Data Science Intern Case Study

Şule Aktaş

0suleaktas@gmail.com

1. Keşifsel Veri Analizi (Exploratory Data Analysis - EDA)

1.1. Python kullanılarak;

1.1.1. Gerekli kütüphaneler ve Excel dosyası yüklendi.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
#Excel dosyasini yukleme
df = pd.read_excel('side_effect_data.xlsx')
```

1.1.2. Veri kümesi kapsamlı bir şekilde incelendi.

1.1.2.1. Veri Seti hakkında genel bilgi almak için **info()** fonksiyonu kullanılarak aşağıdaki kod satırı yazıldı.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2357 entries, 0 to 2356
Data columns (total 19 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Kullanici_id                    2357 non-null   int64
 1   Cinsiyet                        1579 non-null   object
 2   Dogum_Tarihi                    2357 non-null   datetime64[ns]
 3   Uyruk                           2357 non-null   object
 4   Il                              2130 non-null   object
 5   Ilac_Adi                        2357 non-null   object
 6   Ilac_Baslangic_Tarihi           2357 non-null   datetime64[ns]
 7   Ilac_Bitis_Tarihi               2357 non-null   datetime64[ns]
 8   Yan_Etki                        2357 non-null   object
 9   Yan_Etki_Bildirim_Tarihi        2357 non-null   datetime64[ns]
 10  Alerjilerim                     1873 non-null   object
 11  Kronik Hastaliklarim            1965 non-null   object
 12  Baba Kronik Hastaliklari        2201 non-null   object
 13  Anne Kronik Hastaliklari        2140 non-null   object
 14  Kiz Kardes Kronik Hastaliklari  2260 non-null   object
 15  Erkek Kardes Kronik Hastaliklari 2236 non-null  object
 16  Kan Grubu                       2010 non-null   object
 17  Kilo                            2064 non-null   float64
 18  Boy                             2243 non-null   float64
dtypes: datetime64[ns](4), float64(2), int64(1), object(12)
memory usage: 350.0+ KB
```

1.1.2.2. Veri Setindeki ilk 5 satırı görmek için **head()** fonksiyonu ve son 5 satırı görmek için **tail()** fonksiyonu kullanılarak aşağıdaki kod satırları yazıldı.

```
#Ilk 5 satirini ogrenme
df.head()
```

	Kullanici_id	Cinsiyet	Dogum_Tarihi	Uyruk	Il	Ilac_Adi	Ilac_Baslangic_Tarihi	Ilac_Bitis_Tarihi	Yan_Etki	Yan_Etki_Bildirim_Tarihi
0	107	Male	1960-03-01	Turkiye	Canakkale	trifluoperazine	2022-01-09	2022-03-04	Kabizlik	2022-02-19 18:28:43
1	140	Male	1939-10-12	Turkiye	Trabzon	fluphenazine hcl	2022-01-09	2022-03-08	Yorgunluk	2022-02-03 20:48:17
2	2	Female	1976-12-17	Turkiye	Canakkale	warfarin sodium	2022-01-11	2022-03-12	Carpinti	2022-02-04 05:29:20
3	83	Male	1977-06-17	Turkiye	Adana	valproic acid	2022-01-04	2022-03-12	Sinirlilik	2022-02-08 01:01:21
4	7	Female	1976-09-03	Turkiye	Izmir	carbamazepine extended release	2022-01-13	2022-03-06	Agizda Farkli Bir Tat	2022-02-12 05:33:06

```
#Son 5 satirini ogrenme
df.tail()
```

	Kullanici_id	Cinsiyet	Dogum_Tarihi	Uyruk	Il	Ilac_Adi	Ilac_Baslangic_Tarihi	Ilac_Bitis_Tarihi	Yan_Etki	Yan_Etki_Bildirim_Tarihi
2352	9	NaN	1957-01-04	Turkiye	NaN	desoximetasone spray, non-aerosol	2022-01-13	2022-03-04	Ishal	2022-02-12 19:13:43
2353	101	Female	2004-11-09	Turkiye	Mersin	olanzapine-fluoxetine	2022-01-02	2022-03-05	Agizda Farkli Bir Tat	2022-02-19 17:39:48
2354	127	Female	1951-11-29	Turkiye	Mersin	trazodone	2022-01-02	2022-03-12	Yorgunluk	2022-02-03 20:48:17
2355	178	Male	1980-01-30	Turkiye	Kayseri	duloxetine hydrochloride	2022-01-02	2022-03-08	Carpinti	2022-02-04 05:29:20
2356	174	Female	1986-11-07	Turkiye	Istanbul	valproic acid	2022-01-06	2022-03-06	Istah Artisi	2022-02-17 07:08:01

1.1.2.3. **axes(), ndim(), size() , shape() ve describe()** fonksiyonları kullanılarak veri setindeki satır ve sütun bilgisine, boyut bilgisine, toplam eleman sayısı bilgisine, satır ve sütun sayısı bilgisine, sayısal sütunların istatiksel bilgilerine aşağıdaki kod satırları yazılarak ulaşıldı.

```
#Satir ve sutun bilgisini ogrenme
df.axes
```

```
[RangeIndex(start=0, stop=2357, step=1),
 Index(['Kullanici_id', 'Cinsiyet', 'Dogum_Tarihi', 'Uyruk', 'Il', 'Ilac_Adi',
        'Ilac_Baslangic_Tarihi', 'Ilac_Bitis_Tarihi', 'Yan_Etki',
        'Yan_Etki_Bildirim_Tarihi', 'Alerjilerim', 'Kronik Hastaliklarim',
        'Baba Kronik Hastaliklari', 'Anne Kronik Hastaliklari',
        'Kiz Kardes Kronik Hastaliklari', 'Erkek Kardes Kronik Hastaliklari',
        'Kan Grubu', 'Kilo', 'Boy'],
       dtype='object')]
```

```
#Boyut bilgisini ogrenme
df.ndim
```

```
2
```

```
#Toplam eleman sayisini ogrenme
df.size
```

```
44783
```

```
#Satir, sutun sayisini ogrenme
df.shape
```

```
(2357, 19)
```

```
#Sayisal sutunlarin istatistiklerini ogrenme
df.describe()
```

	Kullanici_id	Kilo	Boy
count	2357.000000	2064.000000	2243.000000
mean	97.216801	80.863857	174.638431
std	57.017200	18.635269	16.516552
min	1.000000	50.000000	145.000000
25%	47.000000	65.000000	160.000000
50%	97.000000	83.000000	176.000000
75%	146.000000	96.000000	187.000000
max	196.000000	110.000000	203.000000

1.1.3. Değişken türleri incelendi.

1.1.3.1. Kategorilerin değişken tipleri öğrenmek için **dtypes()** fonksiyonu kullanılarak aşağıdaki kod satırı yazıldı.

```
#Sutunlarin veri tiplerini ogrenme
df.dtypes
```

```
Kullanici_id                          int64
Cinsiyet                             object
Dogum_Tarihi                 datetime64[ns]
Uyruk                                object
Il                                   object
Ilac_Adi                             object
Ilac_Baslangic_Tarihi        datetime64[ns]
Ilac_Bitis_Tarihi            datetime64[ns]
Yan_Etki                             object
Yan_Etki_Bildirim_Tarihi     datetime64[ns]
Alerjilerim                          object
Kronik Hastaliklarim                 object
Baba Kronik Hastaliklari             object
Anne Kronik Hastaliklari             object
Kiz Kardes Kronik Hastaliklari       object
Erkek Kardes Kronik Hastaliklari     object
Kan Grubu                            object
Kilo                                float64
Boy                                 float64
dtype: object
```

1.1.4. Eksik veriler tespit edildi.

1.1.4.1. Kategorilerdeki eksik verileri tespit etmek için **isnull().sum()** fonksiyonu kullanılarak aşağıdaki kod satırı yazıldı.

```
#Eksik degerleri ogrenme
df.isnull().sum()
```

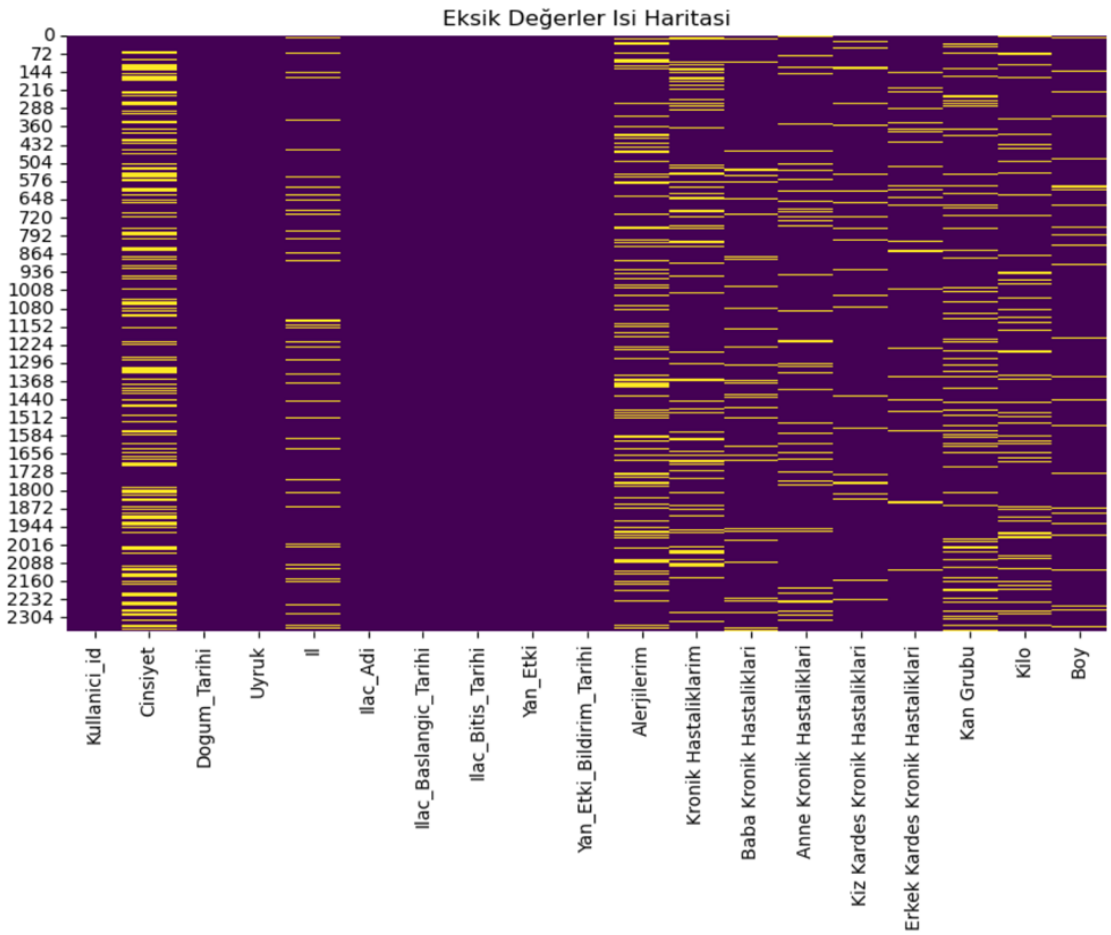
```
Kullanici_id                          0
Cinsiyet                            778
Dogum_Tarihi                          0
Uyruk                                 0
Il                                  227
Ilac_Adi                              0
Ilac_Baslangic_Tarihi                 0
Ilac_Bitis_Tarihi                     0
Yan_Etki                              0
Yan_Etki_Bildirim_Tarihi              0
Alerjilerim                         484
Kronik Hastaliklarim                392
Baba Kronik Hastaliklari            156
Anne Kronik Hastaliklari            217
Kiz Kardes Kronik Hastaliklari       97
Erkek Kardes Kronik Hastaliklari    121
Kan Grubu                           347
Kilo                                293
Boy                                 114
dtype: int64
```

1.1.5. Veri görselleştirme teknikleri kullanıldı. (Matplotlib, Seaborn)

1.1.5.1. **heatmap()** fonksiyonu kullanılarak Isı Haritası oluşturuldu. Grafik için bir figür (çerçeve) yapıldı ve **figsize=(10,6)** argümanı ile bu figürün genişlik ve yükseklik boyutları ayarlandı.

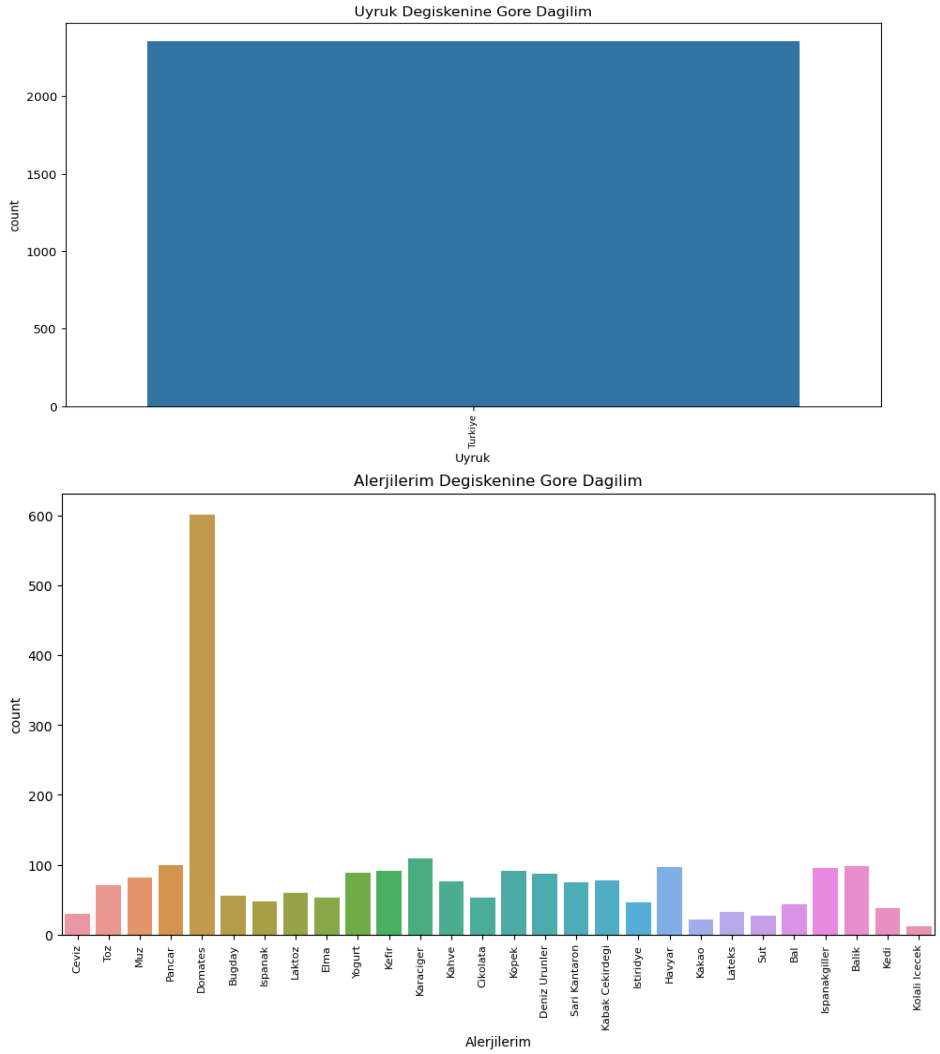
1.1.5.2. Bu, görselin 10 birim genişliğinde ve 6 birim yüksekliğinde olmasını sağladı. **df.isnull()** kısmı, DataFrame'deki (df) eksik değerleri (NaN ya da boş hücreler) tespit edildi. **isnull()** fonksiyonu ile veri setindeki her hücrede eksik değer olup olmadığı kontrol edildi ve True/False (1/0) döndürdü. **viridis**, Seaborn ve Matplotlib'deki popüler bir renk haritasıdır ve sarıdan mora doğru bir renk geçişi sunar. **title()** fonksiyonu ile Haritaya başlık belirlendi ve **show()** fonksiyonu ile grafik ekrana bastırıldı.

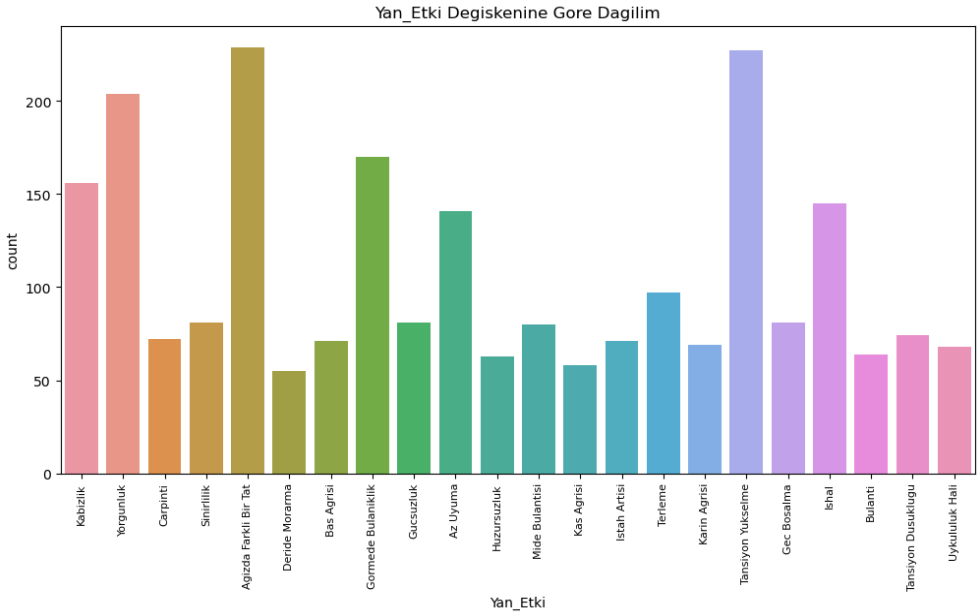
```
#Eksik degerlerin isi haritasinda gosterilmesi
plt.figure(figsize=(10,6))
sns.heatmap(df.isnull(), cbar=False, cmap='viridis')
plt.title('Eksik Değerler Isi Haritasi')
plt.show()
```



1.1.5.3. **df.select_dtypes(include=['object'])** Bu komut ile veri setindeki yalnızca kategorik değişkenleri seçildi. Kategorik değişkenler, genellikle object veri tipinde olan ve sayısal olmayan sütunlar (örneğin metin ya da kategorilere ayrılmış veri) içerir. Seaborn kütüphanesinden gelen **countplot()** fonksiyonu ile çubuk grafik oluşturuldu. **plt.xticks()** fonksiyonu ile x eksenindeki kategorik değerlerin (etiketlerin) yönünü ayarlandı. **rotation=90 ile** etiketler 90 derece döndürülerek dikey hale getirildi. **fontsize=8 ile** etiketlerin yazı boyutu 8 birim olarak ayarlandı. Örnek grafikler aşağıdaki gibidir.

```
#Kategorik degiskenler icin cubuk grafigi olusturma
kategorik_sutunlar = df.select_dtypes(include=['object']).columns
for sutun in kategorik_sutunlar:
    plt.figure(figsize=(12, 6))
    sns.countplot(x=sutun, data=df)
    plt.title(f'{sutun} Degiskenine Gore Dagilim')
    plt.xticks(rotation=90, fontsize=8)
    plt.show()
```





2. Veri Ön İşleme (Data Pre-Processing)

2.1. Boşluk karakterlerini NaN'a dönüştürme işlemi yapıldı.

2.1.1. **replace()** fonksiyonu kullanılarak **' '** (boşluk karakteri) ile eşleşen değerler **np.nan** ile değiştirilir

```
#Bosluklari NaN a donusturme islemi
import numpy as np
df.replace(' ', np.nan, inplace=True)
```

2.2. Eksik verileri tamamlama işlemi yapıldı.

2.2.1. Sayısal sütunlardaki eksik veriler için **SimpleImputer** sınıfından bir nesne oluşturuldu ve stratejisini **'median'** olarak belirlendi. Bu strateji, eksik olan (NaN) hücreleri ortanca değerler ile belirledi. **sayisal_sutunlar** değişkeni ile sayısal sütunlar seçildi. **fit()** fonksiyonu her sütun için en sık görülen değeri bulur ve model olarak saklar. **transform()** fonksiyonu bu bulduğu en sık görülen değerleri kullanarak, her sütundaki eksik hücreleri doldurur.

```
#SimpleImputer ile Eksik verileri tamamlama
from sklearn.impute import SimpleImputer

sayisal_sutunlar= ['Kilo', 'Boy']
imputer = SimpleImputer(strategy='median')
df[sayisal_sutunlar] = imputer.fit_transform(df[sayisal_sutunlar])
```

2.2.2. Kategorik sütunlardaki eksik veriler için **SimpleImputer** sınıfından bir nesne oluşturuldu ve stratejisini **'most_frequent'** olarak belirlendi. Bu strateji, eksik olan (NaN) hücreleri o sütunda en sık görülen değerle doldurdu. **kategorik_sutunlar** değişkeni ile kategorik sütunlar seçildi.

```
kategorik_sutunlar = ['Kiz Kardes Kronik Hastaliklari','Erkek Kardes Kronik Hastaliklari',"Cinsiyet", "Il", "Alerjilerim"
                      'Anne Kronik Hastaliklari']
imputer = SimpleImputer(strategy='most_frequent')
df[kategorik_sutunlar] = imputer.fit_transform(df[kategorik_sutunlar])
```

2.3. Eksik veriler tespit edildi.

2.3.1. Kategorilerdeki eksik verileri doldurduktan sonra test etmek için **isnull().sum()** fonksiyonu kullanılarak aşağıdaki kod satırı yazıldı.

```
#Eksik degerleri sayma islemi
missing_values = df.isnull().sum()
print(missing_values)
```

```
Kullanici_id                      0
Cinsiyet                          0
Dogum_Tarihi                      0
Uyruk                             0
Il                                0
Ilac_Adi                          0
Ilac_Baslangic_Tarihi             0
Ilac_Bitis_Tarihi                 0
Yan_Etki                          0
Yan_Etki_Bildirim_Tarihi          0
Alerjilerim                       0
Kronik Hastaliklarim              0
Baba Kronik Hastaliklari          0
Anne Kronik Hastaliklari          0
Kiz Kardes Kronik Hastaliklari    0
Erkek Kardes Kronik Hastaliklari  0
Kan Grubu                         0
Kilo                              0
Boy                               0
dtype: int64
```