KREDİ KARTI DOLANDIRICILIĞI ANOMALİ TESPİTİ

CREDIT CARD FRAUD ANOMALY DETECTION

BUSE YENER, HELEN HACER UZUNÇAYIR

BİLİŞİM SİSTEMLERİ MÜHENDİSLİĞİ

KOCAELİ ÜNİVERSİTESİ

busee.yener@gmail.com

BİLİŞİM SİSTEMLERİ MÜHENDİSLİĞİ

KOCAELİ ÜNİVERSİTESİ

211307061@kocaeli.edu.tr

Özet

Bu dökümanda, kredi kartı dolandırıcılığı tespiti için veri seti üzerinde ön işleme ve görselleştirme işlemleri gerçekleştirilmiş, ardından makine öğrenmesi modelleri geliştirilmiştir. Kafka ile gerçek zamanlı veri işleme altyapısı kurularak, anomali tespiti süreci etkin bir şekilde işlenmiştir.

Abstract

In this document, pre-processing and visualization processes were performed on the dataset for credit card fraud detection, and then machine learning models were developed. By establishing a real-time data processing infrastructure with Kafka, the anomaly detection process was processed effectively.

1.Giriş

Bu proje, kredi kartı dolandırıcılığı tespitine yönelik gerçek zamanlı bir anomali tespit sisteminin geliştirilmesini hedeflemektedir. Günümüzde finansal işlemlerdeki sahtecilik ve dolandırıcılık faaliyetleri büyük bir risk oluşturmakta ve bu durum hem bireyler hem de kuruluşlar için önemli maddi kayıplara yol açmaktadır. Bu bağlamda, anomali tespiti için makine öğrenmesi tekniklerinden yararlanılarak daha etkili ve doğru sonuçlar elde etmek amaçlanmıştır. Projede, Kafka ile entegre edilerek gerçek zamanlı veri işleme sağlanmış ve yüksek performanslı bir sistem tasarlanmıştır.

Proje kapsamında, veri ön işleme, görselleştirme, model geliştirme ve gerçek zamanlı analiz adımları titizlikle uygulanmıştır. Kullanılan veri seti, anomalileri tespit etmek için yeterli çeşitliliğe sahip olup, işlemler sırasında eksik veri analizi, veri normalizasyonu ve aykırı değer işleme gibi ön işleme teknikleri kullanılmıştır. Anomali tespiti amacıyla hem geleneksel makine öğrenmesi algoritması (Random Forest) değerlendirilmiştir.Kafka teknolojileriyle oluşturulan altyapı sayesinde, sistem anlık veri akışlarını analiz ederek tespit edilen anomalileri hızlı bir şekilde rapor edebilmektedir. Bu özellik, özellikle büyük veri analitiği gerektiren uygulamalarda büyük bir avantaj sağlamaktadır.

2. Veri Seti Seçimi ve Ön İşleme

Kaggle'dan alınan kredi kartı dolandırıcılığı veri seti üzerinde kapsamlı bir ön işleme süreci gerçekleştirilmiştir. Veri seti, finansal işlemlerden oluşmakta ve işlemleri dolandırıcılık (anormal) ya da normal olarak etiketlemektedir. Dolandırıcılık vakalarının sayısı normal işlemlere kıyasla oldukça düşüktür. Bu dengesiz yapısı nedeniyle, modelin anormal verileri öğrenmesi ve doğru şekilde sınıflandırması zorlaşmaktadır.

Bu sorunu aşmak için veri seti üzerinde random yöntemleriyle veri çoğaltma işlemi uygulanmıştır. Bu işlem, dolandırıcılık sınıfına ait örneklerin sayısını artırmayı ve veri setini daha dengeli bir hale getirmeyi amaçlamaktadır. Bu süreçte, rastgele örnekleme ve veri manipülasyonu teknikleri kullanılarak yeni örnekler türetilmiştir. Bu yaklaşım, modelin her iki sınıfı da daha iyi öğrenmesine olanak sağlamış ve sınıflandırma performansını artırmıştır.

Kodda ilk adım olarak, veri seti pandas kütüphanesi kullanılarak cleaned_fraud_data.csv dosyasından yüklenmiştir. Pandas, veri manipülasyonu ve analizi için yaygın olarak kullanılan güçlü bir Python kütüphanesidir. Veri yükleme işlemiyle, analiz sürecine uygun bir veri çerçevesi oluşturulmuştur.

Veri ön işleme aşamasında, ilk olarak analize doğrudan katkısı olmayan Time sütunu veri setinden çıkarılmıştır. Bu sütunun kaldırılması, modelin gereksiz bilgilerle karmaşık hale gelmesini önlemeyi amaçlamıştır. Daha sonra, işlemler sırasında ödemeleri temsil eden Amount sütunu logaritmik dönüşümle dönüştürülmüştür. Bu dönüşüm, büyük değişimlere sahip olan miktar verilerini daha dengeli bir dağılıma getirmek için kullanılmıştır. NumPy kütüphanesinin sağladığı log1p fonksiyonu, sıfır değerleriyle uyumlu bir şekilde logaritmik dönüşüm yaparak hem dağılımı düzenler hem de aşırı değerlerin etkisini azaltır. Log dönüşüm sonrası, orijinal Amount sütunu kaldırılmış ve yeni log_amount sütunu analiz için kullanıma hazır hale getirilmiştir.

Veri, hedef değişken olan Class (anormal ya da normal işlem) ve özellik sütunları (X) olarak ayrılmıştır. Hedef değişken, makine öğrenmesi modellerinin öğrenmesi gereken sınıfları temsil etmektedir. Özellik sütunları ise modelin eğitimi için kullanılan bağımsız değişkenleri oluşturur. Bu adım, veri setini model eğitimi için organize etmenin temel bir parçasıdır.

Özellik sütunları, MinMaxScaler kullanılarak ölçeklendirilmiştir. MinMaxScaler, veri değerlerini 0 ile 1 arasında yeniden ölçeklendirerek her bir değişkenin aynı ölçekle çalışmasını sağlar. Farklı ölçeklerdeki değişkenler, makine öğrenmesi modellerinin performansını olumsuz etkileyebilir. Bu nedenle, veri ölçeklendirme işlemi modellerin daha etkili ve doğru çalışması için kritik bir öneme sahiptir.

Ölçeklendirme işleminden sonra, yeniden düzenlenmiş özellik sütunları ve hedef değişken birleştirilerek yeni bir veri çerçevesi oluşturulmuştur. Veri setinin son hali, standardized_data.csv dosyasına kaydedilmiştir. Bu adım, veri setinin daha sonraki analizlerde ve model eğitiminde yeniden kullanılabilir olmasını sağlar. Ayrıca, veri setinin boyutu ve ilk birkaç satırı ekrana yazdırılarak yapılan işlemlerin doğruluğu ve veri yapısının uygunluğu kontrol edilmiştir.

Sonuç olarak, bu süreç, Kaggle'dan alınan kredi kartı dolandırıcılığı veri setini analiz edilebilir, dengeli ve ölçeklendirilmiş bir hale getirmiştir. Bu ön işleme adımları, makine öğrenmesi modellerinin performansını artırmaya yönelik kritik bir altyapı sağlamaktadır. Pandas, NumPy ve scikit-learn gibi güçlü kütüphaneler sayesinde, veri manipülasyonu, dönüşümü ve ölçeklendirilmesi etkili bir şekilde gerçekleştirilmiştir.

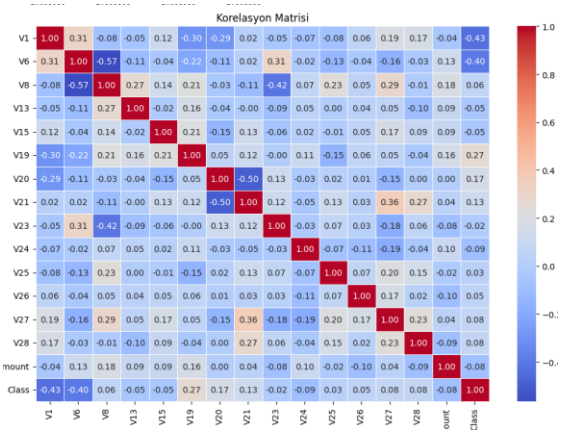
3. Veri Görselleştirme

Kredi kartı dolandırıcılığını tespit etmek amacıyla kullanılan 564.897 işlemi içeren bir veri seti üzerinde gerçekleştirilen analiz sürecini kapsamaktadır. Veri setindeki işlemler, "normal" ve "dolandırıcılık" olarak sınıflandırılmıştır.

Veri setinin genel yapısını anlamak için önce ilk 5 satır incelenmiş ve sütunlar hakkında bilgi alınmıştır. Tüm sütunların uygun veri tipinde olduğu ve eksik veri bulunmadığı tespit edilmiştir. İstatistiksel özet, sütunların ortalama, standart sapma ve minimum-maksimum değerlerini ortaya koyarak özelliklerin veri dağılımı hakkında ipuçları sunmuştur. Veri setindeki bazı sütunların geniş bir değer aralığına sahip olduğu, bu nedenle ölçeklendirme işlemlerinin kritik olduğu belirlenmiştir.

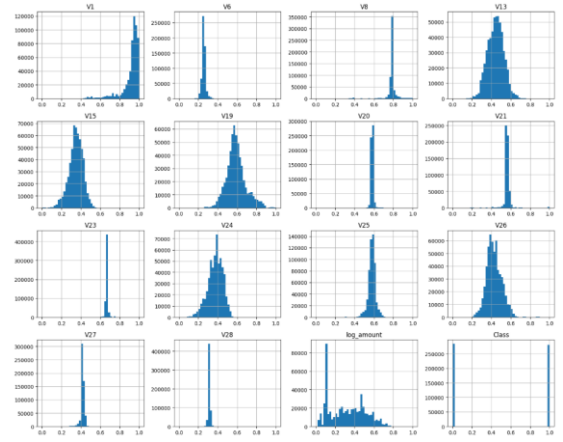
```
Veri Seti Bilgisi:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 564807 entries, 0 to 564806
Data columns (total 16 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   V1          564807 non-null  float64
 1   V6          564807 non-null  float64
 2   V8          564807 non-null  float64
 3   V13         564807 non-null  float64
 4   V15         564807 non-null  float64
 5   V19         564807 non-null  float64
 6   V20         564807 non-null  float64
 7   V21         564807 non-null  float64
 8   V23         564807 non-null  float64
 9   V24         564807 non-null  float64
 10  V25         564807 non-null  float64
 11  V26         564807 non-null  float64
 12  V27         564807 non-null  float64
 13  V28         564807 non-null  float64
 14  log_amount  564807 non-null  float64
 15  Class       564807 non-null  int64
dtypes: float64(15), int64(1)
```

Veri setindeki değişkenler arasındaki ilişkileri görmek için korelasyon analizi yapılmıştır. Korelasyon matrisi, değişkenler arasında doğrusal ilişkilerin derecesini göstermiştir. Seaborn kütüphanesi ile oluşturulan sıcaklık haritası (heatmap), bazı sütunlar arasında yüksek korelasyon olduğunu açıkça göstermiştir. Bu tür yüksek korelasyonlar, modelleme sürecinde multikolinearite sorunlarına neden olabileceği için dikkate alınmalıdır. Özellikle hedef değişkenle (Class) daha yüksek korelasyona sahip özellikler, dolandırıcılık işlemlerinin tespitinde daha önemli bir rol oynar.

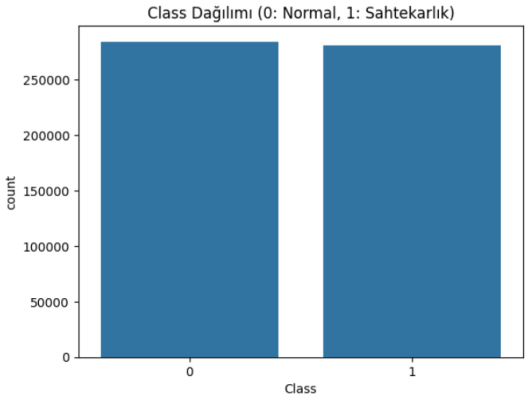


Korelasyon matrisi, veri setindeki değişkenler arasındaki doğrusal ilişkileri göstermektedir. Class (hedef değişken) ile özellikle V1 (-0.43), V6 (-0.40) ve V19 (0.27) gibi değişkenler arasında anlamlı korelasyonlar bulunmaktadır. Bu değişkenler dolandırıcılık tespitinde önemli sinyaller sağlayabilir. Ancak, V6 ve V8 (-0.57) gibi bazı değişkenler arasında yüksek korelasyon görülmektedir, bu da multikolineariteye yol açabilir

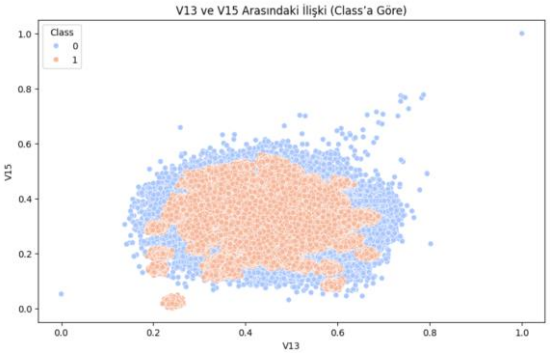
Histogramlar, her bir özelliğin değerlerinin nasıl dağıldığını görselleştirmiştir. Çoğu özelliğin normal dağılıma yakın olduğu gözlemlenirken, bazı sütunların sağa çarpık (skewed) bir dağılım gösterdiği belirlenmiştir.



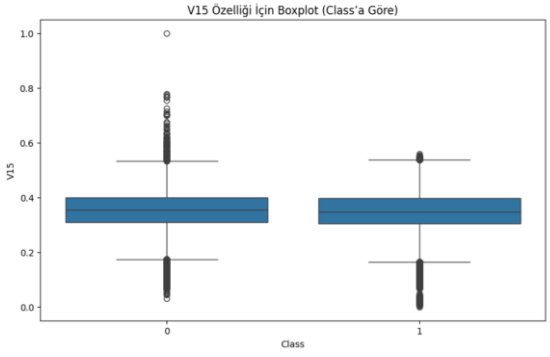
"Class" değişkeninin dağılımını görselleştiren bu grafik, veri setindeki "Normal" (0) ve "Sahtekarlık" (1) sınıflarının sayısal dağılımını gösteriyor. Grafik, her iki sınıfın da yakın sayılarda olduğunu ortaya koyuyor. Bu durum, modelin her iki sınıf için de dengeli bir şekilde eğitilmesi gerektiğini ve veri setindeki her iki sınıfın da model tarafından doğru şekilde sınıflandırılması için yeterli örneğe sahip olunduğunu gösterir. Bu tür bir dağılım, modelin her iki sınıf arasında dengesizliğe bağlı hatalar yapma olasılığını düşük tutar.



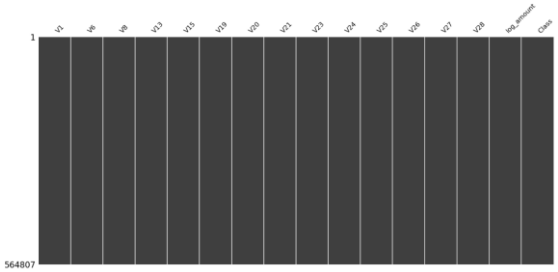
Veri setindeki özellikler arasındaki ilişkiler, scatter plot yardımıyla incelenmiştir. Örneğin, V13 ve V15 özellikleri arasındaki ilişki hedef değişkene göre görselleştirilmiş ve dolandırıcılık işlemlerinin belirli bir bölgede daha yoğunlaştığı görülmüştür. Bu tür görselleştirmeler, dolandırıcılığı etkileyen önemli özelliklerin tespit edilmesine olanak tanımaktadır.



Ayrıca, boxplot ile V15 özelliği hedef değişkene göre analiz edilmiştir.



Eksik veri analizi için missingno kütüphanesi kullanılmış ve eksik değerlerin olmadığı doğrulanmıştır. Bu durum, veri setinin temiz bir yapıya sahip olduğunu ve ek veri temizleme işlemlerine gerek olmadığını göstermektedir.



Bu analiz, kredi kartı dolandırıcılığı tespiti için kullanılan veri setinin özelliklerini keşfetmek ve görselleştirmek adına değerli bilgiler sağlamıştır. Analiz sonuçları, hangi değişkenlerin dolandırıcılık işlemleri üzerinde önemli bir etkisi olduğunu ortaya koymuş ve bu bilgiyi, modelin başarısını artıracak şekilde kullanılabilir hale getirmiştir.

Görselleştirme araçları, modelin başarısını artırmak adına doğru özelliklerin ve sınıflandırma yöntemlerinin belirlenmesine yardımcı olmuştur.

4. Makine Öğrenmesi Modelleri

Kredi kartı sahtekarlığı tespiti için üç farklı makine öğrenmesi modelinin performanslarını karşılaştırmak amaçlanmıştır. Kullanıcılar, genellikle kredi kartı işlemleriyle ilgili anormal davranışları tespit etmek için makine öğrenmesi algoritmalarını kullanırlar. Sahtekarlık tespiti, dolandırıcılık olaylarını en aza indirmek için büyük önem taşır. Sahtekarlık tespiti için Logistic Regression (Lojistik Regresyon), Random Forest (Rastgele Orman) ve Decision Tree (Karar Ağaçları) modellerini kullanarak bu modellerin doğruluk ve diğer metriklerini karşılaştırılmıştır.

Veri Seti: Kredi kartı işlem verilerinden oluşmaktadır ve her bir işlem, bir dizi özellik içermektedir. 'Class' adlı hedef değişkeni kullanılmıştır. Bu değişken, işlemin sahte (1.0) veya gerçek (0.0) olduğunu belirtir.

Bağımsız Değişkenler (X): 'Class' dışında kalan tüm özellikler veri setinin bağımsız değişkenlerini oluşturur.

Bağımlı Değişken (y): 'Class' sütunu, hedef değişkeni temsil eder ve işlemin gerçek veya sahte olduğunu belirtir.

Veri seti, %80 eğitim ve %20 test seti olarak ikiye ayrılmıştır. Eğitim seti, modelin öğrenmesi için, test seti ise modelin doğruluğunu değerlendirmek için kullanılacaktır.

Veri, daha iyi sonuçlar elde etmek amacıyla standardize edilmiştir. Bu işlemde, her bir özellik için ortalama değeri 0 ve standart sapmayı 1 olacak şekilde yeniden ölçeklendirme yapılmıştır. Bu işlem, özellikle makine öğrenmesi algoritmalarında, özellikle mesafeye dayalı yöntemlerin (örneğin lojistik regresyon) başarısını artırmak için önemlidir.

4.1 Kullanılan Modeller

Logistic Regression (Lojistik Regresyon): Lojistik regresyon, doğrusal sınıflandırma problemlerinde kullanılan temel bir makine öğrenmesi modelidir. Bu model, her bir veri noktasına ait özelliklerin doğrusal bir kombinasyonunu alarak, sınıflar arasında olasılık temelli bir ayrım yapar. Modelin temel avantajı, hız ve basitliktir. Ancak doğrusal olmayan ilişkilerde sınırlı performans gösterebilir.

Model Eğitimi: Lojistik regresyon, max_iter=1000 parametresiyle eğitilmiştir. Bu parametre, modelin daha stabil çalışabilmesi ve daha yüksek sayıda iterasyon yaparak en uygun parametreleri öğrenmesi için kullanılmıştır.

Sonuçlar:

- Accuracy (Doğruluk): 0.9831
- Precision (Kesinlik): 0.9839
- Recall (Duyarlılık): 0.9769
- F1 Skoru: 0.9804

Random Forest (Rastgele Orman): Random Forest, birden fazla karar ağacının birleştirildiği bir ansamble (topluluk) modelidir. Bu modelin temeli, her bir karar ağacının bağımsız olarak kararlar vermesi ve ardından bu kararların çoğunluk bazında birleştirilmesidir. Random Forest, genellikle yüksek doğruluk oranı ve düşük aşırı uyum (overfitting) eğilimi ile dikkat çeker.

Model Eğitimi: Random Forest modelinde, random_state=42 parametresi kullanılarak sonuçların tekrarlanabilirliği sağlanmıştır.

Sonuçlar:

- Accuracy (Doğruluk): 0.9971
- Precision (Kesinlik): 0.9972
- Recall (Duyarlılık): 0.9965
- F1 Skoru: 0.9969

Decision Tree (Karar Ağaçları): Karar ağaçları, verileri sınıflandırırken her bir özellik üzerinde dallanarak kararlar alır. Bu modeller oldukça yorumlanabilir olup, görsel olarak da anlaşılabilir. Ancak, aşırı uyum yapma eğilimleri olabilir, yani eğitim verisine çok iyi uyarlanmaları durumunda yeni verilerde başarısız olabilirler.

Model Eğitimi: Karar ağacı modeli de random_state=42 parametresiyle eğitilmiştir.

Sonuçlar:

- Accuracy (Doğruluk): 0.9890
- Precision (Kesinlik): 0.9891
- Recall (Duyarlılık): 0.9845
- F1 Skoru: 0.9868

Random Forest, en yüksek accuracy, precision ve recall değerlerine sahip olmuştur. Bu, modelin sahtekarlık tespiti konusunda mükemmel bir denge sağladığını göstermektedir. Yüksek recall değeri, modelin sahte işlemleri doğru bir şekilde tanımlamada çok başarılı olduğunu gösterir.

5.Kafka Yapılandırması

5.1 Kafka Producer

Bu projede, Apache Kafka kullanarak veri gönderimi yapılmıştır. Kafka, yüksek hacimli veri iletimine olanak tanıyan ve dağıtık sistemlerde verilerin etkin bir şekilde işlenmesini sağlayan bir platformdur. Kafka producer kullanarak rastgele oluşturulan veri kümelerinin bir Kafka topic'ine gönderilmesi süreci detaylandırılmıştır. Veri üretimi, anomaliler ve normal durumlar arasındaki farkları içeren bir veri seti ile yapılmış ve saniyede 833 veri gönderilmesi sağlanmıştır.

Kafka Producer Yapılandırması: Öncelikle, Kafka producer'ı confluent_kafka kütüphanesi ile yapılandırılmıştır. Producer, Kafka broker'a bağlanarak verileri gönderecek olan bileşendir. Bağlantı için kullanılan konfigürasyon parametresi bootstrap.servers, Kafka'nın çalıştığı sunucu adresini belirtir.

Aşağıda kullanılan konfigürasyon örneği verilmiştir:

```
conf = {
    'bootstrap.servers': 'localhost:9092',  # Kafka'nın çalıştığı adres
}
```

Burada, Kafka sunucusunun localhost:9092 adresinde çalıştığı varsayılmaktadır. Bu parametre, Kafka broker'a bağlanarak veri gönderimi yapılmasını sağlar.

Veri Üretimi: Veri üretme işlemi, rastgele değerler üreten bir fonksiyon olan generate_data() fonksiyonu ile yapılmaktadır. Bu fonksiyon, her bir veri örneği için aşağıdaki alanları içeren bir sözlük döndürür:

- *V1, V6, V8, V13, V15, V19, V20, V21, V23, V24, V25, V26, V27, V28:* Veri setindeki özellikler (rastgele sayılar ile oluşturulur).
- *year, month, day, hour, weekday:* Zaman ile ilgili kategorik özellikler (0 veya 1 olarak rastgele seçilir).
- *log_amount:* Herhangi bir işlem miktarını temsil eden rastgele bir sayı.
- *Class*: Verinin normal (0) veya anomali (1) olduğunu belirten etiket.

Veri Gönderme ve Kafka Producer Kullanımı: Veri gönderimi, producer.produce() fonksiyonu ile gerçekleştirilir. Bu fonksiyon, belirli bir topic'e veri gönderir. Burada, input_topic adında bir topic kullanılmıştır. Gönderilen her mesajın başarıyla iletilip iletilmediği, delivery_report fonksiyonu aracılığıyla kontrol edilir. Bu fonksiyon, her mesaj gönderildikçe callback olarak çağrılır ve gönderim sonucunu kullanıcıya bildirir. Eğer mesaj başarıyla iletilirse, mesajın içeriği ekrana yazdırılır.

Veri gönderimi, saniyede 833 veri gönderilmesi hedeflenerek yapılandırılmıştır. Bu hedef doğrultusunda, her veri gönderimi arasında belirli bir süre beklemek için

time.sleep(interval) fonksiyonu kullanılmıştır. Gönderilen veri sayısı target_count değişkeni ile belirlenmiştir ve toplamda 100,000 veri gönderilmesi planlanmıştır.

Veri gönderim döngüsü şu şekilde işlemektedir:

- Her döngüde generate_data() fonksiyonu ile yeni bir veri örneği oluşturulur.
- producer.produce() fonksiyonu ile bu veri, Kafka'nın input_topic adlı topic'ine gönderilir.
- producer.poll(0) fonksiyonu, producer'ın buffer'ındaki mesajların gönderilmesini sağlar.
- Döngü, belirlenen hedef veri sayısına ulaşana kadar devam eder.

Bu sistem, Kafka kullanarak yüksek hızda veri gönderimini verimli bir şekilde gerçekleştirir. Gerçek zamanlı veri akışı sağlamak için kullanılan bu yapı, sahte (anomali) ve normal verilerin etkili bir şekilde iletilmesini mümkün kılar. Kafka'nın düşük gecikme süresi ve yüksek veri işleme kapasitesi, büyük veri akışlarını yönetmek ve analiz etmek için güçlü bir altyapı sunar.

Kafka'nın güçlü altyapısı sayesinde, veri gönderimi anında gerçekleştirilir ve bu sayede gerçek zamanlı analizler yapılabilir.

5.2 Kafka Consumer

Kafka producer bileşeni aracılığıyla üretilen verileri alıp, yüklenen Random Forest makine öğrenmesi modelini kullanarak tahminler yapmaktadır. Tahminlerin ardından veriler, sonuçlarına göre iki farklı topic'e (anomaliler için "anomaly_topic" ve normal veriler için "normal_topic") yönlendirilmektedir.

İki ana Kafka bileşeni olan Producer ve Consumer kullanılmıştır. Her iki bileşenin de Kafka ile iletişimi sağlamak için gerekli konfigürasyonları yapılmıştır.

Consumer Config:

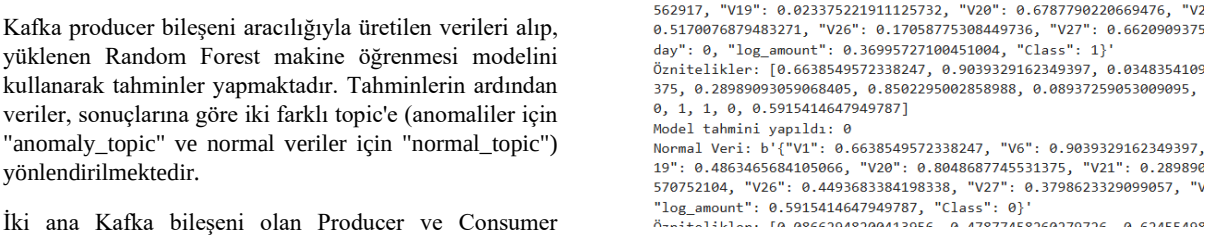
- *bootstrap.servers:* Kafka sunucusunun adresi olarak localhost:9092 belirlenmiştir. Bu, Kafka broker'ının çalıştığı sunucuya işaret eder.
- *group.id*: Consumer group ID olarak consumer_group belirlenmiştir. Bu, Kafka'nın hangi consumer grubuna ait olduğunu belirtir ve aynı topic'e abone olan birden fazla consumer'ın koordinasyonunu sağlar.
- *auto.offset.reset:* Bu parametre, verilerin hangi noktadan alınacağını belirler. earliest değeri, Kafka'nın topic'teki en eski verilerden başlayarak veri almasını sağlar.

Producer Config:

- *bootstrap.servers:* Producer, Kafka'ya veri göndermek için aynı Kafka sunucusuna (localhost:9092) bağlanır.

Modelin Yüklenmesi: Anomali tespiti için Random Forest makine öğrenmesi modeli kullanılmıştır. Model, daha önce eğitilmiş ve random_forest_model.pkl adıyla kaydedilmiştir. Bu model, consumer tarafından veri alındıktan sonra yüklenir ve veriler üzerinde tahmin yapılır. Modelin tahmin ettiği sonuçlar, verinin normal mi yoksa anomali mi olduğunu belirler.

Verinin Alınması ve İşlenmesi: Sistem, Kafka kullanarak gerçek zamanlı veri iletimi yapmaktadır ve bu süreç, verilerin Kafka topic'lerinden alınarak işlenmesiyle başlar. Consumer, Kafka'dan veri almak için input_topic adlı topic'e abone olur. Bu, sistemin sürekli olarak bu topic'teki yeni mesajları almasına olanak tanır. Her bir mesaj, JSON formatında olup içinde çok sayıda öznitelik barındırır. Bu öznitelikler, modelin doğru tahminler yapabilmesi için gerekli olan verilerdir. Öznitelikler arasında V1, V6, V8, V13, V15, V19, V20, V21, V23, V24, V25, V26, V27, V28 gibi sürekli değerler yer alırken, zamanla ilgili olan year, month, day, hour, weekday gibi özellikler de bulunur. Ayrıca, log_amount değeri ve sınıflandırma için kullanılan Class (0: Normal, 1: Anomali) öznitelikleri de veride yer almaktadır. Veriler alındıktan sonra, JSON formatındaki mesajlar işlenir ve her bir öznitelik, modelin tahmin fonksiyonuna uygun bir şekilde hazırlanır. JSON verisi, Python dilindeki dict formatına dönüştürülür, böylece her bir özniteliğin kolayca erişilebilmesi sağlanır. Bu aşamadan sonra, belirli öznitelikler çıkarılır ve modelin giriş verisi olarak kullanılır.



Tahmin Yapılması: Veriler, modelin tahmin fonksiyonuna gönderildikten sonra, model verilen özelliklere dayanarak bir tahmin yapar. Bu tahmin sonucu, verinin normal veya anomalik olduğunu belirler. Modelin tahmin ettiği sonuç iki olasılıktan birine dayanır: Eğer model sonucu 0 (normal) olarak tahmin ederse, veri normal olarak kabul edilir; eğer model sonucu 1 (anomalik) olarak tahmin ederse, veri anomali olarak sınıflandırılır. Bu aşamadan sonra, tahmin edilen sonuca göre bir işlem yapılır. Eğer modelin tahmin ettiği değer 1 ise, yani verinin anomali olduğu tespit edilirse, bu veri anomaly_topic adlı Kafka topic'ine gönderilir. Aksi takdirde, veri normal olarak kabul edilip normal_topic adlı

Kafka topic'ine iletilir. Bu sınıflandırma işlemi, sistemin veriyi doğru bir şekilde kategorize etmesini sağlar ve her iki kategorideki veriler, farklı topic'lerde saklanarak ilerleyen aşamalarda daha detaylı bir şekilde analiz edilebilir.

Mesajların Yönlendirilmesi :Tahmin sonrasında, Producer kullanılarak mesajlar uygun topic'lere iletilir. Mesajın içeriği, tahmin edilen sonuca göre belirlenir. Anomali tespit edilen veriler "anomaly_topic" adlı topic'e gönderilirken, normal veriler "normal_topic" adlı topic'e gönderilir.

Sonsuz Döngü ve Mesajların İşlenmesi: While True döngüsü içerisinde sürekli olarak Kafka'dan veri almaya devam eder. Döngü, her seferinde consumer.poll(timeout=1.0) fonksiyonu ile Kafka'dan bir mesaj almaya çalışır. Eğer bir mesaj alınmazsa, işlem 1 saniye boyunca bekler. Mesaj alındığında, veri işlenir ve model tahminini yapar. Bu işlem, sürekli olarak devam eder ve Kafka'dan alınan veriler sürekli olarak işlenir ve yönlendirilir.

Kafka üzerinde çalışan bir producer ve consumer ile gerçek zamanlı anomali tespiti yapılmıştır. Sistem, Kafka'dan gelen verileri işleyerek, her veriyi modelle tahmin eder ve tahmin sonuçlarına göre uygun topic'e iletir. Bu yapı, yüksek hızda veri iletimi ve anomali tespiti yapabilen bir sistem için güçlü bir altyapı sunmaktadır.

6. Proje Github Linkleri

211307061 Helen Hacer Uzunçayır–

https://github.com/Helen205/big_data

211307048 Buse Yener -

https://github.com/buseyener/KrediKartDolandiriciligindaAnomaliTespiti

Kaynakça

1. https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud
2. https://kafka.apache.org/documentation/
3. https://spark.apache.org/docs/latest/
4. https://www.veribilimiokulu.com/buyuk-veri-on-isleme-makale-notlari/
5. https://www.python.org/
6. https://scikit-learn.org/1.5/modules/generated/sklearn.ensemble.RandomForestClassifier.html
7. https://spark.apache.org/docs/latest/streaming-programming-guide.html
8. https://www.scala-lang.org/