Veri ön işleme

Kalp rahatsızlığı veri seti

Hedefler

Kalp rahatsızlığı veri setini kullanarak:

1. Veri temizleme adımlarını yazılım ile gerçekleştirmek
2. Veri yeniden yapılandırma adımlarını yazılım ile gerçekleştirmek
3. Veriyi, veri madenciliği modellerine ait algoritmalar uygulanacak hale getirmek.

Kütüphaneler

scikit-learn

- Tahmine dayalı veri analizi için basit ve verimli araçlar sunar.
- Herkes tarafından erişilebilir ve çeşitli bağlamlarda yeniden kullanılabilir.
- NumPy, SciPy ve matplotlib üzerine kuruludur.
- Açık kaynak, ticari olarak kullanılabilir - BSD lisansı ile kullanılabilir.
- Basit makine öğrenmesi çalışmaları için hazır yapılar içerir.

NumPy

- Hızlı ve çok yönlü NumPy vektörleştirme, indeksleme ve yayınlama kavramları, günümüzde dizi hesaplamanın fiili standartlarıdır.
- Kapsamlı matematiksel fonksiyonlar, rastgele sayı üreteçleri, lineer cebir rutinleri, Fourier dönüşümleri ve daha fazlasını sunar.
- Yüksek seviyeli sözdizimi nedeniyle herhangi bir arka plan veya deneyim seviyesinden programcılar için erişilebilir ve onları üretken hale getirir.
- Açık kaynak, ticari olarak kullanılabilir - BSD lisansı ile kullanılabilir.

Etkinlikte gerçekleştirilecek işlemler

Veri temizleme

- Kayıp veriler sorununun çözülmesi
- Gürültülü veri sorununun çözülmesi

Veri yeniden yapılandırma

- Normalizasyon
- Dönüştürme

Veri setimizi tanıyalım

Tablo 1: Kalp rahatsızlığı veri seti açıklaması

Öznitelik(sütun) adı	**Anlamı**	**Veri Türü**	**Değerler**
yas	yaş bilgisi	sayısal	beyan edilen yaş bilgisi
cinsiyet	cinsiyet bilgisi	kategorik	{erkek,kadin}
gogus_agrisi_tipi	Görülen göğüs ağrısı tip bilgisi	kategorik	{asemptomatik,atipik anjinal,anjinal olmayan,atipik anjinal}
hareketsiz_kan_basinci	Dinlenmiş haldeki ölçülen kan basıncı değeri	sayısal	ölçülen değer
serum_kolestrol	Kandaki kolestrol değeri(Mg / dl)	sayısal	ölçülen değer
aclik_kan_sekeri	Açlık kan şekeri>120 den büyükse--> 1 değilse 0 (mg / dl)	kategorik	{0,1}
elektrokardiyografi	ekg yorumu	kategorik	{ST-T anormal,sol ventikuler hipertrofi,normal}
en_yuksek_kalp_hizi	ölçülen en yüksek kalp hızı	sayısal	ölçülen değer
anjin_bagli_egsersiz	egzersize bağlı anjin var-->1 yok -->0	kategorik	{0,1}
st_depresyonu	Dinlenme durumuna egzersize bağlı ölcülen ST depresyon değeri	sayısal	ölçülen değer
st_egimi	ST değerninin eğim yönü	kategorik	{yukari egimli,duz,asagi egimli}
buyuk_damarlar	pik egzersiz için floroskopi ile renklendirilmiş büyük damar sayısı	sayısal	ölçülen değer
talasemi	talasemi tipi	kategorik	{normal,sabit defekt,tersinir defekt}
kalp_rahatsizligi	kalp rahatsızlığı var->1 yok --> 0	kategorik	{0,1}

İşlem adımlarımızı sıralayacak olursak.

1. Gerekli kütüphanelerin eklenmesi.
2. Veri setinin bir dataFrame yapısına aktarılması.
3. Sütun isimlerinde kolaylık olması için kısaltma yapılması.
4. dataFrame yapısına aktarılan veri setinin bilgilerinin alınması ve veri setinin incelenmesi.
5. Gürültülü ve kayıp verilerle ilgili işlemlerin yapılması.
6. Sayısal verilerin normalizasyon işleminin yapılması.
7. Metin ile ifade edilmiş kategorik verilerin sayılar ile etiketlenmesi.
8. Sayısal verilerin kategorik verilere dönüştürülmesi.

* Bu işlemleri gerçekleştirmeye geçmeden önce size verilen *Dijital Materyal O.4.6.1* dosyasını proje dosyanıza dm461.csv adı ile kayıt ediniz.

Gerekli Kütüphanelerin eklenmesi ve veri setinin aktarılması

Kod 1:

- İlk 5 satırda gerekli kütüphanelerin dahil edilmesi sağlanmıştır.
- 6. satırda veriler dosyadan veri isimli bir dataFrame yapısına aktarılmıştır.

```
1  import pandas as pd
2  import numpy as np
3  from sklearn.preprocessing import minmax_scale
4  from sklearn.impute import SimpleImputer
5  from sklearn.preprocessing import LabelEncoder
6  veri = pd.read_csv("dm461.csv")
```

Kod 1: Gerekli Kütüphanelerin eklenmesi ve veri setinin aktarılması sağlayan kod

Sütun isimlerinin değiştirilmesi

Kod 2:

- 7. satırda *rename* fonksiyonu ile sütun isimleri değiştirilmiştir.
- *columns* parametresine verilen sözlük veri tipi ile sütun isimleri değiştirilmiştir. Sözlük tipinin anahtar/ değer ikilileri ‘eski_ad’:’yeni_ad’ yapısında değişecek sütun isimlerini ve yeni adları verilmiştir.
- *inplace* parametresine verilen *True* değeri, değişikliklerin *dataFrame* üzerinde yapılmasını sağlamıştır.

```
7   veri.rename(columns={
        "gogus_agrisi_tipi":"gat",
        "hareketsiz_kan_basinci":"hkb",
        "serum_kolestrol":"sk",
        "aclik_kan_sekeri":"aks",
        "elektrokardiyografi":"ekg",
        "en_yuksek_kalp_hizi":"eykh",
        "anjin_bagli_egsersiz":"abe",
        "st_depresyonu":"st_d",
        "st_egimi":"st_e",
        "buyuk_damarlar":"bds"},inplace=True)
```

Kod 2: Sütun isimlerinin değiştirilmesini sağlayan kod

veri setinin incelenmesi-1

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   yas                 303 non-null    float64
 1   cinsiyet            303 non-null    object
 2   gat                 303 non-null    object
 3   hkb                 303 non-null    float64
 4   sk                  303 non-null    float64
 5   aks                 303 non-null    int64
 6   ekg                 303 non-null    object
 7   eykh                303 non-null    float64
 8   abe                 303 non-null    int64
 9   st_d                303 non-null    float64
 10  st_e                303 non-null    object
 11  bds                 299 non-null    float64
 12  talasemi            301 non-null    object
 13  kalp_rahatsizligi   303 non-null    int64
dtypes: float64(6), int64(3), object(5)
memory usage: 27.3+ KB
None
```

Görsel 1: info() fonksiyonuna ait dönen bilgilerin çıktısı-1

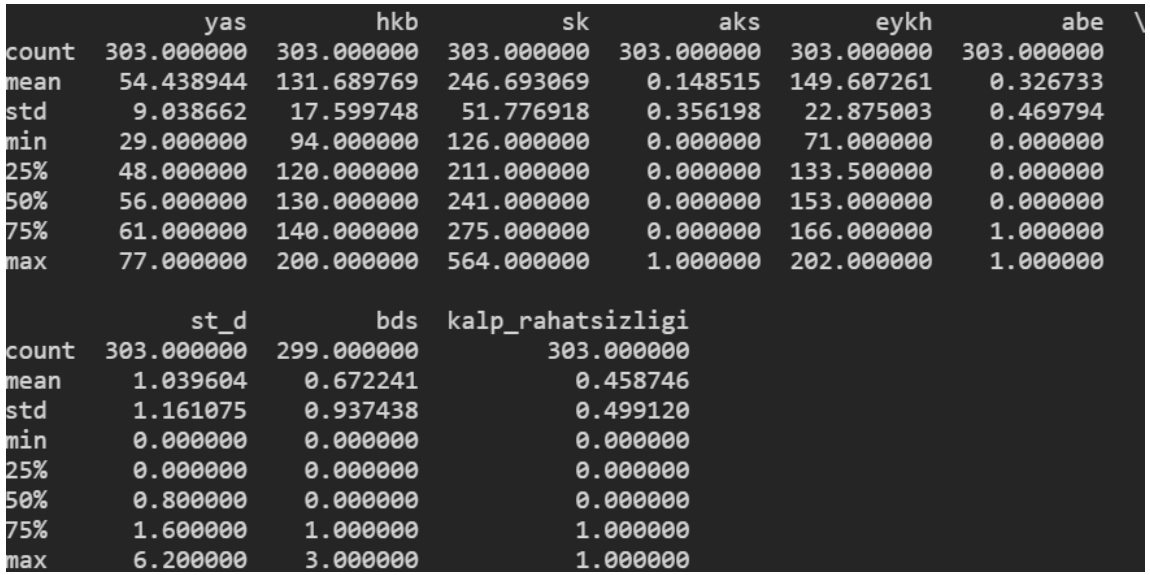
```
8   print(veri.info())
```

Kod 3: Veri setinin incelenmesini sağlayan kod-1

Kod 3:

- 8. satırda dataFrame yapısında buluna info() fonksiyonundan dönen veriler Görsel 1'de gösterildiği gibi ekrana yazdırılmıştır.
- Burada veri setinin kaç kayıttan oluştuğu(RangeIndex:302 etires,0 to 302),kaç sütundan oluştuğu(total columns:14), sütun isimleri(Column), sütun sırası(#), sütunlarda kaç veri olduğu(Non-Null Count), veri tipleri(Dtype) ve hafızada ne kadar yer kapladığı(memory usage) bilgileri görülmektedir.
- Bu bilgilerden *bds* verisine ait 4 girdinin, *talasemi* verisine ait 2 verinin kayıp olduğu görülmektedir.

veri setinin incelenmesi-2



	yas	hkb	sk	aks	eykh	abe
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.438944	131.689769	246.693069	0.148515	149.607261	0.326733
std	9.038662	17.599748	51.776918	0.356198	22.875003	0.469794
min	29.000000	94.000000	126.000000	0.000000	71.000000	0.000000
25%	48.000000	120.000000	211.000000	0.000000	133.500000	0.000000
50%	56.000000	130.000000	241.000000	0.000000	153.000000	0.000000
75%	61.000000	140.000000	275.000000	0.000000	166.000000	1.000000
max	77.000000	200.000000	564.000000	1.000000	202.000000	1.000000

	st_d	bds	kalp_rahatsizligi
count	303.000000	299.000000	303.000000
mean	1.039604	0.672241	0.458746
std	1.161075	0.937438	0.499120
min	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000
50%	0.800000	0.000000	0.000000
75%	1.600000	1.000000	1.000000
max	6.200000	3.000000	1.000000

Görsel 2: describe() fonksiyonuna ait dönen bilgilerin çıktısı-1

```
9  print(veri.describe())
```

Kod 4: Veri setinin incelenmesini sağlayan kod-2

Kod 4:

- 9. satırda dataFrame yapısında buluna describe() fonksiyonundan dönen veriler Görsel 2'de gösterildiği gibi ekrana yazdırılmıştır.
- Burada veri setinde bulunan ve sayısal veri içeren her bir sütuna (verisine/özniteliğine/değişkenine) ait, kaç kayıt olduğu (count), ortalaması (mean), standart sapması (std), en küçük değeri (min), en büyük değeri (max) ve çeyreklik dilimlerine (Q1,Q2 veQ3) ait değerler gözükmektedir.
- Bu bilgiler bize aykırı değerleri ayıklamada yardımcı olacaktır.

Kayıp veri probleminin çözülmesi-1

Kod5:

- 10. satırda kullanılan *drop* fonksiyonu veri üzerinden satır ve sütun silme işlemi yapılmasını sağlamaktadır. Geriye belirtilen silme işlemi yapılmış yeni bir dataFrame döndürülmüş ve o yapı *veri2* değişkenine aktarılmıştır.
- *labels* parametresi silinecek sütun isimlerini içeren bir listedir. bds ve talasemi sütunları burada silinerek kayıp veri olan sütunlar (veriler) silinmiştir.
- *axis* parametresi (0:satırlar, 1: sütunlar) dataFrame silme yöntemini belirtmektedir.

Kod6:

- 11. satırda kullanılan *dropna* fonksiyonu veri üzerinde boş veri içeren satır ve sütunların silinmesini sağlar.
- *axis* parametresi (0:satırlar, 1: sütunlar) dataFrame silme yöntemini belirtmektedir. Burada kayıp veri içeren satırlar (kayıtlar) silinmiştir.

```
10 veri2=veri.drop(labels=["bds","talasemi"],axis=1)
```

Kod 5: Kayıp veri probleminin çözülmesini sağlayan kod-1

```
11 veri3= veri.dropna(axis=0)
```

Kod 6: Kayıp veri probleminin çözülmesini sağlayan kod-2

- Kod5 ile kayıp veri olan sütunlar ve Kod6 ile kayıp veri olan satırlar silinmiştir.
- Bu genelde istenen bir durum değildir.
- 4. etkinliğimizde gördüğümüz merkezi ölçütlerden birinin konulması yöntemi daha çok tercih edilmektedir.

Kayıp veri probleminin çözülmesi-2

Kod7:

- 12. satırda *scikit-learn* kütüphanesinin *SimpleImputer* sınıfından *imputer* isminde bir nesne türetilmiştir.
- Bu sınıftan türetilen nesneler genelde veri setlerindeki eksik verileri basit yollar ile gidermeyi sağlamaktadır.
- Bu nesne *missing_values* değeri olarak *np.nan* yani boş geçen değerler ile ilgileneceğini, *strategy* parametresine verilen *most_frequent* değeri ise kayıp verilere o veride en çok geçen değer verileceğini ifade etmektedir. Bu parametreye *mean*, *median* veya *constant* değerleri de verilebilmektedir.
- 13. ve 14. satırlarda ilgili sütunlara ait veriler tek bir veri içerdiği için bir sütun olarak düzeltilip değerleri *bds* ve *tal* gibi iki değişkene aktarılmıştır.

```
12  imputer=SimpleImputer(missing_values=np.nan,strat
    egy="most_frequent")
13  bds=veri["bds"].values.reshape(-1,1)
14  tal=veri["talasemi"].values.reshape(-1,1)
15  veri["bds"]=imputer.fit_transform(bds)
16  veri["talasemi"]=imputer.fit_transform(tal)
```

Kod 7: Kayıp veri probleminin çözülmesini sağlayan kod-3

Kod 7-devamı:

- 15. satırda 11. satırda oluşturulan *imputer* nesnesinde bulunan fit_*transform* fonksiyonuna 13. satırda hazırlanan *bds* değişkeni verilmiştir. Burada bu fonksiyon *bds* içindeki boş değerleri veride en çok geçen değerler ile doldurmuştur. Burada elde edilen yeni değerler ile oluşan değişken dataFrame'de bulunan *bds* sütununa aktarılmış, bu sayede *bds* sütunu kayıp veri sorunundan kurulmuştur.
- 16. satırda aynı işlem talasemi sütunu için gerçekleştirilmiştir.

Kayıp veri probleminin çözülmesi-2

Kod8:
Burada örnek olarak sk (serum kolestrol) sütununa ait gürültülü veriler tespit edilmiş ve temizlenmiştir.

- 17. ve 18. satırda *sk* verisine ait Q1 ve Q3 değerleri describe() fonksiyonu yardımı ile bulunmuştur.
- 19. satırda *sk* verisine ait IQR değeri bulunmuştur.
- 20. ve 21. satırlarda alt ve üst sınırlar bulunmuştur.
- 22. satıda bulunan al ve üst sınır değerlerinden yararlanılarak veri dataFrame yapısında filtreleme yapılmıştır.
- *Bir sonraki sayfada, Görsel 3'te veri* dataFrame'ine ait describe() fonksiyonu ekran çıktısı, Görsel 4'te ise *veri* dataFrame'ine ait info() fonksiyonu ekran çıktısı görülmektedir.

```
17  sk_q1=veri["sk"].describe()[4]
18  sk_q3=veri["sk"].describe()[6]
19  sk_IQR=sk_q3-sk_q1
20  alt_sinir=sk_q1-1.5*sk_IQR
21  ust_sinir=sk_q3+1.5*sk_IQR
22  veri=veri[(veri["sk"]>alt_sinir)&
    (veri["sk"]<ust_sinir)]
23  print(veri.describe())
24  print(veri.info())
```

Kod 8: Gürültülü veri probleminin çözülmesini sağlayan kod

Veri dataFrame’inin son haline ait bilgiler.

	yas	hkb	sk	aks	eykh	abe
count	298.000000	298.000000	298.000000	298.000000	298.000000	298.000000
mean	54.302013	131.620805	243.479866	0.147651	149.506711	0.328859
std	9.038790	17.675160	45.063008	0.355350	23.049244	0.470589
min	29.000000	94.000000	126.000000	0.000000	71.000000	0.000000
25%	47.250000	120.000000	211.000000	0.000000	133.000000	0.000000
50%	55.000000	130.000000	240.000000	0.000000	152.000000	0.000000
75%	60.750000	140.000000	273.750000	0.000000	166.000000	1.000000
max	77.000000	200.000000	360.000000	1.000000	202.000000	1.000000

	st_d	bds	kalp_rahatsizligi
count	298.000000	298.000000	298.000000
mean	1.025168	0.654362	0.459732
std	1.156392	0.927324	0.499214
min	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000
50%	0.650000	0.000000	0.000000
75%	1.600000	1.000000	1.000000
max	6.200000	3.000000	1.000000

Görsel 3: describe() fonksiyonuna ait dönen bilgilerin çıktısı-2

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 298 entries, 0 to 302
Data columns (total 14 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   yas                 298 non-null    float64
 1   cinsiyet            298 non-null    object
 2   gat                 298 non-null    object
 3   hkb                 298 non-null    float64
 4   sk                  298 non-null    float64
 5   aks                 298 non-null    int64
 6   ekg                 298 non-null    object
 7   eykh                298 non-null    float64
 8   abe                 298 non-null    int64
 9   st_d                298 non-null    float64
 10  st_e                298 non-null    object
 11  bds                 298 non-null    float64
 12  talasemi            298 non-null    object
 13  kalp_rahatsizligi   298 non-null    int64
dtypes: float64(6), int64(3), object(5)
memory usage: 29.1+ KB
None
```

Görsel 4: info() fonksiyonuna ait dönen bilgilerin çıktısı-2

Sayısal verilerin normalizasyon işleminin yapılması.

Kod9:

- Görsel 4'te görüldüğü gibi: 0,3,4,7,9,11 numaralı kolonlar sayısal verilerdir. (0. sütundaki yas verisini ileride başka bir işlemde kullanacağımız için normalize etmiyoruz.)
- 25. satırda ilgili sayısal verilere ait sütunlar tüm satırlarına ait veriler *scikit-learn* kütüphanesindeki *preprocessing* sınıfına ait *min_max_scale* fonksiyonu ile 0-1 değerleri arasına normalize edilir.
- Burada *X* parametresi normalize edilecek verilerin([n_örnek,n_özellik]) bilgisini alırken; *feature_range* parametresi verilerin normalize edilecek değer aralığını temsil eden 2 elemanlı bir demet yapısı vermemizi bekler.

```
25  veri.iloc[:,[3,4,7,9,11]]=minmax_scale(X=veri.iloc
    [:,[3,4,7,9,11]].values,feature_range=(0,1))
```

Kod 9: Sayısal verilerin normalizasyon işleminin yapılmasını sağlayan kod

Metin ile ifade edilmiş kategorik verilerin sayılar ile etiketlenmesi.

Kod10:

- Görsel 4'te görüldüğü gibi: bazı verilerin veri tipi (Dtype) object olarak görülmektedir. Çoğu veri madenciliği algoritması sadece sayısal veriler ile işlem yaptığı için burada kategorik verilerin her bir kategori ifade eden metnine karşılık bir sayı atanması amaçlanmaktadır.

- 26. satırda *scikit-learn* kütüphanesindeki LabelEncoder() sınıfından ait le isimli bir nesne türetilmiştir.
- 27,28,29,30 ve 31. satırlarda oluşturulan *le* nesnesinin *fit_transform()* fonksiyonu yardımı ilgili veriler sayısal değerler ile etiketlenmiştir.
- *Bir sonraki sayfada,* Görsel 5'te ise *veri* dataFrame'ine ait info() fonksiyonu ekran çıktısı görülmektedir.

```
26 le=LabelEncoder()
27 veri["cinsiyet"]=le.fit_transform(y=veri["cinsiyet"])
28 veri["gat"]=le.fit_transform(y=veri["gat"])
29 veri["ekg"]=le.fit_transform(y=veri["ekg"])
30 veri["st_e"]=le.fit_transform(y=veri["st_e"])
31 veri["talasemi"]=le.fit_transform(y=veri["talasemi"])
32 print(veri.info())
```

Kod 10: Gürültülü veri probleminin çözülmesini sağlayan kod

Veri dataFrame’inin son haline ait bilgiler.

- Görsel 5’te görüldüğü gibi tüm veri tipleri sayısal veri tipi haline gelmiştir.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 298 entries, 0 to 302
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   yas                298 non-null    float64
 1   cinsiyet           298 non-null    int32
 2   gat                298 non-null    int32
 3   hkb                298 non-null    float64
 4   sk                 298 non-null    float64
 5   aks                298 non-null    int64
 6   ekg                298 non-null    int32
 7   eykh               298 non-null    float64
 8   abe                298 non-null    int64
 9   st_d               298 non-null    float64
 10  st_e               298 non-null    int32
 11  bds                298 non-null    float64
 12  talasemi           298 non-null    int32
 13  kalp_rahatsizligi  298 non-null    int64
dtypes: float64(6), int32(5), int64(3)
memory usage: 29.1 KB
None
```

Görsel 5: info() fonksiyonuna ait dönen bilgilerin çıktısı-3

Sayısal verilerin kategorik verilere dönüştürülmesi.

Kod11:

- 33. satırda 40'tan küçük değerler 0, 60'tan büyük değerler 2 ve diğer değerler (40≤i ≤ 60) 1 değeri ile kategorize edilmiştir.
- Bu yöntem ile yas sayısal verisi kategorik bir veriye dönüştürülmüştür.
- 34. satırda ilk 5 satıra ait veri Görsel 6'daki gibi ekrana yazdırılmıştır.

```
33  veri["yas"] =[0 if i<40 else 2 if i>60 else 1 for
    i in veri["yas"]]
34  print(veri.head(5))
```

Kod 11: Gürültülü veri probleminin çözülmesini sağlayan kod

	yas	cinsiyet	gat	hkb	sk	aks	ekg	eykh	abe	st_d	\
0	2	0	3	0.481132	0.457265	1	2	0.603053	0	0.370968	
1	2	0	1	0.622642	0.683761	0	2	0.282443	1	0.241935	
2	2	0	1	0.245283	0.440171	0	2	0.442748	1	0.419355	
3	0	0	0	0.339623	0.529915	0	1	0.885496	0	0.564516	
4	1	1	2	0.339623	0.333333	0	2	0.770992	0	0.225806	

	st_e	bds	talasemi	kalp_rahatsizligi
0	0	0.000000	1	0
1	1	1.000000	0	1
2	1	0.666667	2	1
3	0	0.000000	0	0
4	2	0.000000	0	0

Görsel 6: Veriye ait ilk 5 kaydın ekran çıktısı

Ek

Kod12:

- 35. satırda, *scikit-learn* kütüphanesindeki model_selection() sınıfına ait train_test_split() fonksiyonu koda dahil edilmiştir.
- 36. satırda, veri setindeki etiket kısmına ait veriler *y* değişkenine aktarılmıştır.
- 37. satırda, veri setindeki öznitelik kısımları *X* değişkenine aktarılmıştır.
- 38. satırda *train_test_split()* fonksiyonu ile *X* ve *y* değişkenlerini ilk iki parametre olarak almış, test_size paramteresinde belirtilen oran kadar kısmını test verisi, geri kalanını da eğitim verisi olarak ayırıp geriye 4 dizi yapısı döndürmüştür.

```
35 from sklearn.model_selection import train_test_split
36 y = veri.iloc[:,13].values
37 X = veri.iloc[:,0:-1].values
38 X_train,X_test,y_train,y_test=train_test_split(X,y,
   test_size=0.2)
```

Kod 12: Verinin, test eğitim verisi olarak bölümlenmesi

Kaynakça

Kane, F. (2017). Hands-On Data Science and Python Machine Learning. Packt.

UCI Machine Learning Repository. (1997). UCI Machine Learning Repository: Statlog (Heart) [Veri seti]. http://archive.ics.uci.edu/ml/datasets/Statlog+%28Heart%29

Uğuz, S. (2019). Makine Öğrenmesi Teorik Yönleri Ve Python Uygulamaları İle Bir Yapay Zeka Ekolü. NOBEL AKADEMİK YAYINCILIK.

Görsel kaynakça

Görsel 1: info() fonksiyonuna ait dönen bilgilerin çıktısı-1

Görsel 2: describe() fonksiyonuna ait dönen bilgilerin çıktısı-1

Görsel 3: describe() fonksiyonuna ait dönen bilgilerin çıktısı-2

Görsel 4: info() fonksiyonuna ait dönen bilgilerin çıktısı-2

Görsel 5: info() fonksiyonuna ait dönen bilgilerin çıktısı-3

Görsel 6: Veriye ait ilk 5 kayıdın ekran çıktısı