VERİ TEMİZLEME (DATA CLEANİNG)

Veri Kalitesi Neden Böyle Bir Sorun?

Bir iş zekası sistemi, bir veri ambarı sistemine yüklenen verileri analiz etmeyi ve raporlamayı çok daha basit hale getirirken, verilerin varlığı tek başına yöneticilerin sorunsuz kararlar almasını sağlamaz; kalite verileri önemli.

Veri kalitesi sorunlarının nedenlerinden biri , operasyonel sistemler ve kurumsal uygulamalardan oluşan bir yama içinde barındırılan kaynak verilerdir . Bu veri kaynaklarının her biri dağınık veya yanlış yerleştirilmiş değerlere, güncel olmayan ve yinelenen kayıtlara ve müşteriler, ürünler, işlemler, finanslar ve daha fazlası için tutarsız (veya tanımlanmamış) veri standartlarına ve biçimlerine sahip olabilir. Veri kalitesi sorunları, bir kuruluş bir birleşme veya satın alma sırasında verileri konsolide ettiğinde de ortaya çıkabilir . Ancak veri kalitesi sorunlarına belki de en büyük katkı, verilerin insanlar tarafından girilmesi, düzenlenmesi, sürdürülmesi, manipüle edilmesi ve raporlanmasıdır. Bu sorunlar yalnızca buluttaki modern veri ve analitik çözüm mimarilerinde daha da kötüleşir . Çoğu veri göllerini veya diğer ham veri alım veri işlem hatlarını içerir ve bunların birincil amacı hızdır, kaliteye gerek yoktur. Örneğin, akış ve IoT gibi diğer kullanım durumları, eski veri kalitesi araçlarının ve süreçlerinin şu anda kapsamadığı yeni modellerdir. Stratejik karar vermeyi etkileyen iş açısından kritik operasyonel bilgilerin doğruluğunu ve değerini korumak için işletmeler, veri kalitesi tekniklerini iş süreçlerine, kurumsal uygulamalara ve veri entegrasyonuna yerleştiren bir veri kalitesi stratejisi uygulamalıdır. Ayrıca, modern veri ve analitik mimarilerinde aşağı akışa yönelik yaklaşımları da düşünmelidirler.

Yüzeysel olarak bakıldığında, veri kalitesinin kötü verileri - bir şekilde eksik, yanlış veya geçersiz olan verileri - temizlemekle ilgili olduğu açıktır. Ancak verilerin güvenilir olmasını sağlamak için, verilerin ne kadar "kötü" olduğunu değerlendirmek için veri kalitesinin temel boyutlarını anlamak önemlidir.

Veri kalitesinin **6 boyutu şunlardır**:

- Tamlık
- Tutarlılık
- Uygunluk

- Doğruluk
- Bütünlük
- Zamanındalık

1. **Tamlık**

Tamlık, beklenen kapsamlılık olarak tanımlanır. İsteğe bağlı veriler eksik olsa bile veriler tamamlanabilir. Veriler beklentileri karşıladığı sürece verilerin eksiksiz olduğu kabul edilir. Örneğin, bir müşterinin adı ve soyadı zorunludur, ancak ikinci adı isteğe bağlıdır; bu nedenle, bir ikinci ad mevcut olmasa bile bir kayıt tamamlanmış olarak kabul edilebilir.

2. **Tutarlılık**

Tutarlılık, tüm sistemlerdeki verilerin aynı bilgileri yansıttığı ve kuruluş genelinde birbiriyle senkronize olduğu anlamına gelir. Örnekler:

- Bir iş birimi durumu kapalı, ancak bu iş birimi için satış var
- Çalışan durumu sona erdirildi, ancak ödeme durumu aktif.

3. **Ugunluk**

Uygunluk, verilerin veri türü, boyutu ve biçimi gibi standart veri tanımlarını takip ettiği anlamına gelir. Örneğin, müşterinin doğum tarihi "aa / gg / yyyy" biçimindedir.

4. **Doğruluk**

Doğruluk, verilerin gerçek dünya nesnesini VEYA açıklanan bir olayı doğru şekilde yansıttığı derecedir. Örnekler:

- İş biriminin satışları gerçek değerdir.
- Çalışan veritabanındaki bir çalışanın adresi gerçek adrestir.

5. **Bütünlük**

Bütünlük, ilişkilerdeki verilerin geçerliliği anlamına gelir ve bir veritabanındaki tüm verilerin izlenebilmesini ve diğer verilere bağlanabilmesini sağlar. Örneğin, bir müşteri veri tabanında, aralarında geçerli bir müşteri, adresler ve ilişki bulunmalıdır. Müşterinin olmadığı bir adres ilişkisi verisi varsa, bu veri geçerli değildir ve artık bir kayıt olarak kabul edilir.

6. **Zamanındalık**

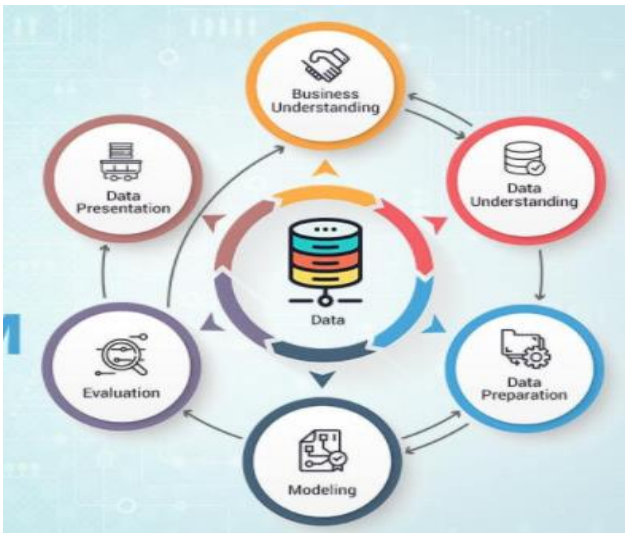
Zamanlılık, bilginin beklendiğinde ve ihtiyaç duyulduğunda mevcut olup olmadığını belirtir. Verilerin güncelliği çok önemlidir. Bu şu şekilde yansıtılır:

- Üç aylık sonuçlarını belirli bir zaman çerçevesi içinde yayınlaması gereken şirketler
- Müşterilere güncel bilgiler sağlayan müşteri hizmetleri
- Kredi kartı hesap hareketlerini gerçek zamanlı olarak kontrol eden kredi sistemi

Zamanındalık, kullanıcı beklentisine bağlıdır. Konaklama işletmelerinde oda tahsis sistemi için çevrimiçi veri kullanılabilirliği gerekli olabilir, ancak gecelik veriler bir faturalama sistemi için tamamen kabul edilebilir olabilir.

Gördüğünüz gibi, veri kalitesi dikkate alınması gereken önemli bir konudur.

Standart Veri Madenciliği Süreci



CRIPS-DM Metodolojisi (Cross-Industry Standard Process for Data Mining)

CRIPS-DM analitik, veri madenciliği ve veri biliminde en popüler metodolojidir. Veri madenciliği projelerini planlama ve yürütmede kullanılan bir süreç modelidir. Bu model 6 aşamadan oluşmaktadır.

1. İşi Tanımlama (Business Understanding): Başlangıç olarak proje hedeflerini ve ihtiyaçlarını anlama ve bunu veri madenciliği tanımına dönüştürme aşamasıdır.

2. Veriyi Anlama (Data Understanding): Bu aşamada veri toplama işlemiyle başlar, veri kalitesi problemlerini belirleme, veriden ilk görüleri çıkartma.. diye verinin probleme ne kadar çözüm getirdiğiyle devam eder.

3. Veriyi Hazırlama (Data Preparation): Topladığımız veriden veri seçme, veri temizleme, veri dönüştürme… gibi model uygun son veri setini elde etmek için yapılan işlemlerdir.

4. Modelleme (Modeling): Bu aşamada çeşitli modelleme tekniklerinin ve algoritmalarının seçilmesi, parametrelerin seçilmesi ve uygulama işlemleri gerçekleştirilir.

5. Değerlendirme: Bu aşamada oluşturulan modelin deneme ve gözden geçirilmesi yapılır, gerekiyorsa iyileştirmeler yapılır.

6. Uygulama: Son aşamada ise modelin analistlere ve son kullanıcılara sunulup iş süreçlerinde kullanılacak hale getirilir.

Linux AWK Komutu Nedir?

Bir programlama dili olan awk'dır. Adını tasarımcılarının Alfred Aho, Peter Weinberger and Brian Kernighan baş harflerinden almıştır.

AWK ile CSV gibi metin-tabanlı dosyalardaki veriler düzenlenebilir ve dönüştürülebilir ve veriler değerlendirilip isteğe göre raporlanabilir. sed gibi akış editörlerinde kullanılan veri bulma/düzenleme/dönüştürme komutlarına ek olarak C deki gibi genel programlama yapıları içermektedir, bu sebepten dolayı tam donanımlı bir programlama dili olarak geçmektedir.

AWK'ın komut satırında kullanılışı aşağıdaki gibidir :

awk [parametreler] -f program_dosyası [--] dosya ...

awk [parametreler] [--] program_kodu dosya ...

<u>Derste yapılan örneklerin ekran görüntüleri aşağıda bulunmaktadır.</u>

Komut satırı:

```
cat access.log.2.txt |grep '"POST'| wc -l
```

Ekran çıktısı:

```
93039
```

Komut satırı:

```
awk '/^42/ { print $0 }' access.log.2.txt^C
```

: access.log.2.txt incelenecek(kullanılacak) dosya

/^42/ filtresi

- Köşeli parantezden sonraki kısım proses kısmı ne iş yapacağı yazdığımız kısım. Dosyanın satırlarını incele ve 42 ile başlayan satırları getir ve ekrana bas.

Ekran çıktısı:

```
Kit/601.2.4 (KHTML, like Gecko) Version/9.0.1 Safari/601.2.4 facebookexternalhit
/1.1 Facebot Twitterbot/1.0" "-"
42.110.154.133 - - [10/Feb/2021:20:19:43 +0100] "GET /apache-log/access.log HTTP
/1.1" 200 28136902 "-" "MobileSafari/604.1 CFNetwork/1220.1 Darwin/20.3.0" "-"
42.110.154.133 - - [10/Feb/2021:20:19:55 +0100] "GET /apache-log/access.log HTTP
/1.1" 200 28137061 "-" "Mozilla/5.0 (iPhone; CPU iPhone OS 14_4 like Mac OS X) A
ppleWebKit/605.1.15 (KHTML, like Gecko) Version/14.0.3 Mobile/15E148 Safari/604.
1" "-"
42.110.154.133 - - [10/Feb/2021:20:20:03 +0100] "GET /favicon.ico HTTP/1.1" 404
217 "http://www.almhuette-raith.at/apache-log/access.log" "Mozilla/5.0 (iPhone;
CPU iPhone OS 14_4 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) Versi
on/14.0.3 Mobile/15E148 Safari/604.1" "-"
42.236.10.81 - - [11/Feb/2021:00:33:09 +0100] "GET / HTTP/1.1" 200 10439 "http:/
/almhuette-raith.at/" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (K
HTML, like Gecko) Chrome/50.4.2661.102 Safari/537.36; 360Spider" "-"
42.236.10.113 - - [11/Feb/2021:00:35:59 +0100] "GET / HTTP/1.1" 200 10479 "http:
//www.almhuette-raith.at/" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.
36 (KHTML, like Gecko) Chrome/50.4.2661.102 Safari/537.36; 360Spider" "-"
```

Dosyada en uzun satırı ekrana bastırma

Komut satırı:

```
elif@elif-VirtualBox:~/Desktop$ awk '{ if (length($0) > max) max = length($0)}
> END {print max}' access.log.2.txt
```

Ekran çıktısı:

```
7821
```

1000 karakter uzunluğundan büyük olan logları ekrana bastırma

Komut satırı:

```
awk 'length($0) > 1000' access.log.2.txt| wc -l
```

Ekran çıktısı:

```
36
```

Awk kendi içerisinde okuduğu dosyayı boşluklara göre ayırır. Herbir satırı parçalarken yani split ederken herbir token oluşmuş olur. Örneğin $0 : 0. Token iken $1 : 1. Token anlamına gelir. Yani boşluğa kadar olan kısımlar tokenlerdir.

Awk nın head ve tail olarak iki farklı komutu bulunur. Herhangi bir dosyanın head dediğimizde ilk 10 satırı ekrana bastırılırken tailde son 10 eleman ekrana bastırılır.

Access.log.2.txt dosyasının ilk 20 satırındakini token 1 i görme komutu ve çıktısı --

```
awk ' {print $1}' access.log.2.txt | head -n 20
```

```
13.66.139.0
157.48.153.185
157.48.153.185
216.244.66.230
54.36.148.92
92.101.35.224
73.166.162.225
73.166.162.225
54.36.148.108
54.36.148.1
162.158.203.24
35.237.4.214
42.236.10.125
42.236.10.125
42.236.10.125
42.236.10.125
42.236.10.117
42.236.10.114
42.236.10.114
```

Access.log.2.txt dosyasının ilk 20 satırındakini token 4 ü görme komutu ve çıktısı --

```
awk ' {print $4}' access.log.2.txt | head -n 20
```

```
[19/Dec/2020:13:57:26
[19/Dec/2020:14:08:06
[19/Dec/2020:14:08:08
[19/Dec/2020:14:14:26
[19/Dec/2020:14:16:44
[19/Dec/2020:14:29:21
[19/Dec/2020:14:58:59
[19/Dec/2020:14:58:59
[19/Dec/2020:15:09:30
[19/Dec/2020:15:09:31
[19/Dec/2020:15:16:50
[19/Dec/2020:15:22:40
[19/Dec/2020:15:23:10
[19/Dec/2020:15:23:11
[19/Dec/2020:15:23:11
[19/Dec/2020:15:23:11
[19/Dec/2020:15:23:11
[19/Dec/2020:15:23:11
[19/Dec/2020:15:23:11
```

Access.log.2.txt dosyasının token 4 ünde aylardan Şubat ayında log sayısı komutu ve çıktısı --

```
awk ' $4 ~ /Feb/' access.log.2.txt | wc -l
```

```
24026
```

Access.log.2.txt dosyasının token 4 ünde aylardan Şubat ayında logları gösteren komut ve çıktısı --

```
awk '$4 ~ /Feb/ ' access.log.2.txt | head -n 20
```

```
193.106.31.130 - - [01/Feb/2021:00:04:03 +0100] "GET /administrator/index.php HT
TP/1.0" 200 4250 "-" "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0)" "-"
193.106.31.130 - - [01/Feb/2021:00:04:03 +0100] "POST /administrator/index.php H
TTP/1.0" 200 4481 "-" "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0)" "-"
193.106.31.130 - - [01/Feb/2021:00:04:03 +0100] "POST /administrator/index.php H
TTP/1.0" 200 4481 "-" "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0)" "-"
193.106.31.130 - - [01/Feb/2021:00:04:03 +0100] "POST /administrator/index.php H
TTP/1.0" 200 4481 "-" "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0)" "-"
193.106.31.130 - - [01/Feb/2021:00:04:03 +0100] "POST /administrator/index.php H
TTP/1.0" 200 4481 "-" "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0)" "-"
193.106.31.130 - - [01/Feb/2021:00:04:04 +0100] "POST /administrator/index.php H
TTP/1.0" 200 4481 "-" "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0)" "-"
193.106.31.130 - - [01/Feb/2021:00:04:04 +0100] "POST /administrator/index.php H
TTP/1.0" 200 4481 "-" "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0)" "-"
193.106.31.130 - - [01/Feb/2021:00:04:04 +0100] "POST /administrator/index.php H
TTP/1.0" 200 4481 "-" "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0)" "-"
193.106.31.130 - - [01/Feb/2021:00:04:04 +0100] "POST /administrator/index.php H
TTP/1.0" 200 4481 "-" "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0)" "-"
193.106.31.130 - - [01/Feb/2021:00:04:04 +0100] "POST /administrator/index.php H
TTP/1.0" 200 4481 "-" "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0)" "-"
```

3. Gün Bitti 😊