ARCENG
INTERNATIONAL JOURNAL OF ARCHITECTURE AND ENGINEERING

ISSN
2822-6895

Article Arrival Date

30.11.2025

Article Published Date

20.12.2025

Oracle PL/SQL Sorgularının Yapay Zekâ Tabanlı Performans Optimizasyonu

Artificial Intelligence-Based Performance Optimization of Oracle PL/SQL Queries

Gökhan TOPSAKAL [1] **Önder ŞAHİNASLAN** [2]

[1]Bilgisayar Mühendisi, Migros Ticaret A.Ş, Bilgi Teknolojileri, Orcid: 0009-0007-9219-4226

[2]Dr. Ögr. Üyesi, Maltepe Üniversitesi, Bilişim Bölümü, Orcid: 0000-0003-2695-5078

Özet

Bu çalışma, Oracle PL/SQL ve Forms tabanlı uygulamalarda yavaş çalışan sorguları analiz ederek performansı artırmayı amaçlayan yapay zekâ destekli bir optimizasyon sistemi geliştirmeyi hedeflemektedir. Araştırmada, Oracle veri tabanında çalıştırılan sorguların performans verileri SQL_TRACE ve EXPLAIN PLAN kullanılarak toplanmış, Python ortamında analiz edilmiştir. Çalışma süresi, mantıksal okuma ve I/O işlemleri gibi metriklere dayalı özellik seçimiyle veri seti oluşturulmuş; sorgu yavaşlığının nedenlerini belirlemek için Random Forest ve XGBoost algoritmaları uygulanmıştır. Tarihsel performans kayıtlarıyla eğitilen modellerin doğruluğu çeşitli metriklerle değerlendirilmiş, sistem gerçek ortamdan alınan yeni sorgularla test edilerek öneri yeteneği geliştirilmiştir. Sonuçlar, önerilen sistemin sorgu yürütme süresini %82,4, mantıksal okuma oranını %84,8, fiziksel okuma oranını %90,9 ve toplam Oracle maliyetini %97 oranında iyileştirdiğini göstermektedir. Model karşılaştırmasında XGBoost, %96,1 doğruluk ve F1 skoru ile daha yüksek sınıflandırma başarısı sergilerken, Random Forest daha hızlı tahmin süreleri sağlamıştır. Bu çalışma, Oracle PL/SQL performans sorunlarını yapay zekâ ile analiz eden özgün bir sistem sunarak veri tabanı yöneticilerine karar desteği sağlamaktadır.

Anahtar kelimeler: Oracle, PL/SQL, Performans Optimizasyonu, Yapay Zeka, SQL Analizi.

Abstract

This study aims to develop an artificial intelligence–based optimization system to analyze and improve the performance of slow-running queries in Oracle PL/SQL and Forms-based applications. Performance data from Oracle queries were collected using SQL_TRACE and EXPLAIN PLAN and analyzed in a Python environment. A dataset was constructed through feature selection based on metrics such as execution time, logical reads, and I/O operations. Random Forest and XGBoost algorithms were applied to identify factors contributing to query slowness, with historical performance records used for model training and evaluation through standard performance metrics. The system was further refined and validated using real-world queries to enhance its recommendation capability. Results indicate substantial improvements: execution time reduced by 82.4%, consistent read rate by 84.8%, physical read rate by 90.9%, and total Oracle cost by 97%. In model comparison, XGBoost achieved superior classification accuracy with 96.1% accuracy and F1-score, while Random Forest provided faster prediction times. This research introduces a novel AI-driven system for diagnosing and optimizing Oracle

ISSN
2822-6895

PL/SQL performance issues, offering decision support for database administrators and contributing to improved query efficiency.

Keywords: Oracle, PL/SQL, Performance Optimization, AI, SQL Analysis

1. GİRİŞ

Dijital dönüşüm çağında, veri yönetimi ile verilerin etkin ve güvenilir bir şekilde işlenmesi, kuruluşlar için stratejik açıdan kritik bir önem kazanmıştır. Artan veri hacmi, iş süreçlerinin giderek karmaşıklaşması ve erişilebilirlik talebinin yükselmesi, ilişkisel veritabanı yönetim sistemlerinin (RDBMS) performansını hayati bir unsur haline getirmiştir. Oracle gibi kurumsal düzeydeki sistemlerde, sorgu yürütme süresinin azaltılması ve sistem kaynaklarının verimli kullanılması, uygulamaların genel başarısını doğrudan etkilemektedir.

Oracle PL/SQL ve Oracle Forms, kurumsal uygulamalarda veri işleme ve iş mantığının kontrolünde temel bir rol üstlenmekte, bu yapılar sistemin omurgasını oluşturmaktadır. Ancak, sistem mimarilerinin zamanla genişlemesiyle birlikte performans sorunlarının ortaya çıkması kaçınılmazdır. Yavaş sorgular, tam tablo taramaları ve verimsiz indeks kullanımı gibi problemler, hem kullanıcı deneyimini hem de sistemin genel verimliliğini olumsuz yönde etkileyebilmektedir.

Bu çalışma, Oracle PL/SQL ve Forms ortamlarında karşılaşılan performans darboğazlarını belirlemek, analiz etmek ve iyileştirmek amacıyla gerçekleştirilmiştir. Geleneksel performans iyileştirme yöntemlerinin ötesine geçilerek, yapay zeka (YZ) ve makine öğrenmesi (ML) teknikleriyle desteklenen bir sistem geliştirilmiştir. Bu yapı, Python programlama dili ile entegre edilerek SQL sorgu performans izlerini analiz etmekte ve otomatik optimizasyon önerileri üretmektedir.

Araştırmanın temel amacı, Oracle tabanlı uygulamalarda performansı artırmaya yönelik bilimsel temelli ve pratik olarak uygulanabilir bir çözüm önermek ve bu çözümü, veritabanı performans optimizasyonu için YZ destekli bir karar destek mekanizmasına dönüştürmektir.

2. LİTERATÜR TARAMASI

Bu bölümde, ilişkisel veritabanı yönetim sistemlerinde performans optimizasyonu, Oracle PL/SQL ve Forms tabanlı uygulamalarda karşılaşılan yaygın performans sorunları ile bu sorunların giderilmesine yönelik geliştirilen yöntemler ele alınmıştır. Ayrıca, son yıllarda veritabanı performans yönetiminde yapay zeka ve makine öğrenmesi tekniklerinin kullanımına odaklanan çalışmalar incelenerek, mevcut yaklaşımların güçlü ve sınırlı yönleri ortaya konmuştur. Bu bağlamda, çalışmanın konumlandığı bilimsel çerçeve belirlenmiş ve önerilen yaklaşımın literatürdeki yeri değerlendirilmiştir.

(Zongheng Yang, 2022) geliştirdiği Balsa adlı sistemde olduğu gibi, klasik sorgu optimizasyon süreçleri yerine makine öğrenmesine dayalı yeni nesil yaklaşımlarla, veritabanı sorgularının yapısal örüntüleri üzerinden optimizasyon stratejileri öğrenilebilmekte ve bu stratejiler uzman müdahalesi olmaksızın sistemler tarafından uygulanabilmektedir. Böylece, hem manuel hatalar en aza indirgenmekte hem de veri tabanı sistemlerinin adaptif ve dinamik şekilde kendini optimize edebilmesi mümkün hale gelmektedir.

Bu yaklaşım sayesinde, veritabanı yöneticilerinin veya geliştiricilerin manuel müdahalesine gerek kalmaksızın; veritabanı sorgularının gerçek zamanlı performans değerleri üzerinden değerlendirilmesi, potansiyel darboğazların önceden belirlenmesi ve iyileştirme yollarının önerilmesi hedeflenmektedir.

Özellikle (Ryan Marcus, 2020) Learning to Steer Query Optimizers çalışmasında, karmaşık sorgu iyileştirme sürecine makine öğrenmesi (takviye öğrenmesi ve sinir ağları) uygulanarak büyük performans kazanımları elde edildiği görülmektedir. Makine öğrenmesi sınıflandırma modellerini kullanarak youtube verileri üzerinde duygu analizi yapabilen başarılı bir örnek çalışmadır(Şahinaslan, 2022).

(Asokan, 2025) Veri tabanı performansı üzerinde yapay zekâ destekli optimizasyonun, otomatik indeksleme ve anomalilerin tespitiyle birlikte organizasyonlara hem hız hem de verimlilik kazandırdığını belirtmektedir. Makine öğrenmesi algoritmalarının gerçek bir süreçte uygulanmasını incelemiş ve metodolojik olarak YZ-tabanlı sistemlerin performans kazanımlarına dair kanıtlar ortaya koymuştur(Şahinaslan, 2023.

Random Forest, birden fazla karar ağacının (decision tree) birleşiminden oluşan topluluk öğrenme (ensemble learning) yöntemidir. Her bir karar ağacı, rastgele örneklenen veri alt kümeleri üzerinde eğitilir ve modelin çıktısı, tüm ağaçların oylamasıyla belirlenir. Bu algoritma, overfitting riskini minimize ederken yüksek doğruluk sağlar. SQL sorgularının sahip olduğu çeşitli performans metrikleri (örneğin: elapsed_time, buffer_gets, physical_reads) modelin giriş değişkenleri olarak kullanılmış, etiket ise "optimizasyon önerisi gerektiriyor mu?" sorusunun cevabı olarak belirlenmiştir.

Bu çalışma kapsamında kullanılan Random Forest tabanlı sınıflandırma yaklaşımı, (Dhanara, 2021)tarafından önerilen "Random Forest Bagging X-Means SQL Query Clustering" yöntemine dayanmaktadır. Bu yöntem, SQL günlüklerinden anomali sorguları başarılı biçimde belirlemek amacıyla Random Forest ile öğrenme sağlamakta, ardından X-Means kümelemesi ve majority voting ile kötü yapılandırılmış sorguları ayırt edebilmektedir. XGBoost, Chen ve Guestrin (2016) tarafından geliştirilen, denetimli öğrenme problemlerinde yüksek doğruluk sunan, ağaç tabanlı bir topluluk (ensemble) makine öğrenmesi yöntemidir. Özellikle büyük veri setlerinde ve yüksek boyutlu problemlerde hızlı çalışması ve overfitting'e karşı direnci sayesinde günümüzde akademik ve endüstriyel birçok çalışmada tercih edilmektedir. XGBoost algoritması, karar ağaçları üzerinde gradyan artırma (gradient boosting) tekniğini optimize eden, paralel işlemeye uygun, düzenlileştirme (regularization) içeren gelişmiş bir versiyonudur. Bu sayede hem eğitim süresi kısalmakta hem de modelin genellenebilirliği artmaktadır. (Chen, XGBoost: A Scalable Tree Boosting System., 2016), çalışmada XGBoost'un temel algoritması sunulmuş, GBM (Gradient Boosting Machine) algoritmasına yapılan optimizasyonlar detaylı olarak açıklanmıştır. Özellikle sparse veri yapıları, paralel işleme kabiliyeti ve düzenlileştirme mekanizmaları gibi performans artırıcı teknikler ön plana çıkarılmıştır.

2021 yılında yapılan bir tez çalışmasında, ağ tabanlı saldırı tespiti problemlerinde XGBoost ve Rastgele Orman algoritmaları karşılaştırılmıştır. XGBoost'un daha yüksek doğruluk ve F1-Skoru sağladığı, ancak eğitim süresi bakımından biraz daha fazla kaynak tükettiği vurgulanmıştır (Buldu, 2021). Veritabanı yönetimi ve kurtarma tekniklerini ele alan ve ilişkisel veritabanlarının kritik operasyonlarına odaklanan bir diğer çalışma da veri kaybı-önleme/pratik sorun çözümleri sunmuştur(Sahinaslan, 2022)

Özetle, literatürde yer alan çalışmalar; veritabanı sorgu optimizasyonu ve performans yönetimi alanında yapay zeka ve makine öğrenmesi tabanlı yaklaşımların, geleneksel kural tabanlı ve manuel yöntemlere kıyasla daha ölçeklenebilir, adaptif ve yüksek doğruluklu çözümler sunduğunu ortaya koymaktadır. Özellikle Random Forest ve XGBoost gibi ağaç tabanlı topluluk öğrenme yöntemlerinin, sorgu performans metriklerini etkin biçimde analiz ederek anomali tespiti ve optimizasyon kararlarında başarılı sonuçlar verdiği görülmektedir. Bu bağlamda, mevcut literatürden elde edilen bulgular ışığında, bu çalışmada Oracle PL/SQL ve Forms ortamlarına özgü performans izlerinin YZ destekli modellerle analiz edilmesi ve otomatik optimizasyon önerileri üretilmesi hedeflenerek, literatürdeki yaklaşımları kurumsal

ISSN
2822-6895

veritabanı sistemlerine uyarlayan bütüncül ve uygulanabilir bir çözüm sunulması amaçlanmaktadır.

3. MATERYAL

Çalışmanın deneysel altyapısı, kullanılan yazılım ve donanım bileşenleri ile performans ölçüm ve veri toplama sürecine ilişkin ayrıntılar sunulmaktadır. Oracle tabanlı kurumsal veritabanı sistemlerinde sorgu performansının gerçekçi ve tekrarlanabilir koşullar altında değerlendirilebilmesi amacıyla, kontrollü bir deneysel ortam oluşturulmuş ve bu ortamda farklı sorgu senaryoları sistematik olarak test edilmiştir. Elde edilen performans izleri, yapay zeka ve makine öğrenmesi tabanlı modellerin eğitimi ve değerlendirilmesi için girdi verisi olarak kullanılmıştır. Bu kapsamda, deneysel ortamda kullanılan teknolojiler ve bileşenler Tablo 1'de özetlenmektedir. Bu çalışma kapsamında Oracle 18c Express Edition üzerinde yapılandırılmış bir veritabanı ortamı kullanılmıştır. Bu ortamda oluşturulan tablolar, sorgu performansının gerçekçi koşullar altında gözlemlenebilmesi için milyonlarca kayıtla doldurulmuştur. Performans ölçümleri, Oracle'ın SQL_TRACE çıktıları ve EXPLAIN PLAN komutları kullanılarak elde edilmiştir. Ölçümler sırasında Elapsed Time, Consistent Reads (CR), Physical Reads (PR) ve Total Cost gibi temel performans metrikleri değerlendirilmiştir. Python ile Oracle arasında oracledb kütüphanesi aracılığıyla bağlantı kurulmuş ve toplanan performans verileri makine öğrenmesi modellerinin eğitimi için kullanılmıştır.

Veritabanı şeması, aşağıdaki tabloları birincil veri kaynakları olarak içermektedir:

T1_CUSTOMERS (500.000 kayıt)

T2_ORDERS (500.000 kayıt)

T3_PRODUCTS (500.000 kayıt)

T4_ORDER_DETAILS (1.000.000 kayıt)

Bu tablolar; müşteri, sipariş, ürün ve sipariş detaylarını kapsayarak gerçekçi iş senaryolarının simülasyonunu mümkün kılmıştır.

Testlerin Oracle'ın standart yapılandırmaları (bellek, listener, arka plan süreçleri vb.) altında gerçekleştirilmesinin, diğer sürümlerde benzer sonuçlar vereceği varsayılmıştır. Ayrıca, SQL_TRACE ve EXPLAIN PLAN çıktılarının sorgu performansını doğru şekilde yansıttığı kabul edilmiştir. Modelleme sürecinde, Random Forest ve XGBoost algoritmalarının yeterli çeşitlilikte yavaş ve hızlı sorgu örnekleriyle eğitildiği varsayılmıştır. Bu varsayım, yapay zeka modellerinin yeni sorgular için güvenilir tahminler sunabileceği beklentisini desteklemektedir.

Tablo 1. Deneysel ortamlar

Teknoloji	Açıklama
Oracle XE 18c	Deneysel Veritabanı Ortamı
Python 3.11+	Veri İşleme ve Model Eğitimi
Pandas	Trace Verilerinin İşlenmesi
Scikit-learn	Makine Öğrenmesi Algoritmaları

Oracledb (cx_Oracle)	Oracle ile Bağlantı Kurulması
SQL Developer	Sorgu Analizi ve Plan İzleme

Arka planda Oracle veritabanı performansı izlenmiş ve analiz edilmiştir. Python programlama dili, veri işleme, analiz ve makine öğrenmesi modellerinin geliştirilmesi için kullanılmıştır. Performans analizleri Scikit-learn, pandas ve matplotlib gibi kütüphaneler aracılığıyla gerçekleştirilirken, veritabanı bağlantısı için oracledb kütüphanesi kullanılmıştır. Tablo 1, deneysel ortamda gerekli teknolojileri özetlemektedir.

3.1. Yöntem

Bu çalışmada, Oracle veritabanı üzerinde SQL sorgularının performans analizi gerçekleştirilmiştir. Analiz süreci aşağıdaki adımlardan oluşmaktadır:

Veri Toplama

SQL sorgularının çalıştırılması sırasında, ALTER SESSION SET SQL_TRACE = TRUE komutu kullanılarak izleme etkinleştirilmiş ve oluşturulan iz dosyaları kaydedilmiştir. Ayrıca, sorguların yürütme planları EXPLAIN PLAN komutu ile elde edilmiştir. Bu verilerden aşağıdaki performans metrikleri çıkarılmıştır:

Elapsed Time

Consistent Reads (CR)

Physical Reads (PR)

Estimated Query Cost

Full Table Scan

Join Count

İz Dosyalarının Analizi

Oluşturulan .trc uzantılı iz dosyaları Python betikleri kullanılarak işlenmiş, performans metrikleri ayrıştırılmış ve bir veri kümesine dönüştürülmüştür. Bu süreçte Elapsed Time, CR, PR, Executions, Plan Cost, Join Count ve Full Table Scan Count gibi özellikler çıkarılmıştır.

Oracle İz Dosyalarının Oluşturulması ve Analizi

Oracle iz dosyalarının (.trc) oluşturulması ve TKPROF gibi araçlarla analiz edilmesi, sorgu yürütme süresi (elapsed time), yürütme sayısı ve tam tablo taramaları gibi kritik performans metriklerinin elde edilmesini sağlar (Oracle Corporation, 2023). Bu çalışmada benzer şekilde, Oracle PL/SQL ortamında üretilen iz dosyaları Python betikleri ile işlenmiş; Elapsed Time, Executions ve Full Table Scan Count gibi metrikler çıkarılarak modelleme amacıyla bir veri kümesine dönüştürülmüştür.

Makine Öğrenmesi Modeli

Elde edilen veri kümesi, scikit-learn kütüphanesi aracılığıyla uygulanan Random Forest ve XGBoost algoritmaları kullanılarak eğitilmiştir. Bu modeller, sorgu performansını sınıflandırmakta ve "Yeterli", "Indeks Ekle", "Partition Önerisi" ve "JOIN Optimize Et" gibi optimizasyon önerileri üretmektedir.

Bu çalışmada kullanılan Random Forest tabanlı sınıflandırma yaklaşımı, (Dhanaraj ve ark., 2021) tarafından önerilen Random Forest Bagging X Means SQL Query Clustering yöntemine dayanmaktadır. Bu yöntem, SQL günlüklerinden anormal sorguları Random Forest ile belirlemekte, ardından X-Means kümeleme ve çoğunluk oylaması ile zayıf yapılandırılmış

ISSN
2822-6895

sorguları ayırt etmektedir. XGBoost, (Chen & Guestrin, 2016) tarafından geliştirilen, yüksek doğruluk sağlayan ağaç tabanlı bir topluluk makine öğrenmesi yöntemidir. Hızı ve aşırı öğrenmeye karşı dayanıklılığı, özellikle büyük ve yüksek boyutlu veri kümelerinde hem akademik hem de endüstriyel uygulamalarda tercih edilmesini sağlamaktadır. XGBoost algoritması, düzenlileştirme (regularization) ve paralel işlem desteği gibi geliştirmelerle Gradient Boosting yönteminin ileri bir versiyonudur. Bu iyileştirmeler, hem eğitim verimliliğini hem de modelin genellenebilirliğini artırmaktadır.

Bu çalışmada, sınıflandırma modelleri olarak XGBoost ve Random Forest seçilmiştir. Bu tercih, literatürde bu iki algoritmanın güçlü karşılaştırmalı performansını gösteren çalışmalarla desteklenmektedir. Örneğin, (Shao ve ark., 2024) XGBoost'un Random Forest'a kıyasla daha yüksek doğruluk sağladığını rapor etmiştir. Ayrıca, sınıf dengesizliği sorununu ele almak kritik öneme sahiptir; (Imani ve ark., 2025) SMOTE + XGBoost kombinasyonunun yüksek oranda dengesiz veri kümelerinde en iyi performansı sağladığını vurgulamıştır. Bu nedenle, tez kapsamında önce SMOTE uygulanmış, ardından her iki algoritma aynı veri kümesi üzerinde eğitilmiş ve karşılaştırılmıştır.

(Chen & Guestrin, 2016) tarafından sunulan XGBoost: A Scalable Tree Boosting System çalışması, XGBoost'un temel algoritmasını ve Gradient Boosting Machine (GBM) üzerindeki optimizasyonları detaylandırmaktadır. Seyrek veri işleme, paralel işlem yetenekleri ve düzenlileştirme mekanizmaları gibi performans artırıcı teknikler vurgulanmaktadır.

Tahmin ve Öneri

Yeni sorgular için aynı performans metrikleri çıkarılmış ve model tarafından analiz edilerek performans iyileştirme önerileri sunulmuştur. Bu öneriler hem istatistiksel analiz hem de yürütme planları ile desteklenmiştir.

Model Karşılaştırması

2021 yılında gerçekleştirilen bir tez çalışmasında, ağ tabanlı saldırı tespit problemleri bağlamında XGBoost ve Random Forest algoritmaları karşılaştırılmıştır. Bulgular, XGBoost'un daha yüksek doğruluk ve F1 skoru sağladığını, ancak eğitim sırasında biraz daha fazla hesaplama kaynağı gerektirdiğini göstermiştir (Buldu & Yıldız, 2021). Bu çalışmada, Random Forest ve XGBoost modelleri doğruluk, F1 skoru ve hata oranları açısından karşılaştırılmış ve en iyi performans gösteren model belirlenmiştir.

3.1.1. Veri kaynağı

Bu çalışmada kullanılan veri seti, Oracle veritabanı üzerinde gerçekleştirilen gerçek sorgu çalışmaları ve simülasyon verilerinden oluşmaktadır. Performans analizi için SQL sorguları, Oracle 18c Express Edition üzerinde oluşturulan test ortamında çalıştırılmıştır. İzleme sürecinde ALTER SESSION SET SQL_TRACE = TRUE komutu ile sorguların yürütülme detaylarını içeren trace dosyaları elde edilmiştir. Ayrıca, sorguların yürütme planları EXPLAIN PLAN komutu ile kaydedilmiştir. Bu kaynaklardan elde edilen veriler arasında sorgu çalışma süresi (Elapsed Time), mantıksal okuma (Consistent Reads), fiziksel okuma (Physical Reads), maliyet (Cost), tam tablo taraması (Full Table Scan) ve JOIN sayısı gibi performans metrikleri bulunmaktadır. Toplanan bu veriler, Python betikleri ile işlenerek makine öğrenmesi modellerinin eğitimi için kullanılacak veri kümesine dönüştürülmüştür.

3.1.2 Veri toplama

Çalışmada kullanılan veriler dört ana tablodan oluşan ve sorgu performansını ölçmeye olanak sağlayacak kadar veri ile yapılmıştır. Tablo ve veri sayıları tablo 2' de verilmiştir.

Tablo 2. DB Tablo Veri Seti

Tablo İsimleri	Kayıt Sayısı	İndex
T1_CUSTOMERS	500.000	Yok
T2_ORDERS	500.000	Yok
T3_PRODUCTS	500.000	Yok
T4_ORDER_DETAILS	1.000.000	Yok

3.1.3 Makine Öğrenmesi ile Model Eğitimi

Bu çalışmada, Oracle veritabanında gerçekleştirilen SQL sorgularının performansını analiz etmek ve iyileştirme önerileri sunmak amacıyla gözetimli öğrenme yöntemi tercih edilmiştir. Özellikle düşük performans gösteren sorguların karakteristiklerinden yola çıkarak, hangi durumlarda optimizasyon gerektiğini belirlemek üzere Random Forest algoritması kullanılarak bir sınıflandırma modeli oluşturulmuştur.

Modelin eğitimi için kullanılan veri seti, çeşitli sorguların performans özelliklerini içermektedir. Bu özellikler arasında; sorgunun tam tablo taraması yapıp yapmadığı (0 veya 1), sorguda yer alan join sayısı, tahmini yürütme maliyeti (cost) ve indeks kullanım durumu gibi parametreler bulunmaktadır. Her bir veri noktası, bu dört özelliği içeren bir vektör şeklinde temsil edilmiştir. Şekil 1'de örnek veri seti gösterilmektedir.

```
X = [
    [1, 3, 1500, 0],  # full scan var, 3 join, yüksek cost, index yok
    [0, 2, 500, 1],   # index var, düşük cost
    [1, 1, 1300, 0],  # full scan, az join, yüksek cost
    [0, 0, 300, 1],   # index var, simple query
    [1, 2, 1600, 0],  # full scan, orta karmaşa
    [0, 3, 800, 1],   # index var, çok join ama iyi optimizasyon
]

```


Şekil 1. Model Eğitim Girdileri

Sorgulara ait sınıf etiketleri (y), alan uzmanlarının değerlendirmeleri doğrultusunda belirlenmiştir. Bu etiketler, sorgunun mevcut haliyle yeterli olup olmadığını veya ek optimizasyona ihtiyaç duyup duymadığını ifade etmektedir. Şekil 1'de bu sınıflandırma etiketleri görselleştirilmiştir.

```
y = [
    'index ekle',
    'yeterli',
    'index ekle',
    'yeterli',
    'index ekle',
    'yeterli'
    ]

```

Şekil 2. Model Eğitim Çıktıları

Modelin eğitimi sürecinde RandomForestClassifier sınıfı kullanılmıştır. Bu algoritma, çok sayıda karar ağacından oluşan bir topluluk (ensemble) yapısı ile çalışarak sınıflandırma işlemini gerçekleştirir. Bu yöntem, hem aşırı öğrenmeyi (overfitting) azaltmakta hem de modelin doğruluk oranını artırmaktadır. Şekil 2'da modelin eğitilmesi ve kalıcı olarak saklanması süreci yer almaktadır.

```
1  from sklearn.ensemble import RandomForestClassifier
2  import joblib
3  model = RandomForestClassifier(n_estimators=100, random_state=42)
4  model.fit(X, y)
5  joblib.dump(model, "model.pkl")
6
```

Şekil 3. Model Eğitim Kütüphanesi

Şekil 3'da görüldüğü üzere, n_estimators=100 parametresi ile 100 adet karar ağacı oluşturulmuştur. Ayrıca random_state=42 parametresi, modelin tekrar üretilebilirliğini sağlamak amacıyla kullanılmıştır. Eğitim tamamlandıktan sonra model, joblib kütüphanesi aracılığıyla. pkl uzantılı bir dosya olarak kaydedilmiştir. Bu sayede model, ilerleyen aşamalarda yeni sorgular üzerinde gerçek zamanlı tahminler yapmak üzere tekrar kullanılabilir hale gelmiştir.

Sonuç olarak, bu aşamada geliştirilen sınıflandırma modeli, geçmiş sorgu verilerine dayanarak yeni sorguların performansını analiz edebilmekte ve gerektiğinde "indeks ekleme" gibi optimizasyon önerileri sunabilmektedir. Bu yapı, performans iyileştirme sürecinde yapay zekâ destekli karar verme mekanizmasının temelini oluşturmaktadır.

3.2. İyileştirme Önerisi ve Uygulama

Modelin aldığı girdiler doğrultusunda, sorgunun Total Cost ve Consistent Reads değerlerinin oldukça yüksek olduğu, ayrıca üç tabloya da Full Table Scan yöntemiyle erişim sağlandığı tespit edilmiştir. Bu analiz sonucunda, model tarafından ilgili tabloya indeks eklenmesi yönünde bir performans iyileştirme önerisi sunulmuştur. Gerçekleştirilen müdahale, öneriye uygun olarak tabloya indeks eklenmesi şeklinde uygulanmıştır. Şekil 4'de görüldüğü gibi T4_ORDER_DETAILS tablosuna Index eklenmiştir.

```
37
38 •  CREATE INDEX IDX_UNIT_PRICE ON OPT.T4_ORDER_DETAILS(UNIT_PRICE);
```

Şekil 4. Index Oluşturulması

Model tarafından önerilen indeks eklendikten sonra, sorgu üzerinde yeniden Trace ve SQL Execution Plan alınmıştır. Yapılan analiz sonucunda, Total Cost ve Consistent Reads değerlerinde belirgin bir düşüş gözlemlenmiş; ayrıca sorgunun artık Full Table Scan gerçekleştirmediği tespit edilmiştir. Çıktılar aşağıdaki gibidir.

- Elapsed Time: 0,49 saniye
- Consistent Reads: 789
- Physical Reads: 13

- Full Table Scan: 0
- Total Cost: 78

3.2.1 Performans İyileştirme Oranı

Performans iyileştirme yüzdesi Şekil 5'de olduğu gibi formülle hesaplanmıştır:

İyileştirme (%) = ((Eski Süre- Yeni Süre) / Eski Süre) * 100

```
eski_sure = 2.78
yeni_sure = 0.49
                iyilesme_orani = ((eski_sure - yeni_sure) / eski_sure) * 100
print(f"%{iyilesme_orani:.2f} performans artışı")
```

Şekil 5. ***Performans İyileştirme oranı***

Sonuç: %82,37 performans artışı.

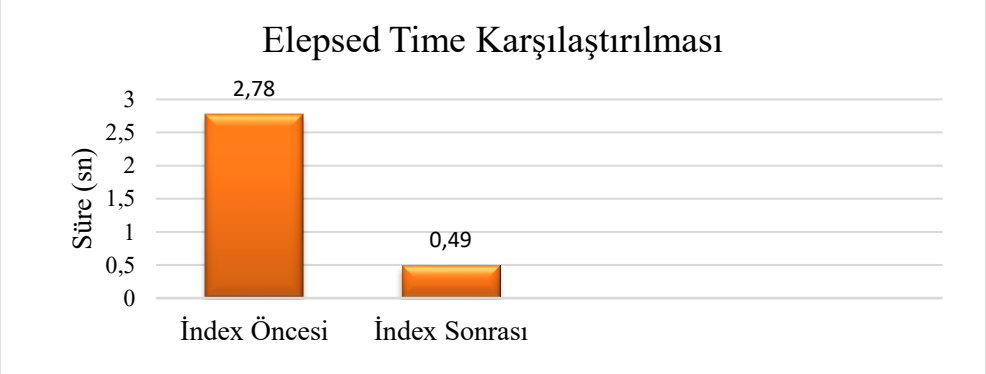
3.2.2 Elapsed Time Karşılaştırması

Index öncesi: 2,78 saniye, Index sonrası: 0,49 saniye.

Yaklaşık %82 azalma (iyileşme) sağlanmıştır.

Formül: İyileştirme Yüzdesi = ((2.78- 0.49) / 2.78) × 100 ≈ 82.37%

Şekil 6'da ise sorgu performansı artırılmadan önce ve sorası içim grafik gösterilmiştir.



Şekil 6. Elepsed Time Grafiği

3.2.3 Consistent Reads Karşılaştırması

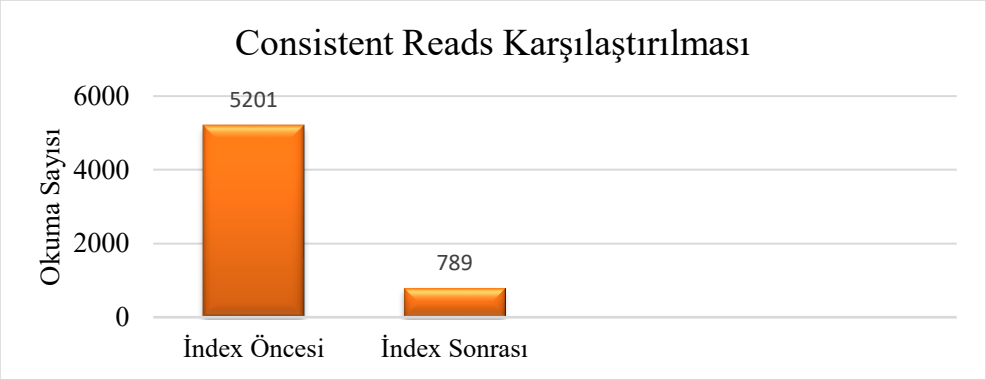
Index öncesi: 5201

Index sonrası: 789

Yaklaşık %84,8 azalma sağlanmıştır.

Şekil 7 'de sorgu performans artırımı yapıldıktan sonra Consistent Reads okuma grafiği gösterilmiş, Consisten Reads okumasında ciddi performans artırımı gözlemlenmiştir.

İyileştirme Yüzdesi = ((5201- 789) / 5201) × 100 ≈ 84.83%



Şekil 7. Consistent Reads Grafiği

3.2.4 Total Cost Karşılaştırması

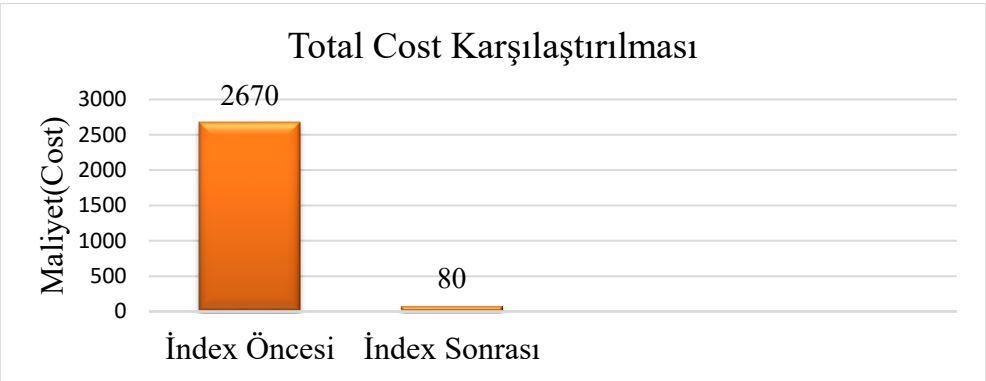
Index öncesi: 2670

Index sonrası: 78

Yaklaşık %97,1 maliyet düşüşü gözlemlenmiştir.

Total Cost maliyeti Python kodu şekil 37'de gösterilmiştir. Performans artırımı öncesi ve sorası şekil 8'de görüleceği gibi grafik oluşturulmuştur. Performans artırımı sonrası ciddi Total Cost azalması gözlemlenmiştir.

İyileştirme Yüzdesi = ((2670- 78) / 2670) × 100 ≈ 97.08%



Şekil 8. Total Cost Grafiği

3.3 Random Forest ve XGBoost Algoritmalarının Performanslarının Karşılaştırılması

Bu bölümde, Oracle SQL sorgularının performans verileri üzerinde kullanılan iki farklı makine öğrenmesi algoritması olan Random Forest ve XGBoost'un tahmin başarıları karşılaştırmalı olarak incelenmiştir. Temel amaç, her iki algoritmanın sorgu süresi tahmini ve performans göstergelerine dayalı optimizasyon önerileri üretme konusundaki etkinliğini değerlendirmektir.

Makine öğrenmesi tabanlı sınıflandırma modellerinin başarımı yalnızca genel doğruluk oranı (accuracy) ile sınırlı kalmamalı; aynı zamanda F1 skoru, duyarlılık (recall), hassasiyet (precision), eğitim süresi ve tahmin süresi gibi farklı ölçütler üzerinden de analiz edilmelidir. Bu çerçevede, çalışmada kullanılan Random Forest ve XGBoost algoritmaları çeşitli performans kriterleri doğrultusunda karşılaştırılmıştır. Aşağıda bu metriklerin tanımları ve hesaplama yöntemleri sunulmuştur.

3.3.1 Kullanılan Performans Ölçütleri ve Hesaplama Yöntemleri

Sınıflandırma algoritmalarının başarısını değerlendirmek için en yaygın kullanılan yöntemlerden biri karışıklık matrisi (confusion matrix)'dir. Bu matris, modelin tahmin ettiği sınıflarla gerçek sınıfların karşılaştırılması sonucunda elde edilen dört temel bileşeni içerir: Doğru Pozitif (DP), Yanlış Pozitif (YP), Doğru Negatif (DN) ve Yanlış Negatif (YN). Tablo 3'de bu bileşenler görsel olarak sunulmuştur.

Tablo 3. F1 Skor Karşılaştırma Matrisi

Matris Başlıkları		Gerçek Değer	
		Pozitif	Negatif
Tahmin Edilen	Pozitif	Doğru Pozitif (DP)	Yanlış Pozitif (YP)
	Negatif	Yanlış Negatif (YN)	Doğru Negatif (DN)

- Doğru Pozitif (DP): Modelin pozitif olarak tahmin ettiği ve gerçekten pozitif olan örneklerdir.
- Yanlış Pozitif (YP): Modelin pozitif olarak tahmin ettiği fakat gerçekte negatif olan örneklerdir.
- Doğru Negatif (DN): Modelin negatif olarak tahmin ettiği ve gerçekten negatif olan örneklerdir.
- Yanlış Negatif (YN): Modelin negatif olarak tahmin ettiği fakat gerçekte pozitif olan örneklerdir.

Bu dört değer yardımıyla çeşitli performans metrikleri hesaplanabilir.

- Doğruluk (Accuracy): Tüm doğru tahminlerin toplam tahmin sayısına oranıdır. Doğruluk = (DP +DN)/ (DP+DN+YP+ YN)
- Hassasiyet (Precision): Pozitif olarak tahmin edilenlerin ne kadarının gerçekten pozitif olduğunu gösterir.

Precision= DP / (DP+YP)

- Duyarlılık (Recall): Gerçek pozitif örneklerin ne kadarının doğru tahmin edildiğini gösterir.
  Recall= DP / (DP+YN)
- F1-Skoru (F1 Score): Precision ve Recall değerlerinin harmonik ortalamasıdır. F1-Skoru= 2*(Kesinlik * Duyarlılık) / (Kesinlik + Duyarlılık)

Bu metrikler sayesinde farklı modellerin sınıflandırma başarımı karşılaştırmalı olarak analiz edilebilir. Örneğin bu çalışma kapsamında kullanılan Random Forest ve XGBoost algoritmalarının değerlendirilmesinde bu matris ve türev metrikleri temel alınmıştır. Bu yöntem, (Yıldırım, 2020) ağ tabanlı saldırı tespiti üzerine gerçekleştirdikleri çalışmada da tercih edilmiştir. Onların çalışmasında, benzer şekilde karışıklık matrisinden türetilen metrikler kullanılarak algoritmaların doğrulukları karşılaştırılmıştır.

ISSN
2822-6895

4. DENEYSEL ÇALIŞMA

Bu bölümde, Oracle veritabanı sorgu performans verileri üzerinde geliştirilen makine öğrenmesi modellerinin eğitimi, değerlendirilmesi ve karşılaştırmalı analizleri sunulmaktadır. Çalışma kapsamında, literatürde veritabanı performans tahmini ve sınıflandırma problemlerinde yaygın olarak kullanılan iki farklı ağaç tabanlı topluluk öğrenme algoritması tercih edilmiştir:

- **Random Forest modeli:** Aşırı öğrenmenin (overfitting) önlenmesi ve genelleme kabiliyetinin artırılması amacıyla GridSearchCV yöntemi kullanılarak optimize edilmiştir. Bu kapsamda, n_estimators=100, max_depth=10, min_samples_split=5 ve min_samples_leaf=2 gibi hiperparametreler en uygun değerler olarak belirlenmiştir.
- **XGBoost Modeli:** Literatürde sıklıkla kullanılan varsayılan parametreler (n_estimators=100, learning_rate=0.3, reg_alpha=0, reg_lambda=1) ile eğitilmiş ve karşılaştırma açısından referans bir model olarak değerlendirilmiştir.

Her iki model de Python ortamında **scikit-learn** ve **xgboost** kütüphaneleri kullanılarak eğitilmiş olup, Oracle SQL_TRACE çıktılarından elde edilen ve *oracle_perf_data.csv* dosyasında yer alan performans metrikleri üzerinde test edilmiştir. Model performansları; doğruluk (accuracy), hassasiyet (precision), duyarlılık (recall), F1-skoru, eğitim süresi ve tahmin süresi gibi hem istatistiksel hem de operasyonel ölçütler üzerinden değerlendirilmiştir.

Tablo 4. Algoritma Performansı

Model	Doğruluk	Hassasiyet	Duyarlılık	F1-Skor	Eğitim Süresi (sn)	Tahmin Süresi(sn)
Random Forest	%94.3	%94.0	%95.0	%94.5	0.36	0.05
XGBoost	%96.1	%96.3	%96.0	%96.1	0.41	0.04

Tablo 4'te sunulan sonuçlar incelendiğinde, XGBoost algoritmasının %96.1 doğruluk ve %96.1 F1-skoru ile Random Forest modeline kıyasla daha yüksek sınıflandırma başarısı sağladığı görülmektedir. Bu durum, XGBoost'un gradyan artırma mekanizması ve düzenlileştirme (regularization) yetenekleri sayesinde karmaşık ve yüksek boyutlu performans verilerindeki örüntüleri daha etkin biçimde öğrenebilmesine bağlanabilir. Random Forest modeli ise %94.3 doğruluk ve %94.5 F1-skoru ile güçlü bir performans sergilemekle birlikte, özellikle karmaşık sorgu senaryolarında XGBoost'un gerisinde kalmıştır.

Eğitim süresi açısından değerlendirildiğinde, Random Forest modelinin daha kısa sürede eğitildiği, buna karşın XGBoost'un tahmin süresinde daha hızlı sonuç ürettiği gözlemlenmiştir. Bu durum, gerçek zamanlı veya çevrim içi performans izleme ve karar destek sistemlerinde XGBoost algoritmasının daha uygun bir alternatif olabileceğini göstermektedir. Öte yandan, Random Forest modelinin daha düşük eğitim maliyeti, sınırlı kaynaklara sahip ortamlarda tercih edilmesini mümkün kılmaktadır. Deneysel bulgular, her iki algoritmanın da Oracle tabanlı sorgu performans analizi için etkili çözümler sunduğunu, ancak yüksek doğruluk ve hızlı tahmin gereksinimlerinin ön planda olduğu senaryolarda XGBoost algoritmasının daha avantajlı olduğunu ortaya koymaktadır. Bu bulgular, çalışmanın ilerleyen aşamalarında geliştirilecek yapay zeka destekli karar destek mekanizmasının algoritma seçimi açısından önemli bir temel oluşturmaktadır.

ISSN
2822-6895

5. BULGULAR VE TARTIŞMA

Bu çalışmada geliştirilen entegre otomasyon sistemi, Oracle PL/SQL sorgularının performansını SQL_TRACE ve EXPLAIN PLAN çıktıları üzerinden analiz ederek; elde edilen verileri (örneğin tam tablo tarama sayısı, birleştirme (join) miktarı, yürütme maliyeti gibi) Random Forest algoritması ile değerlendirmektedir. Bu sistemin sunduğu çıktılar aşağıda özetlenmiştir:

- Random Forest algoritması, değişkenler arasındaki karmaşık ilişkileri öğrenme konusunda güçlü bir topluluk (ensemble) yöntemidir.
- Bu algoritmanın tercih edilme nedeni, aşırı öğrenmeyi (overfitting) azaltma yeteneği ve çoklu karar ağaçları üzerinden tutarlı tahminler sunabilmesidir. Bagging ve rastgele özellik seçimi sayesinde model varyansı düşürülerek daha dengeli sonuçlar elde edilmektedir (Breiman, 2001).
- Literatürde yer alan çalışmalar (örneğin Xuanhe Zhou, 2020 - Database Meets AI: A Survey) da SQL performans optimizasyonunda makine öğrenmesi tabanlı yaklaşımların giderek daha fazla kullanıldığını ortaya koymaktadır.

Modelin doğruluğunu ve genel başarımını artırmak amacıyla, Random Forest'a ek olarak XGBoost algoritması da uygulanmış ve iki model çeşitli metrikler üzerinden karşılaştırılmıştır. Bu karşılaştırmada doğruluk oranı (%96.1 vs. %94.3), F1 skoru (%96.1 vs. %94.5), hassasiyet, duyarlılık, eğitim süresi ve tahmin süresi gibi ölçütler dikkate alınmıştır. Elde edilen sonuçlara göre, XGBoost algoritması sınıflandırma başarımı açısından Random Forest'a kıyasla daha yüksek performans sergilemiştir. Özellikle doğruluk ve F1 skoru açısından anlamlı bir iyileşme gözlemlenmiştir. Ancak tahmin süresi bakımından Random Forest daha hızlı çalışmakta ve bu yönüyle bazı uygulamalarda avantaj sağlamaktadır.

Bu bulgular, Oracle veritabanı performans analizinde tek bir algoritmaya bağlı kalmak yerine, farklı makine öğrenmesi yöntemlerinin birlikte değerlendirilmesinin daha sağlıklı sonuçlar vereceğini göstermektedir. Random Forest, yorumlanabilirliği ve işlem hızıyla öne çıkarken; XGBoost daha karmaşık örüntüleri daha isabetli şekilde öğrenme kapasitesiyle dikkat çekmektedir.

Literatürdeki benzer çalışmalar da bu durumu desteklemektedir. Örneğin, Buldu ve arkadaşları (2020), ağ tabanlı saldırı tespitinde XGBoost'un Random Forest'a göre daha yüksek doğruluk ve daha düşük hata oranı sağladığını göstermiştir. Benzer şekilde, Chen ve Guestrin (2016) tarafından geliştirilen XGBoost algoritmasının, karar ağaçlarına dayalı yöntemler arasında en yüksek sınıflandırma doğruluğu ve işlem verimliliği sunduğu ifade edilmiştir.

Sonuç olarak, Oracle PL/SQL ve Forms uygulamalarında performans iyileştirme amacıyla geliştirilen bu sistemde, makine öğrenmesi modelleri hem teorik hem de deneysel olarak değerlendirilmiş; en uygun algoritmanın uygulama bağlamına göre seçilmesi gerektiği sonucuna ulaşılmıştır. Bu doğrultuda, gerçek zamanlılık, kaynak kullanımı ve doğruluk gibi kriterler göz önünde bulundurularak hibrit yaklaşımların geliştirilmesi önerilmektedir.

ISSN
2822-6895

6. SONUÇ VE DEĞERLENDİRME

Bu çalışmda, Oracle PL/SQL sorgularının performans analizine yönelik bir otomasyon sistemi geliştirilmiş ve bu sistem, geleneksel izleme araçları olan SQL_TRACE ve EXPLAIN PLAN çıktıları ile entegre edilerek sorgu performansına dair kritik verileri toplamıştır. Toplanan veriler, makine öğrenmesi algoritmaları kullanılarak değerlendirilmiş ve sorguların optimizasyon gereksinimleri hakkında öneriler sunulmuştur.

Çalışmanın temel amacı, veritabanı sorgularının performansını analiz edebilen, öğrenen ve öneride bulunabilen bir sistem tasarlamak ve bu sistemin doğruluğunu farklı algoritmalarla test ederek en uygun yaklaşımı belirlemektir. Bu doğrultuda, Random Forest ve XGBoost algoritmaları kullanılarak sınıflandırma modelleri oluşturulmuş ve sorguların performans özelliklerine göre optimizasyon ihtiyacı olup olmadığı tahmin edilmiştir.

Random Forest algoritması, özellikle değişkenler arasındaki karmaşık ilişkileri öğrenme kapasitesi ve aşırı öğrenmeyi azaltma yeteneği ile öne çıkmıştır. XGBoost ise daha yüksek doğruluk ve F1 skoru gibi metriklerde üstün performans sergileyerek, sınıflandırma başarımı açısından daha etkili bir alternatif olarak değerlendirilmiştir. Ancak tahmin süresi bakımından Random Forest daha hızlı sonuçlar üretmiş ve bu yönüyle gerçek zamanlı uygulamalarda avantaj sağlamıştır.

Her iki algoritmanın karşılaştırmalı analizi sonucunda, tek bir makine öğrenmesi yöntemine bağlı kalmak yerine, uygulama bağlamına göre farklı algoritmaların birlikte değerlendirilmesinin daha sağlıklı sonuçlar vereceği anlaşılmıştır. Özellikle büyük ve dinamik veri tabanlarında, sorgu desenlerinin zamanla değişmesi nedeniyle, modellerin periyodik olarak güncellenmesi ve yeniden eğitilmesi gerekliliği ortaya çıkmıştır.

Geliştirilen sistem, SQL sorgularının geçmiş performans izlerinden öğrenerek gelecekteki sorgulara yönelik optimize edici önerilerde bulunabilmekte; bu sayede insan müdahalesine gerek kalmadan performans iyileştirmesi sağlayabilmektedir. Bu yapı, özellikle kurumsal düzeyde büyük veri tabanı sistemlerinde, karar destek mekanizmalarının güçlendirilmesi açısından önemli bir katkı sunmaktadır.

Ayrıca, sorgu iyileştirmesi öncesi ve sonrası elde edilen metriklerin görselleştirilmesi, sistem yöneticilerinin karar alma süreçlerini desteklemekte ve öneri sisteminin doğruluğunu görsel olarak kanıtlamaktadır. Bu bağlamda, karar destek sistemlerine entegre edilebilecek görselleştirme modülleri, sistemin kullanılabilirliğini ve etkinliğini artıracaktır.

Sonuç olarak, bu çalışma, Oracle PL/SQL sorgularının performans analizine yönelik yapay zekâ destekli bir yaklaşım sunmakta hem teorik hem de uygulamalı olarak makine öğrenmesi algoritmalarının veritabanı optimizasyon süreçlerine nasıl entegre edilebileceğini göstermektedir. Geliştirilen sistem, veri tabanı yönetiminde otomasyonun artırılması, insan hatalarının azaltılması ve karar alma süreçlerinin hızlandırılması açısından önemli bir potansiyele sahiptir.

Gelecekte yapılacak çalışmalar için önerilen bu sistem, farklı veritabanı türlerine uyarlanabilir, daha fazla algoritma ile genişletilebilir ve gerçek zamanlı izleme sistemleriyle entegre edilerek daha kapsamlı bir performans yönetim platformuna dönüştürülebilir.

7. KAYNAKLAR

Abar, H. (2020). Prediction of gold prices using XGBoost and MARS methods. EKEV Academy Journal, 83, 427–446.

Asokan, E. (2025). The role of AI in predictive database performance tuning. International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 11(2), 356–369. https://doi.org/10.32628/CSEIT25112365

Buldu, B., & Yıldız, M. (2021). Performance comparison of XGBoost and random forest algorithms for network-based intrusion detection. International Marmara Sciences Congress, 733–739.

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. KDD 2016. https://doi.org/10.1145/2939672.2939785

Chicco, D., & Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. BMC Genomics, 21(6). https://doi.org/10.1186/s12864-019-6413-7

Dhanaraj, R. K., et al. (2021). Random Forest Bagging and X-Means Clustered Antipattern Detection from SQL Query Log for Accessing Secure Mobile Data. Wireless Communications and Mobile Computing, Article ID 2730246. https://doi.org/10.1155/2021/2730246

Doğanlı, B. (2025). Customer segmentation and behavior analysis: Examining income and spending behaviors using random forest. Business Economics and Management Research Journal, 8(1), 52–66. https://doi.org/10.58308/bemarej.1646966

Erdem, Ş., & Bakır, M. A. (2023). Detection of financial failure in the machinery and equipment manufacturing sector using isolation forest and resampling methods. Verimlilik Dergisi, 57(4), 719–734. https://doi.org/10.51551/verimlilik.1301865

Imani, M., Beikmohammadi, A., & Arabnia, H. R. (2025). Comprehensive analysis of Random Forest and XGBoost performance with SMOTE, ADASYN, and GNUS under varying imbalance levels. Technologies, 13(3), 88. https://doi.org/10.3390/technologies13030088

Karakaya, İ. (2025). Evaluation of Machine Learning and Ensemble Learning Models for Classification Using Delivery Data. Productivity for Logistics, 89–104. https://doi.org/10.51551/verimlilik.1526436

Marcus, R., & Narayanan, P. (2020). Bao: Learning to steer query optimizers. SIGMOD 2020. https://doi.org/10.1145/3448016.3452838

Oracle Corporation. (2023). Using SQL Trace and TKPROF (Oracle Database Reference Guide). https://docs.oracle.com/en/database/oracle/oracle-database/23/tgsql/performing-application-tracing.html

Shao, Z., Ahmad, M. N., & Javed, A. (2024). Comparison of Random Forest and XGBoost classifiers using integrated optical and SAR features for mapping urban impervious surface. Remote Sensing, 16(4), 665. https://doi.org/10.3390/rs16040665

Şahinaslan, E., & Şahinaslan, Ö. (2022). Microsoft SQL sunucusunda veritabanı kurtarma teknikleri. *International Journal of Innovative Engineering Applications*, *6*(1), 158–169. https://doi.org/10.46460/ijiea.1070325

Şahinaslan, E., Şahinaslan, Ö., & Dalyan, H. (2022). Naive Bayes sınıflandırıcısı kullanılarak YouTube verileri üzerinden çok dilli duygu analizi. *Avrupa Bilim ve Teknoloji Dergisi*, (38), 320–327. doi: 10.17671/gazibtd.999960

Şahinaslan, E., & Şahinaslan, Ö. (2023). Gümrük beyannamesi sürecinde öğrenmeye dayalı algoritmaların etkinliğinin incelenmesi. *Uluslararası Mühendislik Araştırma ve Geliştirme Dergisi*, *15*(2), 410–421. doi: 10.26650/acin.1057060

Tartuk, M., Nurdağ, T. F., Acar, V., Erdem, S., Akay, F., & Abut, F. (2022). Forecasting call center arrivals using XGBoost combined with consecutive and periodic lookback. Eastern Anatolian Journal of Science, 8(1), 20–25.

Veranyurt, Ü., Deveci, A., Esen, M. F., & Veranyurt, O. (2020). Disease classification using machine learning techniques: Application of Random Forest, K-Nearest Neighbour and AdaBoost algorithms. International Journal of Health Management and Strategies Research, 6(2), 275–286.

Yang, Z., & Chiang, W.-L. (2022). Balsa: Learning a query optimizer without expert demonstrations. SIGMOD 2022. https://doi.org/10.1145/3514221.3517885

Yangın, G. (2019). Application of XGBoost and decision tree-based algorithms on diabetes datasets [Master's thesis]. Mimar Sinan Güzel Sanatlar University.

Yeşilyurt, S. N., & Dalkılıç, H. Y. (2021). XGBoost ve Gradient Boost Machine ile günlük nehir akımı tahmini [Daily river flow prediction using XGBoost and Gradient Boost Machine]. In Proceedings of the 3rd International Symposium on Civil Engineering and Earth Sciences, 36–44.

Örgerim, A., Tunç Abubakar, T., & Tokmak, M. (2025). Performance comparison of neural networks: A case of data scientists' job change prediction. Osmaniye Korkut Ata University Journal of the Institute of Science and Technology, 8(3), 1100–1119. https://doi.org/10.47495/okufbed.1481893