678	5,084873
2025-04-17	5,039846	5,053726	5,031067	5,051868	5,051519	5,082287

Çizelge 4.7. Algoritma Süre Karşılaştırması

Model	Süre (s)
SARIMA	1,251947
Gradient Boosting	0,079847
XGBoost	0,040341
LSTM	10,321755
Prophet	0,429812

8800.85.07:



Şekil 4.4. Motor 8800.85.07 Tahmin Verileri

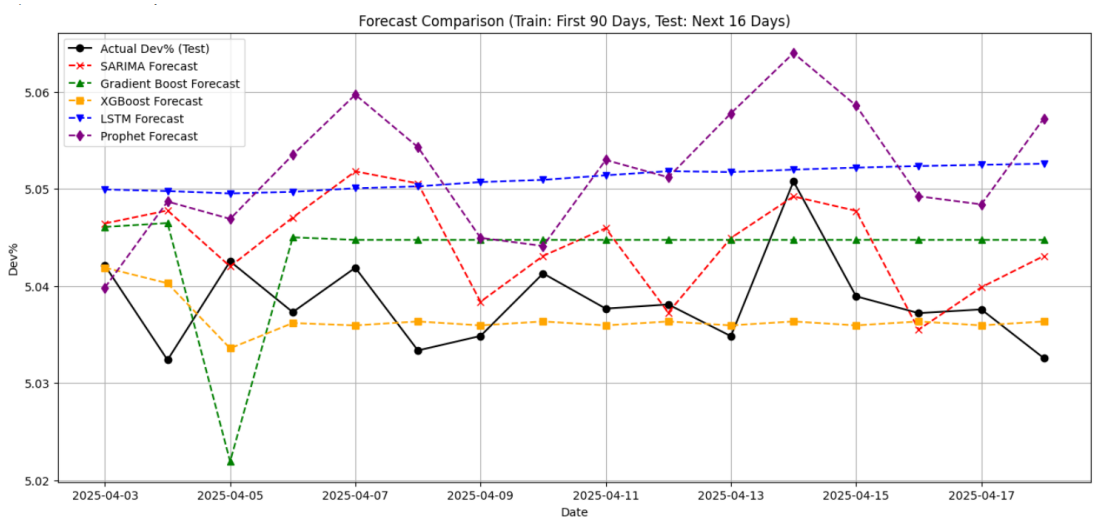
Çizelge 4.8. Algoritma Tahminleri

Date	Actual	SARIMA	Gradient Boosting	XGBoost	LSTM	Prophet
2025-04-02	5,055497	5,077313	5,08719	5,083713	5,080769	5,073112
2025-04-03	5,110547	5,079493	5,072689	5,059024	5,08252	5,070685
2025-04-04	5,073634	4,992871	5,069009	5,075333	5,084033	5,0063
2025-04-05	5,048178	5,046616	5,069009	5,049075	5,084577	5,065499
2025-04-06	5,048696	5,0471	5,069009	5,061124	5,085398	5,067896
2025-04-07	5,053414	5,066468	5,069009	5,055862	5,085937	5,08897
2025-04-08	5,057368	5,05286	5,069009	5,059215	5,087624	5,071916
2025-04-09	5,063806	5,05604	5,069009	5,08597	5,088713	5,072403
2025-04-10	5,046097	5,061635	5,069009	5,059024	5,089965	5,069976
2025-04-11	5,047192	4,97252	5,069009	5,075333	5,090847	5,00559
2025-04-12	5,050977	5,029726	5,069009	5,049075	5,09166	5,06479
2025-04-13	5,042661	5,030913	5,069009	5,061124	5,092562	5,067186
2025-04-14	5,049742	5,051038	5,069009	5,055862	5,09342	5,088261
2025-04-15	5,05461	5,036256	5,069009	5,059215	5,094238	5,071207
2025-04-16	5,052157	5,040179	5,069009	5,08597	5,095049	5,071693
2025-04-17	5,054806	5,046055	5,069009	5,059024	5,095845	5,069266

Çizelge 4.9. Algoritma Süre Karşılaştırması

Model	Süre (s)
SARIMA	0,444668
Gradient Boosting	0,170339
XGBoost	0,040671
LSTM	10,15023
Prophet	0,56906

8800.83.05:



Şekil 4.5. Motor 8800.83.06 Tahmin Verileri

Çizelge 4.10. Algoritma Tahminleri

Date	Actual	SARIMA	Gradient Boosting	XGBoost	LSTM	Prophet
2025-04-03	5,042095	5,046435	5,046073	5,041883	5,049928	5,03983
2025-04-04	5,032379	5,047772	5,046474	5,040277	5,049768	5,048689
2025-04-05	5,042531	5,042006	5,021925	5,033576	5,049524	5,046903
2025-04-06	5,037302	5,047042	5,045004	5,036178	5,049688	5,053489
2025-04-07	5,041874	5,0518	5,04474	5,035936	5,050054	5,059694
2025-04-08	5,033357	5,050553	5,04474	5,036342	5,050252	5,05429
2025-04-09	5,034847	5,038371	5,04474	5,035936	5,050697	5,044952
2025-04-10	5,041286	5,043091	5,04474	5,036342	5,050921	5,044113
2025-04-11	5,037666	5,04594	5,04474	5,035936	5,051389	5,052971
2025-04-12	5,038093	5,037254	5,04474	5,036342	5,051817	5,051185
2025-04-13	5,034831	5,044935	5,04474	5,035936	5,051726	5,057772
2025-04-14	5,050759	5,049217	5,04474	5,036342	5,051974	5,063976
2025-04-15	5,038933	5,047732	5,04474	5,035936	5,052182	5,058572
2025-04-16	5,037198	5,035502	5,04474	5,036342	5,052343	5,049234
2025-04-17	5,037585	5,039856	5,04474	5,035936	5,052473	5,048395
2025-04-18	5,032565	5,043085	5,04474	5,036342	5,052583	5,057253

Çizelge 4.11. Algoritma Süre Karşılaştırması

Model	Süre (s)
SARIMA	0,569418
Gradient Boosting	0,096574
XGBoost	0,105115
LSTM	9,545012
Prophet	0,421988

5. TARTIŞMA ve SONUÇ

Gerçekleştirilen analizlerde makine öğrenmesi ve derin öğrenme modellerinin kullanımının, literatürde rapor edilen geliştirmelere paralel olarak kestirimci bakım performansını iyileştirdiği gözlemlenmiştir. LSTM, XGBoost gibi modellerden elde edilen sonuçlar, yapay zekâ temelli yaklaşımların tahmin doğruluğunu artırdığına dair literatürdeki bulgularla uyumludur. Elde edilen RMSE ve MAE değerleri, benzer endüstriyel uygulamalarda bildirilmiş değerlerle paralellik göstermekte ve algoritmaların arıza tahmin performansının literatürdeki başarı öykülerini desteklediği görülmektedir. Örneğin, derin öğrenme yöntemlerinin yüksek doğruluk ve ölçeklenebilirlik sunduğu; makine öğrenmesi yöntemlerinin ise göreceli olarak hesaplama maliyetinde avantaj sağladığı literatürde vurgulanmıştır (Ucar vd., 2024).

Bu çalışmada elde edilen bulgular, uygulanan makine ve derin öğrenme modellerinin literatürdeki başarılarını teyit etmektedir. İlerleyen zamanlarda bu modellerin şeffaflığını artırmak amacıyla açıklanabilir yapay zekâ tekniklerinin entegrasyonu araştırılmalıdır. Ayrıca gerçek zamanlı kestirimci bakım uygulamaları için IoT verilerinin daha geniş bir ölçekte kullanılması ve dijital ikiz, bulut/edge bilişim altyapısı gibi endüstri 4.0 teknolojilerinin entegrasyonu değerlendirilebilir. Yeni nesil sensörlerin (titreşim, termal, akustik vb.) ve çoklu veri kaynaklarının kombinasyonu, arıza teşhislerinin doğruluğunu ve genellenebilirliğini artırabilir. Sonuç olarak, işbirlikli robotlar ve dijital ikiz gibi yenilikçi uygulamalarla bakım süreçlerinin otonom hale getirilmesi ve optimizasyonu, gelecekte araştırılması gereken önemli konular arasında yer almaktadır (Ucar vd., 2024).

Tez çalışmasında apılan analizlerde SARIMA, Gradient Boost, XGBoost, LSTM ve Prophet algoritmalarının birden fazla motor üzerinde tutarlı tahmin yapabildiği görülmüştür. Çalışmada test verisi üç aylık kullanılmıştır ve her motor için ortalama 250 bin satırdan oluşmaktadır. Çalışmada motorlar ayrı ayrı analiz edildiği ve farklı olmasına rağmen genelde sabit hızlarda çalıştıklarından dolayı algoritmalara kategorik veri beslenmemiştir. Bu durum Gradient Boost ve XGBoost algoritmalarının kategorik veri üzerindeki avantajlarını ortadan kaldırmıştır.

Gradient Boost algoritması uzun verilerde aşırı öğrenme ihtimalinden dolayı motorlardan biri üzerinde yüksek RMSE ve MAE değerleriyle tahmin yapabilmiştir. Motor bakım aralıkları sebebiyle veri uzunluğu fazla olacaktır ve bu sebeple uzun zaman serilerinde yalnızca Gradient Boost algoritması kullanımı yanıltıcı olabilir.

XGBoost algoritması yakın değerler yakalasa da desenleri tahmin etmede zayıf kalmıştır. Bunun sebebi eksik veri tamamlama özelliğinin getirdiği veri düzenlileştirmesidir. Tesis üzerinden elde ettiğimiz veride boşluklar olmadığı için XGBoost yöntemi algoritmasının getirdiği avantajları bu analiz üzerinde kullanamamıştır.

Yeni toplanan veriler ile düzenli olarak SARIMA, XGBoost ve Prophet algoritmalarının birlikte kullanımı hem yakın değerler tahmin edilmesini hem de desenleri yakalayarak tepe değerlerin atlanmamasını sağlayacaktır. Analiz süresini kısaltmak için veriler günlük ortalama alınarak analiz edilmiştir, fakat saatlik yapılan analizlerde RMSE ve MAE değerleri aynı kalmıştır. Bu sebeple tahmin edilen Dev% (Sapma Yüzdesi) %7'nin (Tesisten tesise farklılık göstermek üzere hata sınırı %7 ile %15 arasında belirlenmektedir) üzerine çıktığında analiz saatlik ortalama ile tekrarlanarak hatanın gerçekleşeceği zamanın tutarlılığı arttırılabilir (All Sciences Academy, 2025).

Günümüz depo ve dağıtım sistemlerinde çok sayıda motor olduğu için işlemci kullanım verimliliği sebebiyle ön alarm sistemi kurulabilir. Bu durumda periyodik analiz aralığı uzatılır ve belirlenen ön sınır aşıldığında tahmin edilen tarihe göre motor bazlı detaylı analiz yapılabilir.
İstatistiksel modellerin, makine öğrenmesi ve derin öğrenme algoritmalarının analizlerde kullanılması planlanan bakımlardan odak noktasını doğru seçip bakım verimliliğini arttıracaktır. İlerleyen zamanlarda algoritmaya kategorik veriler eklenip arızanın olacağı tarih tahminine ilave kök neden analizi de yapılabilir.

KAYNAKLAR

Aldarondo, F. J., & Bozer, Y. A. (2020). Expected distances and alternative design configurations for automated guided vehicle-based order picking systems. *International Journal of Production Research*, *60*(4), 1298–1315. https://doi.org/10.1080/00207543.2020.1856438

All Sciences Academy. (2025). Predictive Maintenance in E-Commerce Distribution Systems. *Zenodo*. https://doi.org/10.5281/zenodo.15549040

Bontempi, G., Ben Taieb, S., & Le Borgne, Y.-A. (2013). Machine Learning Strategies for Time Series Forecasting. *Business Intelligence*, *138*, 62–77. https://doi.org/10.1007/978-3-642-36318-4_3

Carvalho, D. V., Pereira, E. M., & Cardoso, J. S. (2019). Machine Learning Interpretability: A Survey on Methods and Metrics. *Electronics*, *8*(8), 832. https://doi.org/10.3390/electronics8080832

Chen, T., & Guestrin, C. (2016). XGBoost: a Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, *1*(1), 785–794. https://doi.org/10.1145/2939672.2939785

Eang, C., & Lee, S. (2025). Predictive Maintenance and Fault Detection for Motor Drive Control Systems in Industrial Robots Using CNN-RNN-Based Observers. *Sensors*, *25*(1), 25. https://doi.org/10.3390/s25010025

Ersöz, O. Ö., İnal, A. F., Aktepe, A., Türker, A. K., & Ersöz, S. (2022). A Systematic Literature Review of the Predictive Maintenance from Transportation Systems Aspect. *Sustainability*, *14*(21), 14536. https://doi.org/10.3390/su142114536

Fausing Olesen, J., & Shaker, H. R. (2020). Predictive Maintenance for Pump Systems and Thermal Power Plants: State-of-the-Art Review, Trends and Challenges. *Sensors*, *20*(8), 2425. https://doi.org/10.3390/s20082425

Henrique, A., Ângelo, M., Silva, & Francisco, G. (2021). A framework to automate fault detection and diagnosis based on moving window principal component analysis and Bayesian network. *Reliability Engineering & System Safety*, *215*, 107837–107837. https://doi.org/10.1016/j.ress.2021.107837

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Kumar Dubey, A., Kumar, A., García-Díaz, V., Kumar Sharma, A., & Kanhaiya, K. (2021). Study and analysis of SARIMA and LSTM in forecasting time series data. *Sustainable Energy Technologies and Assessments*, *47*, 101474. https://doi.org/10.1016/j.seta.2021.101474

Kvak, K., & Straka, M. (2024). The Use of the Internet of Things in the Distribution Logistics of Consumables. *Applied Sciences*, *14*(8), 3263–3263. https://doi.org/10.3390/app14083263

Lekidis, A., Georgakis, A., Dalamagkas, C., & Papageorgiou, E. I. (2024). Predictive Maintenance Framework for Fault Detection in Remote Terminal Units. *Forecasting*, *6*(2), 239-265. https://doi.org/10.3390/forecast6020014

Li, J., Xu, A., Li, Q., Wang, C., Yu, H., Zhou, X., & Geng, Y. (2022). Application of XGBoost algorithm in the optimization of pollutant concentration. *Atmospheric Research*, *276*, 106238–106238. https://doi.org/10.1016/j.atmosres.2022.106238

Li, Y., Zhang, R., & Jiang, D. (2022). Order-Picking Efficiency in E-Commerce Warehouses: A Literature Review. *Journal of Theoretical and Applied Electronic Commerce Research*, *17*(4), 1812–1830. https://doi.org/10.3390/jtaer17040091

Molęda, M., Małysiak-Mrozek, B., Ding, W., Sunderam, V., & Mrozek, D. (2023). From Corrective to Predictive Maintenance—A Review of Maintenance Approaches for the Power Industry. Sensors, 23(13), 5970. https://doi.org/10.3390/s23135970

Parizad, A., & Hatziadoniu, C. J. (2021). *Using Prophet Algorithm for Pattern Recognition and Short Term Forecasting of Load Demand Based on Seasonality and Exogenous Features*. https://doi.org/10.1109/naps50074.2021.9449743

Rana, S. (2025). AI-DRIVEN FAULT DETECTION AND PREDICTIVE MAINTENANCE IN ELECTRICAL POWER SYSTEMS: A SYSTEMATIC REVIEW OF DATA-DRIVEN APPROACHES, DIGITAL TWINS, AND SELF-HEALING GRIDS. *American Journal of Advanced Technology and Engineering Solutions*, *1*(01), 258–289. https://doi.org/10.63125/4p25x993

Rigas, S., Tzouveli, P., & Kollias, S. (2024). An End-to-End Deep Learning Framework for Fault Detection in Marine Machinery. *Sensors*, *24*(16), 5310. https://doi.org/10.3390/s24165310

Serradilla, O., Zugasti, E., Rodriguez, J., & Zurutuza, U. (2022). Deep learning models for predictive maintenance: a survey, comparison, challenges and prospects. *Applied Intelligence*, *52*(10), 10934–10964. https://doi.org/10.1007/s10489-021-03004-y

Shiri, F. M., Perumal, T., Mustapha, N., & Mohamed, R. (2023, June 1). *A Comprehensive Overview and Comparative Analysis on Deep Learning Models: CNN, RNN, LSTM, GRU*. ArXiv.org. https://doi.org/10.48550/arXiv.2305.17473

Taylor, S. J., & Letham, B. (2018). Forecasting at Scale. *The American Statistician*, *72*(1), 37–45. https://doi.org/10.1080/00031305.2017.1380080

Tubis, A. A., & Rohman, J. (2023). Intelligent Warehouse in Industry 4.0—Systematic Literature Review. *Sensors*, *23*(8), 4105. https://doi.org/10.3390/s23084105

Ucar, A., Karakose, M., & Kırımça, N. (2024). Artificial Intelligence for Predictive Maintenance Applications: Key Components, Trustworthiness, and Future Trends. *Applied Sciences*, *14*(2), 898. mdpi. https://doi.org/10.3390/app14020898

Vita, V., Fotis, G., Chobanov, V., Pavlatos, C., & Mladenov, V. (2023). Predictive Maintenance for Distribution System Operators in Increasing Transformers' Reliability. *Electronics*, *12*(6), 1356. https://doi.org/10.3390/electronics12061356

Zhen, L., & Li, H. (2022). A Literature Review of Smart Warehouse Operations Management. *Frontiers of Engineering Management*, *9*(1), 31–55. https://doi.org/10.1007/s42524-021-0178-9

Zheng, H., Paiva, A. R., & Gurciullo, C. S. (2020). *Advancing from Predictive Maintenance to Intelligent Maintenance with AI and IIoT*. ArXiv.org. https://arxiv.org/abs/2009.00351

Zonta, T., da Costa, C. A., da Rosa Righi, R., de Lima, M. J., da Trindade, E. S., & Li, G. P. (2020). Predictive maintenance in the Industry 4.0: A systematic literature review. *Computers & Industrial Engineering*, *150*, 106889. https://doi.org/10.1016/j.cie.2020.106889