metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**MARMARA ÜNİVERSİTESİ**

**TEKNOLOJİ FAKÜLTESİ**

**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

**BİTİRME PROJESİ**

**Veri Tabanı Temelli Derin Öğrenme ile Aykırılık Tespiti**

**Melih BALTA**

**Bengisu ÇALIŞKAN**

**DANIŞMAN**

Doç. Dr. Buket DOĞAN

**İSTANBUL, 2024**

**İÇİNDEKİLER**

**ÖZET……………………………………………………………………..iii**

**ABSTRACT………………………………………………………………iv**

1. **GİRİŞ………………………………………………………………………1**
   1. **İlgili çalışmalar…………………………………………………….2**
   2. **Benchmark Çalışmalar……………………………………………9**
   3. **Amaç………………………………………………………………10**
   4. **Uygulama Kapsamı ve Kullanıcı İhtiyaçlarının Belirlenmesi…10**
2. **MATERYAL VE YÖNTEM…………………………………………….11**

**2.1 Veri Toplama İşlemi…………………………………………………12**

**2.1.1 Hedef Verilerin Toplanması……………………………………18**

**2.2 Sistem Mimarisinin Tanımlanması………………………………….27**

**2.2.1 Apriori Algoritması ve Birliktelik Kuralları………………….28**

**2.2.2 Sınıflandırma ve Karar Ağaçları………………………………29**

**2.2.3 Korelasyon Matrisi……………………………………………..30**

**2.2.4 K-Means Algoritması…………………………………………..31**

**2.2.5 K-En Yakın Komşu Algoritması………………………………32**

**2.2.6 Random Forest (Rastsal Orman)……………………………...32**

**2.2.7 Destek Vektör Makineleri (SVM)……………………………...33**

**2.3 Veri Tanıtımı…………………………………………………………34**

**2.4 Veri Ön İşleme………………………………………………………..36**

**2.4.1 Veriye Genel Bakış………………………………………………36**

**2.4.2 Apriori Algoritması……………………………………………...40**

**2.4.3 Korelasyon Matrisi………………………………………………43**

**2.4.4 İş Parçacıklarının ‘page life expectancy’ değerine etkisi……...44**

**2.4.5 Fp Growth Algoritması………………………………………….44**

**2.5 Makine Öğrenmesi Modelleri…………………………………………46**

**2.5.1 K-Means Kümeleme Algoritması………………………………..46**

**2.5.2 K-En Yakın Komşu……………………………………………….51**

**2.5.3 Random Forest(Rassal Orman)………………………………….53**

**2.5.4 Destek Vektör Makineleri……………………………………….56**

**2.6 Sonuçların Analiz Edilmesi…………………………………………..58**

**3. BULGULAR VE TARTIŞMA……………………………………………59**

**Kaynakça……………………………………………………………………..60**

**ÖZET**

Bu tez, veri tabanı sistemlerindeki potansiyel sorunları erkenden tespit etmek ve bu sorunlara otomatik müdahale yöntemleri geliştirmek amacıyla, makine öğrenimi temelli aykırılık (anomali) tespiti yöntemlerini kullanmaktadır. Çalışma, sürekli toplanan telemetri verilerinin detaylı bir şekilde incelenmesi, ön işleme tabi tutulması ve analiz edilmesi süreçlerini kapsamaktadır. Yerli bir yazılım çözümü geliştirilmesi hedeflenmiş olup, veri tabanlarında yaşanabilecek problemlerin erken tespiti için otomatik bir sistem tasarımı amaçlanmıştır. Tezde veri toplama, ön işleme, analiz etme ve model değerlendirme süreçleri ayrıntılı bir şekilde ele alınmış ve çeşitli makine öğrenmesi modelleri kullanılarak performans değerlendirmesi yapılmıştır. Sonuç olarak, geliştirme sürecindeki deneyimler ve elde edilen bulgular, veri tabanı yönetim süreçlerindeki performansı artırma ve gelecekteki otomatikleştirme çabalarına ışık tutmaktadır. Bu sayede veri tabanlarını daha optimize hale getirip, sorgu maliyetlerini en aza indirmek proje kapsamının genel amaçlarındandır. Gelişmiş versiyonlarında ise sunucu kontrollerinden, sunucu içerisindeki gerekli tüm aksiyonları alabilir duruma gelmesi beklenmektedir.

Anahtar Kelimeler: Veri tabanı, Telemetri Verileri, Aykırılık Tespiti, Veri Ön İşleme, Model Değerlendirme, Performans Değerlendirmesi, Makine Öğrenmesi, Derin Öğrenme, Yapay Zeka, Sunucu Verileri

Github link: https://github.com/baltamelih/vtys-deep-learning-and-anomaly-detection

**ABSTRACT**

This thesis utilizes machine learning-based anomaly detection methods to proactively identify potential issues in database systems and develop automatic intervention techniques. The study encompasses the detailed examination, preprocessing, and analysis of continuously collected telemetry data. The goal was to develop a domestic software solution aimed at the early detection of possible problems within databases, aspiring towards the design of an automatic system. Throughout the thesis, data collection, preprocessing, analysis, and model evaluation processes have been thoroughly examined, and various machine learning models have been employed for performance evaluation. Ultimately, the experiences and findings from the development process shed light on enhancing the performance of database management processes and inform future efforts in automation.

Keywords: Database, Telemetry Data, Anomaly Detection, Data Preprocessing, Model Evaluation, Performance Assessment, Machine Learning, Deep Learning, Artificial Intelligence, Server Data

# GİRİŞ

Günümüzde, teknolojinin hızla gelişmesiyle birlikte, zeki yazılımların tasarımı ve geliştirilmesi, kullanıcı deneyimini artırmak ve verimliliği optimize etmek adına önemli bir yer edinmiştir. Zeki olmayan yazılımlar, kullanıcılarını tekrarlayan işlerle meşgul ederek verim kaybına yol açabilmekte ve bu durum kullanıcıları, daha karmaşık ve akıllı yazılımları tercih etmeye yönlendirmektedir.

Zeka kavramını, D. Lenat ve E. Feigenbaum'un tanımlarından yola çıkarak "karmaşık bir problemi çözmek için gerekli bilgileri toplama ve birleştirme kabiliyeti" olarak düşünebiliriz. Bu, aynı zamanda insan zekasının temel özelliklerini de yansıtmaktadır: düşünme, akıl yürütme, nesnel gerçekleri algılama, kavrama, yargılama, sonuç çıkarma gibi yetenekler. Yapay zeka, bu özelliklere sahip olmayan organik olmayan sistemlerdeki zekayı temsil eder ve Luger ve Stubblefield (1993) tarafından "zeki davranışların otomasyonu ile ilgili bilgisayar bilimleri dalı" olarak tanımlanmıştır.

Yapay zeka uygulamalarının temelini oluşturan makine öğrenimi, yazılımların tecrübelerine dayanarak davranışlarını değiştirebilme yeteneği sunmaktadır. Bu bağlamda, yapay sinir ağları, makinelerin öğrenmesini sağlayan en güçlü yöntemlerden biridir.

Veri tabanları günümüzde neredeyse her firmanın verilerini depolamak ve korumak amacıyla kullandığı bir yapıdır. Bu sebeple veri tabanlarının optimizasyonu büyük önem arz etmektedir. Sürekli olarak veri tabanı yöneticileri bu sistemleri kontrol etse de sistemlerde yaşanan sorunlar firmalar için büyük aksaklıklar oluşturur. Bu aksaklıkları önlemek veya çözmek adına geçirilen süre ise firma için bir kayıptır. Bu sebeple firmaların veri tabanlarının kontrolünün otomatikleştirilmiş bir kontrol yapısına sahip olması zaruri bir durum haline gelmiştir. Tez kapsamında kullanılan yöntemler sayesinde sunuculardan toplanan telemetri datalarının analizi ve işlenmesi ihtiyaç haline gelmiştir.

Bu çalışma, yerli yazılım geliştirme sürecinde ortaya çıkabilecek potansiyel sorunları tespit etmek ve çözmek amacıyla makine öğrenimi ile anomali (aykırılık) tespiti yöntemini ele almaktadır. Veri tabanlarında oluşabilecek sorunların tespiti sayesinde, sürekli eğitilen kural cümleleri ile çözüm önerileri sunarak ileriye dönük veri kaybını, performans sorunlarını ve ek maliyetleri önlemeyi hedeflemektedir. Bu bağlamda, çalışmanın ayrıntılı bir literatür incelemesi ve güncel gelişmelerle desteklenen bir metodoloji sunması, konuya dair daha derin bir anlayış sağlamayı amaçlamaktadır.

Bu tez, geliştirilecek olan yerli yazılımın, veri tabanlarındaki olası sorunları tespit etme ve otomatik olarak çözme yeteneklerini değerlendirmeyi hedeflemektedir. Ayrıca, elde edilen sonuçların, gelecekteki yazılım geliştirme süreçlerinde daha etkili ve sorunsuz uygulamalara olanak sağlayacak bir temel oluşturmasını beklemekteyiz.

## İlgili Çalışmalar

Yapay zeka , Makine Öğrenmesi ve Derin öğrenme teknolojilerinin gelişmesiyle birlikte sağlık, askeri, metin işleme, görüntü işleme, veri işleme, robotik sistemlerde, tarım ve gıda alanında, ekonomi, satış-pazarlama stratejileri, elektronik yapılar ve sosyal medya gibi birçok alanda hayati önemi olan sistemlerde analizler, model eğitimleri ile iş geliştirme süreçleri tasarlanabilmektedir. Yapılan analizle birlikte elde edilmiş olan sonuçlara baktığımızda hedeflenen performansın arttırıldığı, maliyetlerin düşürüldüğü, verinin anlamlaştırıldığı, verimin hedeflenenden fazla yükseldiği , insan gücünün de azaldığı görülmüştür. Bu sebeplerden dolayı Veri tabanı yönetim sistemlerinde de yapay zeka, makine öğrenmesi, derin öğrenme gibi teknolojiler ile birlikte sistemlerin hızının arttırılacağını, performans iyileştirmelerinin yapılacağını, saklanan verilerin daha anlamlı hale gelebileceği düşünülmektedir. Ulusal düzeyde yerli ve milli bir yapının bulunmadığını saptadık. Veri tabanı yönetim sistemlerini yapay zeka ile besleyen bir yapının olmamasıyla birlikte hali hazırda pek çok veri tabanı yöneticisinin sistemleri manuel bir şekilde kontrol etmektedir. Bu durum da sistemler için ek bir maliyet ve en önemlisi sistemin çalışmasını negatif yönde etkilemektedir.

Uluslararası literatürde otomatik veri tabanı yönetim sistemleri çalışmaları bulunmaktadır. Arama tabanlı metotlar ve öğrenme tabanlı metotlarla geliştirilmiş olan çeşitli yazılımlar ile birlikte konfigürasyon ayarlarının optimal değerlerinin tespiti yapılabilmektedir. Ancak bu çalışmalarda bizim çalışmamızda kullandığımız bir büyüklükte veya firmaların gerçek verileri kullanılmamış olup sentetik veriler ile bir çalışma yürütülmüştür.

Tablo 1.1 Literatür tablosu

|  |  |  |  |
| --- | --- | --- | --- |
| Yayın Adı | Tarihi | Yazar(lar) | Özet bilgi |
| **Automatic Database Management System Tuning Through Large-scale Machine Learning**  [7] | Mayıs,2017 | Dana Van Aken, Andrew Pavlo, Geoffery J. Gordan, Bohan Zhang | Veri tabanı yönetim sistemi konfigürasyonunun ayarlanması, veri yoğunluğuyla ilgili uygulama çabalarının önemli bir yönüdür. Veri tabanı yönetim sistemleri, önbellekler için kullanılacak bellek miktarı ve verilerin depolama alanına ne sıklıkta yazılacağı gibi işlemleri kontrol eden kontrol düğmeleri vardır. Bu kontrol düğmeleri ile ilgili standartlaştırılmaması, bağımsız olmaması, evrensel olmaması daha da kötüsü düğmelerin etkileri hakkındaki bilgiler genellikle yalnızca pahalı deneyimlerden meydana gelir. Bu durumlardan kaynaklı zorlukların üstesinden gelmek adına geçmiş deneyimlerden yararlanan veri tabanı yönetim sistemi yapılandırmalarını ayarlamak için yeni bilgiler toplayan otomatik bir yaklaşım sunmakta. En etkili kontrol araçları, görünmeyen veri tabanlarını haritalama işlemi için denetimli veya denetimsiz makine öğrenimi yöntemlerinin bir kombinasyonunu kullanmaktadır. |
| Security in database management system using machine learning [8] | Ocak, 2024 | M. Deepa,  J. Dhilipan | ‘Veri tabanı güvenliği’ terimi, veri tabanlarının gizliliğini, bütünlüğünü ve erişilebilirliğini sürdürmek ve korumak için geliştirilen kuralların, araçların ve süreçlerin bütününü ifade eder. Veri tabanı yönetimi güvenliğini geliştirmek için makine öğreniminin kullanımı daha yaygın hale geliyor. Makine öğrenimini güvenlikte kullanmanın temel amacı, kötü amaçlı yazılım tespit sürecini insanların katılımını gerektiren geleneksel tekniklere göre daha uygulanabilir, ölçeklenebilir ve başarılı kılmaktadır. Bu süreci yönetilmesi gereken sorunların aşılması da gerekir. |
| **Agricultural Databases Evaluation with Machine Learning Procedure [3]** | Ocak, 2023 | Ali Rahmani,  Mahyar Amini | Bu çalışmada Makine öğrenimi tekniklerini tarımsal veri tabanlarına uygulanmasına ilişkin deneyimleri gözden geçirilmiştir. Bu çalışmada makine öğrenimi şemalarını kullanarak belirli bir veri kümesi üzerinde hızlı deneyler yapılmasına olanak tanıyan ve verilerin etkileşimli olarak incelenmesi için çeşitli olanaklara sahip olan bir makine öğrenimi tasarlanmıştır. Ön işleme özellikleri farklı sonuçların değerlendirilmesi ve karşılaştırılması, şemalar ve çevrimdışı yürütülecek karşılaştırılmalı deneylerin tasarlanmasını kapsamaktadır. |
| **A novel deep learning method for query task execution time prediction in graph database [1]** | Ağustos, 2019 | Zheng Chu, Jiong Yu, Askar Hamdulla | Bu çalışmada incelenmiş olup, bu çalışma graf veri tabanlarında sorgu görevlerinin yürütme süresini doğru ve etkin bir şekilde tahmin etmeyi amaçlamaktadır. Gelişen büyük veri, internet nesneleri ve bilgi grafiği teknolojileri ile geleneksel ilişkisel veri tabanlarının performansında da yaşanan zorluklar yüzünden, graf veri tabanlarını gerçek dünya varlıkları ve ilişkilerini eşleştirmek daha uygun görülmektedir. Yöntem olarak ise sorgu planı ağaçlarını işlem dizileri olarak kodlamak için post-sıralı dolaşım önerilmektedir. Seçilen en iyi 20 özellik, 68 aday sistem özelliği ile Random Forest (Rastsal Orman) algoritması kullanılarak bunların arasından seçilmektedir. Zaman bağımlılıkları sebebiyle LSTM yapısı da kullanılmaktadır. Sonuç olarak modelin ortalama doğruluğu (accuracy) %81,34’e ulaştığı ve yüksek bir tahmin verimliliği oranına çıktığı görülmektedir. Bu sonuçlar ise, derin öğrenme yönteminin graf veri tabanlarında sorgu görevlerinin yürütme süresinin tahmin edilmesi noktasında uygulanabilirliğini kanıtlar niteliktedir. |
| **Optimizing Data Warehousing Performance through Machine Learning Algorithms in the Cloud [2]** | Kasım, 2023 | Sina Ahmadi | Bu çalışma kapsamında veri ambarı performansını bulut ortamında makine öğrenimi kullanarak optimize etme üzerine temel teorik bilgileri aktarmaktadır. Veri ambarlarının merkezi bir raporlama ve analiz merkezi olarak tanınmasına dayanan bu yaklaşım mevcut verimsizlikleri de ele alır. Makine öğreniminin uygulanması algoritmaların yalnızca araçlar olmaktan çıkıp, gerçek zamanlı karmaşıklık teoreminin de bir parçası haline gelmektedir. Makine öğreniminin gecikme sürelerini azaltma, sorguları optimize etme, değişken talebi yönetimi rollerini vurgular. Sonuç olarak ise veri ambarı konusunda birçok teorik bilgi vermesiyle birlikte çağdaş veri ambarı sistemlerinin genel verimliliğini ve performans optimizasyonundan da bahsetmektedir. Yöntem olarak veri ambarı performans optimizasyonu için ML (makine öğrenmesi) tercih edilmiştir. Hem denetimli hem de denetimsiz öğrenme performans iyileştirmeleri için otomatik karar verme yeteneği kazandırılmaktadır. |

## Benchmark Çalışmalar

Patent Public Search Basic [PPUBS Basic] sitesinde ‘database and deep learning’, ‘database optimization’, ‘database and machine learning’, ‘database artificial intelligience’ kelimeleri ile patent araştırması gerçekleştirilmiştir. Herhangi bir sonuca ulaşılamamıştır.

Espacenet sitesinde sitesinde ‘database and deep learning’, ‘database optimization’, ‘database and machine learning’, ‘database artificial intelligience’ kelimeleri ile patent araştırması gerçekleştirilmiştir. Herhangi bir sonuca ulaşılamamıştır.

Türk Patent ve Marka Kurumu sitesinde, ‘veri tabanı ve yapay zeka’, ‘veri tabanı’,’veri tabanı ve derin öğrenme’, ‘veri tabanı ve makine öğrenmesi’, ‘database and deep learning’, ‘database optimization’, ‘database and machine learning’, ‘database artificial intelligience’ kelimeleri ile patent araştırması gerçekleştirilmiştir. Herhangi bir sonuca ulaşılamamıştır.

## Amaç

Geliştirilecek olan yerli yazılım ile veri tabanlarında ileride oluşacak sorunları tespit edilmek için makine öğrenmesi ile anomali (aykırılık) tespiti yapılabilecektir. Makine öğrenmesi kullanarak tespit edilen sorunların otomatik olarak çözülebilmesi için sürekli eğitilen kural cümleleri ile çözüm önerileri sunulacaktır. Böylelikle ileriye dönük yaşanacak olan herhangi bir veri kaybı, performans sorunları veya ek maliyet getirecek tüm sorunların önlemi otomatik olarak alınmış olacaktır.

Proje çıktısı ile amaçladıklarımız;

Veri tabanlarını makine öğrenmesi ile düşük maliyetlerle izleyebilmek,

Veri tabanları özelinde toplanan verilerin tanıtılması,

Sunuculardan toplanan ham telemetri verilerinin hedef telemetri verilerine dönüştürülmesi,

İleride karşılaşılabilecek veri tabanı sorunlarının önceden tespit ederek müdahale edebilmek,

Toplanan telemetri verileri ile veri analizinin yapılabilmesi,

Veri analizinden çıkan sonuçların değerlendirilmesi,

Veri analizi ile birlikte sunucu performanslarının kontrolü,

Veri tabanlarının 7/24 performans takibinin yapılabilmesi,

Makine öğrenmesi ile yarı otonom veya otonom yönetimin mümkün kılınması,

Eksik veya kötü indekslerin otomatik olarak tespit edilmesi,

Veri yedekleme süreçlerindeki aksaklıklara karşı bildirimde bulunmak.

## Uygulama Kapsamının ve Kullanıcı İhtiyaçlarının Belirlenmesi

Geliştirilecek olan yerli veri tabanı yönetim platformu ile veri tabanlarında ileride oluşacak sorunları makine öğrenmesi ile anomali tespiti yapılabilecektir. Sistemde sunucuların performans metrik değerlerinden oluşan bu büyük veri seti farklı sektör ve veri tabanları için alt kümelere bölünerek, her bir veri satırı için Normal /Anormal gibi farklı durumları içeren etiketleme süreci veri tabanı uzman görüşlerini yansıtacak biçimde bir veri etiketleme sürecinden geçirilecektir.

Örneğin, orta düzey büyüklüğe sahip bir veri tabanında bir sorgu için 1 milisaniyelik çalışma zamanı (execution time) süresi "normal" bir süre olarak kabul edilebilirken, büyük ve karmaşık bir veri tabandaki bir sorgu için 1 milisaniyelik çalışma zamanı "kısa" olarak kabul edilebilmektedir. Benzer şekilde, büyük ve karmaşık bir veri tabanı için 10 milisaniyelik bir sorgu çalışma zamanı "normal" durum olarak kabul edilebilirken, küçük bir veri tabanına ait bir sorguda aynı çalışma zamanı "anormal" bir durum olarak kabul edilmektedir.

Veri etiketleme sürecinde Normal/Anormal durum etiketinin; veri tabanı yapısı, performans metrikleri ve alan deneyimine bağlı olarak alan uzmanları tarafından kural şeklinde tanımlanması ile verinin etiketlenmesini sağlanacaktır. Bu süreçte oluşturulan alt veri setlerinde indeksler, veri tabanı büyüklük metrikleri gibi farklı metriklere bağlı olarak normal ve anormal durumları belirlemek için alan uzmanı grupları ile gerçekleştirilecek odak toplantılar ile bir grup alan uzmanın görüşü doğrultusunda, kuralların oluşturularak etki gücü yüksek bir etiketleme çalışmasının yapılması planlanmaktadır.

Mevcutta Data Platform şirketinden elde edilen 45 firmaya ait 10 milyon satır ve 8 sütundan oluşan, 100 GB veri bulunmaktadır. Şirketin müşterilerinden elde ettiği veri serileri kullanılacaktır. Veri setlerimiz KVKK (Kişisel Verilerin Korunması Kanunu) kapsamına uygun olarak elde edilmiştir.

Makine öğrenim sürecinin aşağıdaki gibi yapılması planlanmaktadır.

-Verileri anlama ve toplanması

-Verilerin işlenmesi

-Özellik veya modelin seçilmesi

-Verilerin görselleştirilmesi

-Modelin izlenmesi.

Bu kapsamda, araştırılan anomali (aykırılık) tespiti ve makine öğrenmesinde kullanılacak olan kütüphaneler denenecek ve kullanılacak olan algoritmalar arasından en başarılı olanlar geliştirme çalışmalarında kullanılmak üzere seçilecektir. Kullanılan bu farklı yöntemler ile SQL sorgu planlarının karşılaştırmasını yaparak maliyeti en düşük olan sorgu planının seçilmesine olanak sağlanacaktır. Bu sayede yapay zekanın (artificial intelligence) ve makine öğrenmesi metodolojileri birleştirilerek melez bir yapıyla veri tabanı yönetim sistemi ortaya çıkartılacaktır.

Makine öğrenme algoritmaları ve literatürde en fazla kullanılan modellerin birleştirilmesi yöntemleriyle, takip edilerek güncellenen algoritmalarla sürekli yaşayan ve öğrenen bir uygulama olması hedefine ulaşılması beklenmektedir.

# MATERYAL VE YÖNTEM

1. Veri Toplama ve Ön İşleme
   * Hedef verilerin toplanması
   * Verilerin temizlenmesi ve ön işleme tabi tutulması
   * Özellik mühendisliği ve veri setinin hazırlanması
2. Model Geliştirme
   * Makine öğrenimi algoritmalarının seçimi
   * Algoritmalara özgü veri düzenlemeleri
   * Model eğitimi ve parametre ayarlaması
3. Sonuçlar ve Raporlama
   * Modelin nihai sonuçlarının analizi
   * Elde edilen bulguların raporlanması
   * Proje sonuçlarının paylaşılması ve gelecek adımlar

## Veri Toplama İşlemi

Geliştirilecek olan yerli veri tabanı yönetim platformu ile veri tabanlarında ileride oluşabilecek sorunları makine öğrenmesi ile aykırılık tespiti yapılması amaçlanmaktadır. Sistemde sunucuların performans metriklerinden oluşan bu büyük veri seti farklı sektöre ve veri tabanı çeşitleri için alt kümelere bölünerek, her bir veri satırı için ‘Normal’ ve ‘Anormal’ olmak kaydıyla etiketleme süreci veri tabanı uzman görüşlerini de yansıtacak biçimde yapılacaktır. Örnek olarak orta düzey büyüklüğe sahip bir veri tabanında bir sorgu için 1 milisaniyelik çalışma zamanı (execution time) süresi normal bir süre olarak kabul edilebilirken, büyük ve karmaşık bir veri tabanındaki bir sorgu için 1 milisaniyelik çalışma zamanı kısa kabul edilebilmektedir. Benzer şekilde, büyük ve karmaşık bir veri tabanı için 10 milisaniyelik bir sorgu çalışma zamanı normal bir durum olarak kabul edilebilirken, küçük bir veri tabanına ait bir sorguda aynı çalışma zamanı anormal bir durum olarak kabul edilmektedir.   
Veri etiketleme süreçlerinde ‘Normal’ ve ‘Anormal’ olarak durum etiketi alan verilerin, veri tabanı yapısına, performans metrikleri ve alan deneyimine bağlı olarak alan uzmanları tarafından kural şeklinde tanımlanması ile sağlanmıştır. Bu süreçte oluşturulan alt veri setlerinde indeksler, veri tabanı büyüklük metrikleri gibi farklı metriklere bağlı olarak normal ve anormal durumları belirlemek için alan uzmanı grupları ile gerçekleştirilmiş odak toplantılar ile bir grup alan uzmanın görüşü doğrultusunda, kuralların oluşturularak etki gücü yüksek bir etiketleme çalışması yapılmıştır. Oluşturulacak modellerin başarısı, eğitim veri kalitesine bağlı olduğu için bu veri etiketleme aşamasının önce ufak bir veri kesitinde yapılarak, veri tabanına/sektöre özel kural kontrolleri gerçekleştirildikten sonra tüm veri setinde kullanılması da planlar dâhilindedir.

ETL(Extract-Transport-Load) süreçlerinde veri aşağıdaki gibi temin edilecektir.

dbaHCDOC job’ı DOCjobList tablosunu kontrol eder ve periyodu gelen (zamanı gelen) saklı yordamı çalıştırır. Çalışacak olan job’lar ise şunlardır:

1. [dbo].[spDOCbackupHistory]
2. [dbo].[spDOCdiskStatus]
3. [dbo].[spDOCple]
4. [dbo].[spDOCspConfig]
5. [dbo].[spDOCinstanceRecycle]
6. [dbo].[spDOChaStatus]
7. [dbo].[spDOCdbInfo]
8. [dbo].[spDOCdatabaseSize]

DOCintegration ile birlikte veri toplayan saklı yordamlar, toplanmış olan verileri DOCintegration tablosuna yazmaktadır. dbaHCDOCexportTxt Job’ı ile birlikte her saat başı olacak şekilde bir saatlik periyotlar ile çalışacak ve DOCintegration tablosundaki sendTo değeri 0 olan verileri maksimum 20.000 satır olmak üzere önerilen C diskinde dpIntegration klasörü içerisinde bir txt dosyasında depolanmaktadır. Bu dosyalardan en yeni olanını ‘bidata’ olarak belirlediğimiz mail adŞekilize gönderilmektedir. Son adım olarak ise bu klasör içerisinde depolanan txt dosyalarını 3 gün tutulmasının ardından silinmesi için gereken işlemler yapılmaktadır. dpAudit DOCintegration tablosuna gelen veriler bu sorgular çalıştırılarak ilgili tablolara dağıtımı sağlanmaktadır. Analize uygun hale gelen bu telemetri verilerini analiz aşamasına yönlendirip, gerekli analizleri yapabilmek amacıyla işlenmeye hazır hale gelmektedir. Tüm bu işlemleri (veri hazırlama aşamaları, önişleme, analiz) içeren bildiri hazırlığı akademik danışman ile gerçekleştirilmektedir.

1. **[dbo].[spDOCbackupHistory]**

SQL Server üzerinde alınan Full Backup , Differential Backup ve Log Backup’ların Backup alınan veri tabanlarının DatabaseName, BackupStartDate, BackupFinishDate, BackupType, BackupSizeMB ,CompSizeMB ve PhysicalDeviceName bilgileri yer almaktadır. Bu bilgiler ile alınan bir backup ne kadar sürdüğünü, türünü hem normal boyutunu hem de kompresli boyutunu ve backup’nın nereye alındığı takip edilmektedir. [dbo].[spDOCbackupHistory] stored procedure 30 dakika bir çalışıp aşağıda gösterilen select kaynak sorgusu ile backup alınan veri tabanlarının bilgilerini kaydetmektedir.

select

bs.database\_name,

bs.backup\_start\_date,

bs.backup\_finish\_date,

CASE bs.type

WHEN 'D' THEN 'Full Database'

WHEN 'L' THEN 'Log'

WHEN 'I' THEN 'Differential'

WHEN 'F' THEN 'File Level'

WHEN 'G' THEN 'File Level Differential'

WHEN 'P' THEN 'Partial'

WHEN 'Q' THEN 'Differential partial'

END AS backup\_type,

convert(varchar,cast(bs.backup\_size/1024/1024 as money),10) as 'Backup Size in MB',

convert(varchar,cast(bs.compressed\_backup\_size /1024/1024 as money),10) as 'comp Size in MB',

bmf.logical\_device\_name,

bmf.physical\_device\_name,

bs.name AS backupset\_name,

bs.description

FROM msdb.dbo.backupmediafamily bmf

INNER JOIN msdb.dbo.backupset bs ON bmf.media\_set\_id = bs.media\_set\_id

***(Sorgu 1)***

A screenshot of a computer

Description automatically generated

***Şekil 2.1.1*** *Sorgu 1’de verilen sorgunun çıktısı*

1. **[dbo].[spDOCdiskStatus]**

SQL Server file’larının bulunduğu disklerin DiskDrive, DiskLabel, DiskSizeMB, DiskSizeGB, DiskFreeSpaceMB, DiskFreeSpaceGB, DiskFreePercent bilgileri yer almaktadır. [dbo].[spDOCdiskStatus] Stored Procedure 12 saatte bir çalışmakta olup disklerin Drive ismini boyutunu MB ve GB cinsinden boş alanlarını yüzdelik olarak boş alan gibi bilgileri aşağıda gösterilmekte olan sorgu ile kaydedilmektedir.

select

DiskDrive,

DiskLabel,

DiskSizeMB,

([DiskSizeMB]/1024),

DiskFreeSpaceMB,

([DiskFreeSpaceMB]/1024),

CASE WHEN DiskSizeMB=0 THEN 0 ELSE ([DiskFreeSpaceMB]/[DiskSizeMB]) END AS DiskFreePercent

FROM(

SELECT DISTINCT dovs.logical\_volume\_name AS DiskLabel,

dovs.volume\_mount\_point AS DiskDrive,

CONVERT(numeric(18,2),dovs.total\_bytes/1048576.0) AS DiskSizeMB,

CONVERT(numeric(18,2),dovs.available\_bytes/1048576.0) AS DiskFreeSpaceMB

FROM sys.master\_files mf

CROSS APPLY sys.dm\_os\_volume\_stats(mf.database\_id, mf.file\_id) dovs

) AS Tablo

ORDER BY DiskFreeSpaceMB ASC

***(Sorgu 2)***

A table with numbers and letters

Description automatically generated

***Şekil 2.1.2:*** *Sorgu 2’de verilen sorgunun çıktısı*

1. **[dbo].[spDOCple]**

SQL Server ServerName BufferCacheHitRatio ve PageLifeExpectancy değerleri ile sunucu üzerindeki RAM kullanımını ve sorguların planlarının RAM de tutulma süreleri incelenmektedir. [dbo].[spDOCple] saklı yordam (Stored procedure) 10 saniyede bir çalışıp aşağıda gösterilmekte olan sorgu ile SQL Server’ın sorgularının yüzde kaçının RAM de tutulduğunu ve ne kadar süre RAM de kaldığının bilgilerini vermektedir.

DECLARE @Instancename varchar(50),

@PageLifeExpectancy decimal(18,2),

@BufferCacheHitRatio decimal(18,2)

SELECT

@BufferCacheHitRatio = (a.cntr\_value \* 1.0 / b.cntr\_value) \* 100.0

FROM sys.dm\_os\_performance\_counters a

JOIN (SELECT cntr\_value,object\_name FROM sys.dm\_os\_performance\_counters

WHERE counter\_name = 'Buffer cache hit ratio base' AND

object\_name = @Instancename+'Buffer Manager') b ON

a.object\_name = b.object\_name WHERE a.counter\_name = 'Buffer cache hit ratio'

AND a.object\_name = @Instancename+'Buffer Manager';

SELECT

@PageLifeExpectancy =[cntr\_value]

FROM sys.[dm\_os\_performance\_counters]

WHERE

[object\_name]= @Instancename + 'Buffer Manager'

AND [counter\_name]='Page life expectancy';

***(Sorgu 3)***

A screenshot of a computer

Description automatically generated

*(Şekil 2.1.3) Sorgu 3’de verilen sorgunun çıktısı*

1. **[dbo].[spDOCspConfig]**

SQL Server ve sunucudaki ,MinServerMemoryValue ,MinServerMemoryValueInUse ,MaxServerMemoryValue, MaxServerMemoryValueInUse, ServerMemory, ServerCores, SQLCores, MaxDopValue, MaxDopValueInUse, CostThresholdForParallelismValue, CostThresholdForParallelismValueInUse değerleri ile RAM ve CORE ayarlarını göstermektedir. SQL Server üzerinde tanımlı maksimum ve minimum RAM değerlerini SQL Server üzerinde tanımlı Max dop değerini aşağıdaki gösterilmekte olan sorgu ile kayıtlanmaktadır.

select

CAST((select [value] from sys.configurations where configuration\_id=1543) AS INT),

CAST((select [value\_in\_use] from sys.configurations where configuration\_id=1543) AS INT),

CAST((select [value] from sys.configurations where configuration\_id=1544) AS INT),

CAST((select [value\_in\_use] from sys.configurations where configuration\_id=1544) AS INT),

CAST((select int\_val from @xp\_msver where [c\_name]='PhysicalMemory') AS INT),

CAST((select Count(1) from sys.dm\_os\_schedulers where [status]='VISIBLE ONLINE') AS INT),

CAST((select int\_val from @xp\_msver where [c\_name]='ProcessorCount') AS INT),

CAST((select [value] from sys.configurations where configuration\_id=1539) AS INT),

CAST((select [value\_in\_use] from sys.configurations where configuration\_id=1539) AS INT),

CAST((select [value] from sys.configurations where configuration\_id=1538) AS INT),

CAST((select [value\_in\_use] from sys.configurations where configuration\_id=1538) AS INT)

***(Sorgu 4)***

A screenshot of a computer

Description automatically generated

***Şekil 2.1.4:*** *Sorgu 4’de verilen sorgunun çıktısı*

1. **[dbo].[spDOChaStatus]**

SQL Server Always On özelliklerini ve Always On yapısında bulunan veri tabanlarının durumlarını DateKey, CreateDate, ReplicaServerName, Role,RoleDesc, OperationalState, ConnectedState, RecoveryHealth, SyncStateDesc, DnsName, IPConfigString, Port, DatabaseName, SyncStateDesc, AvailabilityMode değerleri ile gözlemlemektedir. [dbo].[spDOChaStatus] 3 dakika bir çalışıp Always On Sağlık durumunun port bilgisin, listener adını, sekron durumlarını aşağıda gösterilmekte olan sorgu ile kayıtlanmaktadır.

Select

CONVERT(nvarchar(100),HARCS.replica\_server\_name COLLATE DATABASE\_DEFAULT),

CONVERT(int,HARS.role),

CONVERT(nvarchar(100),HARS.role\_desc COLLATE DATABASE\_DEFAULT),

CONVERT(nvarchar(100),HARS.operational\_state\_desc COLLATE DATABASE\_DEFAULT),

CONVERT(nvarchar(100),HARS.connected\_state\_desc COLLATE DATABASE\_DEFAULT),

CONVERT(nvarchar(100),HARS.recovery\_health\_desc COLLATE DATABASE\_DEFAULT),

CONVERT(nvarchar(100),HARS.synchronization\_health\_desc COLLATE DATABASE\_DEFAULT),

CONVERT(nvarchar(100),AGL.dns\_name COLLATE DATABASE\_DEFAULT),

CONVERT(nvarchar(100),AGL.ip\_configuration\_string\_from\_cluster COLLATE DATABASE\_DEFAULT),

CONVERT(int,AGL.port),

DB\_NAME(CONVERT(varchar(250),DRS.database\_id)) as DatabaseName,

CONVERT(varchar(50),DRS.synchronization\_state\_desc COLLATE DATABASE\_DEFAULT)

,AR.availability\_mode\_desc

from sys.dm\_hadr\_availability\_replica\_states HARS

LEFT JOIN sys.dm\_hadr\_availability\_replica\_cluster\_states HARCS ON HARS.replica\_id=HARCS.replica\_id

LEFT JOIN sys.availability\_group\_listeners AGL ON HARS.group\_id=AGL.group\_id

LEFT JOIN master.sys.availability\_replicas AR ON HARS.replica\_id=AR.replica\_id

JOIN sys.dm\_hadr\_database\_replica\_states DRS ON HARS.replica\_id=DRS.replica\_id

***(Sorgu 5)***

A close-up of a screen

Description automatically generated

***Şekil 2.1.5:*** *Sorgu 5’de verilen sorgunun çıktısı*

1. **[dbo].[spDOCdbInfo]**

SQL Server üzerindeki veri tabanlarının DateKey, CreateDate, DatabaseName, DatabaseCreateDate

, CompabilityLevel, CollationName, StateDesc, RecoveryModel, PageVerifyOption, FulltextEnabled

, EncryptionEnabled, LogReuseWaitDesc değerlerini kontrol etmektedir. [dbo].[spDOCdbInfo] saklı yordam (stored procedure) 12 saatte bir çalışıp veri tabanlarının bilgilerini aşağıda gösterilmekte olan sorgu ile kayıtlanmaktadır.

Select

[name],

[create\_date],

[compatibility\_level],

[collation\_name],

[state\_desc],

[recovery\_model\_desc],

[page\_verify\_option\_desc],

[is\_fulltext\_enabled],

[is\_encrypted],

[log\_reuse\_wait\_desc],

convert(bigint,[is\_read\_only]),

convert(varbinary,[replica\_id]),

suser\_sname( owner\_sid ) as OwnerName,

user\_access\_desc

FROM sys.databases

***(Sorgu 6)***

*A screenshot of a computer

Description automatically generated* ***Şekil 2.1.6:*** *Sorgu 6’da verilen sorgunun çıktısı*

1. **[dbo].[spDOCdatabaseSize]**

SQL Server üzerinde bulunan tüm veri tabanlarının DateKey, CreateDate, DatabaseName, LogSizeMB, RowSizeMB, TotalSizeMB değerleri ile veri tabanlarının boyut bilgileri gözlemlenmektedir.

[dbo].[spDOCdatabaseSize] stored procedure 1 saate bir çalışarak veri tabanlarının total boyutunu, data ve log file boyutlarını aşağıda gösterilmekte olan sorgu ile kayıtlanmaktadır.

Select

database\_name=DB\_NAME(database\_id),

log\_size\_mb=CAST(SUM(CASE WHEN type\_desc = 'LOG' THEN size END) \* 8. /1024 AS DECIMAL(18,2)),

row\_size\_mb = CAST(SUM(CASE WHEN type\_desc ='ROWS' THEN size END) \* 8. /1024 AS DECIMAL(18,2)),

total\_size\_mb = CAST(SUM(size) \* 8. /1024 AS DECIMAL(18,2))

from sys.master\_files with(NOLOCK)

where database\_id>4

group by database\_id

order by DB\_NAME(database\_id)

*(Sorgu 7)*

A screenshot of a computer

Description automatically generated

***Şekil 2.1.7****: Sorgu 7’de verilen sorgunun çıktısı*

## Hedef Verinin Toplanması

2.1 bölümünde belirtilmiş olan verilerin depolandığı tabloların script (sorgu) geçmişleri detaylıca bahsedilmiştir. Tez kapsamında kullanılacak olan telemetri verileri yukarıdakilere benzer 60 farklı iş parçacığı ile Data Platform şirketinin sunucularında depolanmaktadır. Bu süreç içinde tez kapsamında olan hedef veri setini oluşturmak için tablo birleştirme işlemi gerçekleştirilmiştir. Bu kısımda üzerinde çalışmakta olduğumuz ‘page life expectancy’ değerini etkileyen faktörlerin neler olduğuna karar verilmek üzere veri tabanı uzmanı görüşleri ile bir çalışma gerçekleştirilmiştir. Bu çalışmanın detayları ise aşağıda gösterilmektedir.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

***Şekil 2.1.1.1****: Ana Tabloların Birleştirilmesi*

Şekil 2.1.1.1 ile birlikte belirtilmiş olan ana iki tablomuzu birleştirmek için (jobStatus ve performanceMonitor) öncelikli olarak iki farklı geçici tablo oluşturulmuştur. Ardından bu tabloları özelleştirmek amacıyla gerekli olan kolonlar belirlenmiştir. Bu kolonlar belirlenirken jobStatus tablosu ile performanceMonitor tablosu arasındaki bağlantılı olan yabancı anahtar (foreign key) ve birincil anahtarlar (primary key) belirlenmiş olup buna uygun şekilde geçici (temp) tabloları oluşturulmuştur.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

***Şekil 2.1.1.2:*** *Geçici tablo oluşturulması*

Şekil 2.1.1.2 üzerinde belirtilen geçici tablolama işlemi için bir adet ‘TempPerformanceMonitor’ adında tablo oluşturulmuştur.

Bu tablo iki ana tablomuzu (jobStatus ve performanceMonitor) birleştirmek amacıyla kullanılmaktadır. İçerik bakımından irdelendiğinde başlangıçta belirlenen kolonları (InstanceID, CreateDate, [page life expectancy], [% privileged time], [transactions/sec], [write transactions/sec], [logical connections]) DOCperformanceMonitor tablosundan alınmakta olup diğer kolonlar ise DOCjobStatus tablosundaki sunucunun içinde çalışmakta olan job’ların listesi alınmaktadır. Bu job listesi sunucu özelindedir.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

***Şekil 2.1.1.3:*** *Job kayıtlarının belirlenmesi*

Şekil2.1.1.3 üzerinde gösterilen script ise DOCjobStatus tablosundan alınan job kayıtlarının başlangıç ve bitiş zamanları belirlenmekte olup (Burada DOCjobStatus tablosundan çektiğimiz StartTime’ın üzerine çalışma süresi olan DurationTime değerini ekleyip EndTime değeri oluşturulmuştur.) DOCperformanceMonitor tablosundaki değerler ile karşılaştırılması yapılmaktadır.

Amaçlanmakta olan durum ise şu şekilde özetlenebilir:   
1. Geçici (Temp) performanceMonitor tablosu oluşturulması,

2. Geçici (Temp) jobStatus tablosu oluşturulması,

3. Oluşturulan geçici tablolar üzerinden yeni bir geçici tablo oluşturulup birleştirilmesi,

4. Sonuç olarak ‘page life expectancy’ değerine etki eden faktör analizinin gerçekleştirilmesi.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

***Şekil 2.1.1.4:*** *Tabloların etiketlenme süreci*

*Şekil* 2.1.1.4’te gösterilmek olan sorgu (script) ise daha önce oluşturulan tabloların etiketlenme sürecini yapılandırmaktadır. Bu süreç için hazırlanmış olan sorgu (script) eğer jobStatus tablosundaki iş parçacıklarının (job) çalışma süresi boyunca performanceMonitor tablosundaki kayıtlar ile aynı zaman içerisinde eşleşiyorsa ilgili iş parçacığı (job) etiket değeri 1 (bir) olarak güncellenmekte, eşleşmediği takdirde etiket değeri 0 (sıfır) olarak güncellenmektedir.

Tüm bu işlemlerin ardından ‘page life expectancy’ değeri için iş parçacıklarının (job) önemi araştırılmak istenmiştir. Bu sayede veri analizi noktasında araştırmalar neticesinde bir sonuca ulaşılması beklenmektedir.

## Sistem Mimarisinin Tanımlanması

Burada kullanılacak alt yapı olarak Python alt yapısı tercih edilmiştir.   
Python alt yapısıyla birlikte, hem Google Colab üzerinden hem de Spyder üzerinden çalışmalar gerçekleştirilmektedir.   
Kullanılan kütüphaneler ise şu şekildedir;

Numpy Kütüphanesi: Bilimsel hesaplamalar için kullanılan bir kütüphanedir. Yüksek performanslı ve çok boyutlu diziler için bazı araçlar sağlar.

Math Kütüphanesi: Python’da yerleşik matematik kütüphanesidir. Çeşitli matematiksel işlevlere sahiptir.

Scikit-learn Kütüphanesi: Makine öğrenmesi için kullanılan popüler bir kütüphanedir. Sınıflandırma, regresyon modelleri, kümeleme (clustering), boyut indirgeme gibi çeşitli makine öğrenmesi algoritmalarını içermektedir. Veri ön işleme, model değerlendirme, model ayarlama gibi yerlerde çok fonksiyonlu şekilde yardımcı olmaktadır.

Pandas Kütüphanesi: Veri analizi ve manipülasyonu konularında oldukça yeteneklidir. Etiketli veya Zaman serisine sahip veriler için oldukça kullanışlıdır.

Keras Kütüphanesi: Derin öğrenme modelleri oluşturmak ve eğitmek için kullanılmaktadır. Yüksek seviyeli bir nöral ağ API’dır. Hızlı deney yapmayı ve prototip oluşturmayı desteklemektedir.

PyTorch Kütüphanesi: Bir makine öğrenmesi kütüphanesidir. Tensör işlemleri için dinamik halde bir hesaplama grafiği sağlar ve derin öğrenme modellerini desteklemektedir.

TensorFlow Kütüphanesi: Açık kaynaklı bir makine öğrenmesi kütüphanesi

dir. Tensör işlemleri, derin öğrenme modelleriyle ilgili kapsamlı bir kütüphane desteği sağlamaktadır.

Seaborn Kütüphanesi: Veri görselleştirmek amacıyla kullanılmaktadır. Çeşitli grafik alt yapıları sayesinde görsel anlamda katkı sunmaktadır.

Matplotlib Kütüphanesi: Çeşitli statik, animasyonlu görseller oluşturmaktadır. 2D ve 3D grafikler çizmek için geniş bir arayüz sunmaktadır bu durum da bilimsel ve analitik uygulamalar için çok faydalı olmaktadır.

Kullanmış olduğumuz tüm algoritma sistemlerini verilerimizi anlamlı hale getirmek amacıyla gerçekleştirilmektedir. Her bir kullandığımız sistemin kod bilgisi ve çıktıları bünyemizde mevcut olup bunlarla ilgili gerekli analizler gerçekleşmektedir. Bu analizler sonucunda elde ettiğimiz metrikleri sistem mimarisini tasarlarken kullanılmaktadır.

### Apriori Algoritması ve Birliktelik Kuralları

Birliktelik kuralları, veri madenciliği alanında yaygın olarak kullanılan bir yöntemdir. Bu kurallar, veri kümesinde yaygın olan özellik kümelerini ve bunlar arasındaki ilişkileri belirlemeye yarar. Örneğin, bir perakende mağazasının veri tabanındaki alışveriş geçmişi verileri analiz edilerek, müşterilerin belirli ürünleri aynı anda alma olasılığı gibi ilişkiler ortaya çıkarılabilir. Bu tür ilişkiler, bir özellik kümesi ile başka bir özellik kümesi arasındaki ilişkiyi ifade eder. Birliktelik kuralları, Apriori gibi bazı veri madenciliği algoritmaları ile belirlenir.

Apriori algoritması, 1994 yılında Rakesh Agrawal ve Ramakrishnan Srikant tarafından “Fast Algorithms for Mining Association Rules in Large Databases” isimli makalede ilk kez tanıtılmıştır. Bu makale, veri madenciliği için kullanılan bir kural keşfi yöntemi olarak Apriori algoritmasını tanıtmıştır.

Apriori algoritması, bir veri kümesinde sıklıkla birlikte görülen öğelerin kümelerini keşfeden bir algoritmadır. Bu algoritma “Apriori kuralı” adı verilen bir kurala dayanır; bu kurala göre, bir öğenin kümesinde bulunması, diğer öğelerin de kümesinde bulunma olasılığını arttırır. Algoritma, veri kümesinde belirli bir eşik değerin altındaki destekte olan öğeleri atarak başlar ve sonrasında kalan öğeleri kullanarak sıklıkla birlikte görünen kümeleri bulur.

Apriori algoritmasıyla çalışırken bilinmesi gereken temel kavramlar şunlardır:

Frekans: Bir veri kümesindeki özellik kümesinin occurrences sayısı olarak tanımlanır. Frekans, bir özellik kümesinin veri kümesinde kaç kez tekrar ettiğini ifade eder.

Destek: Bir veri kümesinde o özellik kümesini içeren örneklerin yüzdesi olarak tanımlanır. Destek, bir özellik kümesinin veri kümesinde ne kadar yaygın olduğunu ölçer.

Güven: Bir özellik kümesinin başka bir özellik kümesi ile oluşan ilişkisinin güvenilirliğini ölçer. Güven, A ve B’nin birlikte görülme olasılığının A’nın tek başına görülme olasılığına oranı olarak tanımlanır.

Minimum Destek: Veri kümesindeki özellik kümelerinin frekanslarını analiz etmek için kullanılan bir eşik değeridir. Örneğin, minimum destek eşiği %10 olarak belirlenirse, sadece veri setinde yer alan özellik kümeleri için destek eşiği %10'dan yüksek olanlar kabul edilir.

Apriori algoritması, müşteri davranışları, harcamalar veya ürün satışları gibi verileri analiz etmek için kullanılır. Özellikle market sepeti analizi olarak adlandırılan bir analitik teknikte sıklıkla kullanılır. Ayrıca, web site veya uygulamalarda öneri sistemlerinde de kullanılabilir. Bunların yanı sıra, kimya, genetik, finans veya sağlık gibi farklı alanlarda da başarıyla uygulanmaktadır.

Bu çalışmada veri ön işleme aşamasında hangi özelliklerin veya özellik kombinasyonlarının önemli olabileceğini ve belki de daha detaylı incelenmesi gerektiğini anlamak için apriori algoritması kullanılacaktır.

### Sınıflama ve Karar Ağaçları

Makine öğreniminde, geçmiş verileri kullanarak gelecekteki bir durumu tahmin etmek için denetimli öğrenme kullanılır. Bu, verilerin istenen bir özelliğe göre etiketlendiği denetimli öğrenme olarak bilinir. Bu tür öğrenmeyi kolaylaştırmak için bir dizi algoritma mevcuttur, bunlardan biri de karar ağacıdır.

Karar ağacı, bir sorunu çözmek için tüm potansiyel çözümleri haritalayan akış şeması benzeri bir diyagramdır. Genellikle kuruluşlar, karar ağaçlarını bir dizi kararın tüm olası sonuçlarını karşılaştırarak en uygun hareket tarzını belirlemeye yardımcı olmak için kullanır. Örneğin, bir şirketin genel merkezini hangi şehre taşıyacağına karar vermesine veya bir uydu ofis açıp açmayacağına karar vermesine yardımcı olmak için bir karar ağacı kullanılabilir.

Karar ağaçları, makine öğrenimi alanında da popüler bir araçtır. Bu ağaçlar, bir müşterinin önceki satın alma geçmişine dayanarak bir ürünü satın alıp almayacağı gibi tahminler yapmak için kullanılabilir.

Karar ağaçları, her bir iç düğümün bir özelliği, her bir dalın bir karar kuralını ve her bir yaprak düğümünün sonucunu temsil ettiği akış şeması benzeri bir yapıya sahiptir. Bir karar ağacındaki en üst düğüm kök düğüm olarak bilinir ve herhangi bir alt düğümü olmayan düğümler yaprak düğüm olarak adlandırılır. Karar ağacı algoritması, kök düğümden başlar ve bir yaprak düğüme ulaşana kadar girdi özellik değerlerine göre bir karar vererek ağaç boyunca ilerler. Yaprak düğümdeki değer, tahmin edilen çıktı değerini temsil eder.

Bir karar ağacı algoritmasında kök düğümün yaprak düğüme ayrılması şu şekilde açıklanabilir:

Kök Düğüm: Kök düğüm tüm veri kümesini temsil eder ve ağacı başlatmak için kullanılır. Ağacın başlangıç noktasıdır ve verileri maksimum bilgi kazancı veya minimum Gini Impurity sağlayan özelliğe göre böler.

İç Düğüm: Her bir iç düğüm, verileri iki veya daha fazla alt kümeye ayıran bir özelliği temsil eder. Bölme işlemi özelliğin değerine göre gerçekleştirilir ve her bir gözlemin izleyeceği yolu belirler. İç düğüm daha sonra birden fazla alt düğüme bölünür.

Yaprak Düğüm: Yaprak düğüm, verilerin daha fazla bölünemeyen bir alt kümesini temsil eder. Kendisine ulaşan gözlemler için nihai tahmini içerir. Tahmin, alt kümedeki çoğunluk sınıfına veya hedef değişkenin ortalama değerine dayanır.

Bu çalışmada, veri analizi ve öngörü yeteneklerini geliştirmek amacıyla karar ağaçlarının kullanılmasının temel nedenleri incelenmiştir.

Karar ağaçlarının tercih edilme nedenlerinden biri, algoritmanın açıklanabilir ve yorumlanabilir olmasıdır. Bu özellik, elde edilen sonuçların neden ve nasıl üretildiğini anlamanın yanı sıra, paydaşlarla etkili bir iletişim kurmayı sağlar.

Ayrıca, karar ağaçları veri setindeki değişkenlerin ve ilişkilerin hızlı bir şekilde analiz edilmesine olanak tanır. Algoritma, verileri basit kurallar kullanarak küçük parçalara böler ve bu parçalarda mevcut kalıpları belirler. Bu sayede, karmaşık veri yapıları hakkında önemli içgörüler elde edilebilir.

Karar ağaçlarının önemli bir özelliği de değişken önemi değerlendirmesidir. Bu sayede, hangi değişkenlerin sonuçları daha fazla etkilediği belirlenebilir ve modelin daha doğru bir şekilde yapılandırılması sağlanabilir.

Son olarak, karar ağaçları çeşitli uygulama alanlarına sahiptir ve sınıflandırma, regresyon, örüntü tanıma gibi birçok farklı görevde kullanılabilirler. Bu çok yönlülük, veri analitiği alanında geniş bir yelpazede çözümler sunar. Bu nedenlerle, bu çalışmada karar ağaçlarının kullanılmasıyla, veri setindeki karmaşık yapıların anlaşılması, doğru sonuçların elde edilmesi ve çeşitli makine öğrenimi görevlerinde etkili çözümler geliştirilmesi hedeflenmiştir.

### Korelasyon Matrisi

Korelasyon matrisi, bir veri setindeki değişkenler arasındaki ilişkileri belirlemek ve görsel olarak temsil etmek için yaygın olarak kullanılan bir araçtır.

Korelasyon matrisi, veri setindeki her bir değişkenin diğer değişkenlerle olan ilişkisini nicel olarak ifade eder. Bu matris genellikle Pearson, Spearman veya Kendall gibi korelasyon katsayıları kullanılarak hesaplanır. Sonuçlar genellikle -1 ile 1 arasında bir değer alır; -1 negatif bir ilişkiyi, 0 ilişki olmadığını ve 1 pozitif bir ilişkiyi temsil eder.

Korelasyon matrisi, veri setindeki ilişkileri anlamak için birinci elden bir araç sunar. Bu matris, veri setindeki değişkenler arasındaki güçlü veya zayıf ilişkileri belirlemeye yardımcı olabilir. Özellikle, yüksek korelasyonlu değişkenler birbirleriyle sıkı bir ilişki içindeyken, düşük korelasyonlu değişkenler arasında bir ilişki bulunmamaktadır.

Korelasyon matrisinin kullanılmasının temel avantajlarından biri, veri setindeki değişkenler arasındaki ilişkilerin açıkça anlaşılmasına yardımcı olmasıdır. Bu, gereksiz veya fazla değişkenlerin belirlenmesine ve modelin basitleştirilmesine olanak tanır.

Ayrıca, korelasyon matrisi görsel bir araçtır, bu da ilişkileri daha hızlı ve daha kolay anlamamıza yardımcı olur. Renk kodlu hücreler veya ısı haritaları kullanılarak, güçlü ve zayıf ilişkiler hemen fark edilebilir, bu da veri setinin daha derinlemesine analizini sağlar.

Bu çalışmada korelasyon matrisinin kullanılmasının nedeni, veri setindeki değişkenler arasındaki ilişkileri anlamak ve modelleme sürecinde daha iyi bir değişken seçimini sağlamaktır. Korelasyon matrisi, veri setimdeki değişkenler arasındaki ilişkileri görsel olarak temsil ederek analizimi daha anlamlı hale getirecek ve modelimin doğruluğunu artıracaktır.

### K-Means Kümeleme Yöntemi

K-means kümeleme yöntemi, veri madenciliği ve istatistiksel öğrenme alanlarında yaygın olarak kullanılan, gözlemleri benzer özelliklere sahip gruplara ayırmak için kullanılan denetimsiz bir öğrenme algoritmasıdır. Bu yöntem, veri noktalarını k adet küme içerisine, küme içi varyansı minimize edecek şekilde dağıtır. Her küme, o kümedeki veri noktalarının ortalamasını (merkez) temsil eden bir merkez noktası veya sentroid ile karakterize edilir.

K-means algoritması, rastgele seçilen başlangıç merkez noktaları ile başlar ve iteratif bir süreçle çalışır. Her iterasyonda, her gözlem noktası en yakın merkez noktasına atanarak kümelenir. Daha sonra, her kümenin yeni merkezi, o küme içindeki tüm noktaların ortalaması alınarak hesaplanır. Bu süreç, merkezler arasında daha fazla değişiklik olmadığında veya belirli bir iterasyon sayısına ulaşıldığında sona erer.

K-means yönteminin uygulamada başarılı olmasının anahtar faktörlerinden biri, k değerinin doğru belirlenmesidir. Bu değer, yani küme sayısı, veri setinin yapısına ve analiz edilecek özel duruma göre ayarlanmalıdır. Çeşitli yöntemlerle, örneğin dirsek yöntemi gibi heuristikler kullanarak, en uygun küme sayısına karar verilebilir.

K-means algoritmasının avantajlarından biri, büyük veri kümeleri üzerinde hızlı ve etkili bir şekilde çalışabilmesidir. Bu özellik, özellikle büyük boyutlu veri tabanlarında, veri setlerini anlamlı alt gruplara ayırma ihtiyacı duyulan durumlarda değerli olmaktadır. Ancak, algoritmanın bazı sınırlamaları da vardır; örneğin, kümeleme kalitesi başlangıç merkezlerinin seçimine büyük ölçüde bağlıdır ve algoritma, küme şekillerinin yuvarlak olmasını varsayar, bu da bazı gerçek dünya veri dağılımlarında suboptimal sonuçlar üretebilir.

Bu çalışmada, k-means kümeleme yöntemi, veri tabanımızdaki gözlemleri benzer performans özelliklerine göre gruplandırarak, anomali tespiti ve veri tabanı yönetimindeki performans iyileştirmeleri için önemli içgörüler sağlamak amacıyla kullanılacaktır. K-means algoritmasını kullanarak elde edilen kümeler, benzer davranış gösteren veri tabanı örneklerini tanımlamada ve buna göre optimizasyon stratejileri geliştirmede kullanılabilir.

### KNN (K en yakın komşu) Algoritması

K-En Yakın Komşu (KNN) algoritması, veri bilimi ve makine öğrenimi alanlarında kullanılan popüler bir denetimli öğrenme yöntemidir. Bu yöntem, bir veri noktasını sınıflandırmak veya bir değer tahmin etmek için, o noktaya en yakın olan 'k' adet komşu veri noktasına bakar. Buradaki 'k', kullanıcı tarafından belirlenen bir parametredir ve algoritmanın performansı üzerinde önemli bir etkiye sahiptir.

K-En Yakın Komşu algoritmasının çalışma prensibi, temel bir mantık üzerine kuruludur: Bir veri noktasının sınıfı, en yakın komşularının çoğunluğunun sınıfına göre belirlenir. Örneğin, bir veri noktasının yakınındaki üç komşudan ikisi A sınıfına aitse, bu veri noktası da A sınıfına atanır. Bu yöntem hem kategorik hem de sürekli verilerle çalışabilir ve sınıflandırma ile regresyon görevleri için uygundur.

Algoritmanın başarısı, doğru 'k' değerinin seçilmesine büyük ölçüde bağlıdır. Çok küçük bir k değeri, veri üzerindeki gürültüye fazla duyarlı bir model oluşturabilirken, çok büyük bir k değeri modelin aşırı genelleme yapmasına ve böylece yeni veriler üzerinde düşük performans göstermesine neden olabilir. İdeal k değeri, genellikle çapraz doğrulama yöntemleriyle belirlenir.

KNN algoritmasının uygulanması basittir, ancak hesaplama maliyeti yüksektir çünkü her sınıflandırma veya tahmin yapılırken, veri setindeki tüm noktalarla olan mesafeler hesaplanmalıdır. Bu özellik, özellikle büyük veri kümeleri söz konusu olduğunda zaman ve kaynak tüketimi açısından zorluklar yaratabilir.

Bu çalışmada, KNN algoritması, veri tabanı performans metriklerini sınıflandırmak ve potansiyel performans sorunlarını tahmin etmek için kullanılacaktır. Algoritma, performans metriklerine dayalı olarak veri tabanı örneklerinin sağlıklı veya sorunlu olarak etiketlenmesine yardımcı olacaktır. Ayrıca, algoritmanın basitliği ve anlaşılabilirliği sayesinde, modelin karar verme süreçleri analiz edilebilir ve sonuçlar kolaylıkla yorumlanabilir.

### Random Forest (Rastsal Orman)

Random Forest, sınıflandırma ve regresyon görevleri için kullanılan güçlü bir denetimli öğrenme algoritmasıdır. Bu yöntem, karar ağaçlarından oluşan bir 'orman' kullanarak çalışır. Her bir karar ağacı, rastgele seçilen bir veri alt kümesinden bağımsız olarak oluşturulur ve çoğunluk oyu veya ortalama alma yoluyla sonuçlar birleştirilir. Bu, tek bir karar ağacının karşılaşabileceği aşırı uyum (overfitting) sorununu azaltır ve genel model doğruluğunu artırır.

Random Forest, veri setindeki özelliklerin rastgele alt kümelerini seçerek ve her ağacı bu alt kümelerden biri üzerinde eğiterek işler. Bu süreç, modelin değişik veri yönlerini öğrenmesini sağlar ve böylece modelin dayanıklılığını artırır. Ayrıca, her bir karar ağacının katkısıyla, Random Forest algoritması önemli özellikleri belirlemede etkili bir araç olur, çünkü her ağaç özellik seçiminde bağımsız kararlar verir.

Bu yöntem, hem kategorik hem de sürekli verilerle etkili bir şekilde çalışabilir ve karmaşık veri yapısındaki ilişkileri yakalama yeteneği nedeniyle çok çeşitli uygulama alanlarına sahiptir. Özellikle, veri tabanı yönetimi ve performans izleme gibi alanlarda, Random Forest, olası performans düşüklüklerini tahmin etme ve sistem anomalilerini tanıma konusunda yüksek başarı gösterir.

Bu çalışmada, Random Forest algoritması, veri tabanı yönetim sistemi içindeki çeşitli performans metriklerini kullanarak sağlık durumunu sınıflandırmak için kullanılacaktır.

### Destek Vektör Makinesi

Destek Vektör Makinesi (SVM), veri sınıflandırma ve regresyon görevlerinde kullanılan güçlü bir denetimli öğrenme modelidir. SVM, özellikle iki sınıf arasındaki ayrımı maksimize eden bir hiper düzlem oluşturarak çalışır. Bu yaklaşım, veri noktalarını birbirinden ayırmak için en geniş marjı sağlamayı amaçlar, böylece yeni verilerin sınıflandırılması daha doğru ve dayanıklı olur.

SVM, lineer olmayan sınıflandırma problemleri için de çekirdek (kernel) yöntemlerini kullanarak etkili bir şekilde uygulanabilir. Çekirdek yöntemi, veri noktalarını daha yüksek boyutlu bir uzaya taşıyarak, orijinal uzayda lineer olarak ayrılamayan sınıfların, yeni uzayda lineer olarak ayrılmasını sağlar. Bu özellik, SVM'yi karmaşık veri yapılarında özellikle güçlü kılar.

SVM'nin avantajlarından biri de modelin genelleştirme yeteneğinin yüksek olmasıdır. Kullanılan veriler üzerinde çok iyi performans gösterirken, kullanılmayan yeni veriler üzerinde de stabil sonuçlar sunar.

Ayrıca, SVM, özellikle yüksek boyutlu veri setlerinde etkili olabilir. Bu, büyük ve karmaşık veri tabanı yapılarında, birden fazla performans metriğinin bir arada değerlendirilmesi gerektiğinde önemli bir avantaj sağlar. SVM, bu tür veri setlerinden önemli bilgiler çıkararak, daha etkili ve öngörülebilir veri tabanı yönetim stratejileri geliştirilmesine olanak tanır.

Sentroid: Sentroid, bir küme içindeki tüm veri noktalarının ortalamasıyla belirlenen merkez noktasıdır.

Heuristikler: Heuristikler, karmaşık sorunları çözmek veya kararlar vermek için kullanılan pratik, deneyime dayalı yöntemlerdir.

Suboptimal: Mükemmel veya ideal çözümden daha az etkili veya daha az uygun olan durumlar için kullanılır.

Occurrences: Bir olayın, bir durumun veya bir nesnenin, belirli bir zaman dilimindeki tekrarları veya mevcut oluşları için kullanılır.

### Veri Tanıtımı

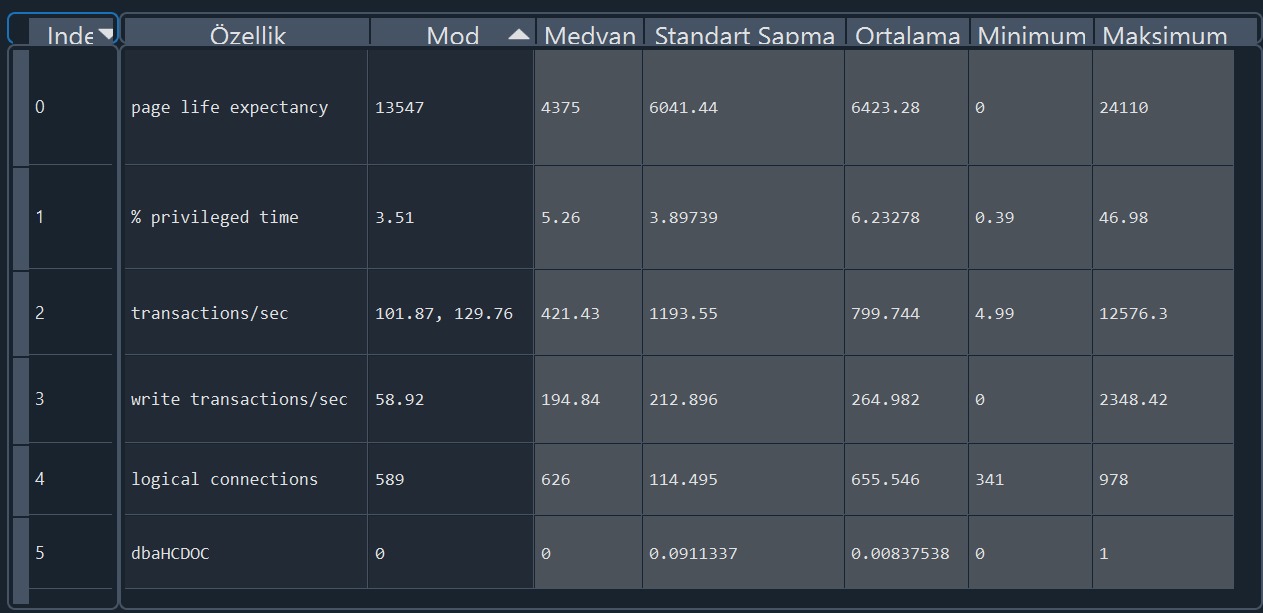
2.1.1 bölümünde detaylandırılan veri toplama süreci sonucunda elde edilen DOCperformanceMonitor Tablosu değişken tanıtımı aşağıda verilmiştir.

***Tablo 2.3.1:*** *DOCperformanceMonitor Tablosu*

|  |  |  |  |
| --- | --- | --- | --- |
| Değişken Adı | Türü | Açıklama | Değerler |
| DateKey | String | Y-A-G (Yıl – Ay -Gün) şeklinde tarih formatı | Değişken |
| CreateDate | String | Tarih ve Saat Bilgisi | Değişken |
| Page life expectancy | Nümerik | İşlemin bellekte kalma süresi | 0>= |
| Buffer cache hit ratio | Nümerik | Sorgular çalışırken execution planlarının ram’de kalma süresi | 0-100 |
| %processor time | Nümerik | Yüzdelik olarak işlemciye verdiği yük miktarı | 0-100 |
| %privileged time | Nümerik | Anlık CPU performans hızı değeri | 0-100 |
| %user time | Nümerik | Kullanıcı bazlı CPU yükselmesi değeri | 0-100 |
| Active Transaction | Nümerik | Aktif durumda olan veri tabanındaki transaction sayısı | 0>= |
| Transaction/sec | Nümerik | Saniyede gerçekleşen transaction sayısı | 0>= |
| Write transaction/sec | Nümerik | Saniyede gerçekleşen yazma işlemi yapan transaction sayısı | 0>= |
| Connection reset/sec | Nümerik | Saniyede reset atılan bağlantı sayısı | 0>= |
| Logical connections | Nümerik | Sunucuya bağlantı sayısı | 0>= |
| User Connections | Nümerik | Toplam kullanıcı bağlantı sayısı | 0>= |
| Current disk queue length | Nümerik | Disk üzerindeki işlenmeyi bekleyen işlem sayısı | 0>= |
| Avg. Disk queue length | Nümerik | Disk üzerindeki işlenmeyi bekleyen işlem sayısı ortalaması | 0>= |
| JobName | String | Veri tabanında yapılan işin ismi | Değişken |
| TimeRun | String | Çalıştırılma zamanı | Değişken |
| JobStatus | Nominal | İş’in durumu | Enabled-Disabled |
| JobOutcome | Nominal | İş’in gerçekleşirken aldığı durum | Succeeded-Failed- Cancelled |
| SendTo | Nominal | Mail gönderilme durumu | 0 veya 1 |
| Duration | String | Çalışan iş’in çalışma süresi | Değişken |

**Kaynak**: DATA PLATFORM şirketinin telemetri verileridir, yalnızca örnek amaçlıdır

Aşağıdaki Şekil 2.3.2’de verilerin etkin bir şekilde anlaşılması ve yönetilmesini sağlamak amacıyla "page life expectancy" ve diğer önemli özelliklerin belirli istatistiksel ölçütlerini hesaplayarak veri tanımlama işlemi gerçekleştirilmiştir. Sunulan istatistiksel ölçütler mod, medyan, standart sapma, ortalama, minimum ve maksimum değerler performans metriklerinin dağılımı ve genel eğilimleri hakkında derinlemesine anlayış sağlamak için kullanılmıştır.



***Şekil 2.3.2:*** *Verilerin özelliklerinin tanıtım tablosu*

Örneğin, "page life expectancy" özelliği için mod değeri 13547'dir, bu da en sık rastlanan değerin bu olduğunu gösterir ve veri tabanı işlemlerinin bu bellek yaşam süresini en çok deneyimlediği süreyi temsil eder. Ortalama değeri 6423.28 olan "page life expectancy", işlemlerin bellekte kaldığı genel sürenin bir göstergesidir ve bu veri tabanlarının genel performansını anlamak için kullanılır. Standart sapma, 6041.44, bu değerler arasındaki varyansın yüksekliğini ifade eder, bu da bize veri tabanı performansının geniş bir yelpazede değiştiğini gösterir.

### Veri Ön İşleme

#### Veriye Genel Bakış

metin, çizgi, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

***Şekil 2.4.1.1:*** *Veri setinin gösterimi*

Şekil 2.4.1.1’de ‘df.head()’ fonksiyonu kullanılarak veri setine genel bir bakış sağlanmıştır.

metin, menü, ekran görüntüsü, doküman, belge içeren bir resim

Açıklama otomatik olarak oluşturuldu

***Şekil 2.4.1.2:*** *Veri setindeki kolonların veri tipleri*

Şekil 2.4.1.2’de df.info() fonksiyonu, kolonlara göre veri tiplerini göstermek amacıyla kullanılmıştır. Veri setimizin içinde eksik veya hatalı değer olmadığı gözlemlenmiş olup nümerik ve kategorik değerler tespit edilmiştir.

metin, ekran görüntüsü, menü, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

***Şekil 2.4.1.3:*** *Veri setinin temel istatistik değerleri*

Şekil 2.4.1.3’de df.describe().T fonksiyonu ile veri setinin count (adet), mean (ortalama), std (standart sapma), min (minimum), %25 (Birinci çeyrek değerleri Q1), %50 (İkinci çeyrek değerleri Q2) , %75 (Üçüncü çeyrek değerleri Q3) ve max (maksimum) değerlerine ulaşılmıştır. Bu tablo veri seti içerisindeki kolonları analiz etmek ve anlamlandırmak amacıyla kullanılmıştır. Geniş değişken yelpazesine sahip olan veri seti için ‘InstanceID’ gibi sabit değerlerin olduğu, ‘page life expectancy’, ‘% priveleged time’, ‘transaction/sec’, ‘write transaction/sec’,’logical connections’ gibi sürekli değerlerin olduğu ve kalan tüm değerlerin ise ikili (binary) değere sahip olduğu sonucu çıkarılmıştır. Bunun neticesinde std (standart sapma) değerinin yüksek çıkmasının sebebi büyük varyans olduğunu ve bu değerlerin nümerik (sayısal) değerler olduğunu göstermektedir. Düşük std değerleri ise genellikle ikili (binary) değerlerde meydana gelmekte olduğundan bu değerlerin 0 ve 1’den oluşan değerler olduğu kanıtlamaktadır.

metin, ekran görüntüsü, diyagram, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

***Şekil 2.4.1.4:*** *‘page life expectancy’ değerinin pasta grafiği*

Şekil 2.4.1.4’te belirtilen pasta grafiğinde, ‘page life expectancy’ değerinin 0-300 aralığında ‘DÜŞÜK’ , 300-5000 aralığında ‘ORTA’, 5000> değerler için ‘YÜKSEK’ olarak belirlenmiştir. Veri setinde ‘page life expectancy’ değerinin sınırları ortalama, mod, medyan değerleri baz alınarak analiz edilip ardından veri tabanı yöneticileri ile istişare edilerek belirlenmiştir. Sonuç olarak %46.4 oranında yüksek (‘ple’ değeri için aşırı normal), %39.3 oranında orta (‘ple’ değeri için normal ama kritik), %14.3 oranında düşük (‘ple’ değeri çok riskli aykırı değer) çıkmıştır. Bu sonuçların neticesinde veri setinde yüksek oranda normal etiketli verinin varlığı tespit edilmiştir.

metin, ekran görüntüsü, diyagram, öykü gelişim çizgisi; kumpas; grafiğini çıkarma içeren bir resim

Açıklama otomatik olarak oluşturuldu

***Şekil 2.4.1.5:*** *‘page life expectancy’ değerinin sınıf dağılımı*

metin, ekran görüntüsü, diyagram, öykü gelişim çizgisi; kumpas; grafiğini çıkarma içeren bir resim

Açıklama otomatik olarak oluşturuldu

***Şekil 2.4.1.6:*** *‘page life expectancy’ değerinin çok ve az olanda birleştirme sonrası sınıf dağılımı*

Şekil 2.4.1.5’te sütun grafiği oluşturulan veri seti için toplam 69967 veri bulunmaktadır. Bu verilerin 59950 tanesinde aykırılık tespit edilmemiş olup, 10017 tanesinde aykırılık tespiti gerçekleştirilmiştir.

Şekil 2.4.1.6’da ise ilk önce Oversampling (Çok olanda birleştirme) yapıldıktan sonra veri setinde 119900 veri bulunmaktadır. Bu verilerin 59950 tanesinde aykırılık tespit edilmemiş olup, 59950 tanesinde aykırılık tespiti gerçekleştirilmiştir.

Şekil 2.4.1.6’da sonrasında Undersampling (Az olanda birleştirme) işlemi yapıldıktan sonra veri setinde 20034 veri bulunmaktadır. Bu verilerin 10.017 tanesinde aykırılık tespit edilmemiş olup, 10.017 tanesinde aykırılık tespiti gerçekleştirilmiştir. Bu çalışma ileride kullanılacak olan algoritmalara yönelik gerçekleştirilmiştir. Bu veri setleri accuracy (doğruluk) değeri yüksek olan algoritmalar, overfitting (aşırı öğrenme) durumu olan algoritmalar, underfitting (eksik öğrenme) durumu olan algoritmalar için kullanılmak üzere tasarlanmıştır.

#### Apriori Algoritması

İstatistiksel öneme sahip frekanslar ve özellik kombinasyonlarını belirlemek için Apriori algoritması kullanılmıştır. Bu algoritma ile belirli özelliklerin birlikte nasıl sıklıkla ortaya çıktığını ve her öğe kümesinin destek oranları elde edilmiştir. Destek oranı, belirli bir öğe kümesinin tüm işlemler içinde ne sıklıkla göründüğünün bir ölçüsüdür.

Analizin ilk adımında, veri tabanı işlemlerinin çeşitli özellikleri, veri setimizdeki dağılımlarına göre binarize edilmiştir. Bu, 'transactions/sec', 'write transactions/sec' ve '% privileged time' gibi sütunlar için her bir sütunun ortalamasının %20'si kadar bir eşik değeri belirleyerek kısıtlamalar gerçekleştirilmiştir. 'Logical connections' için ise ortalama değerin altındaki değerler 1 olarak işaretlenmiş olup, 'page life expectancy' değeri için ise 300 değerinden büyük olanlar için 1 (bir) olarak etiketlenmiştir. Bu ön işleme adımları Data platform şirketindeki donanımlı everi tabanı yöneticilerine danışılarak karar verilmiştir.

metin, ekran görüntüsü, yazı tipi, doküman, belge içeren bir resim

Açıklama otomatik olarak oluşturuldu

***Şekil 2.4.2.1:*** *Birliktelik kuralları*

metin, ekran görüntüsü, yazı tipi, paralel içeren bir resim

Açıklama otomatik olarak oluşturuldu

***Şekil 2.4.2.2:*** *Birliktelik kurallarının grafik üzerinde gösterimi*

Şekil 2.4.2.2’de elde edilen frekanslı öğe kümeleri, özelliklerin birlikte hangi sıklıkla var olduğunu anlamak amacıyla görselleştirilmiştir . 'Page life expectancy' özelliği, projemizde özellikle dikkate alınan bir özellik olup, en yüksek destek değerine (%85.68) sahip olduğu görülmüştür. Bu, işlemlerin büyük bir kısmının yüksek bir bellek yaşam süresine sahip olduğunu göstermektedir. Diğer özellikler arasında, 'logical connections' ve '% privileged time' da sıkça bir arada görülen ve yüksek destek değerlerine sahip özelliklerdir.

çizgi, diyagram, daire içeren bir resim

Açıklama otomatik olarak oluşturuldu

***Şekil 2.4.2.3:*** *Birliktelik kurallarının ağ grafiği üzerinde gösterimi*

Şekil 2.4.1.3’de gösterilmekte olan ağ grafiği veri setimiz üzerindeki birliktelik kuramını apriori algoritması kullanarak görselleştirme işlemi için kullanılmıştır. Bu grafik, veri analizi kısmında birliktelik kuramına en uygun değerlerin belirlenmesi ve veri tabanı yöneticilerinin bilgileri ile karşılaştırılması amacıyla yapılmıştır. Çıkarılan sonuca bakıldığında en yüksek birliktelik (Support değeri) ‘page life expectancy’ ile ‘logical connections’ değerleri arasında ‘0.5742’ olarak gözlemlenmektedir. Bu ağ grafiğini oluştururken ise minimum 0.1 değeri alınmış olup altında kalan birliktelikler yok sayılmıştır. Bunun temel sebebi, özellikle daha yüksek birlikteliklerin tespiti için gerçekleştirilmiştir.

#### Korelasyon Matrisi

Projemizde uyguladığımız korelasyon matrisi analizi, veri tabanı performans metrikleri arasındaki ilişkilerin kapsamlı bir değerlendirilmesini sağlamak amacıyla yapılmaktadır.

metin, ekran görüntüsü, kare, dikdörtgen içeren bir resim

Açıklama otomatik olarak oluşturuldu

***Şekil 2.4.3.1:*** *Korelasyon matrisi*

Şekil 2.4.3.1’de Korelasyon matrisi, özellikler arasındaki doğrusal ilişkilerin yoğunluğunu kırmızı ve mavi tonları ile ifade eden renkli bir ısı haritası olarak görselleştirilmiştir. Isı haritasındaki kırmızı tonlar pozitif korelasyonu (bir özelliğin değeri artarken diğerinin de arttığını), mavi tonlar ise negatif korelasyonu (bir özelliğin değeri artarken diğerinin azaldığını) gösterir. Matristeki daha koyu renkler, daha güçlü korelasyonları temsil ederken, açık renkler zayıf korelasyonları veya hiç korelasyon olmadığını işaret eder.

Analizimizde "page life expectancy" ve "% privileged time" arasında hafif negatif bir korelasyon (-0.15) gözlemlenmiştir, bu da bu iki metrik arasında zayıf bir ters ilişki olduğunu göstermektedir. Diğer yandan, "transactions/sec" ve "write transactions/sec" arasında orta derecede pozitif bir korelasyon (0.28) saptanmıştır; bu da işlem sıklığının ve yazma işlemlerinin sayısının beraber artma eğiliminde olduğunu ifade eder. "logical connections" ve "write transactions/sec" arasındaki orta derecede pozitif korelasyon gözlenlenmiştir. (0.49) değeri bu metriklerin birlikte izlenmesi gerektiğine dair anlayış sağlamıştır.

Kolerasyon matrisi genellikle sayısal veriler arasındaki ilişkiyi ölçmektedir. Bu yüzden kategorik değişkenlerin bu tür bir ilişkiyi ölçmede doğrudan katkısı yoktur. Bu nedenle kategorik değişkenlerden oluşan iş parçacıkları (job) korelasyona dahil edilmemiştir.

#### İş Parçacıklarının ‘page life expectancy’ değerine etkisi

metin, ekran görüntüsü, diyagram, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

***Şekil 2.4.4.1****: İş parçacıklarını değerinin page life expectancy üzerinde etkisi*

Şekil 2.4.4.1 ile ‘page life expectancy’ değerinin aykırı (anomali) olduğu durumlarda, aktif olarak görev alan iş parçacıklarının tespiti için gerçekleştirilmiştir. Bu sütun grafiğinin neticesinde, ‘page life expectancy’ değeri aykırı (anomali) durumda olduğunda dbaMaintenanceReIndex ve dbafullbackup en çok dikkat çeken iş parçacıklarıdır.

dbaMaintenanceReIndex job’ı 0.595573 olarak tespit edilmiştir. Dbafullbackup job’ı ise 0.347987 olarak tespit edilmiştir. Bu grafikteki sonuçlardan yola çıkarak ‘page life expectancy’ değerini etkileyen, hatalı duruma getiren en önemli job’ların tespiti gerçekleştirilmiştir.

#### Fp Growth Algoritması

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

***Şekil 2.4.5.1:*** *Değerlerin kategorileştirilmesi işlemi*

metin, ekran görüntüsü, yazı tipi, paralel içeren bir resim

Açıklama otomatik olarak oluşturuldu

***Şekil 2.4.5.2:*** *Verinin liste halinde kategorik gösterim*

Şekil 2.4.5.1’de belirtilen 5 farklı değere göre analiz sırasında elde edilen bulgulara ve veri tabanı yöneticilerinin söylemlerine uygun bir şekilde kategorileştirme işlemi gerçekleştirilmiştir. Kategorileştirme işleminin ardından, Şekil 2.4.5.1’de görülen transaction katmanı oluşturulmuştur. Bu katmanda her bir satır verinin belirtilen kategori değerlerine uygun şekilde etiketlenme işlemi gerçekleştirilmiştir. Veri seti, yapılan etiketlemeler sayesinde fp-growth algoritmasına uygun hale getirilmiştir.

ekran görüntüsü, metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

***Şekil 2.4.5.3:*** *fp growth algoritmasındaki rules tablosu*

metin, ekran görüntüsü, diyagram, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

***Şekil 2.4.5.4:*** *fpgrowth algoritması kategorilerin birliktelik kuramı*

metin, ekran görüntüsü, renklilik, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

***Şekil 2.4.5.5:*** *fpgrowth algoritması frekans tablosu*

Şekil 2.4.5.3’de belirtilen tablo ilişkilendirme kurallarını (association rules) içermektedir. Bu kurallar, öğe setleri arasındaki ilişkileri ifade etmektedir. Bu kuralların temel metrikleri şunlardır; Antecedents (Öncüller), kuralın sol tarafında yer alan ve bir sonucun (consequent) meydana gelmesini tetikleyen öğe veya öğe setidir. Consequents (Sonuçlar), kuralın sağ tarafında yer alan ve antecedent tarafından tetiklenen öğe veya öğe setidir. Support (Destek), bir kuralın destek değeri tüm işlemler içinde antecedent ve consequent’in bir arada görülme sıklığı anlamına gelmektedir. Confidence (Güven), antecedent içeren işlemler arasında consequent’in de bulunma olasılığını ifade etmektedir. Lift (Kaldıraç), antecedent ve consequent’in birlikte görülme olasılığının, eğer bu iki öğe bağımsız olsaydı beklenen birlikte görülme olasılığına oranını ifade etmektedir. Leverage (Kaldıraç), antecedent ve consequent’in birlikte görülme olasılığı ile bu iki öğenin bağımsız olarak görülme olasılıklarının çarpımını ifade eder. Conviction (İkna), consequent olmadan ne kadar sıklıkla meydana geldiğinin, consequent’in rastgele meydana gelme sıklığına oranıdır. Elimizdeki veri seti neticesinde lift ve confidence değerleri yüksek olan kural setleri, aralarında güçlü ilişkiler olduğunu göstermektedir. Örneğin antecedent ‘Orta’ ve consequent ‘Düşük’ olan kuralda lift ve confidence değerleri oldukça yüksektir. Bu kombinasyonda veri setinde güçlü bir ilişki olduğunu göstermektedir.

### Makine Öğrenmesi Modelleri

* + 1. **K-Means Kümeleme Algoritması**

K-Means kümeleme algoritması denetimsiz bir öğrenme algoritmasıdır. Verilerdeki doğal gruplandırmayı otomatik olarak keşfetmeyi içerir. Genellikle, bir dizi örnekte bulunan anlamlı yapıyı, açıklayıcı temel süreçleri, üretken özellikleri ve gruplamaları bulmak için bir süreç olarak kullanılır. Mevcut uygulamada transactions/sec, write transactions/sec, %priviliged time, logical connections, maintanance reIndex parametrelerinin page life expectancy üzerindeki destek oranlarını sınıflandırarak grafiksel olarak görmek için k-means kümeleme algoritması uygulanmıştır.

çizgi, metin, diyagram, öykü gelişim çizgisi; kumpas; grafiğini çıkarma içeren bir resim

Açıklama otomatik olarak oluşturuldu

***Şekil 2.5.1.1****: K-Means algoritmasında doğru k değerini belirlemek için kullanılan Elbow Methodu*

metin, ekran görüntüsü, çizgi, öykü gelişim çizgisi; kumpas; grafiğini çıkarma içeren bir resim

Açıklama otomatik olarak oluşturuldu

***Şekil 2.5.1.2:*** *K-means algoritmasında doğru k değerini belirlemek için kullanılan Silhouette skorları*

Şekil 2.5.1.1 ve Şekil 2.5.1.2’de gösterilen grafikler K-Means algoritması oluşturmadan önce küme sayısını belirlemede yardımcı bir fonksiyon niteliğindedir. Şekil 2.5.1.1’de belirtilen Elbow Method’u (Dirsek Metodu) olarak karşımıza çıkmaktadır. Bu yöntem, veri setinin kaç kümeye bölünebileceğini belirlemek amacıyla kullanılmaktadır. Grafikte x eksenindeki değerler farklı küme sayılarını gösterirken, y ekseninde ise her bir küme için hesaplanan ‘inertia’ değerlerini göstermektedir.

İnertia, küme içi varyansın toplamını ifade etmektedir. Bir kümenin içindeki noktaların küme merkezine olan uzaklıklarının karelerinin toplamı anlamına gelmektedir. Intertia, kümeleme performansının bir ölçüsü olarak kullanılmaktadır. Düşük inertia değerleri noktaların kendi küme merkezlerine daha yakın olduğunu ve dolayısıyla daha sıkı olduğunu gösterir. Bu grafikte kırılım noktası incelendiğinde k değeri için 2 ve 3 değerleri kapsamında bir çalışma yürütmek gerekmektedir.

Şekil 2.5.1.2’de gösterilen Silhouette skorları da küme sayısı arttıkça Silhoutte skorunun önemli ölçüde düştüğü gözlemlenmektedir. İki küme için silhoutte skoru çok yüksekken, daha fazla küme sayısı için skorlar düşük ve sonrasında neredeyse sabit kalması gözlemlenmektedir. K değeri için 2 ve 3 değerleri en uygun olarak Elbow Method’ta göründüğü gibi ispatlanmıştır.

metin, ekran görüntüsü, harita içeren bir resim

Açıklama otomatik olarak oluşturuldumetin, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

***Şekil 2.5.1.3:*** *Transaction/sec , %privileged time değerine göre kümeleme*

ekran görüntüsü, metin içeren bir resim

Açıklama otomatik olarak oluşturuldumetin, ekran görüntüsü, harita içeren bir resim

Açıklama otomatik olarak oluşturulduŞekil 2.5.1.3’de Transaction/sec, saniyede gerçekleşen transaction sayısı arttıkça anlık CPU hızı düşerek ‘page life expectancy’ değeri azalmaktadır.

***Şekil 2.5.1.4:*** *‘logical connections’ ve ‘transaction/sec’ değerlerine göre kümeleme*

Şekil 2.5.1.4’te k=3 ve k=2 küme sayıları ile ‘logical connections’ ve ‘transaction/sec’ parametrelerinin ‘page life expectancy’ üzerindeki etkisi izlenmektedir.

metin, ekran görüntüsü, harita, diyagram içeren bir resim

Açıklama otomatik olarak oluşturulduekran görüntüsü, metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

***Şekil 2.5.1.5:*** *logical connections’ ve ‘write transactions/sec’ değerlerine göre kümeleme*

Şekil 2.5.1.5’te k=3 ve k=2 küme sayıları ile ‘logical connections’ ve ‘write transactions/sec’ parametrelerinin ‘page life expectancy’ üzerindeki etkisi izlenmektedir.

K-Means algoritması kod parçacığı:

home\_data = pd.read\_csv('xfirma2.csv', usecols = ['% privileged time', 'transactions/sec', 'page life expectancy'])

sns.scatterplot(data = home\_data, x = 'transactions/sec', y = '% privileged time', hue = 'page life expectancy')

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(home\_data[['% privileged time', 'transactions/sec']], home\_data[['page life expectancy']], test\_size=0.33, random\_state=0)

from sklearn import preprocessing

X\_train\_norm = preprocessing.normalize(X\_train)

X\_test\_norm = preprocessing.normalize(X\_test)

from sklearn.cluster import KMeans

kmeans = KMeans(n\_clusters = 2, random\_state = 0, n\_init='auto')

kmeans.fit(X\_train\_norm)

sns.scatterplot(data = X\_train, x = 'transactions/sec', y = '% privileged time', hue = kmeans.labels\_)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import pandas as pd

import seaborn as sns

home\_data = pd.read\_csv('xfirma2.csv', usecols = ['logical connections', 'dbaMaintenanceReIndex', 'page life expectancy'])

sns.scatterplot(data = home\_data, x = 'dbaMaintenanceReIndex', y = 'logical connections', hue = 'page life expectancy')

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(home\_data[['logical connections', 'dbaMaintenanceReIndex']], home\_data[['page life expectancy']], test\_size=0.33, random\_state=0)

from sklearn import preprocessing

X\_train\_norm = preprocessing.normalize(X\_train)

X\_test\_norm = preprocessing.normalize(X\_test)

from sklearn.cluster import KMeans

kmeans = KMeans(n\_clusters = 2, random\_state = 0, n\_init='auto')

kmeans.fit(X\_train\_norm)

sns.scatterplot(data = X\_train, x = 'dbaMaintenanceReIndex', y = 'logical connections', hue = kmeans.labels\_)

sns.boxplot(x = kmeans.labels\_, y = y\_train['page life expectancy'])

Gerçekleştirilmiş olan denetimsiz öğrenme modelinin sonucunda maintanceReindex job’ı aktifleştiği durumlarda ‘logical connections’ sayısı daha düşük olduğu ve ‘page life expectancy’ değerinin azaldığı (aykırı duruma geldiği) gözlemlenmektedir.

* + 1. **KNN (K En yakın Komşu)**

K En yakın komşu (KNN) algoritması, sınıflandırma ve regresyon problemlerinde yaygın olarak kullanılan temel bir makine öğrenmesi yöntemidir. Bu çalışmada, KNN algoritmasının farklı parametre ayarlarıyla performansı değerlendirilmiştir. Accuracy (Modelin doğruluk oranı), F1 Score, Recall (Hatırlama oranı) ve Confusion Matrix (Karmaşıklık matrisi) üzerinden analiz gerçekleştirilmiştir.

***Tablo 2.5.2.1:*** *K En Yakın Komşu algoritmasının performans metrikleri*

metin, ekran görüntüsü, sayı, numara, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

KNN algoritması deneyinde, iki farklı uygulama tercih edilmiştir. Bunlar Normal KNN ve Undersampling KNN’dir. Her iki durum için de ‘k (komşu sayısı)’ 3, 4, 5 ve 6 değerleri tercih edilerek dört farklı model kurulmuştur. Undersampling yöntemi veri setindeki sınıf dengesizliğini gidermek için farklı bir bakış açısı sunmaktadır. Bu yöntemle birlikte çoğunluk sınıfından rastgele örneklerin azaltılması sonucu yeni veri seti elde edilmektedir.

Model performansını accuracy (doğruluk), F1 score, recall gibi metrikler ile değerlendirilmiştir. Yukarıdaki Şekil 2.5.2.1 ile bu durum özetlenmiştir.

Analiz sonucunda, ‘k (komşu sayısı)’ değeri arttıkça modelin doğruluk ve F1 skor metriklerinde genel bir düşüş gözlemlense de dikkate alınmayabilir. Recall değerinin artışı ise modelin azınlık sınıfını tanıma yeteneğinin geliştiğini göstermektedir.

K En yakın komşu algoritmasının uygulamaları, özellikle sınıf dengesizliği göz önünde bulundurulduğunda, farklı ‘k (komşu sayısı)’ değerlerinin modelin genel performansı üzerinde etkisinin yüksek olduğunu ortaya koymaktadır. Karmaşıklık matrisi ise model üzerinde gerçekleştirilen tahmin işleminin ne derecede doğru olduğunu göstermektedir. Undersampling ve Normal KNN modelleri arasında dikkate alınır bir fark görülememiştir.

Program Kodu:

import pandas as pd

from sklearn.neighbors import KNeighborsClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import classification\_report, confusion\_matrix, recall\_score, accuracy\_score, f1\_score

from sklearn.utils import resample

df = pd.read\_csv(r'dosya')

df['etiket'] = df['page life expectancy'].apply(lambda x: 1 if x <= 300 else 0)

label\_counts = df['etiket'].value\_counts()

df\_majority = df[df['etiket'] == label\_counts.idxmax()]

df\_minority = df[df['etiket'] == label\_counts.idxmin()]

df\_majority = df[df['etiket'] == label\_counts.idxmax()]

df\_minority = df[df['etiket'] == label\_counts.idxmin()]

df\_sifira\_gore = resample(df\_majority,

                                   replace=False,

                                   n\_samples=label\_counts.min(),  # azınlık sınıfının büyüklüğüne eşitlenir

                                   random\_state=42)

df\_sifira\_gore = pd.concat([df\_sifira\_gore, df\_minority])

X = df\_sifira\_gore.iloc[:, 2:]  # 2 ila 6 arasındaki sütunlar

y = df\_sifira\_gore['page life expectancy'] > 300  # Hedef etiket

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=84)

knn = KNeighborsClassifier(n\_neighbors=3)  # K değeri

knn.fit(X\_train, y\_train)

y\_pred = knn.predict(X\_test)

# Model performansını değerlendir

from sklearn.metrics import confusion\_matrix, recall\_score, accuracy\_score, f1\_score

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

recall = recall\_score(y\_test, y\_pred)

accuracy = accuracy\_score(y\_test, y\_pred)

f1 = f1\_score(y\_test, y\_pred)

print(classification\_report(y\_pred,y\_test))

print("Confusion Matrix:\n", conf\_matrix)

print("Accuracy:", accuracy)

print("F1 Score:", f1)

print("Recall:", recall)

* + 1. **Random Forest (Rastsal Orman)**

Random Forest (Rastsal Orman) algoritması, karar ağaçlarının bir araya gelmesiyle oluşturulan güçlü bir topluluk öğrenme modelidir. Random Forest algoritmasının farklı ‘n\_estimators (ağaç sayısı) ve ‘max\_depth’ (maksimum derinlik)’ değerleri için performans analizi sunulmaktadır. Bu parametreler modelin karmaşıklığı ve öğrenme yeteneği üzerinde doğrudan etkili olduğundan bu değerler değiştirilerek random forest algoritması tasarlanmaktadır.

***Tablo 2.5.3.1:*** *Random Forest algoritmasının performans metrikleri*

metin, ekran görüntüsü, sayı, numara, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Model tasarımını gerçekleştirirken ‘n\_estimators’ parametresi için sırasıyla 10, 25 ve 100 tercih edilmektedir. ‘max\_depth’ parametresi için sırasıyla 1, 3 ve 5 değerleri tercih edilmektedir. Bu çeşitlilik, modelin farklı parametreler ile nasıl sonuçlar verdiğini analiz etmekte fayda sağlamaktadır. Modelin accuracy (doğruluk) ve F1 skor değerleri için ‘max\_depth’ ve ‘n\_estimators’ değerleri arttıkça genel olarak artış göstermektedir. Özellikle ‘n\_estimators’ değeri 100, ‘max\_depth’ değeri 5 iken accuracy ve f1 skor değeri en yüksek sonuca ulaşmaktadır. Ayrıca bu parametrelerin yanında recall değeri de %97.75 olarak kaydedilmiştir. Bu durum ise modelin pozitif sınıfı tespit etme konusunda son derece etkili olduğunu kanıtlamaktadır.

Karmaşıklık matrislerinin analizi olarak ise düşük ‘max\_depth’ değerlerinde bazı yanlış negatiflerin (FN) ve yanlış pozitiflerin (FP) yüksek olduğunu göstermektedir. Bu durum ise modelin yetersiz öğrenme (underfitting) yaşadığının bir göstergesi olabilmektedir. Kesin bir sonuca varamamak ile birlikte ‘max\_depth’ değerinin artışı ile birlikte hatalar azalmış ve modelin doğruluk ve kesinlik kapasitesi arttığı gözlemlenmiştir.

Program Kodu:

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.preprocessing import LabelEncoder

from sklearn.metrics import accuracy\_score

# Veri setini yükleme

df = pd.read\_csv(r'dosya’)

# İlk iki sütunu atma

def categorize\_page\_life\_expectancy(value):

    if value < 301:

        return 'Dusuk'

    elif 301 <= value < 5000:

        return 'Orta'

    else:

        return 'Yuksek'

# Yeni bir sütun oluşturarak her bir örneği bu kategorilere göre etiketleme

df['page\_life\_category'] = df['page life expectancy'].apply(categorize\_page\_life\_expectancy)

# Her bir satır için 'alışveriş sepeti' oluşturma

transactions = df.groupby(['InstanceID'])['page\_life\_category'].apply(list).tolist()

df.drop(columns=['InstanceID', 'CreateDate'], inplace=True)

# Kategorik değişkenleri sayısallaştırma

label\_encoder = LabelEncoder()

df['page\_life\_category'] = label\_encoder.fit\_transform(df['page\_life\_category'])

# Bağımsız değişkenler (X) ve bağımlı değişken (y)

X = df.drop('page\_life\_category', axis=1)

y = df['page\_life\_category']

# Eğitim ve test veri setlerine ayırma

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Random Forest modelini oluşturma ve eğitme

rf\_model = RandomForestClassifier(n\_estimators=25, max\_depth = 1,random\_state=42)

rf\_model.fit(X\_train, y\_train)

# Modelin test verileri üzerinde tahmin yapma

y\_pred = rf\_model.predict(X\_test)

# Modelin başarısını değerlendirme

accuracy = accuracy\_score(y\_test, y\_pred)

recall = recall\_score(y\_test, y\_pred, average='micro')  # 'macro' olarak ayarla

f1 = f1\_score(y\_test, y\_pred, average='micro')  # 'macro' olarak ayarla

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

cm = confusion\_matrix(y\_test,y\_pred)

df\_cm = pd.DataFrame(cm, index = ['Yüksek','Orta','Düşük'], columns = ['Düşük','Orta','Yüksek'])

df\_cm.index.name = 'Actual'

df\_cm.columns.name = 'Predicted'

plt.figure(figsize = (3,3))

sns.heatmap(df\_cm, annot=True, fmt="d", cbar=False)

print("Confusion Matrix:\n", conf\_matrix)

print("Accuracy:", accuracy)

print("F1 Score:", f1)

print("Recall:", recall)

print(classification\_report(y\_pred,y\_test))

* + 1. **SVM (Destek Vektör Makineleri)**

***Tablo 2.5.4.1:*** *Destek Vektör Makinesi algoritmasının performans metrikleri*

metin, ekran görüntüsü, sayı, numara, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Destek vektör makineleri (SVM), sınıflandırma ve regresyon problemleri için etkili bir makine öğrenmesi yöntemidir. SVM algoritmasının farklı çekirdek türleri kullanılarak gerçekleştirilen performans değerlendirmesi amaçlanmaktadır. Çekirdek türleri (kernel) bakımından lineer, polynominal, RBF (Radial Basis Function) ve sigmoid incelenmektedir.

SVM modelini oluştururken test modelinin boyutu 0,2 olarak, random\_state değeri ise 42 olarak belirlenmektedir. Random\_state değeri test modelini oluştururken bir random durumu belli etmektedir. Bu değer değiştikçe veri setindeki farklı kombinasyonlardaki veriler ile çalışmayı sağlamaktadır. Şekil 2.5.4.1’deki tablodan çıkarılan bulgular, RBF çekirdeği diğer çekirdek türlerine kıyasla daha yüksek doğruluk ve F1 skoru sağlamaktadır. %99,74 accuracy (doğruluk) ve %99,84 f1 skoru olarak analiz edilmiştir. Bu durum, RBF çekirdeğinin veri setindeki özelliklerin doğrusal olmayan ilişkilerini daha iyi yakalayabildiğini göstermektedir. Linear çekirdek ise %100 doğruluk oranına sahiptir. Bu durum ise belirli bir sınıflandırma durumu için aşırı uyum sağladığı (overfitting) anlamına gelmektedir.

Karmaşıklık matrisi incelendiğinde ise, çekirdek türlerinin sınıflandırma hataları üzerindeki etkisi gayet etkileyicidir. Polynominal durumdayken TN değerinin bu kadar yüksek, diğer çekirdek türlerinde ise TP değerinin aynı oranda yüksek olduğu tespit edilmiştir. Bu durum ise SVM için hangi Kernel (çekirdek) türlerini tercih edebileceğimiz konusunda ışık tutmaktadır.

Program Kodu:

import seaborn as sns

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.neighbors import KNeighborsClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import classification\_report, confusion\_matrix, recall\_score, accuracy\_score, f1\_score

from sklearn.utils import resample

from sklearn.svm import SVC

from sklearn.model\_selection import train\_test\_split

df = pd.read\_csv(r'dosya’)

X = df.iloc[:, 2:7]  # 2 ila 6 arasındaki sütunlar

y = df['page life expectancy'] > 300  # Hedef etiket

# Veriyi eğitim ve test kümelerine bölelim

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# SVM modelini oluştur

svm\_model = SVC(kernel='linear')  # Lineer çekirdek

svm\_model.fit(X\_train, y\_train)

# Test verilerini tahmin et

y\_pred = svm\_model.predict(X\_test)

# Model performansını değerlendir

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

recall = recall\_score(y\_test, y\_pred)

accuracy = accuracy\_score(y\_test, y\_pred)

f1 = f1\_score(y\_test, y\_pred)

cm = confusion\_matrix(y\_test,y\_pred)

df\_cm = pd.DataFrame(cm, index = ['True','False'], columns = ['False','True'])

df\_cm.index.name = 'Actual'

df\_cm.columns.name = 'Predicted'

plt.figure(figsize = (3,3))

sns.heatmap(df\_cm, annot=True, fmt="d", cbar=False)

print("Confusion Matrix:\n", conf\_matrix)

print("Accuracy:", accuracy)

print("F1 Score:", f1)

print("Recall:", recall)

print(classification\_report(y\_pred,y\_test))

### Sonuçların Analiz Edilmesi

Bu bölüm, tez kapsamında ele alınan çeşitli makine öğrenmesi modellerinin veri tabanı yönetim sistemlerindeki performans iyileştirme potansiyellerini değerlendirmektedir. Araştırmanın temel amacı, veri tabanı sistemlerindeki olası sorunlara erken müdahale edebilmek ve otomatik müdahale sistemine geçişin adımlarını atmak amacıyla oluşturulmuştur.

Çalışmada kullanılan modeller arasında K-Means Kümeleme, KNN (k en yakın komşu), Random Forest (Rastsal Orman) ve SVM (Destek vektör makineleri) bulunmaktadır. Her bir modelin veri seti üzerindeki performansı çeşitli metrikler (accuracy, f1 score, recall, confusion matrix) kullanılarak ölçülmüştür. Modellerin karmaşıklık matrisleri incelenip her bir model için güçlü ve zayıf yönleri ortaya koyulmuştur.

Modeller arası karşılaştırma her bir algoritmanın avantajlarını ve kısıtlarını ortaya koymaktadır. Örneğin Random Forest (Rastsal Orman) modeli yüksek doğruluk oranları sunarken, SVM modeli farklı Kernel (Çekirdek) türleriyle değişken performans sergilemektedir. Bu durum hem tez kapsamında hem de gelecekte bu konu üzerinde tez yazımı gerçekleştirecekler açısından model seçimi konusunda dikkate alınması gereken önemli bir faktördür.

Bu tez, veri tabanı yönetim sistemlerinde erken sorun tespiti ve müdahalesi için otomatik yöntemlerin oluşturulması, geliştirilmesi, tasarlanması gibi konularda önemli bir mihenk taşı olacaktır. Önerilen modellerin gerçek zamanlı sistemlerde entegre bir şekilde kullanıma geçmesiyle birlikte veri tabanı performanslarında olağanüstü bir artış gözlemlenirken maliyetler de düşüş eğrisi gösterebilir. Ayrıca veri tabanı yöneticileri ve sistem yöneticileri için bir karar-destek araçları olarak kullanılabilecektir.

Bu tez konusunda ileride yapılacak çalışmalar, farklı veri setleri farklı senaryolar ve farklı hikayeler üzerinde ek denemeler yapılması önerilmektedir. Bu durum, modelin genelleştirilebileceğini ve dayanıklılığını arttıracaktır. Ayrıca gelişen teknolojide yeni yaklaşımların keşfi ile birlikte hem bu alanda hem de yapay zeka alanında farklı sorunlarla beraber mevcut sorunlara da çözüm bulunulacağına inanılmaktadır. Bu analizlerin tezin genel amacı doğrultusunda veri tabanı yönetim süreçlerinin otomasyon ve akıllı karar destek sistemlerinin entegrasyonunun önemi vurgulanmaktadır.

Bu tez çalışması, makine öğrenmesi tekniklerinin veri tabanı yönetim sistemlerindeki potansiyel uygulamalarını kapsamlı bir şekilde ele almıştır. Modellerin karşılaştırmalı analizi, hangi durumlarda hangi modelin tercih edileceği konusunda önemli bilgiler sağlamıştır. Ayrıca, otomatik müdahale ve akıllı karar destek sistemlerinin entegrasyonu, veri tabanı yönetimi süreçlerinin daha da optimizasyonu için çığır açıcı olabilir.

Son olarak, bu araştırma sonuçları, akademik ve profesyonel alanlarda ilgi çekici olup, benzer sorunları olan diğer organizasyonlar için de yol gösterici olabilir. Gelecek çalışmalar, bu tezin bulgularını daha geniş bir kontekste ve farklı sektörel uygulamalarla test ederek, genel geçerliğini ve uygulanabilirliğini daha da artırabilir.

# BULGULAR VE TARTIŞMA

Tezimizde, veri tabanı yönetim sistemlerine yönelik geliştirilen yerli yazılımın temel amacı, makine öğrenmesi tekniklerini kullanarak potansiyel sorunları erken tespit etmek ve otomatik müdahalelerde bulunabilmektir. Projede belirtilen amaçlar doğrultusunda, ham telemetri verilerinin işlenmesi, anormalliklerin belirlenmesi ve bu verilerin sistem performansını izlemek ve yönetmek amacıyla kullanılması planlanmıştır. Bununla birlikte, bazı hedeflerin tam olarak gerçekleştirilemediğini belirtmek isteriz; özellikle otomatikleştirilmiş bir veri tabanı yönetim sisteminin tam anlamıyla geliştirilememesi bu duruma örnek gösterilebilir.

Yapılan çalışmalar, ön işleme ve makine öğrenmesi modellerinin veri tabanı yönetimine önemli katkılar sağlayabileceğini göstermiştir. Özellikle, veri etiketleme süreçleri ve anomali tespiti yöntemleri, sistem hatalarını önceden belirlemede ve potansiyel sorunlara proaktif müdahalelerde bulunmada etkili olmuştur. Ancak, projenin uygulama kapsamındaki bazı teknik zorluklar nedeniyle, otomatikleştirme ve kapsamlı yönetim hedeflerine tam olarak ulaşılamamıştır.

Bu süreçte elde edilen deneyimler, makine öğrenmesi tekniklerinin, veri tabanı yönetim sistemlerine entegrasyonunda yaşanan pratik zorlukları ve bu teknolojilerin potansiyel avantajlarını daha iyi anlamamızı sağlamıştır. Özellikle, sistem hatalarının erken tespiti ve otomatik müdahale mekanizmaları gibi özelliklerin, veri yönetim sistemlerinin gelecekteki gelişimine önemli katkılarda bulunacağı açıktır.

Sonuç olarak, tez çalışmanız önemli bir başlangıç noktası oluşturmuş ve makine öğrenmesi tekniklerinin veri tabanı yönetim sistemlerine entegrasyonu konusunda değerli içgörüler sunmuştur. İlerleyen çalışmalar, bu alandaki eksiklikleri gidermek ve daha etkin otomatik yönetim sistemleri geliştirmek için bir temel teşkil edecektir.

**KAYNAKLAR**

[1] https://www.sciencedirect.com/science/article/abs/pii/S0167739X19321156?via%3Dihub

[2] <https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4683244&download=yes>

[3] <https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4331902>

[4] <https://ppubs.uspto.gov/pubwebapp/static/pages/ppubsbasic.html>

[5]<https://worldwide.espacenet.com/patent/search?q=database%20machine%20learning%20deep%20learning>

[6] <https://portal.turkpatent.gov.tr/>

[7] <https://dl.acm.org/doi/10.1145/3035918.3064029>

[8] <https://www.inderscienceonline.com/doi/abs/10.1504/IJESDF.2024.136024?journalCode=ijesdf>

[9]<https://medium.com/deep-learning-turkiye/k-means-algoritmas%C4%B1-b460620dd02a>

[10]<https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a>

[11]<https://www.advancinganalytics.co.uk/blog/2021/06/22/10-incredibly-useful-time-series-forecasting-algorithms>

[12] <https://www.iunera.com/kraken/fabric/top-5-common-time-series-forecasting-algorithms/>

[13] <https://www.analyticsvidhya.com/blog/2021/08/decision-tree-algorithm/>

[14] <https://www.zendesk.com/blog/machine-learning-and-deep-learning/>

[15] <https://www.ibm.com/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks/>