Samsun Üniversitesi
Yazılım Mühendisliği Bölümü

VERİ MADENCİLİĞİ (DATA MİNİNG)

Dr. Öğr. Üyesi Alper Talha Karadeniz

2025-2026

Hafta 10

Büyük Veri ve Gerçek Zamanlı Madencilik

Samsun Üniversitesi
Yazılım Mühendisliği Bölümü

01

Apache Hadoop, Spark

02

Stream veri işleme kavramı

03

IoT ve sensör verilerinde madencilik

04

Uygulama

Büyük Veri Nedir?

Büyük veri (Big Data), geleneksel yöntemlerle işlenmesi zor olan, yüksek hacimli (Volume), hızlı (Velocity), çeşitli (Variety) ve doğrulanabilir (Veracity) veriler bütünüdür.

Özellikler (5V):

- Volume (Veri hacmi)
- Velocity (Üretilme ve işlenme hızı)
- Variety (Farklı veri tipleri: metin, görüntü, sensör verisi vb.)
- Veracity (Güvenilirlik, doğruluk)
- Value (Değer – veriden çıkarılacak fayda)

Kullanım Alanları: Sağlık, finans, sosyal medya, IoT, e-ticaret, savunma vb.

Gerçek Zamanlı Veri Madenciliği Nedir?

Verilerin anında işlenip analiz edilmesi.

Neden Önemli?

- Dolandırıcılık tespiti (bankacılıkta)
- Anlık öneri sistemleri (Netflix, YouTube, e-ticaret siteleri)
- IoT sensör verilerinin takibi
- Siber güvenlik saldırılarını anında fark etme

Zorluklar:

- Yüksek hızda akan verinin saklanması ve işlenmesi
- Donanım ve yazılım altyapısının güçlü olması gerekliliği

Apache Hadoop

Büyük veri işleme için geliştirilmiş açık kaynaklı bir yazılımdır. Çok büyük boyutlardaki verileri farklı bilgisayarlara dağıtarak depolayabilir ve işleyebilir. Bu sayede tek bir bilgisayarın kapasitesini aşan veri kümeleri bile kolayca yönetilebilir. Hatalara karşı dayanıklı olması ve ucuz donanımlarla çalışabilmesi Hadoop'u büyük veri alanında önemli bir araç haline getirmiştir. Ancak, daha çok toplu veri işleme odaklı olduğu için gerçek zamanlı veri analizinde sınırlı kalmaktadır.Temel bileşenleri yandaki gibidir.

Samsun Üniversitesi
Yazılım Mühendisliği Bölümü

HDFS (Hadoop Distributed File System) →

- Veriyi farklı düğümlere (node) böler.
- Her veri çoğaltılarak saklanır. Böylece düğüm çökse bile veri kaybolmaz.
- NameNode: Ana düğüm
- DataNode: Asıl veriyi saklayan düğümler.

MapReduce →

- Hadoop'un veri işleme modeli.
- Map: Büyük veriyi parçalara ayırır ve işleme başlatır.
- Reduce: İşlenen parçaları birleştirip sonuç çıkarır.

YARN (Yet Another Resource Negotiator) →

- Kaynak yönetiminden sorumlu.
- Cluster'daki (küme) CPU ve RAM kullanımını düzenler.

Hadoop Common →

- Diğer tüm bileşenler için gerekli yardımcı kütüphaneler.

Apache Spark

Apache Spark ise Hadoop'tan sonra geliştirilmiş ve daha hızlı çalışan bir büyük veri işleme aracıdır. Hadoop'un aksine verileri bellek üzerinde işlediği için çok daha kısa sürede sonuç üretebilir. Spark, hem geçmiş verilerin analizi hem de anlık verilerin işlenmesi için kullanılabilir. Ayrıca makine öğrenmesi ve veri analizi gibi ileri seviye uygulamalarda da tercih edilmektedir. Yüksek performans sağlamasına rağmen daha fazla bellek ihtiyacı duymaktadır.Temel bileşenleri yandaki gibidir.

Samsun Üniversitesi
Yazılım Mühendisliği Bölümü

Spark Core →
- Temel hesaplama motoru.
- Görev dağılımı, bellek yönetimi, hata toleransı.

Spark SQL →
- SQL benzeri sorgularla büyük veriyi işleme
- Yapısal veriler için DataFrame API sunar.

Spark Streaming →
- Gerçek zamanlı veri işleme (ör. Twitter verisi, IoT sensörleri).
- Veri akışlarını mini-batch'ler halinde işler.

MLlib (Machine Learning Library) →
- Makine öğrenmesi algoritmaları içerir (SVM, karar ağaçları, kümeleme vb.).
- Büyük veri üzerinde ML uygulamaları için hazırdır.

GraphX →
- Grafik tabanlı hesaplamalar (ör. sosyal ağ analizi).

Apache Hadoop & Apache Spark

Avantajları:

- Çok büyük veri kümelerini işleyebilir.
- Hatalara dayanıklıdır (veri kopyalama sayesinde).
- Ucuz donanımlarla (commodity hardware) çalışabilir.

Dezavantajları:

- Disk tabanlı çalışır → yavaştır.
- Gerçek zamanlı analiz için uygun değildir (batch processing).
- Spark kadar esnek değildir.

Avantajları:

- Bellek içi (in-memory) çalıştığı için çok hızlıdır.
- Batch + Streaming işleyebilir.
- Python, Java, Scala, R desteği vardır.
- Makine öğrenmesi ve grafik analizi için zengin kütüphaneler içerir.

Dezavantajları:

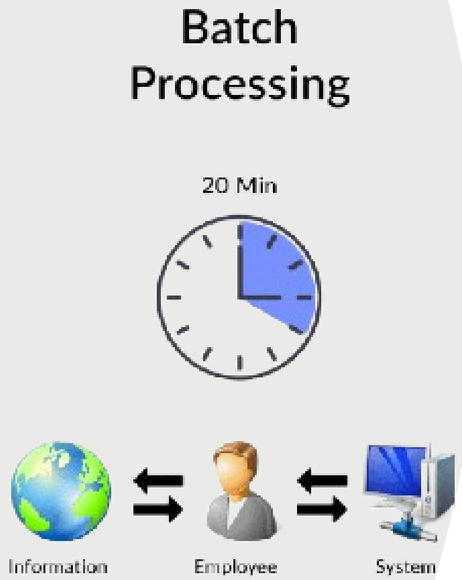
- Daha fazla RAM gerektirir (yüksek donanım maliyeti).
- Küçük veri kümeleri için aşırı olabilir.

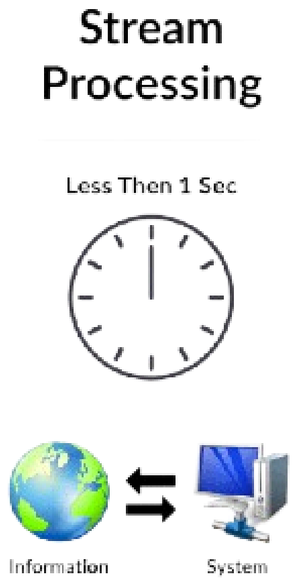
Apache Hadoop & Apache Spark

Özellik	Hadoop (MapReduce)	Spark
İşleme Modeli	Disk tabanlı (yavaş)	Bellek tabanlı (çok hızlı)
Veri Türü	Batch (toplu işleme)	Batch + Streaming
Kütüphaneler	Sınırlı	MLlib, Spark SQL, GraphX
Donanım İhtiyacı	Düşük	Yüksek RAM gerektirir
Hata Toleransı	Yüksek	Yüksek
Kullanım Alanı	Depolama, toplu raporlama	Gerçek zamanlı analiz, ML, IoT

Stream Veri İşleme (Akış Verisi İşleme)

Stream veri işleme (akış verisi işleme), verilerin toplu halde değil, anlık olarak işlendiği bir yöntemdir. Yani veriler bir yerde depolanıp sonra analiz edilmez; sistemden geçerken eş zamanlı olarak analiz edilir. Bu yaklaşım özellikle sosyal medya akışları, sensör verileri, finansal işlemler veya IoT cihazlarından gelen sürekli veriler gibi gerçek zamanlı uygulamalarda kullanılır. Stream işleme sayesinde olaylar anında yakalanabilir, hızlı kararlar alınabilir ve sonuçlar gerçek zamanlı olarak kullanıcıya sunulabilir.





Stream Veri İşleme

Sosyal Medya (Twitter, Instagram, YouTube)

Stream işleme ile tweet atıldığı anda işlenir:

- Trend başlıkları anlık çıkarma.
- Spam ya da sahte hesap tespiti.
- Canlı duygu analizi (örneğin bir etkinlik sırasında insanların yorumları).

IoT ve Sensör Verileri

- Trafik ışıkları, sensörlerden gelen akış verisiyle anlık yönetilir.
- Fabrikadaki makinelerden gelen titreşim/sıcaklık verisi anlık analiz edilerek arıza çıkmadan bakım yapılır.

E-ticaret (Hepsiburada, Amazon, Trendyol)

Kullanıcı davranışlarını anlık izleyebilir:

- Müşteri bir ürüne bakıyorsa hemen benzer ürünler önerilir.
- Ödeme sırasında şüpheli hareket olursa güvenlik sistemi devreye girer.

Finans Sektörü

Bankalar ve borsalar için stream işleme çok önemlidir:

- Şüpheli para transferleri anlık tespit edilir.
- Borsada saniyelik fiyat değişimleri anında analiz edilerek otomatik alım-satım yapılır.

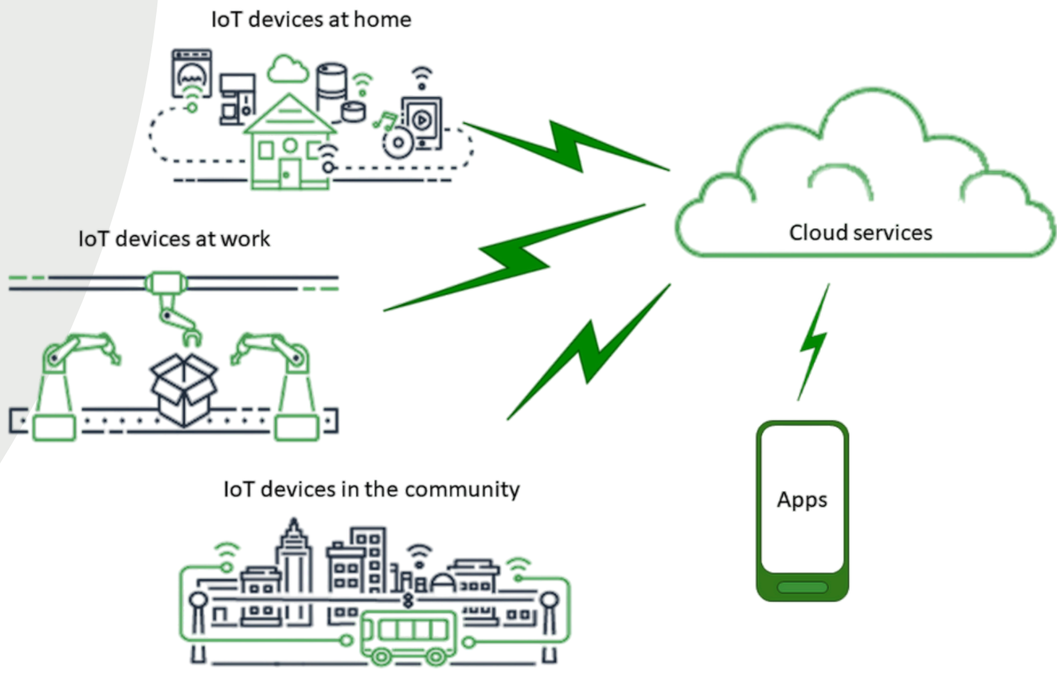
IoT ve Sensör Verilerinde Madencilik Nedir?

IoT (Internet of Things / Nesnelerin İnterneti), fiziksel cihazların (akıllı saatler, sensörler, akıllı ev cihazları, otomobiller, endüstriyel makineler vb.) internet üzerinden birbirine bağlanarak veri üretip paylaşmasını sağlayan bir teknolojidir. Bu cihazlar genellikle çevreden sürekli sensör verisi toplar: sıcaklık, nem, basınç, hız, enerji tüketimi, konum bilgisi, sağlık ölçümleri gibi.

Bu sensörlerden gelen veriler genellikle:

- **Büyük hacimli (High Volume):** Milyonlarca sensör saniyeler içinde veri üretir.
- **Hızlı (High Velocity):** Veriler sürekli ve gerçek zamanlı olarak akar.
- **Çeşitli (High Variety):** Sayısal ölçümler, log kayıtları, GPS konumları gibi farklı veri türleri içerir.

Bu nedenle, IoT sistemlerinde ham veriyi anlamlı bilgiye dönüştürmek için veri madenciliği teknikleri kullanılır.



IoT ve Sensör Verilerinde Madenciliğin Temel Amaçları

Anlamlı Bilgi Çıkarmak:

- Sensörlerden gelen ham veri çoğu zaman çok büyük, düzensiz ve anlaşılması zor olur. Veri madenciliği sayesinde bu ham veri, kullanışlı bilgiye ve örüntülere dönüştürülür

Anormallik Tespiti (Anomaly Detection):

Sistemlerdeki olağan dışı durumları fark etmek kritik bir hedeftir. Örnekler:

- Bir sağlık sensörünün beklenmedik kalp atışı ölçümü → acil uyarı.

Bu sayede sorunlar önceden fark edilip müdahale edilebilir.

Tahmin ve Öngörü (Prediction & Forecasting):

- Gelecekteki olayları önceden tahmin etmek, IoT sistemlerinin en önemli amaçlarındandır. Örneğin bir enerji tüketim sensörü verileri analiz edilerek, gelecek günlerdeki enerji talebi tahmin edilebilir.

Gerçek Zamanlı Karar Alma:

- Sensör verileri çoğu zaman anlık aksiyon gerektirir. Örneğin akıllı trafikte sensör verileri ışık sürelerini anında değiştirmek için analiz edilir.

IoT ve Sensör Verilerinde Madencilikte Kullanılan Yöntemler

Ön İşleme (Data Preprocessing):

Ham sensör verileri çoğu zaman eksik, hatalı veya gürültülü olabilir. Bu yüzden analizden önce veri temizlenir ve standardize edilir.

- **Eksik verilerin tamamlanması:** Boş değerler uygun tahminlerle doldurulur.
- **Gürültü azaltma:** Sensör hatalarından kaynaklanan anormal ölçümler düzeltilir.
- **Normalizasyon / Standardizasyon:** Veriler aynı ölçeğe getirilir.

Sınıflandırma (Classification):

Veriyi önceden tanımlanmış kategorilere ayırma yöntemidir.

Örnekler:

- Fabrika makinelerini “normal” veya “arızalı” olarak sınıflandırmak.
- Sağlık sensörlerinden gelen verileri “kritik” veya “normal” olarak etiketlemek.

IoT ve Sensör Verilerinde Madencilikte Kullanılan Yöntemler

Kümeleme (Clustering):

Benzer veri örüntülerini bir araya getirerek gruplar oluşturma yöntemidir.

Örnekler:

- Farklı kullanıcıların enerji tüketim davranışlarını gruplamak.
- Tarım sensörlerinden gelen verileri benzer iklim ve toprak koşullarına göre segmentlere ayırmak.

Tahminleme (Prediction / Forecasting):

Gelecekteki olayları öngörmek için yapılan analizdir.

Örnekler:

- Gelecek günlerdeki enerji talebini tahmin etmek.
- Bir makinenin arızalanmadan önceki olası performans düşüşünü öngörmek.

IoT ve Sensör Verilerinde Madencilikte Kullanılan Yöntemler

Anomali Tespiti (Anomaly Detection):

Normal davranıştan sapmaları bulma yöntemidir.

Örnekler:

- Sensörün beklenmedik veri göndermesi → arıza veya güvenlik riski.
- Sağlık sensörlerinden gelen anormal ölçümler → acil uyarı

Gerçek Zamanlı İşleme (Real-Time Analytics):

Sensör verileri sürekli ve hızlı aktığı için, veriler anlık analiz edilir ve sistemlere anında geri bildirim sağlanır.

Kullanılan teknolojiler: Apache Spark Streaming, Apache Kafka, Flink gibi araçlar.

Optimizasyon ve Karar Destek (Optimization & Decision Making):

Analiz edilen verilerle sistemlerin verimliliği artırılır ve daha iyi kararlar alınır.

Örnekler:

- Akıllı trafikte ışık sürelerinin optimize edilmesi.
- Akıllı tarımda sulama ve gübreleme zamanlarının belirlenmesi.

IoT ve Sensör Verilerinde Madenciliğinin Uygulama Alanları

- **Akıllı Sağlık:** Kalp ritmi, tansiyon, oksijen seviyesi ve vücut sıcaklığı gibi biyometrik verileri analiz ederek kritik durumlarda uyarı göndermek; kronik hastalıkları takip etmek ve erken teşhis sağlamak.
- **Akıllı Tarım:** Toprak nemi, sıcaklık, ışık miktarı ve pH seviyelerini analiz ederek sulama, gübreleme ve ilaçlama zamanlarını optimize etmek; mahsul verimliliğini artırmak ve su tasarrufu sağlamak.
- **Endüstri 4.0:** Fabrika makinelerinin sıcaklık, titreşim ve basınç sensörlerini izleyerek arıza oluşmadan önce bakım yapmak (predictive maintenance); üretim hattındaki performansı artırmak ve enerji kullanımını optimize etmek.
- **Akıllı Şehirler:** Trafik sensörlerinden gelen verilerle trafik yönetimini optimize etmek; akıllı sokak lambaları ve enerji sensörleri ile enerji tüketimini dengelemek; hava kalitesi ve su kaynaklarını izleyerek şehir yönetimine destek sağlamak.
- **Enerji Yönetimi:** Akıllı sayaçlardan ve enerji sensörlerinden veri toplayarak enerji tüketimini izlemek, talep tahmini yapmak ve tasarruf modelleri geliştirmek; yenilenebilir enerji kaynaklarının kullanımını optimize etmek.
- **Ulaşım ve Lojistik:** Araç sensörleri ve GPS verilerini analiz ederek rotaları optimize etmek; filo takibi ve yakıt tüketimini azaltmak; teslimat sürelerini kısaltmak ve trafik yoğunluğunu azaltmak.

Samsun Üniversitesi
Yazılım Mühendisliği Bölümü

Tanıtım

Google Colab, Python tabanlı projeleri bulut ortamında çalıştırmaya olanak sağlayan ücretsiz bir platformdur. Özellikle makine öğrenmesi ve veri analizi projeleri için GPU/TPU desteği sunmasıyla tercih edilir.
PySpark, Apache Spark'ın Python API'sidir ve büyük veri işleme, dağıtık hesaplama ve gerçek zamanlı analizler için kullanılır. Spark, veriyi paralel olarak işler ve büyük veri kümeleri üzerinde hızlı analiz yapılmasını sağlar. PySpark ile veri madenciliği, makine öğrenmesi ve akış verisi analizi kolaylaşır.
Google Colab üzerinde PySpark kullanarak büyük veri setlerini işlemek, Spark'ın dağıtık hesaplama gücünden faydalanmayı sağlar ve lokal bilgisayarın kaynak sınırlamalarını ortadan kaldırır. Bu sayede hem eğitim amaçlı hem de prototip geliştirme projelerinde hızlı bir başlangıç yapılabilir.

UYGULAMA

Google Colab I

Kod Parçası

```
from pyspark.sql.types import StructType, StructField, StringType, FloatType

# Sensör veri örneği: sensör id, sıcaklık, nem
veri = [
    ("S1", 23.5, 45.2),
    ("S2", 25.1, 50.3),
    ("S3", 22.8, 48.0),
    ("S1", 24.0, 46.5),
    ("S2", 26.2, 51.0)
]

sablon = StructType([
    StructField("sensor_id", StringType(), True),
    StructField("sicaklik", FloatType(), True),
    StructField("nem", FloatType(), True)
])

df = spark.createDataFrame(veri, sablon)
df.show()
```

Kod Çıktısı

```
+---------+--------+----+
|sensor_id|sicaklik| nem|
+---------+--------+----+
|       S1|    23.5|45.2|
|       S2|    25.1|50.3|
|       S3|    22.8|48.0|
|       S1|    24.0|46.5|
|       S2|    26.2|51.0|
+---------+--------+----+
```

UYGULAMA

Google Colab II

Kod Parçası

```
from pyspark.sql import SparkSession
from pyspark.sql.types import StructType, StructField, StringType, FloatType
spark = SparkSession.builder.appName("Sensör Verisi Analizi").getOrCreate()

data = [
    ("S1", 23.5, 45.2),
    ("S2", 25.1, 50.3),
    ("S3", 22.8, 48.0),
    ("S1", 24.0, 46.5),
    ("S2", 26.2, 51.0)
]
schema = StructType([
    StructField("sensor_id", StringType(), True),
    StructField("sicaklik", FloatType(), True),
    StructField("nem", FloatType(), True)
])

# DataFrame oluşturma
df = spark.createDataFrame(data, schema)

# Ortalama hesaplama
ortalama_df = df.groupBy("sensor_id").avg("sicaklik", "nem")

print("📊 Her Sensörün Ortalama Sıcaklık ve Nem Değerleri:")
ortalama_df.show()
```

Kod Çıktısı

```
📊 Her Sensörün Ortalama Sıcaklık ve Nem Değerleri:
+---------+------------------+-----------------+
|sensor_id|     avg(sicaklik)|         avg(nem)|
+---------+------------------+-----------------+
|       S2| 25.65000057220459|50.64999961853027|
|       S1|             23.75|45.85000038146973|
|       S3|22.799999237060547|             48.0|
+---------+------------------+-----------------+
```

UYGULAMA

Google Colab III

Kod Parçası

```
from pyspark.sql import SparkSession
from pyspark.sql.types import StructType, StructField, StringType, FloatType
from pyspark.sql import functions as F
spark = SparkSession.builder.appName("Sensör Verisi Min-Max Analizi").getOrCreate()
veri = [
    ("S1", 23.5, 45.2),
    ("S2", 25.1, 50.3),
    ("S3", 22.8, 48.0),
    ("S1", 24.0, 46.5),
    ("S2", 26.2, 51.0)
]
sablon = StructType([
    StructField("sensor_id", StringType(), True),
    StructField("sicaklik", FloatType(), True),
    StructField("nem", FloatType(), True)
])
df = spark.createDataFrame(veri, sablon)
# Minimum ve maksimum değerleri hesaplama
min_max_df = df.groupBy("sensor_id").agg(
    F.min("sicaklik").alias("min_sicaklik"),
    F.max("sicaklik").alias("max_sicaklik"),
    F.min("nem").alias("min_nem"),
    F.max("nem").alias("max_nem")
)
print("📈 Her Sensörün Minimum ve Maksimum Değerleri:")
min_max_df.show()
```

Kod Çıktısı

```
📈 Her Sensörün Minimum ve Maksimum Değerleri:
+---------+-----------+-----------+-------+-------+
|sensor_id|min_sicaklik|max_sicaklik|min_nem|max_nem|
+---------+-----------+-----------+-------+-------+
|       S2|       25.1|       26.2|   50.3|   51.0|
|       S1|       23.5|       24.0|   45.2|   46.5|
|       S3|       22.8|       22.8|   48.0|   48.0|
+---------+-----------+-----------+-------+-------+
```

Samsun Üniversitesi
Yazılım Mühendisliği Bölümü

VERİ MADENCİLİĞİ (DATA MİNİNG)

Dr. Öğr. Üyesi Alper Talha Karadeniz

2025-2026