GitHub Deposu:
https://github.com/tyfnacici/ec2_cpu_anomaly_detection

Veri Seti Açıklaması

Veri setim şu yapıya sahip:

- **Zaman Damgası**: Ölçümün tarih ve saati
- **Değer**: Sayısal ölçüm değeri
- **Örnekleme Aralığı**: 5 dakikalık aralıklar
- **Zaman Dönemi**: Şubat 2014'ten itibaren

Sistem Mimarisi

Bileşenler

1. **Veri Alım Katmanı**
    - **Apache Kafka**: Mesaj sıralama ve veri akışı için
    - **Konular**: `input_data`, `anomalies`
2. **İşleme Katmanı**
    - **Apache Spark**: Akış verilerini işlemek için
    - **PySpark**: Veri dönüşümü ve model çıkarımı için
3. **Makine Öğrenmesi Katmanı**
    - **Isolation Forest**: İstatistiksel anomali tespiti için
    - **LSTM Autoencoder**: Derin öğrenme tabanlı anomali tespiti için
4. **Depolama Katmanı**
    - **Yerel dosya sistemi**: Model depolama için
    - **Kontrol noktası dizini**: Akış işleme durumu için

İzolasyon Ormanı (Isolation Forest)

Kullanım Alanı:

İzolasyon Ormanı modeli, istatistiksel olarak normallerden farklılaşan verileri tespit etmek için kullanılır.

Model Parametreleri:

- **Kontaminasyon Oranı (contamination)**: 0.1 (Veri setindeki anomalilerin yaklaşık %10'unun anomali olduğu varsayılır)
- **Ağaç Sayısı (n_estimators)**: 100 (Model, 100 adet karar ağacı kullanarak veriler arasındaki ilişkileri öğrenir)
- **Rastgelelik Durumu (random_state)**: 42 (Modelin tekrarlanabilir sonuçlar üretmesi için sabit bir rastgelelik durumu kullanılır)

Eğitim Süreci:

1. Veriler işlenir ve normalleştirilir.
2. İzolasyon Ormanı modeli eğitilir ve verilerden anomaliler tespit edilir.
3. Sonuçlar kaydedilir.

LSTM Mimarisi

Kullanım Alanı:

LSTM (Uzun Kısa Süreli Bellek) modeli, zaman serisi verileri üzerinde çalışarak anomalileri tespit eder. Bu model özellikle karmaşık zaman bağımlılıklarını öğrenmekte etkilidir.

Model Parametreleri:

1. **Birinci LSTM Katmanı**:
    - **Nöron Sayısı**: 50
    - **Aktivasyon Fonksiyonu**: relu
    - **return_sequences**: True (Zaman serisi verisinin tamamını dikkate alır)
2. **Dropout Katmanı**:
    - **Oran**: 0.2 (Aşırı öğrenmeyi engellemek için bazı bağlantılar rastgele bırakılır)
3. **İkinci LSTM Katmanı**:
    - **Nöron Sayısı**: 30
    - **Aktivasyon Fonksiyonu**: relu
4. **Çıkış Katmanı (Dense)**:
    - Veri boyutuna uygun sayıda çıkış birimi içerir.
    - Çıkışlar, her bir zaman adımı için yeniden inşa edilen değerlere karşılık gelir.
5. **Dropout Katmanı**:
    - **Oran**: 0.2
6. **Optimizasyon Algoritması**: adam (Modelin hızlı ve etkin şekilde öğrenmesini sağlar)
7. **Kayıp Fonksiyonu**: mse (Ortalama Kare Hata) (Modelin tahmin ve gerçek değerler arasındaki farkını minimize eder)

Eğitim Süreci:

1. Veriler işlenir ve belirli bir uzunlukta (sequence_length = 10) sıralara dönüştürülür.
2. Model, her bir sıralı veri seti üzerinde eğitilir.
3. Eğitim sırasında veri setinin %20'si doğrulama için ayrılır.
4. Model, 50 epoch boyunca 32'lik batch boyutlarıyla eğitilir.

Verilerin Ön İşlenmesi

1. **Zaman Damgaları**:
    - Veri setinde "timestamp" sütunu varsa, bu sütun datetime formatına dönüştürülür.
2. **Zaman Temelli Özellikler**:
    - "hour" (saat) ve "day_of_week" (haftanın günü) gibi ek zaman temelli özellikler oluşturulur.
3. **Özelliklerin Ölçeklenmesi**:
    - "value", "hour" ve "day_of_week" özellikleri, **StandardScaler** ile ölçeklendirilir.

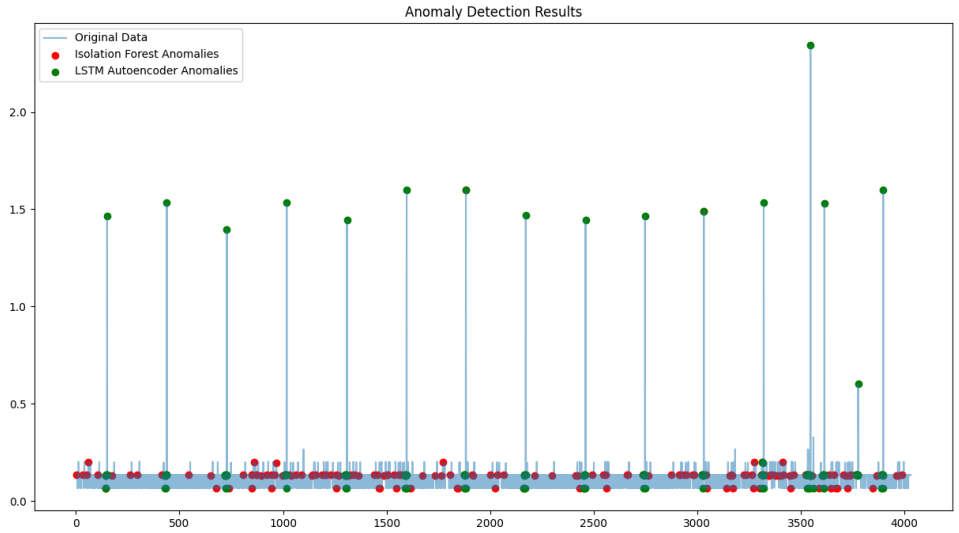
Anomali Tespiti

1. **İzolasyon Ormanı ile Anomali Tespiti**:
    - Model, ölçeklendirilmiş veriler üzerinden çalışır ve her bir örnek için -1 (anomal) veya 1 (normal) sonucunu döner.
2. **LSTM ile Anomali Tespiti**:
    - LSTM modelinin tahmin ettiği değerler ile gerçek değerler arasındaki fark (yeniden yapılandırma hatası) hesaplanır.
    - Yeniden yapılandırma hatası, standart sapmanın 3 katından büyükse bu veri noktası anomali olarak işaretlenir.
3. **Kombinasyon**:
    - Hem İzolasyon Ormanı hem de LSTM sonuçları birleştirilir. Modellerden herhangi biri bir örneği anomali olarak işaretlerse bu veri anomali olarak değerlendirilir.

Modellerin Kaydedilmesi ve Yüklenmesi

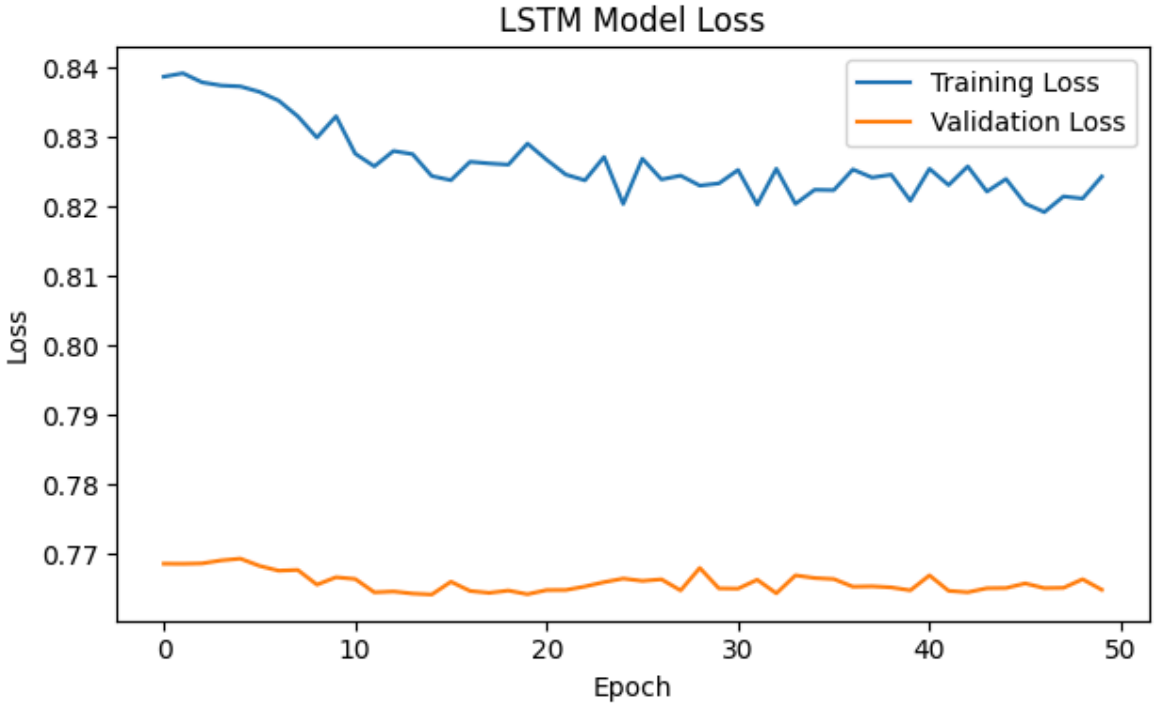
1. **Kaydetme**:
    - İzolasyon Ormanı modeli: `isolation_forest.joblib`
    - Ölçekleyici: `scaler.joblib`
    - LSTM modeli: `lstm_model.h5`
2. **Yükleme**:
    - **joblib** ile İzolasyon Ormanı ve ölçekleyici yüklenir.
    - **TensorFlow** ile LSTM modeli yüklenir.
    -

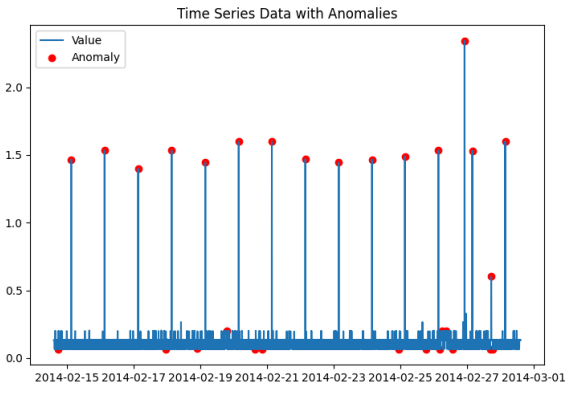
Çıktılar

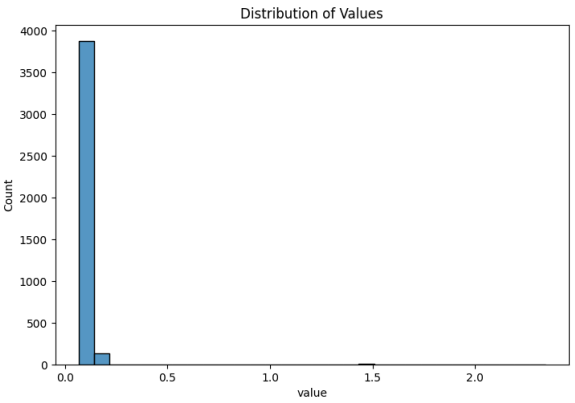


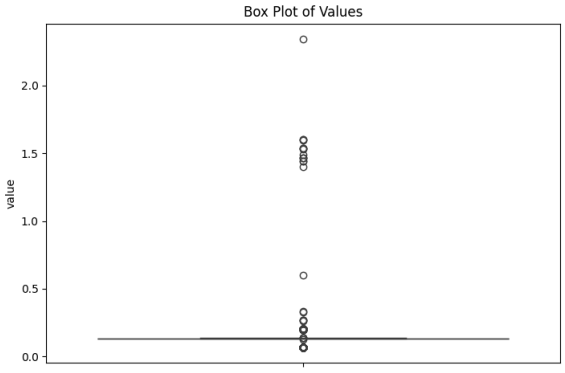
evaluation_results > {...} metrics.json > ...

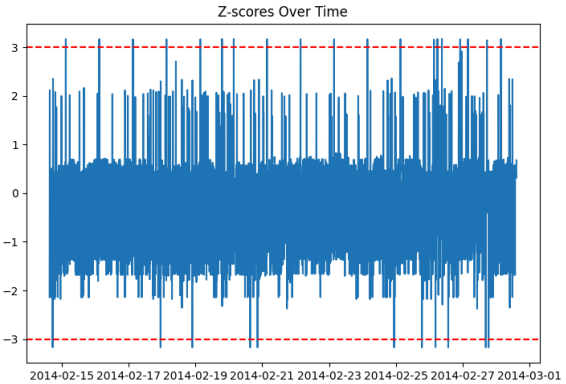

```
{
    "isolation_forest": {
        "precision": 0.09876543209876543,
        "recall": 1.0,
        "f1": 0.1797752808988764
    },
    "lstm_autoencoder": {
        "precision": 0.096969696969697,
        "recall": 1.0,
        "f1": 0.17679558011049723
    }
}
```

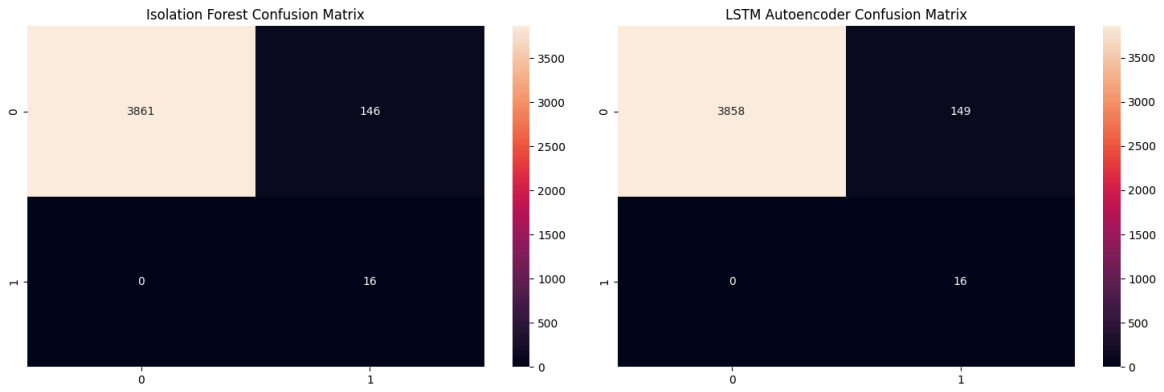

LSTM Model Loss
Training Loss
Validation Loss
Loss
Epoch
0.84
0.83
0.82
0.81
0.80
0.79
0.78
0.77
0
10
20
30
40
50


Time Series Data with Anomalies
Value
Anomaly
2014-02-15 2014-02-17 2014-02-19 2014-02-21 2014-02-23 2014-02-25 2014-02-27 2014-03-01


Distribution of Values
Count
value


Box Plot of Values
value


Z-scores Over Time
2014-02-15 2014-02-17 2014-02-19 2014-02-21 2014-02-23 2014-02-25 2014-02-27 2014-03-01



Anomali tespit sonuçları şu sütunları içerir:

- **timestamp**: Zaman damgası.
- **value**: Veri değeri.
- **is_anomaly**: Genel anomali durumu (True/False).
- **if_anomaly**: İzolasyon Ormanı anomali durumu (True/False).
- **lstm_anomaly**: LSTM anomali durumu (True/False).
- **reconstruction_error**: Yeniden yapılandırma hatası.

Tekrar Uygulama Kılavuzu

Ön Koşullar:

- Python 3.11
- Java 11
- Apache Kafka
- Apache Spark

Ortam Kurulumu:

Homebrew (macOS için)

/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"

Gerekli sistem paketlerini yükleyelim

brew install python@3.11

brew install openjdk@11

brew install kafka

Sanal ortam oluştup ve aktif edelim

python3 -m venv venv

source venv/bin/activate

Python paketlerini yükleyelim

pip install pyspark==3.4.0 kafka-python pandas numpy scikit-learn tensorflow matplotlib seaborn

Sistemi Çalıştırma

Kafka'yı başlatın:

Zookeeper başlat:
zookeeper-server-start /usr/local/etc/kafka/zookeeper.properties

Kafka'yı başlat:
kafka-server-start /usr/local/etc/kafka/server.properties

Kafka konuları oluşturun:

kafka-topics --create --topic input_data --bootstrap-server localhost:9092 --partitions 1 --replication-factor 1

kafka-topics --create --topic anomalies --bootstrap-server localhost:9092 --partitions 1 --replication-factor 1

Modelleri eğitin:

python model_training.py

Model performansını görün:

python model_evaluation.py

Spark akış uygulamasını başlatın:

python spark_streaming.py

Sonuçlar

- Gerçek zamanlı anomali tespiti sistemi başarıyla uygulandı.
- Birden fazla makine öğrenmesi yaklaşımı entegre edildi.
- Ölçeklenebilir bir akış işleme hattı oluşturuldu.