Büyük Veri Analitiği ile Anomali Tespiti

Github Linki: https://github.com/Mehdi-Can-Akbaba/BigDataProject

Kocaeli Üniversitesi
Bilişim Sistemleri Mühendisliği
Alper Sargın Ercun – 221307090

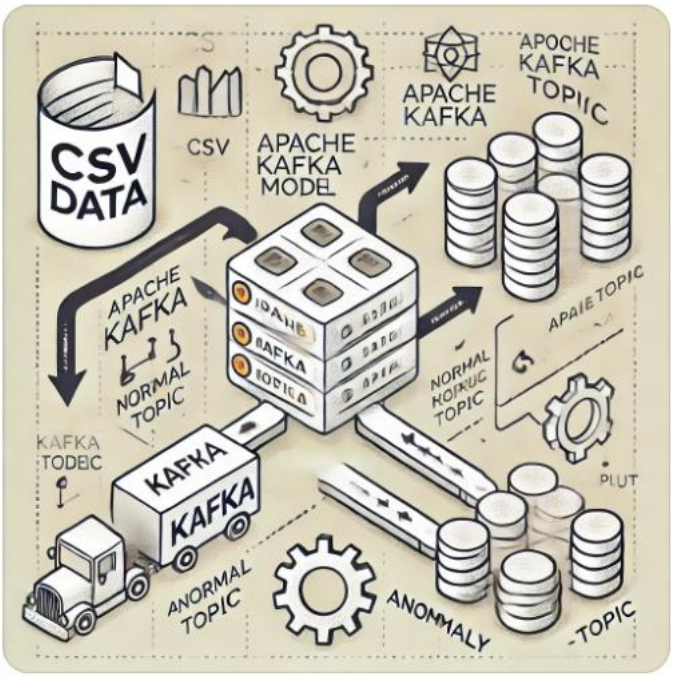
Kocaeli Üniversitesi
Bilişim Sistemleri Mühendisliği
Mehdi Can Akbaba – 221307083

Abstract - In this project, an anomaly detection system was developed using the KMeans algorithm. Real-time data streaming was achieved through Kafka, while the machine learning model classified incoming data as normal or anomalous. Combining data processing, model training, and real-time classification features, an effective system was established.

Özet – Bu projede Kmeans algoritması ile anomali veri setindeki anomaliyi tespit eden ve bunu Gerekli Kafka topiclerine gönderen bir uygulama geliştirilmiştir. Gerçek zamanlı veri akışı Kafka kullanılarak sağlanırken, makine öğrenmesi modeli gelen verileri normal ve anormal olarak sınıflandırmıştır. Veri işleme, model eğitimi ve gerçek zamanlı sınıflandırma özellikleri birleştirilerek etkili bir sistem oluşturulmuştur.

GİRİŞ

Bu proje, veri akışlarında anormalliklerin tespit edilmesini sağlayacak şekilde geliştirilmiştir. Projede, Kafka platformu ve KMeans algoritması kullanılmıştır. Gerçek zamanlı veri akışı için Kafka üzerinde bir altyapı kurulmuş, veriler input-topic üzerinden tüketilmiş ve sonuçlar normal-topic ve anomaly-topic başlıklarına gönderilmiştir.



Şekil I.

KMeans modeli, veri standardizasyonu ve kümeleme yöntemleri kullanılarak oluşturulmuş ve veriler normal ve anormal olmak üzere iki gruba ayrılmıştır. Modelin geliştirilmesi ve test edilmesi için PyCharm geliştirme ortamı kullanılmıştır. Eğitilen model, Python ortamında geliştirilmiş ve joblib kütüphanesi

kullanılarak kaydedilmiştir. Aynı zamanda projede kullanılan veri setiyle ilgili çeşitli grafiklerin çizimi için Python kütüphaneleri de projeye dahil edilmiştir.

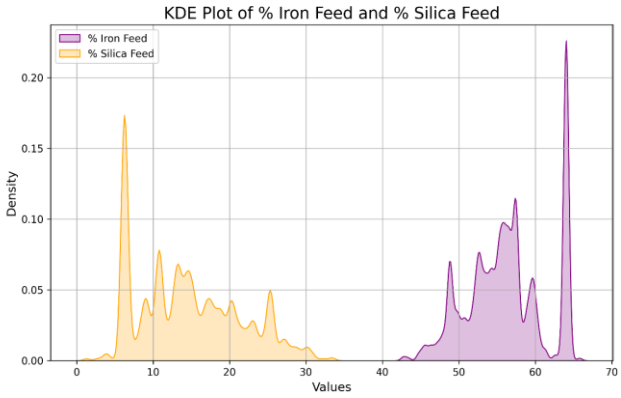
YÖNTEM

Bu projede yöntem olarak, gerçek zamanlı veri işleme ve anomali tespiti için makine öğrenmesi algoritmaları ile birleştirilmiş bir Apache Kafka altyapısı kullanılmıştır. Kafka, veri akışının temelini oluşturmuş ve üç başlıkta veri akışı sağlanmıştır: input-topic başlığı gelen veriler için, normal-topic başlığı normal olarak sınıflandırılan veriler için ve anomaly-topic başlığı ise anormal olarak sınıflandırılan veriler için tanımlanmıştır. Makine öğrenmesi modeli olarak KMeans algoritması seçilmiş ve bu algoritma, verilerin iki kümeye (normal ve anormal) ayrılmasını sağlamıştır. Modelin eğitimi sırasında veri standardizasyonu StandardScaler kullanılarak gerçekleştirilmiş, böylece farklı özelliklerdeki veriler aynı ölçeğe getirilmiştir. Eğitilen model, 1000 veri noktasından oluşan bir veri seti üzerinde test edilmiştir ve bu model joblib kütüphanesi ile kalıcı hale getirilmiştir. Veri akışı sırasında gelen veriler öncelikle temizlenmiş ve sayısal formatlara dönüştürülmüştür.

```
# Verileri standartlaştır
scaler = StandardScaler()
scaled_features = scaler.fit_transform(df)

# KMeans modelini oluştur ve eğit
n_clusters = 2  # Küme sayısını belirleyebilirsiniz
kmeans = KMeans(n_clusters=n_clusters, random_state=42, n_init=10)

# Modeli eğit
kmeans.fit(scaled_features)

# Modeli joblib ile kaydet
joblib.dump(kmeans, 'kmeans_model.joblib')
```

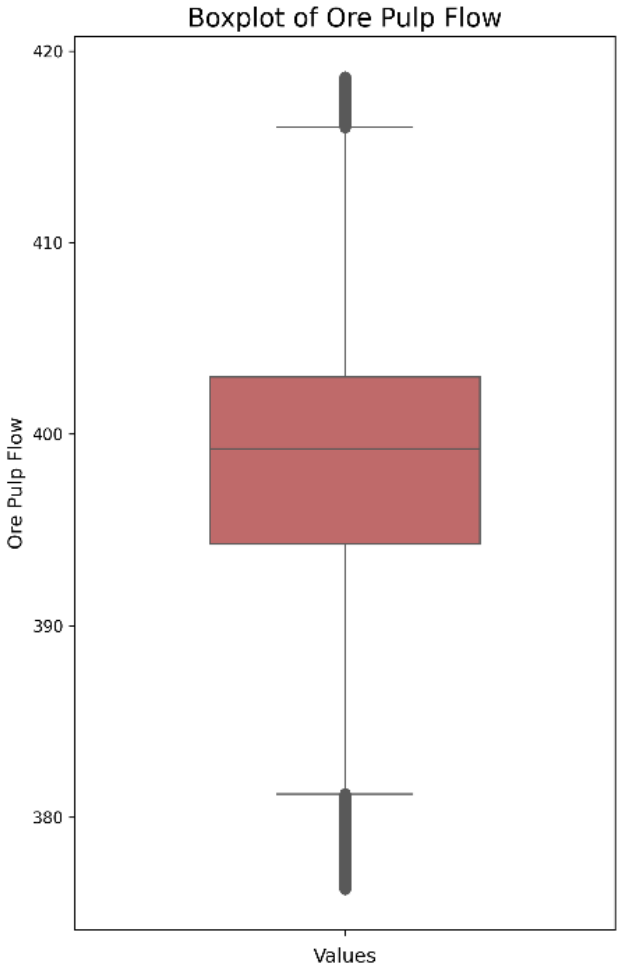
Şekil II.

Daha sonra KMeans modeli kullanılarak her bir veri noktası analiz edilmiş ve ait olduğu kümeye göre sınıflandırılmıştır. Gelen verilerin çeşitliliğini sağlamak için rastgele örnekleme yöntemi uygulanmış, veri setinin başından, ortasından ve sonundan rastgele 1000 örnek seçilmiştir. Bu yaklaşım, veri akışının daha etkin işlenmesini sağlamış ve anormalliklerin daha doğru tespit edilmesine olanak tanımıştır. Ayrıca, veri setinin dağılımlarını ve özelliklerini görselleştirmek için bir ek Python dosyası hazırlanmıştır. Bu dosyada, veri setine ait boxplot, histogram ve KDE (yoğunluk) grafikleri çizdirilmiştir. Bu grafikler, veri setinin genel yapısını anlamaya ve olası anormallikleri daha iyi tespit etmeye yardımcı olmuştur. Örnek olması açısından 3 çeşit grafik aşağıda verilmiştir.



Şekil III.

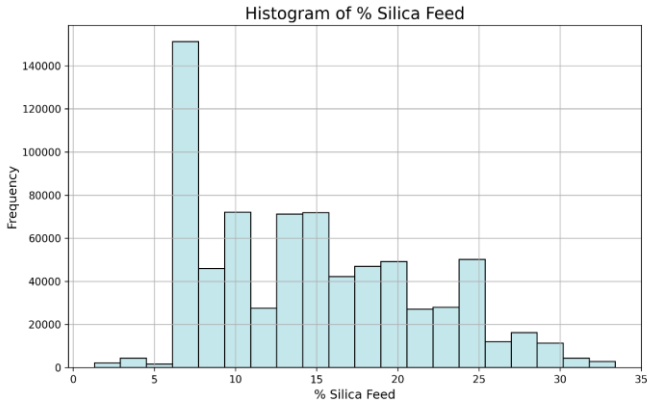
Yukarıdaki şekilde bulunan KDE (Kernel Density Estimate) grafiği, bir sürekli değişkenin olasılık yoğunluğunu gösterir. Yukarıdaki **KDE Plot of % Iron Feed and % Silica Feed** grafiğinde, iki farklı değişkenin (% Iron Feed ve % Silica Feed) yoğunlukları karşılaştırılmıştır. Her bir değişkenin eğrisi, veri setinin bu özelliklerdeki yoğunluğunu ve potansiyel veri gruplarını net bir şekilde görselleştirir. Örneğin, grafikte **% Iron Feed** yoğunluğunda belirli bir zirve noktası, bu özelliğin veri setinde sıkça tekrar eden bir değeri olduğunu gösterir. Benzer şekilde, **% Silica Feed** yoğunluğu, farklı bir eğri profili sergileyerek bu özelliğin farklı bir dağılıma sahip olduğunu ortaya koymaktadır. KDE grafikleri, veri setindeki genel dağılımları anlamak ve anormallikleri belirlemek için oldukça etkili bir araçtır.



Şekil IV.

Diğer bir görselleştirme yöntemi olan boxplot, veri setinde bulunan "Ore Pulp Flow" adlı bir özelliğin değerlerini temsil etmektedir. Grafikte, kutu bölgesi veri setindeki verilerin yüzde 25'lik (alt çeyrek) ve yüzde 75'lik (üst çeyrek) dilimlerini kapsamaktadır. Kutunun içindeki yatay çizgi ise verilerin medyanını göstermektedir. Kutunun dışındaki çizgiler (whisker) minimum ve maksimum değerleri ifade ederken, bu sınırların dışında kalan noktalar aykırı (outlier) olarak değerlendirilmektedir.

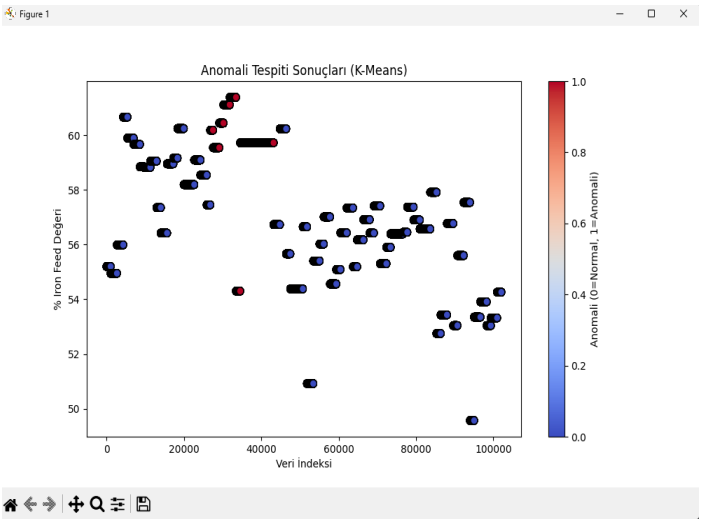
Bu grafikte görüldüğü üzere, bu sütundaki veri seti büyük ölçüde 390 ile 410 arasında yoğunlaşmıştır. Ancak, 380 ve 420 değerleri civarında aykırı veriler bulunmaktadır. Bu aykırı değerler, anomali tespiti sistemimizin kritik noktalarını oluşturabilir, çünkü modelin bu tür verileri aykırı olarak sınıflandırması beklenir. Boxplot'un proje bağlamındaki rolü, veri setinin özelliklerini ve olası sorunlu noktalarını hızlıca analiz ederek modelin doğruluğunu ve performansını artırmak için veri ön işleme aşamasına rehberlik etmektir.



Şekil V.

Son olarak, histogram grafiklerinde ise veri değerlerinin farklı aralıklara göre sıklığı gösterilmiştir. Histogramlar, veri setindeki dağılımı anlamada ve yoğunluğun hangi aralıkta olduğunu belirlemede kullanılmıştır. Bu üç görselleştirme yöntemi, veri setinin detaylı bir şekilde analiz edilmesine olanak sağlamış ve KMeans modelinin daha etkili bir şekilde kullanılmasına katkıda bulunmuştur.

Aşağıda Kmeans sonucunda anomali tespiti sonuçları verilmiştir.



Şekil VI.

SONUÇ

Veri setimizi Kmeans algoritması ile anomali tespitine göre model eğitimi tamamladık. Model çıktısını daha sonra Kafka'dan çekeceğimiz verileri filtrelemede kullandık. Kmeans algoritması ile eğitilen modeli .joblib formatında kaydederek daha sonrası için hazır hale getirdik. Bu model çıktısını daha sonra veri filtrelemede kullandık.

Proje için gerekli olan Zookeper ve Kafka yazılımlarının kurulumlarını gerçekleştirdik. Zookeeper server'ını komut istemcisi içerisinde "**zkserver**" komutu ile başlatarak Kafka işlemlerini yapabilmemiz sağlandı. Kafka programını ise farklı bir komut istemcisi yani farklı bir cmd içerisinde "**kafka-server-start.bat C:\kafka_2.11-1.1.0\config\server.properties**" komutu ile başlattık. Bu komut sonrası iki komut istemcisi üzerinde oluşan değişikleri gözlemledik.

Kafka üzerinde input, anomaly ve normal topiclerini oluşturduk. Komut istemcisi kullanarak " **kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic input-topic**" kodu ile input topic oluşmuş oldu. Diğer topiclerimizi de koddaki topic adını değiştirerek oluşturduk. Komut istemcisine "**kafka-topics.bat --list --zookeeper localhost:2181**" kodunu yazarak var olan topicleri listelememiz sağlandı. Oluşturduğumuz topiclerin var olup olmadıklarını bu kod ile anlayabildik. Veri setimizi yani .csv dosyamızın içeriğini Kafka'da istediğimiz topic'e gönderebilmemiz için komut istemcisinde kullanmamız gereken "**kafka-console-producer.bat –broker-list localhost:9092 –topic input-topic < db.csv**" bu koddur. Bu kod ile var olan csv dosyamızı Kafka'da istenilen topice yani input-topic'e verileri gönderdik.

Python projesi ile daha sonra input-topicinden verileri aldık ve Kmeans modelini kullanarak anomali tespiti gerçekleşti ve normal-topic ve anomaly-topic'lerine ilgili veriler gönderildi. Gönderilen verileri canlı bir şekilde görüntülememiz için aşağıdaki kodları kullandık. "**kafka-console-consumer.bat –bootstrap-server localhost:9092 –topic normal-topic –from-beginning**" kodu ve "**kafka-console-consumer.bat –bootstrap-server localhost:9092 –topic anomaly-topic –from-beginning**" kodu topiclere gönderilen verileri okumamıza olanak sağladı.

En sonunda Kmeans algoritması ile bir model eğitimi gerçekleşti. Bu model çıktısını Kafka'dan aldığımız verileri filtrelemek için kullandık ve iki farklı topice gönderilmesini gözlemlemiş olduk.

KAYNAKÇA

1. https://archive.apache.org/dist/spark/spark-2.4.0/
2. https://medium.com/@sinemhasircioglu/apache-spark-kurulumu-windows-4d411a5c9b43
3. https://www.youtube.com/watch?v=tC2xHj-HyNQ
4. https://kafka.apache.org/downloads
5. https://fatihdagli.home.blog/2019/10/06/apache-kafka-zookeeper-kurulum-ve-konfigurasyonlari/
6. https://www.jetbrains.com/pycharm/
7. https://www.python.org/downloads/
8. https://enes-eren.medium.com/python-matplotlib-ile-grafik-%C3%A7izme-4611f898308
9. https://zookeeper.apache.org/releases.html