Deniz KILINÇ
2.4K Followers

May 18, 2019 · 7 min read

Python ile Veri Tanımaya ve Temel İstatistiğe Dalış

Veri bilimi ve Python…

Python ile Veri Bilimine Dalış isimli yazımızda aşağıdaki konulara değinerek, örneklerle veri bilimine hızlı bir giriş yapmıştık.

- Giriş (Veri Bilimi, Tanıtım, İş Akışı)
- Python Programlama Dili (Geçmişi ve Yapısı)
- Python Değişken Türleri, Koşul Deyimleri, Döngüleri
- Python Veri Yapıları (Listeler, Sözlükler), Referans Türleri ve Nesne Klonlama
- Python Fonksiyon Tanımları ve Lambda Fonksiyonları
- Veri Bilimi ve Nesne Yönelimli Programlama
- NumPy ve Pandas Kütüphaneleri

. . .

Bu yazımızda, kendi oluşturacağımız basit bir veri kümesi üzerinde, Pandas dataframe (veri çerçevesi) tabanlı aşağıdaki konuları inceleyeceğiz.

1. Veri tanıma fonksiyonları
2. Temel istatistik fonksiyonları

Photo by Franki Chamaki on Unsplash

1. Veri Kümesi Oluşturma

12 satır ve 6 adet öznitelikten oluşan veri kümesi, belirli tarihlerde sınava giren kişilere ait demografik bilgileri, meslek bilgilerini ve sınavdan aldıkları puanları içermektedir. Veri kümesini oluşturmak için aşağıdaki kod bloğu kullanılabilir.

2. Pandas DataFrame Kullanarak Veri Tanıma ve Filtreleme

Veri kümesi üzerinde veri tanıma ve filtreleme amacıyla kullanılabilecek temel fonksiyonlar bu bölümde anlatılmıştır.

2.1. İlk 5 satırı görüntüleme

```
df.head()
```

	Meslek	Puan	Tarih	Yaş	ÇocukSayısı	İsim
0	işçi	89	11.11.2010	21	NaN	Ada
1	işçi	87	11.11.2010	24	NaN	Cem
2	memur	77	11.11.2010	25	NaN	Sibel
3	serbest	55	18.11.2011	44	NaN	Ahmet
4	serbest	70	18.11.2011	31	NaN	Mehmet

2.2. Son 5 satırı görüntüleme

```
df.tail()
```

7	sigortacı	79	None	33	0.0	Ayşe
8	işsiz	54	11.11.2010	42	NaN	Hüseyin
9	None	92	None	29	NaN	Necmi
10	None	61	18.11.2011	41	NaN	Nalan
11	memur	69	18.11.2011	43	NaN	Namık

2.3. Rasgele 5 satırı görüntüleme

```
df.sample(5)
```

2.4. Satır ve sütun (öznitelik) sayısını görüntüleme

```
df.shape

#Sonuç: (12, 6)
```

2.5. Özniteliklerin veri türleri, içerdikleri kayıt sayıları ve bellek kullanımı hakkında bilgi edinme

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12 entries, 0 to 11
Data columns (total 6 columns):
Meslek          8 non-null object
Puan            12 non-null int64
Tarih           8 non-null object
Yaş             12 non-null int64
ÇocukSayısı     3 non-null float64
İsim            12 non-null object
dtypes: float64(1), int64(2), object(3)
memory usage: 656.0+ bytes
```

Ekran çıktısından da görüleceği üzere örneğimizdeki data frame (yani df değişkeni) bellekte 656 byte yer kaplamaktadır. ÇocukSayısı özniteliğinin karşısında yazan 3 değeri, bu özniteliğin değerinin sadece 3 kişi için girildiğini ve diğer 9 kişi için eksik olduğunu ifade etmektedir.

2.6. Öznitelik seçerek kayıtları gösterme (projection)

```
df['Yaş']
```

2.7. Öznitelik seçerek ilk 5 kaydı gösterme (top)

```
df['İsim'][:5]
```

2.9. Kayıtlar üzerinde mantıksal koşul uygulayarak (and, or) filtreleme

Yaşı 30'dan büyük **ve (and)** Puanı 50'den büyük kayıtları getir.

```
df[(df['Yaş']>30) & (df['Puan']>50)]
```

Yukarıdaki kod parçasını çalıştırdığımızda aşağıdaki çıktıyı elde ederiz. 30 yaşından büyük olan ve sınavda 50 puandan yüksek alan 7 kayıt listeleniyor.

	Meslek	Puan	Tarih	Yaş	ÇocukSayısı	İsim
3	serbest	55	18.11.2011	44	NaN	Ahmet
4	serbest	70	18.11.2011	31	NaN	Mehmet
6	None	73	None	35	2.0	Veli
7	sigortacı	79	None	33	0.0	Ayşe
8	işsiz	54	11.11.2010	42	NaN	Hüseyin
10	None	61	18.11.2011	41	NaN	Nalan
11	memur	69	18.11.2011	43	NaN	Namık

Mesleği 'işçi' olan **veya (or)** Puanı 90'dan büyük olan kayıtları getir.

```
df[(df['Meslek']=='işçi') | (df['Puan']>90)]
```

	Meslek	Puan	Tarih	Yaş	ÇocukSayısı	İsim
0	işçi	89	11.11.2010	21	NaN	Ada
1	işçi	87	11.11.2010	24	NaN	Cem
9	None	92	None	29	NaN	Necmi

> ***Not:*** *Burada yaptığımız yöntem* ***"indis üzerinden filtreleme"*** *olarak da adlandırılabilir. İçe içe geçen dataframeler içten dışa zincirleme şekilde çalışır. İçten ilgili kayıtların indisleri döner.*

Diğer bir filtreleme yöntemi de dataframe üzerindeki ***query()*** fonksiyonu kullanılarak gerçekleştirilebilir.

```
#query() fonksiyonu ile filtreleme
df_filtered = df.query('Yaş>30 & Puan>50')
df_filtered
```

2.10. Sıralama (sorting)

```
df.sort_values('Puan', axis = 0, ascending = False)
```

sort_values() fonksiyonu kullanılarak gerçekleştirilen sıralama işleminin sonucu (ekran çıktısı) ve fonksiyonun parametre değerlerinin anlamları aşağıdaki gibidir.

- "axis = 0" → satırların sıralanacağı anlamına gelir.
- "ascending = false" → azalan şekilde sıralama yapılmasını sağlar.

	Meslek	Puan	Tarih	Yaş	ÇocukSayısı	İsim
9	None	92	None	29	NaN	Necmi
0	işçi	89	11.11.2010	21	NaN	Ada
1	işçi	87	11.11.2010	24	NaN	Cem
5	None	79	None	27	1.0	Ali
7	sigortacı	79	None	33	0.0	Ayşe
2	memur	77	11.11.2010	25	NaN	Sibel
6	None	73	None	35	2.0	Veli
4	serbest	70	18.11.2011	31	NaN	Mehmet
11	memur	69	18.11.2011	43	NaN	Namık
10	None	61	18.11.2011	41	NaN	Nalan
3	serbest	55	18.11.2011	44	NaN	Ahmet
8	işsiz	54	11.11.2010	42	NaN	Hüseyin

2.11. Koşul ve sıralamayı birlikte kullanma

Yaşı 30'dan ve Puanı 50'den büyük olan kayıtları, puan bilgisine göre azalan şekilde sırala.

```
df[(df['Yaş']>30) & (df['Puan']>50)].sort_values('Puan', axis=0, ascending=False)
```

2.12. Gruplama: Meslek bazında kayıt sayısı görüntüleme (groupby)

```
df.groupby('Meslek').size()
```

Yukarıdaki gruplama kodu çalıştırıldığında aşağıdaki gibi bir çıktı elde ederiz. Dikkat ederseniz, bir mesleğe sahip olan 8 kişinin kaydı meslek bazında gruplanarak getirilmesine rağmen (Örn; işçi olan 2 kişi) geri kalan 4 kişi listelenmiyor. Bunun nedeni bu 4 kişi için meslek girilmemiş olmasıdır. Diğer bir deyişle girilen değerlerin "None" olmasıdır.

```
işsiz          1
işçi           2
memur          2
serbest        2
sigortacı      1
dtype: int64
```

Mesleği girilmeyen ya da None olan kayıtları gruplamaya dahil etmek için aşağıdaki kod parçasını kullanabiliriz. Aşağıdaki kod parçasında olduğu gibi gruplama yapmadan hemen önce astype(str) ile tip dönüşümü yapabiliriz.

```
df['Meslek'] = df['Meslek'].astype(str)
df.groupby('Meslek').size()
```

```
Meslek
None           4
işsiz          1
işçi           2
memur          2
serbest        2
sigortacı      1
dtype: int64
```

2.13. Gruplama: Meslek bazında alınan puanların ortalamasını görüntüleme (groupby)

lambda ve apply() fonksiyonunu birlikte kullanıyoruz.

```
df.groupby('Meslek')['Puan'].apply(lambda x: np.mean(x))
```

```
Meslek
None          76.25
işsiz         54.00
işçi          88.00
memur         73.00
serbest       62.50
sigortacı     79.00
Name: Puan, dtype: float64
```

3. Pandas DataFrame Kullanarak Temel İstatistik Bilgisi Edinme

- ***Dağılım ölçüleri:*** standart sapma, varyans, kovaryans, korelasyon

3.1. Mod (Tepe Değer)

Sayısal bir veri serisi içindeki en çok tekrar eden sayı, o serinin modu olarak adlandırılır. Bazı serilerde maksimum tekrar sayısına sahip birden fazla seri elemanı olabilir. Bu durumda, serinin birden fazla modu olur. Veri kümesindeki "Puan" özniteliğinin (*bu öznitelik değerleri aynı zamanda sayısal değerlere sahip bir seridir*) mod değeri aşağıdaki kod parçası kullanılarak elde edilebilir.

```
#Puan özniteliğinin modu
df['Puan'].mode()

#Sonuç: 79
```

3.2. Medyan (Ortanca)

Sayısal bir veri serisi sıralandığında ortada kalan sayıdır. Veri kümesindeki "Puan" özniteliğinin medyanı aşağıdaki kod parçası kullanılarak elde edilebilir.

```
#Puan özniteliğinin medyanı
df['Puan'].median()

#Sonuç: 75.0
```

3.3. Aritmetik Ortalama

Bir serideki sayıların toplamının serinin eleman adedine bölünmesi ile elde edilen değerdir ve aşağıdaki gibi hesaplanır.

```
#Puan özniteliğinin ortalaması
df['Puan'].mean()

#Sonuç: 73.75
```

Sayısal tüm özniteliklerin ortalaması el edilmek istenirse aşağıdaki kod parçasından yararlanılabilir.

```
#Sayısal tüm özniteliklerin ortalamaları
df.mean(axis = 0, skipna = True)
```

```
Puan           73.750000
Yaş            32.916667
ÇocukSayısı     1.000000
dtype: float64
```

Not: *Benzer fonksiyon kullanımı yaklaşımıyla tüm öznitelikler için mod ve medyan hesaplaması yapılabilir.*

Soru: *df.mode(axis = 0) komutu kullandığımızda neden aşağıdaki çıktıyı elde ediyoruz, değerleri yorumlayarak çözmeye çalışalım?*

0	None	79.0	11.11.2010	21	0.0	Ada
1	NaN	NaN	18.11.2011	24	1.0	Ahmet
2	NaN	NaN	NaN	25	2.0	Ali
3	NaN	NaN	NaN	27	NaN	Ayşe
4	NaN	NaN	NaN	29	NaN	Cem
5	NaN	NaN	NaN	31	NaN	Hüseyin
6	NaN	NaN	NaN	33	NaN	Mehmet
7	NaN	NaN	NaN	35	NaN	Nalan
8	NaN	NaN	NaN	41	NaN	Namık
9	NaN	NaN	NaN	42	NaN	Necmi
10	NaN	NaN	NaN	43	NaN	Sibel
11	NaN	NaN	NaN	44	NaN	Veli

3.4. Standart Sapma

Bir serideki sayıların, serinin aritmetik ortalamasından farklarının karelerinin toplamının, serinin eleman sayısının bir eksiğine bölümünün kareköküdür. Standart sapma ile verilerin ne kadarının ortalamaya yakın olduğunu buluruz. Eğer standart sapma küçükse veriler ortalamaya yakın yerlerde dağılmışlardır. Tam tersi durumda, standart sapma büyükse, veriler ortalamadan uzak yerlerde dağılmışlardır. Bütün değerler aynı olursa standart sapma sıfır olur. Formülü aşağıdaki gibidir.

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2}$$

Standart sapma değeri hesaplama formülü

Veri kümesindeki “Puan” özniteliğinin standart sapması aşağıdaki kod parçası kullanılarak hesaplanabilir.

```
#Puan özniteliğinin standart sapması
df['Puan'].std()

#Sonuç: 12.606816338069583
```

3.5. Varyans

Standart sapmanın karesidir.

$$Covari(x,y) = \frac{\sum_{i=1}^{n}(x_i - x')\,(y_i - y')}{n-1}$$

X ve y değişkenleri arasındaki kovaryans değeri hesaplama formülü

Veri kümesinde yer alan tüm öznitelikler arasındaki kovaryans değerlerini içeren matris aşağıdaki kod parçasıyla hesaplanabilir.

```
#Kovaryans matrisi hesapla
df.cov()
```

	Puan	Yaş	ÇocukSayısı
Puan	158.931818	-87.295455	-3.0
Yaş	-87.295455	64.992424	1.0
ÇocukSayısı	-3.000000	1.000000	1.0

> ***Not:*** *Kovaryans matrisi bize; Puan ile Yaş arasında negatif, Puan ile ÇocukSayısı arasında negatif, Yaş ile ÇocukSayısı arasında pozitif bir ilişkinin var olduğunu göstermektedir. Ancak ilişkinin şiddeti hakkında bir yorum yapamayız.*

3.7. Korelasyon

İki veya daha fazla bağımsız değişken (öznitelik)arasındaki ilişkinin varlığı, bu ilişkinin yönü ve şiddeti **korelasyon analizi** ve sonucunda elde edilen **korelasyon katsayısı** ile belirlenir. Formülü aşağıdaki gibidir.

$$Corr(x,y) = \frac{\sum_{i=1}^{n}(x_i - x')\,(y_i - y')}{\sqrt{\sum_{i=1}^{n}(x_i - x')^2 \sum_{i=1}^{n}(y_i - y')^2}}$$

X ve y değişkenleri arasındaki korelasyon katsayı değeri hesaplama formülü

Korelasyon katsayısı “r”, “-1” ve “1” arasında değişen değerler alır. Değer sonuçları ile ilgili yorumlar aşağıdaki gibidir.

- r = -1 → negatif yönlü mükemmel bir ilişki
- r = 1 → pozitif yönlü mükemmel bir ilişki
- r = 0 → ilişki yok

Veri kümesinde yer alan tüm öznitelikler arasındaki korelasyon katsayısı değerlerini içeren matrisi hesaplamak için aşağıdaki kod parçası kullanılabilir.

```
#Korelasyon matrisi hesapla
df.corr()
```

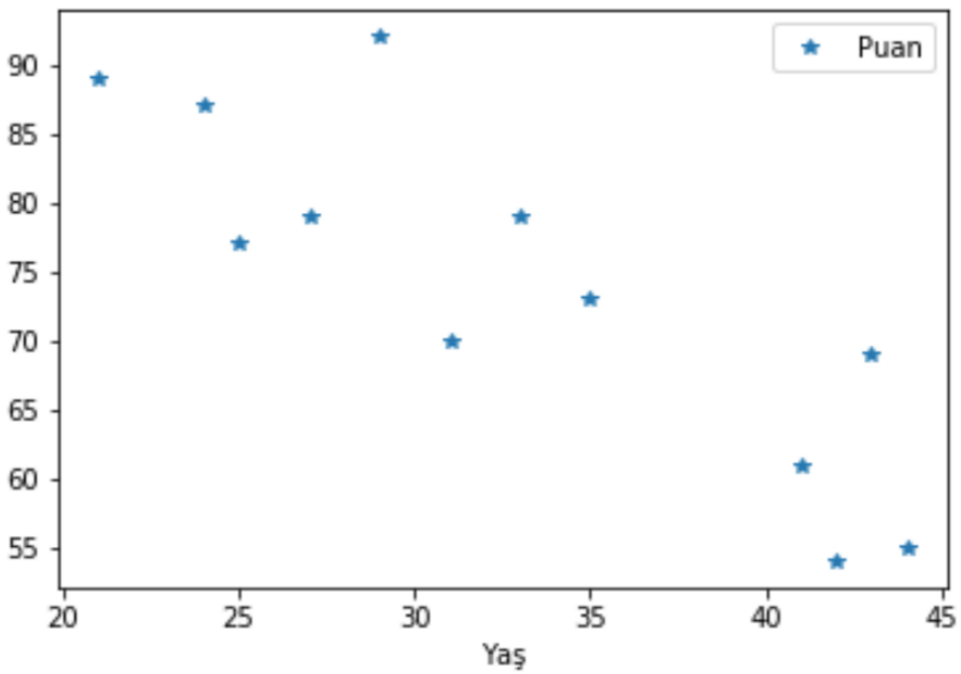
	Puan	Yaş	ÇocukSayısı
Puan	1.000000	-0.858924	-0.866025
Yaş	-0.858924	1.000000	0.240192
ÇocukSayısı	-0.866025	0.240192	1.000000

Not: *Korelasyon matrisine göre Puan ile Yaş arasında negatif (ters yönlü) güçlü bir ilişki, Puan ile ÇocukSayısı arasında yine negatif (ters yönlü) güçlü bir ilişki, Yaş ile ÇocukSayısı arasında pozitif (aynı yönlü) zayıf bir ilişki vardır.*

** Yaş arttıkça sınavdan alınan puan düşmektedir.*
** Çocuk sayısı arttıkça puan düşmektedir.*

Yaş ile puan arasındaki ters yönlü ilişkiyi basitçe çizdirerek, mevcut durumu görmek istersek aşağıdaki gibi kod parçasını kullanabiliriz.

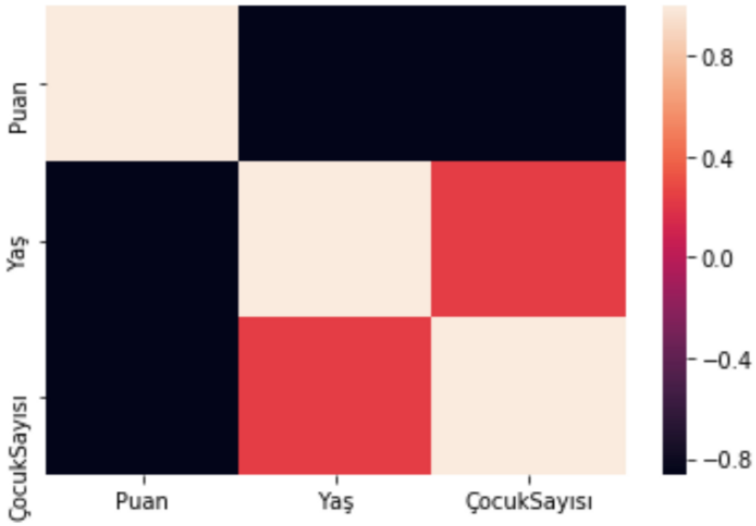
```
df.plot(x='Yaş', y='Puan', style='*')
```



Değişkenler (öznitelikler) arasındaki korelasyon katsayısı değerlerini gösterirken en çok kullanılan yöntemlerden birisi de ***ısı haritası*** ile görselleştirme yapmaktır. ***seaborn*** paketinde yer alan ***heatmap()*** fonksiyonu aşağıdaki gibi kullanılarak değişkenler arasındaki korelasyon değerleri ısı haritası üzerinde görülebilir.

```
1  #Korelasyon Gösterim
2  import seaborn as sns
```

python_sample_correlation.py hosted with ♡ by GitHub view raw



Isı haritası grafiği, negatif yönlü ilişkinin şiddeti arttıkça rengin koyulaştığını, pozitif yönlü ilişkinin şiddeti arttıkça rengin krem rengine doğru açıldığını ifade etmektedir.

Bu yazımızı noktalama zamanı geldi :)

Bana Linkedin üzerinden veya https://twitter.com/denizkilinc adresinden ulaşabilirsiniz. Bir sonraki yazımızda yine aynı veri kümesini kullanarak Python ile Veri Ön İşlemeye Dalış yapacağız.

Sevgiyle kalın, görüşmek üzere….