



VERİ ANALİTİĞİ

BİREYSEL PROJE BİTİRME RAPORU

Sunucu Log Kayıtlarında Sistem Arızası Tespiti

Selçuk Sezer

Ağustos, 2026

softITO Yazılım Bilişim Akademisi, İstanbul

İÇİNDEKİLER

1. GİRİŞ

- 1.1. Projenin Amacı
- 1.2. Proje Kısıtları

2. MATERYAL VE YÖNTEM

- 2.1. BGL Veri Kümesi Kavramı
- 2.2. Log Ayrıştırma (Drain Algoritması) Kavramı
- 2.3. Kayan Pencere (Sliding Window) ve Özellik Çıkarımı
- 2.4. Gözetimsiz Öğrenme ve Isolation Forest Kavramı
- 2.5. Uygulama Adımları
 - 2.5.1. Çalışma Ortamının Kurulması
 - 2.5.2. Ham Logların Parse Edilmesi
 - 2.5.3. Özellik Matrisinin Oluşturulması
 - 2.5.4. Eğitim/Test Ayrımı ve Ölçeklendirme
 - 2.5.5. Isolation Forest Modelinin Eğitilmesi

3. DENEYSEL SONUÇLAR

- 3.1. Isolation Forest Modelinin Deneysel Sonuçları
- 3.2. Denetimli Modellerin Karşılaştırmalı Analizi
 - 3.2.1. Denetimli Modellerin Sonuçları
 - 3.2.2. Başarısızlığın Olası Nedenleri
- 3.3. Genel Değerlendirme

4. SONUÇ

5. KAYNAKÇA

1. GİRİŞ

1.1. Projenin Amacı

Bu çalışmanın amacı, büyük ölçekli bir süper bilgisayar sistemi olan Blue Gene/L (BGL) tarafından üretilen ham sistem loglarından hareketle, gözetimsiz öğrenme (unsupervised learning) yaklaşımlarından Isolation Forest algoritması kullanılarak anormal sistem davranışlarının tespit edilmesi ve olası arızaların (failure) ileriye dönük olarak öngörülmesidir. Bu doğrultuda, ham log satırlarının yapılandırılmış olay şablonlarına dönüştürülmesi (log parsing), ardından zaman tabanlı kayan pencereler (sliding window) üzerinden istatistiksel özniteliklerin çıkarılması ve bu özniteliklere dayalı bir anomali tespit modelinin eğitilmesi süreçleri sistematik ve adım adım biçimde ele alınmıştır. Çalışma, log tabanlı arıza tahmini ve gözetimsiz anomali tespiti alanlarına ilişkin temel kavramların uygulamalı biçimde anlaşılmasına katkı sağlamayı amaçlamaktadır.

Geleneksel denetimli (supervised) sınıflandırma yöntemlerinin aksine, Isolation Forest yaklaşımı arıza etiketlerine gereksinim duymaksızın normal sistem davranışını öğrenmekte ve bu davranıştan istatistiksel olarak sapan pencereleri anomali olarak sınıflandırmaktadır. Modelin başarımı, BGL veri kümesinde önceden bilinen alert etiketleri kullanılarak bir doğrulama (sanity check) niteliğinde değerlendirilmiştir. Raporun 3. bölümünde, bu gözetimsiz yaklaşımın sonuçları, karşılaştırma amacıyla denenen denetimli modellerin sonuçlarıyla birlikte ele alınmaktadır.

1.2. Proje Kısıtları

Kısıt 1: Raporun ağırlıklı odağı, Isolation Forest tabanlı gözetimsiz anomali tespiti ile bu yaklaşım için gerekli veri hazırlama (ayırıştırma, öznitelik çıkarımı) adımlarını kapsamaktadır. Karşılaştırma amacıyla denenen denetimli modellerin (LogRobust/BiLSTM tabanlı mimariler) özet performans sonuçları ve olası başarısızlık nedenleri 3.2 bölümünde sunulmuş olmakla birlikte, bu modellerin ayrıntılı mimari tasarımı, hiperparametre optimizasyonu ve LogBERT ile XGBoost gibi diğer denetimli/derin modellerin kapsamlı karşılaştırması bu raporun kapsamı dışında bırakılmıştır.

Kısıt 2: Kullanılan veriler, Los Alamos National Laboratory tarafından paylaşılan BGL (Blue Gene/L) veri kümesi ile sınırlandırılmıştır. Farklı sistemlere veya farklı zaman dilimlerine ait log verileri üzerinde elde edilebilecek sonuçlar bu çalışmanın kapsamı dışındadır.

Kısıt 3: Anomali tespiti, pencere (window) düzeyinde gerçekleştirilmiştir; tekil log satırı veya oturum (session) düzeyinde tespit, gerçek zamanlı akış (streaming) senaryoları ve dağıtık eğitim ortamları kapsam dışında bırakılmıştır.

2. MATERİYAL VE YÖNTEM

2.1. BGL Veri Kümesi Kavramı

BGL (Blue Gene/L), Los Alamos National Laboratory bünyesinde bulunan ve 131.072 işlemciye sahip, dünyanın en hızlı süper bilgisayarlarından biri tarafından üretilen konsol loglarından oluşan bir veri kümesidir. Sistem tarafından üretilen her log satırı; zaman damgası, uyarı/durum kodu ve düz metin mesajı bileşenlerini içermektedir. Bu veri kümesi, donanım arızaları, dosya sistemi hataları, ağ sorunları ve kullanıcı seviyesi hatalar gibi çok çeşitli olay tiplerini barındırması nedeniyle log tabanlı arıza tahmini araştırmalarında yaygın olarak kullanılan standart bir kıyaslama (benchmark) veri kümesi niteliği taşımaktadır.

Veri kümesinde KERN, APP ve RAS gibi ön eklerle başlayan satırlar “alert” (yani gerçek bir arıza/uyarı durumu) olarak etiketlenmektedir. Bu projede kullanılan bölmede alert oranı yaklaşık %8,4 olarak hesaplanmış olup, sınıflar arasında belirgin bir dengesizlik gözlenmektedir. Çalışma kapsamında kullanılan ham dosya (data/BGL.log) yaklaşık 743 MB boyutunda olup, ayrıştırma (parsing) işlemi sonrasında 3.017.161 log satırına karşılık gelmektedir.

2.2. Log Ayrıştırma (Drain Algoritması) Kavramı

Ham log satırları, yapısal olmayan doğaları nedeniyle doğrudan makine öğrenmesi modellerine girdi olarak kullanılamamaktadır; bu nedenle her satırın öncelikle sabit bir mesaj şablonuna (template) dönüştürülmesi gerekmektedir. Bu amaçla, çevrimiçi (online) çalışan ve hesaplama verimliliği yüksek bir ayrıştırma yöntemi olan Drain (Depth-guided Online Log Parsing) algoritması (He vd., 2017) tercih edilmiştir. Drain algoritması, log satırlarını uzunluklarına göre gruplandırmakta, ardından bu grupları token bazlı benzerlik ölçütüne göre sabit derinlikli bir ağaç yapısı içerisinde yönlendirmektedir. Birbirine benzer her satır grubuna tam sayı tipinde benzersiz bir event_id ataması yapılmaktadır.

config.yaml dosyasında tanımlı Drain parametreleri Tablo 2.2.1'de özetlenmiştir.

Parametre	Değer	Açıklama
similarity_threshold	0,5	Log ile şablon arasındaki benzerlik eşiği
depth	4	Ağaç derinliği
max_children	100	Düğüm başına maksimum çocuk sayısı
max_log_len	128	Mesaj başına maksimum token sayısı

Tablo 2.2.1. Drain algoritması yapılandırma parametreleri

Ayrıştırma işlemi sonucunda data/parsed.csv dosyası üretilmektedir. Bu dosyadaki her satır; zaman damgası (timestamp, milisaniye), ikili alert etiketi (is_alert), olay kimliği (event_id, 1-5000 arası tam sayı) ve şablon metnini (template) içeren sütunlardan oluşmaktadır. Yapılan ayrıştırma sonucunda veri kümesinde toplam 5000 farklı olay tipi tespit edilmiştir.

Şekil 2.2.1. parsed.csv örnek satırları (buraya terminal çıktısının ekran görüntüsü eklenecektir)

2.3. Kayan Pencere (Sliding Window) ve Özellik Çıkarımı

Log verisi, doğası gereği bir zaman serisi niteliği taşımaktadır. Bu nedenle veri kümesi, sabit uzunlukta zaman pencerelerine bölünmüş ve her pencere, ilgili modelleme çerçevesinde tek bir gözlem (örnek) olarak ele alınmıştır. Pencereleme sürecine ilişkin parametreler config.yaml dosyasında tanımlanmış olup Tablo 2.3.1'de sunulmuştur.

Parametre	Değer	Açıklama
window_seconds	900	Pencere genişliği (15 dakika)
step_seconds	900	Pencere kaydırma adımı (örtüşmesiz pencereler)
horizon_seconds	900	Arıza tahmin ufku (15 dakika)
max_seq_len	90	Pencere başına maksimum olay sayısı
min_events_per_window	3	Bu sayının altında olay içeren pencereler dışlanmıştır

Tablo 2.3.1. Kayan pencere yapılandırma parametreleri

Her pencere için isolation_model.py betiğinde tanımlı build_features() fonksiyonu aracılığıyla aşağıdaki öznitelikler hesaplanmıştır:

- Olay frekans histogramı: Pencerede gözlenen her event_id'nin göreceli sıklığını temsil eden 5002 boyutlu vektör.
- Alert oranı: Penceredeki satırların alert olma oranı.
- Benzersiz olay sayısı: Pencerede gözlenen farklı olay tipi sayısı.
- Sekans uzunluğu: Penceredeki toplam olay sayısı.
- En sık 10 olayın yoğunlaşması (top-10 concentration): İlk 10 olayın toplam frekansının tüm olaylara oranı.
- Shannon entropisi: Olay dağılımının belirsizlik derecesi; $H = -\sum p \cdot \log_2(p)$ formülü ile hesaplanmıştır.
- Maksimum tek olay frekansı: Tek bir olay tipine ait en yüksek göreceli frekans değeri.

Sözü edilen 6 meta öznitelik, 5002 boyutlu frekans vektörünün sonuna eklenerek her pencere için toplam 5008 boyutlu bir öznitelik vektörü elde edilmiştir.

Etiketleme sürecinde, bir pencerenin etiketi; pencere kapanışını izleyen horizon_seconds (15 dakika) süresi içinde en az bir alert gerçekleşmiş olması durumunda 1, aksi hâlde 0 olarak belirlenmiştir. Bu yaklaşımla model, “önümüzdeki çeyrek saat içinde bir arıza meydana gelecek mi?” sorusuna yanıt verecek biçimde kurgulanmıştır.

Bu süreç sonucunda toplam 5957 pencere oluşturulmuş olup, bunların %4,48'i (267 pencere) pozitif etiketli olarak belirlenmiştir. Test bölmesinde yer alan 1192 pencereden yalnızca 22'sinin pozitif olması, problemin ciddi bir sınıf dengesizliği içerdiğini göstermektedir.

2.4. Gözetimsiz Öğrenme ve Isolation Forest Kavramı

Isolation Forest (Liu vd., 2008), anomalileri “izole etme” (isolation) ilkesine dayanarak tespit eden, ağaç tabanlı bir gözetimsiz öğrenme algoritmasıdır. Algoritmanın temel varsayımı, anomalilerin veri kümesinde az sayıda bulunduğu ve öznitelik uzayında diğer gözlemlerden belirgin biçimde uzaklaştığı, bu nedenle rastgele bölme işlemleriyle daha az sayıda adımda izole edilebildikleri yönündedir.

Algoritmanın temel bileşenleri aşağıda özetlenmiştir:

- İzolasyon Ağacı (iTree): Rastgele seçilen bir öznitelik ve rastgele belirlenen bir bölme değeri kullanılarak oluşturulan ikili ağaç yapısıdır. Anomali niteliğindeki gözlemler, kısa yol uzunlukları nedeniyle ağacın köküne yakın yapraklarda sonlanma eğilimindedir.
- Ortalama Yol Uzunluğu: Her alt örneklem için birden fazla iTree oluşturulmakta ve bir gözlemin anomali skoru, ağaçlar arasındaki ortalama yol uzunluğu üzerinden $s(x) = 2^{(-E[h(x)] / c(n))}$ formülü ile hesaplanmaktadır. Skorun 1'e yaklaşması gözlemin anomali, 0,5'in altına inmesi ise normal olarak değerlendirilmesi anlamına gelmektedir.
- Contamination Parametresi: Veride beklenen anomali oranını ifade eden ve karar eşiğinin belirlenmesinde kullanılan bir parametredir. Bu çalışmada contamination = 0,06 olarak alınmıştır.

Isolation Forest algoritmasının bu çalışmada tercih edilmesinin başlıca gerekçeleri şunlardır:

- Etiket gereksinimi bulunmaması: Gerçek sistemlerde arıza etiketleri sıklıkla mevcut olmadığından, model etiketsiz biçimde yeni sistemlere aktarılabilir.
- Yüksek boyutlu ve seyrek (sparse) öznitelik vektörleriyle verimli çalışabilmesi.
- Doğrusal olmayan ilişkileri yakalayabilmesi ve paralel hesaplamaya elverişli olması.

Model girdisini oluşturan öznitelikler farklı ölçeklerde bulunduğundan (frekans değerleri 0-1 aralığında, olay sayıları ise onlarca ile binler mertebesinde), eğitim öncesinde StandardScaler yöntemi ile standardizasyon uygulanmıştır (ortalama 0, standart sapma 1). Ölçekleyici yalnızca eğitim bölümü üzerinde öğrenilmiş, aynı dönüşüm test bölümüne de uygulanmıştır.

2.5. Uygulama Adımları

2.5.1. Çalışma Ortamının Kurulması

Çalışma için izole bir Python sanal ortamı oluşturulmuş ve gerekli bağımlılıklar requirements.txt dosyası üzerinden yüklenmiştir. Kullanılan başlıca kütüphaneler arasında scikit-learn (IsolationForest, StandardScaler, değerlendirme metrikleri), pandas ve numpy (veri işleme) ile pyyaml (yapılandırma yönetimi) yer almaktadır.

2.5.2. Ham Logların Parse Edilmesi

BGL.log dosyası, ilk eğitim çalışması sırasında Drain algoritması ile otomatik olarak ayrıştırılmış ve elde edilen sonuç data/parsed.csv dosyasında önbelleklenmiştir.

Şekil 2.5.2.1. Parse çıktısı istatistikleri (terminal ekran görüntüsü)

2.5.3. Özellik Matrisinin Oluşturulması

isolation_model.py betiği çalıştırıldığında parsed.csv dosyası belleğe yüklenmekte, zaman damgasına göre sıralanmakta ve Bölüm 2.3'te açıklanan kayan pencere ve öznitelik çıkarma süreci uygulanmaktadır. Bu süreç sonucunda 5957×5008 boyutlu bir öznitelik matrisi (X) ile 5957 elemanlı bir etiket vektörü (y) elde edilmektedir.

2.5.4. Eğitim/Test Ayrımı ve Ölçeklendirme

Veri kümesinin zaman serisi niteliği dikkate alınarak eğitim/test ayrımı kronolojik olarak yapılmış, rastgele karıştırma (shuffle) işleminden bilinçli olarak kaçınılmıştır. Verinin ilk %80'i (4765 pencere) eğitim, son %20'si (1192 pencere) ise test amacıyla ayrılmıştır. Ardından StandardScaler, yalnızca eğitim bölmesine uydurulmuş (fit) ve elde edilen dönüşüm her iki bölmeye de uygulanmıştır.

2.5.5. Isolation Forest Modelinin Eğitilmesi

Model, scikit-learn kütüphanesinin IsolationForest sınıfı kullanılarak Tablo 2.5.5.1'de listelenen hiperparametrelerle eğitilmiştir.

Parametre	Değer	Açıklama
n_estimators	200	Ağaç sayısı
contamination	0,06	Beklenen anomali oranı
max_samples	auto	Her ağaç için alt örneklem boyutu (256)
random_state	42	Tekrarlanabilirlik
n_jobs	-1	Tüm CPU çekirdeklerinin kullanımı

Tablo 2.5.5.1. Isolation Forest hiperparametreleri

Model, yalnızca eğitim bölmesindeki pencereler kullanılarak fit edilmiştir. predict fonksiyonunun çıktısında yer alan -1 (anomali) değerleri 1'e, 1 (normal) değerleri ise 0'a dönüştürülerek gerçek etiketlerle karşılaştırılabilir hâle getirilmiştir.

Şekil 2.5.5.1. Eğitim çıktısı ve metrik ekranı (terminal ekran görüntüsü)

3. DENEYSEL SONUÇLAR

3.1. Isolation Forest Modelinin DeneySEL Sonuçları

Eğitim sürecinin ardından test bölmesinde (1192 pencere, bunlardan 22'si pozitif) elde edilen genel performans metrikleri Tablo 3.1.1'de sunulmuştur.

Metrik	Değer
Accuracy	0,9706
Precision (anomali)	0,2400
Recall (anomali)	0,2727
F1-skoru (anomali)	0,2553
ROC-AUC	0,8133
Tahmin edilen anomali	25 / 1192
Gerçek anomali	22 / 1192

Tablo 3.1.1. Isolation Forest modeli genel performans metrikleri

Sınıf bazlı ayrıntılı performans raporu Tablo 3.1.2'de verilmiştir.

Sınıf	Precision	Recall	F1	Destek
Normal	0,99	0,98	0,99	1170
Anomali	0,24	0,27	0,26	22

Tablo 3.1.2. Sınıf bazlı performans metrikleri

Anomali skoruna dayalı eşik süpürme analizinin özet sonuçları Tablo 3.1.3'te sunulmuştur.

Eşik	Precision	Recall	F1	Pozitif Oranı (%)
0,2938	0,0185	1,0000	0,0362	100,0
0,2940	0,0318	0,9091	0,0615	52,7

Tablo 3.1.3. Eşik süpürme analizi özet sonuçları

Tam tabloya, `python evaluate_isolation.py` komutunun ürettiği çıktının “Threshold sweep” bölümünden erişilebilir.

Yorumlama: Varsayılan eşik değerinde (contamination = 0,06) model, gerçek anomalilerin %27,3'ünü doğru biçimde tespit ederken (recall), %97,1 genel doğruluğa ulaşmış ve ROC-AUC değeri 0,81 ile anomali skorlarının pozitif pencereleri normal pencerelerden belirgin biçimde ayırt edebildiğini ortaya koymuştur. Görece düşük precision değeri (0,24), sınıf dengesizliğinin (%1,85 pozitif) ve gözetimsiz öğrenme paradigmasının doğal bir sonucu olarak değerlendirilmelidir; model etiket görmediği için “arıza gerçekleşecek pencere” yerine “istatistiksel olarak sıra dışı pencere”yi tespit etmektedir. Eşik değeri düşürüldüğünde recall %91-100 bandına yükseltilebilmekte, ancak bu durumda üretilen alarm sayısı belirgin biçimde artmaktadır. Erken uyarı sistemi niteliğindeki operasyonel senaryolarda yüksek recall değerinin, orta düzey precision değerine kıyasla daha kritik bir öncelik taşıdığı değerlendirilmektedir.

Pencere uzunluğunun model başarımı üzerindeki etkisi de ayrıca incelenmiştir: önceki 30 dakikalık pencere yapılandırmasına kıyasla (ROC-AUC 0,80; recall %57,7), 15 dakikalık pencere yapılandırmasında ROC-AUC değeri hafifçe artmış (0,81), buna karşın recall belirgin biçimde gerilemiştir (%27,3). Kısa pencereler, anomaliyi daha dar bir zaman ufkuna sıkıştırarak skor ayırt ediciliğini artırmakla birlikte, pozitif örnek sayısının azalması (26'dan 22'ye) ve pencere başına düşen olay sayısının düşmesi nedeniyle duyarlılığı olumsuz etkilemektedir. Bu bulgu, pencere genişliği ile tespit başarımı arasında bir ödünleşim (trade-off) bulunduğunu göstermektedir.

3.2. Denetimli Modellerin Karşılaştırmalı Analizi

Proje kapsamında, gözetimsiz öğrenme tabanlı Isolation Forest modeline ek olarak, karşılaştırma amacıyla denetimli (supervised) öğrenme yöntemleri de değerlendirilmiştir. Bu çerçevede, BiLSTM (Bidirectional Long Short-Term Memory) mimarisi ve çok başlı dikkat (multi-head attention) katmanına dayanan LogRobust modeli (Zhang vd., 2019), standart ikili çapraz entropi kaybı (BCEWithLogitsLoss) ve sınıf dengesizliğine karşı geliştirilmiş Focal Loss (Lin vd., 2017) kayıp fonksiyonları ile eğitilmiştir. Ancak elde edilen sonuçlar, gözetimsiz Isolation Forest modelinin gerisinde kalmış ve beklenen başarımı gösterememiştir. Aşağıdaki alt bölümlerde bu sonuçlar ve olası başarısızlık nedenleri değerlendirilmektedir.

3.2.1. Denetimli Modellerin Sonuçları

LogRobust (BiLSTM + Multi-Head Attention) modeli, BGL veri kümesinin ayrıştırılmış (parsed) hâli üzerinde çeşitli konfigürasyonlarla eğitilmiş; her konfigürasyon için elde edilen en iyi sonuçlar, Isolation Forest modelinin sonuçlarıyla birlikte Tablo 3.2.1'de karşılaştırmalı olarak sunulmuştur.

Model	ROC-AUC	Precision	F1
BiLSTM (CrossEntropy)	0,44	0,00	0,00
BiLSTM (BCEWithLogitsLoss)	0,71	0,00	0,00
BiLSTM (Focal Loss)	0,37	0,00	0,00
Isolation Forest	0,81	0,24	0,26

Tablo 3.2.1. Denetimli modeller ile Isolation Forest modelinin karşılaştırmalı sonuçları

Tablo 3.2.1'de görüldüğü üzere, denenen üç BiLSTM konfigürasyonunun tamamında precision ve F1-skoru değerleri 0,00 olarak gerçekleşmiştir; bu durum, söz konusu modellerin test bölmesindeki pozitif (anomali) örnekleri seçilen karar eşiğinde isabetle sınıflandıramadığını, dolayısıyla pratikte kullanılabilir bir ayırt edicilik sağlayamadığını göstermektedir. BCEWithLogitsLoss ile eğitilen model, ROC-AUC bakımından (0,71) diğer iki konfigürasyona kıyasla nispeten daha iyi bir sıralama (ranking) başarımı sergilemiş olsa da, bu üstünlük sınıflandırma eşiğinde precision ya da F1 düzeyinde bir karşılığa dönüşmemiştir. Buna karşın Isolation Forest modeli, aynı test bölmesinde hem daha yüksek bir ROC-AUC değeri (0,81) hem de sıfırdan farklı precision (0,24) ve F1 (0,26) skorları elde ederek pratikte kullanılabilir bir ayırt edicilik ortaya koymuştur.

3.2.2. Başarısızlığın Olası Nedenleri

Denetimli modellerin beklenen başarıyı gösterememesinin ardında birden fazla etkenin bir arada rol oynadığı değerlendirilmektedir:

- Aşırı sınıf dengesizliği: Toplam 5957 pencerenin yalnızca %4,48'i (267 pencere) pozitif etiketli olup, test bölmesinde bu oran %1,85'e (22/1192) kadar gerilemektedir. Derin öğrenme mimarileri, genellikle sağlam bir karar sınırı öğrenebilmek için görece dengeli ve geniş hacimli pozitif örnek kümelerine ihtiyaç duymaktadır; bu denli sınırlı bir pozitif örnek sayısı, modelin çoğunluk sınıfına yönelik bir kestirim eğilimi geliştirmesine yol açabilmektedir.
- Sınırlı veri hacminin mimari karmaşıklığa oranı: Çok başlı dikkat katmanı içeren bir BiLSTM mimarisi, görece olarak yüksek sayıda öğrenilebilir parametreye sahiptir. 4765 pencerelik bir eğitim kümesinin, bu ölçekte bir mimariyi genelleyebilecek biçimde eğitmek için yetersiz kalması, aşırı öğrenme (overfitting) veya yetersiz öğrenme (underfitting) riskini artırmaktadır.
- Kayıp fonksiyonu duyarlılığı ve hiperparametre kalibrasyonu: Standart BCE kaybı, ciddi sınıf dengesizliği altında gradyanın çoğunluk sınıfı tarafından baskılanmasına yol açabilmektedir. Bu sorunu hafifletmek üzere tasarlanmış olan Focal Loss'un etkili biçimde çalışabilmesi için odaklanma (gamma) ve sınıf ağırlığı (alpha) parametrelerinin özenle kalibre edilmesi gerekmektedir; bu çalışmada elde edilen ROC-AUC değerinin (0,37) rastgele tahminin (0,50) dahi altında kalması, kayıp fonksiyonunun bu veri kümesi için yeterince kalibre edilemediğine işaret etmektedir..

3.3. Genel Değerlendirme

Elde edilen bulgular, mevcut veri hacmi, pencere düzeyinde toplulaştırılmış öznitelik temsili ve ciddi sınıf dengesizliği koşulları altında, etiket gerektirmeyen gözetimsiz Isolation Forest yaklaşımının, doğrudan aynı veri ve ölçek üzerinde eğitilen denetimli BiLSTM tabanlı modellere kıyasla daha istikrarlı ve pratikte kullanılabilir bir performans sergilediğini ortaya koymaktadır. Bu sonuç, gözetimsiz yöntemlerin yalnızca etiket maliyetinin bulunmadığı senaryolarda değil, aynı zamanda pozitif örnek sayısının son derece sınırlı olduğu erken aşama veri kümelerinde de rekabetçi bir alternatif oluşturabileceğine işaret etmektedir. Bununla birlikte, 3.2.2'de tartışılan nedenler göz önüne alındığında, denetimli modellerin başarısızlığının yöntemin doğasından değil, mevcut veri ve kalibrasyon koşullarından kaynaklandığı; uygun örnekleme, öznitelik temsili ve eşik kalibrasyonu stratejileriyle bu modellerin başarımının artırılabilirliği değerlendirilmektedir.

4. SONUÇ

Bu çalışmada, BGL süper bilgisayar logları üzerinde uçtan uca bir gözetimsiz anomali tespiti hattı kurulmuştur. Süreç; Drain algoritması ile ham logların olay şablonlarına dönüştürülmesi, 15 dakikalık kronolojik pencereler üzerinden frekans, entropi ve yoğunluk tabanlı öznitelik çıkarımı ile Isolation Forest modelinin eğitilmesi adımlarından oluşmaktadır. Karşılaştırma amacıyla, BiLSTM tabanlı LogRobust mimarisine dayanan denetimli modeller de aynı veri üzerinde değerlendirilmiştir.

Elde edilen sonuçlar (ROC-AUC 0,81), hiçbir arıza etiketi görmeden yalnızca normal davranıştan öğrenen bir modelin, gerçek alert pencerelerinin önemli bir bölümünü yakalayabildiğini ve etiket maliyetinin sıfır olduğu senaryolar için anlamlı bir erken uyarı sinyali üretebildiğini göstermiştir. Buna karşın, aynı veri üzerinde denenen denetimli BiLSTM modelleri (ROC-AUC 0,37-0,71; precision ve F1 = 0,00), ciddi sınıf dengesizliği, sınırlı pozitif örnek sayısı ve karar eşiği kalibrasyonu sorunları nedeniyle pratikte kullanılabilir bir başarımla sergileyememiştir. Bu karşılaştırma, mevcut veri koşullarında gözetimsiz yaklaşımın denetimli alternatiflere kıyasla daha güvenilir bir temel (baseline) oluşturduğunu ortaya koymaktadır.

Isolation Forest modelinde gözlenen düşük precision değeri, sınıf dengesizliği ve gözetimsiz paradigmanın beklenen bir sonucu olarak değerlendirilmiş; eşik ayarı yoluyla operasyonel tercihe göre kalibre edilebileceği gösterilmiştir. Pencere genişliği deneyleri ayrıca, kısa pencerelerin skor ayırt ediciliğini (ROC-AUC) artırdığını, ancak duyarlılığı azalttığını ortaya koymuş; pencere uzunluğu seçiminin, uygulama hedefine göre yapılması gereken kritik bir tasarım kararı olduğunu göstermiştir.

Gelecek çalışmalar için: (i) contamination parametresinin zaman içindeki anomali oranına göre dinamik olarak belirlenmesi, (ii) meta özniteliklere pencereler arası zamansal eğilim (trend) bilgisinin eklenmesi, (iii) sınıf dengesizliğini azaltmaya yönelik örnekleme tekniklerinin (SMOTE, ağırlıklı örnekleme vb.) ve karar eşiği kalibrasyonu yöntemlerinin denetimli modellere uygulanarak bu modellerin başarımlarının yeniden değerlendirilmesi ve (iv) gözetimsiz Isolation Forest modeli ile yeniden kalibre edilmiş denetimli modellerin hibrit (ensemble) biçimde birleştirilmesi önerilmektedir.

5. KAYNAKÇA

1. Olinas, R. K., & Barton, T. (2007). Blue Gene/L System Logs Dataset. Los Alamos National Laboratory — <https://www.usenix.org/cfdr-data>
2. Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation Forest. Proceedings of the IEEE International Conference on Data Mining (ICDM), 413-422.
3. He, P., Zhu, J., Zheng, Z., & Lyu, M. R. (2017). Drain: An Online Log Parsing Approach with Fixed Depth Tree. IEEE International Conference on Web Services (ICWS), 33-40.
4. Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825-2830 — <https://scikit-learn.org/stable/modules/ensemble.html#isolation-forest>
5. Xu, W., Huang, L., Fox, A., Patterson, D., & Jordan, M. (2009). Detecting Large-Scale System Problems by Mining Console Logs. Proceedings of SOSP'09.
6. Zhang, X., Xu, Y., Lin, Q., Qiao, B., Zhang, H., Dang, Y., Xie, C., Yang, X., Cheng, Q., Li, Z., Chen, J., He, X., Yao, R., Lou, J.-G., Chintalapati, M., Shen, F., & Zhang, D. (2019). Robust Log-Based Anomaly Detection on Unstable Log Data. Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE), 807-817.
7. Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal Loss for Dense Object Detection. Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2980-2988.
8. Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), 1735-1780.